Chaps Vision

Trustworthy Agentic AI

Building Agentic Systems for the Enterprise

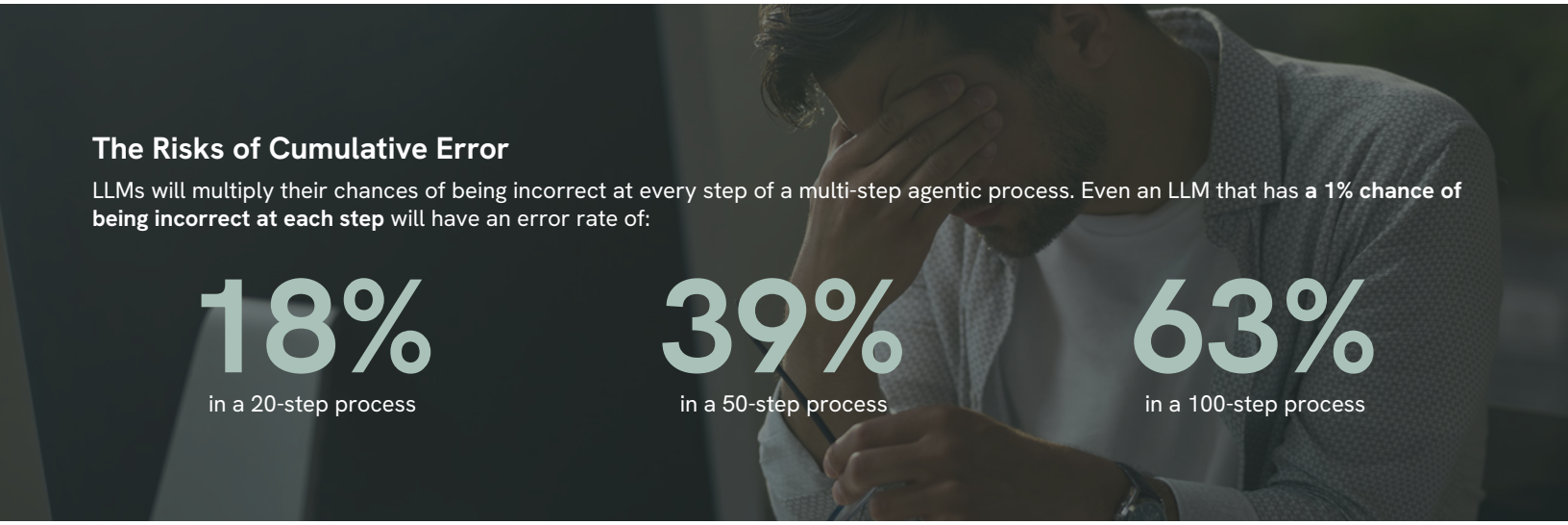
Table of Contents

Chaps Vision

Introduction

The more AI agents can handle, the more value they bring to the business. This creates a clear motivation to deploy agents for critical decisions, more often and with less oversight. However, scaling this value requires **reliable and autonomous** systems.

Today, agentic AI lacks this **dependability** because LLMs remain inherently unpredictable. While LLMs will continue to improve, there is no indication yet that hallucinations will be solved or even reduced to a level where AI agents can be trusted to control important tasks.



The Risks of Cumulative Error

LLMs will multiply their chances of being incorrect at every step of a multi-step agentic process. Even an LLM that has **a 1% chance of being incorrect at each step** will have an error rate of:

18% in a 20-step process

39% in a 50-step process

63% in a 100-step process

This means that in order to deploy agentic AI that can be trusted, the trust must come from something other than the LLM. There are three key components to trustworthy agentic AI:

1. Reliability

Ensuring that AI agents give accurate responses and successfully and consistently perform the tasks that they are given. Equally important is that they decline tasks they are not capable to perform, and catch when they have made mistakes.

Act – and let agents act – with confidence.

2. Security

Always enforcing proper access rights for sensitive or restricted information, so that is not accessed by any employee (or agent) that shouldn't have access.

Never worry about what agents might reveal.

3. Observability

Full transparency and traceability of all AI agent operations, with full monitoring of AI agent execution to detect undesired outcomes, particularly catching situations where AI agents interact with other AI agents in unpredictable ways.

Visibility to control every step and every action.

Managing an ecosystem of AI agents that are able to plan, act, and interact is largely nascent territory. Many vendors are putting together applications to create, deploy, and manage agentic AI. But since the market is immature, few offerings have all the capabilities necessary for trustworthy agentic AI. The essential capabilities for the three pillars of reliable, secure, and observable AI agents are described in the following sections.

