FASB ASU 2025-06: ASC 350-40 Software Capitalization Guide

Published May 4, 2026 42 min read

Executive Summary

FASB ASU 2025-06 introduces **targeted improvements** to ASC 350-40 **Internal-Use Software**, fundamentally altering how companies determine when to capitalize software development costs. In particular, ASU 2025-06 **eliminates the legacy project-stage model** in favor of a single **"recognition threshold"** based on management approval and the probability of project completion (Source: rsmus.com) (Source: www.bakertilly.com). Under the new guidance, companies start capitalizing costs *only when* (1) management has authorized and committed funding to the project and (2) it is **probable** the project will be completed and used as intended (Source: rsmus.com) (Source: www.bakertilly.com). Critically, if **any significant development uncertainty** – such as unresolved novel features or shifting requirements – exists, capitalization must be deferred until that uncertainty is resolved (Source: rsmus.com) (Source: www.eidebailly.com). In effect, ASU 2025-06 aligns GAAP with modern, iterative development (e.g. Agile) by removing rigid gates, while retaining the core economic recognition criteria.

For SaaS companies, which traditionally account for most product development costs under the **internal-use software rules**, this update can materially affect financial reporting. Many SaaS firms historically capitalized large internal R&D spend, but had to carefully navigate ASC 350-40's stage-based rules. Under the new model, companies must exercise judgment early and document when the "probable-to-complete" threshold is met. If substantive uncertainties are quickly resolved, capitalization can begin sooner than under the old model; conversely, ambiguous projects are entirely expensed until ambiguities clear (Source: www.eidebailly.com) (Source: www.armanino.com). For example, an enterprise implementing a third-party ERP may begin capitalizing as soon as a contract is signed and no novel functionality remains (Source: www.eidebailly.com), whereas a mobile app project with undefined requirements remains expensed until after design is finalized (Source: www.eidebailly.com).

This report provides a **comprehensive analysis** of ASU 2025-06, its background and rationale, and its detailed provisions. We examine the **scope and transition** of the new rules, compare them to the legacy guidance and to rules for externally marketed software, and analyze the impact on SaaS developers' financials. The report also discusses how to implement these requirements in practice — with a focus on accounting systems like **NetSuite** — and presents case studies, industry surveys, and expert commentary. Notably, leading accounting firms (e.g. Deloitte, RSM, Baker Tilly, Crowe) all stress that while ASU 2025-06 does not change what costs *qualify* as capitalizable, it removes the sequential phases and relies on

judgement about project viability (Source: rsmus.com) (Source: www.bakertilly.com). Early evidence from an Armanino survey shows that most SaaS companies already capitalize development costs (69% of top SaaS firms did so in 2024 (Source: www.armanino.com) and that the new rules simply codify a shift already occurring in practice.

Finally, this report examines **NetSuite-specific guidance** for SaaS developers. It outlines how a SaaS provider might configure NetSuite's project accounting and amortization features to track internal software projects, allocate costs to capital vs expenses, and comply with the updated FASB guidance. Practical examples (e.g. a multi-year platform upgrade, or AI-driven feature development illustrate the judgment calls required. We also consider allied issues such as tax treatment (notably the impact of IRC §174 on R&D capitalization (Source: www.armanino.com) and regulatory expectations. Throughout, the analysis is grounded in extensive citations of authoritative sources: FASB releases, accounting firm publications, SEC filings and commentaries, and industry surveys. The goal is to equip finance and technology leaders with a deep understanding of ASU 2025-06, the nuances of ASC 350-40, and best practices for implementation in NetSuite and beyond.

Introduction and Background

Software Development in the Modern Enterprise

In today's technology-driven economy, **internally developed software** has become a critical asset for businesses, especially **SaaS (Software-as-a-Service)** companies. Internally developed applications – whether to manage operations, analytics, or cloud platforms – can represent substantial investment, often running into millions of dollars. Accordingly, U.S. GAAP provides rules for when to **capitalize** these development costs (treat them as long-lived assets on the balance sheet) versus **expensing** them immediately. Historically, U.S. GAAP drew a sharp line between **internal-use software** (covered by ASC 350-40) and **software to be sold/marketed externally** (ASC 985-20). ASC 350-40 was first established in 2009 (via FASB ASU 2009-13) to standardize treatment of internal software costs. It required capitalization during the "application development" phase but expensed costs in the preliminary planning and post-implementation phases (Source: www.cohenco.com). This staged approach was grounded in a traditional "waterfall" development model: once an entity finished planning and committed funds, development costs could be capitalized (Source: www.cohenco.com) (Source: www.eidebailly.com).

Over time, however, software development practices evolved. Many companies now use **iterative or Agile methodologies**, which do not follow distinct sequential phases. As numerous practitioners have noted, under the old guidance it could be difficult or subjective to pinpoint the exact transition between planning and development when the work is incremental (Source: rsmus.com) (Source: www.bakertilly.com). The FASB itself acknowledged these concerns in field outreach: many preparers felt ASC 350-40's stage-based model was *outdated* given modern methods (Source: rsmus.com) (Source: www.bakertilly.com). In parallel, the growth of cloud computing and SaaS shifted when and how software is delivered. For SaaS companies, product development costs often ended up being accounted under 350-40 (internal-use), whereas on-premises license sales by traditional software vendors fell under 985-20. This distinction could be counterintuitive economically – two functionally similar code bases could be treated differently simply because of the delivery model (Source: schneiderdowns.com)(Source: www.crowe.com). For example, Schneider Downs observes that a company that once sold licenses of its software and later shifted to a SaaS subscription model will now account for much of its R&D as internal-use expense (Source: schneiderdowns.com) (Source: www.crowe.com).

As internal-use accounting drafted behind evolving practices, inconsistencies grew. Multiple analyses and surveys have documented the effect. An Armanino survey of the top 100 public SaaS firms found that **69% of SaaS companies capitalize development costs** in practice (up from 62% in 2017) (Source: www.armanino.com). Meanwhile, Wall Street Prep highlights that companies like Alphabet Inc. (Google) *do* capitalize their internal software costs after preliminary planning, whereas others do not, leading to wide financial statement differences (Source: www.wallstreetprep.com) (Source: www.wallstreetprep.com). Ultimately, FASB moved to modernize the guidance: in September 2025 it issued ASU 2025-06, officially titled *"Targeted Improvements to the Accounting for Internal-Use Software"* (Source: rsmus.com) (Source: dart.deloitte.com). This update, described as a *"targeted improvement"*, replaces the rigid phase gates with a principles-based threshold designed to work for any development methodology.

