WORKING AGENTS SERIES

WARM

The Working Agents Reference Model

A map of a system in which agents do sustained work.

Drawn from the system we run, with vocabulary from a survey of 34 sources, then checked against three agent platforms, fourteen published models and ten vendor products.

Model v0.14 · November 2026 · Dave Remy, Inner Stack Labs · *Created in collaboration with AI.*

THE SHORT VERSION

Platform services are widely named. We put them in one vocabulary.

1 · What we mapped

Our own system first, then three agent platforms, fourteen published models and ten vendor products.

2 · The model

Five layers: people, domains, a platform of 25 services in six groups, infrastructure and providers.

3 · What we found

The services are widely named. Six have no full rating in any of ten products. Most models have an area of work only in part.

Why a map

With shared names you can compare systems, and see what your own is missing.

One motivation

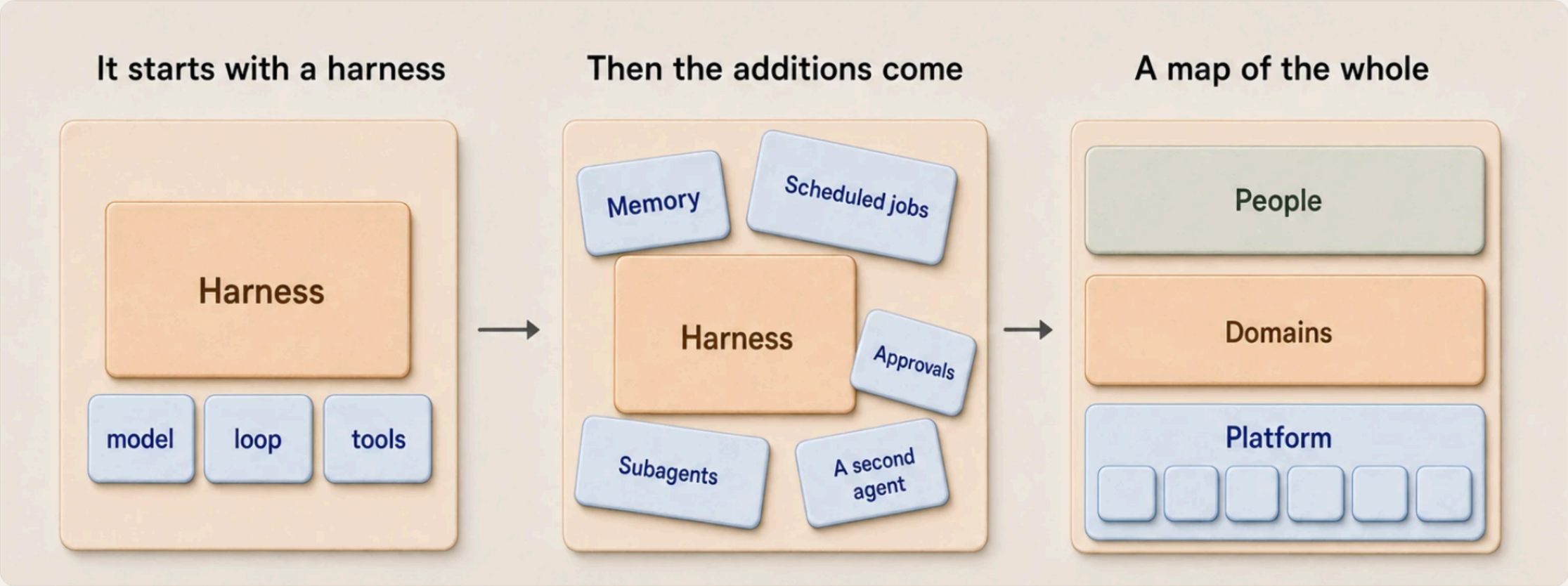
We want agents to finish work without a person approving each action. Saying why work stopped for a person is easier when the parts have names.

Status

A working paper, drawn largely from one system. No evidence yet that organizing a system this way improves it.

WHY

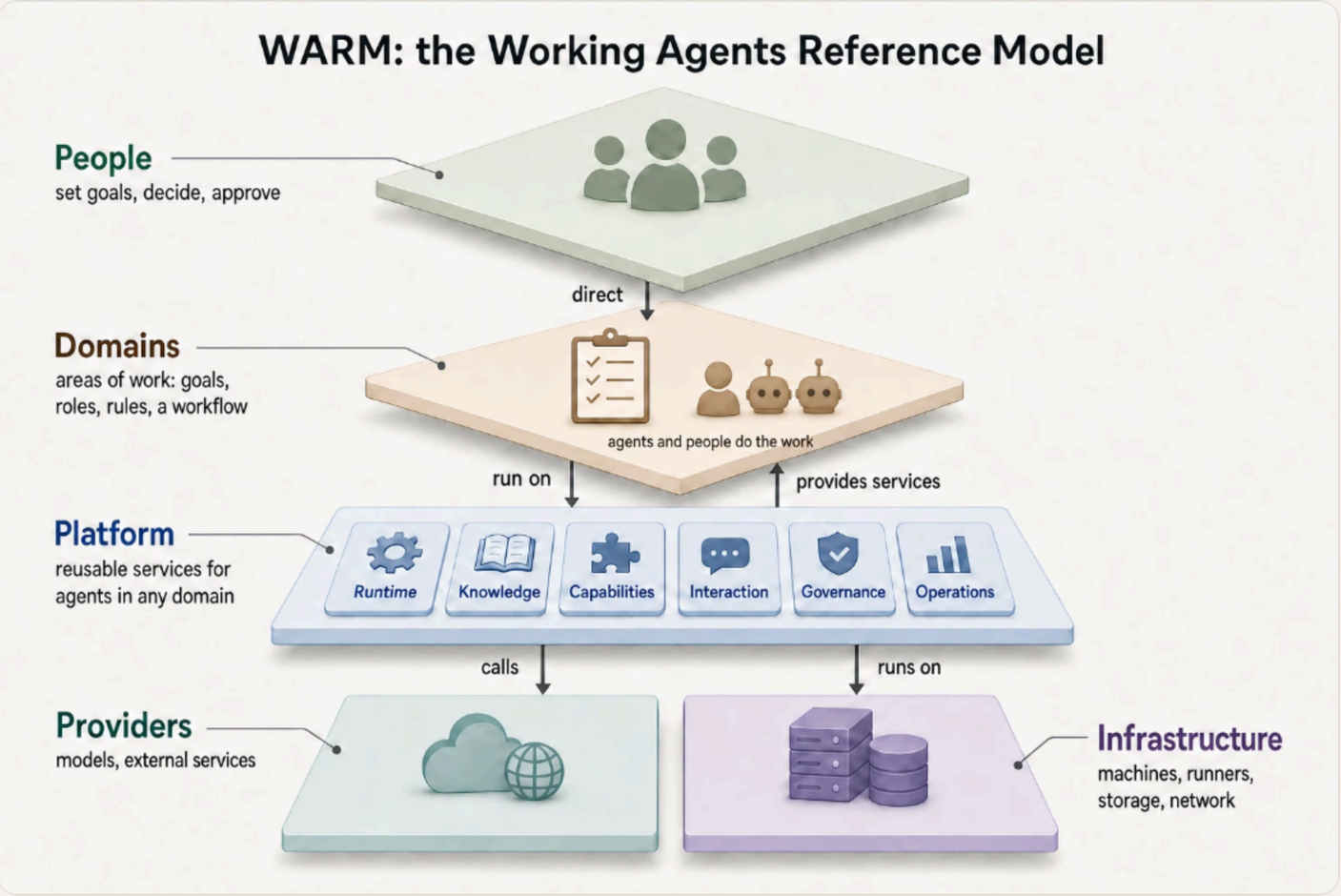
Our system grew one problem at a time



We drew the map from our own system, and a survey of 34 sources gave it vocabulary. None names a platform, an area of work and a way of working as three levels. Older multi-agent platforms had joined them.

THE MODEL

Five layers at the top level



People direct the work

People set goals, make decisions and approve actions.

