certinia

Build vs. Buy in the Age of Vibe Coding

Navigating your options for professional services and customer success

Executive summary

For decades, the build vs. buy debate has been a defining strategic decision for companies adopting business software. The calculus has always favored buying for most organizations — the cost, time, and expertise required to build and maintain enterprise-grade software rarely justified the investment unless the capability in question was a true source of competitive differentiation.

Now, a new force is reshaping the conversation: AI-assisted development, commonly known as "vibe coding." With tools like GitHub Copilot, Cursor, and Claude, developers can generate functional code faster than ever before. The promise is tantalizing — could companies now build their own Professional Services Automation (PSA) or Customer Success platforms cheaply and quickly?

The answer is no. And in some ways, vibe coding makes the case for buying stronger, not weaker.

This whitepaper examines the "build vs. buy" decision through the lens of AI-assisted development, specifically within the PSA and Customer Success software Certinia serves. We will demonstrate that while vibe coding significantly lowers the barrier to entry for writing initial code, it does not reduce—and may even amplify—the hidden costs and technical debt that have traditionally made a pure "build" strategy a high-risk gamble.

However, the most competitive organizations will not choose between these two paths. Instead, they will find the greatest value by anchoring their operations in purpose-built platforms that provide a secure, governed data architecture. By using these platforms as a foundation, organizations can then safely leverage vibe coding for high-value edge cases and customizations, turning AI-assisted development into a multiplier for innovation rather than a source of long-term instability.

Section 1: Understanding the new landscape

What is vibe coding?

Vibe coding refers to the practice of using large language model (LLM)-powered tools to generate, complete, or refactor code through natural language prompts. A developer (or even a non-developer) describes what they want in plain English, and an AI model produces working code.

The productivity gains are real. Studies from Microsoft and GitHub have shown that developers using AI coding assistants complete tasks 55% faster on average. For greenfield projects — building something from scratch — the acceleration is even more pronounced.

This has led some technology leaders to consider a tempting proposition: "We have a few developers and access to AI tools. Could we just build our own PSA system?"

The question beneath the question

Before evaluating whether you can build, it is essential to ask whether you should. The build vs. buy decision has never been purely a question of development cost. It is a question of total value — and total value includes acquisition cost, implementation time, ongoing maintenance, opportunity cost, compliance burden, integration complexity, and the compounding value that comes from continuously improving software built specifically for your industry.

Vibe coding changes the numerator (initial development cost) but leaves the denominator — total sustained value — largely unchanged.

2. The true cost of building — even with AI

2.1 The “last mile” problem

AI-generated code excels at producing functional prototypes. It does not excel at the last 20% of an enterprise software project — the part that makes software production-ready.

That final 20% typically includes:

Security hardening — input validation, injection prevention, privilege separation, encryption at rest and in transit

Audit logging and compliance — SOX, GDPR, and industry-specific regulatory requirements

Performance at scale — behavior under thousands of concurrent users, large data volumes, and complex multi-tenant configurations

Error handling and edge cases — the business logic exceptions that only emerge after months of real-world use

Accessibility — WCAG compliance for enterprise deployments with diverse user populations

AI tools can technically attempt all of the above — but they do not meaningfully reduce the cost of getting them right. Producing secure, performant, production-ready code through AI requires significant iteration: the model generates, tests, adjusts, and tests again, because it is not reasoning from deep engineering knowledge but predicting the next most likely token. For complex problems like performance optimization or security hardening, this loop can run many cycles before converging on a workable solution — and token costs accumulate rapidly with each pass. The result is that LLM-assisted development in these areas can still be faster than purely manual approaches, but the cost and effort savings are far smaller than proponents suggest, and the outputs demand the same rigorous human review as any other code. For a PSA platform handling project financials, revenue recognition, and professional service delivery, the stakes of getting these wrong are severe — and no amount of iteration speed changes that calculus.

2.2 The ongoing maintenance tax

The most underestimated cost of building software is not building it — it is maintaining it. Research from Planet Crust and Forrester found that initial development represents only 20% of the total cost of ownership over a five-year period. The remaining 80% — the invisible costs — are where custom builds consistently destroy value. McKinsey's analysis of the cost breakdown over a five-year horizon is instructive: ongoing maintenance and IT support (~25%), integration and API management (~20%), innovation and customization (~20%), compliance and security (~10%), and hardware and infrastructure (~5%) dwarf the initial build cost. Perhaps most sobering, McKinsey and the University of Oxford found that on average, large IT projects run 45% over budget and 7% over time, while delivering 56% less value than predicted.