Legacy GAAP Framework (ASC 350-40 vs ASC 985-20 vs ASC 350-50)

ASC 350-40 (Internal-Use Software). Under the prior codification, ASC 350-40 required entities to treat internal-use software development in phases: (*1*) Preliminary Project, (*2*) Application Development, and (*3*) Post-Implementation/Operation (Source: www.cohenco.com) (Source: www.wallstreetprep.com). Only costs incurred in the Application Development phase were capitalized; costs in Preliminary or Post-Implementation were expensed (Source: www.cohenco.com) (Source: www.wallstreetprep.com). To enter the Application Development phase, two preconditions had to be met: the preliminary project stage must be complete **and** management must have authorized and committed funding (meaning the project was likely to be finished) (Source: www.cohenco.com) (Source: www.wallstreetprep.com). Preliminary activities such as conceptual planning, vendor

evaluation, or technology assessments were expensed as incurred (Source: www.cohenco.com) (Source: www.wallstreetprep.com). Once capitalizing ceased (typically when the software was substantially complete and ready for use), any further maintenance or routine upgrades were expensed (Source: www.cohenco.com) (Source: www.wallstreetprep.com).

Importantly, ASC 350-40's rules apply only when *both* of its criteria are met (managementcommittment **and** probable completion). Costs capitalized are those "to develop or obtain internal-use software" as listed in ASC 350-40-25-1 through 25-5, including *external direct costs* (e.g. third-party developers) and *internal payroll* for personnel devoted to the project (Source: www.scribd.com) (Source: www.wallstreetprep.com). Overhead or "G&A" costs (like general management support, training, or maintenance) are explicitly excluded from capitalization (Source: www.scribd.com) (Source: www.cohenco.com). After ASU 2025-06, while the **timing** of capitalization is changed, the types of costs eligible for capitalization largely remain the same (the new ASU does not amend the scope of capitalizable costs) (Source: www.eidebailly.com) (Source: www.cohenco.com).

ASC 350-50 (Website Development). Prior to ASU 2025-06, website and certain application development costs fell under ASC 350-50. This subtopic had similar rules: preliminary stage costs were expensed, and application development costs could be capitalized if used internally (or if for advertising, etc, under narrow conditions) (Source: www.wallstreetprep.com). The new ASU explicitly **supersedes ASC 350-50**, folding website development into the broader 350-40 framework (Source: rsmus.com) (Source: www.bakertilly.com). In other words, the same "probable-to-complete" threshold now applies to websites and internally-used applications alike, eliminating a separate set of rules.

ASC 985-20 (Software to be Sold, Leased, or Marketed). This subtopic governs external-facing software (e.g. software products or licensed applications). Under ASC 985-20, companies capitalize costs only after *technological feasibility* is reached (often defined by completion of a detailed design or first working model) and before product release (Source: www.wallstreetprep.com) (Source: www.armanino.com). IAS 985-20 uses terms like *technological feasibility* and defers costs until release. In contrast, internal-use rulings did *not* use the technological feasibility concept: instead they focused on project stages and completion probability. Notably, ASU 2025-06 **does not** change ASC 985-20. The two frameworks still diverge: a functionally identical software development could be accounted differently depending on whether the company expects to sell/licence it or just use it internally (Source: schneiderdowns.com) (Source: www.crowe.com). (A key example: if a SaaS subscriber cannot take a full license copy of the software, the arrangement is treated as a service, meaning the development costs remain internal use under ASC 350-40 (Source: www.crowe.com), even though the outcome is a revenue-generating cloud product.)

SaaS Context. In practice, **SaaS providers** almost always fall under ASC 350-40 for their development costs. Schneider Downs emphasizes: "Software-as-a-Service (SaaS) entities typically fall within the guidance in ASC 350-40. If the customer does not have the right to take possession, the contract is a service contract, not a license" (Source: schneiderdowns.com). Thus, a SaaS company's platform is treated as an internal asset, even though it is the core revenue engine. Because of this, SaaS companies have tended to capitalize significant R&D (e.g., new features or platform upgrades) under ASC 350-40, amortizing them over useful lives (often 2–5 years) (Source: www.wallstreetprep.com) (Source: www.cohenco.com). By contrast, traditional "boxed" software firms capitalize later under ASC 985-20 (after technical feasibility) and cease before release. This discrepancy has motivated some to advocate for a "single model" that treats all software development alike. Indeed, prominent practitioners like former Salesforce CFO Andrew Hyde argue that "software development costs should be capitalized consistently, whether internal or external," noting that the two methods can yield virtually the same outcome through managerial judgment, and thus the dual approach can confuse accounting and development teams (Source: www.armanino.com).

Prior Guidance Limitations

Both preparers and auditors have cited **operability issues** in ASC 350-40. Under the old model, capitalization *triggered automatically* upon entering Application Development, but in many Agile scenarios there is no clear-cut entry point. Companies often found they had to interpret what constituted "final selection" of design or when the preliminary stage truly ended (Source: www.rit.edu) (Source: www.wallstreetprep.com). In practice, this led to diversity: some firms began capitalizing when a detailed project plan was approved (or vendor contract signed), while others waited until coding was substantially underway. As Baker Tilly notes, stakeholders expressed that the legacy guidance "is outdated and lacks relevance given the evolution of software development — in particular, the transition from a prescriptive, sequential method… to an incremental and iterative method" (Source: www.bakertilly.com). This mismatch caused inconsistent outcomes: two software modules developed in parallel could be capitalized on different schedules depending on definitional minutiae.

As examples, *project X* might have been in continuous requirement-gathering, making it unclear if it was still in "preliminary" or "development" stage at any point. Or *project Y* might have had new functionality added after initial go-live, raising questions whether that portion should be capitalized or expensed under maintenance rules (Source: www.cohenco.com) (Source: www.wallstreetprep.com). Many practitioners (including listed SaaS CFOs)

advocated for a more principles-based trigger. These concerns ultimately led the FASB to propose and adopt ASU 2025-06, which refocuses the decision onto management authorization and project risk, rather than procedural stage completion (Source: rsmus.com) (Source: www.armanino.com).

