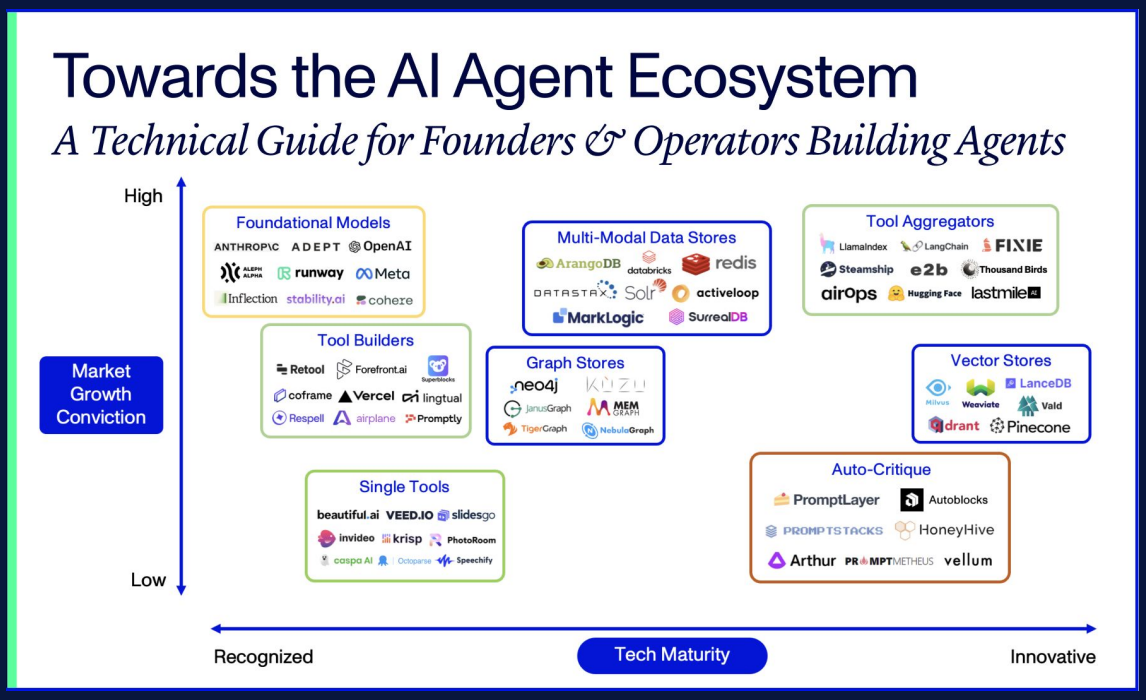
ACTIVANT RESEARCH

Towards the AI Agent Ecosystem

A Technical Guide for Founders & Operators Building Agents

Marc Wu, John Matheson, Julius Stener, Andrew Steele



Q3 2023

We are hardly the first to claim a new technology is poised to upend work as we know it. With every wave of innovation comes a chorus of such pronouncements, many of which turn out to be hyperbole. Technology may make our jobs easier, but most of the time, the core of what we do remains fundamentally the same. But spend five minutes on ChatGPT, or any comparable Large Language Model (LLM), and it's clear: it's different.

The reason is simple. Most technologies help us accomplish tasks. They're tools, a means of making us more productive. AI, on the other hand, can think (or at least, provide the appearance of thinking). It's like having another knowledge worker on the payroll.

Activant Perspective 1: LLM-enabled software will autonomously accomplish highly complex tasks.

This is the beauty (and threat) of LLMs: they can do work we thought would long be the purview of humans.

Given this, one could be forgiven for worrying their job will soon be automated away by an AI. In the short term, this will lead to dislocation. But technological revolutions are never zero-sum – in fact, they've always been accretive. As workers adapt, they become more productive, and the companies they work for can do more with less. Some stop there, to be sure, but most businesses opt to invest the gains back into their infrastructure and people. In the end, revenues go up, and the economy grows.

That said, we're still in the very first days of this cycle – and critically, of the technology itself.

Activant Perspective 2: LLMs alone are not enough to achieve Perspective 1.

Case in point: a large language model can't complete most tasks on its own. After all, an LLM is merely a dense set of attention networks; essentially, statistics on a massive scale. Yes, they can engage in human-like "thought," but to generate real value from any of these models, a human would have to integrate them into actual workflows – and make use of their decision-making prowess.

For example, imagine that you hired someone to lay bricks. You could lead them to a stack of bricks and explain where and how you want them laid. They could perfectly understand your instructions. Yet, understanding isn't enough – the worker needs to be able to, well, lay bricks.

In this case, they need to have more than just cognition – they need to have arms, legs, and physical strength. Furthermore, they need a memory to recall how you wanted each of the bricks to be arranged, and a way of correcting any errors so they don't cause the eventual structure to fail.[1]

Put another way, if LLMs are to extend beyond mere Q&A, they need (1) a suite of Tools to actually impact the real world, (2) memory repositories to remember what actions they've taken, and (3) auto-critique algorithms to error correct along the way.

When all these components come together, an "Agent" is born.

Technical Definition: An Agent is orchestration software that combines an LLM with memory, tools, and auto-critique algorithms.

Non-Technical Definition: An Agent is a highly advanced bot that can complete work that has been assigned to it in natural language.

Though still in their infancy, Agents are poised to fundamentally reshape the way we work. They will become natural extensions of your organization and complete day-to-day tasks – akin to hiring more employees. They'll help you get work done, even collaborate with others on your behalf. Eventually, they'll be able to work together, breaking down tasks into atoms of work able to be handled by a full "office" of Agents, some of which orchestrate, while others specialize.

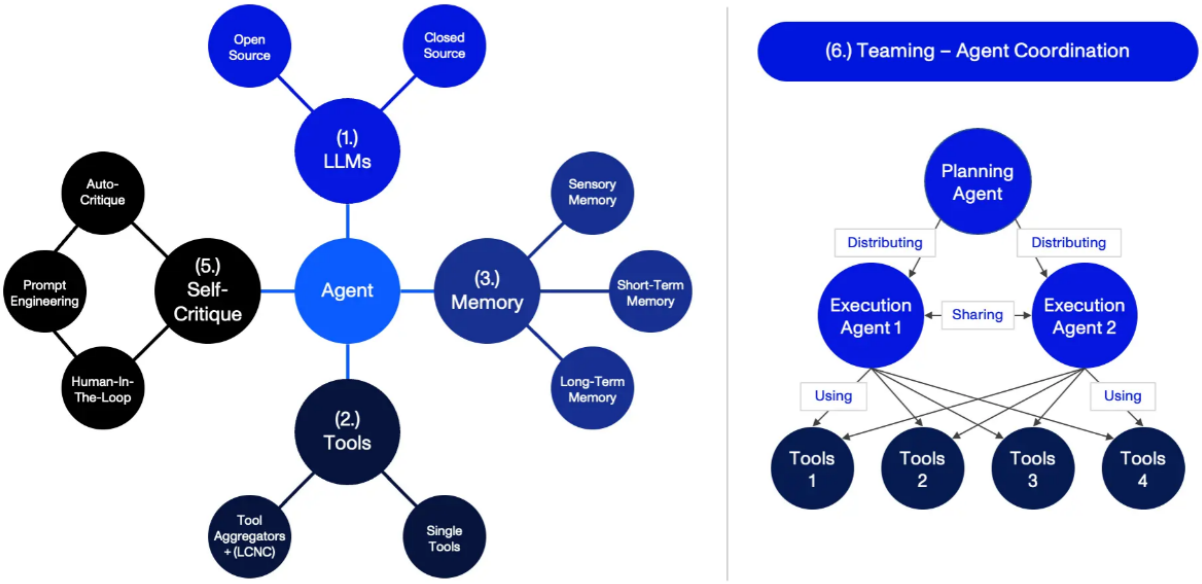
The result is why this time is different: working together, Agents will be able to handle increasingly complex tasks, replacing knowledge workers in a variety of fields. As they do, what it means to work – and to lead – will change drastically, as both will increasingly depend on a mix of human and human-like capital.

