

Buy-versus-Build in the Agentic Era

What the Rise of AI Builders Means for Your SPM Investment

A Varicent perspective for sales, revenue, finance, and HR compensation leaders navigating the generative AI moment in enterprise software.

Part One

A Perspective for Business Leaders

Generative AI has changed what is possible in software development. IT teams that once needed months to scaffold a working application can now prototype in days. English-language prompts produce functional code. The barrier to starting a build has never been lower – and that has reignited a conversation that enterprise software vendors and their customers have been having for decades: why buy when we can build?

We think this is a healthy question. Organizations should evaluate their software investments rigorously, and the emergence of powerful new tools is a legitimate reason to revisit assumptions. At Varicent, we are sharing this perspective across our customer base not to argue against that evaluation, but to ensure it is grounded in a complete picture of what the build path actually requires – not just at the start, but over the three-to-five year horizon that matters for enterprise software decisions.

The short version of our view: generative AI has meaningfully compressed the time required to build the first version of most software. It has not changed what it costs to sustain, govern, and scale software in a production environment. And for a category as operationally complex and compliance-critical as incentive compensation management, that distinction matters more than in almost any other domain.

What We Are Seeing Across the Market

IT teams at organizations of all sizes are exploring whether AI-assisted development tools – Cursor, Claude Code, Lovable, Replit, and others – change the calculus for building internal SPM systems. In most cases, the exploration begins with genuine enthusiasm and a working prototype that demonstrates real promise. The challenges emerge later, when the prototype meets the full operational requirements of a live compensation

program: retroactive calculations, multi-party credit assignment, audit trails, dispute resolution, and plan governance. These are not problems that AI tools solve. They are problems that require deep domain knowledge to even recognize.

What Has Actually Changed – and What Has Not

It is worth being precise about what generative AI has and has not changed in enterprise software development, because the conflation of the two is where most build-vs-buy evaluations go wrong.

What has genuinely changed: the time and cost required to build a working first version of most software applications has compressed dramatically. A skilled developer using modern AI tools can produce functional code faster than at any prior point in history. GitHub's controlled trials have shown productivity gains of up to 55 percent for AI-assisted development. For simple, well-defined applications, this changes the economics of building meaningfully.

What has not changed: the majority of the cost and risk in enterprise software occurs after the first version is built. Industry benchmarks consistently show that 60 to 80 percent of total software lifecycle costs are incurred post-launch – in maintenance, enhancements, integration upkeep, compliance updates, and operational support. (O'Reilly / ICEAA industry benchmarks.) AI tools accelerate the build. They do not reduce the cost of running what you built.

For software categories with low ongoing complexity – simple workflow tools, basic reporting applications, internal portals – this shift genuinely changes the calculus. But for categories characterized by deep integration requirements, strict audit obligations, and cascading retroactive calculations, the ongoing cost structure is the dominant factor. Incentive compensation is one of those categories.

The Build Resilience Landscape

Not all software is equally suited to internal development. The attributes that create genuine complexity for an internal build are: depth of cross-system integration, algorithmic and constraint complexity, auditability and compliance requirements, change-propagation and retroactive calculation requirements, and the cost of error when the system produces a wrong answer. Systems that score high across all five of these dimensions are the least suited to being built and maintained internally, even as development tools improve.

The following reflects observed build-vs-buy dynamics across enterprise software categories. A score of 5 indicates the highest complexity for a sustainable internal build; a score of 1 indicates categories where internal development is more straightforward.

Software Category	Build Complexity (1-5)	Key Factor
Generic CRM modules	1 – Low	CRUD-based; easily replicated
Contract lifecycle management	2 – Low-Moderate	AI accelerates drafting; governance hard
ITSM / Ticketing	2 – Low-Moderate	Composable workflows; scale favors platforms
Marketing automation	3 – Moderate	Campaign flows buildable; compliance harder
FP&A / Planning	3 – Moderate	Buildable internally; governance favors platforms
Pricing / CPQ	3-4 – Moderate-High	Complex rules and approval chains
Supply chain planning	3-4 – Moderate-High	Optimization and auditability demand platforms
Quota and territory management	3-4 – Moderate-High	CRM/ERP integration makes internal builds challenging
Payroll calculation	4-5 – Very High	Highly regulated; errors are catastrophic
Incentive Compensation / SPM	4-5 – Very High	Integration-heavy, retroactive, audit-critical

It is worth noting that payroll – the category most closely related to incentive compensation in terms of regulatory obligation and error consequence – scores identically. **Organizations that would not consider building their own payroll engine should apply the same logic here.** The two systems share deep structural similarities in their integration requirements, audit obligations, and consequences when something goes wrong.

