WARM: the Working Agents Reference Model

The parts of a system in which agents do sustained work. Derived from a working implementation, three agent platforms and fourteen published models, and mapped to ten vendor products

DAVE REMY · WORKING PAPER · VERSION 0.1.1 (MODEL VERSION 0.14) · 17 NOVEMBER 2026 · CC BY 4.0 ·
DOI:10.5281/ZENODO.24238372

Changes in 0.1.1 (17 November 2026). The subtitle and abstract now lead with what was mapped and what was found, and present the review of stops for a person as one proposal of the model. The conclusion and Section 7 now say that the refactoring of the reference implementation has begun and that its measures have not yet been set. No count, rating or table changed.

Version 0.1 is the tenth draft, with its text of 13 November 2026 unchanged. Drafts 1, 3, 6 and 8 were reviewed. Draft 9 applied the final review of draft 8 once, and draft 10 applied the author's answers to eight questions on draft 9. Neither set of changes has been reviewed.

ABSTRACT

The system of AI agents behind this paper was built one addition at a time and had no shared description. WARM, the Working Agents Reference Model, names the parts of such systems and how they relate. People direct the work, domains define it, agents carry it out, and a platform of 25 services in six groups provides what agents in every domain share. The model was drawn first from that system and took its vocabulary from a survey of 34 sources. It was then mapped to three agent platforms and to its source system and compared with fourteen published models, and these exercises corrected it in more than twenty places. Every platform service has at least a partial counterpart in nine or more of the seventeen platforms and models. A rating under written criteria found that two of the seventeen define an area of work in their model. Under the stricter reading of disputed cells, none received a full rating for specifying the area apart from the agents that do the work or for recording one accountable area for each responsibility; under the other defensible reading, two and three would. The organizational multi-agent literature of 2002 to 2004 already specifies such areas, and platforms of that literature had already joined an organizational specification to shared infrastructure. Ten vendor products were read from their vendors' documentation: four harnesses, five personal agents and one agent platform. The first readings were made on 12 November 2026, and further readings and decisions on 13 November. As version 0.13 of the model defines the services, every product rates in full on the agent loop, retrieval and tools, and five of the ten on approvals. No reading established a full rating on a directory, identity, policy, evaluation, improvement or releases. The model also proposes that every stop for a person is recorded and reviewed, a rule that the source system does not yet carry out; of 40 sources and products read for it, one received a full rating for such a review. The paper offers one vocabulary across these bodies of work, with the mappings behind it. It is a dated map: the products are changing, and the method is written down so that a mapping can be repeated. WARM is a working model drawn largely from one system, and no evidence yet shows that organizing a system this way improves it.

KEYWORDS AI agents, reference model, agent platform, multi-agent systems, agent governance, human oversight, domains of work

1 Introduction

1.1 The problem

The system this paper draws on was not designed in advance. The author started with a harness product (a loop that calls a model and runs tools) and added capabilities as needs arose: memory, scheduled jobs, approvals, subagents, a second agent, a third. After some months the system worked, and it had no shared description. Its parts had local names, several parts did two jobs, and a new piece of work had no obvious place to go. A team can

also start from a managed platform or a workflow engine, and this paper does not claim that most systems develop as this one did.

The system was built for one purpose, which is that agents finish work without a person approving each action. A shared description serves that purpose. Each time work stops for a person, because an action waits for a decision or because an agent hands over a task it could not do, the question is which part of the system would have had to be different for the work to continue: a rule set in advance, a tool, an access, or an agent's instructions. The question is easier to answer, and to compare across stops, with names for the parts. The model in this paper supplies the names, and its central rule is that every such stop is recorded and reviewed (Section 4.5).

LifeOS is a system of nine persistent agents and one software development team that has run continuously for about three months. Its agents manage software development, personal finance, health records, correspondence, research, marketing and customer support for one person. An inventory taken for this paper listed 1,137 components, among them 197 scheduled jobs, 48 skills, 36 hooks and 13 agent definitions (Section 3.4). Two companion papers describe how its agents are run and how they develop software [35, 32].

1.2 Contribution, scope and status

WARM is a reference model in the sense of the OASIS definition: "an abstract framework for understanding significant relationships among the entities of some environment", which is "independent of specific standards, technologies, implementations, or other concrete details" [1]. A real system that fills in one of the model's parts is an **implementation** of that part, and LifeOS is the **reference implementation**. Appendix D defines the model's terms (Table D1). The paper contributes one vocabulary against which the organizational literature, recent models, frameworks and vendor products are compared, the mappings and ratings behind those comparisons, and a reference implementation that is being refactored toward it. The evidence behind the ratings is in a supplement that accompanies the paper, except for the parts that Section 7 lists. It is not novelty of the parts. The paper makes four contributions, each limited by the evidence behind it.

1. **The model.** It has five layers, six groups of platform services, 25 services, a definition of a domain as a charter of nine parts, two kinds of agent, three kinds of human decision, and a rule that every stop for a person is reviewed (Section 4). Most of its parts have precedents (Section 5), and the paper's work is to put them in one model and to say where each came from.
2. **Three descriptive exercises.** The model was mapped to three agent platforms that the author did not build, compared part by part with fourteen published models, and mapped to the reference implementation, each time with a second rater on a sample (Sections 3 and 5). Each exercise changed the model, and none is a held-out test of the final version.
3. **An account of what the exercises changed.** They added three services, restated others, and withdrew five claims of difference from prior work while keeping narrower observations (Section 5.7).
4. **A dated map of ten vendor products.** Four harnesses, five personal agents and one agent platform were read against the 25 services and the four criteria, first on 12 November 2026 and with further readings and decisions on 13 November, with a second reader on a sample (Sections 3.5 and 5.5). This exercise did not change the model.

WARM describes the system around agents, namely who directs them, what work they do and what they run on. It does not describe how one agent reasons, which the research literature already models [3]. It is not a proposal for

a standard, and it defines no conformance test. It is a dated map with stated coverage. The author expects the method (a protocol, written criteria and a date for every reading) to outlast the ratings, so that a mapping can be repeated when the sources change. The model is at version 0.14, has changed in every version, and has the limits stated in Section 7.

The model was written for the author's own use, to identify the parts of the reference implementation that its next major version will retain, separate or build. The paper is intended for publication so that others who run agents can use the model, correct it or reject it, and it belongs to a series of working papers on running agents [36]. The work has already changed the model (Sections 4.5 and 5.4).

2 Background and related work

Five bodies of work bear on the model. Sections 5.2 and 5.3 report the part-by-part comparison.

2.1 Reference models

The OASIS reference model for service-oriented architecture separates a reference model from a reference architecture, which gives abstract solutions built from its concepts, and from a concrete architecture [1]. WARM is a reference model and not a reference architecture. It states responsibilities and a few design constraints, and it prescribes no product and no deployment architecture. The OSI model, which named layers and left protocols open, is the usual example of the kind [2]. Its layers are strictly stacked, whereas WARM's layers are logical categories, and one of its domains (Software) changes the platform that it runs on, which a strict stack cannot express.

2.2 How vendors decompose an agent platform

A survey on 9 November 2026 read 34 sources, among them fourteen vendor platforms and frameworks, as a search for vocabulary and not as a systematic review. Amazon, Google, Microsoft and Anthropic each name most of a common core: a runtime that hosts agents, sessions, memory, tools reached through a gateway or the Model Context Protocol, identity, observability and sandboxed code execution [4, 5, 6, 7]. "Harness" became a product term in 2026 for the loop that calls the model and runs its tool calls, and several vendors place it as a layer separate from the hosting infrastructure [5, 8].

The survey found no source among the 34 that separates a platform, an area of work and a way of working as three named levels. The Model Context Protocol and the Agent2Agent protocol have evidence of adoption [9, 10], and they define interfaces without modeling a platform as a whole. The United States National Institute of Standards and Technology opened an initiative on agent standards in February 2026 [11], and no ISO/IEC standard specific to agent platforms was found.

2.3 Published models of agent systems

A documented search on 11 November 2026 (Section 3.1) found 61 sources that present a general model of an agent system. Fourteen were compared with WARM part by part. Six came from an earlier reading: Yu, Rabhi, Zhu and Lu [12]; Zhu [13]; Goujon's AAM/5C [14]; Microsoft's Multi-Agent Reference Architecture [15]; Amazon's prescriptive guidance [16]; and Google Cloud's Architecture Center [17]. Eight came from the search: Charterion [18], AEON [19],

CEAD [20], Salesforce's agentic enterprise [21], AOS [22], Steinder and Franke's agent operating system [23], ASOC [24] and EAAF [25].

Several sources describe parts that WARM lacks or states less fully. Charterion and AAM/5C give more complete accounts of authority granted in advance, and AAM/5C separates proposing, deciding, issuing, enforcing and executing an action among five holders. AOS and Steinder and Franke treat the lifecycle of an agent as a primary component. Charterion defines a charter for an agent in more detail than WARM defines a charter for a domain. Among the fourteen mapped models, the readers found no evidence of a named adopter, and the most cited had 19 citations on 11 November 2026. WARM differs from these sources mainly in joining the platform, the domains of work and the agents in one model. The combination is not new, because platforms of the older multi-agent literature made it for the agents of their time (Section 2.4).

2.4 Organizations of agents

Research on multi-agent systems specified a system as an organization well before agents were built on language models. The documented search admitted only work from 2023 onward and missed this literature, and five sources were read after a reviewer of the first draft pointed it out.

In the Agent/Group/Role model of Ferber, Gutknecht and Michel, an organization is made of groups, a role is local to a group, and an agent may hold several roles in several groups [39]. The Gaia methodology of Zambonelli, Jennings and Wooldridge models a system as an organization and its suborganizations, defines each role by its responsibilities and permissions before any agent is assigned to it, and states rules that hold for the organization as a whole [40]. MOISE+ separates three specifications of an organization: its structure of groups and roles, its functions as goals and missions, and the obligations and permissions that tie roles to missions [41]. OperA separates an organizational model from the agents that enact its roles under contracts, and it includes facilitation roles such as a gatekeeper and a matchmaker that keep the organization running [42]. Horling and Lesser survey organizational paradigms, among them federations in which a facilitator routes requests by capability [43].

These sources establish the ideas that WARM's domain layer restates for agents built on language models. An area of work and its roles are defined apart from the agents that fill them, a role belongs to the area, one agent may hold several roles, and roles carry responsibilities, permissions and obligations. In these five sources the facilitation services are roles played by agents.

Other work of the same literature supplies a platform, and three such works were read after the fourth draft of this paper. FIPA's agent management specification defines an agent platform as a set of services: a management system that creates and deletes agents and keeps their names and addresses, a message transport service, and a Directory Facilitator that "provides yellow pages services to other agents" [50]. ISLANDER specifies an electronic institution as roles, scenes, a structure that connects the scenes, and norms [51]. AMELI is a general-purpose middleware that runs any such specification, mediates every interaction of the participating agents and enforces the institution's rules [52]. JaCaMo joins a language for agents, the MOISE model of organizations and an environment of shared artifacts in one programming platform, and it keeps the state of the organization in infrastructure outside the agents, where they can observe it [53]. Its authors call it "the first successful combination" of agent, organization and environment programming in one platform [53]. Joining an organizational specification to shared infrastructure is therefore not a contribution of WARM. None of the three gives each responsibility one accountable area (Section 5.3), and all predate agents built on language models.

Four web searches found no recent model of an LLM-agent system built on this literature. The full texts of four of the later recent sources [45, 46, 47, 48] and of one further search result were searched for the older authors, and none cites them. The references of the fourteen models of Section 2.3 and of the three platforms were not audited, and no citation search was run. One 2024 position paper on LLM-based social simulation cites AGR, MOISE+ and related work and argues for the organizational approach without building a model on it [49].

Five recent sources that bear on domains were read at the same time: Microsoft's adoption guidance, which has teams write a charter for a system of agents [44]; a single-operator case study with a domain layer [45]; WorkMate, which maps a company's departments to agents and permissions [46]; and two reference architectures without an organizational unit [47, 48].

2.5 Domains and platforms in human organizations

Models of human organizations also separate domains from shared platforms. Data Mesh puts domains, a self-serve platform and federated governance in one model [26], Team Topologies puts stream-aligned teams and platform teams in one model [27], and Domain-Driven Design divides a large system into bounded contexts and maps the relations among them [28].

WARM's test for what belongs in the platform has precedents in all three. Data Mesh states that a data platform should "not include any domain specific concepts or business logic, keeping it domain agnostic" [26], and Domain-Driven Design requires that generic subdomains contain nothing specific to the application [28]. An older statement of the idea is the separation of policy and mechanism in operating-system design [29]. WARM's published skill follows the Team API of Team Topologies and the data product of Data Mesh.

WARM differs from the three in defining an area of work by a charter, apart from the people and agents who fill its roles. In Team Topologies the area and the team are one, and in Data Mesh a domain is the teams and systems that manage something. Domain-Driven Design comes nearest, because a bounded context bounds a model and a team is only one way to set the boundary [28]. A WARM domain is not the domain of Domain-Driven Design, which is the subject area that a piece of software models.

A search of arXiv and the web on 11 November 2026 found no peer-reviewed paper that applies the context relationships of Domain-Driven Design, the interaction modes of Team Topologies or the governance of Data Mesh to the organization of LLM agents. Practitioner writing does apply them [37, 38]. Writing on Domain-Driven Design for agents mostly assigns one agent to each bounded context [30]. Team Topologies aims to remove handoffs between teams, and WARM treats a handoff between domains as normal (Section 4.7).

3 Method

The model was developed over four days, from 9 to 12 November 2026, in fourteen versions. Each version was written, set against a source of evidence and revised, and Section 7 states what this formative design implies. Appendix A gives the protocol detail.

The model was made with the system it describes. The work ran in the first version of the reference implementation. The agents of its Research domain did the surveys, the readings, the mappings and the second ratings: one persistent agent planned the work and wrote the model, and it started task agents for each reading and rating. Reviews of the model and of this paper were done by a model from a different vendor, except the

review of the sixth draft, which a model of the agents' own family did. A person, the author, directed the work, decided the questions of substance put to him, and approved each change to the model. Some wording choices and consistency corrections were made under approvals he gave in advance and were reported to him afterward. Every reader, mapper and rater named below was therefore an AI agent working for the author, except where the text says otherwise.

3.1 The starting point and the searches

The first versions of the model were drawn from the reference implementation by grouping its components by responsibility. By version 0.10 the model had five layers, six groups and 22 services. The survey of 34 sources (Section 2.2) supplied vocabulary, including the industry terms for each service in Table A3.

A separate reader then ran a documented search with a written protocol. It admitted work from 2023 onward on agents built on large language models, screened only the first page of results of each query, and departed from the protocol in ways that its record states (Appendix A, Table A1). It found 61 sources that met the inclusion rule, 48 of them new. The reader chose the eight new sources that were mapped as the most complete, which is a purposive choice and not a sample.

3.2 Mapping the model to three platforms

On 11 November, version 0.10 was mapped to three agent platforms that the author did not build: Amazon Bedrock AgentCore [4]; LangGraph with LangSmith and LangSmith Deployment [33]; and OpenClaw [34]. OpenClaw is not conceptually independent of the model, because the author used it before building the reference implementation and the implementation took some of its goals from it.

Three readers first listed each platform's components from its own documentation without being shown WARM. The research agent that drafted the model then recorded eight predictions in a dated file before any list existed. Three mappers, one for each platform and not shown the predictions, assigned every component to exactly one of five categories: one WARM service, another layer of the model, builder tooling (development kits, command-line tools and templates, which the model does not describe), a span of several services, or a misfit. The mappers were told to call a doubtful fit a misfit and a service absent only after a recorded search. A second rater sampled 36 assignments, twelve for each platform, and assigned each one independently.

3.3 Comparing the model with published models

Fourteen published models (Section 2.3) were compared with version 0.11 in two rounds of six and eight sources. A reader listed the parts of each source in the source's own words and assigned each part to a WARM service, another layer, guidance, a span or a misfit. The reader then recorded, for each part of WARM, whether the source has a counterpart in full, in part or not at all. A second rater sampled 126 of the assignments of a source's parts.

