**The Art and Science of AI Agents**

**[Dr. Olav Laudy](https://www.linkedin.com/in/olaudy/)**

AI Transformation Executive | GenAI & LLMs

June 30, 2025

**More Than a Chatbot**

There’s a persistent and limiting idea still floating around many enterprises: that Generative AI is just a chatbot on top of a knowledge base. This view reduces its value to a simple question-answer mechanism, useful only when someone wants to retrieve a document or summarize a policy. But GenAI, when deployed properly, is far more powerful. It can become part of the machinery of business processes—not just an interface.

What we’re seeing now is that the design and implementation of AI agents is less about plugging a chatbot into a workflow and more about rethinking what parts of that workflow can be reimagined. There are deeply useful patterns beyond Q&A—classification, extraction, drafting, prioritization, triage, synthesis, judgment—and they don’t always look like a bot.

However, as soon as you call something an "agent," expectations change. People assume a level of autonomy that simply doesn’t match current capabilities. They imagine agents that will go off and complete complex tasks end-to-end, unsupervised. That’s not how it works in practice.

We need to ground the discussion. This piece is about showing that AI agents are not black boxes, nor magic. They are carefully constructed systems embedded in real workflows—and they are built using a mix of art and science.

**The Misunderstood "Agent"**

Let’s be clear: not everything called an agent is autonomous, and not everything autonomous is useful. When we say “agent,” people often imagine something like a software employee—a digital executor that understands intent and just handles it. That’s aspirational. Today’s effective agents are far more constrained and targeted.

One example is email triage. It’s tempting to say, “My agent answers emails.” But if a human still needs to review, sign off, or ensure regulatory compliance, then the agent isn’t really answering—it’s drafting. That distinction matters, especially in high-stakes domains.

Real agent value often lies in what we can call partial autonomy. An agent that classifies incoming requests, identifies the key question, and suggests a response is incredibly valuable—even if it doesn't hit send. Autonomy at the right level of abstraction is what matters.

Designing good agents is not about end-to-end automation. It’s about finding leverage. The art is in slicing the process into subcomponents where GenAI adds meaningful lift without introducing unacceptable risk.

**Embedding GenAI in Real Workflows**

The core design challenge isn’t building a chatbot—it’s integrating GenAI into business operations in a way that respects how work actually happens. Most business processes involve a mix of structured and unstructured data, rules, exceptions, and human judgment.

The winning strategy is decomposition: take a workflow, break it into its functional components, and then identify which of those can be enhanced, accelerated, or offloaded to GenAI. This is rarely a clean, top-down exercise. It’s usually iterative and empirical.

For example, in customer support, the high-value move isn’t to build a bot that solves every issue. It’s to use GenAI to summarize previous tickets, classify intents, suggest responses, and surface relevant context. That lets the agent—i.e., the human—act faster and better.

You don't start by saying, “What can the AI do?” You start by asking, “Where is the friction?” and then design with GenAI as the friction-remover, not the full actor. This shift in mindset is where the best agent use cases emerge.

**Executable Hypotheses**

Here’s where the science starts to look like craft. In early phases, building AI agents is best understood as constructing *executable hypotheses*. These aren’t final systems—they’re live tests. They don’t need perfect data or production-grade code. They need interaction, evaluation, and feedback.

Just like in traditional machine learning, there’s a loop: propose a behavior, generate synthetic or semi-synthetic inputs, observe outputs, and iterate. The agent isn’t done—it’s *being shaped*. What you learn from a fast prototype is far more valuable than what you could write in a specification.

An executable hypothesis isn’t a mock—it runs. It accepts input, produces output, and can be judged by users or other AI systems. It’s not correct, but it’s testable. That’s a major shift in how software is built. The point isn’t just feasibility; it’s fit.

And here’s the kicker: this is how you discover what GenAI *actually* can do in your domain. Most businesses don’t yet understand the capabilities deeply enough to spec them. So you don’t write the blueprint first. You test the boundaries with working artifacts.

**Vibe Coding: How You Do It**

Once you've accepted that early AI work is about hypotheses, the next question is: how do you build those hypotheses? This is where the method of *vibe coding* comes into play. It's not traditional development. It's exploratory, fast, and responsive.

Vibe coding is about working from a gut sense of what the user experience should feel like—not from a formal spec. You start with a rough idea and build a live artifact that someone can actually use. It’s fast, dirty, and designed to generate real feedback.