Domains define the work

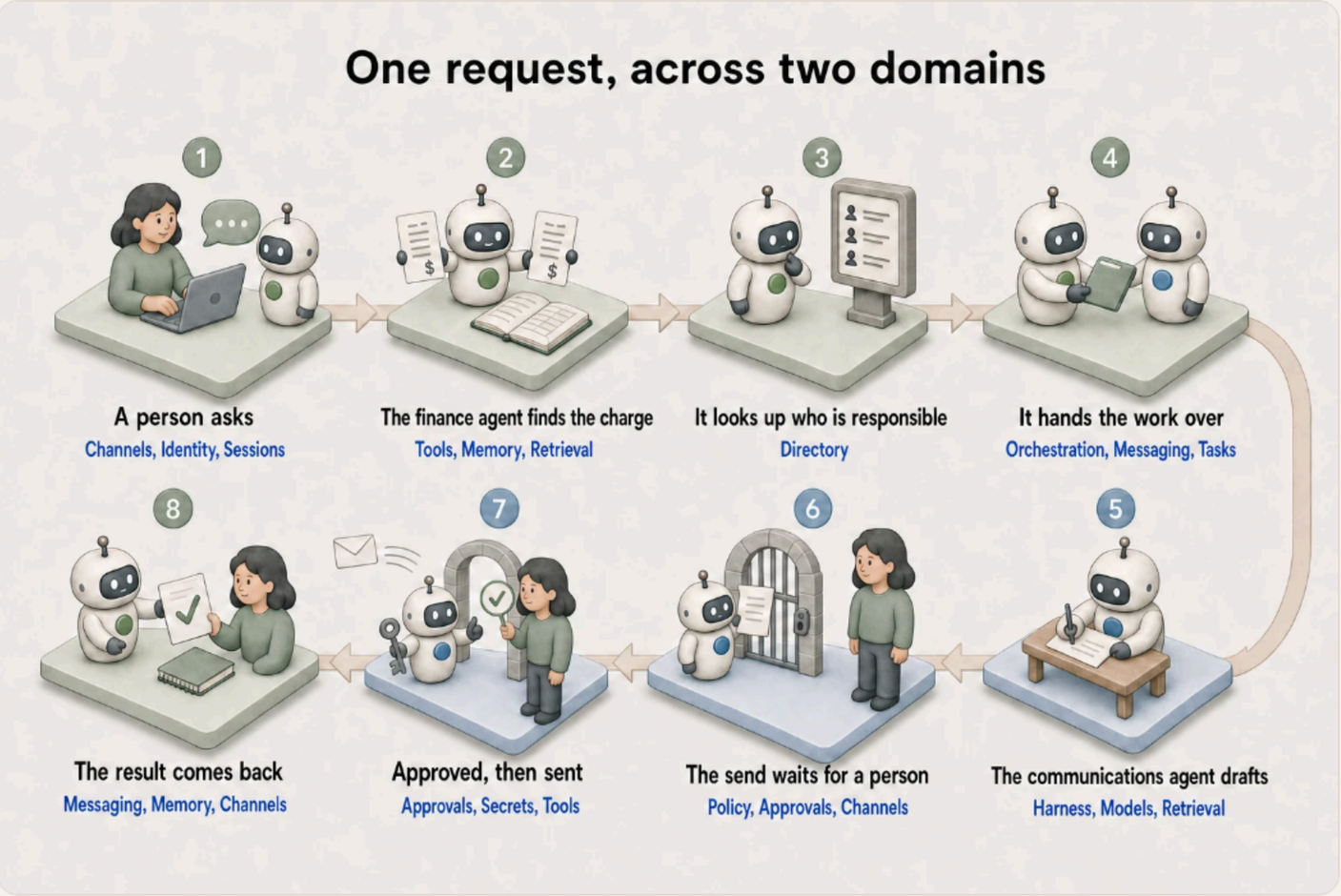
The platform holds what works the same whatever the work is about. A domain holds what depends on the subject of the work.

Infrastructure and providers

The platform runs on infrastructure and calls providers.

AN EXAMPLE TO FOLLOW

One request, across two domains



The request

"There is a duplicate charge from the gym. Dispute it." The finance agent can find the charge. It may not send a message to the merchant.

Two areas of work

Messages in the person's name belong to Communications. The finance agent looks that up and hands the work over.

One stop for a person

The send is held until the person approves. Had the person approved such messages in advance, the platform would check the conditions and send with no stop.

DOMAINS

A domain is an area of work

A domain example: Finance

Responsible for: tracking spending · watching subscriptions · reporting

ITS CHARTER STATES

Goals	Responsibilities	Roles
Workflow	Knowledge	Tools
Rules	Acceptance criteria	**Published skill**

ITS ROLES ARE FILLED BY

a person	a persistent agent	task agents

People direct it. Its work runs on the platform. A domain does not act; the people and agents in its roles do.

A charter, not an actor

Its charter says what the work is. The people and agents in its roles do the work.

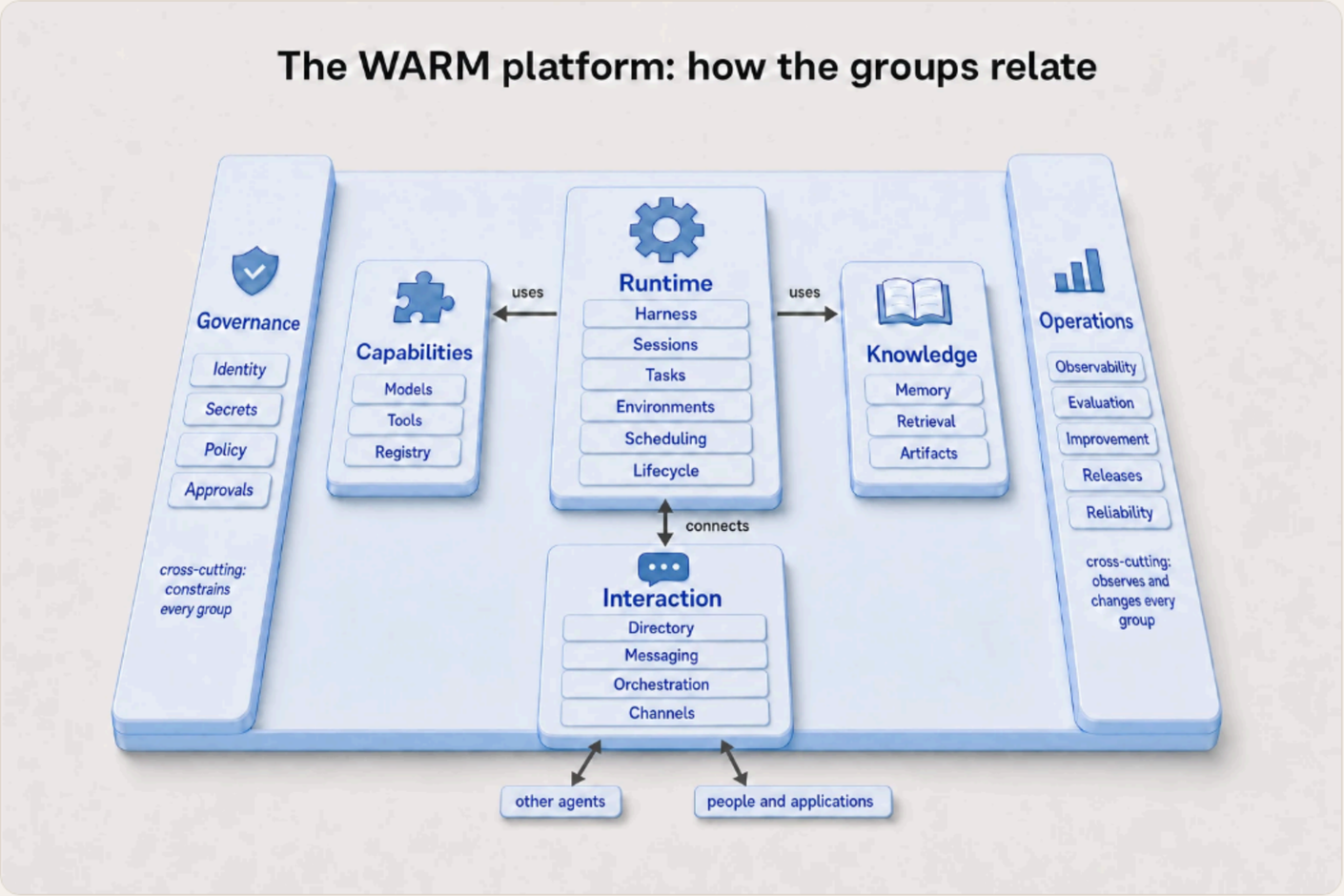
Listed, with a published skill

One domain is accountable for each responsibility, and the Directory records it. The published skill tells other domains when and how to ask.