These verdicts form a grid of the seventeen sources (the fourteen models and the three platforms) by the 25 services and four further parts of version 0.13 (persistent agents, task agents, agent identity apart from a running instance, and pre-approval). Each cell holds *yes* (the notes say the source has a counterpart), *part* (a partial or strained counterpart), *no* (the notes record that it is absent or record a search that found nothing) or *not assessed*. The grid was assembled from the notes alone, and no source was read again for it.

The domain layer and the Directory were rated separately, under four criteria written and frozen before any rating: C1, the model has a defined unit that stands for an area of work and that more than one agent or role can belong to; C2, the source specifies, for the area and not for each agent, what the area is responsible for and the roles that agents or people fill; C3, a record names one accountable area or owner for each responsibility or capability; C4, an agent can look up at run time, from a kind of work, the agent or role that handles it and how to reach it. A separate rater rated the seventeen sources. It was independent only in not having seen the earlier answers, the model text or this paper, and it was an AI agent working within the author's research process. It rated from primary passages where they could be read, from earlier extracts for some sources, and in one case from a generated summary. It read Charterion and AAM/5C only at targeted pages. The AgentCore ratings on C1 and C2 and parts of the AWS and Charterion ratings rest on earlier extracts, not all the Google pages were read again, some OpenClaw pages could not be opened, and the Salesforce "no" on C2 rests on a generated summary. A keyword search showed the rater one earlier verdict on Charterion, which it states it did not use. A "no" therefore means not found in the material read. When two values were defensible the rater recorded the lower one and marked the cell disputable. A second reader rated ten further sources under the same criteria, five from the organizational multi-agent literature and five recent ones (Section 2.4), without reading some sections, among them parts of WorkMate. After the fourth draft the criteria were clarified on three points on which those two readers had differed: a unit that can only be the whole system rates in part on C1, a requirement or an example alone rates in part at most, and a lookup through an intermediary that any agent may ask rates in full on C4. Two further readers, each without the other's ratings or any earlier rating, then rated eight sources under the clarified criteria. Five were the sources on which the first two readers had differed (AAM/5C, OperA, ASOC, Microsoft's reference architecture, and Horling and Lesser), and three were works read at that point (FIPA's agent management specification, ISLANDER with AMELI, and JaCaMo). The two gave the same value in 27 of the 32 cells, and each of the five differences is one step. Where they differ, Table 5 gives the lower value and the higher as the alternative. The other 22 sources keep their single rating under the first version of the criteria; the five full ratings on C1 among them were checked against the clarified wording and hold.

3.4 Mapping the reference implementation

On 12 November the method of Section 3.2 was applied to the reference implementation against version 0.13. A reader who was not shown the model listed 1,137 components of fifteen kinds from the system's repository. A mapper removed 288 duplicates (for example, a scheduled job and the service unit that runs it), assigned the remaining 849 components, and recorded for each of the 25 services whether the implementation has it in full, in part or not at all, with a recorded search behind each statement that something is missing. A second rater drew a stratified random sample of 60 of the 849 assignments, assigned each one before seeing the mapper's answer, and gave an independent verdict on each of the 25 services.

3.5 Mapping ten vendor products

On 12 November 2026 ten vendor products were read against the 25 services of version 0.13 and rated on the four criteria of Section 3.3. Four are harnesses on which a builder assembles a system: Claude Code [55], Codex CLI [56], Grok Build [57] and Google Antigravity [58]. Five are finished personal agents: Claude Duet [59], Dots [60], Grok Bot [61], Tasks in Gemini, which Google launched as Gemini Spark [62], and Muse [63]. One is an agent platform that a customer runs on its own infrastructure: Claude Managed Agents in its self-hosted form [69], which its vendor launched on the day of the reading and supplies on request. The author chose the first nine as the harnesses and

personal agents of five model vendors, and added the tenth, from one of the five, when it launched. They are not a sample of a larger set.

A written protocol governed the reading. Its second version was frozen after a pilot on two products and before the other seven readers started; the pilot's two readers had rated the Directory row by different rules. The two products of the pilot, Claude Code and Codex CLI, keep their reading under the first version, and Claude Code's Directory value was derived afterward. The tenth product was read later the same day, under the second version. Each product had one reader, an AI agent of one model family, which was given the protocol and the criteria and told to open no other file of this work. A reader rated each service *yes* (the vendor's pages describe a feature that carries out the responsibility and that a user can use today), *part* (some of the definition, a mechanism on which the user must still build, a feature announced and not available, or secondary evidence only) or *no* (not found in the pages searched). Each value carries one exact quotation and its address, from a vendor page opened that day. When two values were defensible the reader recorded the lower and marked the cell disputable. The Directory row was not rated. It is derived from C3 and C4: full if both are full, absent if both are absent, and otherwise in part.

A second reader, a model of another vendor that was given only the protocol and the criteria, rated a seeded random sample of eight of the 24 rated services of each product and all four criteria, 80 service cells and 40 criterion cells. The second reading of the tenth product was made on 13 November 2026. One further cell, the tenth product's Approvals, was second-read by choice and not by the sample, because that value alone decided whether every product rates yes on Approvals. The choice followed from the result: the cell stood in the way of a claim that the agent wanted to make. Like cells of the older products, whose rows state no record of single decisions, were not chosen for a second reading, and a second reader that works blind does not make the choice of cell independent of the result. The final review found five such cells, and they were changed (Section 5.5). The persistent agent that planned this work then decided the cells on which the two readers differed. It knew which reader had given which value, and it wrote the paper that these values serve, so it had an interest in low values on the four criteria. The readers differed on 23 cells, the chosen one among them. For 21 of them it searched the vendor's page for the quotations that decided the value (41 passages on 24 pages, all found); one further quotation, on a page that refused the request, was not checked. The pages behind the other two cells, both of one product, were not opened again. A reviewer of the sixth draft, a model of the first readers' family that had not seen the decisions being made, agreed with 16 of the 19 decisions made by then and disagreed with three (Codex CLI on Policy, Dots on Evaluation and Tasks in Gemini on Models). The decided values were kept, and Section 5.5 reports the disagreement. The four decisions on the tenth product were made after that review. The final review, of the eighth draft and by the second readers' model, agreed with three of the four and disagreed with one (Channels), and it disagreed again with two of the three earlier ones (Dots on Evaluation and Tasks in Gemini on Models). Those values were also kept (Section 5.5). After that draft a further reader of the same family applied the rulings behind the decisions to every cell of the services they govern, for the nine products then in the table, and another did the same for the tenth product's column. The author then decided that a cell which does not follow a ruling is changed to follow it. Table 7 marks thirteen such cells: six found by those two checks and seven found by the final review, five of them on Approvals (Section 5.5). The checkers were given the decisions, so their agreement with a decision does not confirm it. The other services of each product, sixteen or for the tenth fifteen, have one reader. All ten first readers were one model, from the vendor of Claude Code, Claude Duet and Claude Managed Agents, so that vendor's model rated its own three products and every competing product, and the reader of Claude Code ran inside Claude Code. Codex CLI and Dots were second-read by a model of their own vendor. The reader of Codex rated Codex CLI and the surfaces that share it (cloud tasks, the desktop application and the editor extension); this paper calls the set Codex CLI. Muse's product site has a robots file that disallows the first reader's crawler by

name, and it carries a notice against automated collection. The reader fetched each public page listed in that site's sitemap once, with a browser's user agent string, and requested no path that the file disallows for crawlers in general. Nineteen of Muse's 24 rated rows cite one of those pages, and twelve cite no other source. The author kept the column and states the condition here. The products were read against the service definitions of version 0.13, the tenth product's later readings among them. Version 0.14, dated 12 November 2026, changed the wording of Approvals and Improvement after the first readings (Section 4.5), and the reading was not repeated.

3.6 Reading the sources for what they do with people's decisions

Version 0.14 added the review of stops after the readings above, so none of them had looked for a counterpart. On 12 November 2026 four further readers read the 30 sources of Section 5.3 and nine of the ten products of Section 3.5 under a protocol of five questions, frozen before any reader started. The tenth product was read on 13 November 2026 by a further reader under the same protocol. The questions ask whether a source defines a record of each decision by a person with the reason a person was needed (H1), a measure of how often people are required (H2), a review of past decisions that changes the rules (H3), an authority that changes with an agent's record (H4), and tasks handed to a person for lack of capability that are recorded and reduced (H5). The values and the rule for disputable cells are those of Section 3.5. A fifth reader made a bounded search outside those sources (twelve queries, the first page of results each) and rated ten of what it found. Each source had one reader, all of one model family, and there was no second reader. That family is the vendor's of Claude Code, Claude Duet and Claude Managed Agents, so those three products were read on these questions by their own vendor's model. The agent that planned the work checked 44 of the readers' quotations and figures against the pages or documents, among them every quotation used in Section 5.6, and found each as written.

4 The model

The model has two aims and one rule that follows from them. The aims are that agents finish work end to end and that the system improves itself continuously (Section 4.2). The rule is that every stop for a person is recorded and reviewed (Section 4.5, Figure 2). The layers, the services and the domains of the following sections are the parts that such a review needs names for: a stop is removed by changing a rule, a tool, an access or an agent's instructions, and each of those belongs to a named part.

This section states the model at version 0.14 and follows one request throughout. A person asks their finance agent, "There is a duplicate charge from the gym this month. Dispute it." The finance agent can find the charge. It may not write to the merchant, because messages sent in the person's name belong to another area of work, Communications, and need the person's approval. Section 4.9 traces the request across two areas of work and sixteen platform services.

4.1 Agents

An **agent** is an identified actor that carries out one or more assigned roles. The agent is the identity, and an **instance** is a running copy of it that can be restarted or moved without becoming a different agent. Every agent identity traces to a person who is accountable for it.

An **agent definition** describes a role, with its instructions and the **skills** (packaged instructions for a kind of task) that it refers to. Several agents can use one definition, and a definition can be revised while the agent remains the

same agent. A persistent agent's identity, memory and history persist through a change of definition, harness or model. Each such change can alter its behavior, so each is versioned and evaluated.

Agents are of two kinds, which differ in how long the agent holds its role. Table 1 also states the model's conventions for memory and communication. A persistent agent can work unattended, and a task agent can ask a person a question. One definition can serve both kinds. A task agent acts under an identity of its own, and the record states on whose behalf it acts. The role that it fills determines the domain accountable for its work, and the agent or schedule that started it is a separate fact.

KIND	WHAT IT IS	WHAT PERSISTS	EXAMPLE
Persistent agent	Holds its roles across many sessions and tasks.	Its identity, its session history, what it has learned, and a channel on which people and other agents reach it.	A finance agent that a person consults, and that remembers the accounts and last month's decisions.
Task agent	Holds a role for one task. It is started for the task by another agent, a schedule or an event, and it is ended when the task ends.	No continuing role and no memory of its own. It returns a result and work products, and the platform retains its records.	A developer agent started for one issue, which delivers a pull request and ends.

Table 1. The two kinds of agent.

4.2 Two aims

The model is written for systems with two aims. The first is that agents finish work end to end, carrying a piece of work from request to result without a person approving each action. People keep control by directing the work, setting the rules in advance, and deciding the cases that the rules reserve for them.

The second aim is that the system improves itself continuously. The people and agents of each domain change its way of working from evidence, the agents of the Software domain can change the platform itself, and each such change is governed like any other action. The two aims meet in one rule, stated in Section 4.5: every stop for a person is recorded and reviewed. Work stops for a person when an action waits for a decision or when an agent hands the person a task that it cannot, or believes it cannot, do. Either stop is a point at which an agent did not finish the work alone, so the author treats it as the main evidence for improvement. The two aims are the author's design choices for the reference implementation. Several published models state autonomy as a goal [14, 21, 25]. CEAD puts constraints ahead of completion at any cost, and Zhu aims at a human-supervised organization [13, 20].

4.3 Five layers

People direct domains. The work of a domain runs on a platform, which runs on infrastructure and calls providers (Figure 1). The layers are logical categories relative to the boundary of one system and are not a deployment diagram. Agents are not a layer. They fill the roles of domains, as people do, and they run on the platform.

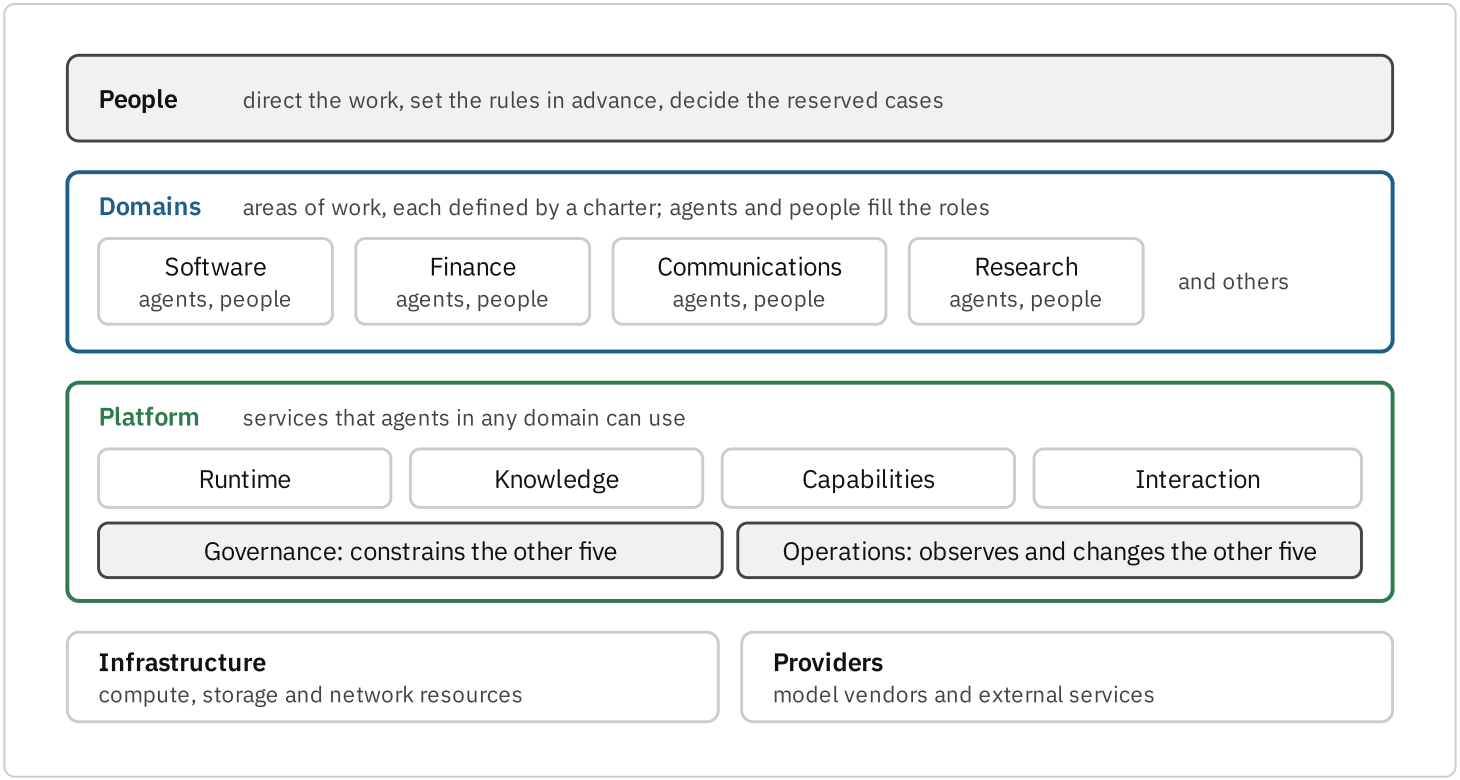


Figure 1. The layers of WARM. The two shaded platform groups apply to the other five groups.

One test places a responsibility in a layer. A responsibility belongs to the platform if it works the same whatever the work is about, and it is domain work if what it does depends on the subject of the work. Holding an action until a person approves it works the same for an email, a payment or a merge, so Approvals is a platform service. The rule that every message in this person's name needs approval concerns one area of work, so it is part of the Communications charter, and the platform applies it.

The test says nothing about how a responsibility is built; a platform service may use a model or an agent inside it, and domain work may be a fixed script. The test is a design choice and cannot be refuted by observation, because a reusable responsibility found inside a domain is treated as a gap in the platform. Its usefulness can be checked by whether a service stays meaningful, unchanged, when a second and a third domain use it.

