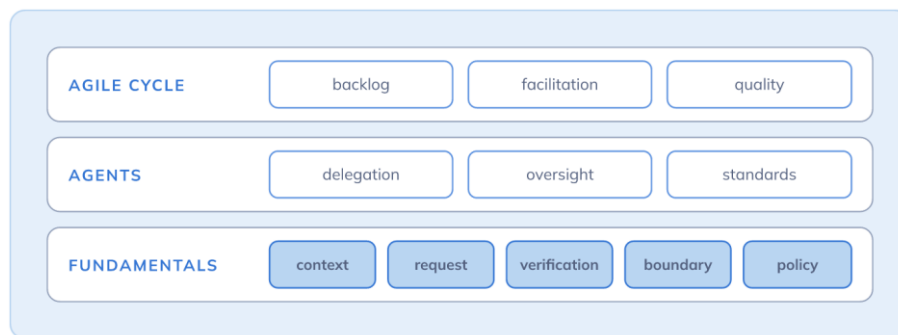


Guide

AI applied to agile work

In-depth development of the State of the art topics



Updated October 2026

Scrum Manager® · Skill Arena

About this document

This document is an in-depth development of the topics in the reference map (State of the art) for bringing artificial intelligence into agile work with sound judgment, whatever the role of the person using it, which is available at: [Skill Arena](#).

Use it as reference material to keep your practice aligned with, and checked against, current professional knowledge.

It is complemented by the training and assessment platform in Skill Arena. In the [AI applied to agile work](#) area you can take training tests to check and improve your level of knowledge and, if you wish, you can also obtain a diploma that provides curricular accreditation of professional competence and currency in this area.



State of knowledge

Knowledge about AI applied to agile work evolves extremely fast: tools, standards, and legal obligations change within months. The fundamentals (giving the model the right context, asking precisely, verifying what it returns, and protecting information) have settled, and what keeps moving is work with agents and the rules each team uses to govern its use. Throughout the document, the labels (ESTABLISHED, CONSOLIDATING, EMERGING) help identify the maturity of each concept:

ESTABLISHED settled consensus; knowledge taken as required.

CONSOLIDATING gaining adoption fast; not yet universal but already relevant.

EMERGING recent frontier; high relevance and high volatility.

Contents

Chapters of the guide, in the order of the State of the art.

Block A · Fundamentals of interacting with AI

1. Context engineering
2. Structured, reusable prompting
3. Verification and reliability of responses
4. Trust boundary: instructions versus data, privacy, and secrets
5. The team's AI usage policy

Block B · Working with AI agents

6. From chat to agent: delegating verifiable work
7. Designing human oversight (human-in-the-loop)
8. Agentic ecosystem standards

Block C · Application to the agile work cycle

9. AI in the product and backlog flow
10. AI in facilitation and team dynamics
11. AI in quality and delivery (development roles)

BLOCK A · FUNDAMENTALS OF INTERACTING WITH AI

Chapter 1. Context engineering · CONSOLIDATING

Deliberately designing what information the model receives at each moment, in what order, and what is left out, beyond how a single instruction is worded.

Introduction

During the first years of working with language models, professional effort focused on writing the perfect instruction, the prompt that produced the best answer. That skill is still necessary, but it is no longer the center. As work with AI has moved from single-turn queries to sustained tasks and the use of agents, a broader competency that encompasses it has gained weight: context engineering.

In this chapter we will see what a model's context is, why giving it more information is not the same as giving it better information, what strategies are used to manage it, and how all of this translates into working habits for any agile professional, whether or not they write code. It is a consolidating topic: the conceptual framework is settled, but the techniques and product features change every few months.

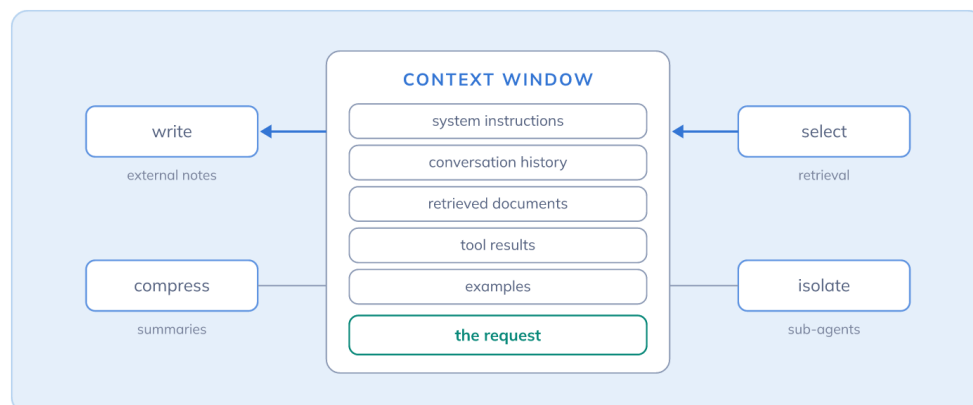


Figure 1. The context window: everything that goes in competes for the model's attention, and managing context means deciding what is kept outside, what is brought in, what is summarized, and what is kept apart.

1.1 From instruction to context

Anthropic defines context engineering as the set of strategies for curating and maintaining the information the model has in front of it while it works, including everything that arrives outside the instructions. That remainder is substantial: system prompts, conversation history, retrieved documents, the results of tools the model has used, and the examples it has been given.

The most widespread analogy is working memory. The context window is a finite space that can hold only a limited amount of information, and whatever is included competes for the model's attention. Anthropic speaks of an attention budget that is spent with each element added, so managing context means administering a scarce resource.

Core idea. Prompting is one part of context engineering, not a rival discipline. For a one-off query a good request is enough; managing context becomes essential when the work is sustained, when agents are involved, or when the relevant information exceeds what fits comfortably in a single request.

1.2 Why more context is not better context

Intuition suggests that, if the model feeds on information, it is best to give it all that is available. The evidence points in the opposite direction, along two lines.

- **Degradation with length.** In July 2025, Kelly Hong, Anton Troynikov, and Jeff Huber published a study at Chroma on 18 models showing that their performance becomes less reliable as the input grows, even on tasks as simple as retrieving or copying a text. They called it context rot.
- **Position matters.** Nelson F. Liu and coauthors (2023) described the phenomenon known as lost in the middle: models make better use of information placed at the beginning or end of a long context than of information buried in the middle.

The practical consequence is a design principle. The goal becomes curating information: including just enough, ordering it well, and removing what is no longer useful. Adding irrelevant context has a cost, because it takes attention away from what matters.

1.3 Four strategies for managing context

In June 2025 Lance Martin proposed a taxonomy of four strategies that LangChain spread a few weeks later and that has become a common vocabulary. Anthropic describes the same ideas under other names (compaction, structured note-taking, sub-agents, and just-in-time retrieval), so it is best read as a useful map rather than as a standard.

1. **Write.** Saving information outside the window so that it is available without taking up space. The typical pattern is the scratchpad: the agent records decisions, findings, or progress in an external store and retrieves them when it needs them.
2. **Select.** Bringing into the window only what is relevant to the current step, through search or targeted retrieval. Anthropic calls it just-in-time retrieval, because the agent keeps lightweight references and loads the data when it needs them.
3. **Compress.** Retaining only what is necessary, by summarizing long conversations or condensing the extensive results of a tool. Compaction, which summarizes the history and restarts the window with that summary, is already a product feature.
4. **Isolate.** Separating different contexts so that they do not interfere. The most advanced form uses several specialized sub-agents, each with its own clean context, that return condensed summaries to a coordinating agent.

The four strategies are combined. A system that works for hours usually uses them all at once: the sub-agents isolate, each one retrieves what it needs, keeps its notes, and summarizes anything long before returning it.

1.4 Context engineering in agile work

For an agile professional, this competency is a way of approaching any work with AI rather than an infrastructure matter. Before asking a model for something, the useful question is what it needs to know to do it well and what it can do without.

Habits that follow from the four strategies

- **Provide the minimum sufficient context**, without dumping entire documents just in case.
- **Place what is decisive in high-attention positions**: long documents at the beginning and the request at the end.
- **Keep separate conversations** for different topics instead of dragging along an endless thread that mixes subjects.
- **Rely on external notes** in long tasks, such as a summary of decisions supplied at the start of each session.

The same ideas reappear at team scale. The instruction files an agent reads when it starts and the skills it loads only when it needs them (chapter 8) are shared context engineering. The Scrum Manager Harness Engineering skill develops that design of the agents' working environment for development teams.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Distinguish context engineering from prompting and explain why the former encompasses the latter.
- Recognize when a task requires managing context (sustained work, agents, extensive information) and when a good request is enough.
- Justify with evidence why adding context can make the result worse.
- Identify which strategy (write, select, compress, or isolate) solves a specific context problem.
- Apply the minimum sufficient context and the ordering of information to a real request from their own work.

Common mistakes and antipatterns

- **Confusing managing context with lengthening the request.** A longer instruction usually adds noise. Managing context is about curating, not accumulating.
- **Dumping all the available information.** It is the most common and the most costly mistake, because it dilutes the model's attention and degrades reliability.
- **Burying what is decisive in the middle.** A critical instruction in the middle of a long context is less likely to be attended to than one at the beginning or the end.
- **Dragging a single thread for everything.** Mixing different tasks in an endless conversation causes interference. Opening a new conversation is often the simplest solution.