Subdomains

A named group of its responsibilities with its own workflow. The domain stays accountable. Example: running the product, in Software.

The platform: six groups, twenty-five services



Four working groups

Runtime: how agents run. Knowledge: what they know. Capabilities: what they can use. Interaction: who they work with.

Two cross-cutting groups

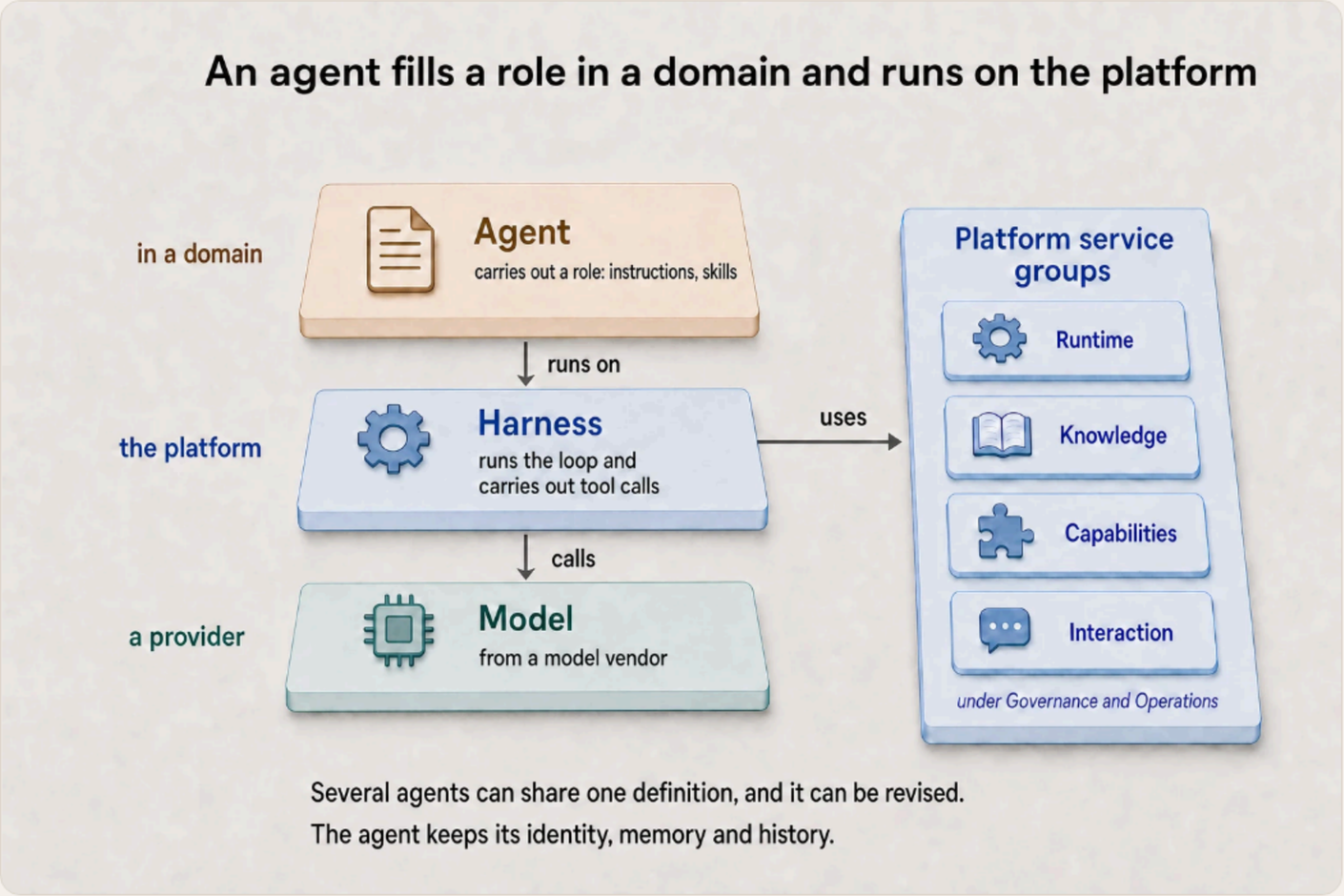
Governance: what agents are allowed to do. Operations: how the system is seen, judged, improved and kept running.

Services are responsibilities

Twenty-five in all. One product may cover several, and a system does not need them all.

AGENTS

An agent fills a role in a domain and runs on the platform



An actor with a role

An agent is an identified actor. Its definition describes the role: instructions and skills. The Registry holds it.

Runs on the platform

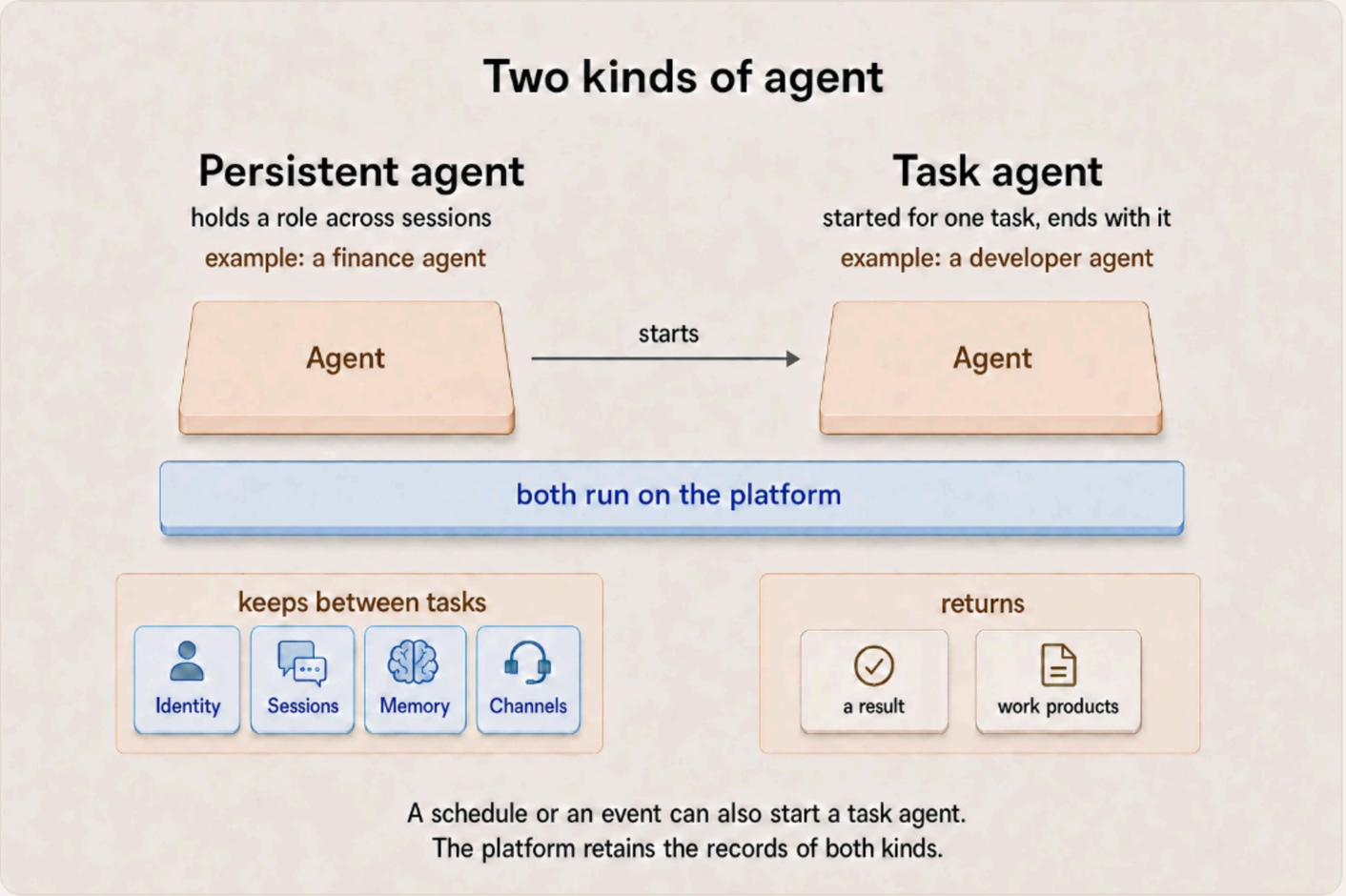
The Harness runs its loop and calls a model. From the loop the agent uses the other services.