Activant Perspective 3: Agent Teaming is the future of autonomously completed, complex work – *and that future is possible today*.

Few have sketched a path from where we are today, to the autonomous knowledge workers of the future. But that future is closer than you'd think, and one for which you almost certainly want a roadmap. To learn more about how this transformation will unfold, and how you can best position yourself within it, read on.

ACTIVANT *Research*

The Agent Ecosystem [2]



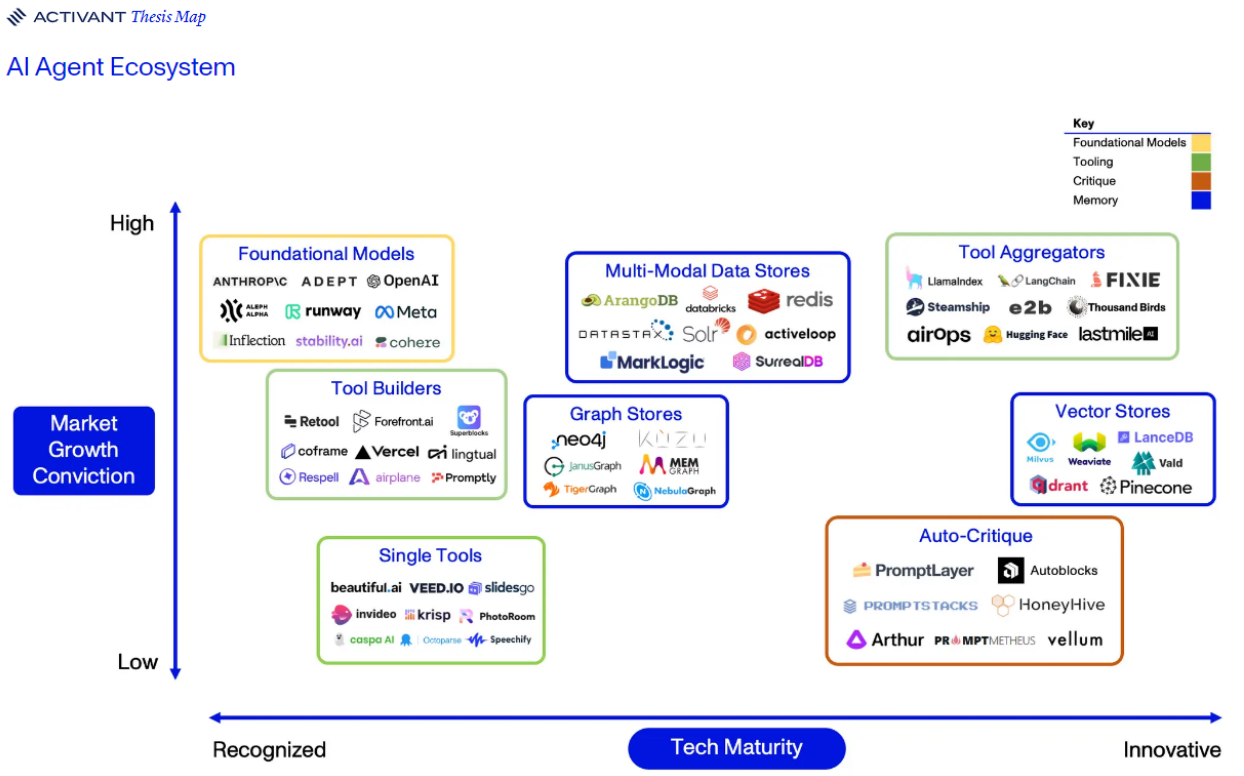
The Agent Ecosystem[2] of the future has four key components:

1. **LLMs (Large Language Models):** provide "conscious thought" within AI Agents and have been trained on text data,
2. **Tools:** external APIs to facilitate task completion and Agent interaction with the real world,
3. **Memory:** includes various data storage mechanisms to retain and recall information, and
4. **Self-Critique:** the ability to correct mistakes when completing tasks.

To better understand the components, we imagine how each one might erode a job we know quite well: that of an analyst at Activant (sorry, John). Among many other things, we often rely on our analysts to create market maps, which requires them to:

1. Understand an industry,
2. Find companies that exist in that industry,
3. Compare these companies qualitatively, and
4. Place them on a 2-dimensional grid according to this comparison.

Until recently, this was almost inconceivable as a task for machines. Now, let's explore how we'd put our "Agent Analyst" to use.