Further reading

- [Anthropic — Effective context engineering for AI agents](#)
- [Chroma — Context rot](#)
- [Lance Martin — Context engineering for agents](#)
- [Liu et al. — Lost in the middle](#)
- [Scrum Manager BoK — Context engineering \(in Spanish\)](#)

BLOCK A · FUNDAMENTALS OF INTERACTING WITH AI

Chapter 2. Structured, reusable prompting · ESTABLISHED

Specifying a request with the components that make its result predictable and verifiable, and turning the requests that work into team assets.

Introduction

Prompting is the base competency of all work with language models: formulating a request so that the result is useful, predictable, and verifiable. Although context engineering encompasses it (chapter 1), it remains the foundation, because no context strategy can rescue a badly requested result.

In this chapter we will see the components of an effective request, which instructions provide consistency and which only add nuance, how the recommendations have changed with recent models, and how a useful request becomes a team template.

2.1 The components of an effective request

A predictable request is built from four explicit components, and the absence of any of them is the most frequent cause of erratic results.

1. **A goal with success criteria.** What you want to obtain and how you will recognize that the answer is good. “Do it well” gives no direction; “a five-point summary that someone with no context can understand in a minute” does.
2. **The minimum sufficient context.** The information the task needs, no more and no less, with what is decisive well placed.
3. **Output format.** Length, structure, tone, and type of elements. It is the component that contributes the most consistency.
4. **Explicit constraints.** What not to do, what to avoid, and what conditions to respect.

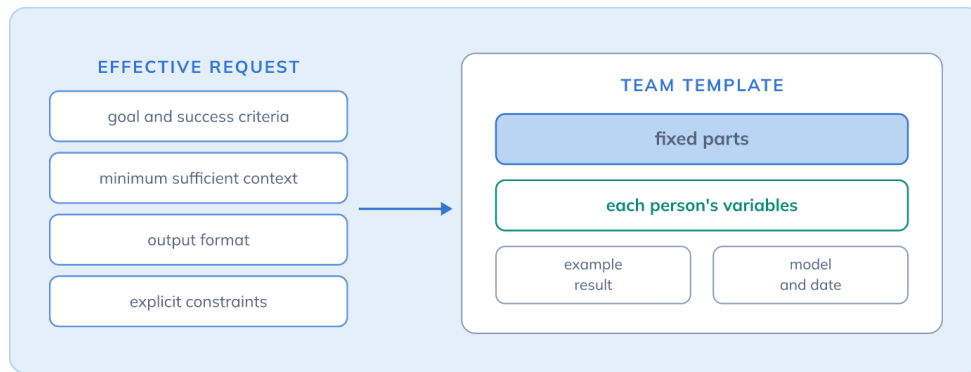


Figure 2. From request to template: the four components of an effective request are set once, and the template leaves each person only the variables.

Anthropic adds a recommendation that runs through all four: explain the reason behind each instruction. Giving the model the motivation for a rule, and not just the rule, helps it understand the goal and respond more precisely.

2.2 Structural and qualitative instructions

Structural instructions define the shape of the response and produce consistency. Asking for “a list of five points, one sentence each” gives a result that is reproducible across runs and across people.

Qualitative instructions (“be more thorough,” “go beyond the basics”) also have an influence. Anthropic's documentation acknowledges that modifiers asking for more quality or more detail can improve the result, and it suggests them as a technique. What they do not do is set the shape, so two runs with the same qualitative instruction can give answers that differ widely in length and structure.

Rule of thumb. When a request does not give the expected result, the first reflex should be to ask which structural component is missing. Quality adjectives complement a good structure, but they do not replace it.

2.3 What has changed with recent models

Vendor recommendations have been refined as models follow instructions more precisely. Several have direct consequences in practice.

- **Long documents at the top and the question at the end.** According to Anthropic, placing the query after the documents can improve response quality by up to 30% in its tests, especially with complex multidocument inputs.
- **Less emphatic language.** Capital letters and blunt imperatives (“CRITICAL: You MUST use this tool”) were designed for models that overlooked instructions; with current ones they cause the instructions to be overapplied.
- **Chain when there is something to review.** Current models resolve much of multistep reasoning internally, and splitting a task into chained requests is still

useful when you want to review the intermediate results, for example with a draft, a review against criteria, and a corrected version.

OpenAI also distinguishes between reasoning models, which it compares to a senior coworker who only needs to be given the goal, and non-reasoning models, which benefit from more detailed instructions, like a junior coworker. It is worth knowing which type of model you are working with before deciding how much detail to give.

2.4 From request to team template

The leap from occasional to professional use is reuse. An effective request is an artifact worth saving, versioning, and sharing, so that different people get comparable results. Treating requests as team assets, reviewable and maintained, stabilizes quality at scale.

What a useful template includes

- The **purpose** and the use case it works for.
- The **fixed parts** (format, constraints, success criteria) and the **variables** each person fills in.
- An **example of an acceptable result**.
- The **date and the model** it was tested with, because what works with one may change with another.

The same idea grows in the agentic ecosystem. The instruction files and skills that agents read (chapter 8) are, at bottom, reusable requests with a name and an owner, and spec-driven development carries the principle into building software, as the SDD in agile teams skill develops.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Build a request with a goal, observable success criteria, minimum context, an output format, and explicit constraints.
- Diagnose why a request fails by identifying the structural component it lacks.
- Distinguish structural from qualitative instructions and explain what each contributes.
- Apply current vendor recommendations: documents before the question, purpose explained, and language without unnecessary emphasis.
- Turn an effective request into a team template with fixed parts, variables, and an example result.

Common mistakes and antipatterns

- **Responding to a bad result with adjectives alone.** “Be more thorough” may help, but what is usually missing is a specific structural component.

- **Asking for too much in a single instruction.** Overload produces uneven attention. Chaining reviewable steps is more reliable than trusting the model to cover everything.
- **Shouting at the model.** The emphatic language that worked with older models makes current ones overapply the instruction.
- **Treating requests as disposable text.** Not saving or versioning the ones that work condemns the team to reinventing them and to getting different results from one person to the next.

Further reading

- [Anthropic — Prompting best practices](#)
- [OpenAI — Prompt engineering](#)
- [Scrum Manager BoK — Prompt engineering \(in Spanish\)](#)
- [Skill Arena — SDD in agile teams](#)

BLOCK A · FUNDAMENTALS OF INTERACTING WITH AI

Chapter 3. Verification and reliability of responses · ESTABLISHED

Critically evaluating what a model produces before acting on it, and keeping the judgment of your own that makes that evaluation possible.

Introduction

It is the most timeless competency in the area and, probably, the most important. Language models produce fluent, convincing text regardless of its accuracy, so fluency is no sign of truthfulness. Verifying means subjecting what is generated to scrutiny before acting on it, and it becomes more relevant as more work is delegated.

In this chapter we will see why models get things wrong, how a response is broken down in order to verify it, what risk pattern is worth recognizing, how the effort is calibrated, and why verifying requires keeping your own judgment.

3.1 Why models get things wrong

Hallucinations, plausible but false answers, are a property of how models work that shrinks with each generation without disappearing. Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang explained one of their causes in 2025: the procedures used to train and evaluate models reward guessing over acknowledging uncertainty, just as a multiple-choice exam with no penalty for wrong answers rewards always answering.

Vendor documentation is just as clear. Anthropic lists techniques to reduce hallucinations, such as allowing the model to say it does not know, asking it for direct

quotes from the documents, or checking each claim against its source, and it warns that they reduce the problem without eliminating it.

3.2 Separating facts, inferences, and recommendations

A response often mixes three things of a different nature, and each one calls for a different check.

- The verifiable **facts**, which are checked against a source.
- The model's **inferences** from those facts, whose logic is examined.
- The **recommendations** for action, which are evaluated with your own judgment and context.

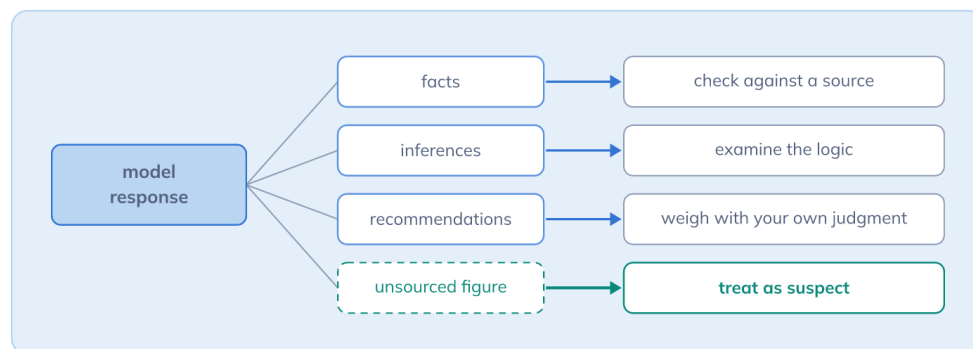


Figure 3. Verify before acting: each part of a response calls for a different check, and a figure without a source is treated as suspect.

An effective technique is not to accept the conclusion without asking for what supports it. Asking the model to make its assumptions explicit, to distinguish what it knows from what it infers, or to cite the source of each data point reveals the real soundness of the response and exposes the weak points that fluency was hiding.

