

Supervision Theatre: Human-on-the-Loop as the Spear Carrier in the Agentic Play

Joana Vieira

OutSystems

joana.vieira@outsystems.com

Carlos Silva

OutSystems

carlos.silva@outsystems.com

Menno Cramer

OutSystems

menno.cramer@outsystems.com

Abstract

The vocabulary of Human-Computer Interaction was established in an era when humans executed and computers assisted. Norman's gulfs, GOMS, Cognitive Work Analysis - these frameworks share an assumption: *the human perceives and acts, the system responds, and good design closes the gap between intention and outcome*. Forty years on, AI and Agentic systems have started challenging that relationship. The human is no longer the sole executor but the supervisor, or the approver, or, more often than designers care to admit, the bystander. Yet we continue to reach for the same conceptual toolkit, dressing supervision-era problems in execution-era language. This flaw in categorization goes beyond the conceptual: it produces systems that cannot name what their human operators are actually doing, and therefore cannot design for it.

The transformation we're witnessing in software development tools is not an incremental evolution but a category shift in what it means to be the human on the loop. We argue that the most common human-agent configuration now being deployed, the developer passively monitoring a capable autonomous system, labelled Human-on-the-Loop (HOTL), is not a guarantee of safe performance. In fact, it is the most cognitively dishonest position of all. It promises supervision while structurally rendering it quite difficult. We call this pattern the *supervision theatre*. In this performance, the human acts more like the spear carrier - the non-speaking background actor whose presence implies a scene they cannot affect.

Our argument rests on Rasmussen's Decision-Ladder and his distinction between skill, rule, and knowledge-based behaviour. HOTL assumes the human can ascend on demand (perceiving, interpreting, evaluating, reformulating intention), in time to intervene meaningfully. But when the system is fast, opaque, and confidence-inducing, the human is held at the skill-based rung, reacting to salient surface signals rather than reasoning about them. Agentic coding tools optimised for frictionless flow create exactly these conditions, and then call it oversight.

Kahneman's account of System 1 and System 2 thinking compounds the problem. System 2 - the slow, deliberate, effortful reasoning that meaningful intervention requires - does not simply activate because the situation demands it. It activates when friction signals that something is wrong. Agentic interfaces optimised for fluency are, in this precise sense, System 2 suppressants: the confident output stream, the seamless commit-and-deploy removes the very friction that would trigger deliberate evaluation. The human watches the agent act, nothing feels wrong, and System 2 never engages.

Aviation and automotive automation are instructive here. Those fields demonstrate what genuine supervisory infrastructure requires: moving yokes, mode awareness displays, and recurrent simulation training. What made them a benchmark for human-automation partnership was not a commitment to HOTL throughout. It was the investment in making supervisory positions cognitively honest, and the discipline to insert Human-in-the-Loop (HITL) gates at moments of genuine consequence, such as the pre-flight checklist or the mandatory handover protocol.

Effective Human-Agent Interaction (HAI) requires knowing where the human actually is in the loop at any given moment, and designing for that position, not the one we assume or the one for which is easier to design. We propose four dimensions (accuracy required, latency tolerated, risk exposure, and cognitive feasibility) for locating the human in the HAI space, and derive corresponding interface

design commitments. Patching a vulnerability identified in a production codebase is not the same kind of task as drafting a commit message, and current interfaces should surface the appropriate differences. The framework is not a taxonomy for its own sake: it is a diagnostic instrument. Each dimension answers one of the questions that current tools and design discourse cannot answer, and we show that inability using the concrete failure patterns already endemic to today's agentic coding tools. Naming the position is the precondition for designing for it.

Software designers are choosing right now which patterns become the default for an industry still forming new Human-Agent interaction habits. These choices should be grounded in the cognitive-engineering tradition that tackled these problems in industries that faced somewhat similar transformations. We argue for the conditions it must meet: design that names the loop position the human inhabits, that treats every transition as a critical surface for design, and that measures transparency by what the human can do with what is shown, not by how much is shown.

1. Introduction

Human Computer Interaction emerged to close a gap. Norman's gulfs of Evaluation and Execution (1986), GOMS (Moran, Newell & Card, 1983), Cognitive Work Analysis (Rasmussen, 1986) are frameworks for a world in which the human acts, the system responds, and good design minimises the distance between human intention and system outcome. HCI did not stop at the reactive computer. Intelligent user interfaces and mixed-initiative interaction (Horvitz, 1999) have modelled systems that take initiative. But even there, the human held execution authority turn by turn: the system proposed, the human disposed, and the gulf being closed was still the one between a human's intention and the system's response to it. Agentic AI does not inhabit this world. When a system autonomously plans, delegates, and executes multi-step tasks, the human's role shifts from actor to supervisor. The gulf that matters now is between what the agent interprets of the human intent, what the agent executes, and what the human understands the agent to be executing. This is not an incremental challenge, but a shift on the interaction paradigm.