What persists

Agents can share a definition, and it can be revised. The agent keeps its identity, memory and history.

AGENTS

Persistent agents and task agents



Persistent agent

Holds its roles across many sessions and keeps its history and memory. Example: a finance agent that a person consults.

Task agent

Holds a role for one task, then ends. Example: a developer agent started for one issue.

Guidance from experience

Run unattended work in a scheduled job or a task agent, not inside a conversation.

DOMAINS · AN EXAMPLE

Software: a domain, a pattern, an implementation

THE DOMAIN

Software

Goal. working software, released

Roles. a coordinator, developers, reviewers; a person as product owner

Rules. who may merge what

Acceptance. tests pass and reviewers approve

THE PATTERN

Factory

Work waits in a backlog

↓

A coordinating agent hands out items

↓

Task agents take each item through fixed stages

↓

Criteria and reviewers judge each result

A pattern also states when a person is involved.

THE IMPLEMENTATION

Agent Scrum

Our implementation of the Software domain in the factory pattern.

Another team could implement the same pattern differently.

A domain usually mixes several patterns. Which patterns recur is an area of study, and we expect to find more.

EVIDENCE

A map of the field

Areas of work: sources that meet each criterion in full

Sources	An area of work exists	Specified apart from who does the work	A record of accountability	Finding where to send work
3 platforms and 14 published models	2	0	0	3
5 models of agent organizations, 2002 to 2004	3	1	0	1
3 platforms of that literature, 2002 to 2013	1	0	0	1
5 further recent sources	0	0	0	1
All 30	6	1	0	6

Rated under criteria written first, mostly from primary passages (some extracts, one generated summary). 23 of 68 cells for the first 17 sources are disputable; lower value shown.

Platform services: widely named

Each of the 25 has at least a partial counterpart in nine or more of 17 sources. Thinnest: Artifacts, Improvement, Lifecycle.

Areas of work: an older idea

The 2002 to 2004 models define an area and its roles apart from the agents that fill them: Gaia in full, three others on one reading. Recent models have it in part.

No undisputed full rating

A record of one accountable area for each responsibility.

EVIDENCE

Ten vendor products, read against the 25 services

1 · Ten products we chose

Four harnesses, five personal agents and one agent platform, from the vendors' own pages, first read on 12 November 2026.

2 · Two readers

One reader each. A second model family re-read a sample: 98 of 120 cells the same.

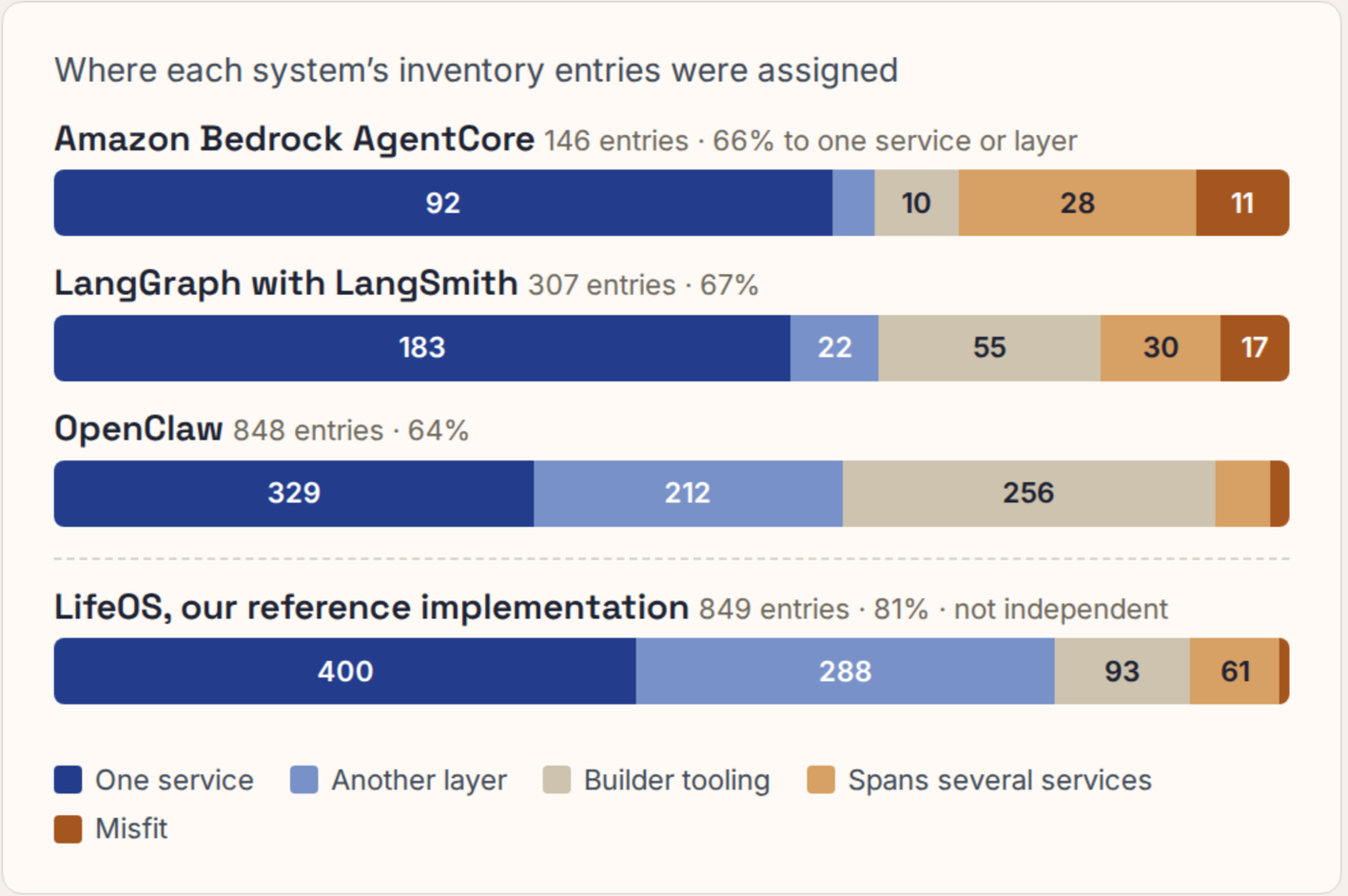
3 · A dated map

One product was renamed before the reading and one launched that day. The method is written down so the reading can be repeated.

Every product	Harness, Retrieval and Tools in full. Approvals in full from five of the ten, as version 0.13 defines it.
No product	Directory, Identity, Policy, Evaluation, Improvement and Releases: no full rating from any of the ten.
Harnesses	Full on 11 to 18 of the 25 services; no service absent.
Areas of work	None in full on an area of work, its specification or a record of accountability. One lets an agent find a shared agent by kind of work.

EVIDENCE

Three platforms and LifeOS, mapped to the model



The platforms

Each has most of the 25 services. AgentCore lacks Scheduling; the other two lack none, some only in part.

LifeOS

No service absent. 14 to 16 present, 9 to 11 partial, after two raters discussed. The model was drawn partly from it.

What no platform has in full

An area of work as a defined part (LangGraph disputable), or a record of who is accountable for what. Research on agent organizations defined areas of work apart from agents in 2002 to 2004.

