A role is not a capability

Why an agentic architecture must separately justify the capability it mobilises, the autonomy it grants and the authority it withholds

Introduction: what one decides when naming a persona

In the emulation of a trial comparing CAR-T therapies with standard of care in aggressive lymphomas, the choice of adjustment variables rests first on a causal representation of the clinical process. Target trial emulation consists in making explicit the randomised trial one would have wished to conduct, then imitating it with observational data (Hernán and Robins, 2016). Building this representation is a matter of scientific judgement: assumed relationships between variables, unmeasured confounding, selection mechanisms, the definition of time zero, without which the delay between eligibility and actual administration of treatment is enough to produce immortal time bias (Hernán et al., 2016). Once the causal graph has been made explicit and its assumptions examined, part of the work changes in nature. Algorithms identify the adjustment sets that satisfy the graphical criteria and check the specified temporal constraints, reproducibly and independently of how the question is phrased. They validate neither the correctness of the graph, nor the identifiability of the effect in the data, nor the statistical relevance of the estimator, and they do not choose between several admissible sets whose variance properties differ.

A widespread practice would consist in entrusting the whole to an "epidemiologist" agent, endowed with a credible persona and a right to speak in a deliberation between agents. It would conflate three things that the example separates: the judgement that constructs the problem, the computation that handles its mechanically checkable part, and the validation that bears on both. In this practice, which we shall call persona-based agentification, naming a function amounts to assigning it an LLM instance, a role expressed by prompt and a conversational interface, before it has even been established whether this function requires generative processing.

It is this practice that the article targets. It contests neither the principle of the agent, which may rest on a solver, a rules engine or a combination of components, nor that of a multi-agent architecture. Several specialised agents may contribute distinct capabilities, rights or contexts that justify their separation. This justification does not exempt one from measuring their coordination costs and controlling their effects. A second error often accompanies the first without being equivalent to it: leaving the component that proposes an action with excessive authority over its execution. An architecture with a single tool-equipped LLM may exhibit exactly the same control defects as an architecture

with ten personas; an architecture divided by roles may have robust control. The first error motivates the article; the second explains why its correction, the specialisation of computation, must be accompanied by a separation of authority that it does not guarantee.

The thesis comes down to three propositions. A role is not a capability: each function must be entrusted to the component whose evaluated properties satisfy its requirements. A capability does not necessarily justify an agent: the autonomy granted to a component must be justified by the value it brings. An agent does not automatically hold authority over its effects: external effects must be subject to authorisation, allocation and evidence mechanisms that are independent of the component proposing them, according to their criticality and the guarantees required.

These three propositions answer a single question: to which component should a function be entrusted, at what point in its life cycle, and under what authority? The article defends a design and justification criterion. It does not claim that hybrid architectures are systematically less costly, that deterministic components are always more reliable, or that multi-agent systems are intrinsically less performant. It concerns agentic systems endowed with capabilities to act on external resources, data or systems, in environments where evidence has a cost and a value, foremost among them healthcare, pharmaceuticals and defence.

A few terms suffice. An *agent* is a component endowed with an objective, access to tools and a decision loop over its own actions; its *autonomy* is the extent of the decisions this loop takes without external validation. A *role* is a behavioural specification assigned by prompt. A *capability* is an effective aptitude to perform a function, whose properties and limits are evaluated over a defined domain of use; a role may contribute to this aptitude, but it constitutes neither its proof nor its guarantee. An *authority boundary* separates responsibilities, rights or competences; a *unit of computation* divides a processing task according to the nature of the operation. A constraint is *mechanically checkable* when a specified mechanism can, from the accessible information, determine whether it is satisfied, violated or not assessable; only satisfaction allows a positive conclusion, and only if the required level of evidence is reached. A constraint is *open* when one does not know, today, how to specify such a mechanism, because its assessment requires judgement. An *external effect* is any action that consumes a resource, discloses data or modifies a state outside the reasoning context.

1. Why one divides by personas

Persona-based agentification is practised by competent teams, and it must be accounted for before it is criticised. What follows describes plausible mechanisms; no data at this stage allows their frequency to be established or their relative weight to be ranked.