Key Component #1

Reliable AI Agents

The first pillar for trustworthy agentic AI is reliability, which refers to the agent's ability to get things right consistently, and to properly handle instances when it cannot do so. Reliability is accuracy under all conditions.

Many factors contribute to reliability of an AI agent, including:

1. Content Comprehensiveness

Ensuring access to all enterprise knowledge.

2. Content Quality

Using high-quality knowledge and ignoring low-quality knowledge.

3. Retrieval Accuracy

Grounding the LLM with only the most relevant content.

4. RAG/LLM Integration

Optimizing the retrieval/LLM combination.

5. Guardrails

Controls to keep agent operation on track.

6. Scope Management

Clearly establishing scope and span of control.

7. Prompt Optimization

Designing prompts with high success rates across all conditions.

8. Reflection

A host of techniques used to check and validate response accuracy and detect errors.

9. Error/Failure Management

Handling refusals and failures, and managing errors.

Robust agentic AI platforms will have capabilities in most of these areas, but the capability within any particular area will range widely from platform to platform. For the most part, each of these factors is independent from the others, so they can be discussed and addressed individually without obsessing over how they relate. Each is discussed in more detail in the pages that follow.

Key Component #1

1. Content Comprehensiveness

The AI agent can only discuss topics that it has information about. If some internal systems or certain kinds of information are excluded, the agent cannot provide an informed response or may not be able to respond at all.

The foundation of an AI agent's value is its knowledge about the enterprise. It can't accomplish something if it knows nothing about it. Therefore, the AI agent must have comprehensive access to all of the content needed for it to achieve its goals. Most often this comes through connectors, which enable the retrieval system to search across:

All systems (standard enterprise repositories plus specialized)

All file formats (typical office formats plus uncommon formats, including compressed and encrypted)

All unstructured data (metadata, entire documents plus figures, attachments, etc.)

All structured data (databases and similar)

All modalities (text, diagrams, images, audio, video)

Including all **security and access control lists** for the content to ensure that data is protected at all times.

But not any content that should be excluded (e.g., masking out PII and other sensitive information).

2. Content Quality

Just as access to content is key, the LLM can only be as accurate as the knowledge it relies on. While good retrieval can mitigate many data quality problems, ensuring that information is ATP (accurate, trusted, and pertinent) and not ROT (redundant, outdated, or trivial) in the first place goes a long way to reliable responses and actions.

Some of the key capabilities here include:

- **Versioning**. The system should be able to handle versioning that is explicit (i.e., understanding the design releases of a part) and implicit (distinguishing the final contract from drafts and working copies).
- **Enrichment**. There are a host of techniques to enrich content to improve its quality for downstream use by LLMs, from automatically populating missing metadata to extracting acronyms to building a knowledge graph.
- **The LLM (and therefore the AI agent) is only as good as the information it has access to**. If the source content has errors or is out of date, that can cause the LLM to be inaccurate. However, good retrieval can often overcome data quality problems by ensuring that only current, high-quality information is sent to the LLM and any outdated or low-quality content is ignored (see the next section). As the volume of information grows in an organization, it rapidly becomes impractical to curate it manually. Therefore, retrieval is the preferred method to ensure high quality.
- **Authoritativeness**. Enterprise data landscapes are often messy, and it's common for the same or very similar documents to appear in multiple places. The system should know what sources or departments or individuals are the authorities for what kind of content should be used.

These capabilities (along with good retrieval) avoid a situation where companies must first get their data house in order before they use AI. Of course, rigorous curation would be the ideal – but is only practical at small scale (think: website content, HR documents, product design releases). As the volume of information grows, it rapidly becomes impractical to curate it manually, necessitating content enrichment and accurate retrieval.

Key Component #1

3. Retrieval Accuracy

Since search results are used to provide knowledge to the LLM that it will use to formulate its answer or guide its next step, **retrieval accuracy is the most important factor for accurate AI agents.**

- Information that is **relevant** provides the knowledge for the LLM to act. If the LLM only has facts that are sort-of related to the topic, it can't respond accurately and will make decisions based on the wrong knowledge. In business contexts, close is rarely good enough.
- If the results aren't **comprehensive**, then the LLM will be missing key knowledge. Without the full picture, it can't provide a complete and contextual response.
- Any information that isn't **focused** serves as a distraction and is likely to introduce inaccuracies or bring irrelevant information into the response (LLMs aren't great at recognizing or ignoring extraneous information).