WARM v0.14 · source: paper, sections 5.1 and 5.4 · platforms from documentation as of 2026-11-11 · *Created in collaboration with AI.*

CLOSE

What WARM offers, and its limits

What it offers

- a taxonomy of the parts: layers, groups, services, domains and agents
- how the parts relate, with one test for what belongs in the platform
- a list to set your own system against

Its limits, and what we are doing

LIMIT	REMEDY
One system (LifeOS), on one harness (Claude Code)	Three platforms mapped. Others map their own systems: we ask on the next slide.
Compared and rated by our own team	Second raters, a review by a model from another vendor, and a re-rating by a rater that had not seen the earlier answers: done. Asking the authors of the published models: not yet started.
Says what each service is for, not yet how it must behave when things go wrong	A paper for each group will propose contracts. Not yet started.
No evidence yet that it improves a system	Our refactor of LifeOS is the first test. Under way; measures still being set.
Published skill, domain pattern and subdomain rest on one system	Pilot one published skill in LifeOS. Not yet started.

CLOSE

Next: the paper, our implementation, the group papers

1

The paper

A full account of the model and the evidence behind it. Version 0.1 is published, with a supplement.

2

Our implementation

LifeOS refactored toward the model, with measures and baselines for the test.

3

The group papers

One paper for each group of services, proposing the contracts that the model leaves open.

Try WARM on one workflow in your own system, and tell us what did not fit. We will also ask the authors of the published models whether we read them correctly.

APPENDIX

Reference slides

Scope, the six groups one at a time, where what an agent knows is kept, agents and domains, implementations, the uses of the model, tracing concepts to code, fourteen published models, three platforms and LifeOS.

APPENDIX · SCOPE

What WARM covers

WARM names the parts of a system in which agents do sustained work, and states how the parts relate.

It describes

- who directs the agents
- what work the agents carry out
- which reusable services they use
- what the whole system runs on

It leaves out

- how a single agent reasons, plans or reflects
- any vendor or product
- the contract of each service, which later papers are to propose

APPENDIX

How to place a part

Does it work the same whatever the work is about?

Platform

Holding an action until a person approves it works the same for an email, a payment or a merge.

Domain

“Every message in this person’s name needs approval” is about one area of work. It belongs to the Communications charter.

A design rule

It sorts responsibilities, not how they are built. It cannot be refuted; its test is usefulness.

The platform answers six questions

Runtime

How, where and when do agents run?

Knowledge

What do agents know and keep?

Capabilities

What can agents use?

Interaction

Who do agents work with?

Governance

What are agents allowed to do?

Operations

How do we see it, judge it, improve it and keep it running?

Six groups hold twenty-five services. The four groups in the top row and at the lower left are working groups; Governance and Operations are cross-cutting.

Working groups and cross-cutting groups

Working group

An agent mostly calls its services for their result.

Runtime, Knowledge, Capabilities, Interaction

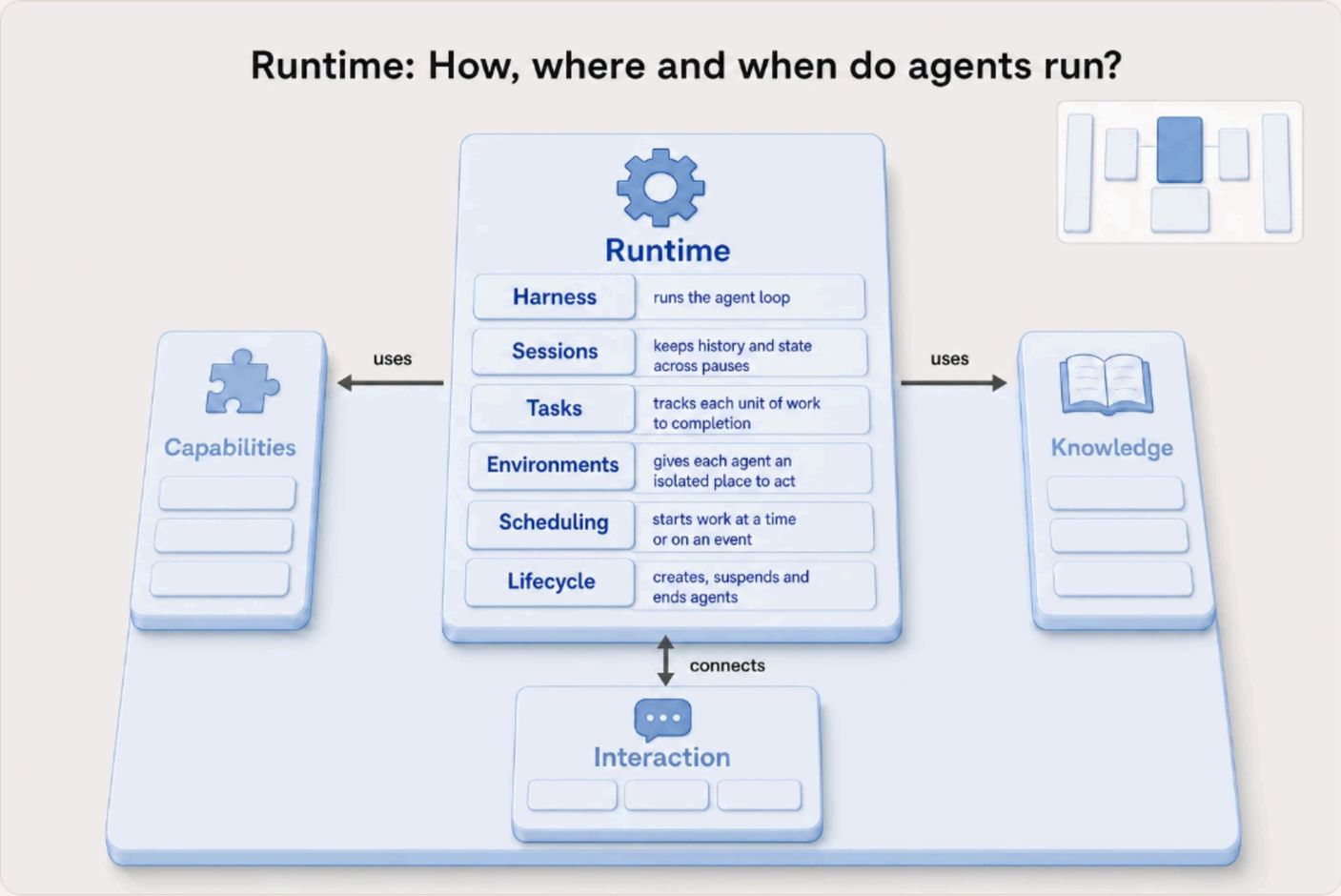
Cross-cutting group

Its services mostly apply to actions whose purpose lies elsewhere.

Governance, Operations

1. The kinds describe a group's main scope. Some are reached both ways: an agent asks Evaluation for a verdict.
2. The two cross-cutting groups apply to each other. A release can require an approval, and an approval is recorded.
3. The relations are not a layering. No group sits on another.
4. Governance covers an action only if it passes an enforcement point, which a record at the resource can check.

Runtime



The question

How, where and when do agents run?

Six services

Harness, Sessions, Tasks, Environments, Scheduling, Lifecycle.

Sessions and tasks differ

Sessions keeps the history of a stretch of work. Tasks tracks the lifecycle of one unit of work.

APPENDIX · WORKING GROUP

Capabilities

SERVICE	ITS ONE RESPONSIBILITY
Models	Mediates requests to models and chooses the model for each task, across more than one model or vendor.
Tools	Mediates calls to tools, local or external.
Registry	Holds the versioned definitions of agents, tools and skills, including each domain's published skill, so that each can be found and loaded when a task needs it.

Used by Runtime: work in a session or a task draws on these services

The question

What can agents use?

Three services

Models, Tools, Registry.

Each service mediates

Each service stands between the agent and something outside it, so the thing outside can change.

APPENDIX · WORKING GROUP

Knowledge

SERVICE	ITS ONE RESPONSIBILITY
Memory	Retains what agents learn across sessions (facts and events), with where each entry came from, and keeps it current.
Retrieval	Fetches material from named sources: documents, data, notes.
Artifacts	Stores what agents produce (reports, files, pages) and its versions.