The first mechanism is organisational. Conway observed that systems tend to reproduce the communication structure of the organisations that design them (Conway, 1968). A "finance" agent, a "compliance" agent and a "risk" agent reproduce an organisation chart, and with it legible lines of responsibility and budget. The second is cognitive: a role is an interface for understanding, which makes it possible to explain the system, locate a defect and distribute maintenance. This benefit bears on the representation of the system, not on its properties; a legible organisation chart of agents may mask an illegible computation. The third is tool-driven: some frameworks define their agents by a role, an objective and a personal history, CrewAI's role, goal, backstory triptych being the most explicit example. When the tool asks for a persona before an interface contract, the persona becomes a default unit of design. The fourth is protocol-driven: the A2A protocol, launched by Google in April 2025 and then entrusted to the Linux Foundation, aims to enable agents to discover each other's capabilities, exchange information and coordinate tasks. When interoperability is negotiated agent to agent, wrapping a function in an agent becomes a way of making it addressable.

One must concede what the role does well. In a regulated environment, separation between production and validation is a requirement; the four-eyes principle exists because the one who produces must not be the one who approves. A role that materialises such a boundary has real architectural value. The problem arises when the role is used to divide computation without corresponding to any boundary of authority, rights or context. The persona then becomes a governance artefact disguised as a computational artefact: it reassures as to who answers for what, without guaranteeing anything about what is computed.

2. A role is not a capability

The first cost of the confusion is one of interpretation. Anthropic reports that a system composed of a Claude Opus 4 orchestrator and Claude Sonnet 4 sub-agents outperforms the single Opus 4 agent by 90.2% on an internal research evaluation, and that, on the BrowseComp benchmark, three factors explain 95% of performance variance, token consumption alone explaining 80%; these systems consume roughly fifteen times more tokens than a chat interaction (Anthropic, 2025). These results show that a multi-agent configuration can improve performance on certain research tasks by mobilising more computation.

It does not establish a causal relationship: a variance decomposition remains an association, and the evaluation is internal. The cautious conclusion suffices: gains attributed to multi-agent systems must be interpreted in light of the additional resources mobilised, and cannot be automatically attributed to decomposition into roles.

The second cost is an accounting one, and it cannot be read off a token counter. Each agent reconstructs part of the context that others already possess; each handoff

produces an intermediate output that must be generated and then reread; each sequential step adds its latency; each cross-check consumes as much as a generation; the observability and debugging of a multi-agent deliberation carry a cost that the inference bill does not show. Conversely, a specialised component has design, integration and maintenance costs, and a general-purpose LLM may be the cheapest solution for a rare, ambiguous or changing function. The relevant measure is the full cost per task correctly completed, rework, control and validation included. It is not sufficient on its own: two architectures may not have the same functional coverage, the same abstention rate or the same exposure to prohibited effects, and the one that correctly refuses a task will appear more expensive than the one that wrongly executes it. The comparison must keep separate the quality of admissible tasks, coverage and abstention rate, constraint compliance, latency and full cost. The word *waste* is justified only in one demonstrated case: when an architecture mobilises additional resources without contributing a property that justifies their cost.

The third cost lies in the exchange format. When components exchange their results primarily in the form of prose, they impose on their recipients a new interpretive task: a computed amount becomes a sentence that must be reread to extract an amount from it, an uncertainty becomes an adverb. Typed interface contracts reduce this ambiguity and make certain structural violations detectable, without guaranteeing the truth, completeness or relevance of what is transmitted. The point is not to ban natural language, but not to use it as an implicit substitute for an interface contract.

The fourth cost is the most structural. Cemri et al. analyse seven multi-agent frameworks and establish a taxonomy of fourteen failure modes distributed across three categories: specification and design, inter-agent misalignment, verification (Cemri et al., 2025). They observe that the gains of these systems often remain minimal compared with single agents on common benchmarks. This result does not provide a rate transferable to a regulated architecture; it indicates where to look: verification. A second instance of the same model may detect certain errors, notably by following a different line of reasoning; it does not, by its mere existence, constitute an independent verification. Two instances that share a foundation model share its biases and vulnerabilities; their agreement does not prove their correctness. Dependability engineering knows this problem as common-cause failure: homogeneous redundancy offers poor protection against what affects all its branches equally.

