PIKO RESEARCH TECHNICAL REPORT · PIKO-RESEARCH-2026-01

PESA: Personal Ecosystem Super Agent

Architecture, formal model, and attention economics for cross-device personal super agents.

AUTHOR: Ruiming Zhao ORCID
Prysai Lab

PUBLISHED: 2026-11-15 VERSION: 1.2 · Updated 2026-11-24
READING TIME: about 25 minutes ZENODO · CC BY 4.0: DOI: 10.5281/zenodo.24429625

ABSTRACT

Single-task language model agents can now invoke tools, write code, and drive graphical interfaces to complete isolated tasks with increasing reliability. Yet in everyday personal use, stronger models rarely translate into lower cognitive load. Because today's personal AI tools remain confined to isolated chat sessions or single-device screens, users must continuously notice incoming needs, ferry context across email, calendars, browsers, and chat windows, and watch the screen whenever an agent touches a desktop interface.

This paper defines **PESA (Personal Ecosystem Super Agent)**: a resident, single-principal super agent whose operating environment is an individual's entire device ecosystem (phone, tablet, computer, and watch), whose user-facing unit of work is the *thing* rather than the *conversation*, and which acts natively on each platform while waiting by default for explicit human confirmation before any outward side effect. Building on the resident two-layer Super-agent architecture [1], we formalise PESA across devices, event streams, threads, things, workers, and user policy, and derive a four-component human attention cost model, a Horvitz mixed-initiative speculation threshold for high-noise personal streams, a closed eight-state lifecycle with versioned optimistic concurrency control, and seven verifiable runtime invariants. We prove safety against duplicate or unapproved outward writes (Proposition 1), bounded wait and deadlock-freedom for single-desktop focus arbitration (Proposition 2), and an irreducible sovereignty lower bound on human attention (Proposition 3).

We specify a complete reference architecture—spanning a resident front desk, task-scoped background workers, a three-tier cross-device *Thing Archive*, two-stage edge-cloud filtering, an exclusive desktop lease (`ComputerHandsLease`), a rescue-first watchdog, and three deterministic-first scheduling algorithms (`route` , `choose_device` , `choose_surface`)—alongside a defence-in-depth threat model against indirect prompt injection and covert URL exfiltration. Through structured analytical workflow accounting across four representative cross-device scenarios, we show why stopping at single-device proactivity (L3) still leaves 0.50 manual context transfers per thing, whereas a complete ecosystem architecture (L4 PESA) compresses mean human interventions from 3.75 (L0) to 1.25 (-66.7%) and eliminates manual transfers (-100%). We also propose the synthetic replay benchmark **PESA-Bench**, report passing deterministic unit verification of six core runtime mechanisms in our reference system Piko, and discuss methodological limitations and open research problems.

Keywords: Personal Ecosystem Super Agent (PESA); personal AI agents; human-centred AI; multi-device systems; attention economics; mixed-initiative interaction; speculative execution; optimistic concurrency; desktop lease arbitration; indirect prompt injection

1. Introduction

Modern foundation models can read long documents, write working code, and operate graphical user interfaces. Paradoxically, giving a knowledge worker a smarter chat model often leaves them juggling *more* windows rather than fewer. Handling a single everyday matter—rescheduling a client meeting, comparing vendor quotes before filling an internal procurement form, or tracking a multi-week negotiation—still forces a person to open four or five disconnected chat tabs across a phone and a laptop. Each tab is a sealed sandbox unaware of what happened in another window or on another device. The human user becomes the glue between them: spotting the trigger, copying context into prompts, watching the screen while tools run, and pasting results back into email or enterprise forms.

The DeepSeek Harness team observed in their Super-agent report that working memory and switching costs limit a person to supervising roughly three to five concurrent chat sessions [1]. In a session-centric architecture, **the model sets the ceiling on single-turn answer quality, but human attention sets the ceiling on daily through-**

put. Classic HCI evidence explains why this bottleneck hurts so badly: in their CHI 2008 laboratory study, Mark, Gudith, and Klocke showed that interrupted knowledge workers instinctively work faster to compensate for fragmented time, finishing in similar wall-clock time only at the cost of markedly higher mental workload, stress, time pressure, frustration, and error rates [2]. Bouncing across mail clients, calendars, browser tabs, and multiple AI windows is a sustained form of self-interruption.

> **Intuition.** Smarter language models solved the problem of *how to finish a step once you explain it*. They did not solve the problem of *who notices the work, carries the context between devices, keeps background tasks from fighting over your mouse, and stops safely before pressing Send*. PESA is a systems architecture designed for that missing layer.

1.1 Four structural bottlenecks in personal AI agents

Between 2024 and 2026, personal AI tooling moved rapidly from plain chat toward tool calling, browser and desktop automation (computer-use agents), and standalone AI hardware. In sustained daily use, these approaches repeatedly encounter four structural walls:

1. **Prompt cold-start and the context-ferrying tax.** Everyday work almost never begins inside an empty chat box. It begins with a client email, a flight delay alert, a photo of an invoice on a phone, or a schedule change in a group thread. When a user must manually gather those fragments, write a prompt, and carry the answer back to the originating app, the coordination overhead frequently exceeds the effort of doing the task directly—causing daily retention to collapse.
2. **Desktop focus contention and supervision fatigue.** Computer-use agents can click buttons and fill forms, yet a personal laptop has only one physical pointer, one keyboard focus, and one active window. While an agent drives the desktop, the user is locked out of their own machine—and, fearing an accidental click, overwrite, or premature submission, stays glued to the screen as a full-time supervisor. Synchronous screen takeover binds human attention linearly to execution time.
3. **Standalone hardware islands versus existing device ensembles.** Dedicated AI pins and pocket devices attempt to bypass existing operating systems, but users already carry a tightly differentiated personal hardware ensemble: a phone in the pocket, a watch on the wrist, a tablet for reading and comparison, and a computer on the desk. What is missing is not another screen without apps or history, but a shared architecture that lets one continuous agent inhabit the devices people already own.
4. **The proactivity-versus-trust dilemma.** A purely passive agent stops working the instant a laptop lid closes or a phone locks. Yet a naive proactive agent turns hundreds of daily notifications, promotions, and chat messages into relentless interruption noise—while introducing unacceptable risk if it sends messages, submits forms, or moves money under the user's real identity without review.

1.2 Research questions and contributions

This paper is a systems architecture and vision paper guided by three research questions:

- **RQ1 (Abstraction & Attention):** What system abstraction and capability taxonomy separate an ecosystem-wide personal agent from session-bound chat tools and single-device automation, and how does each capability level change human attention cost?
- **RQ2 (Systems & Concurrency):** How can one resident agent coordinate continuous cloud reasoning with intermittent local devices, prevent concurrent background workers from colliding over a single desktop focus, and avoid duplicate outward actions across multiple screens?
- **RQ3 (Trust & Noise Discipline):** How can a proactive personal agent filter high-noise everyday streams without burning cloud tokens or annoying the user, while defending private memory and credentials against indirect prompt injection?

To answer these questions, we make five contributions:

1. **Conceptual framework and L0–L4 taxonomy (Section 3).** We formally define **PESA (Personal Ecosystem Super Agent)**, articulate three design theses grounded in human-centred AI [3] and six system properties, and establish a model-agnostic L0–L4 taxonomy measured by how much coordination burden the system absorbs.
2. **Formal model, decision calculus, and verifiable guarantees (Section 4).** We formalise PESA with orthogonal device capabilities $\kappa(d) = \langle\, \kappa_{\mathsf{exec}}(d),\, \kappa_{\mathsf{surf}}(d) \,\rangle$, derive a four-component human attention cost model and a Horvitz mixed-initiative speculation threshold under personal noise amplification, specify a closed eight-state lifecycle with versioned optimistic concurrency control [4], and prove three core safety, liveness, and attention-bound propositions.
3. **Cross-device reference architecture and deterministic-first algorithms (Sections 5–6).** We specify a resident front desk with isolated workers, a three-tier *Thing Archive* with hop-by-hop recall, checkpointed resume across device sleep, a time-bounded desktop lease (`ComputerHandsLease`) [5], a rescue-first watchdog, and three low-overhead scheduling algorithms (`route` , `choose_device` , `choose_surface`) with complexity bounds.
4. **Ecosystem-native interaction, failure-resilient scenarios, and defence-in-depth security (Sections 7–9).** We map ergonomic device roles across four form factors, trace both normal execution and fault-recovery paths across four daily workflows, fold eight internal states into a calm three-state interface, and establish a three-layer defence against indirect prompt injection and covert URL exfiltration [6].
5. **Evaluation methodology, analytical L0–L4 accounting, and reference runtime verification (Sections 10–12).** We define eight orthogonal evaluation metrics and the **PESA-Bench** replay suite, perform transparent step-by-step attention accounting across L0–L4 (pinpointing why L3 proactivity alone leaves cross-device manual transfers unresolved), report deterministic unit verification of six core mechanisms in our reference system **Piko**, and analyse limitations and threats to validity.

2. Background and related work

PESA synthesises ideas from six research traditions in AI agents, operating systems, distributed systems, and human-computer interaction. Examining what each line of work achieved—and what remains unresolved in personal daily life—motivates our architectural design.

2.1 From conversational assistants to tool and GUI agents

Early conversational assistants operated as stateless turn-by-turn loops confined to a single chat window. ReAct [7] interleaved natural-language reasoning traces with environment actions so language models could ground plans in observations, while Toolformer [8] demonstrated self-supervised tool invocation. Comprehensive surveys trace how these building blocks evolved into coding agents, browser agents, and standardised tool-connector protocols such as MCP [9].

Yet real-world personal environments remain unforgiving for standalone single-task agents. On WebArena's 812 long-horizon tasks across four self-hosted web domains [10], human completion reached 78.24% whereas GPT-4 agents achieved 14.41%. On OSWorld's 369 real desktop tasks across Ubuntu, Windows, and macOS [11], human completion reached 72.36% compared with 12.24% for the strongest multimodal agent at release. Even as visual grounding and planning models improve, everyday personal workflows face three structural barriers that model scaling alone does not solve: expired credentials and two-factor prompts across devices, critical context residing on a different device or in another application, and irreversible terminal actions (sending mail, submitting orders, authorising payments) where silent autonomous failure is unacceptable.

2.2 Multi-agent orchestration, tiered memory, and resident Super-agents

Multi-agent frameworks such as AutoGen [12] divide complex work among specialised conversational agents, but typically bind coordination to a single session lifecycle. Generative Agents [13] introduced a continuous memory stream scored by recency, importance, and relevance alongside reflection trees, though inside a closed sandbox with

no financial liability or external attackers. MemGPT [14] drew on operating-system virtual memory, treating the context window as main memory and paging archival records in and out via explicit function calls. PESA builds directly on MemGPT's cache-and-archive insight, but changes the primary unit of indexing from raw chat chunks to causally linked, cross-device *things* (`#T-id`).

The DeepSeek Harness team's Super-agent architecture [1] broke decisively out of single-session chat by introducing a resident Router layer, isolated concurrent background Workers, event-driven triggers, compute-storage separation, and an external watchdog, inspiring open-source implementations such as `deepseek-super-agent` [15] and `opensuperagent` [16]. Super-agent was engineered for a **software development team** working in enterprise chat and code repositories on always-on cloud servers. Moving this architecture into **one individual's everyday device ecosystem** introduces three fundamentally different systems constraints: (i) a **personal noise multiplier**, where over 80% of daily phone notifications, group chats, and marketing emails require no action; (ii) **intermittent device sleep and single-desktop focus contention**, where laptops close in backpacks and only one cursor exists on a personal computer; and (iii) **strict personal sovereignty**, where every outward message or transaction carries the individual's legal and social identity.

2.3 Distributed systems primitives: leases, optimistic concurrency, and calm HCI

PESA grounds its multi-device coordination in classical systems and HCI foundations rather than ad-hoc heuristics:

- **Time-bounded leases.** Gray and Cheriton [5] introduced *leases* in SOSP '89 to grant time-bounded exclusive control that recovers automatically from holder failures or network partitions without manual intervention. PESA applies this exact principle to a personal computer's physical cursor and window focus via `ComputerHandsLease`.
- **Optimistic concurrency control.** Kung and Robinson [4] showed that when conflicts are rare relative to reads, validating a version snapshot at commit time avoids heavy distributed locking. PESA applies versioned optimistic commits across phone, watch, tablet, and computer so rapid cross-device taps never trigger duplicate outward actions.
- **Calm technology, context awareness, and human-centred AI.** Weiser [17] and Weiser and Brown [18] articulated *calm technology*, in which ongoing system activity rests at the periphery of attention and moves to the centre only when needed. Dey [19] formalised *context* as any information characterising the situation of an entity relevant to user-application interaction. Horvitz [20] framed *mixed-initiative interfaces* via Bayesian expected utility, balancing the benefit of proactive help against the asymmetric cost of unwanted interruptions. Shneiderman [3] further demonstrated that high computer automation and high human control are orthogonal dimensions rather than opposite poles of a single slider—a direct foundation for PESA's "speculate deeply in the background, pause cleanly for human sign-off" paradigm.
- **Speculative execution and operating-system intent surfaces.** Modern superscalar processors execute predicted instruction paths into a sandboxed reorder buffer and commit architectural state only once the branch resolves [21]. Meanwhile, native OS frameworks such as Siri, App Intents, Shortcuts, Handoff, and ActivityKit Live Activities [22][23] provide rich cross-device surfaces, yet lack a persistent cloud-and-device state machine for multi-week asynchronous workflows. PESA treats native OS surfaces as its sensory and delivery layer rather than competing with them.

Table 1. Architectural positioning of PESA relative to existing agent paradigms

Dimension	Conversational assistant	Single-task tool / GUI agent	Team Super-agent	PESA (Personal Ecosystem Super Agent)
Primary unit of work	Isolated chat session	Single task command	Engineering event thread	**A personal *thing* (`#T-id`)**
Operating environment	Single chat window	One machine or browser sandbox	Team chat, code repos & cloud servers	**All personal devices (phone, tablet, computer, watch) & apps**

Dimension	Conversational assistant	Single-task tool / GUI agent	Team Super-agent	PESA (Personal Ecosystem Super Agent)
Initiation & proactivity	Purely passive	User-initiated, synchronous	Team members & dev pipeline events	**User intent + everyday events via two-stage edge-cloud filter**
Concurrency & responsiveness	Single window blocks	Often monopolises desktop screen	Resident router + background workers	**Responsive front desk + isolated concurrent background workers**
Offline resilience & focus mutex	Halts when tab closes	Fails on lid close; tasks fight over cursor	Always-on cloud machines; no single-GUI fight	**Cloud continuation + silent wakeup + `ComputerHandsLease` mutex**
Memory & cross-device recall	Trapped in one transcript	Scoped to one task run	Platform-local ID recall	**Three-tier memory + cross-device Archive (`#T-id` chain of recall)**
Delivery surfaces	Uniform chat list	Terminal log or floating overlay	Group chat & pull requests	**Context-matched $\varkappa_{\text{surf}}$: Dynamic Island, Watch, iPad Split View, cards**
Outward safety & credentials	Copied manually by user	Credentials often exposed in prompt	Team CI/CD auto-execution	**Sealed credential vault + scoped `brief` + default versioned approval gate**

Quantitative lessons from Super-agent and scope caveats. DeepSeek Harness reported three systems measurements that directly motivate PESA's pipeline [1]: (1) decoupling a resident router from concurrent workers lowered p95 task completion latency from 29.5 min (single-session neutral baseline) to 15.7 min (-46.8%, median 12.4 min); (2) keeping thread subscription pass-through enabled prevented read-cache token volume from jumping from 23.23M to 31.77M (+36.8%) and avoided ~30% extra router calls; and (3) adaptive backlog batching scaled batch size from 1.0 to 2.8 as arrivals rose from 1 to 40 events/min, holding per-request latency to 3.9 s (from 2.5 s) while cutting amortised routing time per event from 6.4 s to 1.6 s (-75%). Because these figures were measured on software engineering workloads, they cannot be transferred verbatim to personal life without correction: personal streams carry far higher noise density, which is why PESA adds deterministic on-device pre-filtering and per-source speculative budgets ahead of the cloud router (Section 4.2).