A vocabulary has grown up for positioning humans relative to automated systems. Human-In-The-Loop (HITL), Human-On-The-Loop (HOTL), Human-Behind-The-Loop (HBTL) (Rzeszucinski, 2024) - acronyms that treat supervisory positions as a spectrum with safe options along it. HOTL in particular has acquired the status of a reasonable middle ground: autonomous enough for efficiency, supervised enough for accountability. The argument that this framing is incomplete and actively dangerous is not new (Parasuraman, Sheridan & Wickens, 2000), but it gains new traction in the context of Human-Agent Interaction.

What we are witnessing with AI agentic systems is the appearance of meaningful human oversight without its cognitive substance, a condition we call *supervision theatre*. By supervision theatre we mean an arrangement that stages all the visible signals of oversight: a human present, plans displayed, approvals requested; while denying that human the cognitive conditions needed to actually evaluate what the agent is doing, so the oversight is *performed* rather than *exercised*. Decades of work on supervisory control (Endsley & Kiris, 1995; Parasuraman, Sheridan & Wickens, 2000) have documented when human supervision works and when it fails. This paper argues that the problem is not HOTL as a concept, but HOTL as it is currently practiced in the specific context of Agentic Coding tools. When properly designed with intervention affordances, situation awareness, and task decomposition, supervisory positioning can work - including in agentic contexts where HOTL and HITL operate on the same task. However, these conditions must be designed in, they do not emerge from the architecture of oversight alone.

The current pace of agentic deployment creates pressure to establish accountability structures before the underlying cognitive design has been done. The problem of designing for these tools is, at its root, a vocabulary failure. Our existing frameworks were built to name what happens when humans execute. They give us precise language for the gulf of evaluation, for the breakdown between intention and feedback, for the mapping between mental models and system state. What they cannot

name is the difference between a human who is genuinely supervising and a human who is merely present. Without that vocabulary, we cannot identify when supervision theatre is occurring, cannot specify what cognitive work is actually required at a given loop position, and cannot test whether an interface design delivers that work or merely performs it. The framework proposed in this paper is, in a first instance, a vocabulary for naming what is happening. The design commitments follow from that.

2. The Supervision Theatre

In a paper from 1978, Sheridan and Verplank outlined a spectrum of ten distinct modes of human-automata collaboration, running from "the human does the whole job up to the point of turning it over to the computer" to "the computer does the whole job and tells the human only if it decides it should". The term human-in-the-loop entered the HCI vocabulary through the cockpits of teleoperated vehicles and aircraft before making its way to control rooms (Sheridan, 1981), and ultimately the cubicle that holds a single human and a computer (Parasuraman, Sheridan & Wickens, 2000). A core anxiety runs through all of this initial work: when a human is nominally overseeing a system but not actively paying attention to it, or genuinely understanding it, something dangerous forms in the gap between presence and attention. Wickens (1995) named this the "*out-of-the-loop unfamiliarity*", a state produced by diminished situational awareness, complacency, and the erosion of skills.

Endsley's three-level model of situation awareness (1995) describes that passive monitoring degrades situational awareness through three compounding effects: operators become less vigilant, process information less deeply because they are not acting on it, and receive insufficient feedback to reconstruct what the automation has actually been doing. The result is an operator who can intervene in principle but cannot intervene effectively in practice.

By the 1990s, human-on-the-loop (HOTL) had emerged as the dominant architectural choice in advanced autonomous systems, conceived to reserve human attention for exception handling, with the system acting autonomously in real-time and the default loop closing without the human. Aviation adopted HOTL not naively but with a supporting infrastructure: strict certification, crew resource management training, regulated rest periods, and accident investigation bodies capable of feeding failure modes back into design. Crucially, the human supervisory role was designed around transition moments, from HOTL to human-in-the-loop (HITL) to manual control - not treated as a fixed position.

Rasmussen's work gives us two complementary tools. The decision ladder models how an operator moves from an observed signal to an executed action: detecting an alert, interpreting its meaning, evaluating it against goals, defining a task, and only then acting. Its central insight for our purposes is that a skilled operator does not climb every rung: familiar situations are handled by lateral "shortcuts" that leap from a cue straight to a stored response, bypassing deliberate evaluation entirely. Rasmussen's Skills-Rules-Knowledge taxonomy (1986) illuminates why those transitions matter. In principle, HITL operates at the Knowledge level: the human reasons about a specific case to form adequate goals and plans. HOTL operates at the Skill or Rule level: the supervisor has enough pattern recognition to catch anomalies without deliberate reasoning. Since skill- and rule-based shortcuts are fertile ground for use errors, any system that fails to account for these cognitive tendencies is prone to interaction failures (Silva et al., 2019). AI Agentic systems intensify this further: these systems frequently perform novel actions that exceed a human's ability to form clear goals at the Knowledge level, while their opacity and speed bypass the pattern-recognition on which Skill- and Rule-level monitoring depends.

The cognitive mechanism underlying failures that might happen in these moments has recently been named *cognitive surrender*. Shaw and Nave (2026) define it as the uncritical adoption of AI outputs with minimal scrutiny. The user stops deliberating and accepts the agent's answer as their own. In three experiments, participants followed faulty AI recommendations on roughly four out of five occasions, and their confidence increased regardless of whether the AI was correct or not. When outputs arrive confidently and without friction, the signal that would ordinarily route a response to

System 2 deliberation does not fire. Kahneman's account of dual-process cognition predicts exactly this: System 2 engages when something demands effort or signals error; remove those signals and it stays dormant (Kahneman, 2011).

