

Allocating cognitive work across people, language models and deterministic tools: A task-analytic method

Simon Vajda

simonvajda@gmail.com

Deakin University

Research Article

Keywords: function allocation, levels of automation, adaptive automation, human–AI teaming, language-model agents, hierarchical task analysis

Posted Date: October 26th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11312211/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Abstract

People supervising language-model agents need to know what the agents decided and when to intervene. Reading every response becomes impractical as more work is delegated. We extend a task-analysis method to allocate decisions among deterministic tools, small language models, reasoning models and people, with each task represented as an observe, orient, decide and act cycle. Analysts specify its inputs, decision and output, and derive the information supervisors need from their goals. Allocation depends on how formally the decision can be specified, the cognitive control it demands, its Cynefin domain and whether it establishes facts or makes a normative choice. A machine tier withholds a decision when it declines, fails the output checks or falls below its confidence threshold. The decision then passes to the next tier. People retain normative choices and may store them as scoped policies. In a computational comparison with two open-weight models on 64 synthetic tanker-join instances, the allocated system was correct on 89.6% of required epistemic decisions, against 67.2% for a reasoning model given the whole task. It used about a third of the input tokens and an eighth of the output tokens. A scripted supervisor supplied 13% of its scored decisions. The small-model tier withheld 97% of its wrong candidates and the reasoning-model tier 93%, counting refusals, output checks and confidence thresholds together. The prototype failed to keep one training choice with people. Effects on supervisors' situation awareness and workload require a study with practitioners.

1 Introduction

People supervising language-model agents need to know what happened, whether the work is on course and when to intervene. Agents can search, read, write code and draft documents faster than a person can review their output. The usual interface is a conversation: the person makes a request, reads the response and decides what to ask next. Each additional agent adds material to review. Engineers at Anthropic reported approximately four times as many tokens for single agents as for chat interactions, and approximately 15 times as many for multi-agent systems (Hadfield et al. 2025). Tokens are the units of text a model reads and writes. Woods and Hollnagel (2006) called the tendency to absorb gains in machine speed by delegating more work the law of stretched systems, with the resulting demand falling on the supervisor's attention.

Cognitive engineering has studied this problem for more than four decades. Bainbridge (1983) described the ironies of automation, Woods et al. (2002) examined data overload, and Endsley (2017) described the automation conundrum. As automation becomes more capable and reliable, supervisors can lose situation awareness and become less likely to intervene when needed, with comparable effects reported for generative AI (Endsley 2023a; Simkute et al. 2025).

Model routers and cascades reduce cost by choosing the least costly model expected to answer a query adequately (Chen et al. 2024; Ong et al. 2025). Their unit of allocation is the query. Multi-agent frameworks instead assign roles from human workflows (Hong et al. 2024) or organizational hierarchies (Kim et al. 2025). These methods provide little guidance on the information a supervisor needs or on deriving each agent's decisions from the task itself. Cemri et al. (2025) grouped multi-agent failures into problems of

system design and specification, misalignment between agents and weak verification of results. Task analysis provides a way to address these problems before operation by specifying each decision, the information passed between agents and the checks on their outputs.

Function allocation is the division of work between people and machines, and its principal frameworks were developed for automation whose behavior designers specify (Parasuraman et al. 2000; Sheridan and Verplank 1978). Guidance for situation awareness therefore recommends automating routine tasks while retaining higher-level cognitive functions with people (Endsley et al. 2003; Endsley 2017). Language models can interpret ambiguous instructions and weigh evidence expressed in natural language, but their reliability and the cost of repeated calls complicate that division. We distinguish four classes of decision maker, called tiers: deterministic tools, small language models, reasoning models and people.

This paper extends DCARR (Describe, Collect, Analyse, Recommend, and Report), a model-driven framework developed for personnel conducting distributed synthetic training for the Royal Australian Air Force (Vajda 2023). DCARR decomposes goals by hierarchical task analysis into task models, each following an observe, orient, decide and act (OODA) cycle (Boyd 2018). Observations are represented as finite state machines, orientation as queryable data and decisions as algorithms; a task becomes an automation candidate when its states, actions and decisions can be defined, modeled and represented. Decisions that cannot be encoded as algorithms, and tasks whose cause-and-effect relations cannot be determined in advance, remain with people. We extend that analysis to decisions made by language models.

We propose a method that derives the allocation across tiers from task analysis and treats supervisors' information requirements as an input to the design. We found no existing method that combines these features. The aim is to preserve situation awareness while limiting what supervisors must read, the context each model processes and the cost of each decision. In the tanker example, software computes an intercept heading, models interpret a radio instruction, and a person decides whether to disobey it for training purposes.

The evaluation addresses two research questions:

- RQ1. Relative to a single reasoning model given the entire task, does allocation under the method reduce token consumption and the largest context processed by any model, and at what cost to accuracy?
- RQ2. Do abstention and escalation route the decisions that a tier would decide incorrectly to a higher tier, and do requests to people decline across repeated runs as policies accumulate?

The paper contributes a seven-step design method, an ordered allocation rule, a prototype runtime and a computational comparison. The rule assigns each decision to a tier from the attributes of its task and sets the order of tiers through which a withheld decision passes. The unit of allocation is therefore the decision, where earlier frameworks allocate functions. Hand-off specifications state what each task passes to its parent task or to supervisors. The method applies beyond flight simulation, although both the worked

example and the evaluation use that setting. Its effects on supervisors' situation awareness and workload require a separate experiment with practitioners (Section 6.2).

Section 2 sets out the basis for allocation. Section 3 describes the method and applies it to the tanker scenario. Sections 4 and 5 describe the runtime and computational comparison. Section 6 relates the findings to function allocation and identifies the limits of the study.

2 Background

The method builds on research in function allocation, cognitive control, situation awareness and information fusion, human interaction with generative AI, and language-model agents. Section 2.6 compares the nearest existing methods (Table 1).

2.1 Function allocation and the levels-of-automation debate

Function allocation concerns which functions of a system should be performed by people and which by machines. The earliest systematic answer was the list of relative human and machine capabilities compiled by Fitts (1951). The list has been widely criticized, but de Winter and Dodou (2014) argued that it has persisted because it captures the most important regularity of automation.

Sheridan and Verplank (1978) replaced a capability list with levels of shared control, ranging from no computer assistance to a computer that decides and acts without reference to a person. Parasuraman et al. (2000) applied levels separately to information acquisition, information analysis, decision selection and action implementation. They proposed choosing each level primarily by its consequences for human performance, including workload, situation awareness, complacency and skill degradation.

A meta-analysis of 18 experiments supported the trade-off predicted by this framework (Onnasch et al. 2014). Higher degrees of automation improved routine performance and reduced workload, but they impaired performance when the automation failed and reduced situation awareness. These costs were most pronounced once automation extended from information analysis to action selection. Onnasch et al. concluded that a given allocation can favor either routine performance or the retention of situation awareness and skill, but not both.

One response to this trade-off is to vary the allocation during operation. In adaptive automation, the system changes the allocation or the level of automation in response to a trigger, such as the operator's cognitive state, the task, or events in the environment (Feigh et al. 2012; Kaber and Endsley 2004). In adaptable automation, the operator makes that change. Miller and Parasuraman (2007) argued for adaptable automation in the form of delegation, in which a supervisor assigns tasks to automation much as to a human subordinate. They proposed that delegation requires a hierarchical task model shared by the supervisor and the automation, recast a level of automation as a pattern of task-based roles and authorizations, and implemented the approach in a prototype called Playbook. Endsley (2017) identified levels of automation, adaptive automation, and granularity of control as the central paradigms for human interaction with autonomous systems.

The value of stages and levels for design remains disputed. Kaber (2018) identified imprecise definitions and assumptions about human behavior, but argued for retaining the frameworks until a sound alternative was available. Jamieson and Skraaning (2018) questioned their predictive value, whereas Wickens (2018) defended stages as a model of performance trade-offs and Endsley (2018) defended levels as an aspect of autonomy design. Johnson et al. (2018) proposed coactive design and interdependence analysis. Cummings (2018) related skill-, rule- and knowledge-based reasoning to expertise and uncertainty as a basis for allocation.

These frameworks describe how much of a function specified software performs. Language-model systems also require a choice among tools, models of different capabilities and people. Interpreting an ambiguous instruction may require a model, while calculating a heading can use software; designers therefore need to choose which tier makes each decision and what happens when that tier cannot decide reliably.

2.2 Cognitive control and the character of tasks

Rasmussen (1983) distinguished three levels of cognitive control. Skill-based behavior proceeds as smooth, automated action; rule-based behavior applies stored rules to familiar signs; and knowledge-based behavior reasons from goals in unfamiliar situations. Breton (2008) represented the three levels within the OODA loop, and Cummings (2018) used them, together with expertise and uncertainty, to frame the allocation of functions between people and autonomous systems. Because the levels characterize the demands a decision places on whoever makes it, the method uses them to distinguish decisions suited to a small model from those that require a reasoning model.

The Cynefin framework classifies situations by the relation between cause and effect (Kurtz and Snowden 2003; Snowden and Boone 2007). In ordered domains the relation is either evident or discoverable through analysis, whereas in unordered domains it is apparent only in retrospect or changes too rapidly to be determined. Each domain implies a different mode of decision making. Where the relation is evident, decision makers sense, categorize, and respond, and where analysis can reveal it, they sense, analyze, and respond. Where it is apparent only in retrospect, they probe, sense, and respond, and in chaos they act, sense, and respond. Kurtz and Snowden (2003) further noted that understanding a complex situation requires multiple perspectives. The domain labels have changed across versions of the framework, from known and knowable (Kurtz and Snowden 2003) to simple and complicated (Snowden and Boone 2007); we use the labels clear, complicated, complex, and chaotic.

The four tiers resemble dual-process accounts of cognition, in which fast, automatic processing handles most activity and slow, deliberate processing intervenes when required (Evans and Stanovich 2013; Kahneman 2011). We treat the resemblance as an analogy only. The dual-process typology is contested (Melnikoff and Bargh 2018), whereas Rasmussen's levels are established in cognitive engineering, describe the demands of a decision rather than properties of a mind, and have already been applied to allocation (Cummings 2018).