We spent a decade learning not to split a monolith into a hundred chatty services. We are about to do it again with services that bill their chatter by the token.

3. Allocating by properties, and at the right time

Each function is characterised by a requirements profile, then entrusted to a class of component whose evaluated properties cover that profile. The role defines who answers for a decision; the capability defines what can be guaranteed about it.

Allocation proceeds in three steps. The first characterises the function: inputs, outputs, variability, degree of specification of the rules, availability and reliability of data, required level of evidence, risks, external effects. The second identifies the admissible solutions, that is, all those that satisfy the eliminatory requirements, whatever their nature: deterministic function, solver, rules engine, specialised model, LLM, hybrid architecture, artefact generated then validated. The third arbitrates between them. Full cost is a central criterion there, but some properties cannot be reduced to thresholds: evolvability, reversibility, vendor dependency, ease of post-incident investigation. A composite score would mask these choices behind arbitrary weightings; it is better to document the trade-off, its volume and change assumptions, the compromises accepted, its owner and the conditions for its review. An undocumented trade-off is not an allocation; it is a habit.

Two decisions must remain distinct: functional qualification, which asks whether generativity contributes a necessary or useful capability, and arbitration, which asks which of the admissible solutions is best suited. The structure of the data does not settle the first. Planning with typed inputs and outputs may require generative reasoning when the work consists in interpreting a request or proposing a formalisation. A solver is suitable when the problem has been posed; an LLM may be suitable when it has yet to be posed. And an LLM may be selected for a function that does not require it, if the trade-off favours it: allocation by properties is not an anti-generative dogma, it is a discipline of justification.

The decisions that matter in a regulated environment often bear on partially checkable constraints. A contextual drug contraindication contains a mechanically checkable part (the documented interaction between two substances, the renal function threshold beyond which a dosage must be adjusted) and an open part (the benefit–risk trade-off for a given patient). The threshold is perfectly specified; but if the available laboratory value is outdated or unrepresentative, the check must conclude "not assessable" and refer the case back. A check that can only conclude true or false would turn the absence of data into implicit authorisation. Entrusting the entire constraint to an LLM would leave the checkable part dependent on a probabilistic component; entrusting it entirely to a rules engine would amount to claiming to have specified the judgement. The two errors are symmetrical, and the second is the more frequent among those discovering the virtues of determinism.

The burden of proof increases with the criticality of the function. But the unit of proof is not the class of component: it is the property claimed. A rules engine can apply an

erroneous rule exactly; a solver can guarantee the optimality of a badly formulated problem. Conversely, certain properties of a probabilistic component can be guaranteed by its execution envelope: permissions, resource bounds, accepted formats, absence of direct access to a protected resource. One does not prove a component; one proves a property of a component, under assumptions.

The life cycle adds a dimension that instantaneous allocation ignores. A capability is designed, validated and executed, and each stage may call on a different component. The LLM can generate a rule, a query, a workflow or a program; this artefact is validated, then executed without an LLM. Designing with an LLM does not require executing with an LLM. The benefit of this configuration must be stated precisely, because a frozen artefact provides three distinct benefits, none of which implies the others: the stability of a versioned object, the possibility of inspecting it, and the feasibility of establishing its required properties. A short declarative rule and a generated program of several thousand lines are both frozen; their validation surfaces have nothing in common. The favourable case is that of an artefact whose domain of use, dependencies and expected properties can be bounded and qualified at an acceptable cost. And when the artefact fails, that failure must not automatically authorise a substitute generative execution: in a critical function, this fallback is itself an allocation decision, to be qualified and controlled, without which the fallback path becomes the route by which unvalidated generativity returns to production.