SPM as a Strategic Instrument – and What That Demands of the System

Incentive compensation serves two distinct and equally important functions. The first is operational: calculate correctly, audit completely, pay on time – every cycle, without exception. The second is strategic: it is the primary mechanism by which leadership translates commercial intent into seller behavior across the field.

Sales, revenue, finance, and HR compensation leaders face a consistent challenge: how do you take a large, distributed group of professionals and align them rapidly around the priorities that matter most right now? And how do you do that while getting maximum return from every dollar of commercial investment?

The answer is a well-designed incentive system – but only if the platform that administers it can operate at the speed and flexibility the business requires. Leaders need a full range of instruments across different time horizons: annual plan structures that establish baseline expectations, quarterly accelerators that respond to business conditions, and short-duration contests and SPIFs that can be activated in weeks to close a gap or redirect effort almost immediately. When business strategy shifts – a new product launch, a pricing model change to usage or consumption-based, a territory realignment, a competitive threat – leadership needs to pull the right lever quickly and with confidence that it will translate into the right behavior in the field.

A compensation system that can support that flexibility is a strategic accelerator. One that requires months of development time to reflect a plan change is a constraint – the organizational equivalent of trying to turn a tanker when the moment calls for a speed boat.

This is the argument for a purpose-built platform that is rarely made explicitly, but that every business leader who has managed a live compensation program has felt. The question is not just whether a build can replicate what Varicent does today. It is whether a build can keep pace with what the business will need to do tomorrow.

The Domain Knowledge Question

A purpose-built SPM platform is not just software – it is accumulated domain knowledge about how incentive compensation actually works in practice: how plans are designed, how edge cases are handled, how conflicts between rules are resolved, how exceptions are processed, and how the system must evolve as the business changes. This knowledge is embedded in the platform itself, in the implementation methodology, and in the expertise of the teams that support it. A new build starts at zero. It must re-derive all of it – usually by encountering the hard problems firsthand, in production, during a live compensation cycle.

What the Next 12 to 36 Months Typically Look Like

Based on observed patterns across organizations that have attempted to build or rebuild SPM systems, internal build efforts tend to follow a predictable arc:

- **Months 0-6:** Rapid progress. A working prototype is built. Core calculation scenarios function correctly. Enthusiasm is high and the business case appears to be validating.
- **Months 6-12:** Integration complexity emerges. Connecting the build to CRM, ERP, HRIS, order management, accounts payable, and payroll systems proves more time-consuming than anticipated. Data quality issues surface. The first compensation cycle on the new system requires manual intervention to resolve discrepancies.
- **Months 12-18:** Operational debt accumulates. The first major plan change, territory realignment, or new product introduction requires significant development effort to implement. A retroactive adjustment reveals gaps in historical state management. The first audit request surfaces documentation that the system cannot produce.
- **Months 18-36:** The cost of ownership becomes visible. The engineering team that built the system is now primarily maintaining it. New feature requests require development cycles that delay plan changes. Dispute volume increases as payees lose confidence in calculation transparency.

This arc is not inevitable – but it is common enough to warrant honest examination before committing to the path. The organizations that navigate it successfully are typically those that went in with a clear-eyed view of the full scope required, not just the prototype.

Technical Due Diligence: Questions for IT Leaders and Architects

This section is intended for the IT leaders, architects, and implementation teams who will evaluate and scope an internal SPM build. The questions below are organized by domain area. They surface the categories of complexity that are most frequently underestimated – and that most commonly determine whether an internal build succeeds or fails at production scale.