Platforms like Certinia are updated continuously to reflect changes in Salesforce platform releases (three per year, each requiring compatibility testing), tax law and revenue recognition standards (ASC 606, IFRS 15), security vulnerabilities and patches, customer-driven feature enhancements, and integration protocol changes from connected systems. A custom-built PSA system inherits all of these obligations without the benefit of a dedicated product and engineering organization behind it. Vibe coding can help write new features, but it cannot replace the institutional knowledge embedded in a mature product roadmap built over years in a specific domain.

A useful mental model: vibe coding reduces the cost of writing software. It does not reduce the cost of owning software.

2.3 The domain knowledge gap

Building a PSA or Customer Success platform is not primarily a software challenge. It is a domain knowledge challenge. Effective PSA software must encode:

- Project accounting principles and revenue recognition rules
- Resource management optimization logic
- Utilization and revenue forecasting models
- Services-specific KPIs and reporting taxonomies
- Best practices for professional services and customer success across industry verticals

This knowledge took Certinia years to accumulate through deep collaboration with professional services organizations around the world. It is embedded in data models, workflow logic, reporting structures, and product decisions that are invisible at first glance but critical to operational success.

An AI coding tool generates code. It does not generate this domain expertise. A custom-built system starts at zero on this dimension, regardless of how quickly the code is written.

2.4 The hidden cost of developer attention

Every sprint a development team spends maintaining a custom PSA system is a sprint not spent on the capabilities that actually differentiate your business. For a professional services organization, competitive differentiation comes from delivery excellence, client relationships, and service innovation — not from maintaining a time-tracking module.

A frequently overlooked dimension of this cost is context switching. AI tools do allow engineering teams to produce more software than before with the same headcount — but engineers still need to maintain a mental model of every project they own in order to validate that AI-generated output is correct, secure, and consistent with the existing codebase. That cognitive overhead does not compress the way code generation does. An engineer who delivers 40 productive hours per week on a single focused project may deliver only 10 when context-switching across five projects providing support and maintenance. The productivity loss is not linear — it compounds with each additional system pulled under internal ownership, and at sufficient scale it forces organizations to begin isolating engineers by product area just to preserve output quality.

What this means in practice is that every additional system a company chooses to build and maintain internally increases the strain on each engineer, shrinks the effective output of the team, and raises the maintenance cost per system. Vibe coding accelerates the initial build, but it does not resolve — and may actually accelerate — the accumulation of internal software obligations that drives this problem. As that portfolio grows, the tipping point arrives sooner than most organizations anticipate.

When vibe coding enters the picture, this opportunity cost becomes even more visible. Your developers' time is now more valuable, not less. The question is not "can we use AI to build this?" but "is this the highest-value use of our AI-augmented development capacity?" For most professional services organizations, the honest answer is that maintaining a custom PSA platform is not — and the context-switching tax ensures it never will be.

3. What vibe coding actually changes

To be intellectually honest, vibe coding does change some of the build vs. buy calculus but not in the way proponents typically claim.

3.1 Prototyping speed vs. production readiness

Vibe coding dramatically accelerates the creation of a working prototype, the visible tip of the iceberg. This is the phase that feels like progress: a demo that does the core thing you wanted. But research consistently shows that the initial build represents only 20% of five-year total cost of ownership for traditional software, and the ratio is even more extreme for AI-powered builds, where the initial development accounts for just 10% of full lifecycle cost. The invisible 90%: data infrastructure, observability and performance management, security and compliance, and ongoing model oversight is where the real cost lives.

This acceleration can create a dangerous false sense of proximity to production-ready software. The ratio of prototype-to-production effort in enterprise software has historically been roughly 1:10. Vibe coding may compress the initial build phase, but the gap that remains is still enormous, and the 80% of costs that follow are largely unaffected by how quickly the first version was written.

Compounding this, McKinsey found that indirect costs and maintenance associated with developing a product can account for as much as 80% of full costs over the product's lifespan. For AI-powered systems specifically, Gartner projects that by 2030 the cost per resolution for GenAI will increase as firms realize the massive cost of deteriorating outputs — meaning the maintenance burden for AI builds grows over time rather than stabilizing. The prototype was the easy part.

3.2 Integration complexity unchanged

Modern PSA and Customer Success platforms connect to CRMs, ERP systems, financial platforms, billing engines, and data warehouses. These integrations require maintained API connectors, data mapping logic, error handling, and ongoing updates as external systems evolve. Vibe coding does not simplify this landscape. It can help write individual integration components faster, but the coordination and maintenance burden is unchanged.

3.3 Compliance remains non-negotiable

Regulatory and compliance requirements do not bend to development velocity. A custom-built system handling financial data, project revenue, or customer account information must meet the same standards as any commercial product. These requirements demand certification, documentation, and ongoing audit readiness that vibe coding cannot accelerate in any meaningful way.

3.4 Where vibe coding genuinely helps — for buyers

Interestingly, vibe coding creates more value for organizations that buy mature platforms than for those that build. Modern platforms like Certinia offer extensibility frameworks, APIs, and customization layers specifically designed to let customers adapt the system to their needs without touching core product code.