Used by Runtime: work in a session or a task draws on these services

The question

What do agents know and keep?

Three services

Memory, Retrieval, Artifacts.

Learned, looked up, produced

Memory keeps what agents learn. Retrieval fetches what they look up. Artifacts stores what they produce.

Where what an agent knows is kept

What an agent knows is kept by six services, by how long it lasts and who wrote it.

What is kept	Service
What is in the context window now (working memory)	Harness
The history of one stretch of work	Sessions
What agents learn and keep across sessions: facts and events	Memory
How to do a kind of task: skills, written or revised	Registry, with Releases
Material that is fetched when needed	Retrieval
What agents produce	Artifacts

APPENDIX · WORKING GROUP

Interaction

SERVICE	ITS ONE RESPONSIBILITY
Directory	Records which domain is accountable for each responsibility, and how to reach a role of that domain.
Messaging	Delivers messages between agents, and events to their subscribers.
Orchestration	Coordinates assignments among agents, people and fixed steps, and the results that come back.
Channels	Carries requests and results between agents and the people and applications that use them.

Connects Runtime to other agents, to people and to applications, in both directions

The question

Who do agents work with?

Four services

Directory, Messaging, Orchestration, Channels.

Both directions

An agent sends through these services, and an arriving message or request starts or resumes work.

APPENDIX · CROSS-CUTTING GROUP

Governance

SERVICE	ITS ONE RESPONSIBILITY
Identity	Establishes who is acting, and on whose behalf. Records the person accountable for each agent identity, and revokes an identity.
Secrets	Holds values that agents must not see, and releases each one only at the point of use.
Policy	Decides by rule whether an action may proceed, what content may enter and leave, and what is kept. Issues a grant when an action is allowed.
Approvals	Obtains and records people's authorizations: pre-approvals for a class of actions, and decisions on single actions that are held.

Constrains the other five groups: Runtime, Knowledge, Capabilities, Interaction, Operations

The question

What are agents allowed to do?

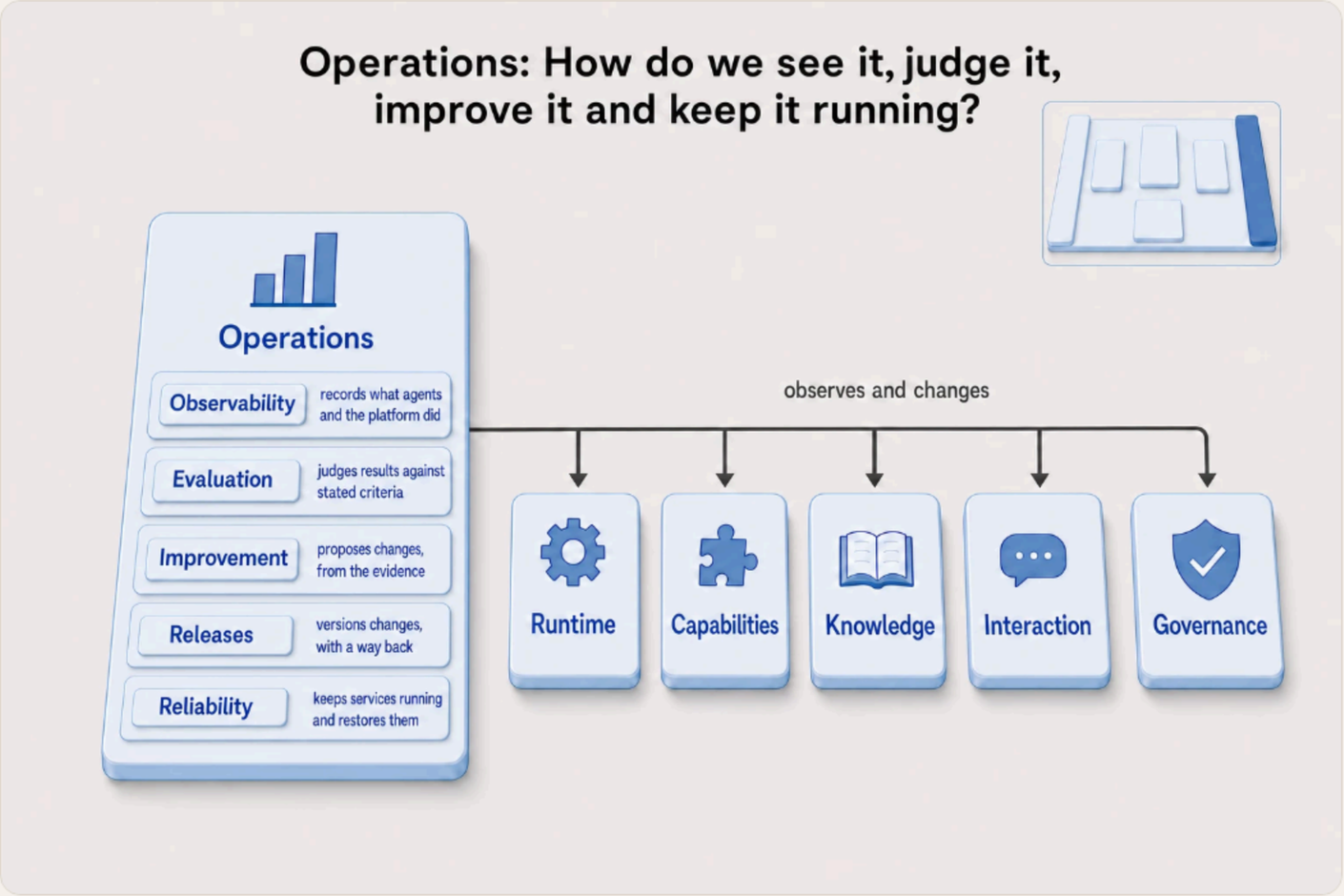
Four services

Identity, Secrets, Policy, Approvals.

Three kinds of human decision

For each class of action: reserved for people, pre-approved under conditions, or held for a person. All authority traces to a person.

Operations



The question

How do we see it, judge it, improve it and keep it running?

Five services

Observability, Evaluation, Improvement, Releases, Reliability.

The improvement loop lives here

Evidence, a judgment, a proposed change, a versioned release and a way back.

Twenty-five responsibilities

The service names are responsibilities. They are not products.

Many to many

One product may cover several responsibilities, and one responsibility may be shared by several products.

Not all are needed

A system with one agent needs no messaging. The model still applies to it.

Contracts come later

A contract is what a service guarantees, for example when a call fails or work is canceled. Later papers are to propose them.

APPENDIX

How a domain reaches the platform

THE CHARTER STATES	COMMONLY CARRIED BY
Goals	set by people
Responsibilities	Directory
Roles	Registry (definitions); Directory (how to reach a role)
Workflow	Tasks, Scheduling, Orchestration
Domain knowledge	Memory, Retrieval, Registry (skills)
Domain tools	Tools
Domain rules	Policy, Approvals
Acceptance criteria	Evaluation
Published skill	Registry (holds it); Directory (points to it)

Nine things a charter states

People set the goals and approve the charter. Platform services commonly carry the other eight.