Only when the results are **relevant, comprehensive, and focused**, is the LLM able to provide a high-quality, accurate response.

That's why it's important that the underlying RAG system uses intelligent hybrid retrieval, which combines traditional search techniques driven by statistics and text mining with deep learning techniques including vectorization and re-ranking. This approach achieves the most precise retrieval that better grounds LLM-dependent systems, and is covered in detail in Appendix B: The Five Kinds of Retrieval.

- If relevant knowledge is left out, then the LLM is missing key information
- Any knowledge that is relevant but not precisely to the topic at hand reduces the focus of the LLM risks diluting the response
- Including irrelevant knowledge is a distraction and is likely to introduce inaccuracies or bring irrelevant information into the response

Assuming that the knowledge needed is available, this is the **single most important determinant of an AI agent's accuracy**. There are a wide range of retrieval systems and various techniques to retrieve all of the relevant information while excluding information that isn't on target. It's important that the underlying RAG system blends several methods using intelligent hybrid retrieval. The considerations and capabilities of robust and accurate retrieval are not implementation details—they are architectural requirements that determine whether AI agents operate with precision or risk operating with false confidence.

4. RAG/LLM Integration

A good RAG pipeline is also critical to proper grounding of the LLM, and there is a wide range of RAG implementations ranging from naïve RAG to sophisticated RAG. A naïve RAG approach with a single search might work for a PoC, but performs poorly at scale, bringing in a lot of noise. RAG that employs multiple queries with query intent, tuned to the retrieval methods, and with intelligent scope management will produce high-quality results for almost any information need.

Assuming the right knowledge is available in the first place, this is the **second most important determinant of an AI agent's accuracy**, so detailed considerations and capabilities of the retrieval pipeline (RAG) are provided in Appendix C: Maximizing RAG Quality.

5. Guardrails

At their core, LLMs are not controllable or predictable, and even the best prompts do not always ensure that an LLM will stay on track or behave as desired. Therefore, agentic AI requires guardrails, which intercept undesirable inputs before they get to the LLM and undesirable outputs before they get to the employee (or the next AI agent).

There are two kinds, rules-based and LLM-based, and they can be used independently or together.

Rules-based: Rules are the most surefire way of protecting against undesirable outcomes, and are a must for any high-stakes applications. A robust rules-based system uses guardrails based on words and their forms, synonyms, phrases, and intent detection.

LLM-based: Because rules are rigid and can be difficult to deploy with the desired effect, using an LLM to apply guardrails can be a more effective way to handle semantically-oriented guardrails.

Since LLMs can interpret meaning, they do a better job of distinguishing between desired and undesired use; however they are also not 100% predictable, so they are not appropriate for situations where there is zero tolerance for error.

It's not enough to simply catch problems with guardrails. It's necessary to establish what should be done when a problem is identified. Should the request be refused? Attempted again? Reported to a system admin? A robust agentic AI platform will provide flexible options for handling and routing guardrail violations, as well as built-in monitoring over time.

Key Component #1

6. Scope Management

There are two main dimensions to scope management. One is defining scope in the first place, and the second is managing out-of-scope situations.

Defining Scope

The more narrowly focused the AI agent is, the lower the likelihood that the LLM will deviate from the task at hand. The key to narrowing scope is creating specialized agents that excel at doing one thing, and doing it very well. An AI agent framework should provide easy controls to narrow the scope of agents, and to do so in a flexible way.

For instance, it should be possible to easily constrain agents' access to information, tools, and other agents, and change this over time. Moreover, it should be possible to reuse agent patterns that are constrained dynamically to their individual scope.

For example, consider an agent that is designed to take action for a particular product. This action may also be needed for other products. Since a best practice is to narrow scope, it's better to have one agent per part than a single agent that attempts to work across all parts. It should be possible to create one agent and then replicate that agent to other agents, each one focused on a different part. This has the added advantage of easier maintenance; it's only necessary to change one agent and that change will propagate to the other product agents.

Managing Out-of-Scope Situations

The AI agent must be able to recognize when it's in over its head, and choose to abandon the task instead of trying to complete it, and make something up in the process. When a refusal or abandonment takes place, it must be handled properly so that the appropriate next steps are taken. Maybe the agent asks for the request to be rephrased, maybe it explores a different tool or another path before a final refusal. Specifying what an AI agent should do when it decides it can't comply or can't complete the task should be integral to any agent orchestration framework.

7. Prompt Optimization

Much has been made of so-called "prompt engineering," the art of crafting a prompt to coax the best possible response from an LLM. As models have improved, the criticality of the "perfect" prompt has diminished. Nonetheless, the quality of the prompt does affect the quality and consistency of the AI agent's performance over the various circumstances it may encounter.