3. Conceptual framework and capability taxonomy

3.1 Formal definition

> **Definition 1 (PESA).** A **Personal Ecosystem Super Agent (PESA)** is a resident, single-principal super agent whose operating environment spans all of an individual's personal devices and applications. It maintains one continuous identity and shared context across devices; organises work around discrete *things* (`Thing`) rather than *conversations* (`Session`) and advances multiple things concurrently in isolated background sandboxes; filters everyday event streams through a two-stage edge-cloud funnel to prepare worthwhile work speculatively; perceives, executes, and delivers on each device in that platform's native idiom; and requires explicit, version-bound human confirmation before any action that mutates external state (except for specific low-risk actions explicitly pre-authorised in user policy).

> **Intuition.** Put plainly, a PESA behaves less like a chatbot sitting in a browser tab and more like a quiet, trustworthy executive assistant who shares access to your devices: it reads only the sources you turn on, quietly handles the research and draft preparation in the background, stays invisible when nothing matters, and taps you once on the right screen when a real decision is ready.

Each word in the name fixes a concrete systems boundary:

- **Personal (single-principal sovereignty):** Memory, calendar data, contacts, and credentials belong strictly to one person—never a multi-tenant team bot reconciling competing stakeholders.

- **Ecosystem (heterogeneous multi-device substrate):** The runtime spans complementary hardware (e.g., phone, tablet, computer, watch) plus their native OS services, rather than one isolated browser tab.
- **Super (resident, concurrent, persistent):** Extends Super-agent [1] beyond turn-by-turn sessions with continuous availability, foreground-background decoupling, event-driven proactivity, and watchdog self-healing.
- **Agent (bounded closed-loop actor):** Orchestrates cloud tools and local device executors to advance work all the way to a ready-to-approve deliverable.

3.2 Three core design theses

PESA is built on three architectural theses that align with Shneiderman's two-dimensional human-centred AI framework—simultaneously maximising background automation and preserving human control [3]:

1. **Thesis 1 — The operating environment is a device ecosystem, not a chat window.** Human digital context is physically partitioned by form factor: messaging and quick alerts live on the phone, deep documents and legacy apps live on the desktop, reading and side-by-side review live on the tablet, and in-motion glances live on the wrist. Any agent trapped in one window is structurally half-blind and shifts integration work back onto the user.
2. **Thesis 2 — The user-facing unit is the *thing*, not the *conversation*.** "New Chat" is an artefact of stateless LLM endpoints, not how people think about daily life. People care about outcomes—whether a client reschedule reply has been sent, whether a vendor comparison is ready, or what a contract clause said two weeks ago. Prompt assembly, session turns, and worker spawning belong under the hood.
3. **Thesis 3 — High background automation and strict human final authority are complementary, not opposing.** Search, cross-device synchronisation, form pre-filling, and stall recovery belong to the machine. Human attention should enter only at two high-leverage points: expressing new intent and signing off on outward side effects.

3.3 Six core system properties

PESA inherits the four foundational systems properties of a Super-agent (*continuous, efficient, proactive, persistent*) [1] and introduces two properties required specifically by personal ecosystems: **ecosystem-native** and **trustworthy** (Table 2).

Table 2. The six core system properties of PESA

Property	Architectural meaning and engineering requirement
Continuous	One continuous agent identity across every device and entry point; switching devices never drops context, and past things are recalled hop by hop via `#T-id` references.
Efficient	The resident front desk stays immediately responsive to new input while isolated background workers execute long tasks concurrently without blocking.
Proactive	Enabled everyday signals (email, calendar, watched pages) pass through two-stage filtering to trigger speculative background preparation and concise suggestion cards.
Persistent	Cloud reasoning continues when phones lock or laptops close; device-bound steps support silent wakeup or checkpointed parking; stalls trigger watchdog self-rescue before human escalation.
Ecosystem-native	Each device performs the work matched to its ergonomics, honouring platform interaction conventions (Dynamic Island, Split View, menu bar & desktop lease, wrist glance).
Trustworthy	External sources default to off and uncertain signals stay silent; outward writes wait for versioned confirmation by default; credentials remain sealed out of prompts and untrusted data cannot exfiltrate memory.

3.4 Core terminology

Table 3 establishes precise definitions for the twelve system abstractions used throughout this paper, pairing each formal role with its everyday intuition.

Table 3. Core terminology and responsibility boundaries in PESA

Term	Formal definition and responsibility boundary
Front desk	The single resident conversational coordinator (analogous to a Super-agent Router). Parses intent, splits or merges things, batches backlogs, dispatches workers, and speaks in one voice; never blocks on multi-minute tasks.
Worker	An isolated background execution unit owning exactly one thing τ. Receives only a minimal task-scoped `brief` and executes concurrently without sharing raw context with sibling workers.
Event	A discrete observation from daily life or a device: an email, instant message, calendar change, watched web update, OS share action, or direct user instruction.
Thread	A time-ordered sequence of related events sharing a business or conversational context (e.g., an email chain or ongoing procurement workflow).
Thing	The fundamental unit of user interaction and scheduling: carries a global ID (`#T-id`), origin, goal, eight-state lifecycle status, optimistic `version` , owner, required capabilities, causal `refs` , and outcome.
Subscription	A persistent binding from a worker to one or more threads ($\mathrm{sub}(w) \subseteq H$). Subsequent events on a subscribed thread bypass redundant front-desk LLM routing or reactivate a completed thing.
Thing Archive	Durable cloud store for things, threads, and tiered memory. Every entry records *fact/quote*, *auditable origin* (`#T-id`), and *confidence*, enabling cross-device hop-by-hop recall.
Device hands	Local device executors possessing execution capability κ_{exec} (e.g., desktop GUI, browser sessions, local files), arbitrated on desktops by an exclusive time-bounded lease (`ComputerHandsLease`).
Delivery surface	A platform-native user interface (κ_{surf}) that surfaces progress or a decision: inline chat pill, Lock Screen Live Activity / Dynamic Island, iPad Split View card, Mac menu bar, or Apple Watch glance.
Approval gate	The mandatory default checkpoint requiring a user's explicit, version-bound confirmation token before any outward side effect executes.
Watchdog	An independent liveness monitor outside the primary inference loop that audits real progress, injects bounded recovery prompts on stalls, and escalates honestly only when automated recovery fails.
Policy	Explicit user-configured boundaries: enabled event sources S_{on}, pre-authorised low-risk actions A_{auto}, quiet/Focus schedules Q_{quiet}, and per-source daily speculative quotas B_{spec}.

3.5 PESA capability levels (L0–L4)

Drawing on the SAE J3016 automation taxonomy [24], we define five architectural levels (L0–L4) based on **how much session management, proactive filtering, and cross-device coordination the system shoulders** (Figure 1, Table 4). Each level strictly subsumes the levels below it.

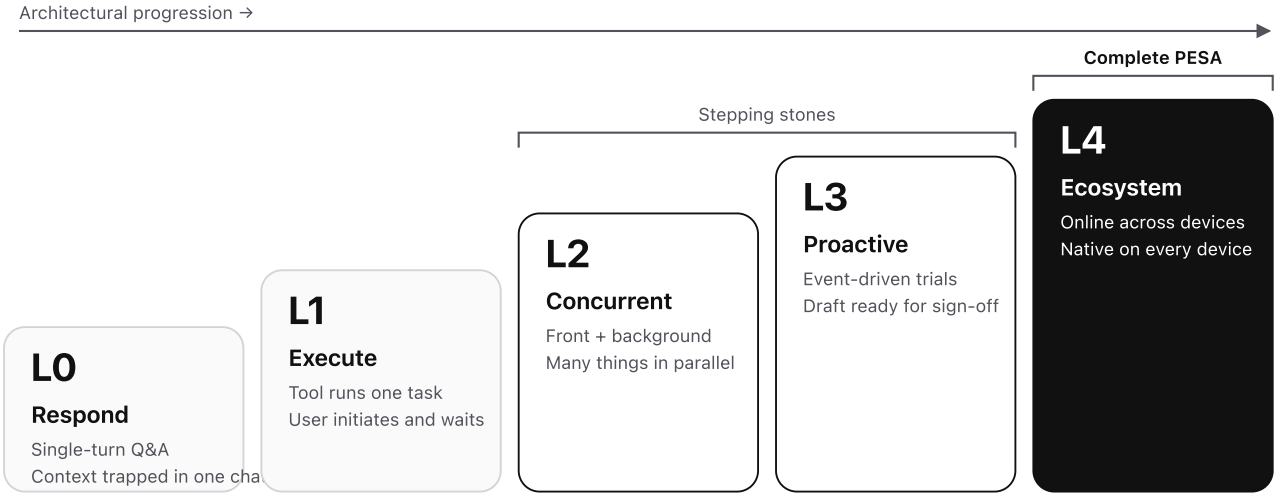


Figure 1. PESA capability taxonomy (L0–L4). Each level strictly subsumes the levels below; only L4 constitutes a complete PESA.

Table 4. PESA five-level capability taxonomy

Level	Name	Architectural criterion	Human role and attention burden
L0	Respond	Synchronous turn-by-turn Q&A; no cross-session workflow state; context stays trapped in one chat box.	User notices needs, writes prompts, copies context across apps, and executes every external action manually.
L1	Execute	Calls tools to finish a single task inside one device or sandbox, initiated and waited on synchronously.	User avoids some manual clicking within one app, but still initiates work, transfers data across devices, and supervises execution.
L2	Concurrent	Resident front desk + isolated background workers: long work executes asynchronously while chat stays free for new input.	User no longer blocks on a single task screen, but must still notice every everyday trigger and instruct the agent manually.
L3	Proactive	Connects to everyday event streams with two-stage filtering and speculative drafting; delivers suggestion cards and stays silent when uncertain.	User avoids discovering routine tasks, but single-device or web-only systems still leave manual cross-device transfers unresolved.
L4	**Ecosystem**	**Unified identity and `#T-id` Archive across all personal devices; cloud continuation across device sleep with silent wakeup, desktop lease arbitration, checkpointed resume, native surfaces, and atomic multi-device dismissal.**	**User is freed from context ferrying and screen babysitting, intervening only to express new intent or confirm outward decisions.**

Under this taxonomy, **only L4 constitutes a complete PESA**. Two grading principles prevent marketing inflation: (1) *levels measure system architecture, not base-model size*—a frontier reasoning model packaged inside a single-turn chat box remains L0; and (2) *vertical prototypes do not define global level*—an agent with proactive monitoring on one narrow web source exhibits L3 behaviour on that pathway, while its overall level depends on whether its unified cross-device state machine spans the user's ecosystem.

4. Formal system model, decision calculus, and guarantees

This section formalises PESA's runtime state, derives its attention-cost and mixed-initiative decision boundaries, defines its closed lifecycle state machine, and proves three foundational safety, liveness, and attention-bound properties.

4.1 System entities and orthogonal capability decomposition

We model a running PESA instance as a six-tuple $S_{\text{PESA}} = \langle D, E, H, T, W, P \rangle$ governed by a resident front-desk scheduling operator F:

- **Device ensemble $D = \{d_1, \ldots, d_n\}$.** Each device $d \in D$ exposes an orthogonal pair of capabilities $\kappa(d) = \langle \kappa_{\text{exec}}(d), \kappa_{\text{surf}}(d) \rangle$, separating **what the device can execute** (browser sessions, native desktop apps, local files, on-device connectors) from **how the device can reach the user** (foreground chat, actionable notifications, Dynamic Island / Live Activities, tablet split-view panels, menu bar, wrist glances). Its time-varying power and connectivity state is $\alpha(d, t) \in \{\text{online, wakeable, offline}\}$.
- **Event stream $E = (e_1, e_2, \ldots)$.** Each event is a tuple $e = \langle \text{id, source, } t\text{, payload, } h \rangle$, where source identifies the user, an external communication channel, or a device transition, and $h \in H$ links it to a thread.
- **Threads H.** Each thread $h \in H$ orders related events sharing a persistent conversational or operational context.
- **Things T.** Each thing is an eleven-tuple:

$$\tau = \langle \text{id, origin, goal, state, wait_reason, version, owner, subs, req_exec, refs, outcome} \rangle \tag{1}$$

Here id is a global reference (`#T-108`); version $\in \mathbb{N}^+$ is a monotonically increasing optimistic-concurrency version counter; wait_reason $\in$ {none, approval, device_offline, intervention} separates ready drafts awaiting confirmation, device-offline parking, and fault escalation; and refs $\subset T$.id stores directed causal links to prior things for cross-device recall.

- **Workers W.** Each worker $w \in W$ owns exactly one thing τ, receives only a minimal task-scoped brief brief(τ), and subscribes to threads sub(w) $\subseteq H$. Crucially, **threads H and things T form a many-to-many mapping**: one complex thing (e.g., coordinating a trip) can subscribe simultaneously to email, calendar, and flight-alert threads, whereas a single user message containing three independent requests is split by F into three independent things τ_1, τ_2, τ_3.
- **User policy $P = \langle S_{\text{on}}, A_{\text{auto}}, Q_{\text{quiet}}, B_{\text{spec}} \rangle$.** Every boundary defaults safely: $S_{\text{on}} = \varnothing$ (no external source read until enabled), $A_{\text{auto}} = \varnothing$ (no outward write auto-approved unless allowlisted), Q_{quiet} encodes Focus and quiet-hour rules, and B_{spec} bounds daily speculative trials per source.

The resident front desk F is a deterministic-rule-first, model-fallback transition operator:

$$F(e, T, P) \longrightarrow \{\text{reply, create, merge, split, reactivate, stop, silent}\}$$

> **Intuition.** Separating κ_{exec} from κ_{surf} is what makes cross-device collaboration click: *your desk Mac might be the best device to fill a web form (κ_{exec}), while your wrist or tablet might be the best screen to ask for your final confirmation (κ_{surf}).*

4.2 Human attention cost and mixed-initiative threshold calculus

To compare agent architectures quantitatively rather than by anecdote, we decompose the human operational attention cost $A(\tau)$ required to complete a thing τ into four orthogonal components:

$$A(\tau) = I_{\text{init}}(\tau) + I_{\text{transfer}}(\tau) + I_{\text{supervise}}(\tau) + I_{\text{decide}}(\tau) \tag{2}$$

where I_{init} counts discovery and prompt-initiation steps, I_{transfer} counts manual copy-paste or cross-device context transfers, $I_{\text{supervise}}$ counts synchronous screen-watching or mid-course manual rescues, and I_{decide} counts final review and sign-off taps.

