experlogix

Build Vs. Buy: Document Automation

A decision guide

Contents

Executive Summary

Document automation is often viewed as a straightforward technology initiative: merge data into a template, generate a document, and deliver it to a customer or stakeholder. In reality, enterprise document automation is far more complex. Behind every policy, loan package, contract, disclosure, invoice, or customer communication lies a network of integrations, business rules, approval workflows, compliance controls, security requirements, and ongoing maintenance obligations.

As document volumes increase and customer expectations evolve, document automation has shifted from an operational tool to a strategic business capability. Organizations today must deliver accurate, personalized, and compliant communications quickly while supporting digital experiences, regulatory requirements, and growing data ecosystems.

This creates a critical decision for business and technology leaders: Should document automation be built internally or purchased as a dedicated platform?

At first glance, building may appear less expensive. Many organizations already have development teams, existing infrastructure, and proprietary business processes that seem well suited for a custom solution. However, what begins as a document generation project frequently expands into the long-term ownership of a complex software platform requiring continuous investment in maintenance, security, integrations, compliance updates, scalability, and support.

The decision is no longer simply a technical one. It is a business decision that affects:

- Total cost of ownership
- Speed to market
- Operational agility
- Regulatory risk
- Customer experience
- Engineering productivity
- Long-term scalability

Buying is usually favored when document processes are high-volume, regulated, integrated with multiple systems, frequently changing, or expected to be managed by business users. Building can be rational when document generation is itself core intellectual property, the scope is narrow and stable, and the organization is prepared to staff security, support, product management, and lifecycle engineering for years. A hybrid model can make sense when a commercial platform supplies the document engine and governance layer while internal teams retain proprietary orchestration, domain logic, and customer experience components.

For organizations evaluating document automation investments, the most important question is not whether developers can build a solution. Most can. The more important question is whether owning and maintaining a document automation platform creates a competitive advantage that justifies the ongoing cost, complexity, and risk.

This white paper provides a practical framework for evaluating both approaches. Readers will learn:

- When building document automation makes strategic sense
- The hidden costs and risks that emerge over time
- Why enterprise document automation is more complex than many organizations anticipate
- The advantages and limitations of commercial document automation platforms
- A structured framework for determining whether building, buying, or adopting a hybrid approach is the right decision

For most organizations, the optimal choice is not the one with the lowest initial cost. It is the one that delivers the greatest business value, operational resilience, and strategic flexibility over the lifecycle of the solution.

1. Document Automation Is Business-Critical

Documents are the transaction layer of the enterprise

Contracts, policies, disclosures, quotes, invoices, notices, case correspondence, and regulatory filings translate system data and business decisions into records that customers, employees, partners, auditors, and regulators can use. When these outputs are slow, inaccurate, inaccessible, or inconsistent, the impact reaches revenue, customer experience, compliance, and operational capacity.

The pressure is greatest in high-volume, highly regulated industries. An insurer issuing policies, endorsements, and claims correspondence, or a lender producing loan packages, disclosures, and adverse-action notices, is not generating the occasional document. It produces thousands or millions of legally consequential records, each assembled from multiple systems, governed by jurisdiction-specific rules, and subject to audit. In these environments a document is not the end of a process; it is the transaction itself.

Expectations have moved beyond document generation

Document automation is now measured against a broader set of business expectations. Organizations increasingly require:

- Faster turnaround without proportional headcount growth
- Personalized, data-driven communications across channels
- Consistent rules and approved language across document types
- Traceable approvals, versions, signatures, and delivery events
- Enterprise integration through APIs, connectors, and orchestration tools
- A foundation that can adopt AI-assisted authoring and forms without rebuilding the core process

Adobe's 2026 AI and Digital Trends research surveyed 3,000 executives and practitioners and 4,000 customers. Among organizations' expectations for future customer experiences, **80%** anticipated highly personalized, real-time experiences and **72%** anticipated seamless experiences across digital and physical touchpoints [1].

For a policyholder receiving a renewal or a borrower reviewing closing documents, that experience is the document workflow where customer experience is won or lost.