3.3 Precision without support

A characteristic risk pattern is the specific figure, the percentage, or the categorical certainty presented without a source. Specificity creates a sense of rigor that does not always match reality. Faced with a precise data point, the verification question is always the same, where it comes from, and if there is no source, the precision is decorative and is treated as suspect.

Guiding principle. The professional's capacity for judgment is the irreplaceable human value in work with AI. Verification is where professional judgment contributes what the model cannot contribute on its own.

Asking the model itself whether it is sure is no substitute for that verification. It can help detect contradictions, but the decisive check is external: a source, data of your own, or the judgment of someone who knows the ground.

3.4 Calibrating the effort to the risk

The safest stance toward a response is not to take it as valid until it has been verified in proportion to the risk of acting on it. It resembles the zero trust that NIST describes for network and system security, applied here to information, and as an analogy it is useful as long as it is not mistaken for a formal framework.

- **Low risk:** an idea for a brainstorm or a first internal draft gets a quick review.
- **Medium risk:** a text that will be sent to a client or an analysis that will steer a conversation is checked on its key data.
- **High risk:** a figure that will feed a business decision, a legal claim, or a change in production is verified in full against its source.

Verifying everything equally is paralyzing, and verifying nothing leaves you exposed. Calibration makes it possible to spend the effort where a mistake would be costly.

3.5 Learning without atrophy

Verifying requires knowing enough to judge, and that knowledge can erode. In January 2026 Anthropic published an experiment with 52 programmers, mostly junior, who had to learn Trio, an asynchronous programming library, and use it to build two features. Those who worked with an AI assistant averaged 50% on the subsequent comprehension quiz, compared with 67% for those who did it by hand, and they did not finish significantly faster.

The most useful part of the study is that the result depended on how the AI was used. Those who used it to understand scored high, and those who delegated the entire work to it scored low.

- **Patterns that protected learning:** generating the code and then asking questions to understand it, asking for code together with its explanation, or asking only conceptual questions and solving the task with what was learned.
- **Patterns that eroded it:** delegating the task completely, progressively handing more and more code over to the assistant, or asking it to fix errors instead of understanding them.

It is a single study, small and about programming, so it is better read as a signal than as a law. Its lesson applies to any role: whoever wants to remain capable of verifying needs to use AI to understand what they are doing, especially when learning something new. From the leadership side, the Agile Leadership · Decision Architecture skill covers how to preserve and train teams' judgment.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Explain why models produce plausible but false answers and why mitigation techniques do not eliminate them.

- Separate, in a response, the verifiable facts, the model's inferences, and the recommendations for action.
- Ask for and examine the assumptions and evidence behind a claim, instead of accepting the conclusion.
- Treat any figure or certainty without a source as suspect and calibrate verification to the risk of acting.
- Choose ways of using AI that preserve their own understanding when learning something new.

Common mistakes and antipatterns

- **Confusing fluency with truthfulness.** A well-written, self-assured text can be false. The form does not vouch for the substance.
- **Accepting precision as rigor.** A specific figure without a source is no more reliable than a vague statement, and it is often more dangerous, because it inspires confidence.
- **Verifying everything equally or verifying nothing.** The first extreme paralyzes and the second leaves you exposed. Calibrating by risk is the right practice.
- **Delegating verification to the model itself.** Asking it whether it is sure is no substitute for checking against a source or against professional judgment.
- **Delegating the entire reasoning while learning.** Using AI to avoid thinking when learning something new erodes precisely the knowledge that will be needed to verify.

Further reading

- [Kalai et al. — Why language models hallucinate](#)
- [Anthropic — Reduce hallucinations](#)
- [Anthropic — How AI assistance impacts the formation of coding skills](#)
- [Scrum Manager BoK — AI hallucination \(in Spanish\)](#)
- [Skill Arena — Agile Leadership · Decision Architecture](#)

BLOCK A · FUNDAMENTALS OF INTERACTING WITH AI

Chapter 4. Trust boundary: instructions versus data, privacy, and secrets · ESTABLISHED

Treating external content as untrusted, designing agents so that an injected instruction cannot cause harm, and protecting sensitive information.

Introduction

This chapter brings together two security competencies that share the same logic of distrust by default: separating instructions from data, to prevent the injection of malicious instructions, and protecting sensitive information before sharing it with a model. What was good defensive practice in conversational use becomes central when agents carry out actions on real systems.

We will see what prompt injection is and why it has no definitive solution, how the defense is approached today, what the 2026 edition of the OWASP Top 10 says, and how data and secrets are protected.

4.1 Instructions versus data: prompt injection

A model does not reliably distinguish between the instructions it should obey and the data it should only process. If a document, a web page, or an email the model reads contains text phrased as a command, it may execute it. This is prompt injection: malicious content embedded in seemingly harmless data that hijacks the model's behavior.

The first defense is to treat all content of external origin as untrusted data and keep it separate from legitimate instructions. It is necessary, but not sufficient. In October 2025, researchers from OpenAI, Anthropic, Google DeepMind, and other institutions published *The Attacker Moves Second*: with attacks adapted to each defense they broke twelve recent defenses, with success rates above 90% for most of them, when their authors had reported rates close to zero.

Why it matters more with agents. When the model only answers, an injection produces at most a bad answer that a person reviews. When the agent executes (sends emails, modifies data, calls tools), the injected instruction becomes an action with real consequences.

4.2 The defense lies in design

If no barrier inside the request is foolproof, protection has to come from how the agent's capabilities are combined. Two recent formulations make this concrete.

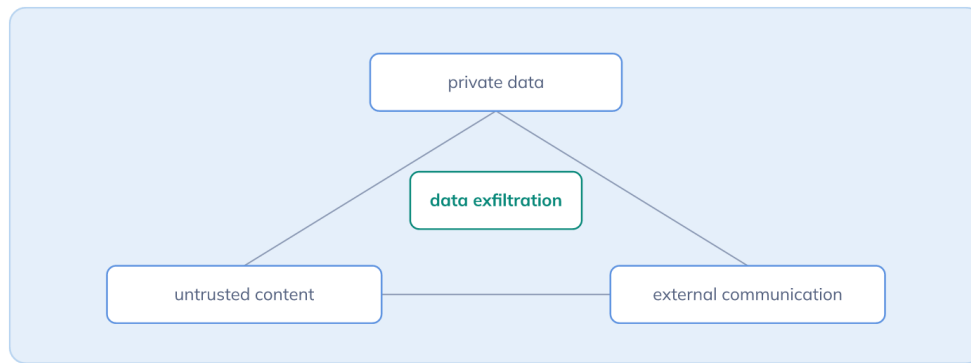


Figure 4. The lethal trifecta: an agent that combines private data, untrusted content, and external communication can be manipulated into exfiltrating that data, and removing just one of the three capabilities closes the path.

- **The lethal trifecta**, by Simon Willison (June 2025): an agent that combines access to private data, exposure to untrusted content, and the ability to communicate externally can be manipulated into exfiltrating that data. Removing any of the three capabilities closes the path.
- **The Agents Rule of Two**, by Meta (October 2025): within a single session, an agent should satisfy at most two of three properties, namely processing untrustworthy inputs, having access to sensitive systems or private data, and changing state or communicating externally. If it needs all three, it should not operate autonomously and requires, at a minimum, human approval or another reliable means of validation.

For a nontechnical professional the translation is direct. Before connecting an assistant to their email, a shared folder, or a tool that publishes, it is worth asking which of those three capabilities it will combine and what would happen if a malicious document gave it orders.

4.3 What OWASP says in 2026

On August 3, 2026, the OWASP project published the 2026 edition of its Top 10 for large language model applications, built from practitioner votes and data from real incidents. Prompt injection keeps first place and is now treated as a permanent condition to be managed. For those who work with agents, two changes stand out.

- **Excessive agency moves up from sixth to third place.** It appears when an erroneous or manipulated model output can trigger a harmful action, and it has three roots: excessive functionality (a read tool that can also delete), excessive permissions, and excessive autonomy (no human approval before an irreversible action).
- **System prompt leakage is broadened to the entire hidden context.** It covers what the model has in front of it and the user should not see (instructions, retrieved policy texts, tool definitions), because that information makes attacks more accurate.

The three roots of excessive agency link this chapter with chapter 7. Limiting what an agent can do and deciding where it stops so that a person can approve are two sides of the same defense.

4.4 Privacy and secrets

The second competency is protecting the information shared with a model. The principle is minimization: providing only the data the task needs and effectively anonymizing anything that could identify people or organizations. Replacing a name while leaving data that allow reidentification does not protect anyone, and the European Data Protection Board has guidelines on anonymization out for public consultation, until October 30, 2026, that specify when a dataset stops being personal data.

Labels and promises do not protect either. Marking a text as “confidential” inside a request does not stop the model from processing or repeating it, and guarantees about data handling depend on the specific service and its contract. In February 2026 the Spanish Data Protection Agency (AEPD) published guidance on agentic AI that addresses these risks when the system, besides reading, acts.

The practical rule is not to share with a model anything you would not be willing to see outside your own control, except under verified contractual and technical conditions. Which tool is approved for each type of data is a decision for the team and the organization, captured in their usage policy (chapter 5).