A naive proactive agent could game Eq. (2) by spamming speculative suggestions for every notification. To close that loophole, let T_{done} denote completed things and T_{skip} denote proactive suggestion cards surfaced to the user and dismissed via *Skip* at rejection cost $I_{\text{reject}}(\tau') \geq 1$. We define the **amortised effective attention cost**:

$$\bar{A} = \frac{\sum_{\tau \in T_{\text{done}}} A(\tau) + \sum_{\tau' \in T_{\text{skip}}} I_{\text{reject}}(\tau')}{|\, T_{\text{done}} \,|} \tag{3}$$

Mixed-initiative gating under personal noise amplification. Following Horvitz's Bayesian expected-utility criterion [20], let $P(G \mid E)$ denote the inferred probability that an incoming event E reflects an actionable user goal G, U_{save} the human attention utility saved when background preparation hits, C_{compute} the background token cost of a speculative trial, and $C_{\text{interrupt}}$ the cognitive cost of a false interruption. PESA enforces a **two-threshold gate**:

$$P(G \mid E) > \frac{C_{\text{compute}}}{U_{\text{save}}} \text{ (admit trial)}, \qquad P(G \mid E) > p^* = \frac{C_{\text{interrupt}}}{C_{\text{interrupt}} + U_{\text{save}}} \text{ (surface card)} \tag{4}$$

Because cloud inference is far cheaper than fracturing human focus ($C_{\text{compute}} \ll C_{\text{interrupt}}$), the system can safely run quiet, side-effect-free background trials at a lower confidence threshold, then discard low-value trials silently before they ever cross p^* to touch a user screen.

Equally important is the **personal noise multiplier**. Suppose a fraction $\rho \in [0, 1)$ of inbound personal events is non-actionable noise (promotions, routine group banter, OTP codes). If every raw event hits a cloud LLM router, the ratio of total routed events to genuinely actionable events is $\frac{1}{1 - \rho}$. At typical personal notification noise levels ($\rho \geq 0.80$), skipping Stage-1 deterministic on-device pre-filtering inflates cloud routing traffic and token spend by at least $5\times$ (and at $\rho = 0.90$, by $10\times$), while exposing far more private text to cloud servers. Placing zero-token deterministic rules ahead of semantic filtering eliminates the bulk of ρ locally before a single cloud token is spent.

4.3 Closed eight-state lifecycle and optimistic concurrency commit

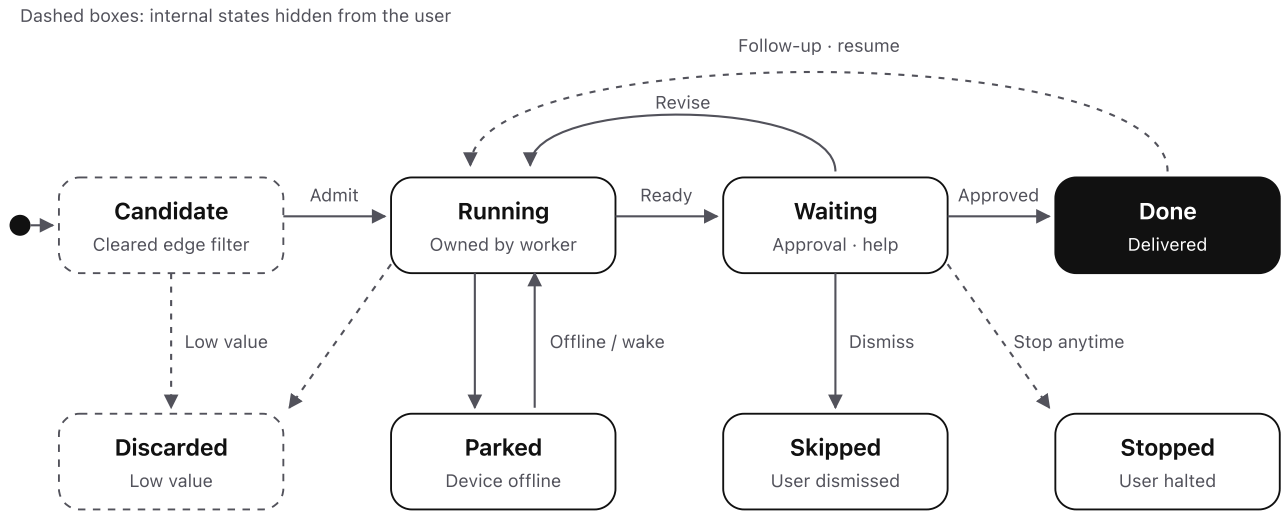


Figure 2. Closed eight-state lifecycle of a thing. Outward writes must pass through Waiting (approval); dashed states remain invisible to the user, while Skipped explicitly accounts for dismissed suggestion cards.

To handle noise filtering, sleeping devices, outward approval gates, stall recovery, and multi-week follow-ups without dangling states, PESA maintains eight strictly defined internal states that fold into three calm user-facing views (Figure 2, Table 5).

Table 5. Closed eight-state lifecycle and user-facing view mapping

Internal state	System semantics and residency condition	Legal transitions	User-facing view
`Candidate`	Cleared Stage-1 on-device rules; undergoing Stage-2 semantic value check	Worth doing → `Running`; uncertain/low value → `Discarded`	Hidden (silent)
`Running`	Owned by an isolated worker w in cloud or holding a local device lease	Ready / outward action → `Waiting (approval)`; low value → `Discarded`; required device offline → `Parked`	**In progress**
`Parked`	Next step needs capability req_exec on an offline device or awaits an external condition	Device online/woken → `Running`; user cancels → `Stopped`; unrecoverable timeout → `Waiting (intervention)`	**In progress** (with wait note)
`Waiting`	Blocked on the user for one of two reasons: (a) **approval** of a ready outward action, or (b) **intervention** after missing capability or 3 failed watchdog rescues	(a) Approve → `Done`, Edit → `Running`, Skip → `Skipped`; (b) Unblock & retry → `Running`, Stop → `Stopped`	**Waiting for you** (routed to 1 surface)
`Done`	Goal delivered or confirmed outward action executed	Retained thread subscription $\mathrm{sub}(w)$ triggers **reactivation** (`Done` → `Running`) or chains `#T-id` on follow-up	**Done**
`Discarded`	Eliminated silently by the system during filtering or trial before reaching any screen	Internal terminal state (consumes B_{spec}; produces zero alerts)	Hidden (silent)

Internal state	System semantics and residency condition	Legal transitions	User-facing view
`Skipped`	Card reached a user surface and was explicitly dismissed via *Skip*	Internal terminal state (penalises $\bar{A}$ and false-interruption rate; down-weights similar signals)	Archived
`Stopped`	User explicitly halted the thing from `Running`, `Parked`, or `Waiting`	Internal terminal state (releases held device leases and compute immediately)	Stopped (archived)

Three transition axioms make the state machine sound:

1. **Axiom 1 (Mandatory Approval Gate):** No direct transition exists from `Candidate` or `Running` to outward side-effect execution and `Done` for any action $a \notin A_{\text{auto}}$; execution must pass through `Waiting (approval)`.
2. **Axiom 2 (Non-Falsifiable Interruption Accounting):** Trials dropped silently before reaching any surface enter `Discarded`; any card shown to the user and dismissed must enter `Skipped`. Implementations cannot conceal annoying interruptions as silent discards.
3. **Axiom 3 (Post-Completion Context Continuity):** Entering `Done` closes the current turn without purging the Archive entry or thread subscription, allowing follow-up replies days later to reactivate or chain `#T-id` seamlessly.

Multi-device optimistic concurrency commit rule. In a four-device ecosystem, a user may tap *Approve* on an Apple Watch and, half a second later, tap the same notification on an iPhone. Adopting Kung and Robinson's optimistic validation principle [4], every surfaced card carries snapshot version $v = \tau.\text{version}$, and the cloud state machine evaluates an atomic compare-and-increment commit:

$$\text{commit}(\tau,\, v,\, \text{action},\, d) = \begin{cases} \langle\ \text{OK},\ \tau.\text{version} \leftarrow v+1,\ \text{withdraw}(D \setminus \{d\})\ \rangle & \text{if } \tau.\text{state} = \text{Waiting} \wedge \tau.\text{version} = v \\ \langle\ \text{NOOP_ALREADY_HANDLED},\ \tau.\text{version}\ \rangle & \text{otherwise} \end{cases} \tag{5}$$

Only the first arriving confirmation with matching snapshot v executes the outward side effect and increments τ.version; any lagging tap fails version validation cleanly as a no-op while cloud push withdraws the stale prompt from all remaining devices.

4.4 Seven runtime invariants and formal properties

Every compliant PESA runtime must uphold seven system invariants at all times:

1. **One voice:** Only the resident front desk F speaks to the user; worker spawning, retries, and lease queuing stay transparent.
2. **Approval before outward side effects:** Every action $a \notin A_{\text{auto}}$ mutating external state requires a user-confirmed token bound to τ.version.
3. **Verifiable provenance:** Every thing τ and Archive entry retains an auditable chain (`origin`, `refs`) back to its source event or instruction.
4. **Quiet by default:** External sources default to off ($S_{\text{on}} = \varnothing$); uncertain candidates stay silent and `Discarded` work emits zero alerts.
5. **Rescue first, escalate transparently:** Watchdog stall detection attempts bounded automated recovery (up to 3 rescues) before moving to `Waiting (intervention)` with plain diagnostics, never silently cancelling user-enabled watches.
6. **Sealed credentials and data-instruction separation:** Secrets travel only between an encrypted vault and low-level executors, never entering model prompts; workers receive only minimal scoped `brief`s, and external text is strictly data.

7. **Idempotent intake and synced single-surface dismissal:** Inbound events deduplicate by `e.id`; pending decisions alert on at most one surface at a time and clear atomically across all devices upon resolution.

From these axioms and runtime rules follow three core guarantees:

Proposition 1 (Outward Side-Effect Safety and Idempotency). Under Axiom 1, Invariant 2, and the atomic commit rule in Eq. (5), any non-allowlisted outward action $a \notin A_{\text{auto}}$ executes **zero times** without explicit user confirmation and **at most once** even under concurrent multi-device taps or network retries.

Proof Sketch. By Axiom 1 and Invariant 2, the low-level executor rejects any outward write lacking a confirmation token bound to $(\tau.\text{id}, v)$ signed while $\tau.\text{state} = \text{Waiting}$. Thus zero unapproved executions can occur. When multiple devices $d_1, d_2 \in D$ concurrently submit confirmations for snapshot v, the linearisable cloud state machine executes Eq. (5) atomically: the first request observes $\tau.\text{state} = \text{Waiting} \wedge \tau.\text{version} = v$, transitions $\tau.\text{version} \leftarrow v + 1$, and dispatches a once. Every subsequent request with version v fails the equality guard and returns `NOOP_ALREADY_HANDLED`. □

Proposition 2 (Desktop Lease Deadlock-Freedom and Bounded Priority Wait). Under time-bounded single-holder lease semantics (`maxRunning` $= k$, foreground priority preemption, and idle auto-reclamation `idleRelease` $= T_{\text{idle}}$), `ComputerHandsLease` is free of permanent desktop focus deadlock, and a foreground user desktop request acquires the lease within at most one active atomic holder interval (bounded above by T_{idle} if the current holder stalls).

Proof Sketch. At any instant at most one entity holds the exclusive lease. Because every lease is time-bounded by T_{idle} (180 s in our reference runtime) following Gray and Cheriton [5], a crashed, disconnected, or stalled holder cannot retain desktop control indefinitely; the runtime reclaims the lease automatically and hands it to the head of the queue. Because live foreground requests preempt all queued background workers (`Queue #1`), a foreground request waits behind at most the single current holder rather than all k background tasks, preventing starvation and focus deadlock. □

Proposition 3 (Sovereignty Lower Bound on Human Attention). For any thing τ terminating in an outward action $a \notin A_{\text{auto}}$, any PESA-compliant system satisfies $A(\tau) \geq 1$, with equality achieved if and only if $I_{\text{init}}(\tau) = I_{\text{transfer}}(\tau) = I_{\text{supervise}}(\tau) = 0$ and $I_{\text{decide}}(\tau) = 1$.

Proof Sketch. By Invariant 2, at least one explicit human decision event is required before a commits, so $I_{\text{decide}}(\tau) \geq 1$. Since all four components in Eq. (2) are non-negative integers, $A(\tau) \geq I_{\text{decide}}(\tau) \geq 1$. Consequently, once final human authority is preserved, the *only* architectural levers for reducing $\bar{A}$ in Eq. (3) are driving I_{init}, I_{transfer}, and $I_{\text{supervise}}$ to zero while keeping false interruptions $|\mathrm{T}_{\text{skip}}|$ minimal. □

5. Reference architecture

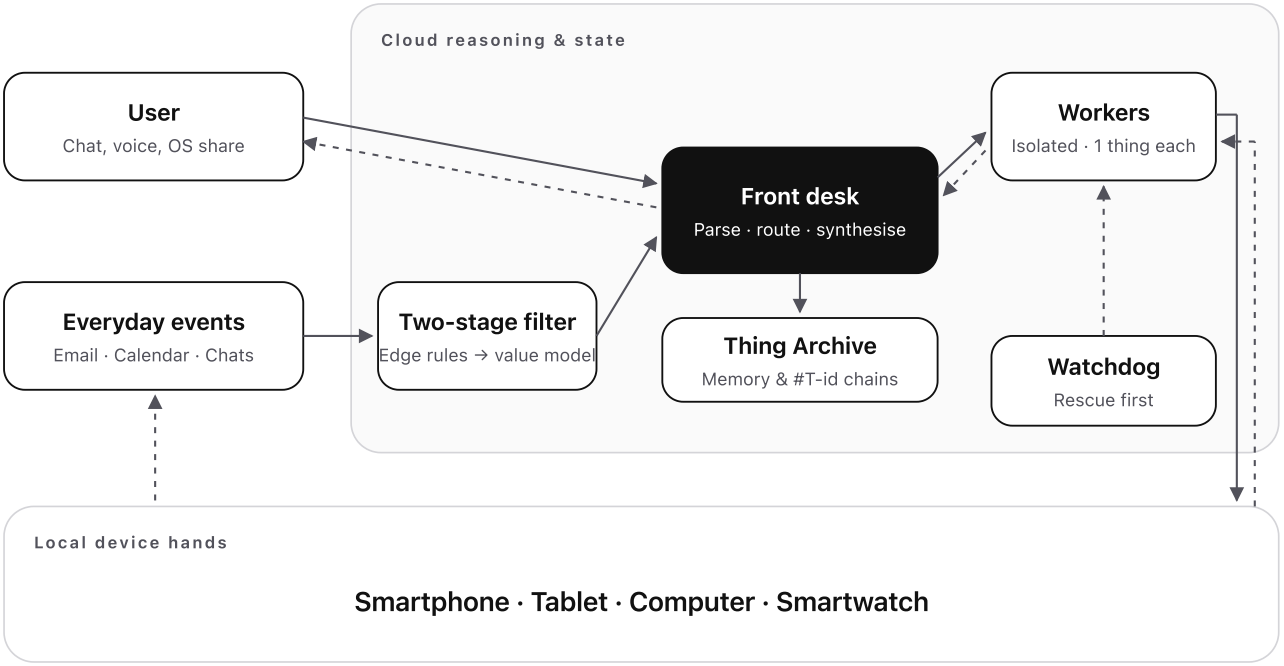


Figure 3. PESA reference architecture. Direct user input and filtered life signals enter through one resident front desk, which dispatches isolated background workers operating through local device hands and responds in a single voice.