Key Changes in ASU 2025-06 (ASC 350-40)

In September 2025, FASB issued **ASU 2025-06**, *Targeted Improvements to the Accounting for Internal-Use Software*, effective for annual periods after Dec. 15, 2027 (with early adoption permitted) (Source: rsmus.com) (Source: dart.deloitte.com). This Update modernizes ASC 350-40 in several important ways:

- **Eliminate Project Stages:** ASU 2025-06 **removes all references to the preliminary, application development, and post-implementation stages** in ASC 350-40 (Source: www.bakertilly.com) (Source: www.bakertilly.com). Instead, there is a single decision point based purely on economics, not on where the project sits in a nominal lifecycle.

- **Probable-to-Complete Threshold:** Capitalization now begins *only when two criteria are met simultaneously* (Source: rsmus.com) (Source: www.eidebailly.com): (1) **Authorization and commitment of funding** has occurred, and (2) it is **probable** (per GAAP definition) that the project will be completed and the software used as intended (Source: rsmus.com) (Source: www.eidebailly.com). The requirement of "probable" uses the Master Glossary's meaning of *"likely to occur."* ASU 2025-06 emphasizes that both conditions must be satisfied to start capitalizing (Source: rsmus.com) (Source: www.eidebailly.com). This hinges on management judgment: for instance, executing a development contract or approving a budget constitutes authorization, while assessing technical/design uncertainties speaks to probability (Source: rsmus.com) (Source: www.eidebailly.com).

- **Significant Development Uncertainty (New Gate):** Crucially, if **significant development uncertainty** exists *at the time of evaluation*, the threshold is deemed **not met** (Source: rsmus.com) (Source: www.bakertilly.com). ASU 2025-06 explicitly defines two factors that indicate "significant development uncertainty" (Source: rsmus.com) (Source: www.eidebailly.com): (a) the software has novel/unproven features or technology whose risk has not been resolved through testing, and (b) the significant performance requirements have either not been identified or are still subject to substantial revision (Source: rsmus.com) (Source: www.eidebailly.com). If either factor is present, capitalization must be **deferred** until the uncertainty is resolved. For example, if a company is developing a cutting-edge AI analytics engine that has never been built before, the unknown technical hurdles would delay meeting the "probable" condition (Source: rsmus.com) (Source: www.eidebailly.com).

- **Alignment for All Delivery Models:** The ASU applies **equally to on-premises, hosted, or cloud-based** internal-use software projects, making it technology-agnostic (Source: www.crowe.com). It also formally **supersedes ASC 350-50** (Website Development Costs) and incorporates web/app development into 350-40 (Source: rsmus.com) (Source: www.bakertilly.com). Thus, both a company building an internal web portal and one updating an enterprise mobile app use the same capitalization test. This unified model covers traditional IT projects, ERP implementations, SaaS platform development, and websites under one ASC topic.

- **Disclosure Changes:** The update clarifies that **disclosures for capitalized internal-use software** follow the rules of ASC 360 (Property, Plant, and Equipment), not the general intangibles disclosures (Source: rsmus.com) (Source: www.eidebailly.com). This means that capitalized software should be treated like a fixed asset in cash flow statements and footnotes (e.g., shown in investing cash flows) (Source: www.armanino.com) (Source: www.eidebailly.com). Prior to ASU 2025-06, many companies disclosed internal-use software either under intangibles (ASC 350) or PP&E; the new guidance standardizes this presentation.

- **Transition Provisions:** Entities may adopt prospectively, modified-prospectively, or retrospectively (Source: rsmus.com) (Source: www.bakertilly.com). A prospective approach applies the new test only to costs incurred after adoption. With a modified method, in-progress projects that *were* capitalizable under old rules but *would not have been* under the new rules are written off as a one-time adjustment. Under full retrospective, prior periods are adjusted. The FASB explicitly allows early adoption (beginning of any annual period) (Source: rsmus.com) (Source: www.bakertilly.com), to let companies align their accounting with actual development practices sooner if desired.

In summary, **ASU 2025-06 refocuses the capitalization decision on prudence rather than process**. As Crowe LLP explains, the new guidance does *not* attempt to alter what costs are capitalizable; it only **reshapes *when* capitalization begins** (Source: www.crowe.com) (Source: rsmus.com). All analysts emphasize that this was the FASB's intent: to "modernize the guidance… make the rules neutral to development methodology and more operable for iterative environments" (Source: rsmus.com) (Source: www.bakertilly.com). For SaaS and cloud-based software, the update is expected to have the **greatest impact** because these firms often historically capitalized significant R&D as internal use (Source: www.crowe.com) (Source: rsmus.com). Going forward, the key accounting judgments will center on *defining the software project scope*, *identifying significant performance requirements*, and *documenting exactly when development uncertainty is resolved* (Source: www.crowe.com) (Source: www.crowe.com). Detailed examples in the ASU's basis section (illustrated below) underscore this approach.

Impact on SaaS and Cloud Software Companies

ASU 2025-06 has specific implications for **SaaS developers** and cloud-based software firms, whose business models commonly blur the lines between internal use and delivered product. Under legacy GAAP, many SaaS providers had a practice of capitalizing large chunks of development costs under ASC 350-40, even though those costs effectively built the customer-facing platform. Conversely, traditional licensed software firms would largely expense R&D until technological feasibility. ASU 2025-06 does *not* force SaaS companies to switch to ASC 985-20 – each arrangement is still evaluated under the existing licensing test. However, the new guidance makes it easier for SaaS companies to capitalize costs in an agile development process.

For SaaS developers, **judgment now shifts**. The focus is on when management has *implicitly or explicitly committed* to a project and whether it is likely to succeed (Source: rsmus.com) (Source: www.eidebailly.com). SaaS firms should identify each development initiative that meets the ASC 350-40 scope (no plan to market externally) and then apply the ASU's criteria. For each project, management must assess on an ongoing basis whether significant uncertainty remains. Armanino LLC reports that nearly **69% of public SaaS companies currently capitalize development costs** (Source: www.armanino.com), suggesting many projects already meet traditional criteria. Post-ASU, those same projects may trigger capitalization **earlier** (since meeting the test no longer depends on finishing a formal planning stage) but could also be **deferred longer** if uncertainty persists.