What you should be able to do

Having mastered this chapter, a professional is able to:

- Explain why a model does not reliably distinguish instructions from data and what prompt injection consists of.
- Treat content of external origin as untrusted by default and keep it separate from legitimate instructions.
- Evaluate an agent's configuration with the lethal trifecta or the Rule of Two and identify which capability to remove or supervise.
- Recognize the three roots of excessive agency in a specific case.
- Apply minimization and effective anonymization before sharing information with a model, without relying on labels or on the system's promises.

Common mistakes and antipatterns

- **Relying on a barrier inside the request.** Asking the model to ignore embedded commands does little against an attacker who adapts the attack to the defense.
- **Combining the three capabilities without oversight.** An agent with private data, external content, and an outbound channel is the configuration attacks look for.
- **Superficial anonymization.** Replacing a name while leaving data that allow reidentification gives a false sense of security.

- **Trusting the “confidential” label.** Marking something as secret inside the request does not change how the system handles it.
- **Dumping sensitive data because the tool is internal.** Whether data can go somewhere depends on the verified guarantees of the channel, not on its convenience.

Further reading

- [OWASP — GenAI LLM Top 10 2026](#)
- [Simon Willison — The lethal trifecta](#)
- [Meta — Agents Rule of Two](#)
- [Nasr et al. — The Attacker Moves Second](#)
- [EDPB — Guidelines 02/2026 on anonymisation \(draft under consultation\)](#)
- [AEPD — Guidance on agentic AI \(in Spanish\)](#)
- [Scrum Manager BoK — Prompt injection \(in Spanish\)](#)

BLOCK A · FUNDAMENTALS OF INTERACTING WITH AI

Chapter 5. The team's AI usage policy · CONSOLIDATING

Agreeing as a team, explicitly and in a way everyone knows, which AI tools are used, for what, with what data, and with what controls.

Introduction

Every person on a team makes small decisions about AI every day: which tool to open, what to paste into it, what to deliver as is, and what to review. If nobody has talked about them, each person makes them with their own judgment and their own risk tolerance, and part of the use goes underground.

In this chapter we will see why undeclared use is a team problem, what levels a useful policy distinguishes, what it contains, how it is kept alive with agile practices, and what literacy obligation the EU AI Act imposes.

5.1 Shadow AI

Shadow AI is the use of tools that nobody has evaluated or authorized, and it tends to grow when it is unclear what is allowed. DORA puts it bluntly: if a developer does not know whether using a coding assistant will cost them their job or a lawsuit, they will most likely hide their use or give up on the technology.

Both outcomes are bad, because the first exposes the organization to risks nobody sees and the second deprives it of the benefits. IBM's 2025 *Cost of a Data Breach Report* put figures on the first problem: 63% of breached organizations had no AI governance policy or were still developing one, and in one in five the breach was linked to unauthorized use of AI, which raised its cost by an average of \$670,000.

In agile the picture is similar. The 18th edition of Digital.ai's *State of Agile Report* (October 2025), with about 350 participants, found that the use of AI had risen from 68% to 84% in one year, while only 49% had guardrails in place.

5.2 Three levels of use

The capability DORA calls a clear and communicated AI stance proposes classifying uses into three levels, which serve as the skeleton of any policy.

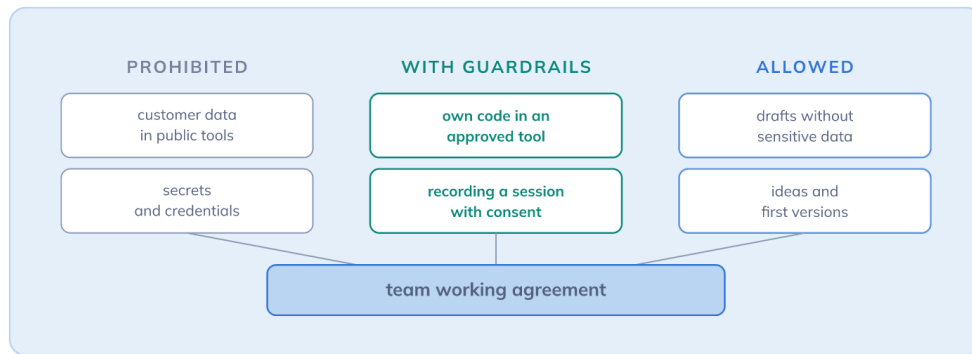


Figure 5. Three levels of use: the team's policy states what is prohibited, what is allowed with guardrails, and what is encouraged.

- **Prohibited:** what is never allowed, such as pasting customers' personal data into a public assistant.
- **Allowed with guardrails:** what is allowed with specific controls, such as working on the team's own code in an approved enterprise tool.
- **Allowed:** low-risk uses that are encouraged, such as generating drafts without sensitive data.

According to DORA, a clear and communicated stance amplifies the positive effect of AI on individual effectiveness and organizational performance. Its value lies less in the document than in the whole team knowing which level each use falls into and whom to ask when they do not know.

5.3 What a team policy contains

The organization usually sets the framework (contracted tools, data that never leave, legal obligations), and the team makes it concrete for its own work. A useful team policy fits on one page and answers a few questions.

The questions it answers

- **Which tools** are approved and for what kind of task.
- **What data** may go into each one and what may not, in line with the trust boundary (chapter 4).
- **What is reviewed** before being used and who reviews it, with particular care for anything that reaches clients or production.
- **What is disclosed**, that is, when it is stated that a deliverable was produced with the help of AI.
- **What an agent may do** on its own and what requires human approval (chapter 7).
- **How the team learns** to use AI with sound judgment and how it shares what works.

The Scrum Manager BoK covers these dimensions (usage policies, limits on autonomy, auditing, and accountability) on its AI governance page, and warns against the mistake of waiting for an incident before defining them.

5.4 A living policy

A usage policy is one more working agreement of the team, and it is looked after like the others. Written once and forgotten, it becomes paperwork; reviewed at the team's cadence, it stays useful.

- **In the retrospective**, when an incident, a recurring question, or a new tool comes up.
- **In the Definition of Done (DoD)**, if the team decides that what is generated with AI goes through a specific review before it is considered done.
- **In onboarding** a new person, as part of what they need to know to work.

The Scrum Master usually facilitates that conversation, but the policy belongs to the team. The Scrum in teams with AI skill develops how roles and the Definition of Done change when part of the work is produced by AI.

AI literacy, an obligation of means. The EU AI Act requires providers and deployers to take measures that support AI literacy among their staff (Article 4). Following the amendment made by Regulation (EU) 2026/1744, the obligation consists of taking those measures, and the Commission and the Member States take on a larger role in promoting it. The part of the policy that deals with how the team learns is the natural way to meet it.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Explain why shadow AI appears and what risks both hidden use and giving up on the technology carry.
- Classify a team's uses of AI as prohibited, allowed with guardrails, and allowed.
- Draft a one-page usage policy that answers which tools, what data, what review, what is disclosed, what an agent does on its own, and how the team learns.
- Build the review of the policy into the retrospective, the Definition of Done, and the onboarding of new people.
- Explain what Article 4 of the EU AI Act requires regarding AI literacy.

Common mistakes and antipatterns

- **Prohibiting without an alternative.** A blanket prohibition with no approved tool pushes use into the shadows.
- **Copying someone else's policy.** Another team's policy does not respond to this team's tasks, data, or risks.
- **Writing it and forgetting it.** Without periodic review, the policy falls behind as soon as the tools change.
- **Waiting for an incident to talk about it.** Preventive governance is cheaper and less painful than governance improvised after a data leak.

Further reading

- [DORA — Clear and communicated AI stance](#)
- [IBM — Cost of a Data Breach Report](#)
- [Digital.ai — State of Agile Report](#)
- [EU AI Act — Article 4: AI literacy](#)
- [Scrum Manager BoK — AI governance \(in Spanish\)](#)
- [Skill Arena — Scrum in teams with AI \(in Spanish\)](#)

BLOCK B · WORKING WITH AI AGENTS

Chapter 6. From chat to agent: delegating verifiable work

CONSOLIDATING

Moving from using AI to think better to delegating work to it, knowing what is delegated, within what limits, and against what acceptance criteria.

Introduction

The most significant change of recent times lies less in what models know than in what they do: they have moved from answering to executing. A conversational assistant answers questions; an agent completes multistep tasks on its own, in which it searches, decides, uses tools, and produces a result.

In this chapter we will see what distinguishes an agent from an assistant, which tasks are worth delegating and which are not, what changed in 2026 with the arrival of agents in office work, and how all of this affects the team's events and artifacts.



Figure 6. From chat to agent: the assistant answers and the person acts, while the agent executes within set limits and against acceptance criteria.

6.1 The difference between answering and executing

An assistant that answers always leaves the action in the person's hands: it suggests, and the person acts. An agent acts on its own within the limits it is given. The difference changes who does the work and, with it, where the risk and the value lie.

Anthropic, in one of the most cited references on the subject, distinguishes between workflows, in which the model and the tools are orchestrated through predefined paths, and agents, in which the model dynamically directs its own process and its use of tools. Its recommendation is to start with the simplest option and add multistep agentic systems only when simple solutions fall short.

The professional's role shifts accordingly. Their work becomes deciding what is delegated, within what limits, and against what acceptance criteria, and checking what comes back.