With AI coding tools, organizations can:

- Build custom reports and dashboards faster
- Develop lightweight integrations to niche internal systems
- Automate repetitive configuration tasks
- Extend standard workflows with business-specific logic

The result is the best of both worlds: a robust, maintained, compliance-ready foundation plus the agility of AI-accelerated customization on top.

4. The strategic case for investing in a purpose built platform

4.1 Time to value

Purpose-built PSA and Customer Success software can be deployed in weeks to months. A custom-built alternative, even with AI assistance, requires months to years before it reaches production-ready status across the full breadth of functionality. In professional services, where project profitability and client experience are already under pressure, this is a gap that directly affects revenue.

4.2 Network effects of a shared product roadmap

When Certinia serves hundreds of professional services organizations, every customer's feedback, every edge case discovered, and every best practice identified feeds back into the product. Customers benefit from a roadmap shaped by the collective intelligence of the industry. A custom-built system has a roadmap shaped by the priorities of one organization's IT backlog.

4.3 Risk transfer

Buying enterprise software transfers meaningful risk to the vendor. The vendor is responsible for uptime, security, data protection, regulatory compliance, and platform compatibility. A build decision retains all of these risks internally — and with AI-generated code, potentially introduces new risks around code quality, security vulnerabilities, and maintainability that are harder to audit than code written by experienced engineers.

4.4 Ecosystem and partner network

Platforms like Certinia sit within rich ecosystems: the Salesforce AppExchange, implementation partners, integration marketplaces, and community resources. This ecosystem accelerates deployment, reduces implementation risk, and provides access to pre-built capabilities that would take years to replicate in isolation.

4.5 Validated by real-world outcomes

The case for buying is not theoretical. Organizations across industries have made this shift and measured the results directly.

A leading global software quality and improvement firm previously operated on a fragmented build of five disconnected internal systems, forcing leadership to manage the business through spreadsheets. The data gaps and reporting inaccuracies that resulted were not a technology problem. They were a consequence of building rather than buying a platform designed for the work. After moving to a purpose-built PSA platform, the firm achieved a 70% increase in project margin accuracy and recovered thousands of hours previously lost to manual data reconciliation.

A Fortune 100 technology and networking company found that operating across disconnected legacy tools with no single source of truth created a massive administrative burden, leading to an estimated $100 million in lost productivity as teams manually reconciled data between systems. Unifying over 20,000 users onto a single standardized platform redirected engineering attention toward core innovation rather than internal data management.

A leading multinational financial institution made the decision to move away from custom-coded architectures after recognizing that the operational friction of maintaining bespoke systems was actively hindering its ability to launch and scale new development programs. Standardizing on an enterprise platform freed its engineering organization to focus on serving tens of millions of customers rather than maintaining back-office infrastructure.

Each of these cases reflects the same underlying pattern: the cost of the build was not just financial, it was the opportunity cost of talent, attention, and strategic focus consumed by systems that were not core to the business.

4b. Where vibe coding belongs in an enterprise software strategy

The most productive use of AI-assisted development is not replacing packaged software it is extending it. Purpose-built platforms like Certinia are designed with extensibility in mind: open APIs, metadata-driven configuration layers, workflow automation frameworks, and scripting environments specifically intended for customer-specific logic. Vibe coding amplifies the value of these extensibility layers dramatically.

Configuration and no-code customization

Modern enterprise platforms expose rich configuration surfaces — custom fields, page layouts, workflow rules, validation logic, and approval processes — that have historically required dedicated administrators to manage. With AI coding tools, business analysts and power users can now describe desired behavior in plain language and generate the configuration or formula logic needed to achieve it. This is perhaps the safest and highest-return application of vibe coding in an enterprise context: the underlying data model and business logic remain governed by the platform, while customization becomes dramatically faster.

Custom reports, dashboards, and analytics

Every organization has reporting needs that go beyond standard out-of-the-box content. Building custom reports, calculated metrics, and executive dashboards has traditionally required either a skilled BI developer or a significant time investment from an administrator. Vibe coding makes this accessible to a much wider group. Describing a desired visualization or metric in natural language and generating the underlying query, formula, or dashboard configuration is a low-risk, high-value application. The data lives in a governed system, and the output is easily reviewed and validated.

Workflow automation and business process extensions

Enterprise platforms typically include automation frameworks — flow builders, process automation tools, scripted actions — that allow customers to encode business-specific logic without modifying core product code. These are ideal targets for vibe coding.