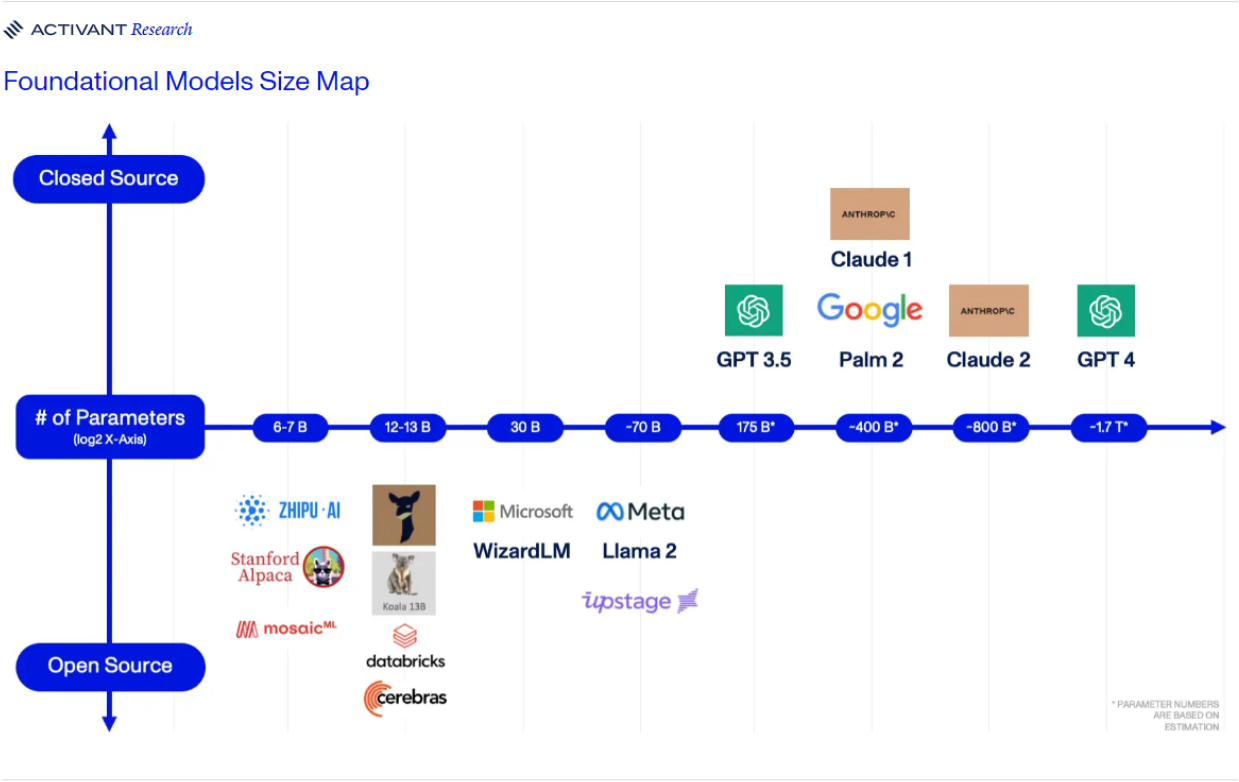


1. Understanding the Industry (and the Task) – Large Language Models

For our Agent Analyst to understand the task at hand, it requires the capacity to comprehend instructions stated in natural language. While LLMs provide that ability, they come in many different shapes and sizes (literally). For that reason, here we outline a basic framework for evaluating LLM providers, including both foundational model providers and companies offering Fine-Tuning as-a-Service (FTaaS[3]).

When comparing foundational models, the number of parameters in a model is often a good approximation of output quality. As a rule of thumb, the more parameters a foundational model has, the better its performance across different benchmarks, such as reasoning, web browsing, or even game playing. Although fine-tuning a model can boost its performance on a particular task, larger models with more parameters tend to have better generalization skills – making them more useful.

As seen below, model sizes are increasing on an exponential basis with base 2, and although not pictured, model performance is similarly increasing – just not at the same rate.

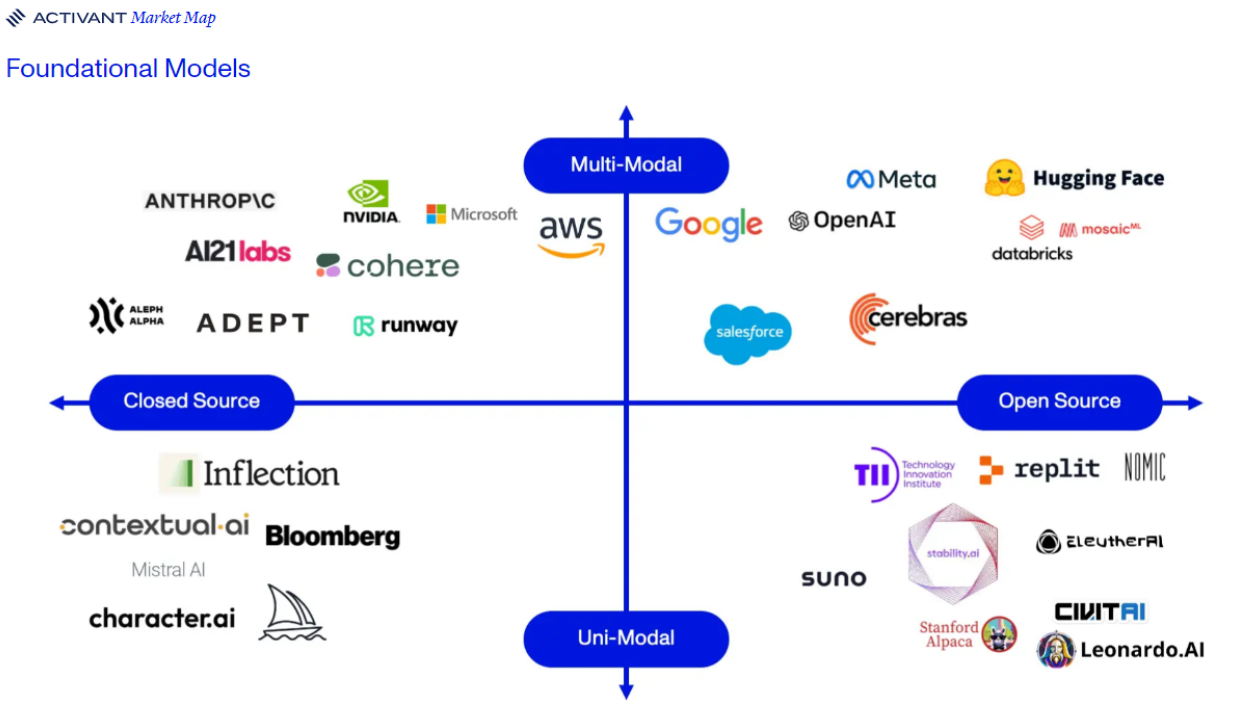


Open vs. Closed Source LLMs

At the heart of any LLM lies its architecture: a blueprint dictating how various attention heads and layers are organized. Then, there are the "weights" — numeric values assigned to parameters in the neural network, refined and adjusted through training to enhance performance. Training involves feeding data into the model, allowing it to learn and make predictions, and then adjusting the weights based on its accuracy. It's through this process, often requiring substantial computational resources, that the model obtains its ability to understand and generate human-like text.

Open-source models, such as Llama 2 by Meta, "openly" share their architecture, weights, and sometimes even their training methodologies. This approach not only reveals the intricacies of their design but also democratizes access, allowing enterprises, researchers, and even enthusiasts to build upon, refine, or entirely repurpose these models for their own applications.

Conversely, some entities opt for a more guarded approach. Closed-source models, such as GPT-4 by OpenAI, often protect the details of their architecture, weights, or training processes, presenting them as somewhat of a black box. Instead of allowing direct model access, they employ APIs as intermediaries, thereby controlling and often monetizing the interaction between the end user and the model.



Provider Modality

"Provider Modality" relates to the data type a provider offers models for. This includes various transformations such as text-to-video, text-to-text, text-to-image, and image-to-text, among others. Companies that operate in just one domain are labeled as "Uni-Modal"; by contrast, "Multi-Modal" companies provide models spanning multiple domains. As Agents continue to evolve, we expect a growing emphasis on modality in LLM models, as those able to perform more diverse tasks and offer easier integrations will have a distinct advantage in the marketplace.