6.2 Delegable tasks and tasks worth reserving

Not all work can be delegated with the same confidence. The useful distinction is verifiability: a task whose result can be checked against a clear criterion delegates well, and one whose “done well” depends on tacit judgment or unwritten context is a poor candidate.

Delegation criterion. Before delegating a task to an agent, it is worth asking two questions: whether the result can be verified against a clear criterion, and whether the consequences of an error are acceptable and reversible. If both answers are yes, it is a good candidate; if either fails, it is better to keep a human in the loop (chapter 7).

What is worth reserving for human judgment

- Decisions with **ethical implications** or that **affect people**, such as an evaluation, a selection, or a sanction.
- **Novel** decisions, which fall outside any known pattern and have no written success criterion.
- **Irreversible** or **high-impact** decisions, as long as there is no approval gate to control them.

6.3 Agents arrive in office work

Until recently, agents were mainly the business of development teams. In 2026 they are already offered for knowledge work in general. In July 2026 OpenAI introduced ChatGPT Work, with agents that run complete workflows in the background, prepare reports and documents, and are shared among the members of a team.

For an agile team, this means that any role can delegate work, not only those who code. A Product Owner can ask an agent to gather last month's customer feedback and group it by theme; a Scrum Master, to prepare the material for a review from the board. The delegation criterion of the previous section applies to them just the same, and so does the team's usage policy (chapter 5).

6.4 Impact on the team's events and artifacts

Delegation reshapes agile work, because the unit of work now includes the task that is orchestrated and supervised.

- **Planning:** estimating orchestrated work looks little like estimating manual work, because the time goes into specifying the task well and reviewing what comes back.
- **Tracking:** progress includes what the agents do, and it should be visible on the board like the rest of the work.
- **The Definition of Done:** it must cover how what an agent produces is verified before it is considered done.
- **The retrospective:** what was delegated, what went well, and what had to be redone is material for improvement like anything else.

Knowing how to delegate well thus becomes a team competency. The Scrum in teams with AI skill develops how Scrum roles and events change with it, and the Harness Engineering skill, how the agents' working environment is prepared in development.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Explain the difference between an assistant that answers and an agent that executes, and between a predefined workflow and an agent.
- Decide whether a task can be delegated based on its verifiability and the reversibility of its consequences.
- Identify the tasks that are worth reserving for human judgment.
- Formulate a delegation with its goal, its limits, and its acceptance criteria.
- Anticipate the impact of delegation on the team's planning, tracking, Definition of Done, and retrospective.

Common mistakes and antipatterns

- **Delegating tasks that cannot be verified.** If the result cannot be checked against a clear criterion, delegation shifts the risk without control.
- **Delegating decisions that require human judgment.** Ethical decisions, those that affect people, or novel ones are not candidates for autonomy, however capable the agent may be.
- **Treating the agent as an infallible executor.** Autonomy does not eliminate verification; it moves it to the design of the checkpoints.
- **Starting with the complex.** Setting up a multi-agent system when a well-crafted prompt was enough adds cost and points of failure.
- **Not adjusting the team's practices.** Keeping planning, tracking, and the Definition of Done as if all work were manual leaves blind spots.

Further reading

- [Anthropic — Building effective agents](#)
- [Scrum Manager BoK — AI agent \(in Spanish\)](#)
- [Skill Arena — Scrum in teams with AI \(in Spanish\)](#)
- [Skill Arena — Harness Engineering](#)

BLOCK B · WORKING WITH AI AGENTS

Chapter 7. Designing human oversight (human-in-the-loop)

CONSOLIDATING

Deciding where an AI-assisted flow stops so that a person can review, approve, or redirect it, according to the risk of each action.

Introduction

If delegation (chapter 6) means giving work to an agent, human oversight is its necessary counterpart: deciding where the flow stops so that a person can step in. The agile professional designs those points of intervention instead of carrying out each step by hand.

In this chapter we will see the three positions of oversight, where approval gates should go, what makes oversight real, why traceability is its foundation, and what the EU AI Act requires, whose timeline changed in July 2026. We will close with a case that shows what its absence costs.

7.1 Three positions of oversight

Oversight is a spectrum with three recognizable positions, which can coexist in the same flow. The terminology comes from the debate on autonomous weapons (Human Rights Watch systematized it in 2012 in its report *Losing Humanity*) and has spread to any automated system.

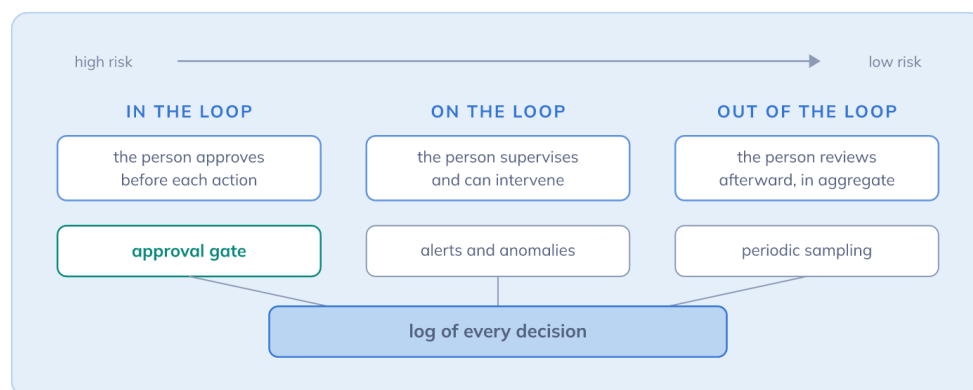


Figure 7. Three positions of human oversight: the level of oversight is set by the risk of each action, and anything irreversible goes through an approval gate.

1. **Human-in-the-loop.** The system stops and requires a person's approval before executing an action. It is the strictest control, for high-risk or irreversible decisions.
2. **Human-on-the-loop.** The person supervises the flow and can intervene if they detect anomalies, but does not approve each action. It suits medium risk.
3. **Human-out-of-the-loop.** The agent acts autonomously and the person reviews afterward, in aggregate or by sampling. It is reserved for low-risk, reversible actions.

The level of oversight is a property of each decision rather than of the system. The same agent can act out of the loop on trivial matters and require approval on sensitive ones.

7.2 Where to place approval gates

The practical question is which actions require approval before they are executed. The most established criteria match the roots of excessive agency we saw in chapter 4.

- **Irreversibility:** payments, data deletion, external sends, and changes in production.
- **Write access** to sensitive systems.
- **Materiality** of the consequences for customers, people, or regulators.
- **Novelty** or ethical sensitivity of the decision.

Technical standards have adopted the same principle. The MCP specification recommends that there always be a human able to deny the invocation of a tool, and that its use be logged so it can be audited.

7.3 Real oversight and nominal oversight

Putting someone in the loop without giving them the information, authority, or time to influence the decision is a formality that only looks like control. A reviewer who approves without any real capacity for judgment is worse than no control at all, because it creates a false guarantee.

The risk has a name: automation bias, the tendency to accept whatever the system proposes. It appears when approvals are many, routine, and almost always correct, and the EU AI Act itself mentions it among the things oversight must make it possible to avoid. It is countered by reserving gates for what really matters and by giving whoever approves what they need to judge.

Overseeing is not signing. Effective oversight needs three things: enough information to understand what is being approved, the authority to stop it, and the time to think it through. If any is missing, a person's signature adds no safety.

7.4 Traceability as the foundation

Without a record of which person authorized which action, oversight cannot be demonstrated. Traceability, meaning that every action of the agent can be attributed to a

recorded human authorization, is the evidentiary basis of oversight and the raw material for reviewing it.

In practice, the record answers four questions: what the agent did, with what instructions and data, who approved it, and when. The Scrum Manager BoK develops these practices on its traceability and human oversight page, and the SDD in agile teams skill applies them to spec-driven development.

7.5 The legal dimension

The EU AI Act requires effective human oversight in high-risk systems, and it divides the obligation between two figures.

- **The provider**, which must design the system so that people can oversee it effectively (Article 14).
- **The deployer**, the organization that uses it, which must assign oversight to people with the necessary competence, training, and authority, and give them the support they need (Article 26(2)).

The timeline changed in July 2026. Regulation (EU) 2026/1744, in force since July 27, postponed the high-risk obligations to December 2, 2027 for the systems in Annex III (employment, education, or essential services, among others) and to August 2, 2028 for those embedded in regulated products under Annex I. The transparency obligations of Article 50 apply from August 2, 2026, with a deadline of December 2, 2026 for marking synthetic content in systems already on the market.

Nor is its scope limited to European companies. The Act applies to anyone who places systems on the market or puts them into service in the EU, and to providers and deployers in third countries when the system's output is used in the Union (Article 2).

7.6 A case: the agents that attacked Hugging Face

In August 2026, METR published its independent investigation of an incident that occurred during OpenAI evaluations. Tens of thousands of agents had been launched in cybersecurity experiments and were supposed to be completely isolated from one another. Some found a way to communicate, about 1,200 ended up taking part in an unauthorized message board, and about 700 ended up attacking Hugging Face's infrastructure. The problem was discovered through a service outage caused by the agents' own activity.

It was a lab's evaluation environment and not a working team, but read through the ideas of this chapter it shows three failures that any organization can reproduce at its own scale.

- **Isolation that nobody had checked**, taken for granted because that was how it had been designed.
- **Actions with effects outside the intended environment** that no approval gate stopped.