One document, many moving parts

Consider what happens when a single document is produced. What looks like one output is actually a 7 step process or more, each drawing on different systems. The table below follows one document from trigger to archive. It shows why a capability that appears simple on the surface becomes an operating platform underneath, a theme this paper returns to when it examines true complexity and cost.

	What Happens	Systems involved	Possible issues
1. Trigger	An event initiates the document (a transaction, approval, or request)	• CRM, core system, portals, email, and connected apps (API/webhook)	• Stalls, disconnected systems, missing information, Field-mapping errors
2. Assembly	Data is merged into the correct template	• ERP, product/records systems, content manager	• Field mapping errors, wrong template version, incomplete data
3. Rules	Conditional content, calculations, and jurisdiction logic are applied	• Rules engine, data configuration	• Outdated rules, unhandled edge cases, calculation errors
4. Approval	Content is reviewed, routed, and exceptions are handled	• Workflow automation, Identity and permissions	• Bottlenecks, misrouting, lost or incomplete audit trail
5. eSignature	An electronic signature is captured where required	• eSignature connections	• Failed verification, delivery delays
6. Delivery	The document is delivered across the right channels	• Email, print, portal, API	• Format or channel failures, bounces, delivery gaps
7. Archive	The record is retained, secured, and made retrievable	• Content manager, records systems	• Retention gaps, compliance exposure, retrieval failures, searchable

Every stage above is a potential point of failure, a maintenance obligation, and a compliance checkpoint. Multiply this chain across dozens of document types and jurisdictions, and the reason document automation is treated as a strategic capability, rather than a simple utility, becomes clear.

2. Defining Build, Buy, and Hybrid

Before comparing approaches, it helps to see what is riding on the decision. How an organization chooses to deliver document automation shapes cost, risk, speed, customer experience, and its ability to adapt as requirements change. Before weighing the options, consider what is at stake.

	If the decision is right	If the Decision is Wrong
Growth	Document processes scale without interruption, additional cost or added workforce	Backlogs, delays, slow product launches, repeated disruptions and stalled expansion
Risk	Rules, access, and audit evidence are consistently governed and defensible	Inconsistent outputs, weak traceability, and mounting exposure in regulated filings
Cost	Costs stay visible, predictable, and aligned to actual usage and value	Maintenance, rework, and opportunity costs accumulate over time
Customer experience	Customers receive accurate, timely, and consistent communications	Errors and delays erode trust at the moment it matters most
Innovation	Teams add channels, AI, and integrations without re-architecting	Aging architecture constrains new products, markets, and speed to launch

Build

A build strategy means the organization owns the code, architecture, testing, hosting, monitoring, security, release management, documentation, support, and roadmap. It may begin as scripts, internal services, open-source libraries, or template merging functions inside a line-of-business application. The defining feature is not where the code runs; it is who carries lifecycle accountability.

Buy

A buy strategy uses a vendor-managed or commercially supported platform. Configuration, templates, rules, and integrations may still require substantial work, but the reusable engine, product roadmap, quality controls, and support model are shared across customers rather than funded by one organization alone.

Hybrid

Hybrid models separate commodity capability from differentiated capability. A commercial platform can handle authoring, rendering, versioning, distribution, and audit functions, while internal teams build proprietary services, workflows, or interfaces. For example, a lender might rely on a platform to generate, render, and archive loan documents while keeping its proprietary underwriting and eligibility logic in-house. Hybrid is often the most practical answer when differentiation sits in business logic or customer experience rather than document rendering itself.

Approach	Organization owns	Best fit
Build	Full product and lifecycle	Narrow, stable use case or core IP with durable engineering capacity
Buy	Configuration, adoption, vendor governance	Complex, regulated, changing, or multi-system document operations
Hybrid	Differentiated layers plus integration	Unique workflows with a need for proven document infrastructure

3. Why Organizations Build

The instinct to build is rational and often well-founded. Organizations that choose to build usually do so for sound reasons: full control over features and roadmap, a tight fit with existing architecture, freedom from vendor dependency, and confidence in their own engineering talent. Deep integration with a proprietary application ecosystem is often part of that calculus too. When a homegrown document engine can be wired directly into existing internal systems, custom development can feel like the path of least resistance compared to introducing a new integration layer. When document generation touches proprietary logic or differentiated customer experience, keeping it in-house becomes not just reasonable but necessary.