4.4 Six groups of platform services

The platform answers six questions about agents. Runtime answers how, where and when agents run, and agents use the other working groups from a session or a task. Knowledge answers what agents know and keep, Capabilities what they can use, and Interaction who they work with, among other agents, people and applications. Governance (what agents are allowed to do) constrains the other five groups, and Operations (how the system is seen, judged, improved and kept running) observes and changes them. These two are cross-cutting groups.

Governance follows three rules, derived from the platform and published-model comparisons (Appendix B).

1. **Governance covers an action only if the action passes an enforcement point.** A call that reaches a tool, a model or a store without passing one is not governed, whatever the rules say (Table D1 defines the term).
2. **A positive decision is issued as a grant**, which is authority for one action or one task.

3. **Whoever proposes or performs an action does not approve it.** An agent cannot grant itself authority.

4.5 Human decisions and pre-approval

For each class of action, people choose one of three kinds of decision.

1. **Reserved for people.** An agent may prepare the step and may not carry it out.
2. **Pre-approved under conditions.** A person authorizes a class of actions in advance under stated conditions, for example that two independent reviewers agree, that the checks pass, and that the action can be undone and is reported afterward. At run time Policy checks the conditions and issues the grant, and no person is asked. A condition may be a judgment made by a model, by an agent other than the one that proposes the action, and not only a rule that a program can check.
3. **Held for a person.** When the rules allow an action but no pre-approval applies or its conditions are not met, the action waits for a person's decision. This kind is the fallback.

All authority traces to a person. An agent has a role or a rule because a person decided it, and changing a role or a rule is itself an action that is reserved, pre-approved or held. A system that improves itself continuously can change its own roles and rules only if some of those changes are pre-approved. An earlier version of the model reserved such changes to a person. After the mapping of the reference implementation (Section 5.4), the author retained the practice found there as a design choice and revised the model.

The model prefers a pre-approval to holding each action, because a system in which every action waits for a person does not finish work end to end. Two rules apply to every pre-approval. First, it states its term. It names an end date or a condition that ends it, or it says explicitly that it does not expire, and a person can revoke it at any time. Second, it covers an action that cannot be undone only if it names that kind of action explicitly. Whether to pre-approve such actions at all is a choice for the people who direct a system, and the published models disagree. AAM/5C requires a person for each irreversible act [14], and Charterion allows autonomy where a compensating step exists [18]. The kinds of decision apply to a class of action, so one agent may merge reviewed changes under a pre-approval and have every outbound message held (Appendix B).

Every stop for a person is reviewed. The preference for pre-approval says what to prefer, and this rule is how a system gets there. Work stops for a person in two ways. In the first, an action waits for a decision, and Approvals records why it stopped, what the person decided and what followed. In the second, an agent hands the person a task because it cannot, or believes it cannot, do it, such as a login or a document it could not reach, and Orchestration records what was handed over and why. Both write to one record of stops, which Improvement reads. Work stops for one of five reasons: the step is reserved for people; no pre-approval covers the action; a pre-approval's conditions were not met; an agent chose to ask; or an agent lacked, or believed it lacked, the means to do the step. Improvement reviews each record and answers three questions. Did this stop need a person? For a task that was handed over, the question is whether an agent could have done it. If the stop needed a decision, could the person have made it in advance, as a pre-approval with conditions, and not while the work waited? What change would remove this stop next time: a rule, a pre-approval, a tool or an access, or the agent's instructions? The review runs after the work and does not hold it. A stop like one reviewed recently is not reviewed again: it is recorded and counted against the earlier review, and a high count reopens that review. The author added this treatment of repeated stops on 13 November 2026, after the final review of this paper, and the model's text will carry it at its next version. A review ends in one of five outcomes: remove the stop; propose a pre-approval to the person; give agents the means to do the step; keep the action held; or keep the step reserved, with

the reason. A proposed pre-approval is a change to a rule, so a person decides it. The review of a stop is itself work that a model can do: it reads why a person was needed and what the person did, and it proposes the change. Because a model does the review and it runs after the work, the person is not asked a second time and sees only what the review proposes. The author expects that this lets a system review every stop, and not a sample of them or a rate (Section 6). A stop that no rule required, where an agent chose to ask, is the first kind to remove: when the review finds that the stop did not need a person, removing it needs no new pre-approval and no new tool, only a change to the agent's instructions. The review is what moves a class of action from held toward pre-approved; the three kinds of decision do not change, and the model still defines no levels of autonomy. AAM/5C names a related idea, graduated escalation, in which the level of human involvement is set in advance by risk and amount [14].

An action that a person approves as proposed many times is a candidate for a pre-approval, and a task handed over many times is a candidate for a tool. A hand-off can be removed by giving agents the means to operate the applications a person would have used (computer use), an access they lacked, or a tool. The measure of autonomy counts both kinds of stop: the work that waits for a person's decision, and the work an agent hands to a person because it cannot, or believes it cannot, do it. The count in a period, with the share of decisions approved as proposed, shows how far a system finishes work end to end (Figure 2). A domain pattern such as the factory (Section 4.7) is where this review leads: the person's decision has moved from each result to the rules set in advance. The rule was added in version 0.14 at the author's direction, after the exercises of Section 5 and not from a finding of them. The reference implementation does not yet carry it out. The rule is not new: Charterion defines a review of an agent's record that changes its authority [18], and fourteen other sources and products have part of one (Section 5.6).

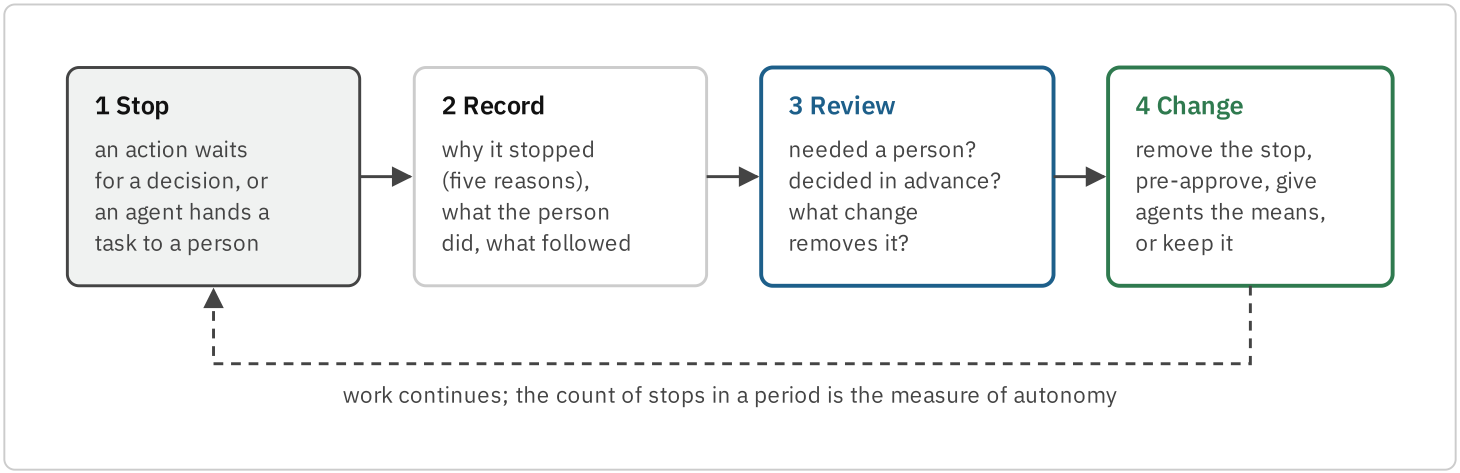


Figure 2. The review of stops for a person. Approvals records a decision and Orchestration records a task handed over, both to one record; Improvement reviews it; a change to a rule is itself decided by a person. The loop is a proposal of version 0.14 and has not been run.

4.6 Twenty-five services

Each group holds a few services, each with one responsibility (Table 2). The names are responsibilities and are not products, so one product may cover several responsibilities and one responsibility may be shared by several products. The model does not state each service's contract, which is what the service guarantees on failure, cancellation, delegation, stale state and recovery.

Group	Service	Its one responsibility
Runtime	**Harness**	Run the agent loop: decide what goes into the context window, send it to the model, and carry out the tool calls the model asks for.
Runtime	**Sessions**	Keep the history and state of a stretch of an agent's work, so that the agent can pause, resume and hand off, and so that the record remains after the work ends.
Runtime	**Tasks**	Track each unit of work from start to finish, coordinate its retry, recovery and cancellation, and keep the original request.
Runtime	**Environments**	Give each agent a place to act, isolated from the others: sandboxes, workspaces, browsers.
Runtime	**Scheduling**	Decide when work starts: at a time, on an event, and within the limits on how much may run at once.
Runtime	**Lifecycle**	Create, suspend, resume and end agents, and keep the list of which agents exist.
Knowledge	**Memory**	Retain what agents learn across sessions (facts and events), with where each entry came from, and keep it current.
Knowledge	**Retrieval**	Fetch material from named sources: documents, data, notes.
Knowledge	**Artifacts**	Store what agents produce (reports, files, pages) and its versions.
Capabilities	**Models**	Mediate requests to models and choose the model for each task, across more than one model or vendor.
Capabilities	**Tools**	Mediate calls to tools, local or external.
Capabilities	**Registry**	Hold the versioned definitions of agents, tools and skills, so that each can be found and loaded when a task needs it.
Interaction	**Directory**	Record which domain is accountable for each responsibility, and how to reach a role of that domain.
Interaction	**Messaging**	Deliver messages between agents, and events to their subscribers.
Interaction	**Orchestration**	Coordinate assignments among agents, people and fixed steps: who does what, what depends on what, and the results that come back.
Interaction	**Channels**	Carry requests and results between agents and the people and applications that use them, on the channel each one uses.
Governance	**Identity**	Establish who is acting (each agent, person and service, and on whose behalf), record the person accountable for each agent identity, and revoke an identity.
Governance	**Secrets**	Hold values that agents must not see, and release each one only at the point of use.
Governance	**Policy**	Decide by rule whether an action may proceed, what content may enter and leave, and what stored data is kept; issue a grant when an action is allowed.
Governance	**Approvals**	Obtain and record people's authorizations: approvals given in advance for a class of actions under stated conditions, and decisions on single actions that are held, each with why a person was needed and what the person decided.
Operations	**Observability**	Collect evidence of what agents and the platform did.
Operations	**Evaluation**	Assess behavior and work products against stated criteria, and report "unknown" when a check could not run.
Operations	**Improvement**	Turn evidence (traces, evaluations, reviews, people's decisions and tasks handed to people) into proposed changes to agents, skills, rules and the platform.
Operations	**Releases**	Version and release changes to agents, skills, rules, the platform and the infrastructure the system manages, with a way back.
Operations	**Reliability**	Keep services available and restore them after a failure.

Table 2. The 25 platform services. Table A3 gives the industry terms for each service.

Three boundaries need a statement. Sessions, Tasks and Memory keep different things, namely the history of a stretch of work, the lifecycle of a unit of work, and what was learned. One store may serve more than one of them. The Registry is a catalog of the agents, tools and skills that exist, and the Directory records who is accountable for what. Improvement proposes a change, a person or a release process decides on it, and Releases makes it.

4.7 Domains

A **domain** is an area of work: a body of specialized knowledge, a set of responsibilities that belong together, and one or more patterns for carrying them out. Its **charter** writes these down (Table 3). A domain does not act. The agents and people who fill its roles act, on the platform's services.

THE CHARTER STATES	WHAT IT IS	PLATFORM SERVICES THAT COMMONLY CARRY IT
Goals	What the work is for, and who is accountable.	(set by people)
Responsibilities	What the domain is accountable for.	Directory
Roles	The positions in the domain and their duties, filled by agents and people. Each role that an agent fills has an agent definition.	Registry, Directory
Workflow	How work starts, and how it moves from request to result.	Tasks, Scheduling, Orchestration
Domain knowledge	What must be known to do this work, including its skills.	Memory, Retrieval, Registry
Domain tools	Connections this work needs.	Tools
Domain rules	Limits, pre-approvals, required approvals and preferences.	Policy, Approvals
Acceptance criteria	How to tell that a work product is good.	Evaluation
Published skill	What the domain offers other domains, and how to ask for it.	Registry, Directory

Table 3. The nine parts of a domain's charter. The third column lists common carriers and is not complete.

Responsibilities and the Directory. A domain's responsibilities belong together because they draw on the same knowledge, tools and rules and serve the same goals. A responsibility is stated at the size at which an agent of another domain would hand work over. "Messages sent in a person's name" is a responsibility, and one checking account is knowledge and a tool inside Finance. At that size exactly one domain is accountable for each responsibility, people set the division, and the Directory records it. An entry holds a responsibility, the accountable domain, and how to reach a role of that domain. Finding a destination grants nothing, and Identity, Policy and Approvals still decide whether the request may be made and whether the work may be done.

The published skill. Each domain states, for the agents of other domains, when to ask it, how to ask, and what comes back. This part of the charter is its published skill. The Registry holds the skill, and the Directory's entries point to it.

Handing work over and starting work. An agent that meets work for which another domain is accountable hands it over and gets a result back. Messaging carries the request, the platform starts an agent if the accountable domain has none running, and Orchestration tracks the assignment and the result, since delivery alone

establishes neither acceptance nor completion. Agents in different domains ask each other in ordinary language, and an agent reaches a platform service through a defined interface with stated guarantees. Work also starts when a person or an application asks through Channels, or when a time or an event arrives through Scheduling.

Subdomains. A **subdomain** is a named group of a domain's responsibilities that has its own workflow. It keeps the domain's goals and rules, and the domain remains accountable for its work. A subdomain is internal to its domain and does not appear in the Directory, and the domain determines its internal division of work. A subdomain is optional, and the model states no rule for when a group of responsibilities should become a subdomain or a domain of its own. On 12 November 2026, after the mapping, the author decided to treat running the product in production as a subdomain of the reference implementation's Software domain, because the two kinds of work arrive differently, are done by different roles and are judged done differently while serving the same goals. The system's own records had listed product operations separately, and the second rater read it as a separate domain. In Domain-Driven Design a subdomain is a part of the subject area that software models [28]; AEON uses the word as WARM does [19].

Domain patterns. A domain pattern is a recurring way of carrying out a group of a domain's responsibilities. It states how the work arrives, who carries it out, who judges it done, and when a person is involved. A pattern attaches to a group of responsibilities (a subdomain, where the domain names one), and a domain usually mixes several. WARM treats the software factory [31] as one pattern. In the factory pattern, work waits in a backlog, a coordinating agent hands items to task agents, each item passes through fixed stages, and each result is checked against stated criteria and by reviewers before it is accepted. The reference implementation's Software domain carries out its main work this way [32]. Which patterns recur across domains is an open question.

The model sets no number of agents for a domain. A domain may have one or several persistent agents, or none, with task agents started when work arrives, and one persistent agent may hold roles in more than one domain. For each task that is a domain's work, one role is active and one domain is accountable for the result.

4.8 Implementations and uses

"Implementation" is one relation that applies at every layer. Agent Scrum implements the main work of the Software domain in the factory pattern [32], and Claude Code with the LifeOS tooling built around it implements the platform of the reference implementation. A domain can have more than one implementation. The model is meant for describing a system by mapping its parts, placing new work with the platform test, comparing two systems service by service, choosing a kind of decision for each class of action, and structuring a body of code or writing. A part that fits nowhere raises a question about the part or the model, and a match of names does not make two services equivalent. A system does not need every service, and a missing service may be a deliberate choice.

4.9 The request, traced

Table 4 follows the request through the model. The whole trace is intended behavior under the model, following the reference implementation's rules for email, and is not a record of that system. An asterisk marks the steps that the system does not yet do or that were not verified, namely the published skill (step 3), tracking of the handoff (step 4, Table 6) and release of the credential only at the point of use (step 7).