This is the cognitive complement to supervision theatre. Where supervision theatre names the design pattern, an interface that claims oversight while preventing it; *cognitive surrender* names what happens inside the developer watching the stream of confident agent output. The Human-on-the-loop becomes the *spear carrier*. In theatre, the spear carrier is the non-speaking actor who stands in the background of a scene: their presence completes the stage picture and implies a world they take no part in shaping. The HOTL supervisor is increasingly cast in this role on stage, but unable to affect the action.

2.1. The Current State: where-in-the-loop in Agentic Coding Tools?

In the space of roughly three years, Agentic coding tools traversed the full HITL-to-HOTL spectrum, driven by productivity framing rather than any explicit theory of human supervisory control. The consequence is a set of tools where the human is technically In or On the Loop, but the conditions of the system (speed, volume, opacity, automation bias) render oversight effectively illusory. The hybrid approach of IDEs coupled with LLMs capable of generating code (such as GitHub Copilot, Cursor, Visual Studio) sits closer to the HITL end. Depending on settings, each code suggestion may require active human acceptance. Cognitive engagement is higher and more continuous than in fully agentic approaches. The failure mode here is mostly reflected in an automation bias in suggestion acceptance: the developer approves suggestions faster than they can genuinely evaluate them.

Agentic coding tools (such as Claude Code, OpenAI Codex, Google Jules, GitHub Copilot Workspace) represent the genuine HOTL shift. The developer specifies a goal: "migrate this codebase to Python 3," "add authentication with 2FA", "set up an Approval workflow", and the agent plans, executes, and iterates with minimal per-step human involvement. The human might still review logs, inspect diffs, and approve merges, but this is identical to how a pilot supervises an autopilot during cruise: authority is delegated, the human monitors, and the expectation is that intervention is reserved for exceptional events. Although software development is not a real-time task like flying an airplane (between a developer's inattention and its consequences sits a test suite, a code review and a staging environment - a pilot has none of those) the problem we identify is not concerned with the timing of the consequences, but instead with its root cause. A human watching a fast, fluent, confidence inducing yet opaque process is fertile ground for disaster. Development-time errors are committed, replicated, and executed later, in production, often by someone else, and the absence of real-time pressure is not a mitigation strategy. The flying an airplane analogy is instructive precisely because aviation did not stop there, it built the cognitive infrastructure that made that supervisory position honest (Sebok et al., 2015). Agentic coding tools, so far, have not.

Compiling a catalog of performative design traits in today's agentic coding tools is the definition of aiming at a moving target. These tools change at an unprecedented speed, but nevertheless some patterns have remained confusing or performative:

- The code snippets and reasoning that pass you by at lightning speed** - Cursor originally used it extensively, streaming code snippets to users at the agent's generation speed, though this has been moderated in newer releases. Claude.ai employs a similar approach for artifact generation, and nearly every other agentic AI tool uses it to provide a window into the agent's internal reasoning. While these movements signal activity and suggest transparency, the sheer volume and velocity of information often exceed human processing capabilities. There is a growing trend toward abandoning this practice. It remains unclear if this shift is driven by usability concerns or, as OpenAI co-founder Greg Brockman suggested (2026), a strategic decision to stop exposing raw chains of thought. According to Brockman, hiding these reasoning steps serves two purposes: (1) preventing *distillation*, as exposing internal reasoning allows competitors to more easily replicate the model's technology; and (2) preserving *faithfulness*, as the pressure to format thoughts for user

approval might lead the model to prioritize appearances over actual problem-solving accuracy.