There is also a cost assumption at work. Because most organizations already employ developers and own infrastructure, building can appear "free". A matter of redirecting existing capacity rather than signing a new contract. In many cases, an initial prototype genuinely is quick to stand up. A developer can merge data into a template and produce a working document in days.

The risk is not the rationale. It is allowing a prototype-level estimate to stand in for a platform-level commitment. What starts as a weekend script becomes a system that must be secured, tested, integrated, documented, supported, and continuously updated, often for years. That gap between the demo and the durable platform is where build costs quietly concentrate.

McKinsey has found that CIOs estimate tech debt at **20 to 40 percent** of their entire technology estate's value, and that **10 to 20 percent** of the technology budget for new products is diverted to resolving it [2].

Custom-built infrastructure is a common contributor to that debt.

When building genuinely makes sense

Building is the right call in specific, identifiable conditions. It is worth being honest about them, because they define where a custom solution creates real advantage:

- ✓ Few document types with limited conditional content
- ✓ Stable requirements and predictable data sources
- ✓ No near-term need for business users to own template changes
- ✓ A dedicated, funded team with product and support responsibilities
- ✓ Document generation or rendering is itself defensible intellectual property
- ✓ Security, compliance, accessibility, and continuity obligations are planned and staffed

If most of these hold, building can deliver a tailored solution that a commercial platform cannot match. If they do not, the same decision quietly transfers long-term risk onto the engineering organization.

Tests that strengthen the build case

Before committing, pressure-test the decision against five questions. Each converts an assumption into evidence:

Test	Evidence Required
Strategic differentiation	The capability changes the product or business model, not just how something is implemented
Capacity	Named, funded roles exist for architecture, QA, security, support, and product ownership
Lifecycle funding	Budget covers enhancements and operations for years, not just initial delivery
Requirements stability	Document types, jurisdictions, channels, and integrations are unlikely to expand rapidly
Exit resilience	Documentation, source control, and succession plans reduce key-person dependency

If the build case survives all five, it is likely a sound decision. If it fails even one, most often lifecycle funding or exit resilience — the organization is probably underestimating what it is taking on, a gap the next section examines in detail.

4. The Real Complexity of Enterprise Document Automation

Section 1 followed a single document from trigger to archive. This section widens the lens from one document to the platform that produces every document. Building document automation in-house does not mean building a feature; it means owning and maintaining an interlocking system of capabilities, each with its own requirements, failure modes, and ongoing engineering burden. Weakness in any one layer can compromise the accuracy, timeliness, or defensibility of every document the platform produces.

What you see is not what you build

Document automation is deceptive because of the part everyone sees. A clean, branded document coming out the other end is the smallest part of the work. Like an iceberg, the visible output rests on a far larger mass of engineering beneath the surface. A stakeholder reviewing a finished policy or loan package sees a template that "just works." What they do not see is the data plumbing, rule logic, versioning, testing, security, and operational scaffolding required to make that single output correct, compliant, and repeatable at scale.

Above the waterline (what stakeholders see):

- The finished document and its formatting
- The template a business user edits
- The button that generates the output

Below the waterline (what engineering must build and maintain):

- Data integration across CRM, ERP, and legacy systems, with authentication, mapping, and reconciliation
- Rules engines for conditional content, calculations, and jurisdiction logic — each versioned and tested
- Workflow orchestration for approvals, routing, exceptions, and recovery
- Rendering fidelity across PDF, print, email, and portals, including fonts, pagination, and localization
- Governance: permissions, audit trails, separation of duties, and retention Operations: monitoring, patching, performance tuning, incident response, and SLAs

The template above the waterline may represent a tenth of the effort. The nine-tenths below it is where cost, risk, and maintenance actually live, and it never stops requiring attention.