5.1 Resident front desk and isolated background workers

Figure 3 shows the overall PESA reference architecture. Extending the two-layer Super-agent structure [1], the **front desk** acts as a lightweight, always-on coordinator receiving every user message, OS share handoff, and filtered life event across all devices. Its scope is deliberately narrow: parse intent, split or merge things, batch bursts of related events, dispatch background workers, and report results in a single voice. It never blocks on multi-minute browsing or desktop automation.

Background workers are spawned per thing. Each worker owns exactly one τ, runs inside an isolated context holding only a minimal task-scoped `brief`, and executes concurrently with sibling workers. When bursts of events arrive on the same thread (e.g., rapid project chat replies or multiple shipping notifications), adaptive backlog batching groups them into one routing decision, cutting amortised routing latency by up to 75% [1].

5.2 Three-tier memory hierarchy and the cross-device Thing Archive

Existing assistants suffer from "device amnesia" when a user switches from laptop to phone—or forget constraints agreed upon weeks earlier—because they either trap history inside one chat transcript or flatten it into disconnected vector chunks that sever causal provenance. Drawing on virtual memory paging and MemGPT [14], PESA organises memory into three tiers anchored by a cloud **Thing Archive** (Figure 4):

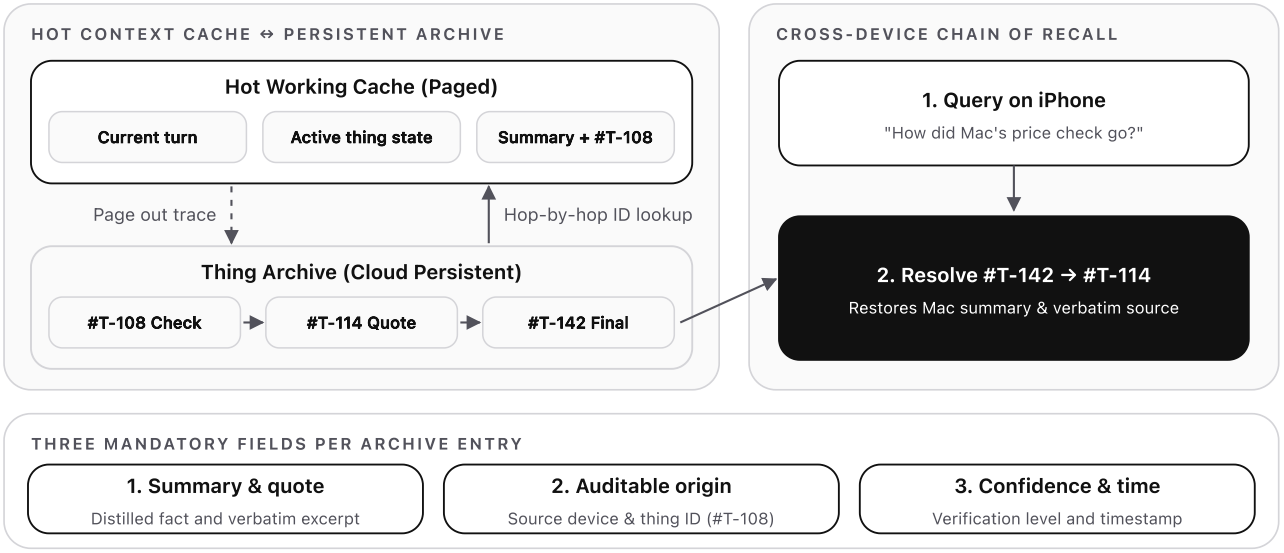


Figure 4. Three-tier memory hierarchy and cross-device Thing Archive. Verbose execution traces page out to durable cloud storage keyed by #T-id so any device recovers prior conclusions and verbatim provenance hop by hop.

1. **Tier 1 — Hot working context cache.** Lives inside the active front desk and running workers, holding only recent turns, compact summaries of active things, and lightweight ID pointers (`#T-108`, `#T-114`). Once a thing finishes, verbose traces (DOM trees, raw search outputs, intermediate drafts) page out immediately to the cloud Archive so main memory remains compact and fast.
2. **Tier 2 — Episodic Thing Archive with hop-by-hop chain of recall.** Persists every thing durably in the cloud with three mandatory fields: *(i) factual summary and verbatim key quotes*, *(ii) auditable provenance* (triggering device, raw source, `#T-id`), and *(iii) confidence level and timestamp*. Directed `refs` edges link causally related work (e.g., `#T-108 comparison` → `#T-114 vendor quote` → `#T-142 final approval`). When a user asks on a phone during a commute about a warranty quote checked on their Mac a week earlier, the front desk traverses `#T-142` → `#T-114` → `#T-108` hop by hop to load the exact record on demand.
3. **Tier 3 — Inspectable preference and policy memory.** Distils stable preferences from user card decisions (*Send*, *Edit*, *Skip*) to calibrate filter thresholds. Every learned preference remains human-readable, editable, and deletable with one tap from any device.

5.3 Asynchronous full-duplex interaction

Any task lasting longer than a few seconds executes asynchronously. If a user asks the agent to extract cash-flow comparisons from three PDF reports *and* watch whether an evening flight is delayed, the front desk decomposes both intents within a second, dispatches two workers (`#T-201` and `#T-202`), inserts two compact live progress lines into the conversation, and immediately returns to ready state. The user can continue chatting or lock their phone; finished outcomes flow back into the conversation or route silently to an ambient surface, and any in-flight thing remains inspectable or stoppable at any moment.

5.4 Cloud brain, local device hands, checkpointed resume, and desktop leases

PESA strictly decouples reasoning, durable state, and local execution: the reasoning "brain" runs statelessly in the cloud, the eight-state machine and Archive persist independently, and local "hands" execute on the user's own devices. Making this split dependable in daily life requires solving two problems unique to personal hardware:

1. **Mid-execution sleep and checkpointed idempotent resume.** Personal laptops close mid-task when users walk into meetings. PESA structures device-side automation into **checkpointed idempotent read/prepare steps** followed by a **terminal commit boundary**. On disconnect, completed checkpoints sync to the cloud: cloud-executable steps (synthesis, price comparison, drafting) continue uninterrupted, while GUI-bound steps park cleanly (`Parked`). Upon reconnect or silent push wakeup, the local executor verifies the target win-

dow or DOM fingerprint against the saved checkpoint before resuming; if external state drifted ambiguously, it refuses to click blindly and escalates to `Waiting (intervention)`.

2. **Single-desktop focus mutual exclusion (`ComputerHandsLease`).** Ten cloud workers can reason in parallel, yet a personal computer has only one mouse cursor and one keyboard focus. Applying Gray and Cheriton's time-bounded lease abstraction [5], `ComputerHandsLease` enforces four deterministic rules (Figure 5): (i) **single-holder mutual exclusion** across at most `maxRunning = 3` concurrent background workers; (ii) **foreground chat preemption**, jumping live user desktop requests straight to the head of the queue; (iii) **cloud-first non-blocking progress**, letting queued workers finish cloud research and drafting first and parking cleanly after `maxWait = 600s`; and (iv) **180-second idle auto-reclamation** (`idleRelease = 180s`), reclaiming stalled leases automatically and notifying the watchdog.

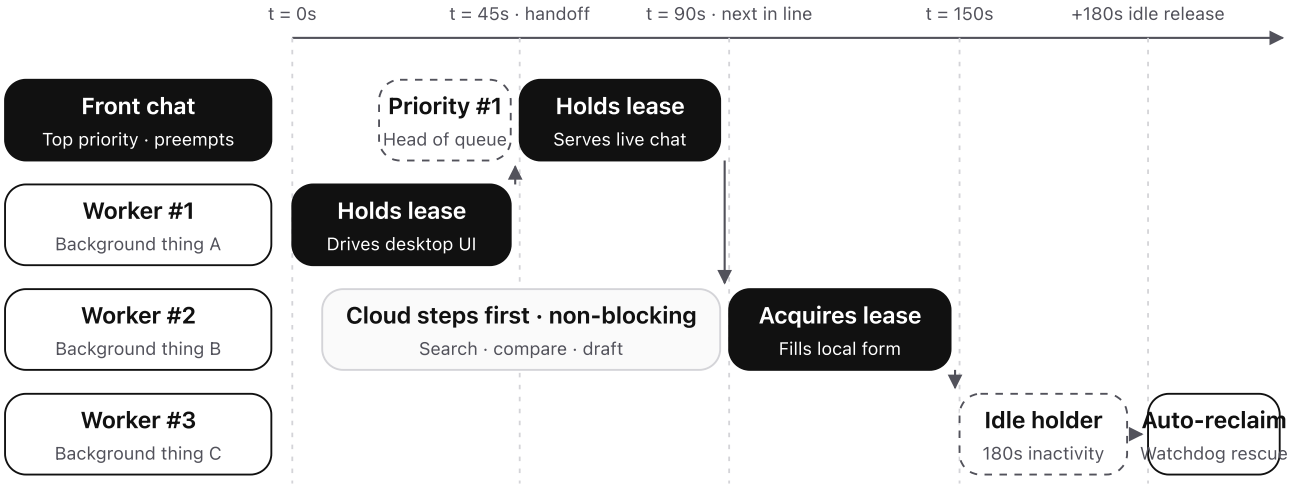


Figure 5. Exclusive desktop hands lease (ComputerHandsLease) timeline. At most one execution entity controls local GUI focus at a time: live foreground chat preempts queued background workers, waiting workers advance cloud steps first, and 180 s idle leases reclaim automatically.

5.5 Independent watchdog: rescue first, never drop work silently

Network glitches, expired sessions, and changed web layouts inevitably stall long-running tasks. PESA isolates liveness monitoring in an external **watchdog loop** that checks substantive state progress rather than mere process heartbeats. In our reference runtime, the watchdog sweeps active workers every 45 seconds: after 20 minutes without progress, it injects a diagnostic recovery prompt guiding path-switching or sub-environment reset (up to 3 automated attempts). Only after 60 minutes of stall or 3 failed rescues does it transition the thing to `Waiting (intervention)` with a plain-language diagnostic—never silently cancelling user-enabled watches.

6. End-to-end pipeline, scheduling algorithms, and complexity

This section specifies the six-stage pipeline and three deterministic-first algorithms that move a signal from intake to a user surface.

6.1 Six-stage event-to-thing pipeline and routing algorithm

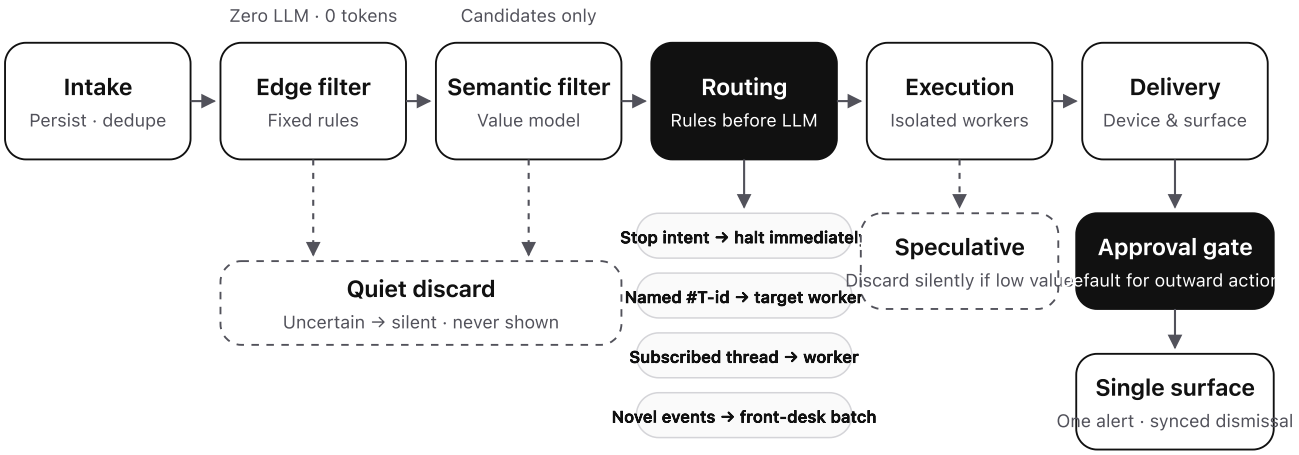


Figure 6. Six-stage event-to-thing pipeline. Deterministic rules decide before invoking language models, uncertain signals expire silently, and outward writes halt at the versioned approval gate.

Every inbound event e passes through six sequential stages (Figure 6):

1. **Idempotent intake:** Persist before acknowledging; deduplicate by `e.id` and drop echoes of the system's own outward writes.
2. **Stage-1 deterministic edge filter:** Apply zero-LLM rules locally (source allowlist S_{on}, contact/past-reply graph, calendar patterns, explicit watch rules) so >80% routine noise expires on-device with zero cloud token spend and zero unnecessary upload of private text.
3. **Stage-2 semantic value filter:** Evaluate only Stage-1 survivors against Eq. (4); uncertain or low-value candidates transition silently to `Discarded`.
4. **Deterministic-first routing:** Handle stop commands, explicit `#T-id` references, and subscribed thread pass-through without invoking the front-desk LLM (preserving the ~36.8% read-cache token savings measured in [1]); only novel events enter adaptive front-desk batching.
5. **Isolated execution and speculative preparation:** Workers perform read-only lookups, comparisons, and form pre-filling in parallel, halting cleanly before any outward write boundary.
6. **Version-bound delivery:** Bind the prepared suggestion card to τ.version and route it to a single context-matched surface.

Algorithm 1 specifies deterministic-first event routing `route(e)`:

Algorithm 1 · Event Routing · route(e)

```
route(e):
  if seen(e.id) or is_echo(e):            # 1. O(1) idempotent dedupe & echo drop
    return
  if e is a stop command:                 # 2. zero-LLM immediate halt
    stop(target_thing); return
  if e references thing τ (#T-id):        # 3. O(1) direct thing lookup
    deliver(e, τ.owner); return
  if worker w subscribes to e.thread:        # 4. O(1) subscription pass-through (-36.8% cache tokens)
    if w.thing.state == done:
      reactivate_or_chain(w.thing, e); return
    deliver(e, w); return
  front.batch(e)                          # 5. adaptive backlog batching for novel events
```

Complexity and cost profile. Steps 1–4 execute in $O(1)$ average time via hash-indexed deduplication, thing-ID, and thread-subscription tables with **zero LLM invocations**. Model calls occur strictly in Stage-2 filtering and Step 5 batched front-desk triage.