2.3 Situation awareness and information fusion

Situation awareness comprises the perception of elements in the environment, the comprehension of their meaning, and the projection of their status in the near future (Endsley 1995). Automation can degrade situation awareness. Endsley and Kiris (1995) automated a navigation task with an expert system and found that reduced situation awareness accompanied slower decisions after the system failed. The degree of control operators retained over the automation moderated this loss, which Endsley and Kiris attributed primarily to the shift from active to passive processing.

For human–AI teams, Endsley (2023b) distinguished the awareness that team members require of the task, of one another's state as agents, and of the teamwork itself, and she described a design process for displays that support each. The hand-offs and abstention records in our method provide one means of supporting awareness of machine agents, by reporting what each tier decided, what it declined to decide, and why.

Information fusion combines data from multiple sources into a form that supports decisions. The revised Joint Directors of Laboratories model organizes fusion into levels, from signals through objects and situations to the impact of a situation (Steinberg et al. 1999). Blasch and Plano (2002) observed that the model described only machine processing and proposed an additional level, user refinement, for the person the fusion system serves. The OODA loop originated in Boyd's briefings on conflict and command (Boyd 2018). The task models in this paper take their OODA representation from the fusion literature, in which a Canadian research program modeled decision makers with cognitive and modular variants of the loop (Breton 2008).

2.4 Human interaction with generative AI

Bainbridge (1983) observed that automation leaves people with the tasks designers could not automate, often the most difficult ones, while removing opportunities to practice. It also assigns people a monitoring role for which they are poorly suited. Endsley (2023a) described further ironies of AI: hidden limitations and biases, difficulty judging reliability, and difficulty understanding and adapting to the system. Simkute et al. (2025) identified four causes of productivity loss with generative AI. Users move from producing work to evaluating it, workflows are restructured unhelpfully, interruptions increase, and easy tasks become easier while hard tasks become harder.

Reliance should match the capabilities of the automation (Lee and See 2004). Excessive reliance is misuse, and too little reliance is disuse (Parasuraman and Riley 1997). Allocation of attention contributes to complacency and automation bias (Parasuraman and Manzey 2010). In an experiment with 199 participants, Buçinca et al. (2021) found that cognitive forcing functions reduced overreliance more than explanations, although participants rated the most effective designs least favorably. Shaw and Nave's (2026) findings on cognitive surrender, discussed in Section 3.5, raise a related concern for generative AI.

Klein et al. (2004) set out 10 challenges for making automation a team player in joint activity with people, and Johnson et al. (2014) argued that systems should support interdependence between people and machines through observability, predictability, and directability. A conversational interface that returns a single extended response to each request satisfies few of these requirements. The person must read the

output to observe what occurred, cannot anticipate the content of the next response, and can redirect the work only by composing a further request.

2.5 Language-model agents

Engineering practice has moved from single requests to agents and to systems of multiple agents. The engineering report cited in the Introduction also found that token usage alone explained 80% of the variance in performance on a web-browsing evaluation (Hadfield et al. 2025). The report is not peer reviewed, but its figures suggest that much of the benefit of multiple agents comes from spending more tokens.

Two lines of work reduce this expenditure or add structure to it. Cascades and routers direct each query to the least costly model expected to answer it adequately (Chen et al. 2024; Ong et al. 2025). MetaGPT encodes standardized operating procedures as sequences of prompts and assigns roles to agents in the manner of an assembly line (Hong et al. 2024).

Tiered Agentic Oversight allocates tasks by complexity through agents modeled on a clinical hierarchy of nurse, physician and specialist. A physician at the top tier improved triage accuracy from 40% to 60% in its user study (Kim et al. 2025). OrchVis converts a person's intent into structured goals, verifies execution and displays dependencies between agents (Zhou 2025). These functions let the person supervise the workflow without directing every step.

2.6 Nearest methods

Table 1 compares what the nearest methods allocate, how they choose the allocation and what each leaves unresolved for our design problem.

Table 1
Methods nearest to the proposed method

Method	Object and basis of allocation	Limitations for the present problem
Types and levels of automation (Parasuraman et al. 2000)	Each stage of a function, to a level of automation chosen by its consequences for human performance	Assumes a single class of automation whose behavior designers specify
Skill, rule, and knowledge taxonomy for autonomy (Cummings 2018)	Relates the three levels of cognitive control to expertise and uncertainty to inform allocation	A conceptual taxonomy rather than a procedure for deriving allocations from a task analysis; predates language-model agents
Coactive design (Johnson et al. 2014, 2018)	Designs for the interdependence of people and machines through observability, predictability, and directability	Does not address allocation among automated components that differ in capability and cost
Delegation interfaces (Miller and Parasuraman 2007)	Tasks in a hierarchical task model shared by the supervisor and the automation, delegated by the supervisor during operation	Allocation is chosen by the supervisor at run time rather than derived from task attributes; does not distinguish automated agents by capability and cost
Model routing and cascades (Chen et al. 2024; Ong et al. 2025)	Each query, to a stronger or a weaker model, by a learned estimate of quality and cost	Allocates per query at run time; supervisors' information requirements are outside its scope
Standardized operating procedures in multi-agent systems (Hong et al. 2024)	Roles and prompt sequences for agents, taken from human workflows	Roles are derived from a software team's workflow rather than from task analysis; people are not part of the allocation
Tiered Agentic Oversight (Kim et al. 2025)	Tasks, by complexity, through tiers of agents modeled on a clinical hierarchy, with a physician at the top in a user study	Tiers follow an organizational hierarchy; evaluated on safety benchmarks and triage accuracy
OrchVis (Zhou 2025)	Goals and tasks among language-model agents, with automated verification and a planning view for a supervising person	An interface for supervision; no task-analytic basis for allocation is reported
This method	Each decision of each task model, to deterministic tools, small models, reasoning models, or people, by describability, task type, level of control, Cynefin domain, and decision kind; hand-offs to people follow their information requirements	Depends on the quality of the underlying task analysis

Existing methods offer separate parts of the design. Cognitive engineering allocates functions between people and specified automation; machine-learning methods allocate queries among models by cost and

quality. Agent frameworks add structure from workflows and organizational hierarchies. Our contribution is to derive the allocation of decisions across tools, models and people from task analysis, using supervisors' information requirements to specify their hand-offs.

3 Method

Analysts use the method to specify which tier begins each decision and what information the task passes on. It extends Vajda's (2023) task analysis to decisions expressed in natural language, with conditions for passing a decision to another tier and specifications for its output. As in any task analysis, analysts observe the work and validate their task models against it.

The tanker example shows how these choices differ. A tool computes the heading on which an aircraft intercepts the tanker. A small model interprets the radio instruction, and a reasoning model may resolve an ambiguous addressee. The exercise supervisor decides whether to comply or deliberately create a problem for the trainee. That choice concerns what ought to happen and therefore remains with the person responsible for the exercise, whose decision can be stored as a policy for use when the same situation recurs. The supervisor receives questions requiring judgment and a fused summary of the run.

3.1 Terms

A language model generates text from input text. We distinguish small models, used for less costly calls, from reasoning models, used here for decisions requiring multistep inference. Reasoning models are larger, slower and more costly in this comparison. An agent runs a model in a loop, allowing it to call tools such as a search engine or calculator and use their results. The harness is the software that assembles input, calls the model, executes tool calls and records the results. A model's context is the text it receives in one call, and its context window is the maximum it can receive. Both are measured in tokens, the subword units models process; an English token averages approximately four characters.

A task model represents one task from the hierarchical task analysis as an OODA cycle (Vajda 2023). A tier is a class of decision maker, as defined in Section 1. Abstention means withholding a decision; escalation passes it to the next tier. A policy stores a person's decision for reuse in the same situation. A hand-off is the output a completed task passes to its parent task or to supervisors, the people who assign work to the system and are accountable for its outcomes.

3.2 Overview of the steps

The method comprises seven steps (Table 2). Steps 1 to 3 follow the task analysis of Vajda (2023), Step 5 extends its characterization of tasks, and Steps 4, 6, and 7 are new. The output is a set of task models that a harness can execute, as described in the Implementation section. Figure 1 summarizes the resulting allocation: each decision begins at the tier the allocation rule assigns and moves to a higher tier only when the current tier abstains.

Note

Tiers run from deterministic tools to people, whose attention is the scarcest resource. Each decision begins at its assigned tier and escalates when that tier abstains; clause letters refer to Table 4. In complex domains, a reasoning model passes its decision to people as a proposal.

Table 2
Steps of the method

Step	What it produces	Source
1. Map objectives to goals	A map in which every goal serves an objective and every objective has a goal	Vajda (2023)
2. Decompose goals into task models	A task hierarchy with plans, and a plain-language description of each task: action, object, output, tooling, and context	Hierarchical task analysis (Annett 2003; Stanton 2006); Vajda (2023)
3. Represent each task model as an OODA cycle	The observations, orientation entities, decision, and action of each task model	Vajda (2023)
4. Derive supervisors' information requirements	Supervisors' goals, the decisions each goal requires, and the information each decision requires	Goal-directed task analysis (Endsley et al. 2003)
5. Characterize each task model	Describability, task type, level of cognitive control, Cynefin domain, decision kind, reversibility, and impact	Vajda (2023), extended here
6. Allocate decisions and actions	A tier for each decision and a level of automation for each action	New (Table 4)
7. Specify hand-offs	The fields, fusion level, and token budget of each hand-off	New

3.3 Describing and representing task models (Steps 1–3)

Analysts begin by recording the scenario, objectives and goals, with each goal mapped to at least one objective (Step 1). They decompose each goal by hierarchical task analysis until its subtasks can be described (Step 2; Annett 2003; Stanton 2006). A description answers five questions: what action is performed, on what object, with what output, using what tools and in what context; a subtask that remains undescribed is decomposed again. If neither description nor decomposition is possible, clause A assigns it to people (Table 4).