- The enigmatic Stop button** - Present in almost all agentic coding tools, the stop button does one thing for sure, it stops the agent work right in its track. The user is left with an eerie sense of not knowing what, or if something, was written. And if it was, was it reversed? Official documentation from the biggest players clearly explaining this behavior is surprisingly difficult to find. Fortunately you can ask the agents and according to Claude Opus 4.7, the stop button seems to stop the agent execution guaranteeing that it doesn't do it in the midst of running a tool but reverting any code changes will depend on when the stop happens - "...*"code being generated" is discarded; "code already handed to the Write/Edit tool" is likely already on disk by the time you can react"*". That the most authoritative available account of the stop button's semantics comes from interrogating the agent itself, rather than from any documentation the vendor publishes, is, of course, part of the point. Cursor and Codex seem to apply a similar strategy and a similar resistance to put it out on official documentation.
- The unscrutable permission requests** - The default configuration for IDEs coupled with AI assistants typically restrict consequential autonomous agent actions, requiring manual authorization for most write operations. But ultimately this constant stream of permission request prompts, coupled with a high success rate, tends to condition users to bypass scrutiny, a behaviour we have described as *cognitive surrender* (Shaw & Nave, 2026). Developers frequently resort to options like whitelisting in Cursor, selecting "always allow" or "bypass permissions" in Claude Code, or enabling permissions to apply to all projects in VS Code. This shift begins with transparent requests but eventually extends to more obscure ones that even technically proficient users may find difficult to decipher.
- The unpredictable context filling** - Context filling is frequently represented by a circular progress indicator or token counter, but this representation systematically misleads. What fills the context window is opaque to the developer: is it the conversation history? Tool calls? File contents the agent retrieved? The counter tells you how full the context is without telling you what is in it. When the context approaches full capacity, agent behaviour degrades in ways that are difficult to predict and harder to diagnose. The agent might silently drop earlier instructions, lose track of architectural decisions made at task outset, or repeat work it has already done. The degradation is rarely announced. Some tools offer a "compact" or "summarise" action to reclaim context space, but the summary is itself produced by the model, substituting an original record for the lossy compression, while the developer does not know what was lost. This is a transparency problem at the centre of the supervisory relationship. A human supervisor who cannot know what their agent remembers cannot meaningfully evaluate what their agent is about to do.
- Long plans, big small changes** - Agentic tools have developed a pattern of presenting elaborate, multi-step plans before execution with numbered lists of intended actions that signal thoroughness and invite implicit approval. The plan creates an expectation, but the execution can break it, not through spectacular failure but through an accumulation of small, locally reasonable deviations: a dependency quietly upgraded, an abstraction layer introduced that was not specified, a configuration file modified because it was in the way of the task's main goal. Each individual decision might be defensible but the aggregate is a codebase that has drifted from what the developer understood themselves to be building. This is precisely the *out-of-the-loop* degradation described before: the human approved a plan, not a process, and the agent executed a complex, hard to follow process. The design problem is that plan presentation has become its own form of supervision theatre. A ritual of the numbered lists substituting for the harder work of making moment-to-moment agent decisions genuinely legible.

This is not an exhaustive list and it is one focused on the core of the development tasks. One could easily add to it the difficulty to predict token usage (and pricing) and other struggles that are more related to the customer journey. These are patterns we kept running into as practitioners building and shipping software on a development platform. Also important to say is that these features might behave differently depending on the tool but, at the time of the writing of this article, the described problematic behaviors were endemic to most of the more popular tools.

2.2. The Moral Crumple Zone: Coming Soon to a Supervision Theatre Near You

The concept Elish (2019) introduced as the "*moral crumple zone*" describes what happens when the conditions of illusory supervision play at scale. The term names the scenario in which a human operator, positioned in the loop by design, absorbs blame for a failure they could not realistically have prevented given the oversight conditions they were actually working under. The crumple zone is a predictable consequence of designing accountability into a system without designing the cognitive infrastructure that would make accountability meaningful.

In its original context, Elish drew on cases from robotics and autonomous vehicles, namely the human safety driver in the Uber fatality of 2018, supervising a system operating faster and with more sensor data than any human could process in real time. This pattern is now migrating into software development tools and the stakes, while lower in consequence, are probably higher in volume. Grudin (2009) documented how AI and HCI spent decades talking past each other, developing separate vocabularies for what is ultimately the same problem: *how does a human work effectively with a system that knows more than they do, faster than they can process, across a task space they cannot fully observe?* The cognitive ergonomics tradition answered this question carefully for aviation, nuclear operations, and process control. The agentic coding tools are answering it again, from scratch, and seemingly without much reference to the previous answers. They are arriving, predictably, at the same failure modes.

To avoid going through the same errors, we propose that the first step should be towards effectively defining the Human-Automation partnership in the context of the new AI Agentic Systems.

3. Defining HAI against its neighbours

Human-Computer Interaction assumes the human executes and the computer assists. The Human-AI Interaction literature treats the AI as a single interactive surface to be made legible and correctable (Amershi et al., 2019; Shneiderman, 2020). Neither addresses what AI agentic systems put on the table: a job-to-be-done can be performed by the human, by the agent, or by both, and the question of whether a human should be in the loop has become detached from whether their presence does any work.

We define Human-Agent Interaction (HAI) as a design stance concerned with deciding, for each job-to-be-done, whether a human belongs in the loop at all; if so, where, why, and under what conditions their presence is cognitively honest rather than performative. Therefore, HAI's contribution is not a taxonomy of loop positions, it is a principle of refusing to claim oversight that the design cannot deliver.

The frameworks we draw on each answer a real question: Sheridan and Verplank (1978) and Parasuraman, Sheridan and Wickens (2000) tell us how much authority has been delegated; Endsley (1995) tells us what the human must perceive to act; Bainbridge (1983) documented what happens when the residual role is not performable. What we found, working with agentic tools daily, is that these models quietly assume the human's loop position is stable, assigned by design and held, with any transitions few, scripted, and trained for, as aviation's handover protocols illustrate. In agentic development, neither assumption holds. The position shifts constantly and without script, within a single task and sometimes within a single minute: plan approval is one position, watching execution is another, being asked mid-stream to permit a file write is a third, and no protocol governs any of these transitions. Once the position is that fluid, feasibility can no longer be a background finding, it has to

be assessed continuously, per job-to-be-done, which is why we make cognitive feasibility (C) an explicit dimension rather than an inherited assumption.