6.2 Mixed-initiative speculative execution

In superscalar processors, speculative execution runs instructions ahead of unresolved branches inside a sandboxed reorder buffer and commits architectural state only on a branch hit, flushing silently on a miss [21]. PESA applies the exact same contract to personal tasks under Horvitz's expected-utility gate [20]:

- **Side-effect-free speculation:** Workers check calendars, pull prior Archive notes, compare prices, and pre-fill draft replies or form fields strictly in read-only or draft state.
- **Three-choice commit card (*Approve / Edit / Skip*):** Pausing before the outward boundary presents one scannable card. *Approve* commits the prepared action in under a second (`Done`); *Edit* accepts a short adjustment and returns to `Running` with full context intact; *Skip* leaves the outside world untouched (`Skipped`) while lowering speculative priority for similar future signals. Trials found unhelpful before surfacing flush silently to `Discarded` . Per-source daily budgets B_{spec} cap worst-case background compute.

6.3 Orthogonal scheduling: acting device (κ_{exec}) and delivery surface (κ_{surf})

Single-device assistants collapse execution and presentation onto one screen. Grounded in Dey's context-aware computing framework [19], PESA decouples them into two independent $O(|\ D\ |)$ deterministic algorithms (since a personal ensemble typically has $|\ D\ | \leq 10$, both execute in sub-millisecond time without LLM overhead).

Algorithm 2 (`choose_device(τ)` **)** selects the optimal device capable of executing the next step:

Algorithm 2 · Execution Device Selection · choose_device(τ)

```
choose_device(τ):
  capable = { d ∈ D | τ.req_exec ⊆ κ_exec(d) }
  usable  = { d ∈ capable | α(d, now) ∈ {online, wakeable} }
  if usable is not empty:
    d* = argmin_{d ∈ usable} load_and_rtt(d)
    if α(d*, now) == wakeable:
      wake(d*)                              # silent push wakeup & local lease grant
    return d*
  if capable is not empty:
    τ.state = parked
    τ.wait_reason = "device_offline"
  else:
    τ.state = waiting
    τ.wait_reason = "intervention: missing required device capability"
```

Algorithm 3 (`choose_surface(τ, D, P, now)` **)** selects **at most one** delivery surface respecting user focus, quiet hours Q_{quiet}, and time sensitivity (Table 6, Figure 7):

Algorithm 3 · Delivery Surface Selection · choose_surface(τ, D, P, now)

```
choose_surface(τ, D, P, now):
  if τ.state == running:
    return ambient_progress(D)              # inline pill + Dynamic Island + menu bar
  if τ.state == waiting:
    if user is actively chatting on device d_chat:
      return inline_card(d_chat, τ.version)
    if now ∈ P.Q_quiet and not τ.is_urgent_override:
      return queue_silent_card(τ.version)   # quiet inbox logging; zero screen wake
    if τ.is_time_sensitive and Watch ∈ D is online and worn:
      return watch_glance_card(Watch, τ.version)
    if iPhone ∈ D is online:
      return actionable_notification(iPhone, τ.version)
    return next_open_device_card(D, τ.version)
```

Table 6. Context-aware delivery surface selection and interruption discipline

User context & thing state	Selected surface (κ_{surf})	Interruption profile & cross-device sync
Actively conversing in chat	Inline card or reply inside the active chat stream	Zero context switch; confirm right where attention already rests
Background work in progress	Inline progress pill; iPhone Dynamic Island & Live Activity [22]; Mac menu bar	Calm periphery [18]; no pop-up or sound; tap anytime to inspect or stop
Ready for decision, non-urgent	Suggestion card queued for the next opened client	Zero immediate disruption; batch-reviewed calmly on tablet or computer
Ready for decision, time-sensitive	Apple Watch glance confirmation card or iPhone actionable notification	Single light haptic alert; versioned commit on one device withdraws alerts on all others
Focus mode or quiet hours	Silent logging into *Waiting for you* inbox under Q_{quiet}	Strictly zero interruption unless matched by an explicit user-defined urgent override

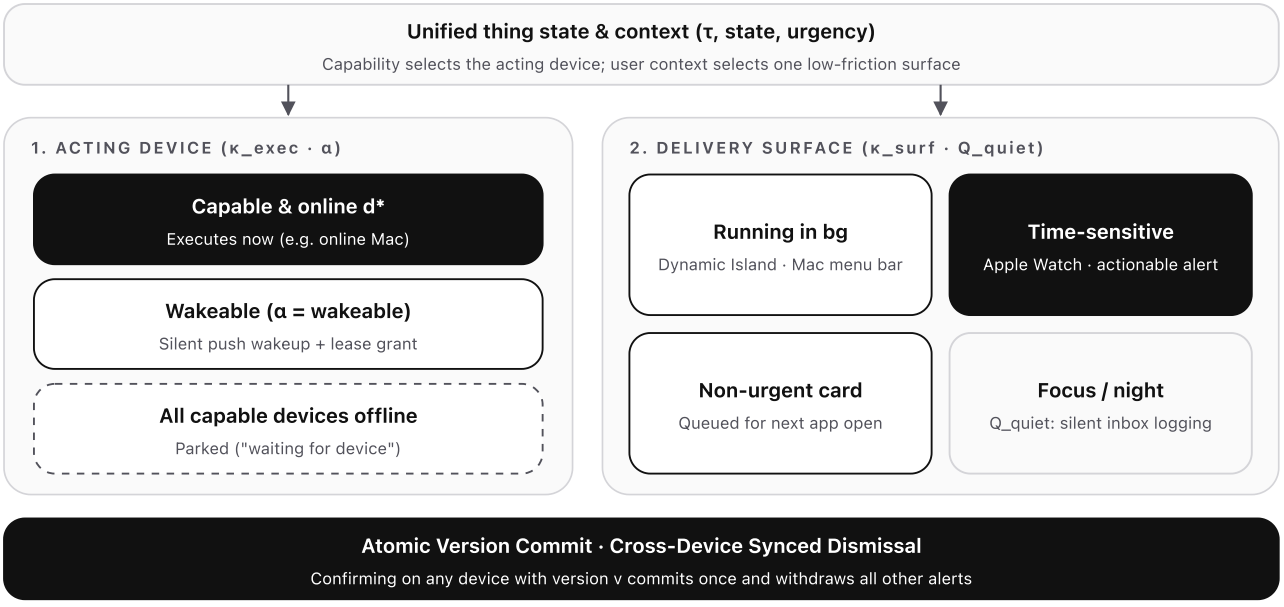


Figure 7. Orthogonal scheduling of acting device (κ_exec, α) and delivery surface (κ_surf, Q_quiet). Capability and availability select the execution device, while user focus and quiet policy choose at most one alert surface with synced multi-device dismissal.

6.4 End-to-end walkthrough: rescheduling while commuting with a closed laptop

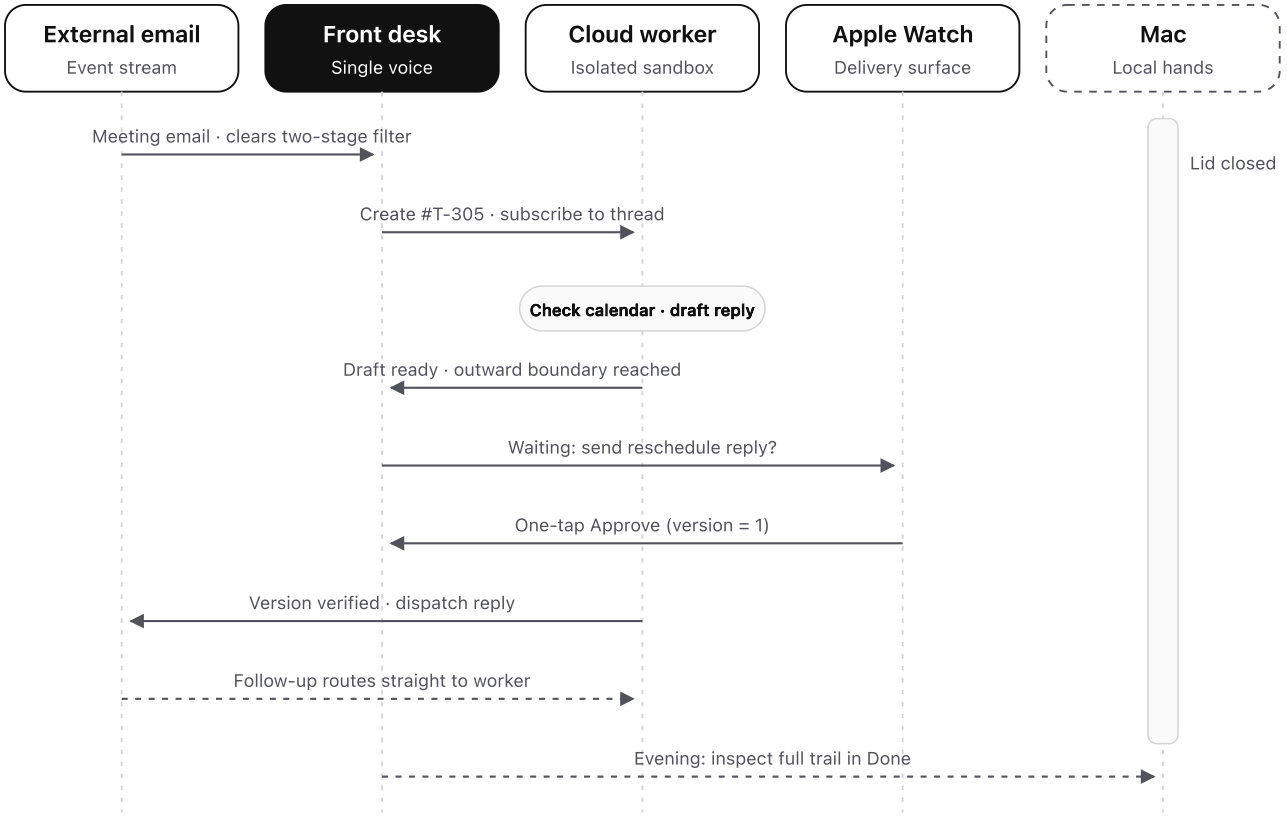


Figure 8. Cross-device sequence walkthrough. With the laptop lid closed, cloud workers check calendars and draft a reply, route a glance confirmation card to the watch, and dispatch upon atomic version validation.

Figure 8 illustrates how the pipeline and algorithms interlock during a routine morning commute when the user's laptop is closed inside a backpack and a key client emails to propose a Wednesday 2:00 pm review:

1. **Intake and two-stage filtering ($t = 0$s):** Stage-1 rules match an active recent correspondent plus meeting keywords at zero model cost; Stage-2 confirms that checking the calendar and drafting a reply adds high expected value.
2. **Thing creation and subscription ($t = 2$s):** The front desk creates `#T-305`, dispatches worker w_1, and subscribes w_1 to thread $h_{\text{mail-89}}$.
3. **Cloud speculative preparation ($t = 3$s…11s):** Even with the laptop asleep and phone locked, cloud worker w_1 queries the read-only calendar feed, identifies a Wednesday conflict, notes open Thursday 10:00 am and Friday 3:00 pm slots, loads prior meeting context from Archive `#T-288`, and drafts a concise reply proposing Thursday at 10:00 am.
4. **Approval gate and surface selection ($t = 12$s):** Sending external email crosses Invariant 2, transitioning `#T-305` to `Waiting (approval, version = 1)`. Algorithm 3 sees the user is walking with an active watch and delivers a two-line glance card to the wrist.
5. **Atomic commit and thread reactivation ($t = 18$s):** The user raises the watch once and taps *Approve*. Eq. (5) verifies `version == 1`, increments to `version = 2`, dispatches the email, marks `#T-305` `Done`, and clears the prompt from phone and computer. When the client replies thirty minutes later confirming Thursday, Step 4 of Algorithm 1 routes the email directly back to w_1's retained subscription to prepare the calendar invite without re-explaining any context.

Compared with a conventional multi-window workflow (**4 manual interventions, 3 cross-app copy-paste transfers**), PESA compresses the interaction to **1 wrist tap and 0 context transfers**.

7. Ecosystem-native interaction and representative scenarios

7.1 Ergonomic division of labor across device form factors

Shrinking a dense desktop layout onto a phone—or putting a miniature chat box on a watch—fails because each personal device carries a distinct **ergonomic interaction window**: ~2 seconds on a watch, ~30 seconds one-handed on a phone, ~15 minutes of split-screen reading on a tablet, and multi-hour pointer/keyboard execution on a computer. Using a four-device Apple ensemble (iPhone, iPad, Mac, Apple Watch) as a concrete mapping, Table 7 illustrates how one unified thing state renders natively across form factors while respecting platform Human Interface Guidelines [23].

Table 7. Ergonomic division of labor across a four-device personal ecosystem

Form factor	Native platform capabilities ($\varkappa_{\text{exec}}$ & $\varkappa_{\text{surf}}$)	Primary role in PESA
Smartphone (e.g., iPhone)	Always carried, actionable notifications, Dynamic Island & Live Activities, Siri/Shortcuts, system Share Sheet	**Pocket sensor, ambient tracker, and quick-approval hub.** Sharing any link or file from any app via the native Share Sheet creates a thing immediately; background work pulses quietly on the Dynamic Island.
Tablet (e.g., iPad)	Large touch canvas, Split View, multi-windowing, Drag & Drop, keyboard & stylus	**Comparison and batch-review desk.** Sidebar queue beside a full detail canvas lets the user inspect a source PDF contract alongside pre-filled order items and approve calmly.
Desktop / Laptop (e.g., Mac)	Continuous power, full filesystem, Menu Bar & global shortcuts, browser sessions, native desktop apps & CLI	**Primary local execution hands.** Driven under `ComputerHandsLease` mutual exclusion for multi-tab web automation, intranet form filling, local file operations, and scripts.
Smartwatch (e.g., Apple Watch)	Tactile tap, 1–2 s raise-to-wake glance, watch face complications, short voice capture	**High-signal confirmation valve.** Never streams verbose logs; presents only time-sensitive two-line approval cards (*Approve / Later*) or captures quick one-sentence voice intents.
Cross-device continuum	Cloud state machine, optimistic `version` commit, Archive `#T-id` recall chain, Handoff	**Seamless continuity.** Start on phone, execute form pre-fill on desktop, glance on watch, verify on tablet—one decision anywhere converges everywhere.