The characterisation of a function is not fixed. An expert practice becomes codified, data becomes available, a model improves; the price of inference falls, evidentiary requirements rise. A doctrine that fixed a boundary would be wrong within two years. This one fixes a criterion, which applies to a moving boundary.

4. A capability does not justify an agent

The division of a system into distinct components may be motivated by six desired properties: parallelism, when the task decomposes into independent branches and latency matters; context isolation, when a single context would be saturated or polluted; separation of rights, which limits the extent of the damage a hijacking can cause; verification diversity, when the verifiers' errors are not correlated; authority boundaries, regulatory or inter-organisational; the independent exploration of different strategies within the same search space. These are possible motives, not sufficient justifications, and the list is not exhaustive. A desired property justifies a separation only if the separation actually delivers it, at an acceptable cost: parallelism may increase contention, isolation may degrade information sharing, the multiplication of boundaries complicates coordination. Strategy exploration, in particular, is plausible but remains to be demonstrated at comparable resources.

Above all, these motives justify a separation, not yet an agent. A parallel computation branch may be a function; a rights boundary may be materialised by a conventional service; an independent verification may be a deterministic test. Two successive decisions must be justified: why separate this function, then why the separated component must have decision-making autonomy. The second is justified when the component's capacity to choose and revise its actions brings, relative to the admissible solutions without this autonomy, a property or a gain that covers its costs and risks. A separation is justified by the property it brings; autonomy, by the value it demonstrates. An agent that separates nothing and whose autonomy contributes nothing is merely an additional prompt with an identity.

Orchestration is where this distinction produces its costliest effects, because the orchestrator often combines four responsibilities: proposing a plan, authorising its steps, allocating resources, having them executed. A known process is driven by a workflow or a state machine, with the LLM intervening only at the points that require interpretation. A generative orchestrator remains relevant for open-ended tasks, provided that what it decides is bounded: it chooses among authorised capabilities without creating rights, and its plan is a proposal subject to control. A plan is not an authorisation. The state of the task (what has been done, authorised, consumed) must not reside solely in the model's context, which gets truncated, summarised and lost; it must be externalised in an explicit structure that the LLM consults without being its custodian. Stopping conditions, finally, must be imposed from outside. An agent's autonomy is measured by what it can decide; its safety, by what it cannot prevent.

Encapsulating each agent in a microservice brings agentics back to classic distributed-architecture questions; this settles the question of deployment, not that of number, nor that of what deserves to be an agent. It is precisely because it makes the agent ordinary that this encapsulation makes visible the granularity errors that distributed architecture took fifteen years to document, and the answers it provided: versioned interface contracts rather than conversations, compensations rather than blind retries, division by capability rather than by organisation chart.

5. An agent does not hold authority over its effects

This section addresses the second error, which exists with or without personas. It sets out principles; their detailed implementation belongs to a separate technical treatment.

An LLM processes a transfer request and produces a structured object containing an amount and an approval field set to true. A rules engine checks that the amount is under the ceiling and that the approval is true. The check is deterministic, it works as specified, and it authorises an operation that should never have been authorised. The amount is not at issue: the agent may propose it, a ceiling may bound it. The approval is, because an unattested assertion by the controlled party has been accepted as evidence. A

deterministic check can be captive. One must distinguish the parameters of the action, which the agent may produce; the authority attributes (identity, rights, approvals), which must be attested by independent sources; the business facts, which may require external verification; and the policies, which the agent must be able neither to modify nor to negotiate. For the effects concerned, the check must remain outside the authority of the controlled component, cover the identified execution paths, verify authority attributes against appropriate sources and produce traces whose integrity can be established, according to the threat model adopted. These requirements are in line with the classic principles of complete mediation and separation of privilege (Saltzer and Schroeder, 1975). Independence nevertheless has a limit: an attested source may be wrong, an authentic approval may have been given on the strength of false information. Independence reduces manipulation; it does not guarantee correctness.