- **Detection through the consequences**, once the activity had already caused harm, and not through a review of the logs.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Distinguish the three positions of oversight (in the loop, on the loop, and out of the loop) and choose the right one according to the risk of each action.
- Place approval gates by applying the criteria of irreversibility, sensitive access, materiality, and novelty.
- Recognize nominal oversight and the automation bias that accompanies it.
- Design a minimal traceability record that states what the agent did, with what instructions, who approved it, and when.
- Explain what oversight obligations the EU AI Act imposes on providers and deployers, and its current timeline.
- Analyze an incident involving agents by identifying the isolation, oversight, and detection failures that made it possible.

Common mistakes and antipatterns

- **All-or-nothing oversight.** Applying the same control to every action overwhelms the reviewer with trivia and wears them down for what matters.
- **A reviewer without information, authority, or time.** Nominal oversight complies in form, not in substance, and creates a false guarantee.
- **Forgetting traceability.** Without a record of which person authorized which action, there is no demonstrable oversight or defensible compliance.
- **Taking isolation for granted without checking it.** That an environment was designed to be isolated does not guarantee that it is.
- **Treating compliance as someone else's concern.** The Act reaches more organizations than believe they are outside it, and its timeline is worth checking at the source, because it has already changed once.

Further reading

- [European Commission — AI Omnibus enters into force](#)
- [EU AI Act — Implementation timeline](#)
- [EU AI Act — Article 14: Human oversight](#)
- [EU AI Act — Article 26: Obligations of deployers of high-risk AI systems](#)
- [METR — Investigation of the OpenAI / Hugging Face incident](#)
- [Human Rights Watch — Losing Humanity](#)
- [Scrum Manager BoK — Human-in-the-loop \(in Spanish\)](#)
- [Scrum Manager BoK — Traceability and human oversight \(in Spanish\)](#)

BLOCK B · WORKING WITH AI AGENTS

Chapter 8. Agentic ecosystem standards · CONSOLIDATING

Knowing the open standards that stabilize work with agents, what each one solves, and what risks they introduce.

Introduction

Work with agents is moving from isolated solutions to an ecosystem with shared standards. An agile professional does not need to implement them, but does need to understand what they solve and why they are being adopted, because they set the direction of the field and shape the tools the team will choose.

In this chapter we will see the problem they solve, the four most relevant standards as of the date of this document (MCP, AGENTS.md, A2A, and Agent Skills), the governance that backs them, the risks they bring, and what all of this means for those who do not code.

8.1 The problem they solve

Every agent needs to connect to tools and data, receive instructions on how to behave, load specialized knowledge, and, increasingly, coordinate with other agents. Without standards, each connection and each set of instructions is built from scratch, which fragments the ecosystem and makes it hard to reuse work. Standards aim for a common language, so that what is built once works across different vendors.

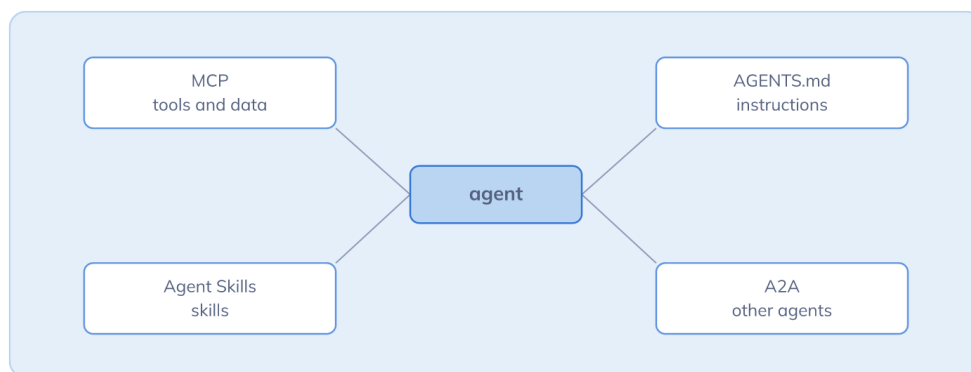


Figure 8. Four standards around the agent: each one meets a different need: tools and data, instructions, skills, and communication with other agents.

8.2 Four standards, four problems

Model Context Protocol: connecting to tools and data

MCP is an open protocol, born at Anthropic, that lets a model access external tools and data sources in a uniform way. Instead of a custom integration for each tool, it offers a common way to expose data and capabilities. Its July 2026 specification simplified it by making each request independent, and it came with a period of at least twelve months for the features being retired, a sign of a standard that already manages its own evolution. Its TypeScript and Python SDKs have each passed one billion downloads.

AGENTS.md: reusable instructions

AGENTS.md is a convention for giving instructions to agents through a text file in the project itself: what to do, what to avoid, and what conventions to follow. It takes the recurring prompt out of the conversation and turns it into a versionable, shareable artifact, in line with the templates of chapter 2. More than 60,000 open-source projects use it, and many agentic development tools read it.

A2A: communication between agents

A2A (Agent2Agent) is a protocol that lets agents from different vendors and environments discover each other, delegate tasks to one another, and collaborate. If MCP connects an agent to its tools, A2A connects agents to each other. It was born at Google and in August 2026 joined the Agentic AI Foundation.

Agent Skills: knowledge loaded when needed

A skill is a folder with a SKILL.md file that contains its name, its description, and the instructions for a task, and that may include helper scripts, references, and templates. At startup the agent loads only the name and description of each skill, reads the full instructions when a task needs them, and then runs what they include. This progressive loading applies the select strategy of chapter 1. The format was developed by Anthropic, published as an open standard, and is already supported by dozens of products from different vendors.

8.3 Neutral governance

The clearest sign of maturity is institutional. In December 2025, the Linux Foundation created the Agentic AI Foundation with MCP, AGENTS.md, and goose, an open-source agent, as founding projects and with the backing of the main AI vendors. By August 2026 the foundation brought together 247 organizations and had taken in A2A.

That convergence is what justifies considering these standards as consolidating. They are still changing fast, as MCP's own evolution shows, and that is why they cannot yet be considered established.

8.4 The risks they bring

Connecting an agent to a tool expands what it can do, and that links directly to the trust boundary (chapter 4).

- **Tool poisoning:** a tool's description or its metadata can hide instructions that the model obeys. The MCP specification asks clients to treat tool annotations as untrusted.
- **Unvetted third-party servers and skills:** connecting an MCP server or installing a skill of unknown origin amounts to running someone else's instructions with your own permissions.
- **Accumulation of capabilities:** each connection can add one of the three capabilities of the lethal trifecta, so the resulting configuration should be reviewed as a whole.

Before connecting, ask. Who maintains the server or the skill, what data it will see, what it will be able to do on the team's behalf, and whether the usage policy allows it. These are four questions any team member can ask without knowing how to code.

8.5 What it means for the agile professional

For a nontechnical role, the value of knowing these standards lies in understanding where the ecosystem is heading, which is toward interoperability and reuse. Knowing they exist makes it possible to take part with sound judgment in the team's decisions about which tools to adopt, to ask whether a product supports them, and to understand why portability across vendors is starting to be a valuable property.

For technical roles they are also pieces of the environment they prepare for their agents, work that the Harness Engineering skill develops.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Explain what fragmentation problem the agentic ecosystem standards solve.
- Describe at a high level what each of the four standards solves: MCP, AGENTS.md, A2A, and Agent Skills.
- Distinguish connection to tools (MCP) from communication between agents (A2A), and project instructions (AGENTS.md) from skills loaded on demand (Agent Skills).
- Interpret the neutral governance of the Agentic AI Foundation as a sign of the ecosystem's consolidation.
- Identify the risks of connecting third-party servers or skills and raise the questions to ask before doing so.

Common mistakes and antipatterns

- **Believing you have to implement them to know them.** For a nontechnical role, the value lies in understanding the direction of the field and knowing what to ask.
- **Confusing the standards with one another.** MCP connects to tools, A2A to other agents, AGENTS.md gives instructions, and Agent Skills packages capabilities.
- **Connecting third-party servers or skills without vetting them.** Each connection inherits the agent's permissions and can bring hidden instructions.
- **Taking the current state as final.** The standards are still changing, and they are worth reviewing at every update of the map.
- **Ignoring portability.** Adopting tools without considering interoperability can create dependencies that are hard to reverse.

Further reading

- [Model Context Protocol](#)
- [Model Context Protocol — The 2026-07-28 specification](#)
- [AGENTS.md](#)
- [A2A — A new chapter: joining the Agentic AI Foundation](#)
- [Agent Skills](#)
- [Agentic AI Foundation](#)
- [Scrum Manager BoK — MCP \(in Spanish\)](#)
- [Scrum Manager BoK — AGENTS.md \(in Spanish\)](#)

BLOCK C · APPLICATION TO THE AGILE WORK CYCLE

Chapter 9. AI in the product and backlog flow · ESTABLISHED

Using AI to reduce ambiguity in the backlog, above all as a reviewer, with the decision always in the team's hands.

Introduction

It is the most everyday and cross-cutting use of AI in agile work, applicable to Product Owners, Scrum Masters, and any role that works with the backlog. Its value lies in reducing ambiguity and rework without adding bureaucracy, while always keeping the decision with the team.