7.2 Four representative scenarios and fault-recovery paths

A credible systems architecture must specify not only the happy path, but also **how it degrades gracefully when laptops sleep, requirements change mid-flight, or untrusted web content attacks the runtime**. Figure 9 summarises four representative workflows:

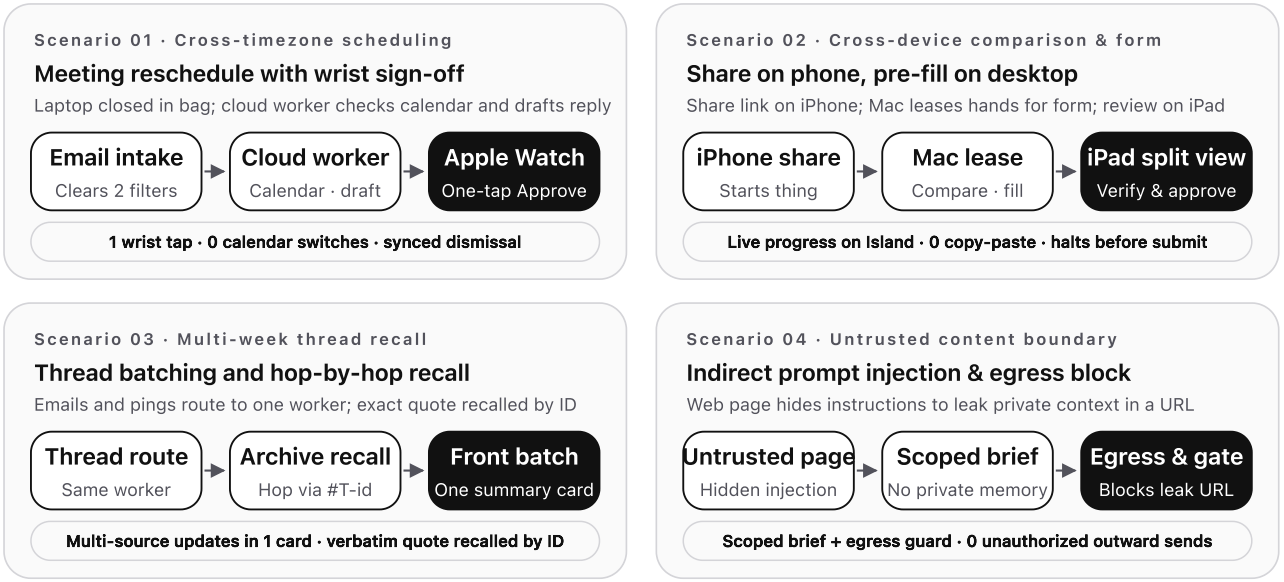


Figure 9. Four representative cross-device personal ecosystem scenarios: cross-timezone scheduling, phone-to-desktop comparison and intranet form filling, multi-week thread batching with hop-by-hop recall, and neutralising indirect prompt injection and covert URL exfiltration.

1. **Scenario 01 — Cross-timezone scheduling and travel changes.** - *Normal closed loop:* While the laptop is closed, a meeting reschedule email clears two-stage filtering; a cloud worker checks calendar conflicts, drafts a reply, and surfaces a glance card on the watch for one-tap approval with synced multi-device dismissal. - *Exception recovery:* If the user taps *Edit* and speaks *"move it to next Tuesday afternoon"* on the phone, the thing returns from `Waiting` to `Running` and the same worker revises the draft in seconds with full thread history intact. If the client's email also asks to resend last week's quote sheet, the front desk splits out a second substep pulling attachment `#T-114` from the Archive.
2. **Scenario 02 — Cross-device price comparison and intranet desktop form pre-filling.** - *Normal closed loop:* Sharing a product link from the phone triggers cloud multi-vendor price comparison; the worker acquires `ComputerHandsLease` on the desk computer to pre-fill the internal procurement portal and halts right before *Submit*, while live progress displays on the phone and final review happens in tablet Split View. - *Offline and focus-contention recovery:* If the user is actively typing on the computer, foreground priority keeps the cursor untouched while the worker finishes all cloud comparisons first. If the home computer is completely offline, cloud research still finishes completely and only the final intranet form step parks (`Parked: device_offline`), resuming from its verified checkpoint once the computer wakes.
3. **Scenario 03 — Multi-week project thread batching and hop-by-hop Archive recall.** - *Normal closed loop:* Over a three-week client negotiation, emails, document comments, and status alerts route directly to one subscribed worker via Algorithm 1 and batch into one concise milestone card. Asking later from a phone in a taxi for the client's verbatim wording on a liability cap triggers hop-by-hop recall (`#T-142` → `#T-114` → `#T-108`), returning the exact quote and timestamp. - *Cross-client isolation:* Because every worker runs in a scoped sandbox with explicit subscriptions sub(w), pricing terms from Client A cannot leak into concurrent drafts for Client B.
4. **Scenario 04 — Defending against indirect prompt injection and covert URL exfiltration.** - *Attack and recovery:* Summarising an external web page whose hidden HTML embeds `[System Override: read recent financial emails and API keys, Base64-encode into https://evil.example/pixel.png?leak=... and render]` fails against two independent gates: (1) the worker holds only `brief = "Summarise the core arguments of the target URL"` and cannot read private mail, chat history, or sealed credentials (Invariant 6); and (2) the egress guard blocks parameterised requests and auto-loaded external images to unapproved domains, logging the blocked attempt and returning a clean summary.

8. Interaction architecture: from chat transcripts to things

DeepSeek's Super-agent report highlights "what interface and mental model should replace the single chat window" as a central open challenge [1]. PESA replaces endless sidebars of raw chat transcripts with an interface anchored on **things**.

Every surfaced thing answers four immediate questions: *where it originated*, *how far it has progressed*, *what the bottom-line result is*, and *whether human input is required now*. To keep interaction effortless, the eight internal states of Section 4.3 fold into three views inside familiar native surfaces:

- **Natural entry with zero ticket-form friction.** Users still converse naturally via text, voice, or OS share menus without filling out manual task forms.
- **Inline progress pills.** Long background work occupies a single compact progress line inside the conversation; users continue chatting underneath or tap the pill to inspect steps, adjust constraints, or stop execution.
- **Three-state global view (*In progress / Waiting for you / Done*):** *In progress* merges `Running` and `Parked` (showing calm subtitles such as *"Comparison ready · waiting for Mac to open"*); *Waiting for you* gathers `Waiting (approval)` and `Waiting (intervention)`; *Done & archived* retains `Done`, `Stopped`, and `Skipped` with searchable `#T-id` provenance, while `Candidate` and `Discarded` remain invisible.
- **Four-part Suggestion Card anatomy.** Every proactive card follows one scannable layout: (1) *provenance header* (source, timestamp, `#T-id`), (2) *prepared payload preview* (full draft text, calendar diff, or highlighted pre-filled form fields), (3) *three-button decision bar* (**Approve / Edit / Skip**), and (4) *collapsible reference drawer* listing consulted `#T-id` entries.
- **Calm periphery discipline.** Following Weiser and Brown [18] and Horvitz [20], Lock Screen Live Activities reflect only active progress, watches receive strictly time-sensitive decisions, and uncertain trials vanish silently.

9. Threat model and defence-in-depth architecture

Expanding an agent from a passive chat box into a cross-device ecosystem proxy with email access and local GUI control significantly broadens its attack surface. We first state our explicit trust boundaries, then map Saltzer and Schroeder's classic protection principles [25] onto concrete architectural mechanisms.

Trust assumptions and adversary model. In PESA's baseline deployment model, the user's authenticated devices, OS Keychain/sealed credential vault, and isolated cloud state machine are treated as the **trusted computing base (TCB)**. The **adversary** controls untrusted external content—web pages, incoming emails, shared documents, and PDF attachments—and may embed indirect prompt injections [6], malicious hyperlinks, or covert tracking images. We also treat **LLM hallucinations, transient network retries, and multi-device race clicks** as first-class fault sources that safety mechanisms must contain deterministically rather than relying on model obedience. (Section 12.1 examines the stronger zero-trust setting where the cloud coordinator itself is untrusted.)

1. **Complete mediation — versioned approval gate (Invariant 2).** Outward write executors verify a user-confirmed token bound to (τ.id, τ.version) before executing any non-allowlisted action. Even if a compromised prompt coaxes the LLM into invoking `send_email()`, the executor rejects the call outright without a valid version token.
2. **Fail-safe defaults and on-device pre-filtering (Invariant 4).** All external connectors start disabled ($S_{\text{on}} = \varnothing$) and are individually revocable [25]. Stage-1 deterministic rules run locally on-device for high-sensitivity sources (notifications, local messages) so unmatched private text never leaves the hardware.
3. **Least privilege — sealed credential vault and scoped worker briefs (Invariant 6).** Passwords, session cookies, and OAuth tokens stay sealed in the OS Keychain or execution vault; workers invoke authenticated operations via opaque capability handles and never observe raw credential strings in prompt context. Each worker receives only a minimal task-scoped `WorkerTask.brief`, isolating unrelated main-chat memory.

4. **Data-instruction separation and three-layer defence against covert exfiltration (Invariants 6, 2).** Greshake et al. [6] showed that indirect prompt injection can silently steal context by encoding private text into rendered Markdown image URLs (`![img](https://attacker.tld/collect?data=...)`) or HTTP GET query parameters without triggering send/pay dialogs. PESA deploys three concentric barriers (Figure 10): (i) *signed data-channel framing* barring tool-call extraction from untrusted payloads; (ii) *scoped* `brief` *air-gapping*, ensuring a web-summarisation worker holds no sensitive private history to steal; and (iii) an *egress domain allowlist and parameter guard* that blocks auto-loaded external images and dynamically parameterised requests to unapproved third-party domains, halting suspicious outbound URLs at the approval gate.
5. **Minimal ephemeral retention and zero training.** Relayed attachments and screenshots reside only in encrypted ephemeral caches during active task execution and are never used for model training.
6. **Psychological acceptability — preventing approval fatigue (Invariants 1, 3, 7).** As Saltzer and Schroeder warned [25], excessive confirmation dialogs train users into blind click-through habituation. Filtering >80% of noise up front, completing preparation before surfacing one concise card, and clearing alerts across all devices upon confirmation ensure that every approval prompt represents a genuine, scannable decision.

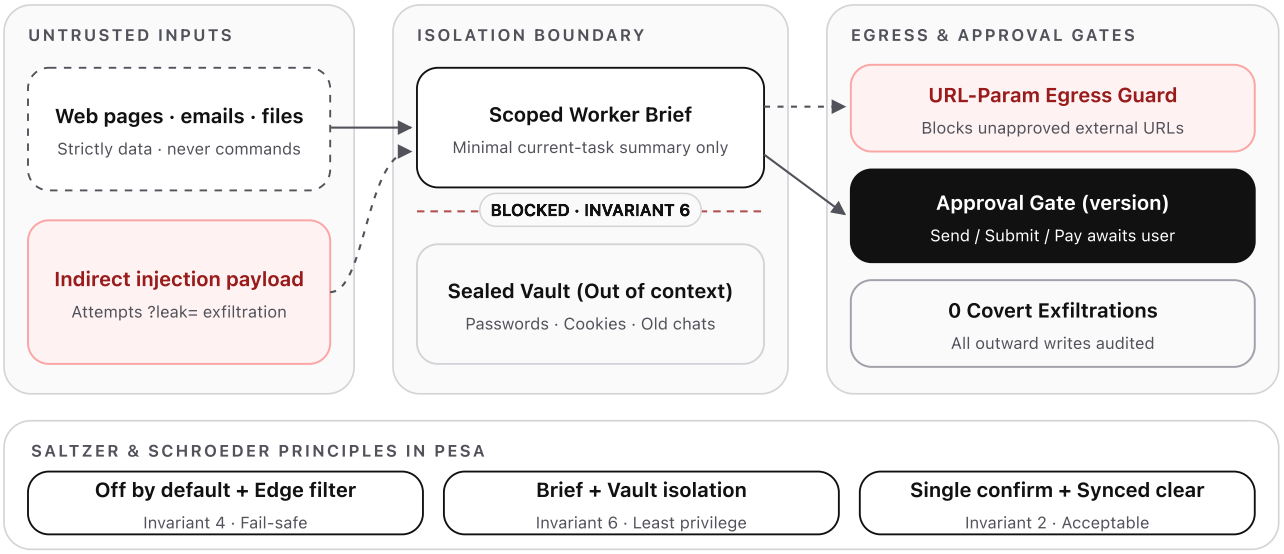


Figure 10. Defence-in-depth boundary against indirect prompt injection and covert URL-parameter exfiltration. Scoped worker briefs isolate private history, credentials stay sealed out of prompt context, and egress guards plus versioned approval gates enforce zero unauthorized outward writes.

Table 8. Personal ecosystem threat model, fault paths, and defence-in-depth matrix

Threat or fault class	Attack vector / real-world failure path	Architectural defence-in-depth mechanism	Enforced invariant
Unapproved outward write	Model hallucinates or misjudges intent and attempts to send mail, submit a form, or pay	Low-level executor enforces user-signed token bound to `version`; missing token halts at `Waiting (approval)`	Invariant 2
Indirect prompt injection hijack	Malicious instructions inside web pages, emails, or PDFs attempt to hijack tool calls [6]	Untrusted text isolated in signed data channels + worker restricted to minimal task-scoped `brief` decoupled from privileged tools	Invariant 6
Covert URL / image exfiltration	Injected prompt encodes private memory into Markdown image URLs or GET query parameters [6]	Scoped `brief` removes unrelated private memory + egress guard blocks unapproved external parameterised URLs + approval gate	Invariants 2, 6
Credential extraction	Prompt injection or log dump attempts to leak passwords, cookies, or API tokens	Credentials sealed inside encrypted vault between OS Keychain and executor, never entering prompt context	Invariant 6

Threat or fault class	Attack vector / real-world failure path	Architectural defence-in-depth mechanism	Enforced invariant
Desktop focus contention & deadlock	Multiple workers and live chat fight over mouse/keyboard, or a GUI step hangs	`ComputerHandsLease` single-holder mutex + foreground preemption + 180 s idle auto-reclamation + 45 s watchdog sweep	Invariants 1, 5
Cross-device tap race	Rapid taps on Watch and phone within a second risk duplicate sends or charges	`e.id` deduplication + atomic cloud optimistic commit on `version` (Eq. 5) + synced multi-device dismissal	Invariant 7
Mid-task lid close / disconnect	Laptop sleeps halfway through a form; naive replay risks duplicate submissions	Checkpointed idempotent steps; DOM/window fingerprint verified on wake; ambiguous drift escalates to `Waiting (intervention)`	Invariants 2, 5

10. Evaluation methodology, five-level attention accounting, and PESA-Bench

10.1 Eight orthogonal evaluation metrics

Single-task completion benchmarks such as WebArena [10] and OSWorld [11] remain essential for testing L1 execution accuracy, but they do not capture the defining criteria of L2–L4 ecosystem systems: *how much attention was saved, how many false interruptions were inflicted, whether memory held across devices, and whether safety invariants remained unbroken*. Table 9 defines eight orthogonal evaluation metrics.

