Engineering is (Almost) All You Need: Leveraging Agentic AI for Business Process Automation

Introduction

Agentic AI is getting a lot of attention as the next big thing in automation and digital transformation. But after working on several Agentic initiatives, one thing is clear: all the traditional engineering skills still matter. In fact, they're more important than ever. The "almost" in the title is intentional—yes, prompt engineering and context management are new and critical skills, but they don't replace the fundamentals. They build on them.

Agentic systems, at their core, are just another kind of distributed system. They need the same rigor in design, monitoring, and operations as any other enterprise software. The difference is that now, the "brain" of the system—the agent—can decide what tools to use and how to use them. But it still needs a body: APIs, integrations, and all the digital plumbing that makes up your business. If you don't have that, your agent can't do much.

This paper lays out what's actually new about Agentic AI, and what stays the same. It's not about throwing out everything you know about software engineering. It's about understanding where the new skills fit in, and how to combine them with proven engineering practices to build robust, scalable, and valuable Agentic solutions.

What is an Agentic Initiative?

Agentic initiatives are focused software engineering products where agents—AI systems that use data and tools to assist or automate human tasks—are are put to work. The core idea is simple: use agents to take on tasks that were previously manual, repetitive, or required a lot of human oversight. The goal isn't just automation for its own sake, but to create systems that can make decisions, adapt, and improve over time.

Traditional rules based approaches to automation can be rigid and brittle, requiring plenty of upkeep to accommodate changes in the environment. This is exacerbated by the need for rules to be spelled out in code no matter how pedantic or small. Agents, on the other hand, are able to use reason and common sense to enable handling more scope and adapting to change with less specificity and verbosity in the solution.

Good examples of Agentic Initaitives are products that assist or automate writing letters, making recommendations, or routing customer services calls and collecting the context required for resolution.

What’s Actually New About Agentic Initiatives?

Agentic AI is a new paradigm, but it’s not a clean break from everything that’s come before. The “almost” in the title is a nod to this: while prompt engineering, context management, and evaluation are new and absolutely critical, they don’t replace the fundamentals—they build on top of them. Data driven product management has always been the gold standard for baselining and improving business processes, but it is absolutely required by Agentic Initiatives.

So what *is* actually new here? The biggest shift is in the “brain” of the system. Traditional automation is about building tools and wiring them together with rules. Agentic systems, on the other hand, introduce a layer that can decide *which* tools to use, *when* to use them, and *how* to adapt as things change. The agent is the brain, but it still needs a body—APIs, integrations, and all the digital plumbing that powers your business. The end-to-end solution still needs to be instrumented, observed and maintained by a quality operational team just as any other software product would be.

What's New - The Brain

There is an AI team that handles the new responsibilities and techniques unique to the brain, the new part of building Agentic Initiatives. The AI team is responsible for:

- **Prompt and Context Engineering:** Designing effective prompts and managing context windows are now first-class engineering skills. Getting this right is the difference between an agent that’s helpful and one that’s unpredictable.
- **Evaluation and Feedback Loops:** You need new ways to evaluate agentic systems—both online (in production) and offline (in test environments). This means building systems for continuous feedback, labeling, and experimentation.
- **LLM Operations:** Deploying and monitoring LLM endpoints is a new operational challenge. You need to track not just uptime and latency, but also usage patterns, and model drift. LLMs introduce new cost models. You need to monitor and manage usage, set budgets, and optimize for both performance and spend.
- **Human-in-the-Loop Patterns:** Many Agentic Initiatives require humans to review, approve, or correct agent actions. Designing these workflows is a new product and engineering challenge.
- **Experimentation at Scale:** Running A/B tests, canary deployments, and versioning models are now part of the standard playbook. You need infrastructure to support rapid iteration and safe rollouts.
- **Safety:** LLMs can make up facts and reason poorly. Inputs and outputs should be logged so issues can be looked at by compliance and security teams when necessary and the solution continually improved. Prompt injection attacks should be considered and red teaming maybe a good idea depending on the criticality of the solution. PHI, PII and other confidential information needs to be carefully planned how it will be used and protected in prompts and tool usage.

- **AI Architectural Review Board (ARB):** A new ARB should be stood up that has membership form senior AI Engineering and AI Researchers in the company. The AI team should bring the "brain" of the solution for periodic review where the AI ARB can recommend new technology and approaches along with guidance away from known problems.