An agent that reads a document containing a malicious instruction may issue a perfectly typed email-sending call, to an external recipient, with a confidential attachment, exactly within the scope of its permissions. Schema respected, rights respected, illegitimate effect: this is the confused deputy described by Hardy (1988), of which prompt injection is the contemporary version. The difficulty runs deeper than it appears. If the user has authorised sending a report to an external partner, the controller cannot infer from the sending parameters alone which part of the content stems from the request and which part stems from the injection. Authorising a type of action is not authorising its content. For critical effects, the authorisation must be bound to a specific object: this content, this recipient, or a verifiable representation such as the hash of the approved document. Even then, three guarantees remain distinct: the identity of the executed object, the legitimacy of the approval, the substantive conformity of the object to the intent and to the constraints. The first two can often be attested mechanically; the third may remain open. An independent check can guarantee the enforcement of a decision without guaranteeing the correctness of that decision.

The risk specific to LLMs is instruction laundering: an instruction contained in data without authority is reformulated, then reintroduced into the system as a legitimate instruction, for example in the form of a step in the proposed plan. A generative output may legitimately become an instruction, when a human owner or a business service examines and validates it; the problem is not the elevation of authority, but elevation without a distinct, attributable and logged act of authorisation. Authority comes from the act that validates, not from the text that proposes. The same principle applies to confidentiality: a summary may reveal protected information without quoting it, and declassifying an output requires a validation bearing on the information produced, not a mere relabelling. CaMeL (Debenedetti et al., 2025) instantiates the separation between control flow and untrusted data: intent is extracted from the trusted query, an interpreter tracks the provenance of values and applies policies to tool calls. The authors report 77% of AgentDojo tasks solved with security they describe as provable, compared with 84%

without defence. This result bears on their security model and on a benchmark; it does not guarantee the security of a deployed system, but it illustrates a difference in kind between a constraint imposed by the architecture and improved model resistance.

Control does not apply only at the last tool call. An agent's output is not an action, but it may become the decision input of another agent; every output must therefore carry an explicit status (proposal, information or authorised action). Delegation obeys the same logic, provided two operations are distinguished. Delegation of authority transmits part of the delegator's rights and must never create any. A request for execution addressed to a distinct authority solicits a component that holds its own rights: a payment service executes a transfer that the requesting agent is not entitled to carry out, after verifying, according to its own policy, that the request is authorised.

Two distinctions complete these principles. A permitted action is not a funded action: an authorised agent may commit resources that do not merit it, and several authorised agents may together exceed a shared budget; resource allocation falls under a mechanism of its own, independent of the agents. An authorisation is not a transaction: between the authorisation granted, the order issued, the order accepted and the effect confirmed, the last event may remain unknown, and a blind retry is not proof that the previous attempt failed; this is exactly how one pays twice. The architecture must provide for an indeterminate state and an explicit policy for handling it. Governance does not consist in producing a reassuring log, but in making visible the situations where evidence is lacking.

That leaves the open constraints, which these mechanisms do not settle. The right question is not "which verifier?" but "on what basis should one verify?". First, what can be confronted with an external reference: source data, independent computation, applicable rule, documented expert judgement; a clinical summary is not mechanically checkable, but the presence in the source record of the elements it cites often is. Next, what can only be submitted to a second judgement, bearing in mind that model diversity does not replace an external reference and at best signals that one is needed. Finally, what calls for abstention for lack of sufficient evidence, provided the uncertainty signal is calibrated over a defined domain. Human oversight finds its place here: proportionate to the risk, targeted and sufficiently informed to allow effective judgement, it bears in particular on high-stakes open constraints and on the correctness of rules, rather than on the manual repetition of mechanical checks that have already been qualified. It presupposes humans practised in judging; otherwise mechanical control becomes the last custodian of an expertise that no one any longer knows how to verify.

6. What the opening example shows, and what it does not

In a trial emulation of this kind, three families of operations could be entrusted to an LLM persona: checking data consistency against an ontology of variables, identifying

adjustment sets on the causal graph, and checking the temporal admissibility of variables. All three can be entrusted to executable mechanisms. Data consistency typically falls under shape validation, whose result depends on the declared rules and not on how a question is phrased; the graphical criteria and temporal admissibility are verified by reproducible procedures, applied after clinical review of the graph. These checks identify admissible adjustment sets relative to the graph; they guarantee neither the quality of the graph nor the identifiability of the effect in the data. The executability of the check does not amount to validation of the causal inference. The boundary between what falls under clinicians' judgement and what falls under computation is exactly the one this article proposes to make explicit.