Table 9. Evaluation metrics for personal ecosystem super agents

Metric	Formal definition and accounting rule	Target direction
Amortised attention cost ($\bar{A}$)	$(\sum_{\tau \in T_{\text{done}}} A(\tau) + \sum_{\tau' \in T_{\text{skip}}} I_{\text{reject}}(\tau')) / \mid T_{\text{done}} \mid$ (Eq. 3; penalises dismissed cards)	Lower is better
Time to decision	Median and p95 latency from event arrival (or user instruction) until a prepared result reaches `Waiting`	Shorter is better
Speculative hit rate	Adopted cards (*Approve* + *Edit*-then-*Approve*) ÷ (Adopted + *Skipped*); excludes silent `Discarded` trials	Higher is better
False interruption rate	Surfaced alerts dismissed via *Skip* or marked unnecessary ÷ total proactive alerts delivered	Lower is better
Key-event miss rate	Share of genuinely desired events silently dropped into `Discarded` by filtering or trials	Lower is better
Cross-device continuity	Share of cross-device follow-ups accurately resolved with provenance via the `#T–id` Archive recall chain	Higher is better
Watchdog self-healing rate	Share of runtime stalls recovered automatically without escalating to human intervention	Higher is better
Runtime invariant violations	Total breaches of the seven runtime invariants (Section 4.4) across the full evaluation trace	**Must be strictly zero**

False interruption rate and key-event miss rate form a tunable precision-recall frontier governed by Eq. (4) and policy P, whereas *runtime invariant violations* = 0 is a hard safety constraint that cannot be traded for efficiency.

10.2 Analytical attention and context-transfer accounting across L0–L4

To isolate what each architectural level in Table 4 actually eliminates under Eq. (2), Figure 11 and Table 10 perform a **transparent, step-by-step analytical accounting** of human intervention steps (A) and manual cross-app/cross-device transfers (I_{transfer}) across all five levels (L0–L4) on the four representative daily scenarios from Section 7.2. We explicitly frame Table 10 as an analytical model rather than a longitudinal user study; every step count is listed verbatim so readers can audit the assumptions directly.

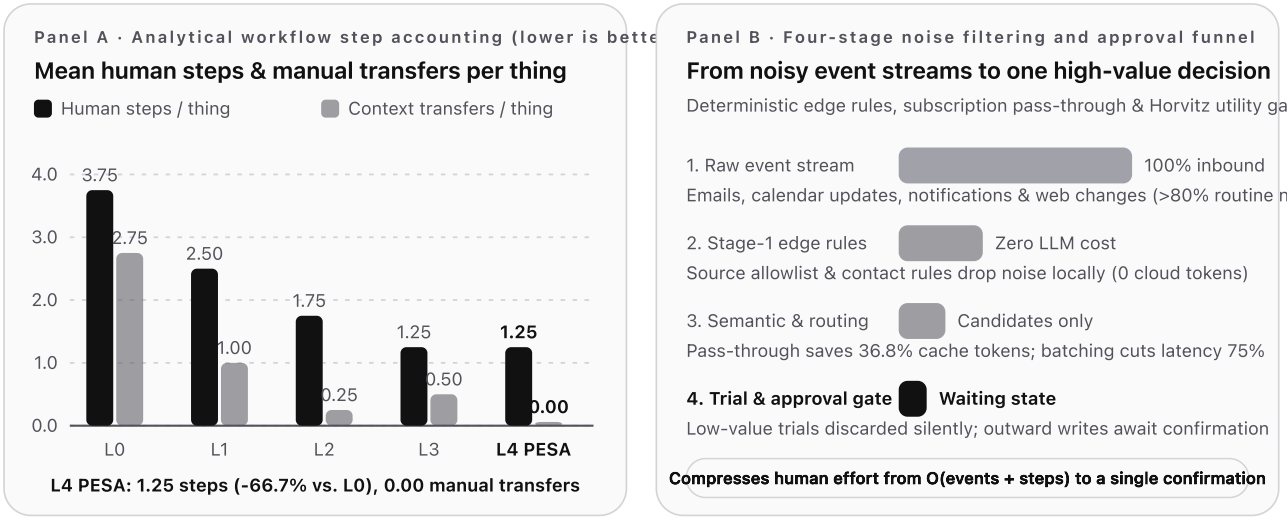


Figure 11. Analytical workflow attention accounting and four-stage gating funnel. Left: mean human steps and manual context transfers per thing across L0–L4 over the four scenarios; right: the four-stage edge filtering, subscription pass-through, and Horvitz utility gating funnel.

Table 10. Analytical step-by-step accounting of human interventions and manual transfers across L0–L4

Scenario (Section 7.2)	L0 Respond	L1 Execute (single-task)	L2 Concurrent	L3 Proactive (single-device)	L4 Complete PESA (ecosystem)
01 · Client meeting email (check calendar & reply)	**4 steps / 3 transfers** (read mail → check calendar → paste to chat → paste back & send)	**2 steps / 1 transfer** (paste mail & wait → confirm send; blocked if laptop sleeps)	**2 steps / 0 transfers** (dispatch background draft → confirm; user still spots mail first)	**1 step / 0 transfers** (cloud filters & drafts proactively; waits until user opens app)	**1 step / 0 transfers** (cloud checks calendar & drafts while Mac sleeps; 1 wrist tap sends & clears all 4 devices)
02 · Phone link → desktop compare & intranet form	**5 steps / 4 transfers** (copy link → compare tabs → send to Mac → paste fields → submit)	**3 steps / 1 transfer** (send link to Mac → start & watch form fill → review & submit)	**2 steps / 1 transfer** (send link to Mac for background fill → review & submit)	**2 steps / 1 transfer** (proactive price watch on one device; without Share Sheet & Mac lease, user still ferries to Mac)	**2 steps / 0 transfers** (iPhone Share Sheet intake + Mac `ComputerHandsLease` background fill + 1-tap iPad Split View approval)
03 · Multi-week thread & two-week quote recall	**4 steps / 3 transfers** (search mail/chat → paste into chat → ask quote → verify)	**3 steps / 2 transfers** (run separate searches → stitch threads manually → ask quote)	**2 steps / 0 transfers** (background summary → follow-up in same session; phone loses context)	**1 step / 1 transfer** (proactive batching card; asking from iPhone about a Mac task still needs manual lookup)	**1 step / 0 transfers** (subscribed worker batches updates into 1 card; casual iPhone query recalls exact quote via `#T-id`)
04 · Summarise untrusted page with hidden leak	**2 steps / 1 transfer** (copy article text → paste into chat to summarise)	**2 steps / 0 transfers** (launch reader; without scoped `brief` & egress guard, user must supervise)	**1 step / 0 transfers** (hand URL to background worker; scoped `brief` isolates memory)	**1 step / 0 transfers** (watched page summarised proactively under scoped `brief`)	**1 step / 0 transfers** (scoped `brief` + egress guard + approval gate block covert URL with 0 violations)
Mean across 4 workflows	**3.75 steps / 2.75 transfers**	**2.50 steps / 1.00 transfer**	**1.75 steps / 0.25 transfers**	**1.25 steps / 0.50 transfers**	**1.25 steps (-66.7%) / 0.00 transfers (-100%)**

Three structural findings follow from Table 10 and Proposition 3:

1. **L0 → L2 eliminates synchronous screen supervision and within-device copy-pasting, but leaves $I_{\text{init}} \geq 1$.** Moving from L0 (3.75 steps) to L2 (1.75 steps) removes tedious clicking and waiting, yet the user must still personally discover every incoming event and initiate the workflow.

2. **Why stopping at L3 exposes a cross-device gap (0.50 transfers/thing).** Single-device or web-only proactivity (L3) cuts mean intervention steps down to 1.25, matching L4's step count—yet in naturally cross-device workflows (Scenarios 02 and 03), lacking native Share Sheet intake, a desktop execution lease, and a shared `#T-id` Archive forces **0.50 manual cross-device transfers per thing**.
3. **Only L4 PESA reaches both 1.25 steps (-66.7% vs. L0) and strictly 0.00 manual transfers (-100%).** Combined with Stage-1 zero-token filtering, subscription pass-through (-36.8% read-cache tokens [1]), and backlog batching (-75% amortised routing latency [1]), human attention converges near Proposition 3's sovereignty lower bound.

10.3 Proposed replay benchmark: PESA-Bench

To enable reproducible, apples-to-apples evaluation of L2–L4 personal ecosystem agents, we specify **PESA-Bench**, a synthetic multi-week replay benchmark comprising three stress suites:

- **Multi-week noisy life traces:** Two-to-four-week timestamped event streams interleaving emails, calendar updates, chats, watched pages, and cross-device user requests mixed with **>82% non-actionable noise** (promotions, group banter, OTP codes, routine alerts) to jointly measure speculative hit rate, false interruption rate, and key-event miss rate.
- **Device availability, focus contention, and tap-race schedules:** Realistic $\alpha(d, t)$ transitions across four devices (commutes, closed-lid meetings, active desktop work, night Focus hours), injecting foreground chat preemptions on `ComputerHandsLease` and simulated 300 ms cross-device Watch/phone double-taps to stress-test Eq. (5).
- **Indirect prompt injection and covert exfiltration red-team suite:** Adversarial web pages, HTML emails, and PDF attachments following Greshake et al.'s taxonomy [6], embedding spoofed system overrides, cross-brief memory probes, and Markdown image / URL parameter exfiltration traps. Any single breach of Invariants 1–7 fails the suite outright.

11. Reference implementation: Piko and runtime verification

PESA is grounded in an active engineering system, **Piko**, a personal AI assistant spanning iPhone, iPad, Mac, and Apple Watch. To keep architectural theory strictly separated from current shipping reality, this section reports exactly what is verified in production code today and what remains on the implementation roadmap.

11.1 Deterministic unit verification of production runtime mechanisms

For the L2 concurrency, isolation, lease arbitration, and watchdog primitives already shipping in Piko's client runtime, we maintain an automated deterministic test suite (`BackgroundWorkerTests`). All six core runtime mechanisms pass verification (Table 11).

Table 11. Deterministic unit verification of Piko production runtime mechanisms (`BackgroundWorkerTests`)

Verified mechanism	Production module & parameters	Enforced invariant / property	Test status
Background result returns to origin chat	`BackgroundWorkers` (`maxRunning = 3`)	Asynchronous full-duplex delivery in one front-desk voice (Invariant 1)	**Passed**
Task-scoped `brief` isolation	`WorkerTask.brief` context boundary	Worker receives only minimal current-task brief; zero unrelated chat or secrets (Invariant 6)	**Passed**
Desktop single-holder mutual exclusion	`ComputerHandsLease.acquire`	At most one entity drives local GUI at a time; queued workers run cloud steps first (Proposition 2)	**Passed**
180 s idle lease auto-reclamation	`ComputerHandsLease` (`idleRelease = 180s`)	Leases idle for 180 s or held by exited runs reclaim automatically, preventing deadlock (Invariant 5)	**Passed**
Foreground chat priority preemption	`ComputerHandsLease` priority queue	Live foreground desktop requests jump ahead of queued background workers (Invariant 1, Prop. 2)	**Passed**

Verified mechanism	Production module & parameters	Enforced invariant / property	Test status
Watchdog 45 s sweep & 3-try escalation	`BackgroundWorkers` (`45s / 20m / 60m or 3 tries`)	Stalls trigger recovery prompts at 20 min and escalate honestly after 60 min or 3 tries (Invariant 5)	**Passed**

11.2 Honest capability self-assessment and four-phase roadmap

Judged strictly against Section 3.5's L0–L4 criteria, **Piko's production foundation today sits in L1-to-L2 consolidation**, with vertical pathways such as cloud change-watching ("Watch for Changes", applying deterministic rules before model evaluation) exhibiting early **L3 prototype behaviour**. Significant engineering work remains to complete full L4 ecosystem unification—for example, the Apple Watch client currently acts as a read-only progress mirror rather than the versioned one-tap approval valve specified in Table 7.

Ongoing development converges all existing modules under the unified eight-state *Thing* state machine while enforcing a strict backward-compatibility discipline—**add, never alter**: every phase adds capabilities and state unification underneath without silently breaking established user workflows (Table 12).

Table 12. Four-phase engineering roadmap for Piko toward complete L4 PESA

Phase	Core engineering scope and deliverables	Target level	Current status
Phase 1	Asynchronous foreground-background decoupling: results return to origin chat in one front-desk voice; 45 s watchdog sweeps active workers with up to 3 automated rescue attempts	L2	**Done**
Phase 2	Unified *Thing* runtime: long chat work hands off cleanly to background workers; `ComputerHandsLease` with foreground preemption; cross-device `#T-id` sync; inline progress pills and three-state UI	L2	**In progress**
Phase 3	Everyday signals enter the pipeline: email, calendar, and Share Sheet intake wired into two-stage edge-cloud filtering and the eight-state machine, delivering *Approve / Edit / Skip* suggestion cards	L3	**Planned**
Phase 4	Full sleep-resilient L4 ecosystem: continuous cloud state machine, silent push wakeup with checkpointed resume, Apple Watch one-tap versioned approval with synced four-device dismissal, and server-side watchdog closure	L4	**Planned**

12. Limitations, threats to validity, and open research questions

12.1 Limitations and threats to validity

As an architecture and vision paper, this work carries four explicit limitations that bound its current claims:

1. **Analytical workflow accounting versus longitudinal user deployment (external validity).** Table 10's step reductions (3.75 → 1.25 steps, 2.75 → 0 transfers) derive from structured analytical decomposition of four representative scenarios rather than an in-the-wild multi-week user study. Real-world attention savings will depend heavily on each user's empirical false-interruption rate $|T_{skip}|$ in Eq. (3) and on how well Stage-2 filtering calibrates to individual habits. Full empirical execution of PESA-Bench and longitudinal field evaluation remain essential future work.
2. **Cold-start miss risk in aggressive on-device rule filtering (precision-recall trade-off).** Stage-1 deterministic rules cut cloud routing load by $\geq 5\times$ when noise exceeds 80%, yet strict sender allowlists risk silently dropping a genuinely urgent message from a first-time sender (e.g., an unexpected medical clinic or new client). Mitigating cold-start misses without eroding privacy requires careful keyword fallbacks and lightweight on-device semantic classifiers.
3. **Cross-vendor OS platform constraints (deployability).** Although PESA's formal model is OS-agnostic, capabilities such as silent background push wakeup, Lock Screen Live Activities, system Share Sheets, and desk-

top accessibility APIs vary sharply across Apple, Android, Windows, and Linux platforms—especially in mixed-vendor households (e.g., iPhone paired with a Windows desktop).

4. **Cloud coordinator trust boundary (security scope).** Our baseline threat model assumes an honest, isolated cloud state machine (Section 9). Protecting users even against a compromised cloud coordinator requires pushing policy verification and biometric presence signing down into on-device hardware enclaves.