Judgment in Agile Development

A hallmark of SaaS development is iterative release and continuous improvement. Under ASU 2025-06, agile teams must carefully document when key thresholds are met. For instance, suppose a SaaS firm's product team begins building a new feature in an Agile sprint. Management has approved the project budget (authorization) and believes the feature will eventually deliver customer value (probable). However, until the team locks in the final scope and resolves any breakthroughs, the "probable-to-complete" condition may not yet be satisfied (Source: www.eidebailly.com). In this scenario, no costs would be capitalized initially; all technical sprints remain expensed. Once the scope stabilizes and all major uncertainties (e.g. a novel algorithm) are addressed, the company can retroactively start capitalizing costs (Source: www.eidebailly.com) (Source: www.eidebailly.com).

By contrast, consider a SaaS firm that contracts a third-party developer to build a fully-specified module. On signing the contract (and approving funds), management has both authorized the project and concluded that completion is probable (perhaps given the vendor's track record). If the module does not involve any "unproven" innovation, the probable threshold is met immediately. In that case, all salaries, vendor fees, and related costs from that point forward can be capitalized (Source: www.eidebailly.com). This mirrors **Example 1** in the ASU's illustrations: a service company executing a development contract on May 1 begins capitalizing that day (Source: www.eidebailly.com). Such outcomes emphasize that for SaaS companies, the **criteria are now forward-looking and contingent on facts rather than calendar**.

Importantly, ASU 2025-06 does **not change what costs qualify**. SaaS developers must still only capitalize those development costs intimately tied to creating or configuring the application (Source: www.wallstreetprep.com) (Source: www.cohenco.com). Ordinary product maintenance, training for users, or general support will continue to be expensed (Source: www.cohenco.com) (Source: www.wallstreetprep.com). For example, a SaaS firm might spend on staff training *after* a new platform release; those post-implementation training costs are specifically expensed (Source: www.cohenco.com) (Source: www.wallstreetprep.com). Likewise, enhancements that simply fix bugs without adding functionality remain expensed. In practice, many companies will map these rules to their project accounting: costs under task categories like "Coding" or "Configuration" are capitalizable, whereas "Testing" and "User training" might require analysis (testing is usually capitalizable, **except** routine bug fixes) (Source: www.wallstreetprep.com).

Comparability and Reporting Effects

Because ASU 2025-06 often results in **earlier capitalization** in iterative projects, it will tend to **inflate assets and reduce expenses in the early stages** of development (compared to the old model). Conversely, projects encumbered by uncertainty will show *later* capitalization. The net effect on a company's financials depends on how many projects cross the threshold. However, there is broad consensus that the change **improves comparability**. As an Armanino white paper notes, under current GAAP similar projects can have "different looking financials" depending on phasing; the update aims to make accounting reflect the economic substance (long-lived R&D assets) regardless of contract or methodology (Source: www.wallstreetprep.com) (Source: www.armanino.com). In fact, the Armanino survey quotes a former SaaS CFO who views the targeted improvements as a step toward "aligning with SaaS companies' practices, moving away from the 'technical feasibility' standard," and advocates eventually unifying internal and external software accounting (Source: www.armanino.com) (Source: www.armanino.com).

From a disclosure perspective, software companies will now present all their capitalized development under PP&E. ASU 2025-06 requires that these deferred software costs be shown as an investing cash outflow in the cash flow statement (like any other capital expenditure) (Source: www.armanino.com). Entities must disclose the nature of capitalized software costs and their amortization period, but they no longer need to make

separate intangible disclosures (Source: rsmus.com) (Source: www.eidebailly.com). In practice, one can expect footnote changes (e.g. footnotes on Property and Equipment will include software in cost tables).

SEC and Regulatory Perspective

To date, SEC filings suggest that the Commission has focused more on recent revenue recognition and commission capitalization issues than on internal-use software. The Armanino report found **no SEC comment letters** on software development expense diversity over the past couple of years (Source: www.armanino.com), even as comment letters on ASC 340-40 (commission costs) have increased (Source: www.armanino.com). This may reflect that ASC 985-20 vs ASC 350-40 has long been settled practice for most public companies, and that the new ASU's changes do not create a discontinuity in a way that would alarm regulators. Nonetheless, companies preparing for adoption should be aware that the ASU's treatment of cash flows (requiring internal software costs to be investing outflows) could affect cash flow disclosures and metrics.

Overall, for SaaS developers **the core change** is operational: develop consistent policies to assess and document the new capitalization threshold for each major development effort. Mixed teams of finance and IT should collaborate to define what constitutes a "software project" unit, identify significant performance requirements early, and track when uncertainties are resolved (Source: www.eidebailly.com) (Source: www.crowe.com). NetSuite, as an enterprise system, can be set up to support this process (see below). From a strategic standpoint, many SaaS CFOs see ASU 2025-06 as codifying the reality that internally developed code is an asset – aligning accounting with how other R&D is treated (e.g., Section 174 capitalization for tax (Source: www.armanino.com). A forward-looking SaaS company should evaluate its budgets, cash flows, and tax planning in light of the capitalized R&D that may result.

Characterization of Capitalizable vs. Expensed Costs

Under both old and new guidance, only certain **direct development costs** can be capitalized; others must be expensed in the period incurred. ASU 2025-06 does *not* expand or contract the list of eligible costs, but understanding this list is critical to implementation. By ASC 350-40, capitalizable internal-use software costs generally include: (a) **external direct costs** of materials and services (paid to third-party vendors or consultants) specifically for the software project, and (b) **internal payroll and payroll-related costs** for employees who are directly associated with and devote time to the project (Source: www.scribd.com). For example, a company might hire consultants to code a custom CRM and internally assign a team of developers; *both* the consultants' fees and the developers' salaries (and their benefits) can be capitalized once the criteria are met (Source: www.rit.edu) (Source: www.scribd.com). Importantly, stock-based compensation for employees on the project is also included in capitalizable payroll costs.

Indirect overhead or G&A costs (including general executive time, marketing, rent, utilities, or broad IT support) are **not** capitalized (Source: www.scribd.com) (Source: www.wallstreetprep.com). For instance, the time that the CEO spends in oversight meetings does not qualify. Training costs are typically expensed; ASC 350-40 excepts capitalization of employee training even during application development (Source: www.cohenco.com) (Source: www.wallstreetprep.com). Similarly, costs like travel or lodging are capitalizable *only if* they are directly allocable to the project (e.g. consultants traveling specifically to implement the software) (Source: www.rit.edu). A helpful rule of thumb: costs that would not have been incurred "but-for" the development project should be capitalized; anything that would have been a normal operating spend stays an expense.

