// WHITE PAPER · PUBLISHED 16 SEPTEMBER 2026

Capitalize or Expense: Your Jira Board Is Now Audit Evidence

A field guide to ASC 350-40 after ASU 2025-06, and to the artifacts that decide it

Part 3 of 10 in *The Money Language Series*. Each piece stands on its own. The others: Part 1 · Part 2 · Part 4 · Part 5 · Part 6 · Part 7 · Part 8 · Part 9 · Part 10.

Jason Newell · MAX Research Collective · Report WPR-BIZ-004 · Published 16 September 2026
Licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0), https://creativecommons.org/licenses/by/4.0/
Original: https://www.maxresearchcollective.com/articles/capitalize-or-expense.html
Cite as: Newell, J. (2026). Capitalize or Expense: Your Jira Board Is Now Audit Evidence. MAX Research Collective, WPR-BIZ-004. https://www.maxresearchcollective.com/articles/capitalize-or-expense.html

There is a conversation that happens in every software company around month eleven of the fiscal year. Finance asks engineering what percentage of the team's time was capitalizable. Engineering, who has never been asked this before, produces a number. Finance uses it. Nobody is comfortable.

I have been on both ends of that call. It goes badly because the rules were written in 1998, for waterfall, and everyone has been translating ever since. That translation just got easier, and stricter.

What changed, and when it bites

FASB issued ASU 2025-06 on 18 September 2025. It modernizes ASC 350-40, the rules covering software you build to run your own business. It takes effect for annual reporting periods beginning after 15 December 2027, with early adoption permitted, which for calendar-year companies means fiscal 2028 (two full planning cycles from now, and one budget cycle from the point it stops being theoretical).

The headline change: it removes every reference to project stages.

	OLD GUIDANCE, THE SOP 98-1 LINEAGE	ASU 2025-06
Model	Three stages: preliminary project, application development, post-implementation	No stages at all
Capitalization trigger	You entered the application development stage	Management authorized and committed funding, and completion is probable
Brake	Implicit	Explicit. Significant development uncertainty blocks capitalization
Website costs	Separate guidance under ASC 350-50	Folded into 350-40
Expected effect		Less capitalization, more expensing

That last row is the one to carry into your next planning cycle. The new significant development uncertainty concept deliberately mirrors technological feasibility from ASC 985-20, the rules for software you sell. FASB expects the two to produce more consistent outcomes, which in practice means internal-use software gets treated more like product software: harder to capitalize early.