Each described subtask becomes a task model with four elements (Step 3). Observations are the inputs the task reads. Orientation entities are the additional knowledge a decision requires, such as the commander's intent, capabilities, and objectives, represented as queryable data. The decision is the reasoning from observations and orientation to a choice, and the action is the output that follows. These four elements correspond to the four stages of information processing that Parasuraman et al. (2000) used to define types of automation: information acquisition, information analysis, decision selection, and action implementation. The correspondence allows the method to assign a level of automation to each

stage of each task model. Task models form a three-level hierarchy: an OODA metamodel, the task models that instantiate it, and instances that record each occurrence of a task.

Orientation entities acquire additional importance when the decision maker is a language model. Because a model receives only the observations and orientation entities declared in its task model, these declarations fully specify its context. An analyst can therefore inspect, before any run, all the information a model will receive when it makes a decision.

3.4 Deriving supervisors' information requirements (Step 4)

Step 4 establishes supervisors' information requirements before any decision is allocated. Analysts apply goal-directed task analysis (Endsley et al. 2003) to identify supervisors' goals, the decisions each goal requires, and the information each decision requires at each level of situation awareness. These levels are perception of the relevant elements, comprehension of their meaning for the goal, and projection of their future status (Endsley 1995). The result is a set of requirements that the hand-offs to supervisors must satisfy (Step 7).

Allocation determines what supervisors can observe, so deriving their information requirements first gives analysts a basis for judging how a proposed allocation affects the information available to those directing the work. This matters when agents produce output faster than a person can read it and gains in speed invite more delegation (Endsley 2017; Woods and Hollnagel 2006).

Woods et al. (2002) described data overload as too much displayed data, too much to process in the available time, or difficulty finding what is relevant. They identified relevance as the harder problem because it depends on context. A lengthy response adds data and delegation reduces processing work, but neither establishes relevance; Step 4 ties relevance to supervisors' goals so analysts and supervisors can examine it before operation. Requirements will still need revision as new situations arise.

3.5 Characterizing task models (Step 5)

Step 5 records seven attributes of each task model (Table 3). Task type, Cynefin domain, and impact are taken from Vajda (2023), and describability adapts that work's tests of whether a task can be described and whether its decision can be encoded as an algorithm. The level of cognitive control, the decision kind, and the reversibility of the action are added here.

Table 3
Attributes recorded for each task model

Attribute	Values	Source
Describability	State machine, algorithm, plain language, not describable	Adapted from Vajda (2023)
Task type	Functional, procedural	Vajda (2023)
Level of cognitive control	Skill, rule, knowledge	Rasmussen (1983); Breton (2008)
Cynefin domain	Clear, complicated, complex, chaotic	Kurtz and Snowden (2003); Snowden and Boone (2007); Vajda (2023)
Decision kind	Epistemic, normative	New
Reversibility of the action	Reversible, irreversible	New
Impact	Distance from the objective; whether failure causes the objective to fail	Adapted from Vajda (2023)

Describability records how formally a decision can be specified. A state-machine decision has observations and outcomes represented by a deterministic finite automaton. An algorithmic decision can be expressed as pseudocode over declared inputs, while a plain-language decision answers the five description questions but lacks an adequate algorithm for the scope analyzed. A decision is not describable when its description itself fails. Vajda (2023) treated only decisions encoded as algorithms as automation candidates. We extend allocation to the plain-language decisions, while leaving undescribable decisions with people.

Task type distinguishes functional tasks, which apply a set of rules to an observable state, from procedural tasks, which invoke other tasks and require multiple orientations and decisions. The level of cognitive control records whether the decision is skill-based, rule-based, or knowledge-based (Rasmussen 1983). The method uses task type and level of control together to distinguish decisions that a small model can make from those that require a reasoning model.

The Cynefin domain records whether cause and effect in a task are clear, complicated (knowable through analysis), complex (apparent only in retrospect), or chaotic (Kurtz and Snowden 2003; Snowden and Boone 2007). The domain depends on who encounters the task. The scenario used in the worked example illustrates this. It describes the radio instruction as clear for trainees with proficiency but complicated for students in initial training. Flying to the tanker is likewise clear for a pilot or an experienced simulation pilot but quickly becomes complicated for anyone else (Vajda 2023, pp 67–68). We record the domain as it presents to a competent practitioner and rely on abstention when a tier finds the task more difficult. This is a simplification, and a task model may require a different domain for each tier.

Decision kind is the first additional attribute. An epistemic decision establishes what is the case, for example to whom a radio call is addressed or where the tanker is located. A normative decision

determines what ought to be done, for example whether to present a trainee with a problem or how much risk to accept. Machine tiers may make epistemic decisions, whereas normative decisions are made by people or by applying a policy that people have set for the same situation.

We retain normative choices with people because supervisors are accountable for them and need control of the system (Endsley et al. 2003; Endsley 2017). A model's recommendation may also discourage scrutiny. Shaw and Nave (2026) reported three preregistered experiments with 1,372 participants. In the first, a correct AI assistant raised reasoning accuracy by 25 percentage points relative to no assistant, while an incorrect assistant reduced it by 15 points. These preprint findings suggest a risk of adopting the recommendation offered. We therefore present people with options when a normative decision is required.

Reversibility, the second additional attribute, records whether the action can be undone or corrected after the fact, and it determines the level of automation for the action stage (Step 6). Impact records a task's distance from the objective, where zero denotes a task on the critical path, and whether the task's failure would cause the objective to fail.

3.6 Allocating decisions and actions (Step 6)

Step 6 applies an ordered rule to each task model (Table 4). The first clause whose condition holds determines the tier that makes the decision.

Table 4
The allocation rule

Clause	Condition	Tier
A	The decision can be neither described nor decomposed	People
B	The decision is normative	People, unless a stored policy covers the situation
C	The Cynefin domain is chaotic	People
D	The decision is a state machine or an algorithm	Deterministic tools
E	The decision is in plain language and the domain is complex	Reasoning models, proposing only
F	The decision is in plain language, and the task is functional at the skill or rule level	Small models
G	Any other plain-language decision	Reasoning models
Note. Clauses apply in order; the first that holds sets the tier.		

Clauses A, C and D retain the earlier task-analysis rules (Vajda 2023). That work's review of the literature associated clear tasks with autonomy, complicated tasks with a person in the loop, complex tasks with decision support and chaotic tasks with human control. We retain human participation through the action

levels and decision support through clause E. Clauses B and E to G add normative authority and allocation of plain-language decisions.

Clause order matters. Clause B keeps normative decisions with people before clause D can allocate an algorithm to software. A person can set a policy that specifies such a decision. Clause E ensures that a reasoning model in a complex domain proposes options for people to judge. The proposal gives a rationale for each option because complex situations call for probing and multiple perspectives (Kurtz and Snowden 2003).

Clauses F and G distinguish narrow rule-based decisions from procedural or knowledge-based decisions. We expect a small model to handle a functional task applying rules to an observable state, an expectation examined in the evaluation, while procedural and knowledge-based decisions start with a reasoning model. Routers make a related choice during operation from a learned estimate of each query's difficulty (Chen et al. 2024; Ong et al. 2025). Here analysts make the initial allocation from the task attributes, and supervisors can review it before operation.

Clauses E to G depart from a guideline in design for situation awareness, namely that automation should assist with routine tasks rather than assume higher-level cognitive functions (Endsley et al. 2003; Endsley 2017). Section 6.1 discusses the departure and the safeguards that accompany it.

Step 6 sets how much authority the action stage has, using levels of automation (Parasuraman et al. 2000; Sheridan and Verplank 1978). People retain authority over their decisions, and a model's output in a complex domain is a proposal; an irreversible action waits for approval unless a policy covers it. An action on the critical path whose failure would cause the objective to fail proceeds subject to veto. Other actions proceed and are reported, including through any hand-off addressed to people.

People set confidence thresholds because the tolerated risk is a normative choice. The prototype defaults to 0.9 for tasks on the critical path or whose failure would cause the objective to fail, and 0.7 for other tasks. These thresholds govern one of several abstention checks. Section 5 examines their relation to observed accuracy using held-out calibration for the study.

3.7 Specifying hand-offs (Step 7)

Each task model declares the fields, data types, fusion level and token budget of its hand-off. A parent task receives its subtasks' completed outputs, while supervisors receive the hand-offs addressed to them together with the judgment requests, approval requests and notices required by the action level. The hand-off carries the result without a transcript of the reasoning or the subtask's inputs.

We use the revised Joint Directors of Laboratories fusion levels (Steinberg et al. 1999). Level 1 identifies objects and their states. Level 2 relates them to one another and the goal. Level 3 assesses the situation's implications for the goal, including its likely development. Supervisors' hand-offs use Level 2 or 3 and contain the information identified in Step 4.

Task analysis proceeds in the opposite direction to information fusion, decomposing a decision into the information that informs it. The method uses both directions: Steps 2 and 3 decompose each decision into observations and orientation entities at design time, and at run time each hand-off fuses a task model's result one level up the hierarchy. The same task models therefore support both analysis and fusion.

Hand-off budgets limit what supervisors must read and what the next model processes, addressing the limited attention of people and the data overload caused by displays that disregard it (Woods et al. 2002). Language-model accuracy also declines when relevant information lies in the middle of a long context (Liu et al. 2024). Increasing input length can reduce performance even on simple reasoning tasks (Levy et al. 2024). These findings support a common design response, a budget on each hand-off, although they concern different mechanisms in people and models.

Task models and hand-offs address the failure categories described by Cemri et al. (2025). They specify each agent's decision and constrain the information passed between agents. Deterministic tools verify outputs where the decision can be specified as an algorithm or state machine. Kambhampati et al. (2024) proposed a related arrangement for planning, with models generating candidates and external verifiers checking them.

3.8 Running the task models

The harness runs task models depth first in hierarchical task-analysis plan order. Each model completes an OODA cycle by reading its observations, retrieving orientation entities and making a decision. A matching policy supplies the decision. Otherwise the allocated tier begins, and abstention passes the decision to the next tier. Figure 2 shows one cycle.

Note

Shapes follow the task metamodel of Vajda (2023): observations are shown as an ellipse, orientation as a rounded rectangle, the decision as a diamond, and the action as a rectangle. The shaded panel expands the decision step. The checks comprise the confidence threshold, the fields and token budget of the hand-off, and the rule that a machine tier may not make a normative decision without a policy. People are the final tier; they always decide and may store their decision as a policy.