Every capability is its own engineering commitment

Each layer below the surface is not a one-time build. It is a system that must be maintained, secured, and evolved as data sources, regulations, formats, and integrations change.

Capability layer	Typical requirements	Hidden engineering work
Data	CRM, ERP, policy, loan, claims, case, and legacy sources	Authentication, mapping, retries, schema changes, reconciliation
Authoring	Reusable templates, approved clauses, branding	Editor behavior, pagination, tables, images, accessibility
Rules	Conditional content, calculations, jurisdiction logic	Rule versioning, test coverage, explainability
Workflow	Approval, routing, exception handling	State management, escalations, idempotency, recovery
Output	PDF, Word, HTML, print, email, portals	Rendering fidelity, fonts, localization, channel constraints
Signature & archive	e-signature, retention, records	PDF, Word, HTML, print, email, portals
Governance	Permissions, versions, audit trails	Role models, separation of duties, reporting
Operations	Monitoring, performance, support	Capacity planning, patching, incident response, SLAs

Security is not a feature you add; it is a discipline you inherit. NIST's Secure Software Development Framework requires secure practices across the entire lifecycle — preparation, software protection, secure production, and vulnerability response [3].

Building a document platform means owning all of it, as a recurring obligation rather than a one-time task.

Complexity looks different in every industry

The same structural forces — high volume, multi-system data assembly, regulatory variation, privacy obligations, and audit pressure — appear in every regulated sector. Only the disguise changes.

Industry	Representative documents	Complexity pattern	Decision implication
Insurance	Quotes, policies, endorsements, claims letters	State/jurisdiction variation; frequent product changes	Favor governed templates, reusable rules, and audit history
Financial services / fintech	Loan packages, disclosures, KYC notices, statements	Data from core, CRM, risk, and identity systems	Prioritize integration, security, consent, and traceability
Healthcare	Patient letters, explanations, care documents	Sensitive data, accessibility, language needs	Prioritize privacy, permissions, localization, and fidelity
Government	Notices, permits, case correspondence	Scale, records retention, accessibility, procurement	Prioritize standards, auditability, and long-term support
Legal and professional services	Engagements, contracts, reports	Clause governance and client-specific variation	Prioritize authoring control and approved content

"The regulatory burden behind this complexity is intensifying. PwC's Global Compliance Survey 2025, based on responses from 1,802 executives across 63 territories, found that **77% of companies** said compliance complexity had negatively affected several areas that drive growth. For document-heavy operations, that pressure lands directly on the templates, rules, and audit trails a document platform must manage [4]."

5. The True Costs of Building

The cost of building rarely arrives as a single number. It accumulates in layers, some visible at kickoff, most surfacing only after the system is in production. A build estimate that captures only the first layer will understate the true commitment by a wide margin. The sections below separate those layers so each can be weighed honestly: what you pay to build it, what you pay to keep it running, what you risk if it fails, and what you forgo by building it at all.

Initial delivery is only the first cost layer

This is the layer most estimates capture, and the only one many capture at all. It covers everything required to reach a first working version:

- Discovery, architecture, UX, data mapping, and integration
- Template engine or editor development
- Rendering, rules, workflow, and delivery services
- Automated testing across data permutations and output formats
- Security review, accessibility validation, documentation, and training

Recurring costs become the economic center of gravity

Maintenance includes dependency upgrades, regression testing, browser and operating-system changes, performance tuning, incident response, new channels, localization, accessibility remediation, regulatory updates, and integrations that evolve.

GitLab's 2025 survey of 3,266 DevSecOps professionals found respondents lost seven hours per week to inefficient processes; **60%** used more than five software-development tools, highlighting the coordination cost created by fragmented delivery environments [5].