Examples in the PSA and Customer Success context include:

- Automating project status notifications based on utilization thresholds
- Triggering customer health score alerts when engagement metrics drop
- Generating draft project plans or resourcing proposals from historical templates
- Routing approval workflows based on project type, region, or contract value

Because these automations operate within the guardrails of the platform's data model and permission framework, AI-generated logic is easier to review, test, and maintain than equivalent custom-built application code.

Point integrations to niche internal systems

Most organizations have at least a few internal systems — a legacy HR platform, a proprietary billing engine, an internal knowledge base — that are not covered by the platform's standard integration library. These point integrations are a natural fit for vibe coding. The scope is bounded, the data flows are well-defined, and the maintenance surface is limited. Using AI tools to generate the connector code, transformation logic, and error handling for a specific integration accelerates delivery without taking on the risk of building a full application.

What vibe coding should not touch

The corollary to identifying where AI-assisted development belongs is being equally clear about where it does not.

Core data model changes should be approached with caution. Modifying the fundamental structure of how projects, resources, contracts, or customer accounts are represented can have cascading effects on reporting, integrations, and future upgrades. These decisions require platform expertise and a clear architectural rationale — not just speed.

Security and permission logic should never be AI-generated without rigorous expert review. Access control, data visibility rules, and field-level security in a multi-tenant or multi-region deployment are areas where a subtle error can expose sensitive financial or customer data. This is not a domain where development velocity should be prioritized over precision.

Revenue recognition and financial calculation logic requires domain expertise and compliance review that goes beyond what any AI tool can provide. In PSA environments, incorrect financial logic can result in material misstatements. Changes to these areas should follow formal change control processes regardless of how they are developed.

Upgrade-sensitive customizations — code that directly interfaces with core platform APIs or internal data structures — should be limited and well-documented. Vibe coding tends to produce working code quickly, but it does not always produce code that is easy to maintain or upgrade-proof. In a packaged software environment where the vendor releases updates multiple times per year, customizations that are tightly coupled to internal implementation details create compounding risk over time.

The discipline is not to avoid vibe coding in an enterprise context, it is to apply it where the blast radius of a mistake is small, the output is reviewable, and the value of speed is high.

5. Decision framework for technology leaders

The following framework helps technology leaders evaluate whether a build, buy, or hybrid approach is appropriate for any capability under consideration.

Buy when:

- The capability is not a source of competitive differentiation
- The domain requires deep, accumulated industry knowledge
- Regulatory and compliance requirements are significant
- The capability needs to scale with the business without proportional engineering investment
- Time to value is a priority

Build when:

- The capability is genuinely proprietary and represents a core competitive advantage
- No commercial solution exists that addresses the specific need
- The organization has deep domain expertise in the capability being built
- The ongoing maintenance commitment is understood and resourced

Extend (buy + vibe code) when:

- A commercial platform provides 80–90% of the needed functionality
- Specific customizations or integrations require code that the vendor's standard product does not support
- AI-assisted development can dramatically accelerate the customization work within the vendor's extensibility framework

For PSA and Customer Success software, the vast majority of professional services organizations fall into the "buy" or "extend" categories. The domain knowledge embedded in purpose-built platforms like Certinia, combined with the ongoing investment required to maintain and evolve enterprise software, means that building from scratch — even with AI assistance — is rarely the right strategic choice.

6. Certinia's perspective

Certinia has spent years building a platform that encodes the best practices of professional services delivery and customer success management.

That platform reflects:

- Deep integration with Salesforce, enabling a single data model across the customer lifecycle
- Revenue recognition logic built to ASC 606 and IFRS 15 standards
- Resource management capabilities refined through thousands of customer deployments
- Real-time project financials that connect delivery performance to business outcomes

The emergence of vibe coding does not diminish this investment — it amplifies its value. As AI tools make software development faster, the differentiator shifts from "who can write code" to "who has the domain knowledge, the proven platform, and the ecosystem to deliver outcomes." On all three dimensions, purpose-built software wins.

We encourage technology leaders evaluating PSA and Customer Success platforms to ask not "could we build this with AI?" but "what would we have to stop doing to maintain it — and is that a trade worth making?"

Conclusion

Vibe coding is a genuine productivity innovation for software development. It lowers the barrier to building prototypes, accelerates customization, and democratizes some aspects of software creation. But it does not change the fundamental economics of owning and maintaining enterprise software. The domain knowledge gap, the compliance burden, the integration complexity, and the ongoing maintenance tax remain as real as ever.

For professional services organizations evaluating PSA and Customer Success platforms, the build vs. buy decision in the age of vibe coding reaches the same conclusion it always has — only faster. Buy a purpose-built platform. Use AI tools to extend and personalize it. Invest your development capacity where it creates genuine competitive advantage.

certinia

Certinia delivers Professional Services Automation and Customer Success Software on the Salesforce platform, connecting project delivery, resource management, and financial performance in a single system of record. To learn more about Certinia's platform, visit certinia.com.