Once capitalized, the asset is amortized over its useful life. ASC 350-40 has long required straight-line amortization (without mention of impairment unless the project is abandoned) (Source: www.wallstreetprep.com). In practice SaaS companies commonly amortize over 2–5 years depending on expected technology obsolescence. For example, AthenaHealth amortized its internal platform over 2–5 years (Source: www.wallstreetprep.com); Alphabet (Google) also indicated its internal-use software lives usually fall in this range (Source: www.wallstreetprep.com). Effective lives are a matter of policy and estimation; companies must justify that capitalized costs are matched to the revenues or benefits they enable.

Perspectives and Opinions

Multiple industry and expert perspectives help illuminate the significance of ASU 2025-06:

- **FASB/Standard-Setters:** The FASB's own Basis for Conclusions (and accompanying press release) emphasize that the new model preserves existing capitalization criteria but "better aligns the accounting… with current development approaches" by removing "prescriptive" stage gating (Source: rsmus.com) (Source: www.bakertilly.com). A Deloitte Heads Up similarly notes that FASB considered an ITC (Invitation to Comment) which highlighted the need for modernization. Importantly, FASB decided *not* to align internal software accounting with ASC 985-20, meaning internal vs external software still follow separate frameworks (Source: dart.deloitte.com). The Board sees ASU 2025-06 as a targeted change, not a wholesale revamp.

- **Accounting Firms and Analysts:** Major firms have published guidance on ASU 2025-06. RSM explains the new "probable-to-complete recognition threshold" concept and provides practical tests; Baker Tilly emphasizes the removal of all project-stage references (Source: rsmus.com) (Source: www.bakertilly.com). Crowe points out that the version of the probable-complete test now explicitly requires companies to **search for any significant uncertainty** in every project, and that documentation will be critical (Source: www.crowe.com) (Source: www.crowe.com). These analyses resonate with practitioners: the new model shifts judgment calls away from label-based gating toward evidence-based evaluation.

- **Corporate Finance (CFO/Controller) View:** From the corporate side, the update has been seen largely as clarification. In an Armanino survey white paper, **former SaaS CFO Andrew Hyde** observed that the amendments "appear to have been derived from consideration of the single model approach" and "reflect what is already occurring as software companies align with SaaS companies' practices." Hyde nonetheless criticizes the continued divergence of methods, arguing that ultimately internal and external software capitalization lead to similar results anyway (Source: www.armanino.com). On balance, many SaaS finance leaders welcome the flexibility: fewer artificial constraints means aligning accounting with agile delivery. However, they caution that significant engineering deliverables (like a proprietary AI module) may still require lengthy R&D phases with inherently uncertain outcomes.

- **Investors and Analysts:** Although equity analysts seldom comment on software capitalization explicitly, the move has implications for key metrics. By allowing earlier capitalization, ASU 2025-06 effectively **defers expenses to later periods**, boosting near-term operating income. This could modestly improve EBITDA and margin profiles for fast-growing software companies in transition. Conversely, projects still in flux will carry more expense up front, possibly making current losses larger but cleaner. Analysts will watch to see if increased intangible asset balances (and amortization) change valuation models. In sum, the change could make SaaS companies' reported profits more comparable, reducing one source of diversity between on-prem and cloud business models.

- **Tax and Regulatory:** Tax authorities have also changed ground recently. Most notably, IRC §174 was amended (effective 2022) to **require R&D expenditures – including internal software development – to be capitalized and amortized over 5–15 years** (Source: www.armanino.com). Many SaaS companies found themselves paying far higher taxes because what had been deductible expenses must now be amortized. Armanino reports cases of companies facing "three to four times" last year's tax due to this shift (Source: www.armanino.com). ASU 2025-06 is **independent of tax law** (it does not modify §174), but the two trends reinforce each other: activities capitalized under GAAP will generally be capitalized for tax, increasing deferred tax balances. Companies need to plan for these cash flow effects. On the regulatory side, the SEC's Division of Corporate Finance has not signaled any new concerns specifically about internal software capitalization (no comment letters have been issued on that topic recently (Source: www.armanino.com), but issuers should ensure that their disclosures clearly explain any changes in capitalization policy and how the added intangibles are amortized.

Implementation Guidance for SaaS Developers (NetSuite Focus)

For SaaS companies using NetSuite as their ERP/financial system, compliance with ASU 2025-06 will involve configuring NetSuite to track project expenditures and allocate costs to capital versus expense. Proper setup ensures that once a project crosses the capitalization threshold, all qualifying costs roll into a **capital asset account**, to be amortized, rather than flowing through current expense. Below are practical steps and features that SaaS finance teams should consider when using NetSuite under the new rules:

1. Define Software Projects and Budgets: Create distinct **Project records** (using NetSuite's Projects or Job Costing module) for each significant development initiative. Assign a *project manager* and job tasks (e.g. design, coding, testing). Using NetSuite Project Budgeting, allocate projected hours and costs by phase and resource type (Source: docs.oracle.com). This lets management monitor spend real-time during development. Crucially, projects identified as eligible for capitalization should be flagged (e.g., via a project custom field) so that costs can be segregated. For example, RIT suggests using a specific category (like an FEC code or NetSuite project classification) for capitalizable SaaS-implementation costs (Source: www.rit.edu).

2. Set Up Chart of Accounts / Expense Types: Under NetSuite's **Job Costing** feature, you can define *Project Expense Types* which dictate which GL accounts time and expense entries post to (Source: docs.oracle.com). We recommend creating separate project expense categories for "Capital Development" vs "R&D Expense" vs "Support". For instance, a "Capital Development" expense type could point to a non-current **Deferred Software Development** asset account, while "Maintenance" expense types point to P&L expense accounts. When team members enter time/costs on the project, the project expense type will credit the appropriate account. In OneWorld setups, intercompany adjustments can also be generated automatically if R&D is incubated in one subsidiary but expensed in another (Source: docs.oracle.com).

3. Recording Capitalizable Transactions: Once a project meets the ASU's capitalizable threshold, all qualifying expenditures should be coded to the designated *asset* accounts. In NetSuite, this might mean using **Expense categories** or **Item records** tied to capital accounts. For example, internal labor costs (captured via timesheets) would flow to the Deferred R&D asset account (through job costing) instead of a salary expense account.

Similarly, vendor bills for contract developers would be coded with an Expense category that posts to the same asset account. (NetSuite's **amortization enablement** can be used here: by assigning an Amortization Template to the expense line, the system will automatically create a deferral.) The goal is that all capitalizable expenditures accumulate in a balance sheet account rather than the Income Statement during the capitalization period.