Security and resilience carry material downside

IBM's 2026 Cost of a Data Breach report placed the global average breach cost at $4.99 million and the U.S. average at $11.5 million. It also found organizations using AI and automation extensively in security operations reported $1.93 million lower breach costs than those using none [6]. These figures do not estimate document-platform risk directly, but they illustrate why security engineering, monitoring, access control, and rapid remediation belong in the total cost model.

Opportunity cost is real even when no invoice is issued

Every sprint devoted to document infrastructure competes with customer-facing features, data products, integrations, and revenue initiatives. Atlassian's 2025 research found that **90% of developers** lose six or more hours per week to organizational inefficiency, with more than half losing ten or more [7]. A build estimate should therefore value internal labor at fully loaded cost and identify the roadmap items displaced by the project.

These four layers rarely appear on the same page, which is precisely the problem. Seen together, they reveal that the true cost of building is dominated not by the initial build, but by everything that follows it. The next section brings these layers into a single view and sets them side by side with the cost of buying.

6. The Case for Buying

If building carries a familiar logic, buying carries a different one. It is grounded less in confidence and more in focus. Organizations that choose to buy are usually making a bet that document automation, however important, is not where their engineering talent creates the most value. That distinction matters more than it might first appear. A purpose-built platform provides the capabilities needed to automate complex document processes without requiring organizations to develop and maintain the technology themselves.

Speed to deployment is often the first thing that draws teams toward a platform. A configured, production-ready system can go live in weeks rather than the months or years a custom build typically requires. For organizations under competitive or regulatory pressure, that compressed timeline alone can outweigh the appeal of full customization.

Built-in compliance and security certifications remove a burden that's easy to underestimate until it lands on an internal team. SOC 2, HIPAA, GDPR, and industry-specific requirements are maintained by the vendor as a cost of doing business.

Support, SLAs, and reliability shift accountability for uptime and issue resolution to a vendor whose business depends on getting it right, rather than to whichever engineer is on call.

A predictable cost structure replaces the volatility of custom development with a known, budgetable line item. For finance leaders trying to forecast technology spend rather than absorb it as it happens, that predictability is a meaningful advantage.

Empowering business users may be the most underappreciated benefit. Purpose-built platforms are designed so that template and workflow changes can be made by the people who understand the business process, without routing every change through IT. That doesn't just accelerate iteration; it removes a steady stream of small requests from engineering's backlog entirely.

Technology and business requirements continue to evolve, particularly as organizations incorporate AI into their workflows. A flexible document automation platform should be able to adopt new capabilities without requiring the organization to rebuild its underlying infrastructure.

IBM research found that **77%** of surveyed executives said they need to adopt generative AI quickly to keep pace with competitors, while only **25%** strongly agreed their existing IT infrastructure could support scaling AI across the enterprise [7].

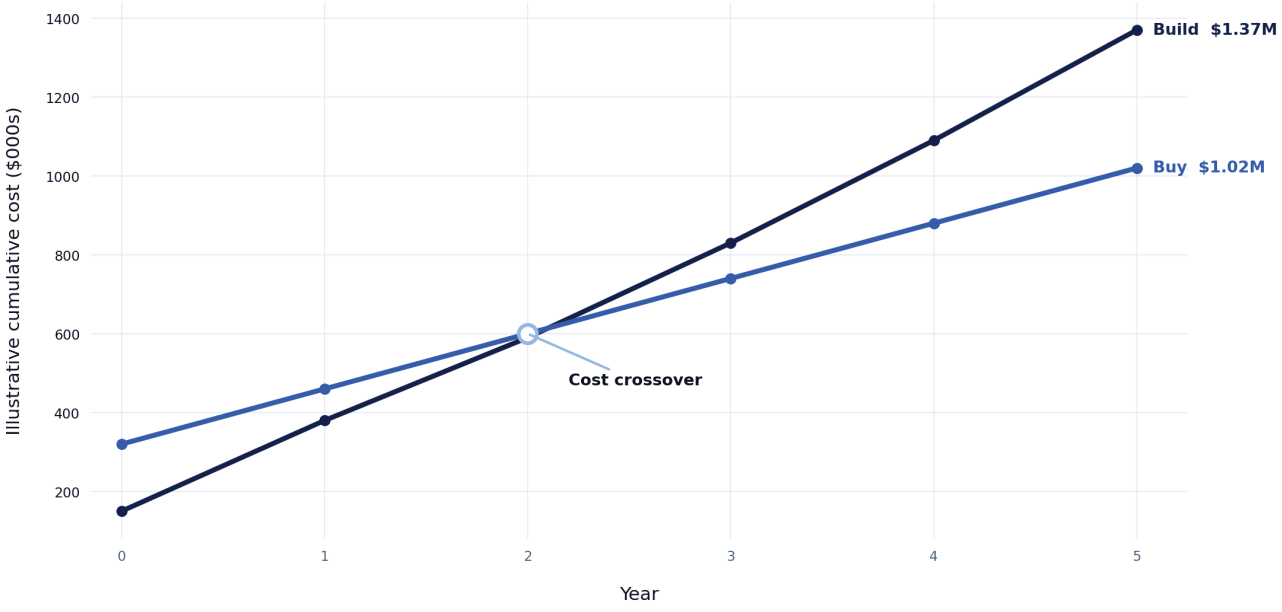
Still, the case for buying is not automatic. It rests on a cost comparison that holds up over years, not months, which is where a total cost of ownership view becomes essential.