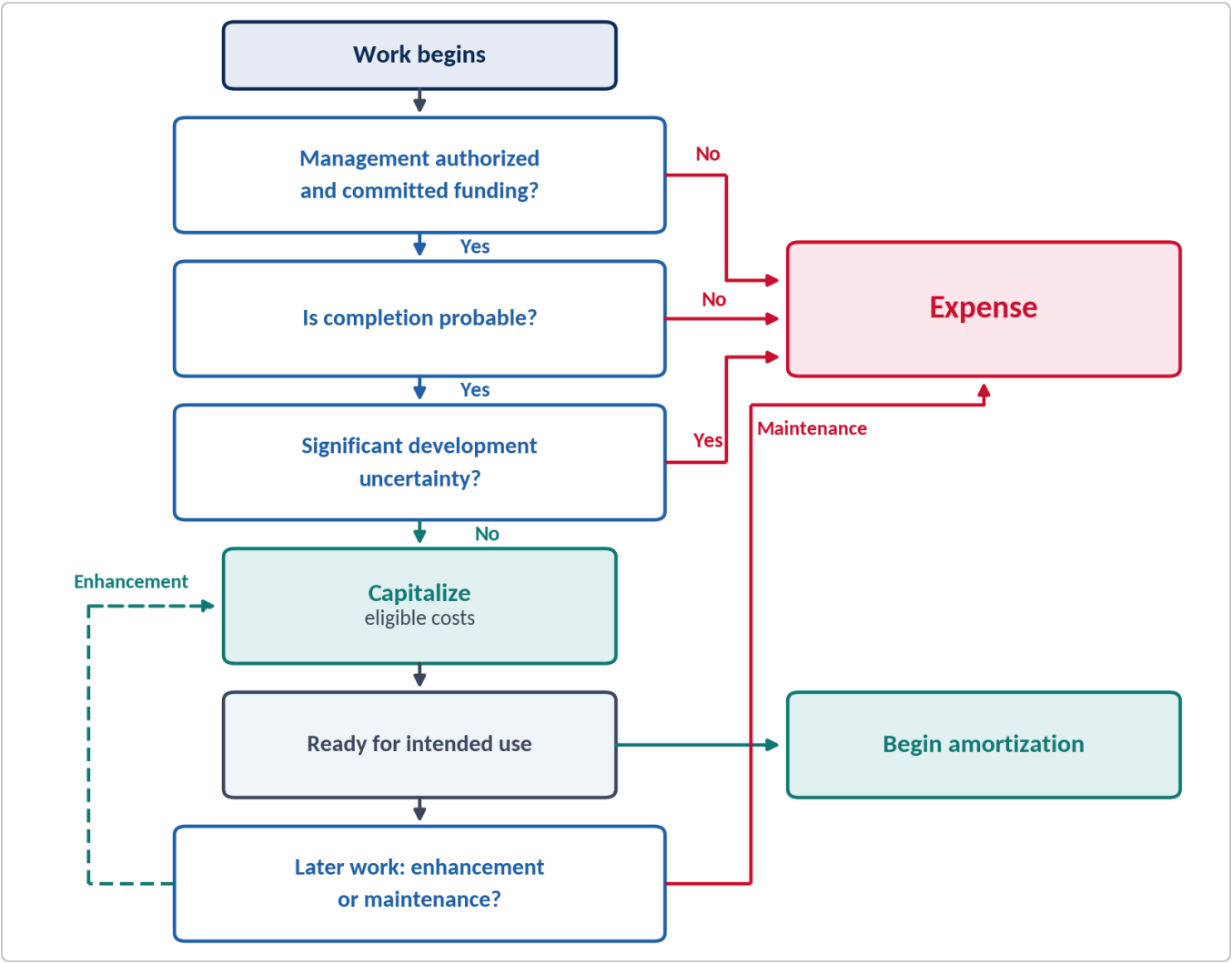


Figure 1. Three gates and one sink. Work reaches the asset only by clearing all three, and it keeps clearing them after launch.

What significant development uncertainty means for agile

This is the part that matters to you, because you run agile teams and the standard was written to accommodate that.

Uncertainty is significant while the project is still working out whether the thing can be built, or what it fundamentally is. That is a narrower window than it sounds. A spike evaluating three approaches sits in that territory. So does a discovery sprint, and so does a proof of concept where failure is a real outcome. Research into whether a model can reach the accuracy the product needs belongs there too, and all of it gets expensed.

Once the approach is settled, funding is committed, and the remaining question is execution rather than feasibility, the uncertainty stops being significant, and capitalization can begin from that date forward rather than from the day somebody first drew the architecture on a whiteboard and started calling it a project.

The practical translation is worth stating plainly. Under the old stage model, teams argued about which stage a sprint belonged to, which was a fiction nobody believed. Under the new model the argument is about whether you still might fail, and that is a question engineers can answer honestly. The honesty is the improvement.

Some things hold either way. Training costs and most data conversion costs stay outside capitalization permanently. General and administrative overhead stays outside. So do post-launch maintenance and bug fixes. Enhancements that add genuinely new functionality can come back in.