3.1. The fallacy at the centre of the market

"Human in the loop" has become the industry's reassurance phrase. It is invoked to license agent deployment in high-stakes domains, to satisfy regulators, and to absorb accountability, often without any examination of what the human in the loop is actually doing. When a reviewer faces 10,000 loan applications and a confident agent recommendation on each, the human is not in the loop. They are next to the loop, pressing approve. The system inherits a signature of human oversight it did not earn, and the organisation inherits a defence, "a human reviewed it", it cannot honestly mount. The EU AI Act's Article 14 requires that high-risk systems be designed so they "*can be effectively overseen by natural persons*", and explicitly warns deployers about automation bias (European Parliament & Council, 2024). The aspiration is correct. The implementation, in most deployments, is not, and this is worse than no human at all. A system without a human in the loop is at least honest about its autonomy; its failures attach to its designers. Green and Kak (2021) call this "*the false comfort of human oversight*": a legal and rhetorical device that protects institutions while doing little for the people the oversight was meant to protect. Crootof, Kaminski and Price (2023) make the related legal argument in their identification of the MABA-MABA trap, *Men Are Better At / Machines Are Better At*, originally named by Dekker and Woods (2002), in which policymakers respond to algorithmic incapacities by inserting a human into the decision without ensuring they have the authority, information, or time to exercise meaningful judgement.

The cognitive evidence is older and equally unambiguous. Bainbridge (1983) named the dynamic forty years ago: the ironies of automation. The more you automate, the more the residual human role degrades into monitoring, and the worse humans get at the monitoring. For instance, the switch to semi-automated vehicles intended human drivers to spend less time in basic tasks such as steering, and spend more attention monitoring the traffic environment. Nevertheless, automation increased the likelihood these drivers performed non-driving related tasks, thus reducing situation awareness and response times to alerts (Jansen et al., 2019).

The five patterns documented in Section 2.1 are the evidence of this claim. The streaming code/reasoning pattern cannot answer whether the cognitive work of verification is feasible under the conditions the interface creates. The permission request pattern cannot answer what specific cognitive work the human is supposed to perform when approving. The plan presentation pattern cannot answer whether the human's approval of the plan constitutes meaningful oversight of the execution. The context filling pattern cannot answer what the agent currently knows, which means neither can the human. The stop button cannot answer what loop position the human transitions into when they use it. These are not implementation failures, but rather patterns derived from a failure to ask the right questions before the design was built. The framework below provides those questions.

3.2. The decision space: accuracy, latency, risk, and cognitive feasibility

Three dimensions characterise most jobs-to-be-done. **Accuracy required (A)** which is the tolerance for incorrect output; **Latency tolerated (L)** which is the time available for cognition between action and consequence; and **Risk exposure (R)** which is the blast radius of an incorrect action. These three set the demand for human involvement. A fourth dimension sets the supply: **Cognitive feasibility (C)**: whether a human, under the conditions the interface actually creates, can perform the work the loop position would require. C is what most "human in the loop" claims silently assume and routinely violate. Reviewing 10,000 applications is not cognitively feasible. Reviewing a streaming code diff faster than it is generated is not either. Catching a subtle logic error in a confident agent output, when nine out of ten previous outputs were correct, is not cognitively feasible, and Parasuraman and Manzey's (2010) attentional model predicts exactly this failure. Green and Chen (2019) showed it empirically: reviewers presented with algorithmic recommendations underperformed the algorithm and introduced new disparities rather than catching old ones.

This decision space is helpful to decompose tasks and understanding the H*TL implication of each job-to-be-done. Let's walk through the example of "Unifying auth login flows", as it is an extremely complex task which can be decomposed into at least five jobs-to-be-done, each with a different A/L/R/C profile, resulting in a different honest loop position.