For each area, the question is not whether you can build a solution. The question is whether the solution you build will remain maintainable, accurate, and compliant as the business evolves – and whether it can be built and transitioned to without disrupting a live compensation program in the process.

A Note on AI-Assisted Development

AI coding tools are genuinely powerful at generating boilerplate, scaffolding data models, and accelerating early development cycles. They are less effective at encoding domain knowledge – understanding, for example, how a retroactive transaction reversal should propagate through a multi-tier accelerator structure, or how date-effective hierarchy data must be maintained to support historical recalculation. The questions below are not about whether AI can write the code. They are about whether the right questions are being asked before the code is written.

1. Transaction Processing and Retroactive Calculation

The calculation engine is where most build enthusiasm begins – and where the first serious challenges emerge. The initial calculation is usually straightforward. The requirement to recalculate accurately when transactions change after the fact is where complexity compounds.

Question to Consider	Why It Matters
When a transaction is reversed or adjusted after compensation has been paid, how	A single reversal may affect not only the direct commission on that transaction, but also attainment thresholds, accelerator rates, and bonus qualifications

Question to Consider	Why It Matters
will your build determine the full scope of downstream recalculation required?	triggered by that transaction in prior periods. The recalculation scope is rarely limited to a single record – and a system that does not correctly identify the full scope will produce compensation errors that are difficult to detect and expensive to correct.
How will your build handle accelerator tiers triggered by cumulative attainment in a prior period, where a subsequent reversal would have prevented that tier from being reached?	If a representative reached an accelerator threshold based on transactions that are later reversed, all compensation paid at the accelerated rate – not just the reversed transaction itself – must be evaluated for adjustment. This requires complete attainment history to be maintained and re-evaluated as of each relevant period.
What is the projected processing time for a full retroactive recalculation across your payee population at peak transaction volume? Have you modeled this at 2x and 5x projected scale? And how will your build support incremental recalculation – processing only affected records – when a full recalculation is not feasible within a payroll processing window?	Retroactive calculations are computationally intensive. Systems that perform adequately at moderate scale frequently encounter performance degradation as transaction volumes grow. The ability to run targeted incremental recalculations is a material architectural requirement for large organizations – without it, a single reversal may require a full population recalc that cannot complete within a payroll cycle.
How will your build process multi-event compensable transactions – where a single deal generates compensation at booking, shipment, invoicing, cash receipt, consumption and usage thresholds, or other triggering events – while maintaining reconciliation across CRM, order management, accounts receivable, accounts payable, HRIS, and other related systems?	Many compensation plans pay on multiple triggering events tied to the same underlying opportunity, and the specific triggers vary significantly by industry, pricing model, and commercial structure – including usage-based and consumption-based arrangements that require continuous monitoring. As pricing models evolve, the compensation system must reflect those changes immediately. The list of source systems is rarely static, and the reconciliation architecture must be designed to accommodate new data sources as the business changes.

2. Credit Assignment and Date-Effective Hierarchy

Credit assignment – determining who receives compensation credit for a given transaction, and in what amount – requires the system to know the organizational state at the time the transaction occurred, not just today. Source systems like CRM are not date-effective: querying a record today returns its current state, not its historical state. The SPM build must maintain this data independently and continuously.

Question to Consider	Why It Matters
How will your build assign compensation credit when a single transaction generates entitlements for multiple individuals simultaneously – direct sellers, management roll-ups, overlay specialists, channel partners, agents, brokers, resellers, and indirect contributors such as BDRs and pre-sales consultants?	It is not uncommon for a single closed transaction to generate credit events for ten or more individuals, each governed by different plan rules, different rate structures, and different quota allocations. The build must evaluate each entitlement independently while maintaining consistency in the underlying transaction data.
Many source transactions – shipment records, AP payments, usage events, and others – do not carry the name of the responsible sales representative. Where will your build maintain the date-effective mapping between transactions and the individuals entitled to credit, and how will that mapping be updated as territories and assignments change?	The SPM build must serve as the authoritative, date-effective source of truth for crediting assignments – information that cannot be reliably sourced from CRM or ERP at query time. This mapping must be maintained continuously and versioned so that retroactive calculations reference the correct assignment as of any prior date.
How will your build handle credit assignment when the organizational hierarchy at the time of payout differs from the hierarchy at the time the original transaction closed?	When a transaction is reversed or corrected months after it closed, the build must apply compensation logic as it existed at the original transaction date. This requires complete, date-effective historical preservation of all hierarchy, territory, and assignment data – maintained within the build itself, not sourced from current system state.
When a transaction is reversed that was originally credited to a representative	Recovery of previously paid compensation must be attributed to the individual who received the original