What Stays the Same

The Engineering team is responsible for providing the brain the tasks and tools needed to do its job. Both the AI Team and the Engineering teams are responsible for ensuring all components follow all the know best practices including:

- **Product Management:** Determining what to build is usually significant harder than building a system. Product Management is the discipline to only build systems that have ROI, take into account opportunity costs of addressing one opportunity vs another, meet the actual customer need, and function in the real world in a valuable way.
- **Software Development Lifecycle (SLDC):** Source control management, automated CI/CD pipelines, everything-as-code, devsecops and cloud native design should be standard for all software products developed by an enterprise.
- **Observability:** The end-to-end functionality of the software system must be defined, measured and observed so as to alert when events happen outside of the expected tolerance. User impact metrics are key so that system performance is mapped to the real world consequence of what value the product is designed for.
- **Resilience:** Distributed systems can fail at all points and at any time. This must be anticipated and planned for, handling all the various edge cases using industry standard practices like eliminating single points of failure, circuit-breakers, disaster recovery and chaos testing.
- **Digital Transformation:** The basic data and capabilities of an enterprise must be made digital so that software products can act in the real world. This requires integration of systems and partners, documentation of how business processes work and establishing best practices and standard operating procedures.
- **Platforming:** Creating a catalog of re-usable capabilities that are run as products themselves is a powerful mechanism to speed up the digital transformation of an enterprise that allows for Agentic Initiatives to be taken on.
- **Engineering ARB:** Senior engineering leaders should periodically review the end-to-end approach of new and existing solutions to take advantage of new technologies, platform capabilities, and steer solutions away from known problems.

For all that’s new, the foundation is the same for any custom software product. Agents are just another kind of distributed system. They need robust APIs, composable tools, circuit breakers, error handling, and all the monitoring, logging, and tracing you’d expect from any enterprise software. SDLC best practices—versioning, CI/CD, canary testing—still apply. And Agentic

Initiatives are still products: they need user experience and process discovery using techniques like empathy sessions or domain-driven-design event storming, with clear documentation.

In short: Agentic Initiatives introduce new skills and patterns, but they're layered on top of the engineering rigor that's always been required to build, operate, and scale enterprise software systems.

Types of Agentic Initiatives

Not all Agentic Initiatives are created equal. They vary in complexity, the level of AI engineering required, and how much human oversight is needed. Here's a rough breakdown:

- **AI Light:** These are the simplest. Think of light LLM prompting, straightforward evaluation, and basic tool use—like simple classification tasks. There's usually a human in the loop, but only for light oversight. Example: routing calls in a call center.
- **AI Medium:** Here, things get more complex. You'll see more advanced prompting, more involved evaluation, and more complex classification. These projects need a small to medium-sized team and heavier oversight. Example: automating core business processes.
- **AI Heavy:** This is where you're building custom ML models, fine-tuning LLMs, and expecting the agent to handle full automation with heavy reasoning. These initiatives need a large team, require AI research or might even be outsourced to a vendor with specific domain and AI expertise. Example: autonomous coding for medical billing.

Note: Zero to One vs. One to One Hundred

There is a big difference between getting started and optimizing. The "zero to one" (0-1) phase is all about baselining the existing process—usually a process that's still done by humans. You're figuring out what's really happening, what the edge cases are, and what success looks like.

Once you've got something working, you move into the "one to one hundred" (1-100) phase. Now you're improving on an existing automation. The focus shifts to optimization—making the agent more accurate, more reliable and pushing it closer to the theoretical maximum of performance.

Both phases are important, but they require different skills, mindsets, and approaches. Zero to one is about understanding and translating the human process using empathy and domain understanding. One to one hundred is more about engineering, measurement, and continuous improvement.

Note: Task Distributions

Not all tasks that can be automated are created equal. In some processes, each task is about as important as the rest, there aren't a lot of outliers in value or complexity. This is known as a normal or Gaussian distribution. In other processes, a few types or instances of tasks carry all the importance or weight. This is known as the 80/20 rule, power law or Pareto distribution. Identifying this distribution is important for designing the appropriate solution.

When dealing with normal distributions we can fairly safely let the agent try its hand at most tasks as the outcomes are roughly the same from task to task. However, when dealing with a power law distribution we might want to put special attention into classifying which tasks are important and letting humans handle those or spend extra resources on reasoning and correctness.