STEP	WHAT HAPPENS	DOMAIN	PLATFORM SERVICES
1	The request reaches the finance agent in the conversation the person is using.	Finance	Channels, Identity, Sessions
2	The finance agent finds the two charges and checks them against past decisions.	Finance	Harness, Models, Tools, Memory, Retrieval; Policy checks each call
3*	A dispute needs a message to the merchant. Messages in the person's name belong to Communications, so the finance agent loads the Communications skill to see how to ask.	Finance	Directory, Registry
4*	It hands the work over: the facts, what is wanted, and what counts as done.	Finance to Communications	Orchestration, Messaging, Tasks
5	The communications agent drafts the message, using the contact and the person's voice.	Communications	Harness, Models, Retrieval
6	The proposed send is checked. No pre-approval covers a message in the person's name, so the action is held and the person is asked.	Communications	Policy, Approvals, Channels
7*	The person approves. Only then is the mail credential released and the message sent.	Communications	Approvals, Secrets, Tools
8	The result goes back. The finance agent records that the dispute message was sent and tells the person.	Finance	Messaging, Orchestration, Memory, Channels

Table 4. One request across two domains, as intended under the model.

The request waits for a person because this class of action is held. Had the person pre-approved dispute messages to merchants under stated conditions (a duplicate charge, an amount under a limit, a fixed wording), step 6 would issue the grant and the person would be told afterward. If the handoff is not acknowledged or the send fails, the assignment stays open in Orchestration and Tasks for the finance agent and the person to act on. Step 8 records that a message was sent, which is not the same as the charge being reversed.

5 Results

5.1 Three platforms

The three mappers assigned 1,301 components (Table A2, Figure 3). Between 64% and 67% of each platform's inventory entries were assigned to one service or one other layer of the model. The prediction was 90%. These shares depend on how each reader divided a system into entries, and the entries are not a common unit, since one may be a product and another a configuration option. With builder tooling set aside the shares are 71%, 81% and 91%, so much of the apparent agreement among the three comes from the choice of denominator.

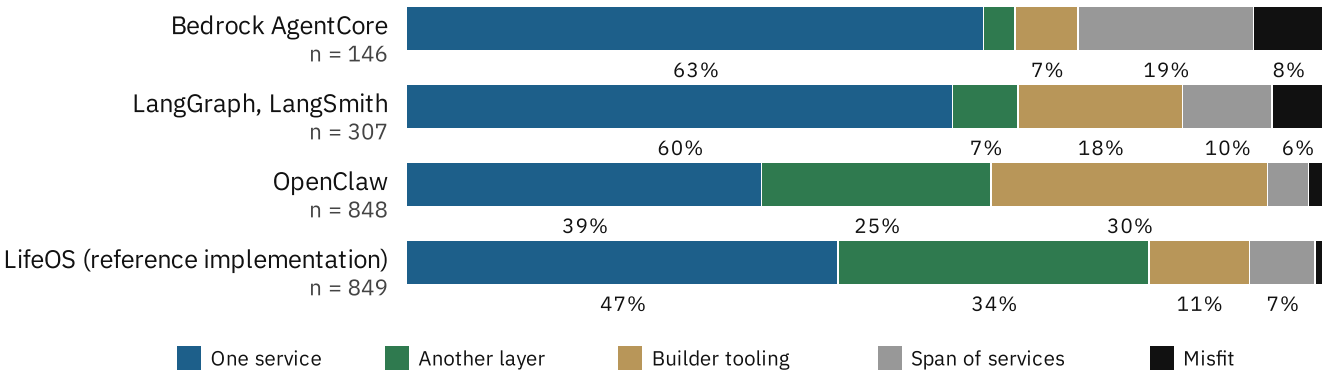


Figure 3. Share of inventory entries in each assignment category. The LifeOS bar is not comparable evidence of fit, because the model was drawn partly from LifeOS and revised after the platforms were mapped. Most of its "another layer" share is domain work (239 of 288 components).

Predictions. Of the eight predictions written before the mapping, one was confirmed, three were partly right and four were wrong (Appendix A.3). The prediction confirmed on all three platforms is that, in the mappers' reading, none of the three fully has a layer for domains or a record of which area of work is accountable for what.

What did not fit. The misfits and spans identified parts that the model omitted or had assigned to an unsuitable service, among them a way for a program to call an agent and a loop that turns evidence from past work into changes (Appendix B).

Second rater. The second rater agreed with the mappers on 29 of 36 sampled assignments, disagreed on four and found three unclear (Appendix A.3).

5.2 Platform services across seventeen sources

Figure 4 shows the grid of Section 3.3 for the 25 services and four further parts, with the reference implementation added as a first row. Over the seventeen sources, 206 of the 493 cells are a counterpart in full, 222 are a counterpart in part, 54 are absent and 11 were not assessed.

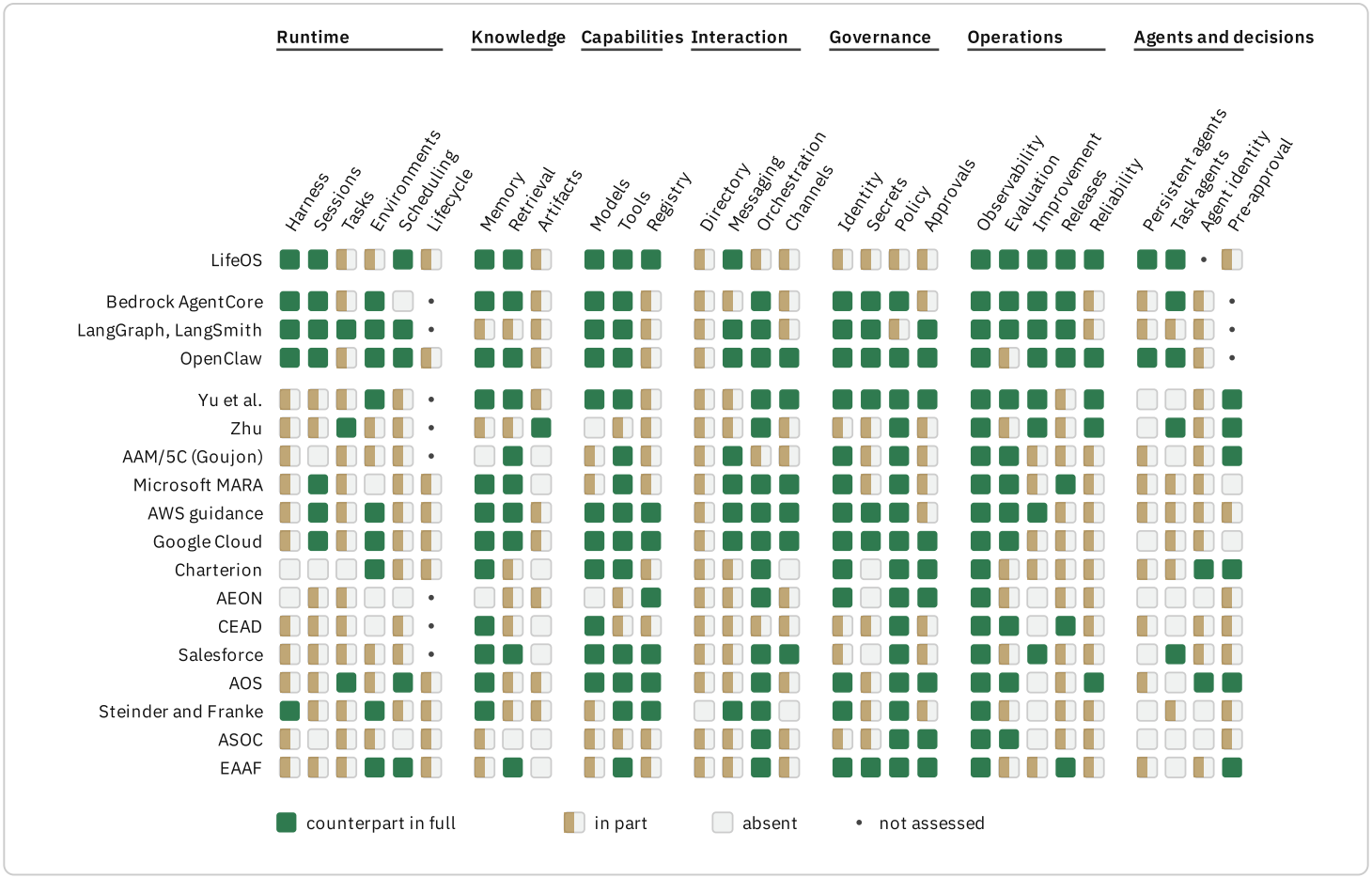


Figure 4. Eighteen rows by 29 parts of the model. The first row is the reference implementation, from which the model was partly drawn, assessed on what it has built; the other rows are assessed on what they describe. A dot marks a cell that the assessments against versions 0.10 and 0.11 did not cover. The Directory column uses the separate rating of Section 5.3, and the first row uses the mapper's values for the two services that the raters dispute (Section 5.4).

The platform services are widely covered. Every one of the 25 services has at least a partial counterpart in nine or more of the seventeen sources. Eleven services have at least a partial counterpart in all seventeen: Tools, Registry, Messaging, Orchestration, Identity, Policy, Approvals, Observability, Evaluation, Releases and Reliability. Observability has a full counterpart in all seventeen, Policy in sixteen and Orchestration in fifteen.

Three services have limited coverage in the sources. Artifacts has a full counterpart in one source, a partial one in nine and none in seven. Improvement is absent from five sources. Lifecycle has no full counterpart; nine sources have it in part, and eight were not assessed because the service entered the model after they were mapped.

The two kinds of agent are rarely both present. One source, OpenClaw, has both persistent agents and task agents in full. Seven sources have no persistent agents and seven have no task agents.

Pre-approval is common in the published models. Six of the fourteen models state authority granted in advance under conditions in full, six state it in part and two do not state it as a whole. The three platforms were not assessed on this point.

Second rater. On the 126 sampled assignments of source parts, the second rater agreed with the readers on 107 (85%), disagreed on eight, found three unclear, and differed on eight more only because version 0.12 had added a part that the readers' version lacked. Without those eight, agreement is 107 of 118 (91%). This sample did not

include the reverse-direction verdicts from which the grid is built, so it is not evidence for the reliability of the grid's cells.

5.3 Areas of work, their specification, and the directory

Table 5 gives the ratings on the four criteria of Section 3.3 for the seventeen sources and for thirteen sources read after the reviews of the first and fourth drafts.

Source	C1. An area of work exists	C2. The area is specified apart from who does the work	C3. A record of accountability	C4. Finding where to send work
Platforms				
Bedrock AgentCore	no	no	no (part?)	**yes**
LangGraph, LangSmith	part (yes?)	part	no	part
OpenClaw	part	part	part	part
Published models of LLM-agent systems				
Yu et al.	no	no	no	part
Zhu	part	part	no (part?)	part
AAM/5C (Goujon) †	part (yes?)	part (yes?)	part (yes?)	part
Microsoft MARA †	part	no (part?)	part	part (yes?)
AWS guidance	part	part	part	part (yes?)
Google Cloud	part (yes?)	part	no	part
Charterion	**yes**	part (yes?)	part	no (part?)
AEON	**yes**	part	part (yes?)	**yes**
CEAD	part (yes?)	part	part (yes?)	part
Salesforce	part	no	part	part (yes?)
AOS	no (part?)	no (part?)	part	**yes**
Steinder and Franke	no	no (part?)	no	no (part?)
ASOC †	no (part?)	no (part?)	part	part
EAAF	part	no (part?)	no	part
Organizational multi-agent literature, 2002 to 2004				
AGR (2004)	**yes**	part (yes?)	no	no
Gaia (2003)	**yes**	**yes**	no (part?)	no
MOISE+ (2002)	**yes**	part (yes?)	part	no
OperA (2004) †	part	part (yes?)	part	part
Horling and Lesser (2004) †	part (yes?)	part	part	**yes**
Agent platforms and organization infrastructure, 2002 to 2013				

FIPA agent management (2004) †	no (part?)	no	part	**yes**
ISLANDER and AMELI (2002, 2004) †	part	part	part	part
JaCaMo (2013) †	**yes**	part (yes?)	part	part (yes?)
Further recent sources				
Microsoft adoption guidance	part	part	part	part
Beyond the Agent Harness (2026)	part	part	no	no
WorkMate (2026)	part	part	part	part
Kandogan et al. (2025)	no	no	no (part?)	**yes**
EDDOps (2025)	no	no	no	no

Table 5. Four criteria, rated from primary passages where they could be read, from earlier extracts for some sources and in one case from a generated summary (Section 3.3). A "no" means that the criterion was not found in the material read. A value in parentheses marks a cell the rater judged disputable and gives the other defensible value; the rater recorded the lower value in every such case. Rows marked † were rated by two readers under the clarified criteria (Section 3.3); where the two differ, the cell gives the lower value and the higher as the alternative. Each other row was rated once, by one of two earlier readers.

Among the seventeen sources. Two sources, Charterion and AEON, have an area of work as a defined part of their model; ten have one in part and five were rated no. No source received a full rating for specifying the area apart from who does the work, and nine received a partial one. No source received a full rating for a record of accountability, and ten received a partial one. Three sources let an agent look up where to send a kind of work. Twenty-three of the 68 cells are disputable. If every disputable cell took its higher value, six sources would meet C1, two would meet C2 and three would meet C3, and AEON would meet the derived Directory criterion. Of the seven disputable cells for C2, two have "yes" as the alternative, and of the five for C3, three do. The zero results for C2 and C3 therefore hold only under the stricter reading of those two and three cells. The nearest source is AEON, which meets C1 and C4. It specifies each subdomain's scope and not its roles, and its registry names the subdomain that provides each capability without calling it accountable.

An earlier grid assembled from notes was stricter than this rating wherever the two differ (Appendix A.3).

The older literature on agent organizations. AGR, Gaia and MOISE+ have an organizational unit as a defined part of the model, and a system can hold several [39, 40, 41]. Under the clarified criteria both readers rated OperA in part on C1, because its unit is the whole society [42], and one of the two rated the paradigms of the survey in full [43]. Gaia is rated in full on C2 [40], and AGR, MOISE+ and OperA in part or, on the other reading, in full. None of the five received a full rating for a record of accountability.

Platforms of the older literature. FIPA's specification has no organizational unit. Its Directory Facilitator lets any agent find, from a type of service, the agents that offer it and their addresses, which both readers rated in full on C4. Its entries can name an owner for a service, and the specification states that registering implies no "future commitment or obligation" [50]. Both readers rated ISLANDER with AMELI in part on all four criteria: the institution is the whole system, its inner units are activities, and the middleware keeps each agent's pending obligations [51, 52]. JaCaMo was rated in full on C1 for its groups and in part on the other three. An artifact of its

platform records which agents are obliged to achieve each goal of the organization and by when [53], and the platform's documentation lists the groups responsible for each scheme of goals [54]. That is the nearest record to the model's Directory among the 30 sources, and it allows several responsible groups for one scheme.

The boundary cases were rated again. In the fourth draft one reader had rated the seventeen sources and another the ten later ones, so reader and source group were confounded. The two had differed on AAM/5C against OperA on C1 and C2, on ASOC and OperA on C4, and on Microsoft's full rating on C4 (Appendix A.2). Two further readers rated those sources under clarified criteria (Section 3.3). Both rated AAM/5C and OperA in part on C1, since each has one unit for the whole system. Both rated ASOC and OperA in part on C4 and the survey's facilitator in full, and both rated Microsoft's registry in part with a full rating as the alternative. The two differ on C2 for AAM/5C, ASOC and Microsoft, and on C1 for ASOC and the survey. The other sources of both groups were not rated again, so any difference between the older and the recent groups on C2 remains weakly supported.

Across all 30 sources. Six sources meet C1, one meets C2 and six meet C4, and none received an undisputed full rating on C3, a record that names one accountable area for each responsibility. With every recorded higher alternative the counts would be 11, 7, 3 and 10. The nearest are records of an owner for each agent, asset or service, a provider for each capability, obligations for each role, or the obligations that each agent holds while the system runs. Requiring one accountable owner for each responsibility is the model's design choice. AAM/5C deliberately allows two accountable parties on one row of its accountability table, so it differs from the model on this point and is not thereby missing accountability.