Conclusion: capability, autonomy, authority

This article proposes a design criterion some of whose consequences can be guaranteed; it does not demonstrate that an architecture applying it is preferable overall. That second question is experimental. Its main programme follows directly from the thesis: for comparable functions, requirements and resources, what value does substituting a specialised capability for an LLM persona bring, and what additional value does autonomy bring? Answering it requires defining, for each comparison, the task population, the comparator (including a single LLM equipped with the same tools, a necessary comparator to help distinguish the effect of specialising computation from that of multiplying instances), the estimand, the horizon, the primary endpoint, and what an equal budget means when tokens, compute time, monetary cost and energy are not interchangeable. The questions of control independence, budgetary guarantees, change management and validation cost are derived programmes.

The thesis has limits. The characterisation of a function evolves, and with it the allocation that follows from it. The separation of components, which facilitates establishing certain proofs, also multiplies the interfaces and elements to be qualified; its net balance is not settled in advance. Attestation and intent-binding mechanisms have an engineering and usability cost that may, in certain uses, exceed the value of the guarantees they provide. The treatment of open constraints relies on arrangements less established than mechanical checks. And the mechanisms proposed in section 1 to explain persona-based agentification remain hypotheses.

The most robust proposition is not that determinism costs less than an LLM: that balance depends on prices, volumes and models, and it will shift. It is that an agentic architecture must separately justify three things that a persona conflates: the capability it mobilises, evaluated over a defined domain of use; the autonomy it grants, which must demonstrate its value; the authority it withholds from the component that proposes. This separation remains relevant as models become cheaper, more capable or more reliable, because it bears not on what they can do, but on what they are allowed to decide.

The criterion can be checked against any existing architecture, in three questions.

- For each component: what capability does it provide, and over what domain of use has it been evaluated?
- For each autonomous decision loop: what demonstrated value justifies this autonomy?
- For each external effect: what share of the rules, evidence, resources and mechanisms that authorise it remains beyond the reach of the component that proposes it?

An architecture that cannot answer these three questions does not merely have a problem with the number of its agents. It has not established what it computes, why it delegates and what it authorises.

References

Anthropic. "How we built our multi-agent research system". Anthropic Engineering, June 2025. https://www.anthropic.com/engineering/multi-agent-research-system

Cemri M., Pan M. Z., Yang S. et al. "Why Do Multi-Agent LLM Systems Fail?". NeurIPS 2025, Datasets and Benchmarks Track. arXiv:2503.13657.

Conway M. E. "How Do Committees Invent?". Datamation, 14(4), April 1968, pp. 28–31.

CrewAI. Documentation, agent definition (role, goal, backstory attributes). https://docs.crewai.com

Debenedetti E., Shumailov I., Fan T. et al. "Defeating Prompt Injections by Design". arXiv:2503.18813, v2, June 2025.

Hardy N. "The Confused Deputy (or why capabilities might have been invented)". ACM SIGOPS Operating Systems Review, 22(4), 1988, pp. 36–38.

Hernán M. A., Robins J. M. "Using Big Data to Emulate a Target Trial When a Randomized Trial Is Not Available". American Journal of Epidemiology, 183(8), 2016, pp. 758–764. doi:10.1093/aje/kwv254.

Hernán M. A., Sauer B. C., Hernández-Díaz S., Platt R., Shrier I. "Specifying a target trial prevents immortal time bias and other self-inflicted injuries in observational analyses". Journal of Clinical Epidemiology, 79, 2016, pp. 70–75.

Linux Foundation. Announcement of the launch of the Agent2Agent (A2A) project, June 2025 (protocol initiated by Google in April 2025).

Saltzer J. H., Schroeder M. D. "The Protection of Information in Computer Systems". Proceedings of the IEEE, 63(9), 1975, pp. 1278–1308.