A deterministic tool abstains when its state machine rejects the input because no transition is defined for the observed state. A model abstains if it declines, falls below the confidence threshold, violates its hand-off specification or budget, or attempts a normative decision without a policy. People may decide to defer; when escalation reaches them, the request explains why the lower tiers abstained and gives the options they identified along the configured path for that task.

Abstention allows a decision maker to reject a case where the risk of error is too high. Chow (1970) derived an optimal rule for classifiers with this option. Language-model cascades pass unreliable answers to larger models on a related principle (Chen et al. 2024), while here the task analysis specifies the escalation path and abstention conditions for each decision. Supervisors can therefore review the decisions that may reach them before a run.

When people make a normative decision, they may store it as a policy. The prototype matches a policy to the task design and its version, the declared orientation and intent, and the values of observations designated as policy features, and it validates the stored answer against the current hand-off specification before use. A different situation or intent returns the choice to people. Matching is exact, because a policy that generalized to similar situations would itself be exercising the normative judgment that the method reserves for people. The scope of each policy is therefore a design decision rather than an inference by a model. As equivalent situations recur, judgment requests should decline.

Escalation and policies are the two ways in which an allocation changes during operation. Escalation is a form of adaptive automation triggered by the state of the automation, and a policy is a form of adaptable automation in which a person delegates a recurring decision (Section 6.1).

Allocations are revised when circumstances change. When a model, a procedure, or the domain changes, analysts repeat Step 6, and the record of abstentions indicates which task models to revisit first.

3.9 Worked example: joining a tanker in synthetic training

We applied the method to a Royal Australian Air Force synthetic-training scenario in which White Force personnel facilitate exercises and play roles. A member of the Air Operations Simulation Centre supplied it for the earlier method's validation workshop (Vajda 2023). A simpilot, a White Force member who flies simulated aircraft, controls two aircraft. A trainee in the control agency instructs aircraft 2 to fly to the tanker and report at 20 nautical miles. We extended the earlier analysis's seven tasks with a decision on whether to deviate from the command, an option the scenario permits so White Force can create a problem for the trainee. The earlier analysis recorded deviation as a separate goal. We retained its scope, including the omission of the other aircraft's handling. Table 5 gives the allocation.

Table 5
Allocation of the task models in the tanker scenario

Task model	Decision	Clause	Tier	Action level
Fly to the tanker and respond on radio (parent) (FLY)	Algorithm	D	Deterministic tools	Reports to people
Acknowledge the command (ACK)	Plain language; functional, rule-based	F	Small models	Act and report
Decide whether to deviate from the command (DEV)	Normative	B	People	People act
Locate the tanker (LOC)	State machine	D	Deterministic tools	Act and report
Select aircraft 2 (SEL)	State machine	D	Deterministic tools	Act and report
Adjust heading to intercept the tanker (HDG)	Algorithm	D	Deterministic tools	Act unless vetoed
Configure the distance tool (CFG)	State machine	D	Deterministic tools	Act and report
Wait for the reporting distance (WAIT)	State machine	D	Deterministic tools	Act and report
Contact the control agency (CALL)	Plain language; functional, rule-based	F	Small models	Act unless vetoed
<i>Note.</i> The parent task model's decision is whether its goal has been met.				

We classified six of the nine decisions as state machines or algorithms, and these are allocated to deterministic tools. Two are plain-language decisions allocated to a small model, and one is normative and is allocated to people. The earlier analysis represented the observations of the first six tasks, including the radio, as finite state machines but specified no algorithms for their decisions. It left the task of contacting the control agency undescribed, because that task invokes a separate hierarchy for operating the radio (Vajda 2023).

We used scripted model responses to demonstrate the prototype in three variants of the scenario; Section 5 reports the separate model-performance study. In the clean variant, a person chose compliance and stored the deviation decision as a policy, so a second run at the same trainee level needed no further input from them. In the ambiguous variant, the call addressed “aircraft two or three”. The small model fell below its confidence threshold, and the reasoning model resolved the addressee because only aircraft 2 was under the simpilot's control.

In the scheduled-jamming variant, both scripted model tiers abstained. The reasoning-model surrogate identified the choice as one of training design and offered three options: hold the call, use the backup channel or relay through the other aircraft. The person selected the backup channel and stored the choice, which was reused on the next jammed run. The completion hand-off reported aircraft 2's position, the

transmission to control, goal status, the recorded training choice and the next expected event. These scripted runs demonstrate the escalation logic; the empirical study's scheduled-jamming limitation is reported in Section 5.3.

4 Implementation

We implemented the method in Python to execute and test allocations, abstention, policies and hand-offs, using a common interface that allows model providers or harnesses to be substituted (Table 6). OpenAI Codex and Anthropic Claude Code assisted with the runtime and study code under the author's direction and review. The code is released under the MIT license at <https://github.com/Rainierraul/task-allocation-runtime> (Vajda 2026) and archived at <https://doi.org/10.5281/zenodo.23225358>.

Table 6
Implementation of the method in the prototype

Part of the method	Prototype component	What it produces or checks
OODA metamodel and task models (Steps 1–3)	A typed schema, and one specification file of task models for each scenario	Rejects a set of task models in which an objective has no task model, a task model serves no objective, or a subtask does not name its parent
Attributes (Step 5)	Fields of each task model	Records the seven attributes; rejects a normative decision that names no policy features
Allocation rule (Step 6)	An allocation function	The tier, action level, clause, and reasons for each task model
Hand-offs (Step 7)	A hand-off validator	Checks the fields, their types, and the token budget of every hand-off
Running the task models	The engine, the implementation of each tier, and a policy store	Runs each OODA cycle, escalates on abstention, and applies and stores policies
Instances of task models	An event log	A record of every stage of every task model, and the measures taken from it
<i>Note.</i> Step 4 is an analyst-derived information-requirements analysis; its results enter the prototype as supervisor hand-off fields and decision requests. Appendix A traces them for the worked example.		

4.1 Task models as data

The metamodel is implemented as a schema that can be exported in JSON Schema format for use by other tools. Analysts write task models as structured text files, one per scenario, and the runtime rejects any file that violates the schema before execution. The schema enforces the rules of the task analysis: every objective must be served by a task model, every task model must serve an objective (Step 1), and every subtask must name its parent. A normative decision must also name the observations that key its policies, because otherwise a person's decision could not be matched to a later situation.

The allocation function applies the clauses of Table 4 in order and returns, for each task model, the tier, the action level, the deciding clause, and the reasons. A reviewer can therefore trace each allocation to the attributes that produced it. Table 5 is the function's output for the tanker scenario. Analysts can override an allocation, for example after a small model has failed trials on a task, but the override must record who made it and why, and this record is carried forward with the allocation.

4.2 Implementing the tiers

Each tier has a separate implementation. Deterministic tools are registered functions that implement a task model's state machine or algorithm. Small and reasoning models share a single component that assembles the model's context exclusively from the task model. The context comprises the task model's goal, decision rule, abstention conditions, declared observations, and orientation entities. It also includes the hand-offs of the task model's subtasks, the fields and budget of its own hand-off, and, where a lower tier has abstained, that tier's result. A fixed instruction of approximately 180 tokens requests a single structured reply stating whether the model decided, the hand-off, its confidence, and a one-sentence rationale. Where the model declines, the reply lists the options it identifies and the case for each.

Scripted surrogates return fixed responses for the automated tests and the worked example. For the computational study, an adapter calls any server offering the OpenAI-compatible chat interface, such as vLLM (Kwon et al. 2023), which serves open-weight models locally.

The final tier uses a console responder, or a scripted person in evaluation, with a judgment request that states the question, allocation clause and each lower tier's reason for abstaining. It presents the observations, subtask hand-offs, options and proposals, and says whether the answer may be stored as a policy. The person's answer must satisfy the same hand-off specification as a model's.

4.3 Enforcement at run time

The engine checks a decision against Section 3.8's abstention conditions before applying the action level from Step 6. A tool or person returning a hand-off that violates its specification indicates a task-design fault and halts the run. Rejected model output instead follows the escalation rules.

The policy store retains people's decisions, keyed by task model and by the values of the policy features, and persists between runs. A policy applies only when every feature matches.

4.4 The event log

Every stage of every task model is recorded as an event, including allocation, each model call, each abstention and escalation, each rejection of a normative decision by a machine tier, each request to people, and each hand-off. The log constitutes the instance level of the metamodel hierarchy and supplies the evaluation measures. These include the number of decisions and calls at each tier, abstentions, escalations, requests to people, approvals, notices, and policies stored and applied. They also include the tokens read and written at each tier, the largest context processed by any model, and separate size estimates for judgment requests, approvals, notices and final reports addressed to people.

In the worked example, the largest model input was about 396 tokens, including the fixed instruction. In the clean variant, the text addressed to the person (the question about deviation, two notices and the final report) came to about 300 tokens. In a repeated jammed run, with both policies applied, it came to about 103. These sizes are estimated from character counts.

Automated tests check that the runtime enforces these rules and that the study's scoring and recording behave as specified.

4.5 Limitations of the prototype

The prototype has three limitations relevant to the claims of this paper. First, a model's stated confidence may not track its accuracy, so the confidence threshold gives only coarse control over abstention; the evaluation therefore reports abstention against observed errors. Second, supervisors interact with the prototype only through a console. An interface through which they could monitor and direct a run needs its own design from the Step 4 requirements. Third, the runtime executes task models sequentially. Systems in which several task models decide concurrently, or in which the situation changes while a decision is pending, would need concurrency control and time limits that the prototype does not provide.

5 Evaluation

This section reports a computational comparison that addressed RQ1 and RQ2. The protocol was fixed before data collection and is given in full in Online Resource 1.