The following functionality should be basic capabilities of any agentic AI system.

Prompt management. The ability to create, store, and use different versions of prompts, and give multiple employees access to prompts through a library.

Prompt selection. The ability to have different prompts optimized for different LLMs, so that if the AI agent chooses to use a different LLM, it can select a prompt tailored to that LLM.

Prompt optimization. Tools for testing and refining prompts include a testing environment, A/B testing tools, prompt versioning (for testing purposes only), and perhaps even automatic prompt optimization.

Also, keep in mind that prompts appear in multiple places in an agentic system and the capabilities mentioned above should be available for the various kinds of prompts (note: this list is representative, not exhaustive):

Initialization prompt	The prompt at the beginning of a session with an AI agent, that sets the tone and scope for the activity
Personalization prompt	A prompt use to personalize the activity, by providing contextual information specific to the employee (or AI agent)
Objective prompt	Prompts to describe overall objectives and under what circumstances control should be kept or delegated
Task prompt	Prompts used to specify how to complete a specific task
Tool prompt	Instructions for how to select tools, and descriptions of tools used for MCP servers
Agent prompt	Instructions for how to discover/select agents, and agent descriptions for A2A protocol

Chaps Vision

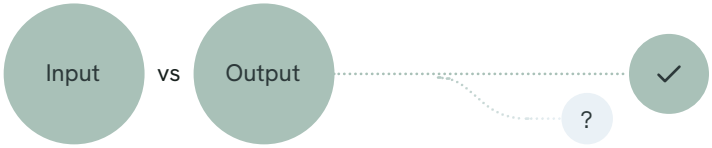
Key Component #1

8. Reflection

Reflection is a broad term referring to any technique used where an LLM reflects on the progress so far, and alters its path if needed. In other words, after drafting a response or planned action, it pauses to review the draft and decide if it is good, or should be altered. In many cases the activity of the AI agent or LLM is evaluated by a different LLM – (in this case, the other LLM does the reflection). If the activity of reflection reveals a problem or a potential issue, the reflecting LLM can change the path of execution. That change could be as simple as "try again," try again with a different agent or scope or tool, halt the process for human review, or abandon the effort altogether.

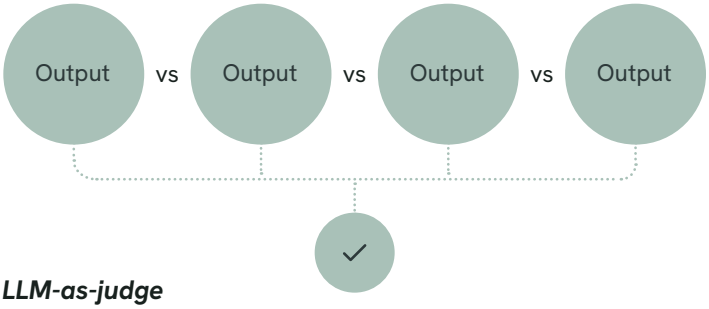
An agentic AI framework should provide easy use of reflection at any point in the execution pipeline as a step, a tool, and/or an agent.

There are multiple techniques, but the most common include:



LLM-as-reviewer

An LLM compares the input and the proposed output (and often the retrieved information that was used to ground the output) and determines if the output is appropriate or should be modified in any way. Common metrics include accuracy, completeness, attribution, context adherence, and instruction adherence.



LLM-as-judge

An LLM (usually a separate one) compares multiple generated outputs and picks the best one, or synthesizes multiple outputs into a consolidated response that contains the best elements of multiple responses.

9. Error/Failure Management

This category does overlap with others (guardrails, scope management, and reflection in particular) but is worth its own section because it's key to reliable agentic AI. The system must be robust to errors and failures throughout the execution of the agents. There are many considerations, here are two examples:

Error categorization: There needs to be a way to classify different kinds of errors. Since it's impossible to anticipate all the ways a system of agents might encounter problems, a classification system must be established so that AI agents know what kind of error they have encountered.

Error resolution: The appropriate action (retry, halt, human intervention, etc.) must be established for each error classification. Some of these may be rules-based, others will be LLM-based and carry their own instructions in the form of prompts.

Chaps Vision

Key Component #2

Secure AI Agents

The second key need for trustworthy agentic AI is to ensure that the AI agents are secure, and that all of the information they handle is kept secure. This is a complex challenge in a multi-agent system where knowledge may be requested from different agents at different contexts and for different purposes.