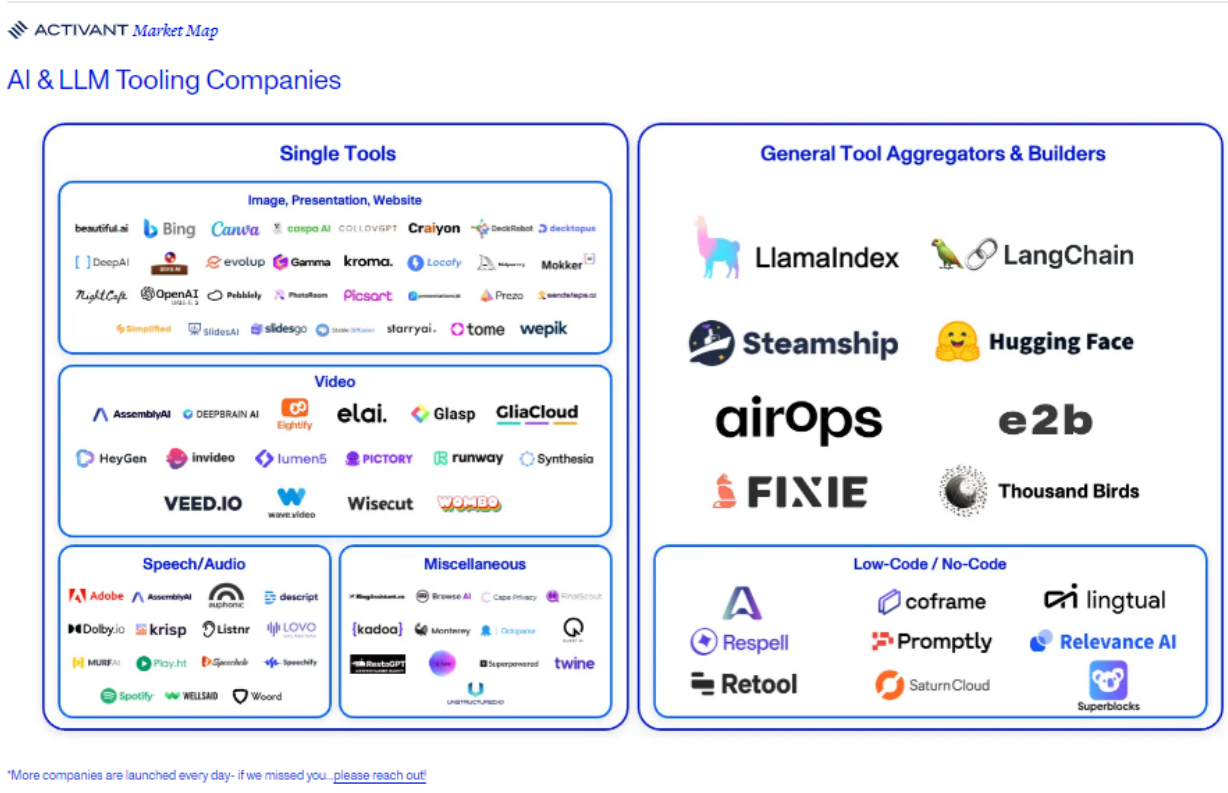
While Agents will no doubt depend on depend on the strength of their underlying LLM, we actually expect most AI agents to remain agnostic to the specific LLM they use. OpenAI, Google, and Anthropic dominate the industry, but this is hardly set in stone. If anything, the next few years will see considerable competition, with whoever invests the most in R&D in a constant arms race. Given this uncertainty – and the promise of better models always around the bend – it's hard to imagine Agents relying on a single provider. From an investment perspective, backing one LLM provider is essentially a vote of confidence in its R&D team, but past performance is not (necessarily) indicative of future results.

2. Finding Companies – Tools

Having understood the industry (and the task), the Agent Analyst now needs the ability to search for companies. In other words, the Agent Analyst needs a Tool.

***Definition*: Tools are snippets of code that foundational models can use to create, modify, and utilize external resources in order to execute tasks they couldn't otherwise complete on their own.**

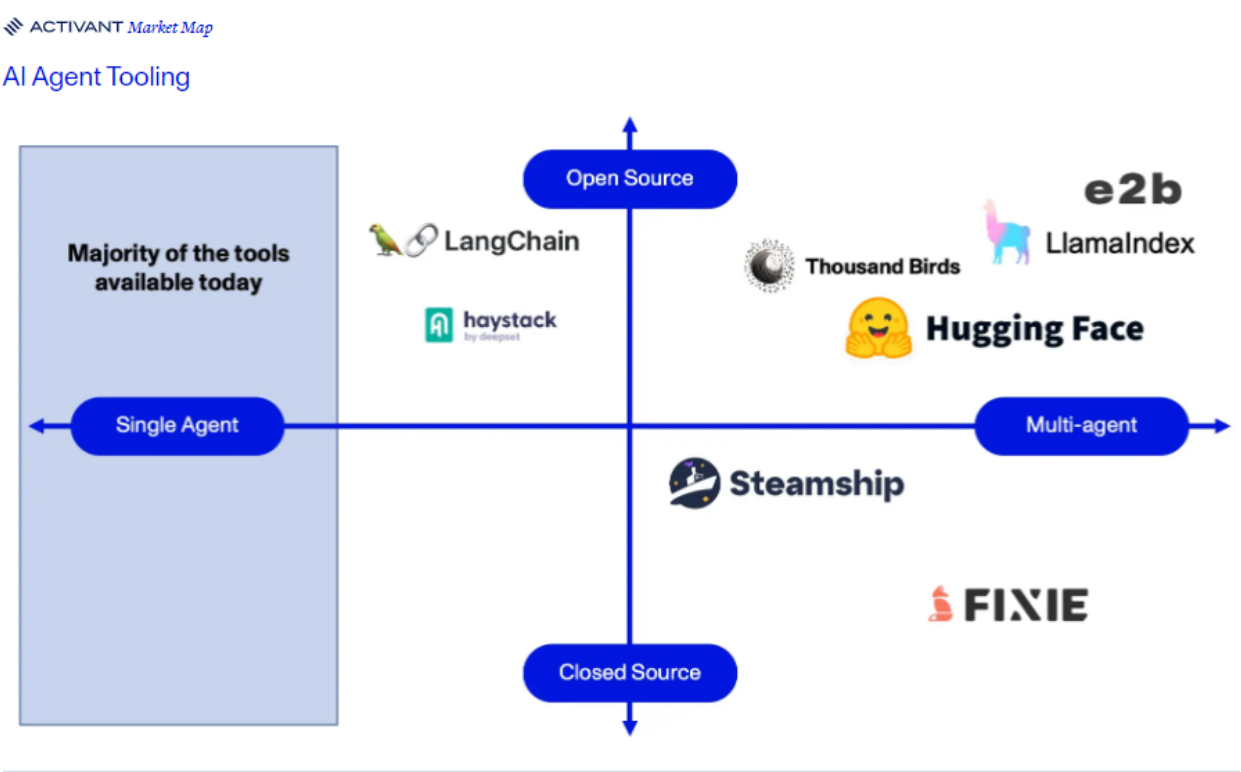
Most Tools are problem specific. In this case, all the Agent Analyst needs is a connection to Google, or SerpAPI, a prominent wrapper for Google's Search API. Unsurprisingly, other tasks will require different Tools. Game this out, and you quickly arrive at the need for Toolkits and Tool Aggregators. As tasks become more complex, Agents will rely on not one, but a variety of Tools to accomplish a single objective. Through Aggregators, Agents will be able to access the resources they need, programmatically, and as those needs arise.