The costs in scope

COST	USUALLY CAPITALIZABLE?	NOTES
Engineer salaries and benefits, directly on the project	Yes	Time must be traceable to the project
Contractor and agency development fees	Yes	Same test as employees
Design and architecture work after approval	Yes	Pre-approval discovery is expensed
Testing and QA on new functionality	Generally yes	Post-launch regression testing sits outside
Project management directly on the build	Often	Depends on how directly attributable
Training users	No	Explicitly excluded
Data conversion	Mostly no	Narrow exceptions
General overhead, recruiting, facilities	No	
Maintenance, bug fixes, patching	No	
Enhancements adding new functionality	Yes	The line against maintenance is judgment
Cloud subscription fees	No	Expensed as consumed, and see below
Cloud implementation work	Yes	Under ASU 2018-15

The cloud wrinkle, ASU 2018-15

If you are implementing a hosted platform rather than building software, different guidance applies, and it is genuinely helpful.

ASU 2018-15 says the subscription fees for a cloud arrangement that is a service contract get expensed as the service is delivered. The implementation costs, meaning the configuration, integration and build work around it, follow the same internal-use software logic and can be capitalized. Those capitalized costs then amortize over the hosting term, including renewal periods you are reasonably certain to exercise.

This is the rule that makes a large ERP, CRM or data platform implementation financially survivable. Without it a $4M implementation hits a single year's P&L in full, against the five or seven years of contract life the thing actually serves. With it, the implementation work spreads across that life alongside the subscription it enables.

One detail people miss: the capitalized implementation asset is presented in the same place as the prepaid service fees, and the amortization appears on the same line as the hosting expense. It never shows up as a software intangible. Your CFO will care about this and your engineering team will not.

Three rulebooks, one organization

Most SaaS companies live in more than one of these at once, and the boundary runs straight through a single engineering org.

RULEBOOK	APPLIES TO	TRIGGER
ASC 350-40	Software for internal use and operations	Authorization plus probable completion, with no significant uncertainty
ASC 985-20	Software you sell, lease or market	Technological feasibility established
ASU 2018-15	Implementing someone else's hosted service	Same as 350-40, amortized over the hosting term

Here is where it gets uncomfortable. A hosted SaaS product delivered over the web is frequently accounted for under 350-40 rather than 985-20, because you are operating the software rather than delivering it to the customer. The same team, in the same sprint, can be doing work that falls under different rules depending on whether it serves customers or serves internal operations. Nobody enjoys this; it is real, and getting the split documented early costs far less than getting it audited late.

Your Jira board is now audit evidence

This is the operational consequence most leaders underestimate.

The new trigger is management authorization and committed funding, plus a probability judgment. Both are demonstrated by artifacts your organization already produces, or fails to produce:

- The funding approval document, with a date and a named approver
- The business case stating expected scope and completion
- Sprint and epic structure that distinguishes discovery from build
- Time allocation that traces to a project rather than to a team
- The decision record showing when the approach was settled

Poor Jira hygiene slows engineering down and removes the evidence your auditors need. When evidence is thin the default answer is to expense everything. One company I watched lost seven figures of capitalizable spend — most of a year's engineering capitalization — because nobody could demonstrate when the uncertainty ended.

Do this before the project starts. Write the capitalization policy with your CFO, name the authorization artifact, and agree the tagging convention in the tracker. It takes one meeting; retroactive reconstruction takes a quarter and fails anyway.

Impairment, the part nobody plans for

Capitalized software is an asset, and assets can be impaired. Build something, capitalize $6M of it, then kill it or materially reduce its expected use. That balance comes off in a lump, through the income statement, at the worst possible moment.

That is the hidden risk of aggressive capitalization, and it outlasts the amortization tail everyone remembers to plan for. A cliff sits in your balance sheet waiting for a strategy change. If you are the CTO who capitalized a platform rebuild and then recommends pivoting off it eighteen months later, know the size of that write-off before you make the recommendation, because somebody is going to ask.

Adoption is a decision

ASU 2025-06 gives you three transition approaches: prospective, modified, or retrospective. That is a real choice with real consequences. It belongs to finance, and you should be in the conversation, because the change in capitalization rate lands on your budget.

Two questions are worth raising early.

1. Do we early adopt? If the new rules reduce capitalization and you are heading into a year where lower EBITDA is survivable, taking the hit early beats taking it in a year you need to look good.
2. What does this do to our covenants? If debt is priced off EBITDA and your capitalization rate drops, reported EBITDA drops with it while the business underneath stays exactly the same. Lenders would rather hear that eighteen months early than read it in a compliance certificate.