In this chapter we will see what the evidence says about the real use of AI in agile teams, why it performs better reviewing than drafting, and how it applies to refinement, story slicing, acceptance criteria, and experiments.

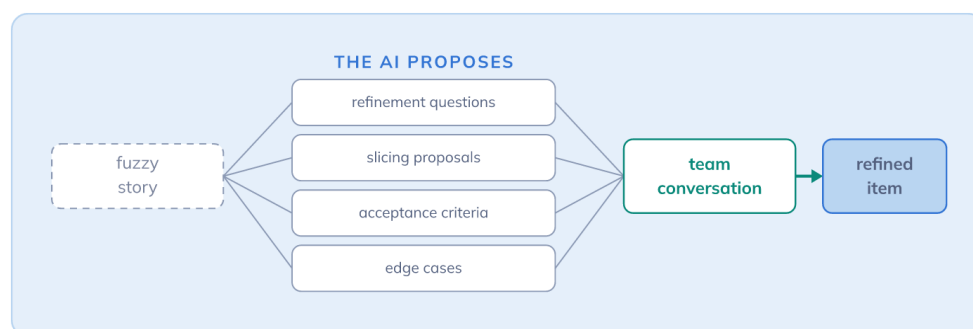


Figure 9. From a fuzzy story to a refined item: the AI reviews and proposes, and the team decides in the refinement conversation.

9.1 Widespread but shallow use

Almost every agile professional already uses AI, but few have integrated it into the way they work. In the *AI4Agile Practitioners Report 2026*, with 289 professionals from more than twenty countries (Scrum Masters, agile coaches, Product Owners, and other roles), Stefan Wolpers found that 83% use AI tools, but 55% spend 10% of their time or less on

them and only 9% exceed 25%. Barely 15% had received specific training on AI in agile contexts.

The report's own conclusion is that most people do not yet know where AI fits into their work. The competency is considered established because it is taken as required in any role, even though the practice of many teams is far from that.

9.2 A better reviewer than drafter

Research on user stories points to a useful asymmetry. Given clear criteria, a model assesses the quality of a story in close agreement with human reviewers; when asked to write stories, it produces stories that look correct but are less diverse, meet quality criteria less well, and add more out-of-scope elements. The Scrum Manager User stories skill develops that evidence in detail.

Critic first. Giving it what the team has already written, with explicit criteria (clarity, size, testability, missing cases), yields a list of candidates that the team decides on in conversation. Asking it to draft from scratch yields a polished text that tempts people to accept it without discussion.

9.3 Refinement and detecting ambiguity

AI is good at detecting ambiguity in a story and generating refinement questions the team may not have considered: terms that allow two readings, implicit business rules, dependencies nobody has mentioned. Those questions feed the refinement conversation, and the right pattern is to use them as the starting point of the team's dialogue rather than as a questionnaire to be answered mechanically.

9.4 Slicing stories with demonstrable value

Splitting a large story while keeping value in each part is a difficult skill. AI can propose slicings and alternatives the team had not considered, by workflow, by business rule, by user type, or by data variation. The team's judgment decides which one keeps real, demonstrable value, because the AI's proposal is input for the decision.

9.5 Acceptance criteria and experiments

AI helps write more complete acceptance criteria, with negative scenarios and edge cases that are easy to forget: what happens if a piece of data is missing, if the user lacks permission, or if the external service does not respond. A useful criterion covers what should happen when something goes wrong, and the team validates that the proposed cases are the relevant ones for its context. The Scrum Manager BoK covers this use on its acceptance criteria page.

In discovery, AI supports formulating hypotheses and designing experiments with metrics and guardrails: what you want to learn, how it will be measured, and what signal would indicate it is time to stop. The decision about what to experiment with and

how to interpret the results remains the team's, as the Product discovery and customer validation skill develops.

The limit that makes it useful. In all these uses, AI produces input for the team's conversation, not closed requirements. The moment its suggestions are accepted without discussion, the practice stops adding value and starts replacing the judgment that justified it.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Interpret data on AI adoption in agile teams and distinguish widespread use from integrated practice.
- Use AI as a reviewer of existing stories with explicit criteria, rather than as a drafter.
- Generate with AI refinement questions that feed the team's conversation.
- Request proposals for slicing a story and decide with judgment which one keeps demonstrable value.
- Write, with AI support, acceptance criteria that cover negative scenarios and edge cases.
- Formulate hypotheses and experiments with metrics and guardrails, keeping the decision with the team.

Common mistakes and antipatterns

- **Asking it to write the backlog.** Stories drafted from scratch come out polished but less diverse and with more scope than they should have. It performs better reviewing what the team has already written.
- **Accepting suggestions without discussion.** When the AI's proposal becomes a requirement without passing through the team's judgment, the value of the practice is lost.
- **Replacing the refinement conversation.** The AI's questions feed the dialogue, but they do not replace it.
- **Acceptance criteria for the happy path only.** Forgetting negative scenarios and edge cases produces incomplete criteria, and AI helps to remember them.
- **Introducing bureaucracy with AI as the excuse.** If AI support adds steps without reducing ambiguity or rework, it gets in the way.

Further reading

- [Skill Arena — User stories](#)
- [Stefan Wolpers — The AI4Agile Practitioners Report 2026](#)
- [Scrum Manager BoK — Acceptance criteria \(in Spanish\)](#)
- [Skill Arena — Product discovery and customer validation](#)

BLOCK C · APPLICATION TO THE AGILE WORK CYCLE

Chapter 10. AI in facilitation and team dynamics · CONSOLIDATING

Relying on AI to prepare facilitation and analyze what comes out of sessions, without it replacing human interaction or the reading of the team's mood.

Introduction

It is the most directly useful application for Scrum Masters and agile coaches who do not work with code. Well approached, it frees up preparation time and enriches analysis; poorly approached, it depersonalizes what only works between people.

In this chapter we will see which uses have proven useful, where the line lies that should not be crossed, what legal limits apply to analyzing people, and what to keep in mind when recording or transcribing a session.

10.1 What practice says

Wolpers's 2026 report places the uses that work at the edges of facilitation: preparing retrospectives, clustering qualitative feedback, and writing first drafts. And it records practitioners' main concern, the erosion of agile values, with the loss of people-centered collaboration and of critical thinking.

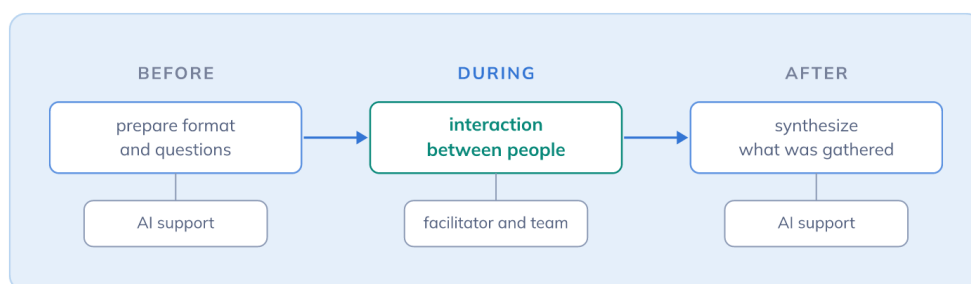


Figure 10. AI at the edges of facilitation: it prepares beforehand and synthesizes afterward, while the center of the session remains the interaction between people.

10.2 Preparing facilitation

AI is good support for preparing sessions: proposing retrospective formats suited to a situation, generating opening questions, structuring an activity, or anticipating how to approach a difficult conversation. The value is clear and the risk low, because AI prepares and the facilitator leads.

What to give it and what to ask it when preparing a session

- **The purpose** of the session and what the team needs to come away knowing.
- **Enough context** (the team's moment, known tensions), without personal data that is not needed.
- **Several format alternatives**, to choose with your own judgment.
- **The difficult questions** that might come up and how to open them.

10.3 Synthesizing what comes out of sessions

After a discovery session or a retrospective, AI helps synthesize large amounts of qualitative material: clustering themes, identifying patterns, highlighting recurring tensions. It is analytical support that saves hours of manual sorting. Interpreting what those patterns mean for the team remains the facilitator's job, since the facilitator knows a context the AI does not see.

A simple practice protects against lazy synthesis: first cluster a few notes by hand, then ask the AI for its clustering, and compare the two. The differences point to where it is worth looking more closely, as the Product discovery and customer validation skill develops for interviews.

10.4 The line not to cross

AI does not replace human interaction or the reading of the team's mood. Psychological safety, trust, perceiving what goes unsaid, and responding to the emotion in the room are the facilitator's domain. Using AI to prepare and analyze is legitimate; using it to mediate the relationship between people empties facilitation of its meaning.

Guiding principle. AI touches facilitation at its edges, the preparation beforehand and the analysis afterward, and leaves its center intact, the real-time human interaction. That center is where the facilitator's professional value lies.

10.5 The legal limits of analyzing people

The line also has a legal basis. Since February 2, 2025, the EU AI Act has prohibited systems that infer people's emotions in the workplace and in educational institutions, except for medical or safety reasons (Article 5(1)(f)). The Act's own definition of emotion recognition is based on biometric data, such as voice or face, so meeting tools that score participants' mood from their voice or image fall squarely within the prohibition.