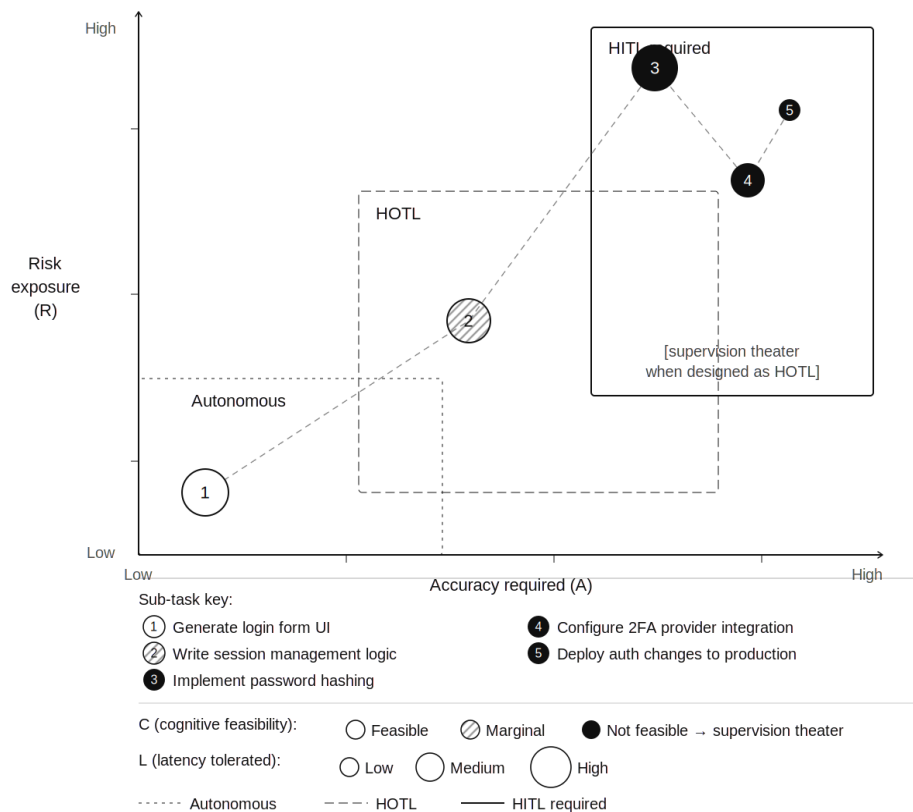


Figure 1 - The HAI decision space applied to the decomposition of a single task: "Unify auth login flows." Each numbered point represents one job-to-be-done, plotted by accuracy required (A), risk exposure (R), and latency tolerated (L, bubble size). Bubble fill encodes cognitive feasibility (C)

The figure makes visible something that neither the standard HITL/HOTL vocabulary nor the five-pattern analysis above can show directly: that jobs-to-be-done 4 and 5 in the Unifying auth login flows example (configure 2FA provider integration; deploy auth changes to production) look like reasonable HOTL candidates on the first three dimensions alone. Add cognitive feasibility, and they become visible as supervision theatre. The framework earns its place by making that category distinction possible. Without C as an explicit dimension, a designer looking at those tasks would see high-stakes work requiring human review and would design a review step. With C, they see that the review step, under the conditions current tools create, is not genuine oversight and they are forced to either redesign the conditions or own the autonomy.

3.3. The decision model

The framework to define a loop position asks four questions, in order:

1. **Does the job require human judgement at all?** Many do not. Low-A, low-R jobs are honest candidates for full agent autonomy, and inserting a human into them is a tax on the human. If every action requires approval, no action is meaningfully approved.
2. **If a human is required, what specific cognitive work must they perform?** Not "review", that is a verb that hides the work. "Detect" what kind of error? "Reconcile" what against

what? “Decide” between which alternatives? If the work cannot be named, the loop position cannot be designed.

3. **Is that work cognitively feasible under the conditions the interface creates?** Volume, time pressure, fluency of agent output, base rate of agent error, and the salience of the cues the human would need, these determine feasibility (Parasuraman & Manzey, 2010). If the answer is no, the loop position is theatre, and the honest response is to redesign the job (reduce volume, increase latency, add second-order checks) or to remove the human and own the autonomy.
4. **What would make the human's presence irreplaceable?** Forcing functions on attention, calibrated friction at moments of consequence, structured disagreement protocols, sampling regimes that make the work tractable, dashboards that surface the absent signal rather than the present one.

The output of the model is a defensible answer to the question: *should a human be in this loop, and if so, what does the design owe them to make their presence real?*

4. Design Implications: Four Questions, Four Commitments.

The decision model proposed above asks four questions, and each question produces a design commitment. Each commitment is also a direct response to a failure pattern documented in Section 2.1. The framework is not proposing new requirements, instead it is naming what the existing failures demonstrate was missing.

1. **Name the loop position.** A commit message and an authentication refactor are not the same kind of job, and the interface should say so before the agent acts. The practical move is a task-level declaration: this action runs autonomously; this one requires plan approval; this one will pause before any irreversible step. Magentic-UI (Mozannar et al., 2025) does this through its co-planning interface, which distinguishes tasks that warrant a direct response from those requiring an editable plan and explicit acceptance. The developer knows, before execution begins, whether they are expected to evaluate or simply to observe, and that expectation shapes how they pay attention.
Claude Code's recently introduced `/goal` command works from the opposite end of the spectrum but honours the same commitment: the developer defines a completion condition and the agent works autonomously across multiple turns until it is met (Anthropic, 2026, release notes). What makes this honest is not the autonomy but the declaration, the interface states up front that no one will be watching turn by turn, and substitutes an explicit, checkable goal for the pretence of continuous supervision. Naming the position as unsupervised is still naming the position.
2. **Specify the cognitive work.** Telling a developer to "review" an agent's output names a role without specifying the work. Review of what, against what standard, to reach what decision? In agentic contexts it's important to show not just what changed but what the agent considered and rejected. A diff shows output. A decision diff shows reasoning: *"I needed to handle the case where a user is not logged in. I could redirect to the login page or return an error. I chose the redirect because every other route in this codebase does the same."* That gives the developer a decision to evaluate rather than a result to accept or refuse. Magentic-UI's answer verification mechanism moved in this direction, providing a structured action history at task completion. The authors learned that users found long action stories hard to monitor and asked for visual aids to quickly assess the agent's work.