Roles in an Agentic Initiative

- **Product Manager:** Define opportunities, size investment, define success criteria with business and finance partners, manage budget and represent ROI. They act as a "mini CEO" of the initiative.
- **Software Engineer:** Create the Body of an Agentic Initative. They provide the data pipelines, the APIs and the tasks and tools required by an agent. They are skilled in the traditional Software Development Lifecycle (SLDC). This involves managing infrastructure at scale, build and deploy pipelines, monitoring the end-to-end performance and health of the solution while ensuring it is safe and secure and at a positive ROI.
- **AI Solution Engineer:** Create the brain of an Agentic Initiative. They pick from the standard menu of AI approaches and are skilled in traditional engineering and data science. They can baseline a problem, create evaluation systems, and facilitate collecting and labeling datasets for baselining and evaluations.
- **AI Scientist**: Determine when novel or complicated AI solutions are appropriate. They create new foundation models, train domain specific models, and innovate new approaches to leveraging AI in Agentic Initiatives. They are familiar with data science, traditional ML, generative AI and other more advanced or academic AI techniques.

TODO: Add responsibilities for each role and how they interact on a pod / initiative.

Phases of an Agentic Initiative

Agentic Initiatives aren’t just about plugging in an LLM or building a new agent—they’re structured projects that move through clear phases. Each phase has its own requirements, and each builds on the engineering fundamentals that have always mattered in software.

1. Kick-Off: Set the Strategic Context

Every agentic initiative starts with the basics: why are we doing this, and how will we know if it worked? This is about more than just picking a cool AI use case. You need to tie the project to a real business need—whether that’s digital transformation, automation, or improving outcomes like speed or accuracy.

- **Objective:** Define the "why." What problem are you solving, and for whom?
- **Evaluation Criteria:** Decide how you'll measure success. This could be hard numbers (time saved, accuracy), softer outcomes (usability, satisfaction), or ROI.
- **Determine Type**: Determine what type of Agentic Initiative this likely is to get started - light, medium or heavy.
- **AI ARB:** The AI ARB should review the problem space and propose solutions to help guide the team on when and how to use different kinds of AI and if any new developments are applicable to the problem. They should use this as guidance for what might need additional research.

2. Start AI Development: Translate Goals into Engineering

Once you know what you're aiming for, you need to turn those goals into something engineers can actually build and measure.

- **Scope Definition:** Set boundaries. What does "good enough" look like? What are your accuracy or latency targets?
- **Datasets:** Figure out what data you need, where it comes from, and whether it's good enough (and ethical) to use.
- **Baseline:** Establish a reference point—how well does the current (usually human) process work? What's the industry standard?
- **Outcomes vs. Actions:** Decide if your agent is making decisions (the "brain") or just executing actions (the "body/arms/legs"). How will its outputs actually get used?
- **Modeling Decisions:** Pick your approach—traditional ML, LLMs, or something else. Frame the problem (classification, generation, etc.), and look for ways to improve reliability.
- **AI ARB:** Review the proposed datasets, baseline, modeling decisions and offer feedback and guidance for the brain.
- **Engineering ARB:** Review the end-to-end solution and validate engineering concerns are met and using best practices.

This is where all the traditional engineering rigor comes in. You still need solid APIs, robust distributed systems, and all the SDLC practices you'd use for any serious software. Agents are just another kind of distributed system, and they need the same monitoring, logging, and versioning as anything else.

3. Launch / Deploy: Operate and Improve

Getting an agent into production is just the start. You need to make sure it keeps working—and keeps getting better.

- **Monitoring:** Track everything—accuracy, drift, latency, usage. Catch issues early, whether it's model degradation or operational hiccups.

- **Continuous Learning & Experimentation:** Build in feedback loops for retraining and fine-tuning. Use A/B testing, versioning, and gradual rollouts to keep improving without breaking things.
- **AI ARB:** Periodically review live system and recommend any changes based off of new developments in the field or new offerings from the platform for safety, increased performance, accuracy or reduced cost.
- **Engineering ARB:** Periodically review live system and recommend any changes based off of new platform offerings, operational history and performance.

Remember, Agentic Initiatives are still products. They need empathy for users, clear documentation, and a platform for safe, repeatable deployment. Platform teams should make it easy to re-use tools, deploy and monitor API endpoints (including LLMs!), manage test data, and run experiments—just like any other modern software product.

Conclusion

Agentic AI brings new possibilities, but it doesn't replace the fundamentals. The core engineering skills—APIs, distributed systems, monitoring, SDLC, and product management—are still essential. The "almost" in the title is a reminder that while prompt engineering and evaluation are new and critical, they build on top of everything we already know about building reliable software.

Agentic transformation is just the next step in digital transformation. Agents need robust tools, clear integration points, and the same engineering rigor as any enterprise system. They are distributed systems and products, and should be treated as such—from empathy-driven design to event sourcing and platform infrastructure.

In short: Agentic Initiatives succeed when they combine new AI capabilities with proven engineering practices. The future is agentic, but engineering is (almost) all you need.