What the four criteria do not cover. The criteria do not test the other parts of a WARM charter, such as acceptance criteria, the area's tools and knowledge, or what the area offers other areas. The reader of the ten later sources noted that none of them gives each unit one written charter of that kind, an observation not rated under a criterion and limited to those ten sources. The older sources carry related content across several artifacts (Gaia's role, interaction and service models; MOISE+'s linked specifications; OperA's contracts), so the difference may be one of packaging the content into one document.

5.4 The reference implementation

Of the 849 components of LifeOS, 400 were assigned to one platform service, 288 to another layer (239 of them to a domain, 41 to providers and 8 to infrastructure), 93 to builder tooling, 61 to a span and seven to no part of the model (Table A2). Appendix A.3 describes the seven misfits.

Services. Neither the mapper nor the second rater found a service absent. The mapper found 14 services present and 11 present only in part. The second rater, deciding independently, found 20 present and 5 partial, which agrees with the mapper on 17 of the 25, and after reading the mapper's reasons moved to 16 present and 9 partial. The rater's notes name anchoring toward the mapper as a risk. The two still differ on Artifacts and Approvals, so the range of 9 to 11 partial services describes the two positions after discussion and is not a statistical interval. Table 6 lists the services that the mapper judged partial, all four Governance services among them; the second rater judged one of the four, Approvals, present. For Artifacts and Approvals, coverage of the service as version 0.13 defined it (Table 2 now gives version 0.14), compliance with the model's wider rules, and preferences that the model does not require are separate questions. On the version 0.13 service definitions alone, the second rater's "present" is the better-supported reading for both, and the mapper's "partial" reflects the additions that the table lists.

SERVICE	KIND OF GAP THE MAPPER RECORDED
Tasks	No single record of a unit of work across the kinds of work the system runs.
Environments	Incomplete isolation between persistent agents. Task agents work in separate workspaces.
Lifecycle	No single list of the agents that exist, task agents included.
Artifacts (disputed)	Work products and their versions are stored, with outside hosts and among the notes. The mapper also looked for a store that the system operates, which the model does not require.
Directory	One responsibility has two accountable domains and one has none. An entry does not say how to start an agent. No domain has a published skill.
Orchestration	A handoff between domains is delivered, and its acceptance and completion are not tracked.
Channels	No interface through which an application can call an agent.
Identity	Incomplete attribution.
Secrets	Incomplete mediation of credentials at the point of use.
Policy	Incomplete enforcement: grants are issued for one class of action only, and not every action passes an enforcement point.
Approvals (disputed)	Held actions and pre-approvals are obtained and recorded. No pre-approval stated its term, which a rule of Section 4.5 requires. The pre-approvals are not kept in one record, which the model does not require.

Table 6. The services that the mapper judged partial in the reference implementation, stated as kinds of gap. The second rater judged Artifacts and Approvals present.

Charters. The system has no charter document for any domain. Of the nine parts of a charter, five are present across the system's files (roles, workflow, knowledge, tools and rules) and four are partial. No goals are written for a single domain. Acceptance criteria are stated for Software and for scheduled jobs, and for no work product of any other domain. The system has no published skill. The instructions for asking another domain are copied into each agent's definition, and a check found one such copy that contradicts the directory.

The directory. The reference implementation's record of 15 responsibilities was created on 11 November 2026, during this work and because of it. It is evidence that the idea can be written down for one system in a day. It is not evidence that the idea was already in use.

Human decisions. All three kinds of decision are in use. Deleting a contact is reserved for the person. Outbound messages, changes to financial records and administrative changes to the product are held. Three pre-approvals were found. Two cover the merging of changes that pass stated reviews and checks. Under the third, agents change agent definitions and shared rules under a standing permission with stated conditions. None of the three stated its term when the mapping was made. The repository holds 816 commits marked as made under the third between 21 August and 12 November 2026, and their compliance with its conditions was not evaluated. The third contradicted the model as it then stood, its effects were not evaluated, and the model was changed to represent the practice (Section 4.5). The author has since decided its term.

Second rater. The second rater agreed with the mapper on 42 of 60 sampled assignments (Appendix A.3). The two split on whether a fit was clean or strained, in both directions, so the mapper's count of strained fits (183 of 400) is

unreliable.

5.5 Ten vendor products

Table 7 gives the values. Of its 250 cells, 106 are yes, 127 are part and 17 are no. No harness has a cell rated no: Claude Code rates yes on 18 services, Codex CLI and Antigravity on 12 each and Grok Build on 11. The five personal agents rate yes on 6 to 12 services, and 16 of the 17 no values are theirs. The platform, Claude Managed Agents, rates yes on 11 services, part on 13 and no on one, Improvement. Thirteen of the 250 cells were changed after the readings so that they follow the rulings behind the decided cells or the wording of a definition, as described below and listed in Table 7a; before those changes the counts were 116, 116 and 18. Dots, at 10, is the product on which the two readers agreed least (four of eight sampled cells).

Service	Claude Code	Codex CLI	Grok Build	Anti-gravity	Claude Duet	Dots	Grok Bot	Tasks in Gemini	Muse	Claude Managed Agents	yes	part	no
Runtime													
Harness	yes†	yes	yes	yes†	yes	yes†	yes†	yes	yes	yes	10	0	0
Sessions	yes	yes†	yes†	yes	part*‡	yes	yes†	yes	part*†	yes	8	2	0
Tasks	yes	part*‡	part*†	part*†	part*†	part§	part	part*	part*†	part*	1	9	0
Environments	yes	yes	yes	yes	yes	part*‡	part*	part§	yes	part*‡	6	4	0
Scheduling	yes†	part*†	part*‡	part†	part*†	yes	yes	yes	part*†	part†	4	6	0
Lifecycle	yes†	yes	yes	yes	no	yes†	part*	part†	part†	yes	6	3	1
Knowledge													
Memory	part*	yes	part*	part	part*	part*	part*	part*	part*	yes†	2	8	0
Retrieval	yes†	yes	yes†	yes	yes†	yes	yes	yes†	yes	yes	10	0	0
Artifacts	yes	part*	part*†	part*†	part†	part*†	part†	part*	part*	part	1	9	0
Capabilities													
Models	yes	yes†	yes	yes†	no*†	yes	yes	part*‡	part*	yes§	7	2	1
Tools	yes	yes	yes	yes†	yes	yes	yes	yes†	yes†	yes	10	0	0
Registry	yes†	part*	yes	yes	part*	part*	part	part	part	yes†	4	6	0
Interaction													
Directory	part	part	part	part	part	part	part	part	part	part	0	10	0
Messaging	yes	part*	part*	yes	no	part*	yes	no*	part	part	3	5	2
Orchestration	yes	part*	yes	yes	no*	part§	part*‡	part	part*	part	3	6	1
Channels	yes	yes	yes	yes	yes	yes	yes†	part*	yes	part*‡	8	2	0
Governance													
Identity	part*†	part§	part	part	part	part*‡	part*	part	part†	part*	0	10	0
Secrets	yes	yes†	part	part*	yes	yes†	yes	part*†	yes	yes†	7	3	0
Policy	part§	part*‡	part§	part*	part*	part*‡	part*	part*	part*‡	part*	0	10	0

Approvals	yes†	yes	part§	part§	part§	part§	yes†	part§	yes	yes‡	5	5	0
Operations													
Observability	yes†	yes	yes†	yes	part*‡	yes	yes†	yes	yes†	yes	9	1	0
Evaluation	part*	part*†	part*†	part	no*	no*‡	no	no	no*	part*	0	5	5
Improvement	part*	part	part*‡	part*	no*	part	part*	part§	part	no†	0	8	2
Releases	part*	part	part	part†	no*†	no*	part†	no	part	part*	0	7	3
Reliability	yes	part	part	part*	no*	part	yes	no*†	part	part*†	2	6	2
yes	18	12	11	12	6	10	12	6	8	11			
part	7	13	14	13	11	13	12	15	16	13			
no	0	0	0	0	8	2	1	4	1	1			

Table 7. Ten vendor products against the 25 platform services, from the vendors' pages as first read on 12 November 2026; second readings, the tenth product's further readings and the decisions used the pages as they stood on 13 November (Section 3.5). An asterisk marks a cell that still carries a mark of disputable after the decisions; the lower value is shown. † marks a cell that a second reader also rated, with the same value. ‡ marks a cell on which the two readers differed, with the value decided afterward. § marks a cell changed after the readings so that it follows a ruling behind those decisions or the wording of a definition (Section 5.5); it carries no asterisk, and Table 7a gives its value as read, which remains the alternative. A cell with no mark has one reader. Directory is derived from C3 and C4 (Table 8). Nineteen of Muse's 24 rated rows rest in whole or in part on pages read under the conditions stated in Section 3.5.

What every product supplies, and what no reading established. All ten products rate yes on Harness, Retrieval and Tools. Five rate yes on Approvals: Claude Code, Codex CLI, Grok Bot, Muse and Claude Managed Agents. Version 0.13 defines Approvals as obtaining and recording people's authorizations, both approvals given in advance and decisions on single actions that are held. Nine first readers rated yes. The final review found that for five of the nine (Grok Build, Antigravity, Claude Duet, Dots and Tasks in Gemini) the reader's row states how an approval in advance is saved and states no record of a decision on a single held action, and that the reading on H1 (Section 5.6) found no such record either. Those five cells now show part (Table 7a). For the other four the row or the H1 reading states such a record: an event logged for each permission decision in Claude Code and in Codex CLI, a recording of actions that keeps the decision on each in Grok Bot, and an activity log that lists the permissions a person has given in Muse [55, 56, 61, 63]. For the tenth, Claude Managed Agents, the value was decided. Its first reader rated part, because approval in advance is a tool's `always_allow` policy with no stated conditions and no record of who approved. A second reader, asked for this one cell, rated yes, and the decision is yes: the pages state settings that authorize a tool or a set of tools, versions of each configuration, and kept events for each confirmation that a person allows or denies [69]. Keeping those events past a restart requires a configured database. The final reviewer agreed with the decision on that evidence. It did not accept the reason that the eighth draft gave, agreement with four products on which both readers had rated a like feature yes, and two of those four cells are among the five now lowered. These ratings use the version 0.13 definitions (Section 3.5). Version 0.14 adds to Approvals a record of each decision with the reason a person was needed. On that question (H1, Section 5.6) one of the ten received a full rating, four a partial one and five none, and one of the four, Grok Bot, has a full rating as its recorded alternative. No reading established a full rating on Directory, Identity, Policy, Evaluation, Improvement or Releases. On Policy four first readers had rated yes and on Identity two, and each of those values was lowered by a decision or by a later change (below). Claude Code alone rates yes on Tasks and on Artifacts, and two products rate yes on Memory (Codex CLI and Claude Managed Agents). A value below yes says that the pages

read did not state the whole of a definition, and it does not show that the product lacks the feature (Section 7). The six services with no full rating are the Directory, Identity, Policy and three of the model's Operations services. Under version 0.13, both raters of the reference implementation judged Evaluation, Improvement and Releases present there (Section 5.4). Two of the six definitions follow that system's own conventions, which Section 7 discusses. In that system they are its own components and not features of its harness, which rates in part on all three.

Criterion	Claude Code	Codex CLI	Grok Build	Anti-gravity	Claude Duet	Dots	Grok Bot	Tasks in Gemini	Muse	Claude Managed Agents
C1. An area of work exists	part*†	part*†	part*†	part*†	no*†	part*†	part*‡	no*†	no*†	part†
C2. The area is specified apart from who does the work	part*†	part†	part†	part*‡	no*†	part†	part†	no*†	no*†	part†
C3. A record of accountability	part†	part†	part†	part†	part‡	part†	part†	part‡	part‡	part‡
C4. Finding where to send work	part*†	part†	part†	part*†	no*†	part†	yes‡	no*†	part†	part†

Table 8. The ten products on the four criteria of Section 3.3, with the marks of Table 7. Every criterion cell had two readers.

Areas of work. No product rates yes on C1, C2 or C3 (Table 8). Seven rate part on C1. Their units are a project defined as a conversation, a folder or a set of files, a workspace or a deployment that is the whole system, a team built for one task, an agent with its listed subagents, or a team or group defined for sharing and settings. Three of the personal agents have no such unit. All ten rate part on C3: each keeps a record that names an owner, a creator or a provider for each agent, plugin, connector, skill or other resource, and none names one accountable area for each responsibility. One product rates yes on C4. In Grok Bot a person can ask any of their Bots whether a shared Bot exists for a kind of work; the Bot "searches the Team Bots you can use by what they do", and a Team Bot "can suggest other team Bots that fit a request and link to a chat with one" [61]. The lookup covers Bots shared within a team, on the product's team and enterprise plans, and among a person's own Bots the routing is inside one coordinating Bot. The derived Directory value is in part for all ten products.

Agreement between the readers. The second reader gave the first reader's value in 65 of the 80 sampled service cells and in 33 of the 40 criterion cells, 82% of the 120. They also differed on the one cell that was second-read by choice (Section 3.5). All 23 differences are one step apart, and in 22 of them one of the two readers had marked the cell disputable. The decision kept the first reader's value in thirteen cells and the second reader's in ten. The lower of the two values stands in seventeen and the higher in six. Six full ratings by first readers became partial (Codex CLI on Policy, Claude Duet on Sessions, Dots on Environments, Identity and Policy, and Claude Managed Agents on Environments), and four cells rose, two of them to a full rating: Grok Bot's C4 and Claude Managed Agents' Approvals. No first reader had given a full rating on any of the four criteria. The decision with the strongest case for the other value was Grok Bot's C1: its teams and groups hold Bots and exist side by side, and its pages define them by who may use a Bot and which settings apply. At least one reader marked 83 of the 240 rated service cells and 23 of the 40 criterion cells disputable. After the decisions, 81 service cells and 18 criterion cells carry an asterisk in Tables 7 and 8. A decision removed the mark from one service cell and five criterion cells, two marked service cells are among the thirteen that were changed and carry §, and the adjudicator added a mark to one cell that neither reader had marked (Grok Build on Improvement).

Consistency across the other cells. The decisions on the differing cells applied general rulings, for example that a rule on which files and network addresses a command may reach is not a rule on content, and that a unit of work with no stated retry and recovery rates part on Tasks. After the sixth draft a further reader applied those rulings to every cell of the services they govern, from the reader files alone, for the nine products then in the table, and another reader did the same for the tenth product's column. The author decided that a cell which does not follow a ruling is changed to follow it. Six cells were changed on those two checks. Four yes values became part: Dots on Tasks and on Orchestration, and Claude Code and Grok Build on Policy. One no became part, Tasks in Gemini on Improvement, and one part became yes, Claude Managed Agents on Models. The final review of the eighth draft checked each of the six against its reader's row and found that each conflicts with its ruling. It found seven more cells, which this draft changes: the five Approvals cells described above, Tasks in Gemini on Environments, whose row states no isolation of an agent from others, and Codex CLI on Identity, whose row says that subagents have no identity of their own. Both readers had rated the last of these yes, and with the change Identity has no full rating. All thirteen cells carry the mark §, and Table 7a lists them with their values as read. Before the changes the counts were 116 yes, 116 part and 18 no; Claude Code rated yes on 19 services, Codex CLI, Grok Build, Antigravity and Dots on 13 each and Claude Managed Agents on 10. The thirteen are recodings of evidence that the readers had already recorded. No vendor page was read again for them, each rests on the judgment of one checker or reviewer, and the checkers had been given the decisions, so their agreement with a decision does not confirm it. Five cells that the checks called unclear were left as read: four yes values that could be part (Grok Build on Orchestration, and Tasks in Gemini, Grok Bot and Claude Managed Agents on Sessions) and one part that could be yes (Claude Managed Agents on Tasks). With those also changed the counts would be 103, 130 and 17.