Single Tools

Single Tools refer to companies or services that specialize in just one function, such as searching the internet, booking a flight, or creating realistic voice audio files. While these offerings do have

certain strengths, integrating them into a Toolbox, and thus into Agents, is often cumbersome. Notably, in prominent Agent libraries[4] like LangChain, Toolboxes need to be set up before an Agent itself is even initialized. This means if there's a need to integrate additional Tools later – for example, if an Agent encounters a new task that requires a Tool outside of its preset toolkit – the Agent would have to initialized all over again. Given these considerations, embedding Single Tools into an Agent's workflow can be quite labor-intensive and time-consuming.



Tool Aggregators

By contrast, Tool Aggregators offer Toolboxes-as-a-Service (TaaS). Theoretically, aggregators can compile an infinite number of Tools and distribute them via a single, unified API. In practice, they typically provide Tools that have been identified by an Agent as required to complete a given task. If an Agent requires determines it needs additional tools, however, it can access them without restarting.

Low-Code / No-Code (LCNC)

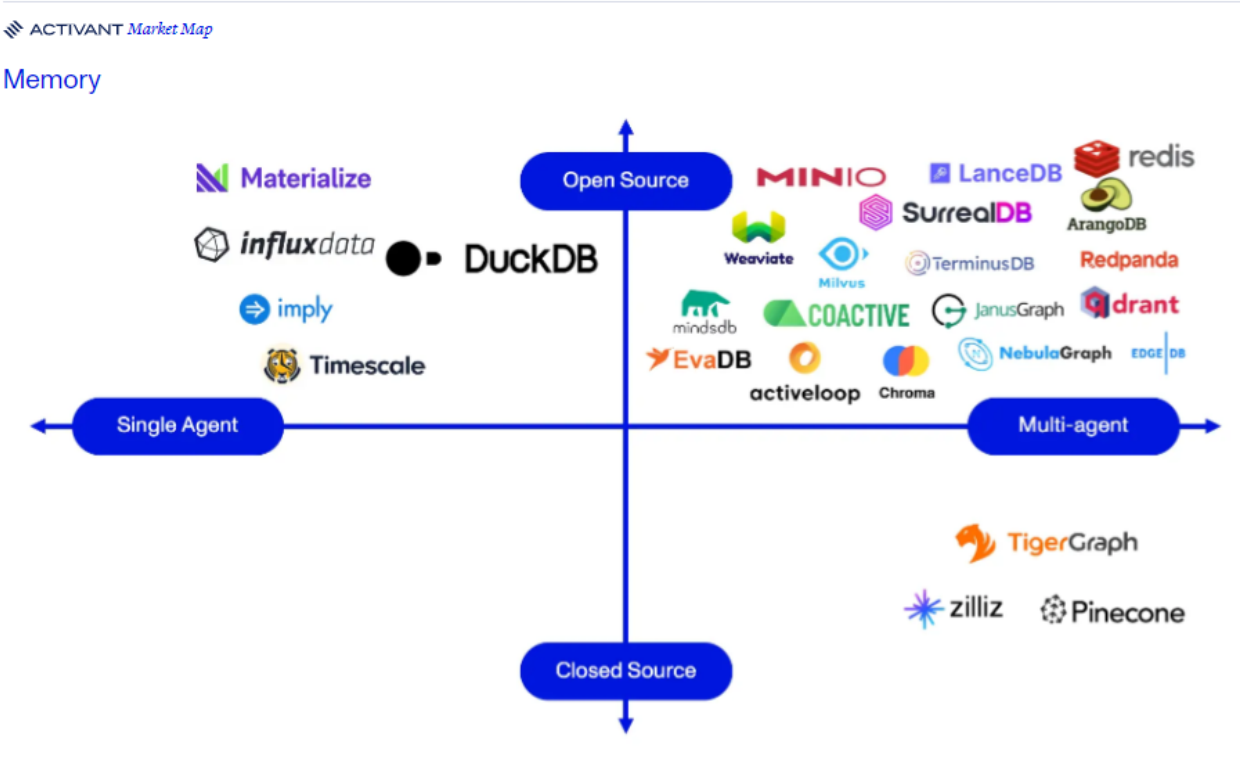
A subset of Tool aggregators, LCNC companies could provision Tools and/or provide tool-building functionality for Agents. As long as they offer an API, Agents can leverage these solutions as they do Tool Aggregators, integrating (and even custom-building) Tools into their workflows on the fly.

Many LCNC platforms have existed prior to the advent of LLMs and already have large libraries of powerful, ready-built Tools to draw upon. Several companies, including Relevance AI and Superblocks, have begun offering Agents-as-a-Service (AaaS), using these libraries to jump ahead of competitors, many of which are starting from scratch. Others may choose to offer APIs to facilitate quick adoption within the Agent Ecosystem and more reliably maintain their market share.

Because of their ease of integration and use, we believe Tool Aggregators and API-based LCNC platforms are the best positioned to take advantage of this new Agent ecosystem. As platforms like LlamaIndex and Relevance AI grow, Agents will be able to dynamically access a vast library of toolsets without needing to update their structure or code – increasing both their functionality and value.

3. Comparing Companies – Memory

As our Agent Analyst has progressed, it has gained the capability to discern tasks and search for companies. However, identifying a company means nothing if an Agent can't remember it for later use. To facilitate this, Agents require a dedicated memory component that can store companies and their associated descriptions.



Over the years, the software industry has seen a multitude of data-structure innovations, from traditional SQL-based relational tables to modern streaming services like Kafka. Yet, to truly grasp the intricacies of Agent development and the depth of virtual cognition, it's helpful to draw parallels with human memory.

At a high level, human memory is split into three core categories: sensory memory, short-term (or working) memory, and long-term memory.

Sensory Memory

In humans, sensory memory can be thought of as the direct inputs from our various senses — taste, smell, sight, touch, and hearing. These are perceived subconsciously, processed then filtered for relevance. Similarly, in the realm of Agents, **sensory memory captures the responses generated from the Tools they employ.** Research in neuroscience and psychology reveals that of the countless sensory inputs humans encounter every moment, only about 5% ascends to conscious awareness, courtesy of various attention mechanisms. Our neural framework efficiently filters out redundant and useless sensory data, elevating only important information for conscious processing. The sensory memory of an Agent operates on similar principles, **sieving through vast data while retaining only what's significant.**

For example, to store the names and descriptions of any relevant companies it finds, our Agent Analyst might choose to create a vector database. To do that, the Agent could use the Pinecone API and receive a specifically formatted JSON response directly. Though the Agent could use the LLM to parse that JSON response, it would be far more efficient for the Agent to use the Pinecone Tool from Llama Hub and receive a pre-processed Python dict. In many ways, this is analogous to how our subconscious mind pre-processes data before it needs to be consciously thought about – i.e. you don't have to process the shoe touching your foot when you're using your finger to touch your phone because your subconscious brain filters the shoe sensation out.