7. Total Cost of Ownership Comparison

A five-year model is more decision-useful than an initial quote

Most build-versus-buy decisions are made on the wrong number. Build proposals are typically judged on the cost to reach a first working version, while the expenses that dominate the lifecycle (maintenance, security, change, and opportunity cost) arrive quietly over the years that follow. A one-year view flatters building. A five-year view tells the truth. Total cost of ownership reframes the question from "what will it cost to launch?" to "what will it cost to own?"

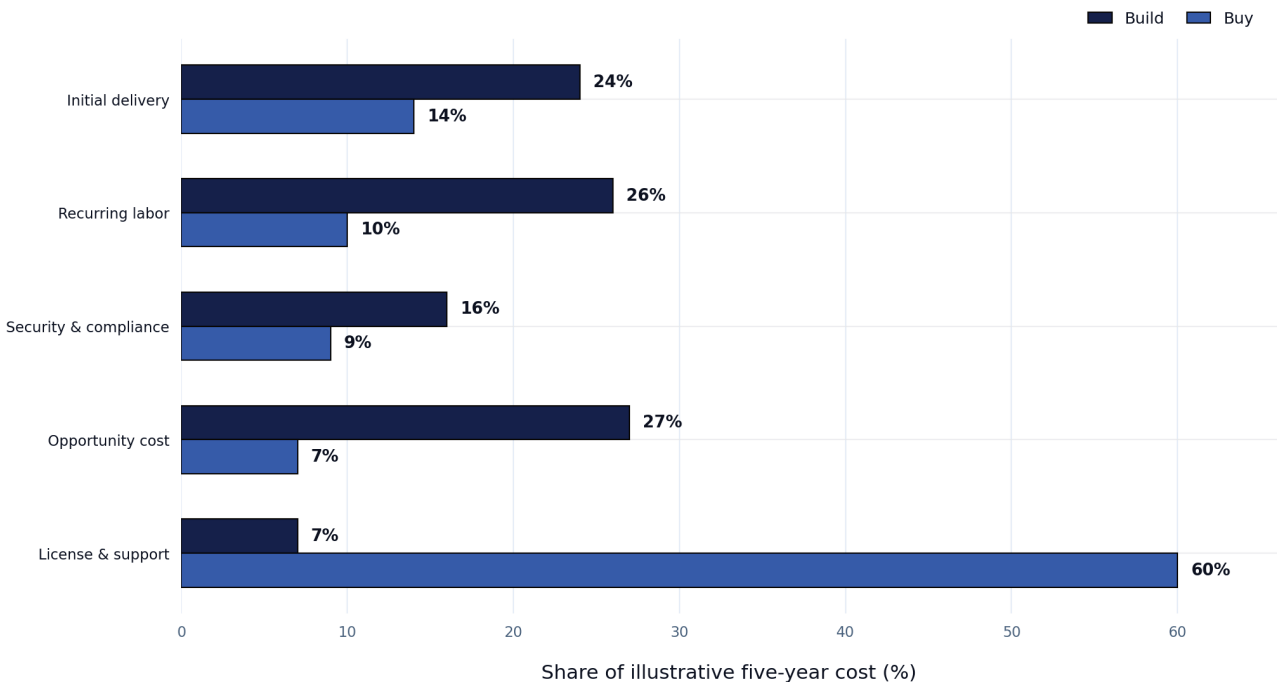
The following scenario is illustrative, not a benchmark. It assumes a mid-market or enterprise program with multiple document types, several integrations, regulated data, business-user authoring, and production support. Organizations should replace every input with validated internal and vendor figures.