The changes do not make the table consistent across its ten columns. The checks and the final review recorded like cases that no ruling settles. Dots rates yes on Scheduling with no stated limit on what runs at once, and Muse rates part with that omission as the reason. Antigravity rates part on Secrets because the protection covers only its remote agents, and Claude Code rates yes for a feature of one surface. Antigravity rates part on Reliability with restarts of its service documented, and Claude Code rates yes for restarts by a supervisor. The decision on isolation between agents that share a session was applied to Dots, Claude Managed Agents and Tasks in Gemini and was not examined again in the other columns. Two reviewers disagreed with decisions that the author kept. The reviewer of the sixth draft disagreed with three of the 19 made by then (Section 3.5): on Dots' Evaluation it would rate part where the decision is no, and on Codex CLI's Policy and on Tasks in Gemini's Models it found the decided value lower than a like case that kept yes. The like cases on Policy have since been lowered. The final reviewer disagreed with one of the four decisions on the tenth product and again with two of the earlier three. On Channels for Claude Managed Agents it would rate yes: the product gives people a console and a terminal and gives applications an interface, and the definition names no channel that people must already use. The decision reads the definition's closing words, "on the channel each one uses", as asking for that, and it gives the lower of two defensible values. On Dots' Evaluation the final reviewer would also rate part. On Tasks in Gemini's Models it found no support for separating a model that a person picks in a menu from one set in a configuration, and the change to Claude Managed Agents on Models makes that difference plainer. With the final reviewer's three values the counts would be 108, 126 and 16.

PRODUCT	SERVICE	AS READ	SHOWN	READERS	REASON
Dots	Tasks	yes	part	one	Retry and recovery of the tracked unit are not described (as decided for Codex CLI).
Claude Code	Policy	yes	part	one	The only content rule is on files and network addresses (as decided for Codex CLI).
Grok Build	Policy	yes	part	one	The only content rule is on files and network addresses (as decided for Codex CLI).
Dots	Orchestration	yes	part	one	Dependencies between tasks are not described (as decided for Grok Bot).
Tasks in Gemini	Improvement	no*	part	two	A skill built from chats on request is the feature rated part for Grok Build.
Claude Managed Agents	Models	part*	yes	one	A model set for each agent in its configuration is the feature rated yes for Codex CLI.
Dots	Approvals	yes	part	one	No record of a decision on a single held action is stated.
Claude Duet	Approvals	yes	part	one	No record of a decision on a single held action is stated.
Grok Build	Approvals	yes	part	one	No record of a decision on a single held action is stated.
Antigravity	Approvals	yes	part	two	No record of a decision on a single held action is stated.
Tasks in Gemini	Approvals	yes	part	two	No record of a decision on a single held action is stated.
Tasks in Gemini	Environments	yes	part	one	Isolation of an agent from others is not stated (as decided for Dots and Claude Managed Agents).
Codex CLI	Identity	yes	part	two	Subagents have no identity of their own (the reading by agent decided for Dots).

Table 7a. The thirteen cells of Table 7 that were changed after the readings (§). Readers is the number of readers that rated the cell; where two did, both gave the value as read. Each change rests on the reader's own row and on no new reading.

5.6 What the sources and products do with people's decisions

One of the 40 sources and products received a full rating for a review of past decisions that changes the rules, and it is Charterion [18]. Its evidence records return to design, where they "drive three kinds of change": narrowing a charter that produces many denials or reversals, widening one that leaves needs unserved, and changes of level. A charter's level "shall be raised only at gate G6, only by one step", when thresholds are met on four rates, among them "the acceptance rate of its escalations by the overseer", and on a condition about evidence that is not attributed to an agent. A raise to the autonomous level has further requirements on oversight and escalation. Charterion is also the only one with a full rating for authority that follows an agent's record (H4). Fourteen others have a partial rating for a review (H3) and twenty-five have none. The eight sources of 2002 to 2013 have none of the five; they model organizations of agents and do not treat people's decisions.

How firm these counts are. Each reader gave the lower of two defensible values, so a full rating in this section is one given under that rule. The final reviewer counted, from the readers' files, the full ratings that each question would have if every recorded alternative of yes were taken: 11 and not 4 on H1, 8 and not 4 on H2, 4 and not 1 on H3, 8 and not 1 on H4, and 2 and not 0 on H5. These are bounds and not findings. On H3 the three further candidates are EDDOps, Grok Bot and Muse, and on H5 the two are Charterion and Bedrock AgentCore. The reviewer also found that the readers did not apply the questions alike. One reader gave Grok Bot part on H1 because its approval cards, which state why an action paused and keep the outcome, are elements of a chat and not a defined log, where the question asks only for a record. One gave Claude Code part on H2 because its counter of permission decisions covers editing tools only, while another gave Bedrock AgentCore yes for a count of browser takeovers alone; by the question as written, Claude Code's counter would rate yes. One gave Codex CLI part on H2 partly because its counter of approval requests defines no rate, and the question admits a count. One rated three harnesses no on H4 because their limit on repeated denials holds within a session or a turn, while another rated a like mechanism in LangGraph part; the question sets no minimum period. No reading was repeated for this draft, so the values stand as read, and the statements below that one source or no product has a feature are statements about full ratings under the lower-value rule.

Records and measures. Three sources and one product received a full rating for a record of each decision with the reason a person was needed: Charterion, AOS-0001, which states that "A human decision is itself evidence" [22], OpenClaw, whose record of an approval carries a reason code [34], and Claude Managed Agents. In that product the event for a tool call records the outcome of its permission check with "an `evaluation` object whose `type` names the policy that produced that outcome", and the person's answer is a further event [69]. Its reader noted that no page says the record names the person who decided, and that without a configured database, which is the default, a restart erases the history. That reader was a model of the product's own vendor and read it a day after the other readings. No reader obtained the release, which is supplied on request, so the value describes behavior that the documentation claims and not the behavior of an identified version. Four received a full rating for a measure of how often people are required: AAM/5C's rate of "agentic tasks completed without human intervention" [14], Charterion's escalation rate, a count of browser takeovers in Bedrock AgentCore [4], and takeovers for each pull request in Beyond the Agent Harness [45].

The ten products. All ten rate in part on a review and none in full. Claude Code has a command that builds a list of allowed actions from past sessions, and Codex advises a user to "analyze past traffic there before changing policy or permissions" [55, 56]. A Grok Bot "can offer to add, change, or remove one of your Auto-review rules on a card you confirm" [61]. Claude Duet's vendor writes that "Approval fatigue is a serious issue" [59]. In Claude Managed Agents a builder can set a tool so that it "executes automatically with no confirmation", and no page describes a review of past approvals that changes such a setting [69]. None of the ten received a full rating for a measure of how often a person is asked. Claude Code and Codex CLI each define a counter of permission or approval decisions and rate part, with yes as the recorded alternative (above). None received a full rating for authority that widens from an agent's record. Two cells are disputable on that point: Meta states that Muse learns over time which decisions need the user's approval, and Antigravity's vendor writes that permissions granted during a conversation "can persist, allowing the agent to learn trusted actions over time" [58, 63]. Neither describes a mechanism that uses the agent's record. Three harnesses stop or escalate after repeated denials: Claude Code's automatic mode "pauses" after three blocked actions in a row, Codex "interrupts the turn after 3 consecutive denials", and Grok Build escalates "only after repeated denials" [55, 56, 57]. The reader rated all three no, with yes as the disputable alternative, because the limit holds within a session or a turn and is not authority set from a record. In at least four of the ten a model decides at run time whether an action needs a person.

Tasks handed to a person. No source or product received a full rating on H5, and twelve received a partial one. Products describe the hand-off: Antigravity "leaves the login page open in the built-in browser tab, pauses, and asks you to finish signing in" [58], and Dots separates an action that needs approval from "a step you must do yourself" [60]. Two treat it as something to reduce. Grok Bot offers a connector when it hands over a sign-in, "and connecting ends the handoff", and Google lists "Asking you to take over when it can normally do the step" among the mistakes its agent can make [61, 62]. None records hand-offs as a class and works from that record.

Outside the sources. The search found no source that rates in full on a review. One paper states its purpose without a process: "repeated human approval of similar actions may indicate that certain decisions can be safely automated" [64]. One vendor post defines a working rule for authority, in which "the rolling rate at which operators reject proposed actions caps the effective safety metric", promotion "requires sustained performance" and "Demotion is immediate" [65]. The search also pointed to an older literature on adjustable autonomy, of which two works were read. One describes a deployed system of personal assistant agents. At each decision the agent recorded the user's choice, it "also asked if the user wanted such decisions taken autonomously in the future", and from the answers it learned "its rules for transferring control" [66]. That is a review of each decision that changed the rules. Its authors report that the agent "over-generalized from training examples", in one case canceling a meeting incorrectly, and they moved to a different design. The rest of that literature was not read.

How the model's rule differs from its nearest counterpart. The review of stops is not new. Its nearest counterpart, Charterion, differs from it in two ways. Charterion changes a charter's scope from its counts of denials, reversals and unserved needs, and its level from rates over a window, and a person may lower a level at any time, whatever the record. The model reviews each stop with its reason. Charterion concerns authority, where the model also counts the tasks handed to a person for lack of capability, on which Charterion rates in part. Neither difference is new outside the 40. The system of 2002 described above reviewed each decision [66]. For tasks handed to a person, site reliability engineering defines as toil the manual work that "a machine could accomplish" as well as a person, measures it, and reserves at least half of each engineer's time for work that reduces it or adds features [67]. Whether either difference matters in practice is not known: neither system has reported running its review. Charterion's specification states that the framework has not had a naturalistic evaluation and that its service was exercised on hand-built and synthetic models [18].

5.7 What the exercises changed

Version 0.10 had 22 services. Version 0.13 has 25, and the responsibilities of several others have been restated. Version 0.14 keeps the 25 and adds the review of stops for people (Section 4.5), at the author's direction and not from an exercise. The first round of the comparison with published models changed the model in fifteen places and the second in eleven more. Appendix B lists the larger changes with the finding behind each (Table B1), the decisions that were the author's judgment, and three things that the sources have and the model still does not place. Early versions of the model also made five claims about how it differs from prior work. The comparison withdrew all five and kept narrower observations (Appendix C). Two bounded findings of Section 5.3 remain only with its qualification. Under the stricter reading of disputed cells, none of the seventeen models and platforms received a full rating for specifying an area of work apart from who does it, and no source received one for a record of one accountable area for each responsibility; on the other reading, two and three would. Forty sources that the search found have not been mapped, and failing to find a model does not show that none exists.

6 Discussion

The review of stops for people is the part of the model most tied to its aim, and the least tested. The model is written for systems in which agents finish work end to end. Most of the sources and products compared here describe how to hold an action for a person and how to approve a class of actions in advance, and fifteen of the 40 also describe, in full or in part, a review of past decisions (Section 5.6). The model's rule concerns what happens afterward: each stop for a person, whether a decision or a task that an agent handed over, is recorded with the reason a person was needed, and reviewed to ask whether the stop needed a person, whether a decision could have been made in advance, and what change would remove it (Section 4.5). The paper has no evidence that the rule works. The reference implementation records such stops by hand and in a log of its gates. Its instructions call for a weekly reading of the tasks handed to a person; decisions are not reviewed, and no review runs as a loop. The rule is not new: Charterion defines a review of an agent's record that changes its authority, and every vendor product read has part of one (Section 5.6). The model's version reviews each stop with its reason and counts tasks handed over for lack of capability. It is stated first in this discussion because the refactoring of the reference implementation will test it first.

Models may assess actions and review recorded stops. Two parts of this account depend on something that systems built on rules alone did not have. The first is judgment at the gate. A rule decides the cases its author listed, and everything else waits for a person. In at least four of the ten products a model now decides at run time whether an action may proceed or needs a person (Section 5.6), and in the model's terms a condition of a pre-approval may be such a judgment. The second is judgment in the review. The counterparts found reduce people's decisions to rates over a window, as Charterion and the vendor post on graduated autonomy do, or leave the review to the user, as the harnesses mostly do. A review of every stop means reading each one with its reason and what the person did, and proposing the rule, tool or instruction that would remove it. The review runs after the work, so it holds nothing up, and a stop like one reviewed recently is counted and not reviewed again (Section 4.5), so the number of reviews should stay small. The author's hypothesis is that a language model can do this reading across stops of many kinds, so that the person is not asked a second time about each stop. The paper has not tested the hypothesis: no source read reports a model reviewing every stop, and the reference implementation has not run such a review. Two harnesses have a model propose a rule in narrower cases: Codex "may propose" a rule when an action is escalated, and a Claude Code command scans past sessions for read-only calls and adds them to a list of allowed actions [55, 56].

The domain layer restates an older idea for a new kind of agent. It carries the organizational idea of Section 2.4 into a model of systems built on language models, beside a list of platform services. Joining the two is not new either: JaCaMo, and ISLANDER with AMELI, joined an organizational specification to shared infrastructure for the agents of their time [51, 52, 53]. What the model adds is narrower. Its list of services is written for agents built on language models, and each responsibility has one accountable area in a directory. The older literature offers considerable material that WARM has not yet used, including formal links between roles and the work they must or may take on, rules that hold for a whole organization, and entry to a unit as a governed act, which AMELI enforces while the system runs [52].

Who supplies the platform may change. The model names responsibilities and does not say who supplies them. A harness product is itself a platform in the model's sense. Claude Code, on which the reference implementation runs, supplies the loop and also sessions, task agents, scheduled jobs, connections to tools, permission rules and messages between sessions. The author expects the vendors of harnesses and models to supply more of the platform services over time, and to supply agents packaged for an area of work, as coding agents already are. For

each service a builder then chooses among adopting the vendor's, keeping their own, and building one, and the choice will be made again as the products change. A reading of ten vendor products, first made on 12 November 2026, fits this account (Section 5.5). Under the version 0.13 definitions every one rates in full on the loop, retrieval and tools, and five on approvals. The harnesses rate in full on 11 to 18 of the 25 services. The readings did not establish full coverage of six definitions in any product: a directory, identity, policy, evaluation, improvement and releases. A builder who needs one of those in full may need more documentation from the vendor or an addition of their own. The reading will soon be out of date: one of the ten was renamed between its launch and the reading, and one launched on the day of the first reading. The paper does not measure how much of each service the harness supplies in the reference implementation.

Governance is well described in the sources, and the mapping found governance gaps in the reference implementation. Policy has a full counterpart in sixteen of seventeen sources and Identity in thirteen. The comparison supplied the rules of Section 4.4, and the mapping showed which of them the reference implementation does not meet (Table 6).

Misfits and spans prompted revisions to the model. Two of the three services added since version 0.10, Improvement and Lifecycle, began as components that fit nowhere.

7 Limitations

1. **One system, one harness, one owner.** The model is drawn largely from one system (LifeOS), built on one harness product (Claude Code), by one person who is its operator and only user. Its domains are the ones that person needed, and the two raters of the system did not agree on how many it has. Other kinds of domain, larger organizations and other harnesses may not fit.
2. **No evidence of benefit.** Nothing here shows that organizing a system by the model improves it. The refactoring of the reference implementation toward the model is the first test. It has begun, and its measures have not yet been set.
3. **The raters were not independent of the author.** Every reader, mapper and second rater was an AI agent working for the author, most were instances of one model family, and the mappers and second raters knew the model. Twenty-two of the 30 sources were rated on the four criteria once, by one of two readers who did not apply the criteria identically. Two further readers rated the other eight under clarified criteria and gave the same value in 27 of 32 cells (Sections 3.3 and 5.3). Thirty-two of the 120 cells are disputable. The ten vendor products had one reader each and a second reader on a sample (Section 3.5). Agreement between raters does not show that the model is intuitive to an outsider, shared errors are possible, and nobody outside the author's system has mapped a system to the model.
4. **Formative, with no held-out check.** The sources that changed the model are those against which its coverage is reported. The platforms, the published models and the reference implementation were assessed against versions 0.10, 0.11 and 0.13. For the platforms, the Channels column still reflects the earlier, narrower service.
5. **Documentation and notes, not running systems.** The platforms were mapped from their documentation on one date, and the reference implementation from file headers and registries without a reading of its code. The grid was filled from mapping notes by several readers with somewhat different conventions. Its four right-hand columns rest on the readers' prose findings and are weaker than the service columns, and for the three vendor guidance sources the cell for agent identity rests on the existence of an identity service and is doubtful.