Over time, we believe that sensory memory will include the ingestion of many different data types, from unstructured data files (such as docx, pdf, epub, or recordings) to semi-structured datasets (such as messages and CRM data). **As of today, it is unclear to us if "sensory" processing should or could be handled at the Tool level,** like it is in the above Pinecone example, because we don't know if Tool Aggregators will provide a standardized API for these tools – or if it's even possible to have a standardized API across all conceivable Tools. What we do believe as a result of our work is that the quality of Agent-handled sensory memory will be highly correlated with the quality of the Agent itself.

Short-Term Memory

Having processed inputs on the subconscious level, Agents are left with the inputs which require conscious thought. In humans, conscious thoughts reside in working or short-term memory. When we actively think about a topic, we do not consciously remember the context for the thought in our

"memory," but it is nevertheless remembered. In fact, our short-term memory is believed to have the capacity of about seven items[5] and lasts for 20-30 seconds. For Agents, using the context windows defined by LLMs to store short-term memory is the default solution of most Agents we see today, and we don't see that changing. As context windows increase in length and new techniques for prompt engineering solve existing constraints, the context window will be an even more effective location for short-term memory.

Interestingly, most short-term memory will be forgotten after a certain time threshold, especially after corresponding actions are finished. Though some of this information may be transitioned into long-term memory (which we discuss below), most of an Agent's short-term memory will not be useful after a task is completed. Therefore, short-term memory data can be stored in the context and does not require the same complex storage techniques long-term memory does.

In the context of our Agent Analyst, short-term memory can be thought of as remembering the larger task at hand (i.e., creating a market map) when deciding what smaller task (i.e., finding a company) needs to be accomplished next. In practice, this information will largely live inside the context window of the LLM API call – and while it's essential for the Agent to complete a task, we believe that there is little opportunity for investment in short-term memory due to the limited additional infrastructure required to build it.

Long-Term Memory

Once subconscious and conscious memory is in place, an Agent Analyst will need some way of saving information for longer periods than the immediate task at hand. This is where long-term memory comes into play.

In humans, long-term memory is finite, due to the limited capacity of the brain. As a result, we have developed two primary mechanisms to retain knowledge: repetition and adrenaline. For obvious reasons (i.e., they can use ACID databases), Agents lack this capacity constraint. Instead, they are limited by the amount of memory that they could conceivably "know" how to access. For example, if our Agent Analyst was integrated into our Drive and needed to search for the PowerPoint example of our Market Map, the size of the context window would limit the number of folders that it could select from in its search.

In our view, connections to long-term memory repositories are no different than Tools, in that they are blocks of code that can be executed and return a response.

Over the next decade, we believe that the entire ecosystem of data management will grow even faster than it has historically because of increased Agent data utilization. Agents will be able to build pipelines from company data, leverage it in their own processes, and integrate outside data into existing workflows. The total spend on databases and database management solutions doubled from $38.6B in 2017 to $80B in 2021[6]. As the Agent ecosystem grows, the demand for data stores will increase even more rapidly, making the volumes of unstructured company data – which today sits untouched – significantly more valuable.

We've listed a handful of data structures, and the impact we anticipate from Agents, below:

ACTIVANT *Research*

AI Agent Use Case w/r/t Databases

	Relational Database	Graph Database	Vector Database	Multi-Model Database
Speed	●	◑	◔	◑
Complex Relationships	◑	●	◔	◕
Semantic Meaning	○	○	●	◕
Multi-modal	◔	◔	●	◕
Corporate Adoption	●	◔	◔	◑
AI Agent Retrieval Methods	Text-to-SQL	Text-to-Cypher, Text-to-Query	Vector Embeddings Similarities	All mentioned
AI Use Cases	Model Training, Business Intelligence, Numerical Analytics	Knowledge graph, Recommendation system, Fraud detection	Semantic Search, Unstructured Search, Classification	All mentioned

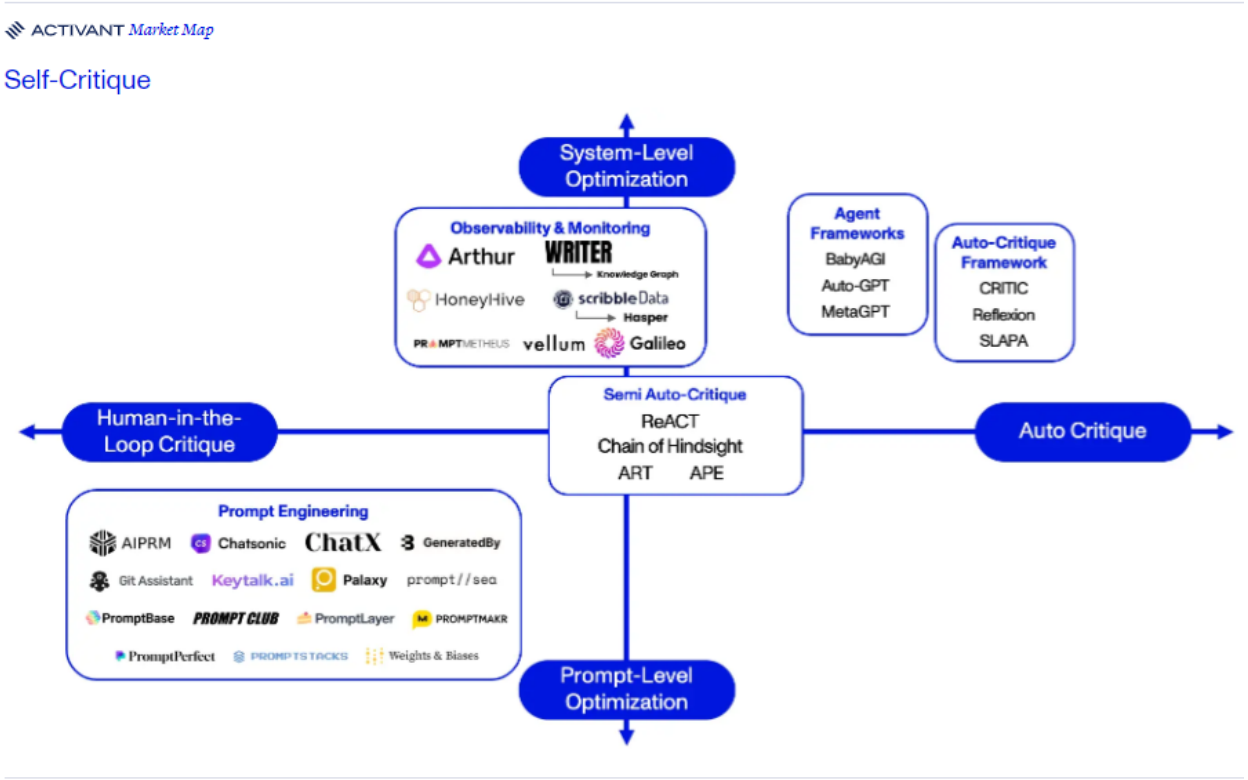
4. Placing Company Logos – Auto-Critique