Roles become agent definitions

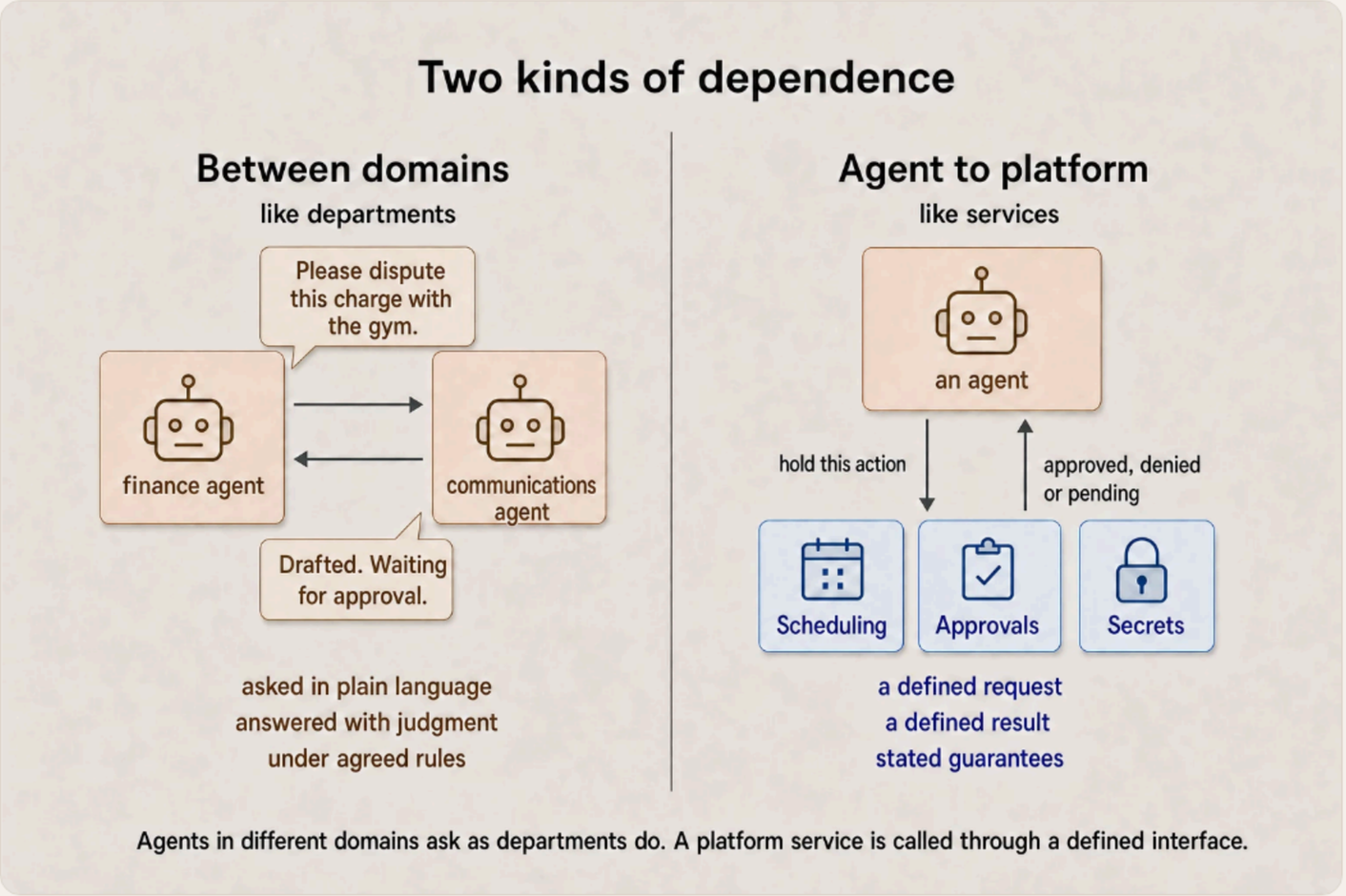
Each role an agent fills is a definition in the Registry. The Directory says how to reach it.

Specific to the domain

What works the same whatever the subject belongs in the platform. A domain can still need new work, such as a connector.

APPENDIX

Two kinds of dependence



Agents in different domains ask as departments do. A platform service is called through a defined interface.

Between domains

One agent asks another in plain language. The other uses judgment, under rules agreed in advance.

Agent to platform

A defined request through a defined interface, with stated guarantees. Results can vary.

Why it matters

What works the same whatever the work is about goes in the platform. The rest stays with a domain.

How many agents a domain has

WARM sets no number. Three arrangements are possible, and one system can use all of them.

One persistent agent

One agent holds the main role of the domain and starts task agents for parts of the work.

Several persistent agents

The domain defines several roles, and each role is held by its own agent.

Task agents only

No agent persists. A schedule or an event starts an agent when work arrives.

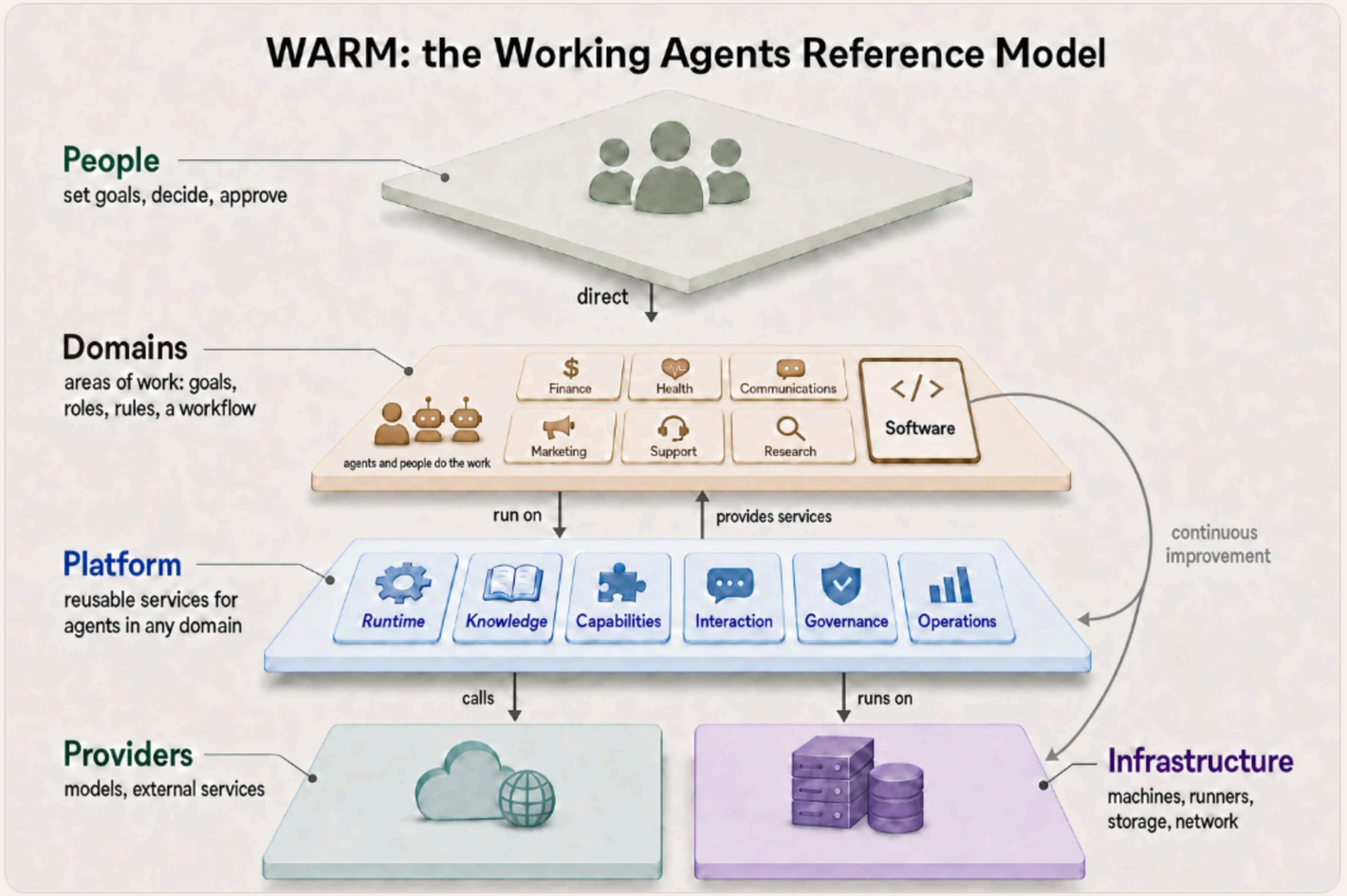
Provisional: an agent can also provide a platform service, such as routing messages or reviewing work. Its definition then comes from the operator of the service and not from a domain.