Question to Consider	Why It Matters
who has since left or changed roles, how will your build process the recovery – and how will it ensure the recovery is applied to the correct individual rather than their current successor?	payment, regardless of who currently occupies that role or territory. Builds that do not maintain complete individual-level payment history cannot perform this calculation accurately.
How will your build handle ramp schedules for newly hired representatives – where quota credit and commission rates are governed by a separate schedule during an onboarding period – and how will it transition those representatives to standard plan terms automatically at the appropriate point?	Ramp schedules introduce a parallel set of plan rules that must be applied correctly for a defined period and then retired without manual intervention. The build must track each representative's start date and ramp status, apply the correct rate schedule for each calculation period, and maintain a complete audit record of the transition.

3. Quota Credit, Commission Credit, and Plan Complexity

Quota credit and commission credit are related but distinct concepts that are frequently conflated in initial build designs. A transaction's total value, its quota credit value, and its commission credit value may all be different figures, governed by different rules, and requiring different calculations.

Question to Consider	Why It Matters
How will your build determine which portion of a multi-product transaction is eligible for quota credit, and how will it allocate that credit across multiple quota components when a representative carries separate quotas for different product categories?	A transaction containing multiple product types may generate quota credit only for specific products, and that credit may need to be allocated across distinct quota buckets – each with its own attainment calculation, threshold structure, and commission rate schedule. The build must evaluate product eligibility and correctly update each quota component independently.
How will your build handle cases where the quota credit amount and the commission credit amount for a single	Compensation plans frequently specify that quota credit is awarded on one basis while commission is calculated on another. The build must maintain these as separate

Question to Consider	Why It Matters
transaction differ – because of product eligibility rules, contractual adjustments, or plan-specific crediting policies?	calculations while ensuring both remain traceable to the same underlying transaction.
How will your build accommodate plan designs developed by senior leadership that do not conform to standard templates – including novel accelerator structures, bespoke contest mechanics, custom SPIFs, and short-duration incentive programs that need to be deployed quickly and retired cleanly?	Compensation plan design is not a standardized discipline. Leaders regularly develop incentive structures that reflect strategic judgment, competitive intelligence, or specific behavioral objectives. A build that can only support predefined plan types will constrain the organization's ability to deploy the strategies leadership believes are most effective – and will require development cycles every time a new structure is needed.
How will your build support mid-year plan changes – including the ability to run parallel calculations under the old and new plan simultaneously during a transition period, with complete traceability to which rules governed each output?	Compensation plans change mid-fiscal year more often than most build evaluations account for. During the transition period, the build must calculate compensation under both the prior and new plan simultaneously within the same processing run – a requirement that is architecturally nontrivial and frequently underestimated.

4. Audit, Compliance, and Data Integrity

Incentive compensation is a financial system, subject to audit and regulatory requirements that are not optional and not easily retrofitted. Data integrity in this context means more than accuracy at a point in time – it means the ability to reconstruct any historical calculation, demonstrate the exact rules that governed it, and produce that evidence on demand.