3. **Test for feasibility.** The relevant question about any transparency feature is not whether the information is shown but whether a developer, under realistic conditions, can act differently because of it. AI Agentic systems should tell the developer not just that the agent “... *is AI and can make mistakes*” (as per Claude’s disclaimer), but how reliably it handles this class of task, so they know how closely to attend. The design test is simple: take any proposed transparency feature and ask what decision it enables. A stream of code passing at generation speed fails that test: the information is present, but the conditions make it unprocessable. A task-level summary such as “I changed how the application handles failed logins across four files; here is the before and after behaviour” passes it, because a developer can read in a fairly short amount of time and decide whether to look closer. During the validation of automation systems in vehicles time-and-motion tests are highly advisable, even if done in a simulated context, as these look to replicate the real world time and motion constraints that the human occupant faces when deciding how to act upon the system (ISO 34502:2022).
4. **Build forcing functions.** Forcing functions are a class of physical or logical constraints that make it impossible to perform an action incorrectly or out of sequence. They force the right behaviour by making the wrong behaviour unavailable (Norman, 1988). In AI agentic systems, the more pressing problem is conveying the consequences of agent actions before they become irreversible. The design commitment is to gate on risk, not on agent uncertainty. Magentic-UI’s ActionGuard architecture operationalises this precisely: every agent action carries a pre-defined irreversibility heuristic (*always*, *maybe*, or *never* irreversible). The gate is triggered by the consequence category, not by agent uncertainty or agent confidence. Pushing to production, modifying authentication logic, or deleting data should require human approval not because the agent is hesitant but because the cost of an error is high. Permission scoping (read-only by default) is itself a forcing function that prevents most users from making consequential changes.

5. Conclusion: The Importance of Being Earnest

The cognitive engineering tradition answered the question of when human supervision works and what it requires. The agentic AI industry is answering the same question from scratch, without much reference to those answers, and arriving at known failure modes. We began this paper by proposing to explore what design for human-agent supervision should look like to be effective and cognitively honest. Our exploration concludes that current agentic coding tools, for most tasks, position the human as a signature, a presence in the loop that produces no different outcome from their absence. This stems from a lack of adequate language for problems of supervision in this agentic era. The framework to diagnose the correct position of the human in the loop, complemented with corresponding design commitments, serve as a first instance of that needed vocabulary. This does not fully solve the design problem, but makes the challenge visible. The design decisions being made now will be the defaults of the next decade. They should be grounded in what we actually know about human cognition under supervisory conditions. If the human cannot meaningfully supervise, own the autonomy. Some jobs-to-be-done should not have a human in the loop at all. If the cognitive feasibility analysis shows that genuine supervision is not achievable under the conditions the interface creates, the honest move is to remove the human, redesign the conditions, or not deploy. The vocabulary this paper offers is a precondition for making that choice more deliberately, rather than by default. On the other hand, if a human can supervise, let’s design for it: name the loop position, specify the cognitive work, test for feasibility, and gate on risk. Accountability, for now, still rests where the law places it: with the human in the loop (European Parliament & Council, 2024). But the cognitive ground beneath that placement is shifting faster than the legal frame around it. The landscape is not yet settled, and we do not claim to settle it. We do insist that where accountability is assigned, the cognitive conditions that make it meaningful must be designed in, not assumed.

One assumption runs beneath the whole argument: that there is a human on the loop at all. The mid-term trajectory may well be agents reviewing agents, with the developer moving from on the loop to nowhere near it. We expect these commitments to hold even then, because underneath they are commitments about legibility and communication at the point of consequence, not about human attention specifically - but that is an expectation we have not yet tested, and we flag it as the boundary of what this paper claims.

In Wilde's *The Importance of Being Earnest*, the comedy depends on the gap between what characters perform and what they actually are. That gap is sustainable as farce. In the design of systems that carry real consequences, it is not.

Acknowledgements

We thank the PPIG reviewers, whose observations, in particular on the societal persistence of the distinction between writing and running software, sharpened the conclusion of this paper. We are equally grateful to the participants of the 37th PPIG workshop in London, whose questions, comments, and generous discussion improved this work, and whose warm welcome made presenting it a pleasure.

6. References

- Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for Human–AI Interaction. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–13. <https://doi.org/10.1145/3290605.3300233>
- Anthropic. (2026, May). *Claude Code release notes, v2.1.139* [Software changelog]. GitHub. <https://github.com/anthropics/claude-code/blob/main/CHANGELOG.md>
- Bainbridge, L. (1983). Ironies of automation. *Automatica*, 19(6), 775–779. [https://doi.org/10.1016/0005-1098\(83\)90046-8](https://doi.org/10.1016/0005-1098(83)90046-8)
- Brockman, G. (Guest), & Parrish, S. (Host). (2026, April 22). *Greg Brockman: AI Is About to Go Parabolic! Here's What's Next*. In *The Knowledge Project Podcast*. YouTube. <https://www.youtube.com/watch?v=6JoUcQ1qmAc>
- Crootof, R., Kaminski, M. E., & Price, W. N., II. (2023). Humans in the loop. *Vanderbilt Law Review*, 76(2), 429–510.
- Dekker, S., & Woods, D. D. (2002). MABA-MABA or Abracadabra? Progress on human–automation co-ordination. *Cognition, Technology and Work*, 4, 240–244. <https://doi.org/10.1007/s101110200022>
- Elish, M. C. (2019). Moral crumple zones: Cautionary tales in human-robot interaction. *Engaging Science, Technology, and Society*. <https://doi.org/10.17351/ests2019.260>
- Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1), 32–64. <https://doi.org/10.1518/001872095779049543>
- Endsley, M. R., & Kiris, E. O. (1995). The Out-of-the-Loop Performance Problem and Level of Control in Automation. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(2), 381–394. <https://doi.org/10.1518/001872095779064555>