5.1 Method

The materials were 64 test instances and 32 separate held-out instances of the tanker scenario, generated from fixed random seeds. Half of the radio calls were addressed to aircraft 2. A quarter were ambiguous but resolvable, naming aircraft two or three when only aircraft 2 was under the simpilot's control. The remaining quarter were addressed to aircraft 3, so the call should not have been acknowledged and the join should not have proceeded. These cases were crossed with four channel states: clear, busy, unexpectedly jammed, and jammed by a scheduled exercise inject. The ordered reporting distance, the starting position and speed of aircraft 2, and the tanker's orbit were sampled from ranges chosen by the author rather than fitted to exercise records. Trainee level, deviation injects and training-plan intent varied independently. The deviation choice was recorded but did not change the flight trajectory. Online Resource 1 gives the seeds, sampling ranges and how the materials were generated.

The four conditions separated whole-task prompting, decomposition, model routing and the complete allocation mechanism. Each condition ran five times on each test instance, in a randomized order, with runs proceeding in parallel on one cluster node. Every run started with an empty policy store; policy accumulation was evaluated separately.

1. Single model: the reasoning model received the scenario facts, public training intent, motion equations and task specifications in one request, and was asked to answer every task, including DEV and the

response to scheduled jamming. The private scripted plan and generator's answer key were withheld.

2. Decomposed single tier: the reasoning model made all eight machine decisions in plan order, each from only the inputs declared for that task. It was required to give an answer and could not abstain. If a reply could not be read as a structured answer, the run failed and its remaining required decisions counted as incorrect; the scripted person did not rescue them. DEV came from the scripted plan.

3. Routed: the six formally specified decisions were assigned to tools. For ACK and CALL, a RouteLLM BERT classifier (Ong et al. 2025; checkpoint routellm/bert_gpt4_augmented, revision 86237e3df400762178ea98379477b8296e66d5e4), run locally, chose between the small and reasoning models. Its threshold was set on held-out cases so that half of their calls went to the reasoning model. The chosen model had to answer and could not abstain. The classifier was trained for a different pair of models, which limits how well it transfers. DEV came from the scripted plan.

4. Allocated: tools were assigned the six formally specified decisions, while ACK and CALL began with the small model. Confidence thresholds, output checks, abstention, escalation and scoped policies operated, and the scripted person authorized DEV. Scheduled jamming also involves a training choice, but the frozen specification marked CALL as epistemic in every case, so that response reached the person through model abstention alone (Section 5.2). Each condition received the reporting distance and motion parameters needed for its decisions.

The models ran on one node of Deakin University's Applied Artificial Intelligence Institute computing cluster, in their published 16-bit floating-point (bf16) form. The small model was Qwen3-4B-Instruct-2507 at revision cdbee75f17c01a7cc42f958dc650907174af0554, and the reasoning model was Qwen3.8-27B at revision 1d4bf0f2ff6012fd82039f2fa52739d0dd7c60c0. vLLM 0.30.0 served both (Kwon et al. 2023), with the reasoning model split across two NVIDIA L40S graphics processors and the small model on a third. The reasoning model used thinking at the medium reasoning-effort setting. Sampling followed the models' published defaults, with output limits of 4,096 tokens for the small model and 16,384 for the reasoning model, including its reasoning. Every condition used the same pair of models, and the server software, model files and processor assignment were recorded for each run. A scripted person supplied DEV in conditions 2 to 4, and supplied reference answers in the routed and allocated conditions when every eligible machine tier failed to decide.

RQ1 measured accuracy and resource use. Accuracy was the proportion of required epistemic decisions correct within each instance. Resource use counted the input and output tokens reported for returned responses, including reasoning tokens. A call addressed to aircraft 3 required only ACK and FLY, including under scheduled jamming. For an acknowledged join instruction under scheduled jamming, CALL and FLY were excluded because they depended on a training choice. DEV was never scored, and failed runs stayed in the denominator. Scoring compared named output fields with reference values and fixed tolerances, without a model judge. Each model's input stated the scoring requirements: HDG had to be within 1 degree and 0.2 minutes of an independently computed intercept, for example, with all tolerances given in Online Resource 1. Secondary measures were the largest model input, time per run and requests to people, with

timing describing shared load on the cluster node; monetary cost was unmeasured. An early error could affect later decisions.

For RQ2, both models answered every eligible ACK and CALL context from a clean reference trajectory five times, regardless of allocation. Each reply provided a best candidate and a decision-or-abstention choice, allowing both to be scored from one response. Calibration used held-out cases to choose the lowest threshold accepting at least ten candidates with no more than 5% wrong; if no threshold qualified, that tier was disabled for the task. Thresholds and router settings were fixed before test calls; the held-out error target did not guarantee the test error rate.

The diagnostic measured withholding for wrong and correct candidates separately, counting refusals, failed confidence or output checks, and confidence below the held-out threshold as reasons to withhold a candidate. Asking for a best candidate may affect willingness to abstain. The diagnostic covered two clause-F tasks, leaving clause G and other decisions untested.

Policy accumulation used the 64 cases in five orders, each with an empty initial policy store, allowing judgment requests in the first and last ten cases to be compared within each order. Exact matching of continuous distances was expected to limit CALL reuse, with any decline mainly from recurring DEV situations. Repetitions were averaged within instances, and 95% percentile intervals used 10,000 bootstrap resamples with paired conditions. The policy orders reused the same cases, so we report their means and ranges without confidence intervals. Online Resource 1 gives the seeds. The study was exploratory, with no hypothesis tests.

5.2 Results

Collection yielded 1,280 condition runs, 1,000 diagnostic replies and 320 policy-sequence runs, with all 6,363 model requests returned, including calibration and diagnostics; there were no failed or repeated requests, and run timing was complete. Fifty-two replies reached the output limit, 48 in the single-model condition. Runs ending with an unusable candidate or action remained in the analysis, with unfinished required decisions scored incorrect: 156 in the single-model condition, 25 in the decomposed, 31 in the routed and 32 in the allocated condition. Of the single-model failures, 87 involved only malformed DEV and 60 involved extra fields. All scored decisions were correct in 78 of its 156 failed runs. Online Resource 2 gives per-task scores and failure counts.

The allocated condition was correct on 89.6% of required epistemic decisions, against 67.2% for the single-model condition and 80.0% for the routed condition (Table 7). Its paired advantage was 22.4 percentage points over the single-model condition (95% CI 15.6 to 29.0) and 9.6 points over the routed condition (3.2 to 15.7). The decomposed condition was correct on 74.7%, 7.5 points above the single-model condition, with an interval that includes zero (− 0.3 to 15.3).

Decomposition cut the largest model input from 4,763 to 626 tokens, but the reasoning model still wrote about 10,800 tokens per run, and a run took about 540 s in both the single-model and decomposed conditions. The routed condition read 1,008 and wrote 862 tokens per run. In that condition, six of the eight

machine decisions were assigned to tools, and the router chose the reasoning model for 270 of its 537 decisions. The allocated condition read 3,208 fewer tokens and wrote 9,413 fewer per run than the single-model condition, its largest model input averaged 576 tokens, and a run took 59 s against 542 s. Against the routed condition, it read 546 and wrote 438 more tokens per run and took 18 s longer. In the allocated condition every ACK and CALL decision began with the small model, and abstention passed many of them to the reasoning model.

In the allocated condition, tools made every location, heading, configuration and waiting decision, and all 240 of each were correct. The reasoning model, making the same decisions in the decomposed condition, was correct on 77% to 91% of the first three but on only 20 of 240 WAIT decisions. In 137 of the incorrect WAIT decisions it reported the current distance as not yet at range, where the decision rule asked it to predict when the aircraft would reach the reporting distance; in 39 it gave no usable answer. Given the whole task and the equations of motion, the single-model condition was correct on 173 of 240 WAIT decisions.

The scripted person supplied 256 of the 1,960 scored decisions in the allocated condition (13%): 103 ACK, 121 CALL and 32 SEL decisions. ACK and CALL reached it after both models had abstained or produced unusable output. In the routed condition it supplied 96 SEL decisions. Its answers were correct by construction, so accuracy in both conditions describes each system together with its simulated supervisor. The person received 1.86 requests per run on average in the allocated condition, 1.17 in the routed condition and 0.68 in the decomposed condition, and none in the single-model condition.

The selection tool requires an exact entity name, so it rejected “Aircraft 2” from ACK, a difference of letter case that the scoring accepts. The scripted person then supplied 26 of these selections in the allocated condition and 36 in the routed condition, and the reasoning model made the other 3 and 11. The remaining person-supplied selections followed an unresolved or unnormalized addressee such as “aircraft two or three”.

Held-out calibration set the acceptance threshold to zero for ACK at both tiers and for CALL at the reasoning model, so in those cases a candidate was accepted unless the model abstained or its output failed the checks. For CALL at the small model, no threshold met the 5% error target, so the small model never decided CALL on its own and passed every CALL decision to the reasoning model.

Scheduled-jamming responses exposed an authority limitation. CALL was marked epistemic, so the runtime handled the training choice as an ordinary CALL decision. Of the 60 responses made in the allocated condition, the scripted person supplied 35 after model abstention; the reasoning model supplied 25 without a request to the person. In the routed condition, the small model supplied all 56. These CALL responses were excluded from accuracy scoring. Online Resource 2 gives the counts by condition.

Table 7
Accuracy and resource use by condition

Condition	Accuracy of epistemic decisions (%)	Tokens read	Tokens written	Largest context (tokens)	Time per run (s)
Single model	67.2 (62.2–72.4)	4,763 (4,761–4,765)	10,713 (9,390–11,939)	4,763 (4,761–4,765)	542 (474–603)
Decomposed, single tier	74.7 (69.9–79.4)	3,045 (2,718–3,346)	10,761 (9,224–12,212)	626 (603–647)	540 (462–613)
Routed	80.0 (72.1–87.3)	1,008 (941–1,075)	862 (755–974)	525 (517–533)	40 (35–46)
Allocated	89.6 (82.8–95.5)	1,555 (1,447–1,659)	1,300 (1,152–1,458)	576 (559–592)	59 (52–66)
<p><i>Note.</i> Values are instance means across five repetitions, with 95% paired bootstrap intervals from 10,000 resamples of instances. Tokens are per run. Accuracy in the routed and allocated conditions includes decisions supplied by the scripted person (Section 5.2). Per-tier token use and failed-run counts are in Online Resource 2.</p>					

Table 8 reports candidates withheld at each tier. The small-model tier withheld 112 of 115 wrong candidates (97.4%) and 245 of 385 correct candidates (63.6%). The reasoning-model tier withheld 90 of 97 wrong candidates (92.8%) and 27 of 403 correct candidates (6.7%). Its withheld wrong candidates comprised 85 explicit refusals and five malformed replies missing the decision flag; all 27 withheld correct candidates were explicit refusals. The small model explicitly declined 102 wrong and 75 correct candidates. Its disabled CALL threshold withheld the remaining 10 wrong and 170 correct candidates. The allocation rules pass a withheld decision to the next tier.