What I actually do

1. One meeting with the CFO before any project over a threshold we agree on. The policy comes first and the build comes second.
2. Tag the tracker from day one, with discovery epics separate from build epics. The tags go on at the start.

Newell, J. (2026). MAX Research Collective, WPR-BIZ-004. CC BY 4.0

3. Keep an approvals file: date, amount, approver, scope. It is boring, and it is decisive in an audit.
4. Argue for capitalization only where I believe it. The credibility cost of being the CTO who inflates the asset is permanent, and the amortization lands on my own future budget anyway.
5. Track the impairment exposure on anything capitalized, so a pivot recommendation arrives with its own price tag attached.
6. Remember that tax is separate. Expensing for book still leaves the Section 174A deduction available, which Part 10 covers.

Glossary

TERM	DEFINITION
ASC 350-40	US GAAP rules for software built for internal use
ASC 985-20	Rules for software you sell or license externally
ASU 2018-15	Rules for the costs of implementing a cloud service contract
ASU 2025-06	The September 2025 update removing the project-stage model
Impairment	Writing down an asset whose value has dropped
Probable	More likely than not, in accounting usage. A real threshold rather than a vibe
Prospective, modified, retrospective	The three ways to transition to a new standard
Ready for intended use	The point amortization starts
Significant development uncertainty	Still unclear whether or what you can build, which blocks capitalization
Technological feasibility	The analogous concept for software you sell

References

- FASB, ASU 2025-06: Intangibles, Goodwill and Other, Internal-Use Software (Subtopic 350-40): Targeted Improvements to the Accounting for Internal-Use Software, issued 18 September 2025
- AICPA & CIMA, CPEA Alert, September 2025
- Forvis Mazars, "FASB's Improvements to Accounting for Internal-Use Software"

- Baker Tilly, "Modernizing software accounting: What ASU 2025-06 means for businesses"
- Grant Thornton, "Improvements to accounting for internal-use software"
- PKF O'Connor Davies, "FASB's Internal-Use Software Guidance: What Companies Need to Know"
- FASB, ASU 2018-15, and Deloitte Heads Up, "FASB Amends Guidance on Cloud Computing Arrangements"
- RSM US, "Customer's accounting for cloud computing implementation costs"

Best next read

Part 4, Depreciation and Useful Life: The Chip Outlives Its Economics. You have decided what becomes an asset. Part 4 covers how long it lives, why that estimate is one of the largest earnings levers in technology, and what happens when you get it wrong.

The full series

PART	TITLE
1	CapEx, OpEx, and EBITDA: The Same Dollar, Two Different Stories
2	Reading the Financials: The Primer for Everyone Who Never Had To Care
3	Capitalize or Expense: Your Jira Board Is Now Audit Evidence, which is this one
4	Depreciation and Useful Life: The Chip Outlives Its Economics
5	Placement Economics: The Right Answer, Applied Everywhere
6	Lift-and-Shift or Re-Architect: The Two That Modernized Later
7	Vendor Economics: The Email in November
8	AI Spend: Three Budgets Pretending to Be One
9	AI and Headcount: Somebody on the Board Did the Arithmetic
10	Outsourcing After Section 174A: Eleven Months to Relearn It

This series is an introduction for technology leaders. It is accounting and tax context rather than accounting or tax advice. Treatment is fact-specific. Model it with your CFO and auditor before you commit.

CAREER & LEADERSHIP WPR-BIZ-004

// ABOUT THE AUTHOR

Jason Newell leads organizations and people in technology, and is a researcher, deep technologist and builder — from a CRAY to agentic AI. He founded MAX Research Collective to build AI for regulated industries that runs where its owner has control, with healthcare as its strongest focus. About Jason → Connect on LinkedIn →

Newell, J. (2026). MAX Research Collective, WPR-BIZ-004. CC BY 4.0