6. **Samples are small, and inventories are not a common unit.** The second raters sampled 36 of 1,301 platform assignments, 126 published-model assignments and 60 of 849 reference-implementation assignments, and the rates for individual strata have wide intervals. The second rating of OpenClaw excluded rows that group classes of components, and the second rater of the reference implementation saw summary counts and some assignment information before completing the assessment.
7. **The search is incomplete.** The documented search screened only the first page of results for each query, forty of the 48 new sources it found are unmapped, and eight could not be opened. The search excluded single-domain architectures and later added an exclusion of subsystems while keeping sources scoped to a harness, choices that could have removed evidence about responsibilities, organizational units or accountability. The organizational multi-agent literature was read only after a reviewer named it. The books on Domain-Driven Design, Team Topologies and Data Mesh were not read, one book that appears to apply mesh ideas to agents, *Agentic Mesh* (Broda and Broda, 2026), could not be obtained, and the authors of the published models have not been asked whether their work is represented correctly.
8. **The model names responsibilities and not contracts** (Section 4.6). It gives no way to tell that a system implements a service correctly.
9. **Several parts are untested proposals.** The rule that every stop for a person is recorded and reviewed entered the model in version 0.14 on 12 November 2026, after the mappings, the comparisons and the first readings of the vendor products and before the reading for counterparts (Section 3.6); the reference implementation does not yet carry it out. Two sources outside the comparison are evidence against one outcome of the review, deciding a class of actions in advance. In the system of 2002 the learned rule over-generalized [66], and a 2026 study of coding agents reports that agents obtained the saved rules they sought "in 58% of experimental instances" [68]. In the model a person decides each proposed pre-approval, which limits that risk and does not remove it. The reading for counterparts (Section 5.6) had one reader for each source and no second reader, 21 of the 70 cells for the fourteen published models are disputable, and of the literature on adjustable autonomy, to which the search pointed, two works were read. The published skill has no implementation in the reference system. The domain pattern rests on an inventory of one system, in which the factory pattern occurs once. The directory was written during this work, and the subdomain rests on one decided case. The two rules on pre-approval come from the author's experience of one system, and none of that system's pre-approvals stated a term when the mapping was made.
10. **The model is not stable.** It changed in each of fourteen versions over four days, and the division into working and cross-cutting groups describes scope and does not sort every service.
11. **The vendor mapping is a dated reading of documentation.** Ten products chosen by the author were first read from their vendors' pages on 12 November 2026, and no product was run to check a value. Second readings, the tenth product's further readings and the decisions used the pages as they stood on 13 November. A value says what the pages state: a feature may exist undocumented, and a documented one may not work as described. A feature in preview, in beta or limited to a paid plan counts as supplied when a user can use it. Each product had one reader of one model family. A reader of another family rated a third of the service cells and all the criterion cells, and the two differed on 22 of 120, and on one further cell that was second-read by choice. One adjudicator, the agent that wrote the model, decided those cells and knew which reader had given which value. Only the cells on which the readers differed were decided, so like cases among the other cells were rated unevenly: two checks and the final review found thirteen cells that do not follow the rulings behind the decisions or the wording of a definition, and five that are unclear. The thirteen were changed, each on the judgment of one checker or reviewer and without a new reading of the vendor's pages, and the five were left as

read (Section 5.5, Table 7a). The changes are recodings of recorded evidence. They do not show that the table is consistent across its ten columns: like cases remain that no ruling settles, and two reviewers disagree with four of the 23 decisions, which the author kept. The final review also found the questions of Section 5.6 unevenly applied, and those values were left as read. One vendor's main sites refused the reader, which read seven of its pages from archived copies, and that product has the lowest agreement (four of eight service cells). One product has no documentation section. Most of one product's column rests on pages of a site whose robots file disallows the first reader's crawler by name and whose notice prohibits automated collection without written permission; the reader fetched those pages once each with a browser's request headers (Section 3.5). The criteria of Section 3.3 were written for published models and were applied to products unchanged. Two of the six services on which no product rates in full have definitions that follow the reference implementation. Evaluation requires a report of "unknown" when a check could not run, which is that system's own convention and decides the value for at least one harness. The Directory is derived from the model's criterion of one accountable area for each responsibility. The contrast with the reference implementation therefore measures in part how closely the definitions follow it. All ten first readers were one vendor's model (Section 3.5). The products were rated against version 0.13, and Table 2 gives the version 0.14 definitions of Approvals and Improvement; under those, one product received a full rating for the added record of decisions (H1, Section 5.6), and the reference implementation's Improvement would be partial, since it does not carry out the rule. The tenth product, a self-hosted agent platform from one of the five vendors, launched on the day of the reading and is supplied on request. It was read after the other nine, and its four differing cells were decided after the review of the sixth draft. One of them, Approvals, was second-read because its value decided a claim of this paper, and the decision raised it to a full rating; the same scrutiny was not given to the like cells of other products until the final review, which lowered five of them. Its first reading of the services and its only reading on H1 to H5 were made by its own vendor's model, and no reader obtained the release or an identified deployed version. Its values therefore describe documented behavior claimed for a release with restricted access. Its inclusion changed claims of this paper in both directions.

12. **Work not done.** An evidence supplement accompanies the paper in its repository, in the folder `supplement`. It holds the frozen criteria, the readers' files with their quotations and rating reasons, the search record, the second readings and the decisions on the cells where readers differed, and a script that recomputes the counts in the tables. It does not hold the source texts, the quotations from one vendor's site that carries a notice against automated collection, or the names and rating reasons for the components of the reference implementation, which is a private system. For that system a reader can recompute the counts and cannot check the judgments. JaCaMo, ISLANDER with AMELI, and FIPA's agent management specification were read only after the fourth draft, and no search was made for other platforms of that literature. FIPA's specification was read from an archived copy, and an extension of JaCaMo's organization model for accountability, named on the project's site, was not read. The vendor products were mapped after the fifth draft. Sixteen of each product's 24 rated services, fifteen for the tenth, have one reader, the first readers' quotations were checked only for the cells that were decided afterward and for one product whose reader checked its own, and no vendor has been asked whether its product is represented correctly. The placement test of Beyond the Agent Harness [45] was not compared in detail with the model's platform test. The final review of the eighth draft was applied once, and its changes have not been reviewed.

8 Conclusion

The comparison found precedents for every platform service and corrected the model in more than twenty places, so the platform side of WARM is mostly a consolidation of parts that are widely named. The domain side was established by the organizational multi-agent literature two decades ago and by the organization of human work before that, and platforms of that literature had already joined an organizational specification to shared infrastructure. Under the stricter reading of disputed cells, no source received a full rating for a record of which area is accountable for each responsibility, and on the other reading three would. A reading of ten vendor products, begun on 12 November 2026, shows the same pattern in products: each rates in full on the loop, retrieval and tools as version 0.13 defines them and five of the ten on approvals, no reading established a full rating on a directory, identity, policy, evaluation, improvement or releases, and one lets an agent find a shared agent by the kind of work it does. So far 688 of the 849 inventory entries of a system that had no shared description have been assigned to one service or one layer, and the mapping identified gaps in that system's governance. The refactoring of the reference implementation toward the model has begun, with measures that have not yet been set, among them how often a person is required. The paper is a dated map, and its method is written down so that the map can be drawn again.

R References

1. MacKenzie, C. M., Laskey, K., McCabe, F., Brown, P. F. and Metz, R. (eds.). *Reference Model for Service Oriented Architecture 1.0*. OASIS Standard, 12 October 2006. docs.oasis-open.org
2. Zimmermann, H. OSI Reference Model: The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications* 28(4), 425-432, 1980.
3. Lu, Q. et al. Towards Responsible Generative AI: A Reference Architecture for Designing Foundation Model based Agents. arXiv:2311.13148
4. Amazon Web Services. Amazon Bedrock AgentCore, developer guide. Read 9 and 11 November 2026. docs.aws.amazon.com
5. Anthropic. Claude Managed Agents, overview. platform.claude.com
6. Google. Gemini Enterprise Agent Platform, overview. Read 9 November 2026. docs.cloud.google.com
7. Microsoft. Foundry Agent Service, overview. 26 October 2026. learn.microsoft.com
8. Trivedy, V. The Anatomy of an Agent Harness. LangChain, 10 March 2026. langchain.com
9. Model Context Protocol. Architecture overview. modelcontextprotocol.io
10. A2A Protocol. Specification, version 1.0.0. a2a-protocol.org
11. National Institute of Standards and Technology. Announcing the AI Agent Standards Initiative. 17 February 2026. nist.gov
12. Yu, Rabhi, Zhu and Lu. A Reference Architecture for Integrating Agentic AI into Enterprise Software. October 2026. arXiv:2610.14818
13. Zhu. Fluid Structure, Rigid Record. August 2026. arXiv:2608.08516

14. Goujon. Agentic Architecture of Information Systems. The AAM/5C Model: A Reference Model for Distributed Autonomy. October 2026. arXiv:2610.30221

15. Microsoft. Multi-Agent Reference Architecture. github.com/microsoft/multi-agent-reference-architecture

16. Amazon Web Services. AWS Prescriptive Guidance on agentic AI (six guides). Read 11 November 2026.

17. Google Cloud. Architecture Center, agentic AI pages (17 pages). Read 11 November 2026.

18. El Hassani. Charterion v2.0. Zenodo, September 2026. doi:10.5281/zenodo.22829848

19. Longmire, J. D. and Longmire, M. AEON: An Enterprise Control Plane Architecture for the Agentic Era. Zenodo, 22 May 2026. zenodo.org/records/20349596

20. deVadoss. CEAD. May 2026. arXiv:2605.08258

21. Salesforce Architects. The Agentic Enterprise: The IT Architecture for the AI-Powered Future. Undated. architect.salesforce.com

22. Sharma and Shah. AOS-0001. August 2026. arXiv:2608.03214. With its precursor AOS-0000, arXiv:2606.01508

23. Steinder and Franke. Towards an Agent Operating System: Lessons from Classical and Cloud OS. July 2026. arXiv:2607.25076

24. Beheshti et al. The ASOC manifesto. July 2026. arXiv:2607.12619

25. Venkiteela, P. An Enterprise Agentic Architecture Framework for Agentic AI Governance and Scalable Autonomy (EAAF). *Scientific Journal of Computer Science* 2(1), 2026. doi:10.64539/sjcs.v2i1.2026.368

26. Dehghani, Z. How to Move Beyond a Monolithic Data Lake to a Distributed Data Mesh. 2019. martinfowler.com. And Data Mesh Principles and Logical Architecture. 2020. martinfowler.com

27. Skelton, M. and Pais, M. Team Topologies: key concepts. teamtopologies.com. And the Team API template, github.com/TeamTopologies/Team-API-template

28. Evans, E. *Domain-Driven Design Reference: Definitions and Pattern Summaries.* 2015. domainlanguage.com

29. Levin, R., Cohen, E., Corwin, W., Pollack, F. and Wulf, W. Policy/Mechanism Separation in Hydra. *Proceedings of the 5th Symposium on Operating Systems Principles*, 132-140, 1975.

30. Dubrov. Domain-Driven Design for multi-agent systems. dinacode.com

31. Greenfield, J. and Short, K., with Cook, S. and Kent, S. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools.* Wiley, 2004.

32. Remy, D. Agent Scrum. Working paper, 2026. doi:10.5281/zenodo.23771170

33. LangChain. LangGraph, LangSmith and LangSmith Deployment documentation. Read 11 November 2026. docs.langchain.com

34. OpenClaw. Documentation. Read 11 November 2026. docs.openclaw.ai

35. Remy, D. Persistent Agents on Claude Code. Working paper, 2026. doi:10.5281/zenodo.23787644

36. Remy, D. Working Agents: a living research program. 2026. doi:10.5281/zenodo.24023471

37. Team Topologies. Team Topologies for the AI age: stewardship of value flow. teamtopologies.com

38. Wulveryck, O. Who does what? Team Topologies for the agentic platform. 22 June 2026. blog.owulveryck.info

39. Ferber, J., Gutknecht, O. and Michel, F. From Agents to Organizations: An Organizational View of Multi-agent Systems. In *Agent-Oriented Software Engineering IV*, LNCS 2935, 214-230. Springer, 2004. author copy

40. Zambonelli, F., Jennings, N. R. and Wooldridge, M. Developing Multiagent Systems: The Gaia Methodology. *ACM Transactions on Software Engineering and Methodology* 12(3), 317-370, 2003. author copy

41. Hübner, J. F., Sichman, J. S. and Boissier, O. A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems. SBIA 2002. author copy

42. Dignum, V., Dignum, F. and Meyer, J.-J. An agent-mediated approach to the support of knowledge sharing in organizations. *The Knowledge Engineering Review* 19(2), 147-174, 2004. doi:10.1017/S0269888904000244

43. Horling, B. and Lesser, V. A Survey of Multi-Agent Organizational Paradigms. *The Knowledge Engineering Review* 19(4), 281-316, 2004. doi:10.1017/S0269888905000317

44. Microsoft. Cloud Adoption Framework for Azure, AI agents: Process to build agents across your organization. 1 December 2025. learn.microsoft.com

45. Hua, L. Beyond the Agent Harness: Boundary Criteria for Software-Factory Orchestration. A Single-Operator Case Study. November 2026. arXiv:2611.06946

46. Li, G. et al. WorkMate: A Human-AI Collaborative Agent Infrastructure for Enterprise Organizational Governance. Zenodo, June 2026. doi:10.5281/zenodo.20644795

47. Kandogan, E. et al. Orchestrating Agents and Data for Enterprise: A Blueprint Architecture for Compound AI. April 2025. arXiv:2504.08148

48. Xia, B. et al. Evaluation-Driven Development and Operations of LLM Agents: A Process Model and Reference Architecture. arXiv:2411.13768

49. Gürcan, Ö. Modélisation à base d'Agents Augmentés par LLM pour les Simulations Sociales: Défis et Opportunités. 27 August 2024. In French. arXiv:2409.00100

50. Foundation for Intelligent Physical Agents. FIPA Agent Management Specification. SC00023K, 2004. Read from an archived copy of fipa.org.

51. Esteva, M., de la Cruz, D. and Sierra, C. ISLANDER: an electronic institutions editor. AAMAS 2002, 1045-1052. doi:10.1145/545056.545069

52. Esteva, M., Rosell, B., Rodríguez-Aguilar, J. A. and Arcos, J. L. AMELI: An Agent-based Middleware for Electronic Institutions. AAMAS 2004, 236-243. author copy

53. Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A. and Santi, A. Multi-agent oriented programming with JaCaMo. *Science of Computer Programming* 78(6), 747-761, 2013. doi:10.1016/j.scico.2011.10.004

54. MOISE project. ORA4MAS organisational artifacts: SchemeBoard and GroupBoard. API documentation, read 12 November 2026. moise-lang.github.io

55. Anthropic. Claude Code documentation. Version 2.1.322 at the reading of the services and 2.1.323 at the reading on H1 to H5. Read 12 November 2026. code.claude.com/docs

56. OpenAI. Codex documentation. Codex CLI 0.167.2. Read 12 November 2026. learn.chatgpt.com/docs

57. xAI. Grok Build documentation and changelog. Changelog entry v1.0.77. Read 12 November 2026. docs.x.ai/build

58. Google. Antigravity documentation. App v2.31.2, CLI v1.5.7. Read 12 November 2026. antigravity.google/docs

59. Anthropic. Claude Duet: product page, help articles and the article on safety, privacy and security. Research preview. Read 12 November 2026. claude.com/product/claude-duet

60. OpenAI. Dots documentation. No version number is published. Read 12 November 2026, in part from archived copies. learn.chatgpt.com/docs/dots

61. xAI. Grok Bot documentation and changelog. Desktop v1.0.1. Read 12 November 2026. docs.x.ai/grok-bot

62. Google. Tasks in Gemini (formerly Gemini Spark): Gemini Apps Help. No version number is published. Read 12 November 2026. support.google.com/gemini

63. Meta. Muse: product site and its posts on security and on design (muse.ai), help center and newsroom posts. iOS app 15.1. Read 12 November 2026. meta.com/help/superintelligence

64. Cheng and Cheng. A Decoupled Human-in-the-Loop System for Controlled Autonomy in Agentic Workflows. 2026. arXiv:2604.23049

65. Amazon Web Services. Closing the AI agent trust gap with graduated autonomy. AWS Architecture Blog, 2026. Read 12 November 2026. aws.amazon.com

66. Scerri, Pynadath and Tambe. Towards Adjustable Autonomy for the Real World. Journal of Artificial Intelligence Research 17, 2002, pp. 171-228. Section 2.2. arXiv:1106.4573

67. Beyer, Jones, Petoff and Murphy (eds.). Site Reliability Engineering. O'Reilly, 2016. Chapter 5, Eliminating Toil. Read 13 November 2026. sre.google

68. Yang, Zhang, Liu, Chen, Cao, Hooi and Zhang. From Approval to Abuse: Privilege Escalation Attacks on Harness-Protected Coding Agents. 2026. Abstract. arXiv:2610.22987

69. Anthropic. Claude Managed Agents, self-hosted: documentation. No version number is published. Available on request; beta. Read 12 and 13 November 2026. platform.claude.com/docs/en/managed-agents/self-hosted

A Appendix A. Method detail

A.1 The starting point and the documented search

The survey of Section 2.2 read six academic architectures, eleven standards efforts and protocols, fourteen vendor platforms and frameworks, and three practitioner accounts. The three platforms of Section 3.2 were chosen as a managed cloud platform, a framework with a hosted service, and an open-source personal agent system. When the first versions of the model were drawn from the reference implementation, an earlier classification of the system's scheduled jobs by two raters was available as a check. A second reading on 11 November covered what the survey of Section 2.2 had left out, including Domain-Driven Design applied to agents and Microsoft's reference architecture.