The Golden Rule:

The safest approach ensures that all of the actions taken by an AI agent are limited to the data accessible by the employee who initiated the agent's operation (or built the agent and set up a trigger that initiates the agent).

This can only be accomplished if the security permissions for the employee are enforced at every knowledge request and every handoff.

The Identity-Locked Handoff

Node 1:
Employee Initiator
(User ID & Group Permissions)

Node 2:
AI Agent A
(Requesting information)

Node 3:
AI Agent B
(Handling the request)

Node 4:
Tool/Data Source
(Validating permissions)

Multi-User Output Considerations

- If the AI agent is preparing something that will be seen by multiple employees, then the knowledge must be restricted to the minimum permissions of all employees involved.

This could result in:

- No allowed information (if access does not overlap), or
- Multiple outputs, each filtered according to the specific permissions of each employee

In secure enterprise systems, AI must dynamically tailor output to the access rights of each recipient.

The Enterprise Security Checklist:

Security is a multifaceted topic but the minimum capabilities for protecting sensitive internal data include:

Category	Requirement
Infrastructure	In cases where confidential or sensitive information cannot be shared with LLM vendors or cloud providers, the ability to run on-premises with on-premises/open source LLMs.
Identity Sync	Connectors that automatically collect permissions of all information they access, matching the existing permissions and security models of every source system.
User Database	A user database that understands the access controls for every employee (or group or department or...) that is authorized to create or use AI agents as well as anyone who is able to interact with or access the output of AI agents.
Agility	The ability to update access control lists as they change (without having to reindex that content).
Permission Propagation	A system that propagates permissions whenever information is utilized or requested, including: • From AI agents to tools • Between AI agents • From AI agents to employees
Default Closed Approach	No information is disclosed to any person or agent that shouldn't have access.
Selective Overrides	The means to selectively override the default permissions (in cases where special permissions are needed or correcting access in the source system is too difficult).
Restricted Discovery	The potential to selectively reveal that relevant information exists but access is restricted so that an individual can request access through proper channels.

Chaps Vision

Key Component #3

Observable AI Agents

The third and final component for trustworthy agentic AI is observability. Observability refers to the ability to monitor and trace the activity of all of AI agents that are deployed, individually and collectively. This is crucial because AI agents are often unpredictable and can operate autonomously, making it difficult to ensure they behave as intended without robust monitoring systems in place.

As agent ecosystems scale to hundreds or thousands of agents, the complexity and unpredictability of their interactions increase. Observability provides essential visibility into the steps agents take, the tools they use, and the metrics associated with their operations (such as number of steps, tokens used, elapsed time, and success rates). This visibility is necessary to detect anomalies, prevent runaway costs, and maintain trust in agent outcomes.

Agent observability enables organizations to:

- Track agent usage and performance metrics (e.g., success rate, latency, refusal rate)
- Monitor resource consumption and infrastructure load
- Receive real-time alerts for failures or suspicious activity
- Govern agent behavior and costs by setting enforceable limits
- Ensure security and compliance by tracing data access and propagation

Observability Factors

1. Agent Usage and Quality Metrics

Tracks agent activity and task success.

2. Agent Workflow Tracing

Tracks workflows within and between agents.

3. Employee Metrics

Identifies employee adoption and usage habits.

4. Infrastructure Metrics

Monitors resource usage and infrastructure demand.

5. Live Tracing and Monitoring

Provides real-time system activity visibility.

6. Alerts and Notifications

Flags failures and high-priority issues.

7. Model Selection and Utilization

Tracks model usage and associated costs.

8. Tool Utilization

Tracks tool usage and performance.

9. Sensitive Data Handling, Compliance Monitoring

Monitors sensitive data access and usage.

10. Security Monitoring

Ensures security policies are enforced.

Chaps Vision

Key Component #3

1. Agent Usage & Quality Metrics

Agent metrics give insight into the value, usage, and performance of each AI agent. Most metrics are viewed in aggregate, giving insight into executions over time for each agent (quantity, average, maximum, and minimum). Unusual results can give an indication that an agent might need to be changed, optimized, or even retired.

Some of the more common metrics include:

Usage	How often is this agent invoked?
Invoked By	What is invoking this agent – is it employees? Other agents? Which ones?
Latency	How long did it take for the agent to perform its task(s)?
Refusal Rate	How often does the agent refuse a request?
Refusal Type	Why did the agent refuse the request?
Success Rate	How often did the agent succeed? (This may be difficult to measure, depending on the agent and the task requested.)
Failure Type	Although failure information may not always be available, it is valuable to understand why the agent failed to accomplish its goal and make improvements.
Execution Delay	How much time the AI agent spent "waiting" for system resources because of overload or because of higher-priority activities.