You wire together a basic LLM prompt, maybe a small tool wrapper or API call, and push some representative inputs through the system. Immediately, you see what works and what doesn’t. Then you iterate. Quickly.

This is how you discover the edge cases, the latent assumptions, the UX blind spots. It’s not academic. It’s empirical. You don’t imagine the problem; you confront it. That’s the difference vibe coding makes in early agent development.

**Prompt Fitting as a Development Discipline**

Once you have your hypothesis running, the question becomes: how do you shape its behavior? This is where prompt fitting comes in. It’s not model tuning, and it’s not writing code. It’s empirical alignment of input-output behavior via structured prompts, examples, and test cases.

The analogy to machine learning is clear. You have “training data” in the form of prompt examples, and “validation data” in the form of synthetic or historical tasks. The goal is to elicit the desired behavior repeatedly—not perfectly, but reliably.

You can generate 25 categories of inputs—say, email types or contract clause types—and then produce 100 variants of each. Now you’ve got 2,500 structured test cases. You run the agent across them, evaluate the output, and refine. You don’t need human labels—you can use a second LLM as a judge.

This is where the science becomes real: you have a prompt, you have test coverage, and you have metrics. Not every use case needs this rigor, but for anything beyond demos, this becomes indispensable. It’s prompt engineering at scale.

**From Vibe to System: AgentOps**

Eventually, the prototype becomes too valuable to throw away—and too risky to run as-is. This is where AgentOps enters: the discipline of turning GenAI prototypes into production-grade systems. Like DevOps or MLOps before it, AgentOps is about scaling, monitoring, and evolving AI-driven functionality.

This includes logging, evaluation, fallback behavior, prompt versioning, context traceability, escalation paths, and continuous testing. Prompts change. APIs evolve. Use cases drift. AgentOps ensures continuity and observability across all of it.

This is not optional if you want reliability. A one-off agent that solves one problem today becomes tomorrow’s legacy if you don’t have lifecycle support. That means CI/CD for prompts, test harnesses for LLM behavior, and dashboards for outcomes.

AgentOps is how we move from craft to system—from clever to credible. Without it, every agent is a snowflake. With it, you get leverage.

**The Case for a Common Platform**

If you allow each agent to be built in isolation, you’ll end up with a zoo: different stacks, different test strategies, different guardrails, different prompts. That fragmentation kills reuse and slows down iteration. Worse, it creates governance and risk headaches.

That’s why a shared platform becomes essential. The components are starting to look standard: prompt registries, evaluation suites, guardrail modules, logging pipelines, synthetic data generators, and agent catalogs. These form the backbone of responsible scale.

Such a platform also allows teams to work faster. You want to reuse an intent classifier? It’s already there. You want to plug in legal review prompts? Already versioned and tested. The platform is a force multiplier.

Just as enterprises standardized APIs and service frameworks, they will standardize agent foundations. This is how you prevent the chaos of disconnected innovation—and how you build a foundation for continuous improvement.

**The Real ROI: Would You Go Back?**

People ask about the ROI. If your GenAI system is truly useful, the ROI isn’t buried in a spreadsheet. It’s on the faces of the people using it. They don’t want to go back. They see the system as indispensable.

In customer support, agents resolve more tickets, faster, with fewer escalations. In legal review, contracts are triaged, clauses flagged, comments suggested, and the lawyer shifts from drafter to reviewer. In both cases, the result isn’t cost-cutting. It’s *friction-cutting*.

The lawyer still has a job. But now they spend more time on judgment and negotiation—and less on copy-paste and clause-hunting. When the system misses something, they guide it: “No, that’s not what I meant. Regenerate, but focus on X.” This is augmentation, not automation.

You can measure ROI via latency, throughput, and coverage. But the deeper signal is stickiness. If users won’t give it up, you’ve succeeded. ROI is not theoretical. It’s behavioral.

**From Craft to Discipline**

We’re in a transitional moment. The best GenAI agents today feel like craftwork: hand-built, carefully tuned, deeply domain-specific. But the patterns are emerging. The scaffolding is becoming standardized. We’re watching the shift from art to science.

Executable hypotheses, vibe coding, AgentOps, platform foundations—these aren’t buzzwords. They’re the ingredients of a new repeatable methodology. They’re how enterprises move from novelty demos to operational transformation.

The work is hard because it’s new. But that’s not an excuse. It’s an opportunity. Those who understand how to structure this kind of system will define the next generation of intelligent workflows.

In the end, AI agents are not magic. But if designed with care, tested with rigor, and deployed with discipline—they will feel like magic to the people who use them.