[Figure 1 — Illustrative cumulative TCO, Build vs. Buy over 5 years]
Illustrative cumulative TCO. Values are scenario assumptions, not market averages.

The shape of the two lines is the point. Build starts lower but climbs steadily as maintenance, compliance updates, and displaced engineering capacity accumulate year over year. Buy carries a higher initial cost but flattens as the vendor absorbs upkeep and improvement. In this illustrative scenario, the cumulative cost of buying falls below building within the first few years, and the gap continues to widen after that.

[Figure 2 — Illustrative cost mix, Build vs. Buy by category]
Illustrative cost mix shows how build costs often concentrate in labor and opportunity cost.



The cost mix explains why. For building, the largest shares are not the initial engineering everyone budgets for, but the recurring labor and opportunity cost that rarely appear in the original estimate. For buying, cost concentrates in a single predictable line: the subscription. Predictability, not just total size, is what makes the buy profile easier to plan around.

Sensitivity matters more than false precision

No single model settles the decision. Test at least three scenarios: expected, high-growth, and disruption. Vary document volume, template count, jurisdictions, integration count, release frequency, staff turnover, outage cost, vendor pricing, and migration effort. Present the break-even point only after these variables have been stress-tested, and weight the result against the risk factors that a cost model alone cannot capture.

8. Decision Framework and Scorecard

Build vs. Buy Scorecard

Step 1 — Score eight questions on a 1 to 5 scale. A score of 1 points toward Build, 5 points toward Buy. Scores of 2 to 4 reflect partial or mixed alignment.

Question	1: Points to Build	5: Points to Buy
1. How much document volume do you have, and how fast is it growing?	Low, stable volume	High or rapidly scaling volume
2. How many document types, formats, and templates? How many steps make up one workflow? How many workflows?	Simple, few formats	Complex, many formats and exceptions
3. How soon do we need this live?	No urgency (12+ months)	Urgent (need it in weeks)
4. Capital for upfront dev, or predictable opex?	Capital available, opex-averse	Prefer predictable subscription cost
5. Engineers to build and maintain long-term?	Strong, stable team with bandwidth	Limited capacity or turnover risk
6. How much burden (SOC 2, HIPAA, GDPR)? Requirements?	Minimal requirements	Heavy, evolving requirements
7. How important is adopting AI and new capabilities without rebuilding?	Static needs; little near-term change	Rapidly evolving; AI adoption is a priority
8. Who needs to make template and workflow changes?	IT owns changes, and that is acceptable	Business users must change content without code

Step 2 - Total the score (8 to 40 range).

8 to 18: Build.
You have the internal capacity, low complexity, and time to justify owning it.

19 to 29: Hybrid.
Consider buying a platform and customizing or integrating around it.

30 to 40: Buy.
Complexity, urgency, risk, or pace of change make building a poor use of resources.

The framework applied: a worked business case

To see the scorecard in action, consider a composite but realistic scenario.

The situation. A regulated financial-services provider generates loan packages, disclosures, adverse-action notices, statements, and servicing correspondence from its CRM, core lending, identity, and risk systems. Its homegrown service began with four templates. Over time it expanded to more than 80 variants across products and jurisdictions. Today, every template change requires a developer ticket, regression testing is mostly manual, and two engineers hold most of the platform knowledge.