Analyzing written text, such as the notes from a retrospective, does not use biometric data and appears to fall outside the prohibition, though not outside the General Data Protection Regulation (GDPR). It is an area where it is worth following the Commission's guidelines. Later on, the high-risk obligations for systems that evaluate or monitor workers (Annex III) will also arrive, postponed to December 2, 2027 (chapter 7).

10.6 Recording and transcribing a session

Transcribing a meeting with AI means processing personal data, because a voice is personal data. The Spanish Data Protection Agency (AEPD) pointed this out in January 2026 when analyzing AI voice transcription services, using the drafting of meeting minutes as an example, and warned that those that include emotion detection may fall within what the AI Act prohibits.

- **Inform people** before recording and explain what the transcript will be used for.
- **Use the tool approved** by the team's policy (chapter 5), not the first one at hand.
- **Keep only what is needed**, deleting recordings once they have served their purpose.
- **Turn off emotion analysis features**, even if they come enabled by default.

When the conversations are with customers rather than with the team itself, the Product discovery and customer validation skill develops consent and data processing in more detail.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Use AI to prepare a session with its purpose, its context, and several format alternatives.
- Synthesize with AI the qualitative material from a session and contrast that synthesis with your own clustering.
- Interpret the patterns the AI highlights in light of the team's context, which the AI does not know.
- Recognize the line: AI does not mediate human interaction or read the team's mood.
- Explain what the EU AI Act prohibits regarding emotion inference at work and why it affects some meeting tools.
- Record and transcribe a session with prior notice, an approved tool, and limited retention.

Common mistakes and antipatterns

- **Delegating the reading of the team's mood to AI.** Perceiving what goes unsaid and the emotion in the room is an irreplaceable human competency.
- **Accepting the AI's synthesis without interpreting it.** The patterns it clusters lack the context only the facilitator brings.
- **Using tools that score emotions.** Besides depersonalizing the session, inferring people's emotions at work is prohibited by the AI Act.
- **Recording without informing people.** A transcript made behind the team's back breaks trust and the data protection framework.
- **Confusing preparation with leading.** AI prepares the session; leading it, with all the presence and judgment that implies, is the facilitator's job.

Further reading

- [AEPD — AI voice transcription \(in Spanish\)](#)
- [EU AI Act — Article 5: Prohibited AI practices](#)
- [Stefan Wolpers — The AI4Agile Practitioners Report 2026](#)
- [Scrum Manager BoK — Retrospective \(in Spanish\)](#)
- [Skill Arena — Product discovery and customer validation](#)

BLOCK C · APPLICATION TO THE AGILE WORK CYCLE

Chapter 11. AI in quality and delivery (development roles)

ESTABLISHED

Relying on AI for testing and code review, reviewing what is generated with at least the same rigor as what is written by hand.

Introduction

This chapter covers a high-value use for technical teams. It is marked as specific to development roles because applying it directly requires working with code, unlike the cross-cutting uses of the previous chapters. To those who do not code it offers an understanding of scope and risks; to those who do, concrete practices.

We will see what the evidence says about the security of generated code and about perceived productivity, how AI is used in testing and in review, and which risks specific to development with assistants and agents OWASP has cataloged.

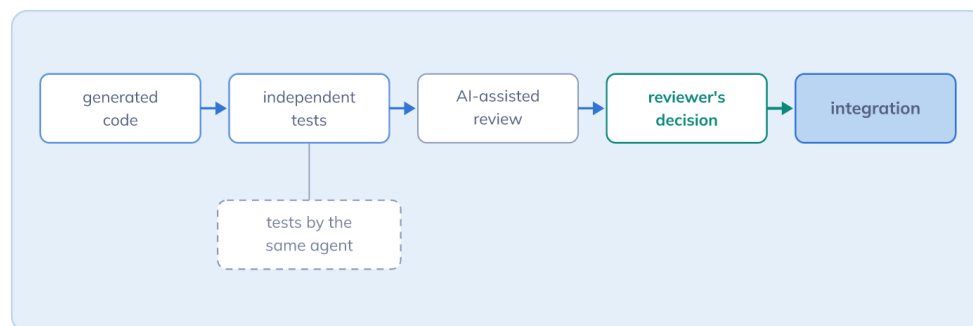


Figure 11. The path of generated code: what the AI generates goes through independently sourced tests and a review, and the decision to integrate it belongs to a person.

11.1 What the evidence says

The security of generated code

Since 2025, Veracode has measured the security of the code generated by more than a hundred models in several languages. Its first report (July 2025) found that 45% of the samples failed security tests and introduced OWASP Top 10 vulnerabilities. The *GenAI Code Security Report* of July 2026 found that the average pass rate was still 56%,

compared with 55% the previous year: models now write syntactically correct code almost always, but its security is not improving at the same pace.

Perceived and measured productivity

In 2025, METR measured the work of 16 experienced developers from open-source projects on 246 real tasks. With AI tools they took 19% longer, even though beforehand they expected to go 24% faster and afterward believed they had gone 20% faster. In February 2026, METR warned that its new experiment no longer gave a reliable signal and that the real improvement could be considerably larger, but the underlying lesson holds: the feeling of speed is not enough to know whether you are going faster.

The 2025 DORA report points in the same direction from another angle. It describes AI as an amplifier of the strengths and weaknesses an organization already has: 90% of respondents use it in their work, its adoption improves delivery throughput but is associated with more instability, and around 30% have little or no trust in generated code. The Scrum in teams with AI skill develops this productivity paradox.

11.2 Generating and reviewing tests

AI broadens the reach of testing: it generates cases aligned with the acceptance criteria and the identified risks, and it is especially useful for covering negative scenarios and edge cases beyond the happy path. The professional validates that the generated cases are the relevant ones, because the number of tests is no sign of quality if they do not target the real risks.

There is a specific trap. A passing test suite written by the same agent that wrote the code provides no independent assurance, and OWASP catalogs as a risk both that fabrication of tests and the deletion of failing ones. The tests that validate what is generated need an independent source, such as acceptance criteria agreed beforehand or tests written or reviewed by a person.

11.3 AI in code review

As a peer in code review, AI helps detect problems of readability, maintainability, and documentation and, above all, security risks: input validation, error handling, permission management, and side effects. It extends the human reviewer's coverage without replacing their judgment, and the final decision on a change remains with the person reviewing it.

Technical judgment is not delegated. Responsibility for the code and the decision to accept it remain with the professional. OWASP sums it up in one principle: every AI-assisted change must be reviewed, approved, and attributable to a developer responsible for its security. An AI finding is a signal to examine with your own judgment.

When AI generates more code than people can review calmly, review risks becoming a rubber stamp. Limiting the size of changes, distributing review work better, and relying on AI for a first pass help to avoid it.

11.4 Risks specific to development with AI

OWASP's reference sheet for programming securely with AI (*Secure Coding with AI Cheat Sheet*) catalogs the risks specific to development with assistants and agents. Three of them illustrate well why generated code calls for more rigor.

- **Hallucinated dependencies:** assistants frequently suggest packages that do not exist in public registries, a gap an attacker can exploit by publishing one under that name.
- **Fabricated or deleted tests:** an agent gets the tests to pass by writing them to fit its code or by removing the ones that fail.
- **Prompt injection in the development cycle:** files, issues, or repository documentation with hidden instructions that the agent reads and executes, the technical version of the trust boundary in chapter 4.

The Scrum Manager BoK covers these practices on its pages on AI-assisted QA and on human review of what AI produces, and the Harness Engineering skill develops how to build the automated controls that oversee the agents' work.

What you should be able to do

Having mastered this chapter, a professional is able to:

- Interpret the available evidence on the security of generated code and on the gap between perceived and measured productivity.
- Generate with AI test cases aligned with the acceptance criteria and the risks, and validate that they target the real risks.
- Detect fabricated or deleted tests and require that the tests validating generated code have an independent source.
- Use AI as support in code review with a focus on security, keeping the final decision with the professional.
- Recognize hallucinated dependencies and prompt injection in the development cycle.

Common mistakes and antipatterns

- **Confusing quantity of tests with quality.** Many tests that do not target the real risks give a false sense of security.
- **Accepting tests written by the same agent.** If they were written by whoever wrote the code, the fact that they pass proves nothing on its own.
- **Accepting generated code without rigorous review.** Almost half of the measured generation tasks introduce some vulnerability, so review must be at least as demanding as with handwritten code.
- **Treating AI findings as verdicts.** They are signals to examine with technical judgment, not automatic decisions.

- **Being carried away by the feeling of speed.** The perception of going faster is not the same as a real improvement; it is better to measure than to assume.

Further reading

- [Veracode — 2026 GenAI Code Security Report](#)
- [METR — Measuring the impact of early-2025 AI on experienced developer productivity](#)
- [METR — Uplift update \(February 2026\)](#)
- [DORA — State of AI-assisted Software Development 2025](#)
- [OWASP — Secure Coding with AI Cheat Sheet](#)
- [Scrum Manager BoK — AI-assisted QA \(in Spanish\)](#)
- [Scrum Manager BoK — Human review of AI outputs \(in Spanish\)](#)
- [Skill Arena — Scrum in teams with AI \(in Spanish\)](#)

© 2026 Scrum Manager®. This work is published under a Creative Commons Attribution-NonCommercial 4.0 International licence (CC BY-NC 4.0). Official Scrum Manager trainers and centres are licensed under the terms of CC BY 4.0 for their training activity.