2. Agent Workflow Tracing

Agent workflow tracing is the ability to track the steps taken by each agent, from invocation to completion, each time the agent executes. It is used for agent debugging, to identify where problems occurred so that they can be fixed. Although production systems may limit logging so as not to overwhelm the system (or the debuggers!), it must be possible to turn on all of the following metrics in order to trace an AI agent's execution and make improvements.

- **End-to-end workflow**: Complete trace of all steps executed by an agent.
- **Tool Use**: Number of times an agent calls each tool.
- **Agent Invocation**: Number of times an agent invokes another agent.
- **Invocation loops/successive invocations**: Number of times agents or agent groups are executing repeatedly to surface potential inefficiencies.
- **Source Utilization**: Number of times was each source system accessed via RAG.
- **Prompt Log**: Prompts sent to LLMs, tools, and agents, including enterprise knowledge retrieved by RAG.
- **Response Log**: Response returned by each LLM, tool, or agent (including reasoning where applicable).
- **Shared Knowledge Log**: Information passed between agents.

Key Component #3

3. Employee Metrics

User metrics include tracking how the various agents are used by employees over a certain period of time (daily, weekly, monthly) or on a cumulative basis. They are key to understanding user adoption and give insights into the usage and value of the system, not only by individual employees but also by groups and departments. They include:

Adoption: Number of unique employees

Stickiness: Number of repeat employees (e.g., monthly active users)

Value: Number of sessions per employee

Engagement: Number of interactions per session/dwell time

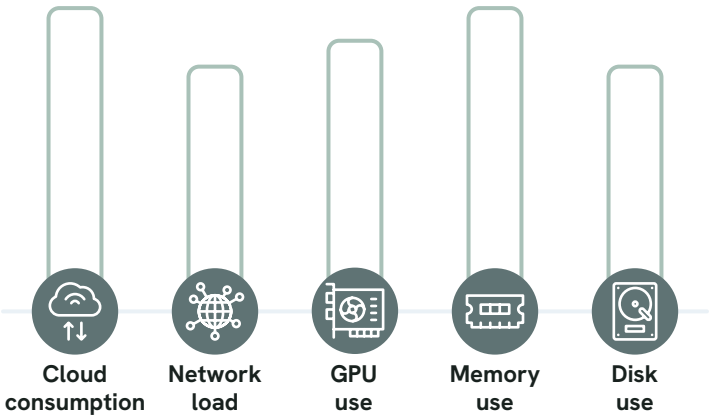
Satisfaction: Any explicit feedback metrics (e.g., thumbs-up vs. thumbs-down reactions)

Since most agentic AI systems allow for self-creation of agents, these metrics apply to the use of 1) the agent builder/orchestration framework, 2) invoking and engaging with shared agents, and 3) invoking and engaging with personal agents.

Not that not all of these metrics make sense for all aspects of an agentic AI system. For instance, dwell time is irrelevant if all the employee is doing is triggering an AI agent to accomplish a task; in that case there is no "dwell time." And none of these metrics apply to AI agents that are scheduled or invoked by triggers, instead of manually.

4. Infrastructure Metrics

Sometimes referred to as system health, infrastructure metrics answer the question: how much of the system's resources are the agents consuming? In addition to logging history, infrastructure metrics should be available in real-time. They include:



5. Live Tracing and Monitoring

While infrastructure metrics provide a way to observe the load on the system, they don't give insight into what is placing the demands on the system. That responsibility falls to live monitoring of what is going on in the agentic AI ecosystem, and is necessary for a full picture to see the demand (requests) and supply (infrastructure). Typical metrics to measure the demand coming from agentic AI include:

- Number of activated agents
- Number of active agents, with statistics on number of steps and elapsed time
- Active requests by LLM, with statistics on prompt size and elapsed time
- Number of active tools, with statistics on elapsed time
- Agent backlog (showing the number of agents waiting for higher-priority agents to release resources)
- Pending LLM requests (showing the backlog created by throughput limits)

6. Alerts and Notifications

When important processes fail, proactive alerting is necessary so that troubleshooting and intervention can occur immediately. If a critical agent can't complete its task, if a system goes offline, if throughput demands are stressing the system – these are situations where the right employees should be alerted so they can be addressed.