Our Agent Analyst has successfully understood the industry, used a search Tool to find all the companies within it, and saved them in a database for future output. As a final step, we need the Agent to place the logos of each of those companies onto an Activant Market Map – fully understanding the significance of each company's position. Naturally, this will involve a Tool that enables the Agent to create graphs. In the short term, this Tool will likely be some wrapper around Our Agent Analyst has successfully understood the industry, used a search Tool to find all the companies within it, and saved them in a database for future output. As a final step, we need the Agent to place the logos of each of those companies onto an Activant Market Map – fully understanding the significance of each company's position. Naturally, this will involve a Tool that enables the Agent to create graphs. In the short term, this Tool will likely be some wrapper around Matplotlib[7] but in the long term, this will undoubtedly be an integration into PowerPoint.

Plotting logos, of course, is the easy part of the challenge; understanding where to place a logo in a subjective 2-dimensional plane is the truly complex task. It requires iterations, constant

comparisons to other companies, and perhaps even outside input from the Activant team. Therefore, the Agent needs a mechanism for critiquing itself. We call this Auto-Critique.

While we foresee potential challenges - like defensibility - this remains a vital component of the ecosystem with several approaches underway: (A) Prompt Engineering, (B) Human-In-The-Loop, and (C) Auto-Critique.



A) Prompt Engineering

Definition: Prompt Engineering is the process of forming and refining natural language statements and questions to achieve an optimal outcome from an LLM.

In practice, LLMs are only capable of comprehending and responding to inputs that fit within a defined context window. For example, GPT4's context window is about eight thousand tokens – meaning only about seven thousand words can be provided as context at any one time.[8] In order to maximize the utility of the model, a prompt engineer could write shorter and more effective prompts.

In our view, prompt engineering is likely a temporary solution, one we expect the industry to shift away from as LLMs improve in terms of task comprehension and context window size (though the timeline for this shift remains uncertain). Meanwhile, existing businesses in this domain have a

chance to pivot their business model, leveraging their current distribution to tap into emerging opportunities.

B) Human-In-The-Loop

Definition: Human-In-The-Loop Critique refers to the use of humans to validate that Agents are accurately processing and reasoning through tasks.

Human-In-The-Loop Critique is just what it sounds like: leveraging humans to correct or modify the reasoning of LLMs. Currently, developers use this method when building Agents, as they want visibility into their inner workings. Because it requires that people scale linearly with compute resources, it will likely be unsustainable in the long term.

C) Auto-Critique

Definition: Auto-Critique refers to the autonomous use of LLMs to check and modify Agent reasoning about a task.

Humans often make decisions in multiple steps: starting with an initial judgement, then assessing its consequences, and evaluating alternatives. Throughout this process, we add new information, remember past experiences, and take note of what others have done. We call this "thinking it through," and LLMs perform a similar process when refining their own outputs. Like the brain, they begin with a rough first pass – a digital gut instinct, one might say – before iterating to either expand on that conclusion or critique it.

Auto-Critique is an autonomous process that allows Agents to improve and amend their own output by reflecting on past responses and errors, all without human intervention. To do this, Agents draw on new input from a variety of sources, including external Tools, other Agents, affiliated LLMs, and heuristic functions. Once they arrive at a sufficient result (governed by predefined metrics for evaluating self-reasoning), a stop condition is granted, allowing Agents to finalize their output.

Teaming: The Rise of the Planet of the Multi-Agent Systems

So far, we've focused largely on the mechanisms and potential of standalone Agents. However, to fully appreciate the potential of this nascent sector, we must also look at the collective strength of Agent Teaming, otherwise known as Multi-Agent Systems (MAS). Within the framework of a MAS, the power of individual Agents is not merely aggregated – it's amplified, boosting the overall effectiveness of the system in a multiplicative and potentially exponential manner.

Returning once again to our Agent Analyst example, we must remember that building market maps is but one of many tasks of an Activant analyst. In fact, a market map is only a small piece of the larger research and investment process – analysts must also deal with financials, retention, defensibility, and much more. In their current state, no one Agent could write a complete investment memo, though individual Agents could accomplish each component. This is the promise of Multi-Agent Systems: they're able to accomplish significantly more complex tasks than single Agents ever could.

Definition: A MAS is a network of intelligent Agents that interact with each other, forming a Machine-to-Machine (M2M) communication network.

Early attempts at the concept (i.e., March 2023), such as Agent-Based Systems (ABS), were inherently limited. In an ABS, though multiple Agents work on the same task, they can't collaborate. They're also forced to work within narrow, predefined constraints. The result? Rather than act

autonomously, Agents are confined to rudimentary roles with very strict and cyclical operating patterns. Think of BabyAGI's cycle of operation: Agents follow relatively one-dimensional orders, are only sequentially prompted, and report back to their "commanding" Agent when given a task is complete. As one can imagine, given the limited scope and flexibility, such systems have been less than transformative.[9]

By contrast, a MAS encourages direct communication between two or more Agents – with minimal rules governing how they operate. This supercharges the system, leveraging Agents' *collective* cognition and problem-solving capabilities to solve complex problems quickly, reliably, and effectively. As systems shift away from rigid and sequential orders to more dynamic, human-like interaction, their ability to tackle such problems will only increase.

Below, we further examine the concept of Multi-Agent Systems, highlighting their novel and significant advantages.