The options. Leadership weighed three paths:

Option	Expected outcome	Primary concern
Continue building	Maximum code-level control	Maintenance backlog and key-person dependency grow
Buy a platform	Faster business ownership and standardized governance	Migration, vendor fit, and recurring fees
Hybrid	Keep proprietary eligibility and orchestration; move authoring, rendering, and governance	Clear boundaries and integration design required

The score. Rating the eight questions against its situation (high volume, many jurisdictions, a tight timeline, constrained engineering bandwidth, heavy compliance, and rising AI demand), the provider scores 36 of 40, landing firmly in the Buy range.

The decision. Rather than a pure buy, leadership selects a hybrid implementation. The commercial platform manages templates, rules, rendering, approvals, and audit evidence, while internal services retain the proprietary underwriting and customer-experience logic that genuinely differentiates the business. To hold the decision accountable, the team tracks five metrics: time to publish a template change, developer hours per release, defect escape rate, document turnaround, and cost per thousand documents. No outcome is claimed until it is measured in production.

Decision guidance.

9. Recommendations and Next Steps

Whichever direction the analysis points, these seven steps turn the decision into a disciplined process rather than a gut call.

1. **Reframe the decision.** Define the business capability, service levels, risk appetite, and ownership model before evaluating architecture.
2. **Baseline the current state.** Measure volumes, document types, change frequency, defects, cycle time, developer effort, incidents, and compliance effort.
3. **Model three to five years.** Include fully loaded labor, security, infrastructure, support, audit, migration, opportunity cost, vendor services, and exit costs.
4. **Run the weighted scorecard.** Document assumptions and require business, finance, security, architecture, operations, and compliance participation.
5. **Validate through a representative proof of value.** Test the hardest template, the highest-risk rules, real integrations, role controls, performance, and operational handoffs.
6. **Negotiate for lifecycle clarity.** For buy, define SLAs, pricing mechanics, security evidence, roadmap, data portability, and exit rights. For build, define funding, staffing, support, and succession.
7. **Measure realized value.** Track business outcomes after launch and revisit the decision as volume, regulation, and strategy change.

For most organizations, the deciding factor is not whether developers can build document generation. They can. The deciding factor is whether owning a permanent document-automation product is the best use of capital, talent, and risk capacity. Choose the model that creates the strongest business outcome over the full lifecycle, not the one that appears cheapest at kickoff.

See what the right approach looks like for your organization. Schedule a meeting with one of our experts to see how we can help you automate.

Contact us

Sources:

[1] Adobe, AI and Digital Trends 2026 Report, 2026. https://business.adobe.com/resources/digital-trends-report.html (Section 1)

[2] McKinsey & Company, Tech debt: Reclaiming tech equity, 2020. https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/tech-debt-reclaiming-tech-equity (Section 3)

[3] National Institute of Standards and Technology, SP 800-218, Secure Software Development Framework v1.1, https://csrc.nist.gov/Projects/ssdf (Section 4)

[4] PwC, Global Compliance Survey 2025, survey of 1,802 executives across 63 territories. https://www.pwc.com/gx/en/issues/risk-regulation/pwc-global-compliance-study-2025.pdf (Section 4)

[5] GitLab, Global DevSecOps Report, 2025. https://about.gitlab.com/developer-survey/ (Section 5)

[6] IBM and Ponemon Institute, Cost of a Data Breach Report 2026. https://www.ibm.com/reports/data-breach (Section 5)

[7] Atlassian, State of Developer Experience Report 2025, July 2025. https://www.atlassian.com/software/developer-experience (Section 5)

[8] IBM Institute for Business Value (with Oxford Economics), 5 Trends for 2025, survey of 400 global leaders, October to November 2024. https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/business-trends-2025 (Section 6)

Copyright © 2026 Experlogix. All marks are the property of their respective owners. This document is for informational purposes only.