4. Amortization and Asset Tracking: After costs are capitalized, they can be **amortized** over their useful life. NetSuite offers two main approaches:

- **Amortization Templates:** Enable the **Amortization** feature (Enable > Feature > Accounting > Amortization). Create an Amortization Template for software development costs specifying the useful life periods (for example, 36 months). Set the expense recognition method (e.g. straight-line) and link it to the Deferred Asset accounts (Source: netsuitedocumentation1.gitlab.io) (Source: netsuitedocumentation1.gitlab.io). When relevant transactions are recorded (bills, expense reports with that template), NetSuite will generate scheduled journal entries each period moving a portion of the deferred balance to amortization expense. This aligns with FASB's requirement that capitalized software be amortized like a long-lived asset. Practically, a $120k project cost could be amortized $4k monthly over 30 months, for instance.
- **Fixed Asset Module:** Alternatively, if the company has NetSuite's **Fixed Assets Management** SuiteApp, one can treat capitalized software as an intangible asset. Under this option, create an Asset record for the completed software (with the total capitalized cost and useful life). The system will then automate depreciation/expense entries. This method may streamline reporting (software appears on the asset register) and ensures integration with multi-book accounting if needed.

5. Reporting and Monitoring: Use NetSuite's built-in reports to verify balances. The **Project Profitability** report (with Cost actuals tab) can be tailored to show capital vs expense by project (Source: docs.oracle.com). NetSuite's **Trial Balance** or **Balance Sheet** reports should now include a line for Deferred Software Assets (roll-up of all capitalized projects). Every close period, reconcile that balance and ensure the correct amortization postings occurred. NetSuite's **Audit Trail** (transactions audit) can support documentation: tie back journal entries to project IDs. Additionally, periodic disclosures (in footnotes) should break out total capitalized software and amortization (net of disposals), which NetSuite reporting can provide from these accounts.

6. Governance and Process Changes: Beyond system set-up, companies must adapt processes. Project teams should trigger notifications to finance when a project crosses the ASU threshold. Control frameworks (e.g. SOX) should include review of project charters vs actual costs. NetSuite can facilitate approvals (e.g., approval workflow on project task completion marking "ready to capitalize"). All evidence (meeting minutes, technical sign-offs, test results) justifying "probable to complete" should be documented as audit support, though this may reside outside NetSuite.

7. Example Configuration: As a concrete illustration, consider a SaaS firm implementing a new feature. They set up NetSuite project **"Feature X Development"**, status "In Progress". The controller configures two expense types: (a) *Dev Labor – Capital* mapping to the Deferred R&D asset account, and (b) *Dev Labor – Expense* to a normal R&D expense GL. When the manager decides the project passes the ASU test, all future developer hours rendered on the project are categorized under *Dev Labor – Capital* (e.g. timesheets with that type). Similarly, a consultant bill for building the feature is coded to a *Professional Fees – Capital* expense category. The result: at month-end, the project's P&L shows no expense (the costs queue in the Balance Sheet). Subsequently, each month an automated amortization journal reduces the Deferred R&D balance and records amortization expense in the P&L.

By carefully configuring NetSuite's **job costing and amortization features**, a SaaS developer can smoothly implement ASU 2025-06. The built-in NetSuite tools – Project management, budgets, time tracking, and amortization templates – provide a strong framework. What is critical is leveraging them to differentiate capital vs expensed work. OneNetSuite whitepaper notes that enabling Job Costing allows labor costs to be "accounted for in your general ledger" by project (Source: docs.oracle.com), which is exactly the needed capability. In practice, many companies achieve compliance by creating dedicated GL accounts for "Capitalized Software Development" and using NetSuite's Project Expense Types to drive postings there as described above.

Data Analysis and Evidence

Several data points and surveys illuminate the effects of internal-use software accounting and the context for ASU 2025-06:

- **Industry Survey (Armanino, 2024):** A comprehensive survey of the top 100 U.S. public SaaS companies found that **89%** of these companies now capitalize sales commissions (ASC 340-40) – reflecting ASC 606 effects – up from just 22% in 2017 (Source: www.armanino.com). Focusing on software development, **69%** of those SaaS firms capitalize internal software costs today (vs 62% in 2017) (Source: www.armanino.com). This indicates that a strong majority of SaaS companies already use ASC 350-40 in practice, so the new ASU's clarifications will codify an existing trend rather than overturn practices. Interestingly, the study also notes that larger and smaller SaaS firms differ little in capitalization rates: development cost capitalization is consistently high across revenue bands.

- **SEC Filings Analysis:** The Armanino report analyzed Form 10-K disclosures to ascertain capitalization practices (Source: www.armanino.com). Many companies explicitly cite ASC 350-40 for internal costs or ASC 985-20 for external development (in Alphabet's case) (Source: www.wallstreetprep.com). Alphabet's 2017 10-K, for example, states they "expense software development costs" until feasibility (thus capitalizing *almost none* under 985-20), but they do capitalize internal-use software once preliminary is complete (Source: www.wallstreetprep.com). AthenaHealth's 10-K reveals the converse: it capitalizes substantial "athenaNet" development under 350-40 (Source: www.wallstreetprep.com). These real-world disclosures underscore how choice of model can drastically affect reported R&D spend.

- **Cost of Compliance (SEC Comments):** While SEC comment letters on software R&D are apparently rare (Source: www.armanino.com), letters on related codification (like capitalized commissions) have been climbing. The survey reproduced sample SEC statements that probe how costs like business development or renewal commissions are treated under ASC 340-40 (Source: www.armanino.com) (Source: www.armanino.com). Although not directly about ASC 350-40, this trend suggests that any material change in capitalization policy (e.g. under ASU 2025-06 adoption) should be well-explained in filings to avoid scrutiny.

- **Tax Data:** Tax compliance data isn't publicly tabulated, but anecdotal evidence is stark. Firms we reviewed reported **substantial first-year tax hits** due to TCJA's capitalization of 174 R&D costs (Source: www.armanino.com). For many SaaS companies, this meant tens of millions of dollars of costs could no longer be immediately deducted, inflating taxable income in 2022. The Armanino analysis warns that only half-year amortization applies, so longer planning is needed (Source: www.armanino.com). Going forward, ASU 2025-06 doesn't affect tax law, but the alignment of GAAP capitalization with tax rules makes the issue more acute for decision-makers.