The Value of Multi-Agent Systems

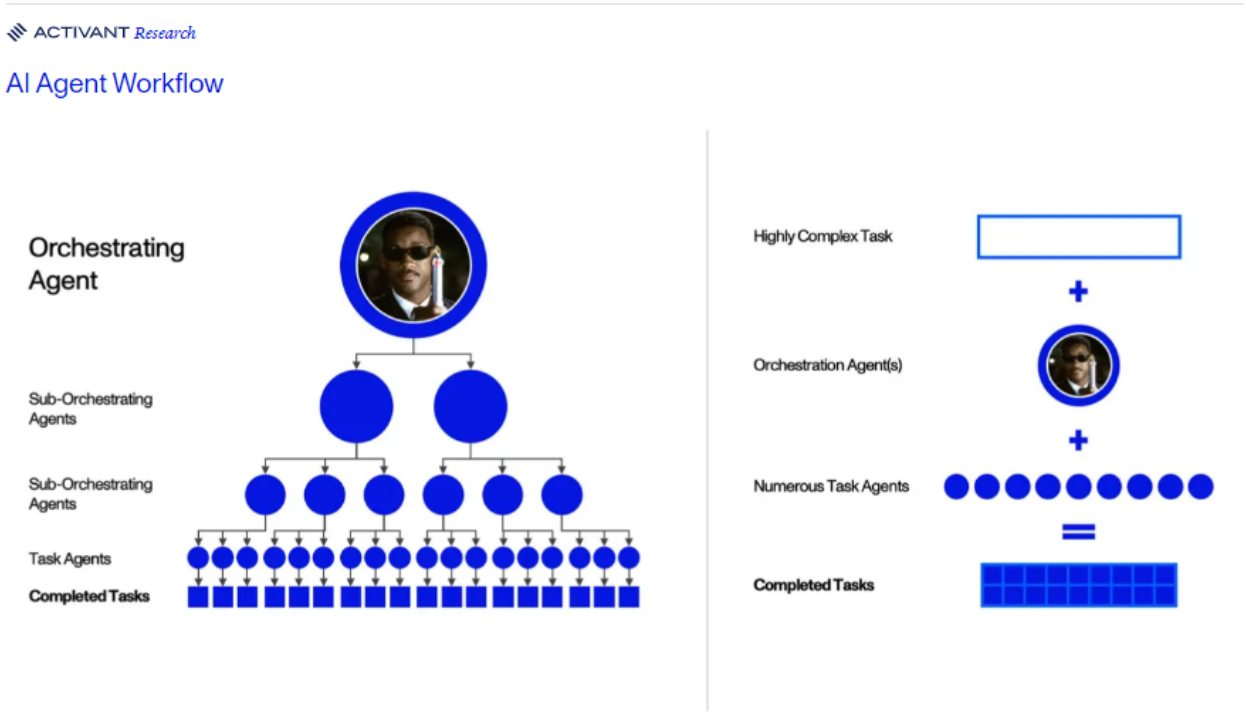
Quantifying the value-add of Multi-Agent Systems is a considerable challenge. The nuances of their applications – and the myriad ways they can be designed and built – mean the perceived benefits can vary widely. That said, there are distinct, universally applicable advantages that are evident even now, despite the infancy of the space.

Distributed Intelligence

MASs address a key shortcoming of ABSs, wherein the executing Agent in an ABS might become overwhelmed and falter when confronted with an exceedingly complex task. In an ABS, the first executing Agent is responsible for the detailed planning and implementation of extensive, high-level goals given by the "planning" Agent – a load that can cause the system to buckle. A MAS instead breaks down tasks across multiple Agents recursively, making them far more resilient.

In theory, you could envision an Agent structure that decomposes workflows and tasks into the smallest unit possible: a "task quantum." A multitude of Agents could then be tasked with handling a singular piece of the overall problem. By divvying up tasks in this way, it's possible for the system to tackle incredibly complex challenges, simply by having each Agent follow a clear, simple set of instructions.

Imagine an architect who designs a grand monument. While the architect can create the blueprint, they almost certainly lack the skills (and the time) to physically construct their vision. Instead, they delegate smaller tasks to teams of workers, who are managed by others experienced in construction tasks. **Just as the architect relies on numerous teams to execute their blueprint, a MAS distributes tasks to many Agents to achieve complex goals.** Each Agent, the equivalent of a bricklayer in our analogy, might not be able to plan and execute the entire task, but can perform its piece of the task – its task quantum – efficiently and effectively. This accumulation of simple tasks, handled by numerous Agents, ultimately accomplishes the overarching objective.



Scalability

As the complexity of a problem escalates, more individual Agents can be integrated into the system, rather than whole new Agent-based systems being added. Scalability also relates to the distributed nature of MASs, where tasks and decision-making processes are spread across multiple Agents. This distribution allows the system to handle larger and more complex tasks – without requiring a necessarily proportional increase in computational resources.

Parallelism

The power of MASs isn't just in distributing tasks. **They also benefit from enabling Agents to communicate directly, allowing them to optimize their own workflows.**

For example, as Agents work together on tasks, they share information on their successes or failures – creating a "knowledge network" that helps Agents refine their efforts. If one Agent encounters a setback, it alerts the others, enabling the collective to bypass strategies that have already proven ineffective and preventing duplicate failures.

Conversely, if an Agent finds a successful approach, it shares the insight with the rest, allowing them to focus on and expand this productive pathway. Through this exchange, groups of Agents can more efficiently navigate tasks, minimizing failures and maximizing productivity.

Decentralization

Decentralization is one of the key advantages of Multi-Agent Systems. While Agents in an ABS may be bound by centralized rules and workflows, MASs, by contrast, ensure Agents can survive independently from one another. Unlike in an ABS – where a single breakdown can cascade into a system-wide disruption – if an Agent fails in a MAS, the network will remain intact, simply reassigning that Agent's tasks.

Final Thoughts

As Agents take over these and many other aspects of our work, we see our roles transforming in exciting ways: instead of constructing market maps, we'll evaluate them; instead of building models, we'll test their assumptions; instead of writing memos, we'll discuss their implications. In our work, and in most fields, experience earned over years of apprenticeship won't be replaced, and human-to-human interaction will continue to be the most important driver of value.

At Activant, we are passionate about the Agent Ecosystem, and this work is only a subset of our research to date. For each of the components we discussed, there are additional considerations and frameworks when applying them to Multi-Agent Systems – everything from the use of Tools across multiple Agents at once to concurrency issues in shared working memory. Additionally, in developing enterprise-ready applications, a few more components are required, such as Agent Security, Compliance Management, Access Provisioning, and Agent Profile Management. On top of all that, we've begun diving into how Agents will be built in verticals like fintech, cybersecurity, commerce, and logistics.

If you're building in this space, thinking about how to approach using Agents in your organization or just looking to ideate – let's talk.

Special thanks to Teddy Cohen and Drew Peterson, Activant Fellows, for their research contributions.

Endnotes

1 Ironically, the brick layer will be among the last to be automated. The most automatable tasks are those that have the smallest action space, and while laying bricks may seem relatively simple compared to working in Excel, the reality is a 2-dimensional Excel Workbook has significantly fewer possible actions than a 3-dimensional job site exposed to the elements.

2 Note, this is similar to Lilian Weng's Agent System; however, our Ecosystem focuses on the investment landscape, rather than the implementation of Agents.

3 Fine-tuning as a Service is the process of training a pre-trained model on subject-matter-specific data to produce a model that performs better in specific use cases.

4 Agent libraries provide the orchestration framework which combines all the components in the Agent Ecosystem.

5 "APA PsycNet." American Psychological Association, American Psychological Association, psycnet.apa.org/record/1957-02914-001. Accessed 19 Sept. 2023

6 Gartner, Database Management Systems Market Transformation

7 Matplotlib is a graphing library written in the Python Language

8 Note that here we are approximating token count to word count which is not strictly accurate but expedient for the example. Additionally, we are leaving roughly 1 thousand tokens available for the response

9 To be clear, we find BabyAGI to be an instrumental step on the Agent journey, and our writing is merely to point out how it can be improved. This piece would likely not exist without it.