Table 8
Tier abstention in the paired diagnostics

Tier	Decisions	Incorrect decisions	Withheld when incorrect (%)	Withheld when correct (%)
Small model	500	115	97.4 (90.9–100.0)	63.6 (56.5–71.5)
Reasoning model	500	97	92.8 (85.7–97.5)	6.7 (3.6–10.2)

Note. Each tier answered 500 diagnostic requests, covering every eligible ACK and CALL context five times. A candidate counts as withheld when the model declined, its confidence or output failed the checks, or its confidence fell below the tier’s held-out threshold. Intervals are 95% bootstrap intervals over instances.

Across the five policy orders, judgment requests fell from a mean of 1.54 per case over the first ten cases to 0.90 over the last ten. Requests fell in four orders, by 0.5 to 1.1 per case, and rose by 0.1 in the fifth. The orders reuse the same cases, so these figures describe sensitivity to order rather than independent replications.

5.3 Threats to validity

The scripted person supplied error-free choices and reference recovery. Real supervisors may err, delay or refuse, so these results measure the systems with simulated supervision. Most single-model failures involved the format of unscored or extra fields. The materials comprise one synthetic scenario, nine task models and author-chosen frequencies. Deviation choices were recorded without changing the trajectory. Task models, tolerances and eligibility rules depend on the author's analysis.

Three implementation findings limit interpretation. Exact name matching caused case-only SEL rejections, recovered by the reasoning model or scripted person. Reference recovery occurred in routing as well as allocation, although the protocol specified it only for allocation; CALL's fixed epistemic classification also allowed models to make a scheduled-jamming training choice. The study therefore fails to demonstrate the normative boundary for that decision. We report the collected behavior without repairing or rescoring it.

The study used one open-weight model pair and a router trained for another; its held-out scores were unsaved, preventing reconstruction of their calibration, although the threshold and test choices remain recorded. Eliciting a best candidate may affect abstention. Exact policy matching limits reuse, and completion time depends on shared cluster load. These findings require testing in other tasks and with other model pairs.

6 Discussion

The allocated condition used far fewer tokens and a much smaller largest model input than the single-model condition, and it was more accurate. The routed condition, which shared the tool assignments, also

showed large resource savings. Because the conditions compared combinations, they cannot isolate what tools, routing, abstention or reference answers each contributed. The scripted person, who supplied 13% of the allocated condition's scored decisions, was also part of the system evaluated. Tier abstention withheld most wrong reasoning-model candidates and few correct ones, but it also withheld most correct small-model candidates, 170 of them through the disabled CALL threshold. Scheduled jamming exposed a separate limit: the training choice reached people only when the models abstained, because the prototype did not enforce the normative boundary for that decision. We relate these findings to levels of automation and adaptive automation, then set out the limitations.

6.1 Levels of automation and adaptive automation

Stages and levels remain useful design dimensions, although their role is contested (Endsley 2018; Jamieson and Skraaning 2018; Kaber 2018; Wickens 2018). We retain levels for action authority and add tiers for choosing who makes a decision. An allocation can specify a sequence governed by abstention, adding the choice of decision maker alongside stage and level to broaden the design space advocated by Jamieson and Skraaning (2018).

Escalation and policies support the two forms of dynamic allocation. Escalation changes allocation automatically when a tier's confidence or output checks fail. Policies let people delegate recurring normative decisions. Adaptive automation often responds to the operator, task or environment (Feigh et al. 2012); this method uses the automation's state as its trigger. That trigger indicates whether a machine decision can be accepted. It leaves the supervisor's capacity to take on the work unmeasured. Policies follow Miller and Parasuraman's (2007) account of delegation, but start from an analyst-derived allocation and cover normative decisions one situation at a time.

Our method also changes the criteria for allocation. Parasuraman et al. (2000) made the consequences for human performance the primary criteria for choosing a level of automation, with automation reliability and the costs of decision and action outcomes as secondary criteria. We keep these criteria in the action levels and confidence thresholds. We add two properties of the decision that come before them: how formally it can be specified, and whether it establishes what is the case or determines what ought to be done. The first, adapted from Vajda (2023), settles whether a decision can be made without a language model. The second settles whether any machine should make it without a person. Cummings (2018) argued that skill-, rule- and knowledge-based behavior should inform decisions about automation; the allocation rule turns that argument into a procedure that analysts can apply to each task model and others can audit.

The method's hand-offs and judgment requests also respond to the call to design for interdependence (Johnson et al. 2014, 2018). Supervisors can observe what each tier decided, what it declined to decide and why, and what it proposed, which supports observability. The allocation table indicates in advance which decisions will reach them, which supports predictability, and policies, vetoes, and approvals allow them to redirect the work, which supports directability.

We depart from guidance that automation should support routine tasks while leaving higher-level cognitive functions to people (Endsley et al. 2003; Endsley 2017). Language models already perform some of these functions, and supervisors cannot review every response. We therefore keep normative authority with people and fix abstention conditions at design time, requiring complex-domain models to propose options and supervisor hand-offs to follow Step 4. The scheduled-jamming finding shows why implementation of those safeguards must be checked. Their effect on situation awareness requires evaluation with practitioners. If such a study found reduced awareness, the proposed delegation would need to be narrowed.

6.2 Limitations

The main limitation of this paper is that its evaluation did not involve supervisors. The computational comparison concerns the machine tiers, whereas the method's design objectives concern the people who direct them. Whether the hand-offs preserve supervisors' situation awareness, reduce their workload and help them detect faults, relative to a conversational interface, requires an experiment with practitioners. Such an experiment could measure situation awareness with the Situation Awareness Global Assessment Technique (Endsley 1988), using queries derived from the Step 4 requirements, and workload with the NASA Task Load Index (Hart and Staveland 1988). Faults seeded into the machine output would allow detection to be measured. The experiment would also need an interface for supervisors designed from the Step 4 requirements, which the prototype does not yet provide.

Allocation leaves people with normative choices and decisions every machine tier withheld, which may be the most unusual and difficult. This is Bainbridge's (1983) irony of automation. Policies remove recurring decisions from view and further reduce opportunities to practice the judgments needed in unfamiliar cases. Fused hand-offs report activity but provide limited practice. Returning some routine decisions to people could address this problem. Skill loss and the effect of such a design would require a longitudinal study.

Task analysis also takes time. In the earlier method's validation, participants said the framework “appeared to further complicate an already difficult process” and requested estimates of its time and cost (Vajda 2023, p 177). This extension adds three steps, extends a fourth and requires specifications software can check, making the cost easier to justify for recurring tasks where one analysis serves many runs. Direct model interaction may be more efficient for a task performed once. Models could draft task descriptions from documents and interviews, but analysts would still need to check them.

The boundaries between tiers can shift as models improve or regress. Clauses F and G encode expectations about narrow rule-based decisions and more demanding plain-language decisions rather than a demonstrated cost optimum. The diagnostics examine ACK and CALL, both clause-F decisions, with one pair of open-weight models, so they cannot validate clause G or show that the assigned tier is the cheapest. Calibration and abstention records can identify allocations that need revision.

Epistemic and normative choices can occur within one decision. Judging whether evidence is sufficient combines an assessment of the evidence with a choice of the required standard. Analysts can decompose

these parts, leaving people to set the threshold and machines to compare evidence against it. Different teams may draw that boundary differently. Scheduled jamming exposed a related problem: CALL's decision kind depended on the situation, but its task model recorded a fixed epistemic kind; enforcement requires that dependency to be represented or the task decomposed further.

The allocation responds to machine state but leaves supervisor workload outside the mechanism. Many simultaneous escalations could exceed a person's capacity. An operator-state trigger could defer or batch requests, as studied in adaptive automation (Feigh et al. 2012; Kaber and Endsley 2004). It could also return routine decisions when workload permits, providing practice. The practitioner study would need to assess both effects.

Exact policy matching limits reuse where situations rarely recur, so people may continue to receive nearly identical normative choices. Letting people specify a policy's scope when they set it is a possible extension, although choosing a scope is itself a normative act.

Two follow-up studies would compare extensions with this baseline. A bounded agent loop could consult declared read-only sources before returning a decision, while the runtime retains control of actions. A separate calibrated decision model could judge whether a candidate is likely to be correct, replacing the deciding model's stated confidence as the abstention gate.

7 Conclusion

We extend DCARR task analysis to allocate decisions among tools, small models, reasoning models and people, with task attributes determining the initial tier, abstention passing decisions upward and scoped policies reusing people's normative choices. Hand-offs specify the information each receiving task or supervisor needs. On the synthetic tanker cases, the allocated condition was more accurate than one reasoning model given the whole task, and used fewer tokens and a smaller largest input. A scripted supervisor supplied 13% of the scored decisions, and the prototype failed to keep a scheduled-jamming training choice with people. These results support the computational design while leaving supervisors' situation awareness and workload to be evaluated with practitioners.

Declarations

Funding:

No funding was received for this study. Competing interests: The author declares no competing interests. Author contributions: Simon Vajda is the sole author and is responsible for the scientific content and final submission.

Ethics and consent: The computational study uses synthetic scenarios and a scripted person. It involves no human participants and no operational exercise data. Ethics approval and participant consent are not applicable.

Related manuscript: The framework and worked scenario derive from the author's doctoral thesis (Vajda 2023). A separate article reporting the earlier automation-candidate framework and its formative design review, DCARR: Selecting Automation Candidates for White Force Capability Multiplication, is under review at The Journal of Defense Modeling and Simulation. The present paper reports the language-model allocation method and a new computational comparison. A copy of the related manuscript can be supplied to the editors on request.