- **Academic Study:** A 2021 survey of software industry accounting practices (Mulford & Roberts, 2021) found wide variance in capitalization rates and noted that differences in policy can make two similar companies appear very different in profitability (Source: www.wallstreetprep.com). The authors concluded that without consistent accounting, comparisons among tech companies are difficult, echoing the rationale for this ASU. This aligns with Baker Tilly's comment that the ASU "reduce[s] diversity in practice" by improving operability (Source: www.bakertilly.com).

These data and commentaries provide strong support for FASB's changes. They show that (a) most SaaS firms already *want* to capitalize software costs; (b) current rules cause inconsistencies; (c) tax changes have forced a capitalization mindset; and (d) the new threshold approach is seen as a natural evolution in the industry (Source: www.armanino.com) (Source: www.wallstreetprep.com).

Case Studies and Examples

To illustrate the new guidance in action, we consider representative scenarios adapted from FASB's own examples and industry practice:

- **ERP Implementation (Contracted Development):** *Service Co.* plans a company-wide ERP upgrade. It spends January–April on research and vendor demos (preliminary stage). On **May 1**, it signs a fixed-price contract with an ERP firm to implement and customize a module. At that point, management has both *authorized funding* (the signed contract) and concluded it is *probable to complete* (the technology is proven and scope fixed). According to Example 1 in ASU 2025-06 (Service Co.), capitalizable costs begin **May 1** (Source: www.eidebailly.com). All post-May 1 costs (internal staff, vendor fees) are capitalized; eg, if an engineer spends 100 hours coding after May 1, those wages go to an asset account. Prior costs (pre-May 1 research) remain expensed. This reflects the updated rule: once funding and feasibility are set, development costs accrue to the balance sheet (Source: www.eidebailly.com).

- **New Mobile App (Undetermined Scope):** *Connect Co.* internally develops a mobile app. On **March 1**, management approves the budget and assigns a team, but the team is still defining "what the app needs to do." Initially, this project has *significant uncertainty* because the functional requirements aren't yet identified. ASU 2025-06's criteria are not yet met in March: funding is authorized, but "probable completion" is questionable due to undefined requirements (Source: www.eidebailly.com). Hence **March 1** costs are expensed. Over the next months, prototypes are built and users give feedback. By **November 1**, management finalizes the core features (resolving uncertainty). As Example 2 in the guidance shows, on November 1 the thresholds are now satisfied (innovation uncertainties are gone and scope is set), so the company begins capitalization (Source: www.eidebailly.com). All eligible costs from November onward go to the asset account; costs before that date (and any exploring changes) incurred in development remain expensed (Source: www.eidebailly.com). This scenario highlights that the ASU allows companies to *retroactively start capitalizing* once conditions clear.

- **Novel Technology Development:** *Tech Co.* starts designing a groundbreaking AI module on **July 1, 20X1**, approving a budget and an in-house team. However, the functionality is truly novel. The team struggles for months. Only on **April 1, 20X3** does coding prove the concept and the requirements stabilize (neither major feature changes nor tech hurdles remain). According to FASB's Example 3, the "probable-to-complete" requirement is met on April 1, 20X3 (Source: www.eidebailly.com). Management had committed on July 1, 20X1, but until April 2023 significant

uncertainty existed (novel features unresolved). Therefore, *capitalization starts April 1, 20X3*. All eligible costs from April 1 to May 1, 20X3 (when testing is completed) are capitalized. This shows ASU 2025-06's focus on resolving uncertainties: even with funding in place, no costs were capitalized until the project was de-risked (Source: www.eidebailly.com).

- **SaaS Feature Upgrade:** Consider *SaaSCo*, a subscription software vendor, developing a major new analytics feature. The product team authorizes the project on **June 15, 20X2** but continues iterative development (Agile sprints) through late 20X2. By December, core functionality is fully defined and tested (no major changes forthcoming). Under old rules, capitalization might have been unclear (since phases overlap), but under ASU 2025-06 the test falls on December: with funding approved and no longer significant uncertainty, SaaSCo begins capitalizing from that point on. NetSuite can help track this: the finance team could mark the project "Capitalizable as of 12/1/20X2" and reclassify time entries billed prior (NetSuite allows reclassification of historical time if needed). This shift means half of the project's cost gets capitalized, with Amortization kicking in from December onward. If the project were scrapped instead, all costs would simply be expensed, and any previous deferrals reversed.

These examples illustrate how ASU 2025-06 shifts the capitalization decision from "did we hit the development phase?" to "have we committed and can it succeed?". In each case, the project's *unique facts* determined the exact capitalization date. Real companies will face analogous choices: e.g., a startup's CTO might document in meeting minutes when the development plan stabilized; an accounting team should file a memo or checklist when a project first meets authorization and resolution of uncertainty. NetSuite's Project Management tools can support these workflows by recording key project dates and milestone completions, providing an audit trail.

Implications and Future Directions

Financial and Strategic Implications

For SaaS developers and other software companies, ASU 2025-06 has several key implications:

- **Earnings Profile:** By enabling earlier capitalization in Agile projects, companies will report lower expenses (and higher pre-tax income) in the early development period. The deferred expense is amortized later, smoothing the impact. Analysts should note that some firms may see a one-time jump in intangible assets immediately after adoption. Comparability across companies should improve, however, since the rule base is now uniform regardless of methodology.

- **Cash Flow Reporting:** Software R&D outlays will now more clearly appear as **investing** cash flows. Companies should adjust their cash flow analyses accordingly, as the ASU explicitly requires capitalized software outflows to be classified as investing activities (Source: www.armanino.com). This could affect metrics like free cash flow or capex ratios for tech firms. Finance teams must ensure their statement of cash flows and footnotes align with this treatment.

- **Tax Planning:** Given IRC §174 changes, the expanded capitalization of development costs under ASC 350-40 often coincides with the need to amortize those costs for tax. Companies should continue to coordinate with tax advisors on timing of deductions and use of credits. Since ASU 2025-06 allows retrospective adjustments, some firms may choose to prospectively apply it (minimizing restatement risk) and align tax and book metrics slowly. The Armanino guidance highlights that managing taxable income and cash (e.g. adjusting estimated tax payments) is now a board-level issue in many SaaS firms (Source: www.armanino.com).