APPENDIX · A SECOND EXAMPLE

A second domain: Finance

The charter states	In the reference implementation	Commonly carried by
Goals	Where the money goes; spending against the plan	(set by people)
Responsibilities	Tracking spending, watching subscriptions, reporting where the money goes	Directory
Roles	An advisor agent gathers and analyzes; the person decides and places every order	Registry, Directory
Workflow	Conversations and scheduled jobs; no backlog found	Tasks, Scheduling, Orchestration
Domain knowledge	The accounts, the plan, past decisions	Memory, Retrieval, Registry
Domain tools	Banks, a budgeting service, a brokerage	Tools
Domain rules	The agent never moves money or a position; no job or task agent trades	Policy, Approvals
Acceptance criteria	Dated balances, recorded categories; most other judgments are the person's	Evaluation
Published skill	None yet: how to ask Finance is copied into other agents' definitions	Registry, Directory

Continuous improvement is built into the model



Every domain improves

Each domain changes its own way of working from evidence.

The platform supports it

Observability collects the evidence. Evaluation judges it. Improvement proposes a change. Releases makes it, with a way back.

Software closes the loop

Its work products are software, and so is the platform. Its agents can change the platform itself, under the same rules as any other change.

WARM v0.14 · figure: warm-domains-improvement · *Created in collaboration with AI.*

APPENDIX · A PROPOSAL

Record and review each stop for a person

1 · Record

Why the work stopped for a person: a decision, or a task an agent handed over.

2 · Review

Did it need a person? Could it have been decided in advance? What change removes it next time?

3 · Change

Remove the stop, propose a pre-approval, give agents the means, or keep it with people.

Why	So that a rule, a tool or an instruction can change and the same stop does not repeat.
Measure	Stops for people each week, and the share of decisions approved as proposed.
Not new	Of 40 sources and products read for it, one has such a review in full and fourteen have part of one.
Status	Added in version 0.14. A proposal: LifeOS does not run it yet.

APPENDIX · IMPLEMENTATIONS

Every part has implementations

WARM is abstract. A real system implements one of its parts, and the same relation applies at every layer.

Part of WARM (abstract)	An implementation (real)
A domain: Software	Agent Scrum, which carries out the domain's main work in the factory pattern
The platform	A harness and what is built around it; the three mapped platforms
Infrastructure	A server, its CI runners, backups and network
Providers	The model vendors and external services a system uses
The agents of a domain	Persistent agents and task agents, in whatever number the domain needs

Five uses of the model

Describe a system

Map each part to a layer, a service or a domain. A part that fits nowhere is a question.

Place new work

Ask whether it works the same whatever the subject. If so, which question does it answer?

Compare systems

List, per service, who supplies it and what it guarantees. A shared name is not equivalence.

Decide what agents may finish alone

For each class of action, choose reserved, pre-approved or held. Write each pre-approval with its conditions and its term.

Structure writing or code

One paper for each group. In LifeOS, one boundary for each service, as an experiment.

APPENDIX · USING THE MODEL

From concept to figure to code

The register gives selected concepts one identifier each. The model, the figures and the code refer to a concept by it.

Identifier	Kind	Defined in	Shown in	Implemented by
layer.platform	layer	Step 1	warm-top	none: a layer is not a module
agent.persistent	agent kind	Step 0	agent-kinds	none: a kind is not a module
group.governance	group	Step 2	platform-groups, group-governance	none: a group is not built
service.sessions	service	Step 3	platform-groups, group-runtime	to be assigned
service.approvals	service	Step 3	platform-groups, group-governance	to be assigned
domain.roles	domain part	Step 4	domain-supplies	agent definitions

APPENDIX · METHOD

How we built it

PUBLISHED WORK

Survey of 34 sources

Documented search: 1,634 screened, 61 met the rule, 48 new, 8 mapped

Fourteen published models compared part by part

Domain-Driven Design, Team Topologies, Data Mesh: summaries read

Autonomy checked across the models

Agent organizations, 2002 to 2004, read

SYSTEMS

Three agent platforms mapped from their documentation

LifeOS, our own system, inventoried and mapped

CHECKS

A second rater on a sample of every comparison and mapping

An outside review by a model from a different vendor

An independent re-rating under criteria written down first

Made with the system it describes. The Research and Software domains of LifeOS did the work. A person decided each change of substance.

APPENDIX · PUBLISHED MODELS

Compared with fourteen published models

1 · Two rounds

Six models, then a documented search: 61 included, 48 new, eight more mapped.

2 · Every component

Listed with its definition, then assigned to a WARM part, a span, guidance or a misfit.

3 · Predictions first

Written before each round.

Predictions	Round 1: 4 confirmed, 2 partly, 2 wrong. Round 2: 1 confirmed, 3 partly or half right, 1 wrong.
Our claims	One withdrawn, three narrowed. Areas of work and roles apart from agents: established by research on agent organizations, 2002 to 2004.
Re-rating	17 sources, criteria frozen first: 2 have an area of work; none fully specifies it apart from who does the work or records one accountable area.
What changed	A 25th service, Lifecycle. Grants, separation of duties, a stop path, kinds of human decision.

APPENDIX · PLATFORM MAPPING

How the mapping was done

1 · Blind inventories

Researchers not shown WARM listed each platform from its documentation.

2 · Predictions first

Written before any inventory existed.

3 · One mapper each

Each component to a service, layer, builder tooling, span or misfit. “Absent” needs a recorded search.

Predictions	1 confirmed, 3 partly right, 4 wrong. Platforms cover more of WARM than predicted, and have more it lacked.
What changed	Eleven changes. Directory and Improvement became services; Channels serves applications; Tasks needs no store of its own.
Second rater	36 assignments: 29 agree, 4 disagree, 3 unclear. Clean fits 12 of 12.
Threats	Raters knew WARM and worked for us. Documentation, not running systems. 36 of about 1,300.

APPENDIX · MAPPING

LifeOS, the reference implementation

849	400	288	93	61	7
components	one service	other layer (239 domain)	builder tooling	spans	misfits

Services

Neither rater found a service absent.

Present: 14 to 16. Partial: 9 to 11. A second rater first found 20 present.

Disputed: Artifacts and Approvals.

Charters and decisions

Charter rows: five of nine present, four partial. No published skill yet.

Three pre-approvals found; none stated a term. Now: does not expire; revocable; shown again when it changes.

Not independent The model was drawn partly from LifeOS. A second rater agreed on 42 of 60 sampled assignments.

APPENDIX · PLATFORM MAPPING

Amazon Bedrock AgentCore

146	92	5	10	28	11
components	one service	other layer	builder tooling	spans	misfits

What fit

Its own top-level names include Harness, Memory, Identity, Policy, Observability, Registry and Evaluations.

One service absent: Scheduling. AWS's own examples use a separate scheduler.

What did not

A program calling the agent: invoke and streaming endpoints (5 misfits). Payments (4).

Recommendations and Web Answer use a model. Policy acts only at choke points.

In WARM

Channels serves applications; Improvement added; skills to the Registry; judgment test replaced; Governance condition stated.

APPENDIX · PLATFORM MAPPING

LangGraph, LangSmith Deployment, LangSmith

307	183	22	55	30	17
components	one service	other layer	builder tooling	spans	misfits

What fit

Evaluation, Observability, Environments, Scheduling, Approvals and Models matched cleanly.

Nothing absent; seven services partial. The open-source library alone covers about four.

What did not

One checkpointer is both session history and run log (11 spans). Identity and policy per product surface.

Misfits: data lifecycle (7), tenancy, outbound webhooks (3), caching (2).

In WARM Tasks needs no store; Policy covers stored data; Scheduling covers queues; Improvement added; skills to the Registry.

APPENDIX · PLATFORM MAPPING

OpenClaw

848	329	212	256	38	13
components	one service	other layer (182 providers)	builder tooling	spans	misfits

What fit

Persistent and task agents, as in WARM.

It swaps the loop and "keeps ownership of channels, sessions, policy, and state".

What did not

Admission and capacity (3 misfits).

Programs calling the agent (4).

Plugin host, marketplace governance (3).

History

Tasks registry merged 2026-03-29, removed 2026-09-27: duplicated state.

Heartbeats became scheduled jobs, 2026-07-22 to 2026-10-22.

Improvement loop: applied by default, no approval.

In WARM Tasks needs no store; Scheduling covers admission; Channels serves programs; Improvement added.