7. Model Selection and Utilization

Model selection and utilization tracks how frequently each model is invoked and how much each LLM is being used. This is particularly valuable when AI agents are able to choose which LLM to use for different tasks (e.g., GPT-4o vs. GPT-o3) or even which model providers (e.g., Gemini vs. Claude).

This information is typically provided at a global level, with the ability to filter to more granular views along several different dimensions, including by AI agent, by initiating AI agent, by employee, and by group or department.

Key metrics include:

- Model selection (number of times each model was chosen)
- Token consumption by model (tracking input tokens and output tokens separately)
- Latency by model

It's often important to know about geographical usage as well, so another dimension is tracking model geos, such as usage in the Americas vs. Europe.

Chaps Vision

Key Component #3

8. Tool Utilization

Tool utilization tracks tool usage, by both individual agents and globally, to provide insight into how tool selection and performance can be optimized. Metrics to track include:

- **Usage**: How often is this tool used?
- **Invoked by**: Which agents are using this tool?
- **Successive Invocation**: Is one agent repeatedly calling this tool over and over?
- **Latency**: How long did it take for the tool to complete its task?
- **Error Handling**: How often was the tool unable to provide a response, and why?

9. Sensitive Data Handling, Compliance Monitoring

If your agentic AI handles sensitive or regulated information, it's important to know how that information is flowing through the system and how it's used. The system must provide observability to identify and track the use of PII and other sensitive information, such as information marked proprietary or confidential, as it propagates throughout the ecosystem.

It's also essential that such information can be traced back to its original source – at what point did an AI agent obtain that information and from which source? That way it's possible to validate that the access was within bounds and verify compliance with any internal or external regulations.

As with LLM utilization, different geographies may have different constraints regarding the use of sensitive or personal data. The ability to trace information flow is critical for compliance with regulations like GDPR or CCPA, including information logged or stored for diagnostic or other purposes.

Tracing data from its source, to an agent, to tools, to and between other agents, and then ultimately written to systems or shared with employees can be quite complex, so an agentic AI framework must have rigorous tracing capabilities that can be configured to meet specialized needs.

10. Security Monitoring

It's important to ensure that information should only be accessible to those who have the appropriate permission to see that information. That means that the information used by AI agents throughout their execution, along with the access rights associated with that information, must be thoroughly observable through the entire workflow.

This is best done by comparing the access rights of the end user, the person who sees the information, with the access rights of all information accessed and obtained over the course of the AI agent's execution. Automating this process can be a challenge, but is essential to catch any information leaks.

If at any time the AI agent collects information beyond what the end user is authorized to access, then a security breach exists somewhere in the course of the agent's execution. Tracing such a breach will require complete visibility into the end-to-end process of the agent, including all tool calls, agent invocations, and information retrieved via RAG.

Governance

Establishing trustworthy agentic AI is critical to deploying AI agents in the workplace. But it isn't the only necessity, because even when there is confidence that the AI agents will be accurate and perform as desired, there are still operational factors that must be managed. That is the realm of governance.

Governance answers questions like "how do we control costs?" and "how do we ensure proper use?" The AI agent ecosystem must have monitoring and controls in place to meet operational objectives and constraints. This requires a layer of controls to govern the system. Observability is the key to informing governance, as it is the foundation for a clear view of what all the AI agents in the entire ecosystem are doing, so that controls can be adjusted as needed.

Governance of agentic AI is even more important – and more difficult – than governance of other applications. This is because unlike any other system, AI agents often choose what to do and can take action, making them at the same time more unpredictable and more impactful than the automated applications and systems used in the past.

It's usually not possible to govern the actions of AI agents directly (because they are not deterministic). Rather, it becomes necessary to establish and enforce limits in the event that something goes wrong. Or maybe in the event that something goes right, but it goes right in an unintended way.

Key Governance Factors and Their Descriptions

Governance Factor	Description
Token Consumption Quotas	Maximum tokens that can be consumed over a given time period
Token Throughput Limits	Limits on simultaneous model use
Agent/Tool Prioritization	Assign resource allocation, execution, or access priorities to agents and tools
Agent/Tool Toggling	Limit activity to authorized agents and tools at any given time period
Tool Use Limits	Maximum number of tool calls in a given time period
Tool Throughput Limits	Limits on simultaneous tool use
Budget Quotas and Assigned Consumption Values	A system that manages the costs of running AI agents based on the value those agents can create
Human Verification and Authorization	Defines the conditions and steps for human-in-the-loop, where a person must check or approve a suggested action before the AI agent can execute it

These limits must be enforceable at various levels, typically at the level of tools and agents, and potentially at the level of departments and agent groups in addition to the entire agentic AI ecosystem.