The documented search covered arXiv, OpenAlex, general web search, vendor and standards sites, and GitHub. It departed from its protocol in ways that its record states. Three web queries were added, the restriction of some searches to a vendor's own site did not hold, some sources were read through a generated summary of the page

and not the page itself, and the raw search responses were not kept. Table A1 gives the counts. The count after removing duplicates is approximate to within a few links. Eight candidate sources could not be opened, among them a reference architecture from the Cloud Security Alliance whose download requires a form, and further material is listed in the search record as unverified.

STAGE	RECORDS
Screened, within the page caps	1,634
After removing duplicates (approximate)	1,314
Passed title screening	258
Met the inclusion rule (a general model of an agent system)	61
Already known from the earlier reading	13
New	48
New sources mapped to WARM	8

Table A1. The documented search of 11 November 2026. Forty of the 48 new sources have not been mapped.

A.2 The comparison, the grid and the four criteria

The second round of the comparison with published models treated the first round's changes as planned. The second rater of that comparison drew nine of the readers' assignments from each source and assigned each from the source's quoted definition before seeing the reader's answer. Two spans counted as agreeing when their main parts agreed. Because the sample did not include the reverse-direction verdicts, it is evidence about how the sources' parts were filed and not about the grid. A further reading checked how the fourteen sources treat autonomy and authority granted in advance.

Each filled cell of the grid cites a line of the notes, and a script confirmed that each cited line exists and contains the quoted words. That check establishes faithful transcription and not the soundness of each judgment. The notes are in the evidence supplement (Section 7). The first grid also had columns for the domain layer, the charter and the Directory. A review of the first draft of this paper found that those columns mixed three questions and rested on one reader's scoring, and they were replaced by the four criteria of Section 3.3.

Boundary cases between the two readers. The first reader rated AAM/5C in part on C1 and C2 because its organizational unit is the single governance body for the whole system. The second reader rated OperA in full on both, although its unit is the whole society, and the frozen C1 criterion does not require several areas. The first reader discounted ASOC's requirement to find and bind services by goal (C4) because the source is a manifesto without a mechanism and uses an orchestrating agent, and neither ground is an exclusion in the criterion. The second reader gave OperA's matchmaker a partial rating on C4 partly because it appears in an example, and gave the survey's facilitator a full one. The first reader gave Microsoft's architecture a full rating on C4 without marking it disputable and noted that a stricter reader would.

The second rating of the boundary cases. The clarified criteria keep the four criteria and their three values and add a ruling on each point above (Section 3.3). The two further readers were instances of one model. Each was given only the clarified criteria and the locations of the eight sources, and was told not to open the notes, the earlier ratings, the model or this paper. They agreed on 6 of 8 cells for C1, 5 of 8 for C2 and all 8 for each of C3 and

C4. The five differences are C2 for AAM/5C (in part with a full alternative, against full), C1 and C2 for ASOC and C2 for Microsoft (in part, against no with a partial alternative), and C1 for the survey (full, against in part with a full alternative). Neither reader could open the JaCaMo article or the ISLANDER article at first. The author obtained both afterward and each reader checked its ratings against them: one reader lowered JaCaMo on C2 from full to in part, and nothing else changed. One reader read only the defining sections of the 195-page AAM/5C manuscript. A script compared the readers' quoted passages with the source texts. The mismatches it found were artifacts of text extraction, apart from one quotation whose opening words had been altered.

A.3 Detail of the results and the second ratings

Table A2 gives the assignment of inventory entries for the three platforms and the reference implementation (Sections 5.1 and 5.4).

PLATFORM	COMPONENTS	ONE SERVICE	ANOTHER LAYER	BUILDER TOOLING	SPAN	MISFIT	SINGLE CATEGORY
Amazon Bedrock AgentCore	146	92	5	10	28	11	66%
LangGraph, LangSmith Deployment, LangSmith	307	183	22	55	30	17	67%
OpenClaw	848	329	212	256	38	13	64%
LifeOS (reference implementation)	849	400	288	93	61	7	81%

Table A2. Inventory entries by assignment. "Single category" is the share assigned to one service or one other layer. Eighteen of the 61 LifeOS spans are entries that group unrelated scripts by file name, not components that span responsibilities. The platforms were mapped against version 0.10 from documentation, and LifeOS against version 0.13 from its repository.

Platforms. The platforms cover more of the model's list of services than the predictions expected, and they contain more that the model lacked. Each platform's own top-level names overlap the model's; AgentCore's include Harness, Memory, Identity, Policy, Observability, Registry and Evaluations. Of the three platforms, one built a shared registry of tasks in March 2026 and removed it in September 2026, giving duplicated state as the reason. One prediction was that each platform would lack several services. AgentCore lacks one (Scheduling), and the other two lack none, although each has some services only in part. All twelve sampled assignments that the mappers had marked as clean fits were confirmed. Every disagreement was in a row that the mapper had marked strained, a span or a misfit.

The first grid and the separate rating. The first grid, assembled from notes, had no source with an area of work in full and three with a charter in part. The two ratings used different rules, so they are not a measure of agreement. Read as three-valued verdicts, they agree on 11 of 17 sources for the area of work, 9 of 17 for its specification and 14 of 17 for the record of accountability, and every difference is the separate rater giving the higher value.

The reference implementation. The seven misfits are four components that manage vendor accounts and usage quota, one agent definition that mixes a platform service with a domain role, the harness's plugin host, and a per-agent display setting. Agreement on the 60 sampled assignments was 14 of 15 on clean one-service fits, 10 of 15 on

strained fits, 8 of 10 on other layers, 6 of 8 on spans, 4 of 5 on builder tooling and none of 7 on misfits. The rater assigned every one of the mapper's misfits to some part of the model before seeing the mapper's answer, and accepted four of them as misfits afterward. On the 24 one-service rows where the two agreed on the service, they split 14 to 10 on whether the fit was clean or strained, in both directions. The disagreements fall in three places: the boundary between Software domain work and a platform service (review, limits on work in progress), components that do two jobs, and the class of vendor accounts and quota, where the mechanism is Secrets and only the purpose is new.

A.4 Industry terms

Table A3 gives, for each service of Table 2, the terms that the survey of 34 sources (Section 2.2) found for it.

SERVICE	INDUSTRY TERMS
Harness	harness, agent loop, context management, compaction
Sessions	session, thread, state, checkpoint
Tasks	runs, jobs, task lifecycle, durable execution
Environments	sandbox, runtime, workspace, computer use
Scheduling	cron jobs, triggers, queues, concurrency limits, rate limits
Lifecycle	agent lifecycle, provisioning, decommissioning, orphaned agents, sprawl
Memory	memory, memory bank, dreaming
Retrieval	retrieval, knowledge base, grounding, RAG
Artifacts	artifacts, outputs, store
Models	models, model routing, multi-model
Tools	tools, MCP, gateway, connectors
Registry	registry, catalog, discovery, skills, hub
Directory	(none found)
Messaging	agent-to-agent, A2A
Orchestration	orchestration, subagents, handoffs, workflows
Channels	channels, surface, agent interface, invoke API, endpoints
Identity	identity, agent identity, authentication
Secrets	secrets, secret manager, vault, credentials
Policy	policy, guardrails, content filters, permissions, hooks, retention
Approvals	human in the loop, tool approval, standing approval
Observability	observability, tracing, audit log
Evaluation	evaluation, evals
Improvement	recommendations, insights, optimization, self-improvement

Releases	deployment, versioning, rollback, change management
Reliability	operations, recovery

Table A3. Industry terms for the 25 platform services. The survey found no term for the combination that the Directory records, accountability for a responsibility together with how to reach it; terms for discovery appear under Registry.

B Appendix B. What the exercises changed in the model

Table B1 lists the larger changes to the model between versions 0.10 and 0.14 (Section 5.7). In the platform mapping, six further findings, each from one platform, were recorded and not adopted at the time: payments, tenancy, agent provisioning, governance of a marketplace of plugins, outbound events and caching. Two of them returned in the comparison with published models and were adopted then.

FINDING	SOURCE OF THE FINDING	CHANGE TO THE MODEL
A program has no place to call an agent	All three platforms	Channels widened from people to people and applications
The platform runs a loop that turns evidence into changes	All three platforms	Improvement added to Operations
No platform keeps a separate store of tasks	All three platforms	Tasks restated as a responsibility that does not require its own store
Skills are versioned definitions	AgentCore, LangSmith	Skills moved from Memory to the Registry
A registry is a catalog, and nothing records who is accountable for what	All three platforms	Directory made a service of its own
Governance is enforced at certain points or for certain products	AgentCore, LangSmith	The first rule of Governance (the enforcement point)
Retention and purging of stored data; queues and limits on what runs next	LangGraph; OpenClaw	Added to Policy and to Scheduling
Authority scoped to one action, issued on a positive decision	Yu et al., Zhu, AAM/5C	The second rule of Governance (the grant)
Whoever proposes an action does not approve it	Yu et al., Zhu, AAM/5C	The third rule of Governance
Filtering of prompts, outputs and tool payloads sits in governance	Microsoft, Amazon, Google	Policy covers content as well as actions
Events are carried between components	Yu et al.; one platform	Messaging delivers events to subscribers
A top-level part for creating and ending agents (only AOS also suspends and resumes them, and none states a duty to keep a list)	AOS, Steinder and Franke, EAAF	Lifecycle added as the 25th service; the list duty is the author's addition

Risk or confidence is assessed at run time and decides whether an action proceeds	Four models of the second round	Evaluation assesses before, during or after an action
Authority is granted in advance under conditions	Twelve of fourteen models, in full or in part; the reference implementation	Pre-approval as a kind of decision. Its two rules (a stated term; irreversible actions named) are the author's and do not follow uniformly from the sources
Agents change their own definitions and rules under a standing permission	The reference implementation	The granting of roles and rules changed from a kind of decision to a principle
Every stop for a person should be recorded and reviewed	The author's direction, not an exercise (version 0.14)	The rule of Section 4.5 added; the definitions of Approvals and Improvement changed

Table B1. The larger changes to the model between versions 0.10 and 0.14, with the finding behind each.

Two decisions in Table B1 were the author's judgment where the evidence showed a gap and did not show how to fill it. Lifecycle could have been a widening of Identity and Tasks instead of a service. Assessment at run time could have been a new service instead of a widening of Evaluation. Artifacts was kept although only one source of seventeen has a full counterpart, because sustained work produces reports, files and pages that outlast a session and need versions. One of the published models argues the opposite, that a file system is enough [23].

Unresolved after the comparison. The second rater of the published models listed three things that the sources have and the model still does not place: a trust score that is kept for each actor, decays, and is read by later decisions; action on a gap between what was authorized and what happened; and a stop that applies to every agent built from one definition. The model's Evaluation and Lifecycle services cover a judgment on one proposed action and a stop of one agent.

Levels of autonomy. The model defines no levels of autonomy. Six of the fourteen published models grade autonomy or decision authority in levels or classes, and they do not grade the same thing: some grade a system, some an agent's contract, and one a class of decision. WARM's kinds of decision apply to a class of action, and one agent commonly acts under different kinds for different classes.

c Appendix C. Claims that the comparison did not support

Early versions of the model made five claims about how it differs from prior work (Section 5.7). The comparison withdrew all five and kept narrower observations.

1. **Vendor-neutral reference models.** An early version claimed that none existed, and this is false. Microsoft calls its architecture's approaches "agnostic" [15], Yu and others present "a technology-neutral reference architecture" [12], and Goujon presents a "prescriptive reference model" [14]. The narrower observation, as of 11 November 2026, is that the readers found no evidence of a named adopter among the fourteen mapped models.
2. **Areas of work apart from agents.** An early version claimed a difference here, and the comparison refuted it. AEON's subdomains and Charterion's business scopes are areas of work apart from agents [18, 19], and the organizational multi-agent literature defines an organization and its roles apart from the agents that fill them [39, 40, 41, 42]. What remains is narrower and was not rated under a criterion. The reader of the ten later sources

noted that none of them gives each unit one written charter that also states its acceptance criteria, its tools and knowledge, and what it offers other units. The older sources carry related content across several artifacts, so the difference may be one of packaging (Section 5.3). The word "charter" is not WARM's either. Charterion charters an agent, and Microsoft's adoption guidance has teams write a charter for a system of agents that states the system's responsibilities, agent roles that do not overlap, and prohibited actions [44].

3. **The agent as an identified actor.** An early version claimed that an agent as an identified actor, distinct from its running instances, is new, and the claim is withdrawn. Charterion defines an agent as "a non-human principal" whose identity "may be instantiated many times" [18], and AOS separates "a persistent agent identity" from "a transient runtime instance" [22]. One difference remains. In Charterion the identity belongs to a chartered agent and is shared by its instances, and WARM gives each task agent an identity of its own.
4. **Persistent agents and task agents.** An early version claimed that separating the two kinds is new, and the claim is withdrawn, because OpenClaw separates them [34]. A separate observation stands. Among the fourteen published models, most have one kind or leave lifetime unstated, and one rejects long-lived agents by design [13].
5. **Pre-approval and autonomy.** An early version claimed a difference in its treatment of pre-approval and autonomy, and the claim is withdrawn. Charterion and AAM/5C state pre-approval more completely than WARM does.

D Appendix D. Glossary

Table D1 gives the meaning of each term as WARM uses it.

TERM	MEANING IN WARM
Agent	An identified actor that carries out one or more assigned roles. Distinct from an instance (a running copy) and from an agent definition.
Agent definition	A description of a role: its instructions and the skills it refers to.
Persistent agent	An agent that holds its roles across many sessions and tasks.
Task agent	An agent that holds a role for one task. A task agent started by another agent is also called a subagent.
Domain	An area of work: a body of specialized knowledge, a set of responsibilities that belong together, and one or more patterns for carrying them out.
Charter	The written statement of a domain: goals, responsibilities, roles, workflow, knowledge, tools, rules, acceptance criteria and the published skill.
Published skill	The part of a charter written for the agents of other domains: what the domain offers, when to use it, how to ask and what comes back.
Domain pattern	A recurring way of carrying out a group of a domain's responsibilities. Distinct from an orchestration pattern, which is how several agents coordinate on one task.
Subdomain	A named group of a domain's responsibilities that has its own workflow. The domain remains accountable for its work.
Platform	The services that agents in any domain can use.

Service	One responsibility of the platform. Only the platform has services.
Harness	The platform service that runs an agent's loop.
Enforcement point	A place where a call is checked against the rules and can be blocked.
Grant	Authority for one action or one task, issued on a positive decision by Policy.
Pre-approval	A person's authorization, given in advance, of a class of actions under stated conditions and for a stated term.
Stop	A point at which work waits for a person: an action held for a decision, or a task that an agent handed to a person.
Hand-off	A task that an agent gives to a person because it cannot, or believes it cannot, do it.
Record of stops	The one record to which Approvals and Orchestration write each stop with its reason, and which Improvement reads.
Implementation	A real system that fills in one part of the model.

Table D1. Terms of the model.

Created in collaboration with AI.