Question to Consider	Why It Matters
How will your build produce a complete, transaction-level audit trail supporting SOX compliance – including the ability to demonstrate, for any payment made, the	SOX compliance requires that compensation calculations be independently verifiable and reproducible. The build must preserve not just outputs but the complete input state – transaction data, plan rules as they existed at

Question to Consider	Why It Matters
exact inputs, rules, and organizational state that produced it?	calculation time, and organizational hierarchy – so that any calculation can be reproduced identically at any future point.
How will your build support the seven-year data retention requirements applicable in many jurisdictions, including the ability to retrieve and reconstruct historical compensation records on demand?	Systems that do not plan for long-term retention from initial design frequently encounter architectural limitations that make compliance difficult or expensive to achieve retroactively. This is rarely addressed in initial build scopes and almost always surfaces as a significant gap during the first external audit.
How will your build support dispute resolution – providing payees with sufficient visibility into their calculation inputs and outputs to evaluate whether their compensation was correctly calculated, without requiring manual forensic investigation by the compensation team for every inquiry?	Compensation disputes are a routine operational reality in sales organizations and represent a significant administrative burden when not properly supported. A build whose calculation logic cannot be traced and explained at the transaction level is not auditable and cannot support a credible, scalable dispute resolution process.
How will your build enforce separation of duties between the individuals who configure compensation rules and the individuals who approve and release compensation calculations – and how will it maintain an audit record of who approved what and when?	SOX and internal audit requirements typically mandate separation between configuration and approval. The build must support a governed workflow that enforces this separation consistently – not as a manual control, but as a system-enforced constraint with complete documentation.
How will your build manage the data integrity requirements that arise when the same underlying transaction appears differently across CRM, ERP, HRIS, order management, and accounts payable – and how will it produce a consistent, auditable compensable value from potentially inconsistent source data?	Source system inconsistencies are a routine operational reality. The build must apply a defined, documented reconciliation logic that produces consistent outputs regardless of source system variations – and that logic must be auditable, meaning the reconciliation decision for any given transaction can be traced and explained.

5. Migration Risk and Transition Architecture

This section is specific to organizations evaluating a move away from an existing SPM platform. Unlike a greenfield build, a migration must be planned and executed while the current compensation program continues to operate. The transition architecture is where migration projects most commonly encounter their most significant and costly challenges.

Question to Consider	Why It Matters
How will your migration preserve the complete historical compensation record from your current environment – including the ability to reference prior period calculations during disputes, audits, and retroactive adjustments that arise after the cutover?	Retroactive adjustments and disputes do not stop at a migration cutover date. For months or years after the transition, the new system must be capable of referencing historical calculations made in the prior environment. If that history is not migrated or made accessible, the organization's ability to respond to audits and disputes is materially compromised.
How will your migration handle the transfer of active compensation plan logic – including all configured rules, eligibility conditions, crediting structures, and exception handling – from the current environment to the new build?	Plan logic is the institutional knowledge of a compensation program. It is rarely fully documented outside the system that executes it. A migration that relies on re-deriving plan logic from memory or incomplete documentation will almost certainly introduce calculation errors that are difficult to detect until they affect payroll.
What is your parallel-run strategy – the period during which both the existing system and the new build calculate compensation simultaneously so that outputs can be compared and discrepancies resolved before cutover?	Parallel runs are the primary mechanism for validating that a new build produces the same outputs as the existing system under real data conditions. They are time-consuming, resource-intensive, and frequently reveal edge cases that were not covered in the build specification. Skipping or shortening the parallel run period is one of the most common sources of post-cutover compensation errors.
How will your organization manage the compensation program operationally during the transition period – including plan changes, organizational restructurings, and mid-cycle	Compensation programs do not pause for migrations. Plan changes, headcount movements, and retroactive adjustments will continue to occur during the transition period. The migration plan must account for how these events will be handled in both systems, and how the new

Question to Consider	Why It Matters
adjustments that occur while both systems are active?	build will be kept current with changes made during the parallel-run period.

A Final Perspective

Varicent shares this analysis not to foreclose the build conversation, but to ensure it is a complete one. The organizations that make the best decisions on questions like this are the ones that evaluate the full scope – prototype to production, day one to year five – with the same rigor they apply to any significant capital investment.

If your evaluation concludes that a build is the right path, we believe that decision should be made with a clear view of what it requires. If it concludes that your current platform remains the stronger long-term investment, we are committed to continuing to earn that confidence through the quality of our product and our partnership.

Either way, the questions in this guide are worth answering before the decision is made – not after.

Varicent

Varicent provides industry-leading software for incentive compensation management and sales performance management. To continue this conversation or to discuss your organization's specific evaluation, please reach out to your Varicent account team.