Use of AI tools: OpenAI Codex and Anthropic Claude Code were used to write and test the runtime and study code, to prepare the computing environment, and to help draft and edit the manuscript. The author directed this work, reviewed its output, and takes responsibility for the content.

Data availability: Online Resource 1 contains the study protocol, including its seeds and how the materials were generated. Online Resource 2 contains per-task scores and descriptive counts. The synthetic materials, task specifications, prompts, calibration, model-server records, analysis code and outputs are archived at <https://doi.org/10.5281/zenodo.23225358>. Code is available at <https://github.com/Rainierraul/task-allocation-runtime>, and model weights at the pinned revisions in Section 5.1. The study collected no human participant or operational exercise data.

Author Contribution

S.V. is the sole author and is responsible for the scientific content and final submission.

Data Availability

Online Resource 1 contains the study protocol, including its seeds and how the materials were generated. Online Resource 2 contains per-task scores and descriptive counts. The synthetic materials, task specifications, prompts, calibration, model-server records, analysis code and outputs are archived at <https://doi.org/10.5281/zenodo.23225358>. Code is available at <https://github.com/Rainierraul/task-allocation-runtime>, and model weights at the pinned revisions in Section 5.1. The study collected no human participant or operational exercise data.

References

1. Annett J (2003) Hierarchical task analysis. In: Hollnagel E (ed) Handbook of cognitive task design. CRC, Boca Raton, pp 17–35. <https://doi.org/10.1201/9781410607775.ch2>
2. 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)
3. Blasch EP, Plano S (2002) JDL level 5 fusion model: user refinement issues and applications in group tracking. *Proc SPIE* 4729:270–279. <https://doi.org/10.1117/12.477612>
4. Boyd JR (2018) In: Hammond GT (ed) A discourse on winning and losing. Air University.. <https://www.airuniversity.af.edu/AUPress/Display/Article/1528758/a-discourse-on-winning-and->

losing/ Maxwell Air Force Base, AL

5. Breton R (2008) The modelling of three levels of cognitive controls with the Cognitive-OODA loop framework. Technical report DRDC Valcartier TR 2008 – 111, Defence R&D Canada – Valcartier
6. Buçinca Z, Malaya MB, Gajos KZ (2021) To trust or to think: cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making. *Proc ACM Hum-Comput Interact* 5(CSCW1):188. <https://doi.org/10.1145/3449287>
7. Cemri M, Pan MZ, Yang S, Agrawal LA, Chopra B, Tiwari R, Keutzer K, Parameswaran A, Klein D, Ramchandran K, Zaharia M, Gonzalez JE, Stoica I (2025) Why do multi-agent LLM systems fail? In: *Advances in Neural Information Processing Systems* 38 (Datasets and Benchmarks Track). https://proceedings.neurips.cc/paper_files/paper/2025/hash/b1041e52d3be19f0a9bc491657488e4a-Abstract-Datasets_and_Benchmarks_Track.html
8. Chen L, Zaharia M, Zou J (2024) FrugalGPT: how to use large language models while reducing cost and improving performance. *Trans Mach Learn Res*. <https://openreview.net/forum?id=cSimKw5p6R>
9. Chow CK (1970) On optimum recognition error and reject tradeoff. *IEEE Trans Inf Theory* 16(1):41–46. <https://doi.org/10.1109/TIT.1970.1054406>
10. Cummings M (2018) Informing autonomous system design through the lens of skill-, rule-, and knowledge-based behaviors. *J Cogn Eng Decis Mak* 12(1):58–61. <https://doi.org/10.1177/1555343417736461>
11. de Winter JCF, Dodou D (2014) Why the Fitts list has persisted throughout the history of function allocation. *Cogn Technol Work* 16(1):1–11. <https://doi.org/10.1007/s10111-011-0188-1>
12. Endsley MR (1988) Situation awareness global assessment technique (SAGAT). In: *Proceedings of the IEEE 1988 National Aerospace and Electronics Conference*. IEEE, pp 789–795. <https://doi.org/10.1109/NAECON.1988.195097>
13. Endsley MR (1995) Toward a theory of situation awareness in dynamic systems. *Hum Factors* 37(1):32–64. <https://doi.org/10.1518/001872095779049543>
14. Endsley MR (2017) From here to autonomy: lessons learned from human–automation research. *Hum Factors* 59(1):5–27. <https://doi.org/10.1177/0018720816681350>
15. Endsley MR (2018) Level of automation forms a key aspect of autonomy design. *J Cogn Eng Decis Mak* 12(1):29–34. <https://doi.org/10.1177/1555343417723432>
16. Endsley MR (2023a) Ironies of artificial intelligence. *Ergonomics* 66(11):1656–1668. <https://doi.org/10.1080/00140139.2023.2243404>
17. Endsley MR (2023b) Supporting human–AI teams: transparency, explainability, and situation awareness. *Comput Hum Behav* 140:107574. <https://doi.org/10.1016/j.chb.2022.107574>
18. Endsley MR, Kiris EO (1995) The out-of-the-loop performance problem and level of control in automation. *Hum Factors* 37(2):381–394. <https://doi.org/10.1518/001872095779064555>
19. Endsley MR, Bolté B, Jones DG (2003) Designing for situation awareness: an approach to user-centered design. CRC, Boca Raton. <https://doi.org/10.1201/9780203485088>

20. Evans JSBT, Stanovich KE (2013) Dual-process theories of higher cognition: advancing the debate. *Perspect Psychol Sci* 8(3):223–241. <https://doi.org/10.1177/1745691612460685>
21. Feigh KM, Dorneich MC, Hayes CC (2012) Toward a characterization of adaptive systems: a framework for researchers and system designers. *Hum Factors* 54(6):1008–1024. <https://doi.org/10.1177/0018720812443983>
22. Fitts PM (ed) (1951) Human engineering for an effective air-navigation and traffic-control system. National Research Council, Washington, DC
23. Hadfield J, Zhang B, Lien K, Scholz F, Fox J, Ford D (2025) How we built our multi-agent research system. *Anthropic*. <https://www.anthropic.com/engineering/built-multi-agent-research-system>. Accessed 7 October 2026
24. Hart SG, Staveland LE (1988) Development of NASA-TLX (Task Load Index): results of empirical and theoretical research. In: Hancock PA, Meshkati N (eds) Human mental workload. *Advances in psychology*, vol 52. North-Holland, Amsterdam, pp 139–183. [https://doi.org/10.1016/S0166-4115\(08\)62386-9](https://doi.org/10.1016/S0166-4115(08)62386-9)
25. Hong S, Zhuge M, Chen J, Zheng X, Cheng Y, Wang J, Zhang C, Wang Z, Yau S, Lin Z, Zhou L, Ran C, Xiao L, Wu C, Schmidhuber J (2024) MetaGPT: meta programming for a multi-agent collaborative framework. In: The Twelfth International Conference on Learning Representations, pp 23247–23275. https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html
26. Jamieson GA, Skraaning G (2018) Levels of automation in human factors models for automation design: why we might consider throwing the baby out with the bathwater. *J Cogn Eng Decis Mak* 12(1):42–49. <https://doi.org/10.1177/1555343417732856>
27. Johnson M, Bradshaw JM, Feltovich PJ, Jonker CM, van Riemsdijk MB, Sierhuis M (2014) Coactive design: designing support for interdependence in joint activity. *J Hum-Robot Interact* 3(1):43–69. <https://doi.org/10.5898/JHRI.3.1.Johnson>
28. Johnson M, Bradshaw JM, Feltovich PJ (2018) Tomorrow's human–machine design tools: from levels of automation to interdependencies. *J Cogn Eng Decis Mak* 12(1):77–82. <https://doi.org/10.1177/1555343417736462>
29. Kaber DB (2018) Issues in human–automation interaction modeling: presumptive aspects of frameworks of types and levels of automation. *J Cogn Eng Decis Mak* 12(1):7–24. <https://doi.org/10.1177/1555343417737203>
30. Kaber DB, Endsley MR (2004) The effects of level of automation and adaptive automation on human performance, situation awareness and workload in a dynamic control task. *Theor Issues Ergon Sci* 5(2):113–153. <https://doi.org/10.1080/1463922021000054335>
31. Kahneman D (2011) Thinking, fast and slow. Farrar, Straus and Giroux, New York
32. Kambhampati S, Valmeekam K, Guan L, Verma M, Stechly K, Bhambri S, Saldyt LP, Murthy AB (2024) Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In: *Proceedings of the 41st International Conference on Machine Learning*. PMLR 235, pp 22895–22907. <https://proceedings.mlr.press/v235/kambhampati24a.html>