Governance Limits and Controls

1. Token Consumption Quotas

To control costs, budgets are typically allocated for LLM usage for a given time period as measured by token consumption (input and output tokens). Limits are typically set for:

- Overall consumption (by model)
- Department consumption (by model)

Limits may also be necessary by geography.

The system must provide warnings if the consumption trajectory is on course to reach the limit before the time period expires, so action can be taken to avoid hitting the limit. If the limit is reached, the system should refuse additional requests with an appropriate message delivered back to the originator (which could be an employee, an AI agent, or an employee that set up an autonomous or event-triggered AI agent).

When limits are hit, a system with good observability will identify which agents, tools, employees, or departments have the highest consumption.

2. Token Throughput Limits

If AI agents working in parallel exceed the throughput capability of the LLMs in use, latencies can balloon and halt all progress, or render the system unusable. There must be mechanisms in place for throttling LLM requests to prevent system overload/downtime.

3. Agent/Tool Prioritization

Some agents will perform more critical jobs than others, so systems can assign resource allocation, execution, or access priorities to specific agents and tools. The goal is to ensure that critical agents or tools receive the resources they need, especially under constraints such as limited compute or budget.

The system must be able to adjust priorities dynamically based on observability data—i.e., real-time monitoring of agent and tool activity across the organization

4. Agent/Tool Toggling

Organizations may need to enable or disable specific AI agents or tools at any given time. Toggling limits activity to only authorized agents and tools, ensuring that only approved components are active and able to perform actions within the system. Toggling is typically used to enforce security, compliance, or operational policies, and can be applied at the level of individual agents, tools, departments, or across the entire agentic AI ecosystem.

5. Tool Use Limits

Some tools may incur costs (either in real terms or in compute/resources) and limits are necessary to keep both in check.

6. Tool Throughput Limits

AI agents have the ability to perform actions (e.g., send an email) and request information (e.g., check calendar availability). Too many requests can overwhelm a system (like a DDOS attack) so there must be limits to ensure that tools do not push systems beyond their limits.

7. Budget Quotas and Assigned Consumption Values

The limits described above are sufficient for simpler deployments, but may be too simplistic for larger deployments. Often, there will be a need to allocate budgets that can weight consumption metrics based on qualitative goals. For instance, imagine an AI agent that has highly variable token consumption. It's impossible to predict ahead of time how many tokens that agent might consume, and charging the department for its usage after the fact means expenses are unpredictable (no one likes getting a surprise consumption bill, whether it's Azure credits or data charges on a cell phone). To address this concern, an agentic AI system needs to be able to assign costs to various components (say, $0.10 for Agent A and $0.30 for Agent B) and track usage of those agents against a budget (say, $100 per month for department C).

8. Human Verification and Authorization

An agentic AI system should be able to require a human to review a proposed action and either approve or reject the action before an AI agent proceeds. These human review points must be:

- Specified as a required step in any AI agent
- Specified as a required step in any tool
- Categorized to specify the kind of review required
- Review categories must be mapped to what roles can review the proposed action
- Queued for review by individuals with the matching role
- Assignable to other individuals (by name or by role)

Conclusion

While generative AI technologies are enabling the era of agentic AI, true success depends on grounding agents in organizational knowledge, ensuring security, and maintaining full observability. Trustworthy Agentic AI is built on the pillars of reliability, security, and observability. Reliable agentic AI consistently delivers accurate results and handles errors appropriately; security ensures strict access controls and data protection; and observability provides full transparency and traceability of agent actions.

ChapsVision's approach combines these elements to create agentic AI systems that are not only powerful and scalable, but also safe, ethical, and enterprise-ready, enabling organizations to confidently deploy AI agents for critical business tasks. To learn more about Sinequa, ChapsVision's platform to enable Enterprise Agentic AI, schedule a consultation today.

Chaps Vision

Sinequa
by ChapsVision

For more information, please visit

Enterprise Agentic AI platform

About Sinequa by ChapsVison
Sinequa transforms how work gets done by providing employees with knowledgeable, accurate, and secure AI Agents that streamline workflows, navigate complex enterprise data, and deliver reliable, traceable insights. By combining enterprise search with generative AI in a configurable, easily managed framework, Sinequa enables organizations to deploy out-of-the-box Agents or tailor specialized workflows, ensuring every interaction is trusted, governed, and compliant while empowering employees to focus on high-value work. For more information, visit **www.chapsvision.com/platform/sinequa/** or **www.sinequa.com**