- European Parliament & Council. (2024). *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. Official Journal of the European Union, L series, 12 July 2024.
- Green, B., & Chen, Y. (2019). Disparate interactions: An algorithm-in-the-loop analysis of fairness in risk assessments. *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19)*, 90–99.
- Green, B., & Kak, A. (2021, June 15). The false comfort of human oversight as an antidote to A.I. harm. *Slate*.
<https://slate.com/technology/2021/06/human-oversight-artificial-intelligence-laws.html>
- Grudin, J. (2009). AI and HCI: Two fields divided by a common focus. *AI magazine*, 30(4), 48-57.
<https://doi.org/10.1609/aimag.v30i4.2271>
- Horvitz, E. (1999). Principles of mixed-initiative user interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99)*, 159–166.
<https://doi.org/10.1145/302979.303030>
- International Organization for Standardization. (2022). *Road vehicles - Test scenarios for automated driving systems - Scenario based safety evaluation framework* (ISO Standard No. 34502:2022). <https://www.iso.org/standard/78951.html>
- Jansen, R. J., van der Kint, S. T., & Hermens, F. (2019). Does agreement mean accuracy? Evaluating glance annotation in naturalistic driving data. *Behavior Research Methods*, 51(6), 2520–2535
<https://doi.org/10.3758/s13428-020-01446-9>
- Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus and Giroux.
- Moran, T. P., Newell, A., & Card, S. (1983). GOMS (Goals, Operators, Methods, and Selection rules). *The Psychology of Human Computer Interaction*. Lawrence Erlbaum Associates.
- Mozannar, H., Bansal, G., Tan, C., Fournery, A., Dibia, V. C., Chen, J., Gerrits, J., Payne, T., Maldaner, M. K., Grunde-McLaughlin, M., Zhu, E., Bassman, G., Alber, J., Chang, P., Loynd, R., Niedtner, F., Kamar, E., Murad, M., Hosn, R., & Amershi, S. (2025). Magentic-UI: Towards Human-in-the-loop Agentic Systems. *arXiv preprint arXiv:2507.22358*.
<https://doi.org/10.48550/arXiv.2507.22358>
- Norman, D. A. (1986). Cognitive engineering. In *User centered system design* (pp. 31-62). CRC Press.
- Norman, D. A. (1988). *The psychology of everyday things*. Basic books.
- Parasuraman, R., & Manzey, D. H. (2010). Complacency and bias in human use of automation: An attentional integration. *Human Factors*, 52(3), 381–410.
<https://doi.org/10.1177/0018720810376055>
- Parasuraman, R., Sheridan, T. B., & Wickens, C. D. (2000). A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3), 286-297. <https://doi.org/10.1109/3468.844354>
- Rasmussen, J. (1986). A framework for cognitive task analysis in systems design. In *Intelligent decision support in process environments* (pp. 175-196). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Rzeszucinski, P. (2024, February 29). *AI, humans and loops*. Medium.
https://medium.com/@pawel.rzeszucinski_55101/ai-humans-and-loops-04ee67ac820b
- Sebok, A., Wickens, C., Laux, L., & Jones, M. (2015). Supporting human-automation interaction in the rail industry by applying lessons from aviation. In *Proceedings of the Human Factors and*

- Ergonomics Society Annual Meeting* (Vol. 59, No. 1, pp. 1661-1665). Sage CA: Los Angeles, CA: SAGE Publications. <https://doi.org/10.1177/1541931215591359>
- Shaw, S. D., & Nave, G. (2026). *Thinking, fast, slow, and artificial: How AI is reshaping human reasoning and the rise of cognitive surrender*. OSF Preprints. https://doi.org/10.31234/osf.io/yk25n_v1
- Sheridan, T. B. (1981). Understanding human error and aiding human diagnostic behaviour in nuclear power plants. In *Human detection and diagnosis of system failures* (pp. 19-35). Boston, MA: Springer US.
- Sheridan, T., & Verplank, W. L. (1978). *Human and computer control of undersea teleoperators*. A Man-Machine Systems Laboratory Technical Report.
- Shneiderman, B. (2020). Human-centered artificial intelligence: Reliable, safe & trustworthy. *International Journal of Human-Computer Interaction*, 36(6), 495–504. <https://doi.org/10.1080/10447318.2020.1741118>
- Silva, C., Masci, P., Zhang, Y., Jones, P., & Campos, J. C. (2019). A use error taxonomy for improving human-machine interface design in medical devices. *ACM SIGBED Review*, 16(2), 24-30. <https://doi.org/10.1145/3357495.3357498>
- Wickens, C. D. (1995) Designing for Situation Awareness and Trust in Automation. *IFAC Proceedings Volumes*, Volume 28, Issue 23. [https://doi.org/10.1016/S1474-6670\(17\)46646-8](https://doi.org/10.1016/S1474-6670(17)46646-8)