33. Kim Y, Jeong H, Park C, Park E, Zhang H, Liu X, Lee H, McDuff D, Ghassemi M, Breazeal C, Tulebaev S, Park HW (2025) Tiered agentic oversight: a hierarchical multi-agent system for healthcare safety. Preprint at. <https://doi.org/10.48550/arXiv.2506.12482>
34. Klein G, Woods DD, Bradshaw JM, Hoffman RR, Feltovich PJ (2004) Ten challenges for making automation a team player in joint human-agent activity. *IEEE Intell Syst* 19(6):91–95. <https://doi.org/10.1109/MIS.2004.74>
35. Kurtz CF, Snowden DJ (2003) The new dynamics of strategy: sense-making in a complex and complicated world. *IBM Syst J* 42(3):462–483. <https://doi.org/10.1147/sj.423.0462>
36. Kwon W, Li Z, Zhuang S, Sheng Y, Zheng L, Yu CH, Gonzalez JE, Zhang H, Stoica I (2023) Efficient memory management for large language model serving with PagedAttention. In: *Proceedings of the 29th Symposium on Operating Systems Principles*. ACM, pp 611–626. <https://doi.org/10.1145/3600006.3613165>
37. Lee JD, See KA (2004) Trust in automation: designing for appropriate reliance. *Hum Factors* 46(1):50–80. https://doi.org/10.1518/hfes.46.1.50_30392
38. Levy M, Jacoby A, Goldberg Y (2024) Same task, more tokens: the impact of input length on the reasoning performance of large language models. In: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp 15339–15353. <https://doi.org/10.18653/v1/2024.acl-long.818>
39. Liu NF, Lin K, Hewitt J, Paranjape A, Bevilacqua M, Petroni F, Liang P (2024) Lost in the middle: how language models use long contexts. *Trans Assoc Comput Linguist* 12:157–173. https://doi.org/10.1162/tacl_a_00638
40. Melnikoff DE, Bargh JA (2018) The mythical number two. *Trends Cogn Sci* 22(4):280–293. <https://doi.org/10.1016/j.tics.2018.02.001>
41. Miller CA, Parasuraman R (2007) Designing for flexible interaction between humans and automation: delegation interfaces for supervisory control. *Hum Factors* 49(1):57–75. <https://doi.org/10.1518/001872007779598037>
42. Ong I, Almahairi A, Wu V, Chiang W-L, Wu T, Gonzalez JE, Kadous MW, Stoica I (2025) RouteLLM: learning to route LLMs from preference data. In: *The Thirteenth International Conference on Learning Representations*, pp 34433–34448. <https://openreview.net/forum?id=8sSqNntaMr>
43. Onnasch L, Wickens CD, Li H, Manzey D (2014) Human performance consequences of stages and levels of automation: an integrated meta-analysis. *Hum Factors* 56(3):476–488. <https://doi.org/10.1177/0018720813501549>
44. Parasuraman R, Manzey DH (2010) Complacency and bias in human use of automation: an attentional integration. *Hum Factors* 52(3):381–410. <https://doi.org/10.1177/0018720810376055>
45. Parasuraman R, Riley V (1997) Humans and automation: use, misuse, disuse, abuse. *Hum Factors* 39(2):230–253. <https://doi.org/10.1518/001872097778543886>
46. Parasuraman R, Sheridan TB, Wickens CD (2000) A model for types and levels of human interaction with automation. *IEEE Trans Syst Man Cybern Syst Hum* 30(3):286–297. <https://doi.org/10.1109/3468.844354>

47. Rasmussen J (1983) Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Trans Syst Man Cybern SMC* –13(3):257–266.
<https://doi.org/10.1109/TSMC.1983.6313160>
48. Shaw SD, Nave G (2026) Thinking—fast, slow, and artificial: how AI is reshaping human reasoning and the rise of cognitive surrender. Preprint at. <https://doi.org/10.2139/ssrn.6097646>
49. Sheridan TB, Verplank WL (1978) Human and computer control of undersea teleoperators. Technical report, Man-Machine Systems Laboratory, Massachusetts Institute of Technology, Cambridge, MA
50. Simkute A, Tankelevitch L, Kewenig V, Scott AE, Sellen A, Rintel S (2025) Ironies of generative AI: understanding and mitigating productivity loss in human-AI interaction. *Int J Hum-Comput Interact* 41(5):2898–2919. <https://doi.org/10.1080/10447318.2024.2405782>
51. Snowden DJ, Boone ME (2007) A leader's framework for decision making. *Harv Bus Rev* 85(11):68–76
52. Stanton NA (2006) Hierarchical task analysis: developments, applications, and extensions. *Appl Ergon* 37(1):55–79. <https://doi.org/10.1016/j.apergo.2005.06.003>
53. Steinberg AN, Bowman CL, White FE (1999) Revisions to the JDL data fusion model. *Proc SPIE* 3719:430–441. <https://doi.org/10.1117/12.341367>
54. Vajda SRL (2023) Model driven White Force automation. Dissertation, Deakin University.
<https://hdl.handle.net/10779/DRO/DU:24769413>
55. Vajda SRL (2026) task-allocation-runtime (software). Zenodo.
<https://doi.org/10.5281/zenodo.23225358>
56. Wickens C (2018) Automation stages & levels, 20 years after. *J Cogn Eng Decis Mak* 12(1):35–41.
<https://doi.org/10.1177/1555343417727438>
57. Woods DD, Hollnagel E (2006) Joint cognitive systems: patterns in cognitive systems engineering. CRC, Boca Raton. <https://doi.org/10.1201/9781420005684>
58. Woods DD, Patterson ES, Roth EM (2002) Can we ever escape from data overload? A cognitive systems diagnosis. *Cogn Technol Work* 4(1):22–36. <https://doi.org/10.1007/s101110200002>
59. Zhou J (2025) OrchVis: hierarchical multi-agent orchestration for human oversight. Preprint at. <https://doi.org/10.48550/arXiv.2510.24937>

Figures

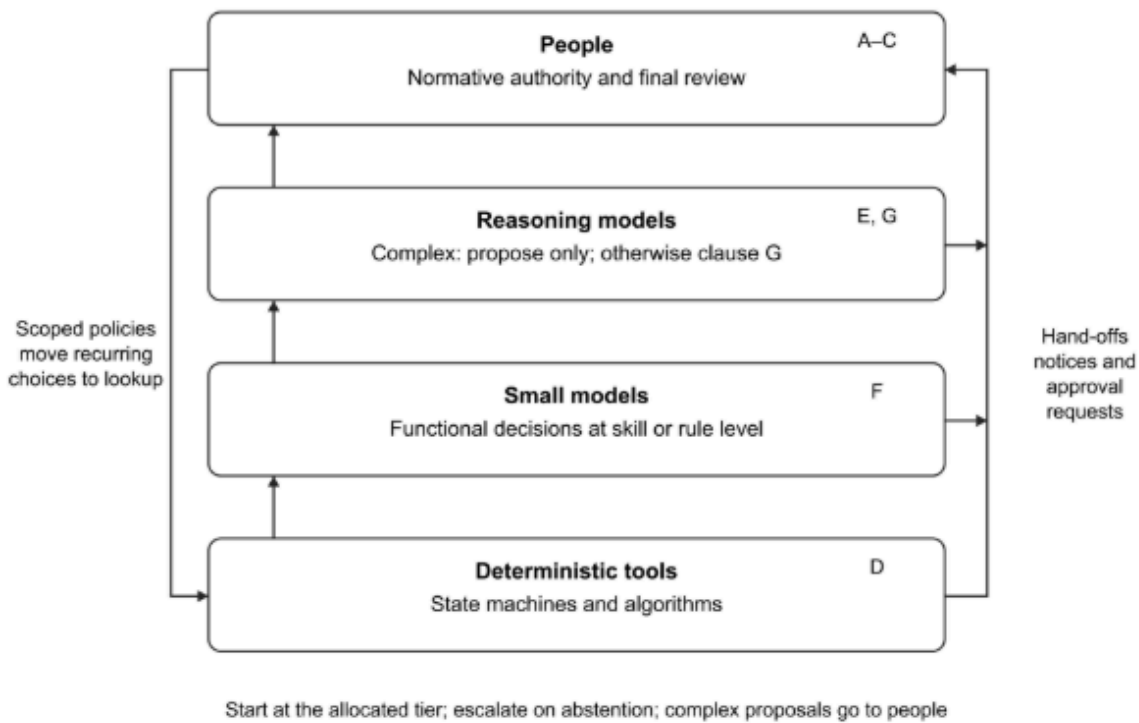


Figure 1

The four tiers and how decisions move between them

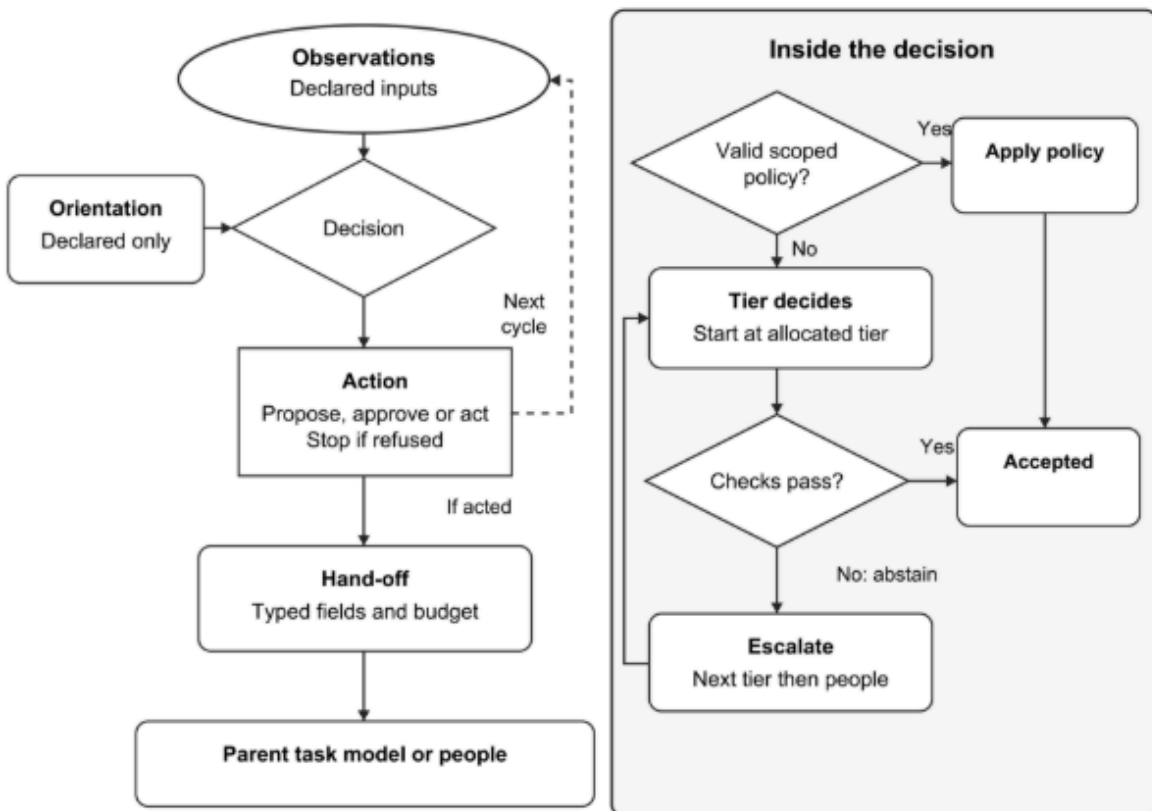


Figure 2

One task model at run time

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [ESM1.pdf](#)
- [ESM2.pdf](#)