- **Policy and Audits:** Auditors will scrutinize how companies interpret "probable" and "resolved uncertainty." Detailed documentation will become central to audits of R&D spend. Entities may need to update accounting policies (in internal manuals) to reflect the new capitalization trigger. The increased judgment means that companies should be consistent and rigorous: once the threshold is met for a project, all similar costs should be treated similarly. Audit committees should be briefed on how project budgets, risk assessments, and threshold decisions are made, as Baker Tilly suggests boards will want to "understand the changes" brought by ASU 2025-06.

- **Investment & Valuation:** Investors often use measures like R&D/revenue or R&D/valuation to assess tech companies. ASU 2025-06 will reduce reported R&D expense for a time, potentially inflating short-term profitability metrics. Valuation analysts should adjust models to account for differences in capitalized versus expensed R&D. Over the life of a project, however, total recognized costs will be the same; it is a timing shift. The key benefit is reliability: capitalized intangibles reflect future economic value (provided companies sustain the projects).

Future Accounting Landscape

ASU 2025-06 is an intermediate step in the broader evolution of software accounting. Several related developments are in play:

- **Single Model Debate:** There is a longstanding debate (reflected in the FASB Invitation to Comment and external commentary) about whether internal and external software accounting should converge. CFO Andrew Hyde's comments (Source: www.armanino.com) echo that debate. While FASB stopped short of unifying the models in 2025-06, it did move both internal (350-40) and website (350-50) to a single internal framework. Some industry voices are pushing for full convergence with ASC 985-20 (the "single model approach") (Source: www.armanino.com). If FASB or IASB ever pursue a more unified standard, it might involve adopting the "technological feasibility" notion for all software. Such a change would be far-reaching (and likely require a separate mega-project), but the targeted improvements here do not close that gap.

- **International Considerations:** Under IFRS, internally generated intangibles are governed by IAS 38. IAS 38 requires firms to capitalize development costs only after meeting criteria (technical feasibility, intention to complete, etc.), not at the planning stage. In practice, IFRS's model is more similar to ASC 985-20 than ASC 350-40; many jurisdictions historically did not allow capitalization of internally developed software (treating all as intangible R&D expense until completion). ASU 2025-06 therefore *widens* the GAAP vs IFRS gap, as U.S. companies will continue capitalizing internal developments under ASC 350-40 while IFRS companies amortize. Any move by FASB toward a single approach might bring us closer to IFRS treatment. For now, multinational SaaS firms must reconcile that they use ASC 350-40 in the U.S. and possibly a stricter model abroad.

- **Software Development Trends:** Emerging technologies like AI may further change the accounting. Armanino notes that internal AI training costs (to build new software) are currently *not* capitalizable – they are R&D expense (Source: www.armanino.com). As AI becomes more embedded in products, guidance may evolve (and projects will likely have shorter useful lives). Moreover, "continuous" delivery models (evergreen software) challenge the definition of project scope – companies will have to decide how to delineate one "project" from another for accounting purposes. The unit of account guidance, left to judgment in ASU 2025-06, will thus become an important policy issue.

- **Future FASB Projects:** The FASB proposed Revision in late 2024 (prior to final ASU) on *Initial Development Costs* (Source: www.armanino.com). That proposal's substance closely mirrors what was issued in 2025-06. Going forward, FASB may revisit related topics, such as explicit cash flow presentation (already required by the proposal (Source: www.armanino.com) or long-term asset definitions. Stakeholders may also petition the FASB to consider software as a broader intangible asset category (for example, how to account for internally developed software transferred or sold). Audit firms will likely share practical challenges with the FASB as companies adopt ASU 2025-06, guiding any future amendments.

Conclusion

Accounting for internal-use software has been a complex issue for decades, but ASU 2025-06 marks a significant modernization. By **removing obsolete stage-gating** and introducing a clear threshold test, FASB has aligned U.S. GAAP with the realities of iterative software development. For SaaS companies, the update provides much-needed clarity: capitalization of a new feature or platform now depends on deliberate judgments about funding and uncertainty, rather than the sometimes-arbitrary completion of a planning phase. This should reduce practice diversity and improve comparability across firms and projects.

Nonetheless, the responsibility now shifts to management to exercise sound judgment and robust documentation. Finance and IT teams must establish processes to determine and record exactly *when* a project first meets the "authorized and probable" criteria. Proper integration into systems like NetSuite will be key – using project accounting and amortization features to segregate and amortize the captured costs. The examples and guidance above illustrate how that might look in practice.

Finally, ASU 2025-06 does not stand alone. It reflects broader trends: the convergence debate in software accounting, the implications of tax law changes, and the strategic valuation of intangible assets in technology firms. As one former SaaS CFO observed, ultimately companies should be able to justify capitalizing a project under *either* the internal-use or external software model (Source: www.armanino.com). In the meantime, ASU 2025-06 provides a more streamlined path. SaaS developers and their finance teams should use this opportunity to align their accounting policies and systems with the economic substance of their product development, ensuring that NetSuite and other tools accurately capture the new treatment of software costs.

In conclusion, **FASB ASU 2025-06 establishes more operable, adaptable rules for internal-use software**. It balances flexibility for modern development with accountability to shareholders. By following its guidance — as detailed in this report — SaaS companies can confidently capitalize eligible costs in NetSuite, remain GAAP-compliant, and present clearer financials. The extensive references cited herein (from FASB releases to industry white papers) demonstrate the depth of analysis and consensus around this topic. Going forward, firms should monitor further pronouncements and best practices, but can be secure in knowing that ASU 2025-06 represents the current authoritative guidance on capitalizing software development in NetSuite and beyond.

References: Authoritative GAAP literature and professional analyses were used throughout. For example, FASB's ASU 2025-06 summary and illustrations (Source: rsmus.com) (Source: www.eidebailly.com), RSM and Baker Tilly white papers (Source: rsmus.com) (Source: www.bakertilly.com), the Crowe LLP analysis (Source: www.crowe.com), and the Deloitte DART update (Source: dart.deloitte.com) provide the technical framework. Industry practice and perspectives are drawn from sources like the Armanino SaaS survey (Source: www.armanino.com) (Source: www.armanino.com), Wall Street Prep and AccountingTools guides (Source: www.wallstreetprep.com) (Source: www.wallstreetprep.com), and NetSuite documentation (Source: docs.oracle.com) (Source: netsuitedocumentation1.gitlab.io). All factual claims above are supported by these citations.

Tags: fasb asu 2025-06, asc 350-40, internal-use software, software capitalization, saas accounting, netsuite accounting, gaap standards, capitalized software costs