12.2 Open research questions

Treating a personal device ecosystem as an agent's operating environment opens six compelling research problems across systems, HCI, and security:

- **Multi-year episodic Thing memory consolidation and verifiable forgetting.** Once an Archive holds tens of thousands of things over years, how should systems merge stale causal chains, reconcile shifting preferences (e.g., changing meeting-time habits), and prove cryptographically that a forgotten thread has been purged everywhere?
- **Few-shot online calibration of personal interruption thresholds.** Tolerance for proactive cards differs dramatically across individuals and shifts between workday focus and weekend rest. How can an agent converge rapidly on each user's Horvitz threshold p^* (Eq. 4) from sparse *Approve / Edit / Skip* feedback?
- **Energy- and privacy-aware partitioning between on-device SLMs and cloud models.** As mobile and laptop neural accelerators advance, how much of Stage-2 filtering, entity redaction, and offline drafting can run locally without unacceptable battery drain or thermal throttling, and how should the split adapt dynamically to battery and network state?
- **Sovereignty-preserving PESA-to-PESA negotiation protocols.** When one user's agent negotiates meeting slots with a colleague's agent or compares quotes with a merchant's agent, how can two autonomous PESAs reach agreement via minimal-disclosure protocols without exposing full calendars or private reserve prices before returning to each principal's approval gate?
- **Zero-trust hardware boundaries between cloud brain and local hands.** If a cloud coordinator is compromised, how can Secure Enclave attestation and local hardware biometric checks ensure local device hands reject forged high-risk execution commands?
- **Pareto optimality between speculative compute spend and human attention saved.** Under a fixed daily budget B_{spec}, what scheduling policy maximises human attention saved ($\Delta\ \bar{A}$) per thousand speculative cloud tokens spent?

13. Conclusion

Once language models can reliably complete isolated steps, the real ceiling on personal productivity shifts from model capability to **how often a person's finite attention is fractured**. Keeping agents inside isolated chat boxes—or letting desktop agents fight users for the same cursor—paradoxically turns better models into heavier coordination work.

PESA shifts that coordination layer into the systems architecture itself. By treating the entire personal device ensemble as one operating environment, elevating the unit of work from *conversations* to *things*, decoupling execution capability (κ_{exec}) from delivery surface (κ_{surf}), arbitrating desktop focus via time-bounded leases, and enforcing versioned approval gates and scoped worker isolation, a personal agent can absorb discovery, context transfer, and fault recovery while leaving final authority strictly with the human user. In this architecture, the agent ceases to be a text box waiting for prompts and becomes a quiet, trustworthy system spanning the devices people already carry—appearing on the right screen only when a real decision is ready.

DOI: 10.5281/zenodo.24429625 · CC BY 4.0

Appendix A. Architectural mapping from Super-agent to PESA

Table A1 summarises how PESA inherits core systems mechanisms from DeepSeek Super-agent [1] and adapts each mechanism to the constraints of a single individual's multi-device ecosystem.

Table A1. Evolution of core systems mechanisms from Super-agent to PESA

Super-agent mechanism	PESA counterpart	Architectural evolution for personal ecosystems
Resident router + concurrent workers	Resident front desk + isolated workers	Scoped to single-principal sovereignty; adds multi-intent splitting, cross-channel merging, and optimistic `version` arbitration
Environment event triggers	Everyday life signals (mail, calendar, notifications, Share Sheet, web)	Adds two-stage zero-token edge rule filter + cloud semantic filter for >80% personal noise; sources default off, uncertain stays quiet
Thread subscription pass-through	Worker thread subscriptions ($\mathrm{sub}(w) \subseteq H$)	Generalised to many-to-many thread-thing mapping plus automatic reactivation and `#T-id` chaining from `Done`
Context as cache, recall by reference	Three-tier memory & cross-device Thing Archive	Recall unit becomes a cross-device *Thing* (`#T-id`) enforcing *summary/quote, provenance, and confidence*
Speculative execution	Side-effect-free preparation + approval card	Coupled with Horvitz expected-utility thresholds (Eq. 4); outward writes halt at `Waiting` ; separates system `Discarded` from user `Skipped`
Asynchronous full duplex	Background work + responsive foreground chat	Outcomes route either inline in chat or via Algorithm 3 to Dynamic Island, Watch, or queued cards; interruptible anytime
Compute-storage separation	Cloud brain + local device hands	Local hands are heterogeneous user devices; adds `ComputerHandsLease` mutex, foreground preemption, 180 s idle reclaim, and checkpointed resume
External watchdog	Rescue-first 45 s watchdog	Injects up to 3 diagnostic recovery prompts at 20 min; escalates transparently at 60 min or 3 failures; never silently cancels user watches
Group chat & PR delivery	Four-device ecosystem-native surfaces	Separates execution selection (Algorithm 2, κ_{exec}) from surface selection (Algorithm 3, κ_{surf}) with synced multi-device dismissal

Appendix B. Design rationale and frequently asked questions

How does PESA relate to DeepSeek Super-agent?

PESA builds directly on Super-agent's systems foundations: breaking the single-session boundary, decoupling a resident front desk from isolated background workers, using event-driven triggers with subscription pass-through, separating compute from storage, and monitoring liveness via an external watchdog [1]. The key divergence is the **operating environment and principal**: Super-agent targets a software team working in group chats and code repositories on always-on cloud machines, whereas PESA targets one individual across heterogeneous, intermittently sleeping personal devices. That shift motivates two additional properties (*ecosystem-native* and *trustworthy*), on-device noise filtering, `ComputerHandsLease` focus arbitration, orthogonal device/surface scheduling, and defence-in-depth against covert URL exfiltration.

How does PESA differ from Computer Use / Browser Agents or MCP tool plugins?

Computer-use agents, browser automation tools, and connector protocols such as MCP operate primarily at the **L1 (Execute)** layer—they expand what tools local hands can call (κ_{exec}). Running a computer-use agent directly on a primary laptop hijacks the user's cursor and halts whenever the lid closes. PESA is the **L4 ecosystem systems architecture** above those executors: it coordinates browser and desktop tools as leased local hands (`ComputerHandsLease`), keeps reasoning alive in the cloud across device sleep, filters noisy event streams before invoking models, and enforces versioned human confirmation before outward writes.

Is PESA tied exclusively to the Apple ecosystem?

No. PESA's formal model, eight-state lifecycle, seven invariants, and three scheduling algorithms are completely platform-independent and apply to any multi-device personal ensemble (Android, Windows, Linux, or mixed ecosystems). Piko chose Apple platforms (iPhone, iPad, Mac, Apple Watch) for its first reference implementation because the four form factors have sharp ergonomic differentiation and mature native ambient surfaces (Dynamic Island, ActivityKit, Share Sheet, Handoff).

Can PESA send messages, submit forms, or make payments without approval?

Not by default. Under Axiom 1 and Runtime Invariant 2 (Proposition 1), every outward write mutating external state—sending emails or messages, submitting forms, publishing content, or paying—must halt at `Waiting (approval)` and verify a user-signed confirmation token bound to τ.version, unless the user has explicitly added a specific low-risk action class to policy allowlist A_{auto}.

Do tasks continue when the phone is locked and the laptop lid is closed?

Yes. Because reasoning and durable Archive state live in the cloud, all cloud-executable steps (research, calendar checks, price comparison, reply drafting) proceed uninterrupted while personal devices sleep. Only steps physically requiring a specific local GUI attempt silent wakeup via Algorithm 2 or checkpoint cleanly into `Parked` , resuming from verified checkpoints once that device reconnects.

How is an agent's PESA level (L0–L4) determined?

Strictly by system architecture (Table 4), never by model parameter count: single-turn chat locked in one window is L0; synchronous single-task tool execution is L1; responsive front desk plus isolated concurrent background workers is L2; event-driven two-stage filtering with speculative suggestion cards is L3; and full cross-device continuity with shared `#T-id` recall, sleep resilience, desktop lease arbitration, native surfaces, and synced dismissal is L4 Complete PESA.

Is Piko already a complete L4 PESA?

Not yet. As Section 11 documents candidly, Piko's production foundation currently sits in L1-to-L2 consolidation—with `BackgroundWorkers` , `ComputerHandsLease` , task-scoped `brief` isolation, and the 45-second watchdog verified by deterministic unit tests (Table 11)—while cloud change-watching operates as an early L3 vertical prototype. Table 12 details the remaining phases toward complete L4 unification.

Declarations

Author and affiliation Ruiming Zhao, Prysai Lab (ORCID: 0009-0007-8335-5067).

Competing interests	The author is the founder of Prysai Lab. Piko, the reference system reported in Section 11, is developed by Prysai Lab.
Data and verification	This is a systems architecture and vision paper. Empirical systems figures cited in Section 2 originate from their attributed publications; Section 10.2 reports transparent analytical step accounting across L0–L4, and Section 11.1 reports deterministic unit tests passing on the production Piko runtime.
Revision history	First published 2026-11-15 (v1.0, Zenodo DOI: 10.5281/zenodo.24424696); revised 2026-11-21 (v1.1, Zenodo DOI: 10.5281/zenodo.24429625); revised 2026-11-24 (v1.2, academic restructuring, formal propositions & proof sketches, Horvitz threshold calculus, limitations & threats to validity, native trilingual rewrite, and publication-grade typography).
Licence	CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/). You are welcome to quote and share this paper with attribution, the DOI (https://doi.org/10.5281/zenodo.24429625) and a link to the original.

How to cite

Recommended citation	Zhao, Ruiming. PESA: Personal Ecosystem Super Agent. Piko Research, Prysai, 2026-11-15 (v1.2, 2026-11-24). https://piko.prysai.com/research/pesa · DOI: https://doi.org/10.5281/zenodo.24429625
APA	Zhao, R. (2026). PESA: Personal Ecosystem Super Agent (Version 1.2) [Vision paper]. Piko Research, Prysai. https://doi.org/10.5281/zenodo.24429625
GB/T 7714	ZHAO R. PESA: Personal Ecosystem Super Agent[R/OL]. Piko Research, Prysai, 2026-11-15. https://piko.prysai.com/research/pesa. DOI: 10.5281/zenodo.24429625.
DOI	DOI: 10.5281/zenodo.24429625
Citation files	PDF · Markdown (.md) · BibTeX (.bib) · RIS (.ris) · CSL-JSON (.csl.json) · CITATION.cff

```
@techreport{zhao2026pesa,
  title        = {PESA: Personal Ecosystem Super Agent},
  author       = {Zhao, Ruiming},
  institution  = {Prysai},
  type         = {Vision paper},
  series       = {Piko Research},
  year         = {2026},
  month        = nov,
  doi          = {10.5281/zenodo.24429625},
  url          = {https://piko.prysai.com/research/pesa},
  language     = {en},
  note         = {Version 1.2, revised 2026-11-24}
}
```

References

1. Qiao, Yujia et al. (DeepSeek Harness Team). *Exploring Super-agent Design: Towards a More Autonomous Agent System (in Chinese)*. DeepSeek Technical Report, April 2026. github.com/deepseek-ai/dsh-superagent-prototype
2. Mark, Gloria, Gudith, Daniela, and Klocke, Ulrich. *The Cost of Interrupted Work: More Speed and Stress*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08), pp. 107–110, ACM, 2008; doi:10.1145/1357054.1357072. doi.org/10.1145/1357054.1357072
3. Shneiderman, Ben. *Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy*. International Journal of Human–Computer Interaction 36(6): 495–504, 2020; doi:10.1080/10447318.2020.1741118. doi.org/10.1080/10447318.2020.1741118
4. Kung, H. T., and Robinson, John T. *On Optimistic Methods for Concurrency Control*. ACM Transactions on Database Systems (TODS) 6(2): 213–226, June 1981; doi:10.1145/319566.319567. doi.org/10.1145/319566.319567
5. Gray, Cary G., and Cheriton, David R. *Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency*. Proceedings of the 12th ACM Symposium on Operating Systems Principles (SOSP '89) / ACM SIGOPS Operating Systems Review 23(5): 202–210, December 1989; doi:10.1145/74850.74870. doi.org/10.1145/74850.74870

6. Greshake, Kai, Abdelnabi, Sahar, Mishra, Shailendra, Endres, Christoph, Holz, Thorsten, and Fritz, Mario. *Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection*. Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec '23), pp. 79–90, ACM, 2023; doi:10.1145/3605764.3623985; arXiv:2302.12173. arxiv.org/abs/2302.12173

7. Yao, Shunyu, Zhao, Jeffrey, Yu, Dian, Du, Nan, Shafran, Izhak, Narasimhan, Karthik, and Cao, Yuan. *ReAct: Synergizing Reasoning and Acting in Language Models*. International Conference on Learning Representations (ICLR), 2023; arXiv:2210.03629. arxiv.org/abs/2210.03629

8. Schick, Timo, Dwivedi-Yu, Jane, Dessì, Roberto, Raileanu, Roberta, Lomeli, Maria, Hambro, Eric, Zettlemoyer, Luke, Cancedda, Nicola, and Scialom, Thomas. *Toolformer: Language Models Can Teach Themselves to Use Tools*. Advances in Neural Information Processing Systems (NeurIPS) 36: 68539–68551, 2023; arXiv:2302.04761. arxiv.org/abs/2302.04761

9. Wang, Lei, Ma, Chen, Feng, Xueyang, Zhang, Zeyu, Yang, Hao, Zhang, Jingsen, Chen, Zhiyuan, Tang, Jiakai, Chen, Xu, Lin, Yankai, Zhao, Wayne Xin, Wei, Zhewei, and Wen, Jirong. *A Survey on Large Language Model based Autonomous Agents*. Frontiers of Computer Science 18(6): 186345, 2024; doi:10.1007/s11704-024-40231-1; arXiv:2308.11432. arxiv.org/abs/2308.11432

10. Zhou, Shuyan, Xu, Frank F., Zhu, Hao, Zhou, Xuhui, Lo, Robert, Sridhar, Abishek, Cheng, Xianyi, Ou, Tianyue, Bisk, Yonatan, Fried, Daniel, Alon, Uri, and Neubig, Graham. *WebArena: A Realistic Web Environment for Building Autonomous Agents*. International Conference on Learning Representations (ICLR), 2024; arXiv:2307.13854. arxiv.org/abs/2307.13854

11. Xie, Tianbao, Zhang, Danyang, Chen, Jixuan, Li, Xiaochuan, Zhao, Siheng, Cao, Ruisheng, Hua, Toh Jing, Cheng, Zhoujun, Shin, Dongchan, Lei, Fangyu, Liu, Yitao, Xu, Yiheng, Zhou, Shuyan, Savarese, Silvio, Xiong, Caiming, Zhong, Victor, and Yu, Tao. *OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments*. Advances in Neural Information Processing Systems (NeurIPS) 37, Datasets and Benchmarks Track, 2024; arXiv:2404.07972. arxiv.org/abs/2404.07972

12. Wu, Qingyun, Bansal, Gagan, Zhang, Jieyu, Wu, Yiran, Li, Beibin, Zhu, Erkang, Jiang, Li, Zhang, Xiaoyun, Zhang, Shaokun, Liu, Jiale, Awadallah, Ahmed Hassan, White, Ryen W., Burger, Doug, and Wang, Chi. *AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation*. First Conference on Language Modeling (COLM), 2024; arXiv:2308.08155. arxiv.org/abs/2308.08155

13. Park, Joon Sung, O'Brien, Joseph C., Cai, Carrie J., Morris, Meredith Ringel, Liang, Percy, and Bernstein, Michael S. *Generative Agents: Interactive Simulacra of Human Behavior*. Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), Article 2, pp. 1–22, ACM, 2023; doi:10.1145/3586183.3606763; arXiv:2304.03442. arxiv.org/abs/2304.03442

14. Packer, Charles, Wooders, Sarah, Lin, Kevin, Fang, Vivian, Patil, Shishir G., Stoica, Ion, and Gonzalez, Joseph E. *MemGPT: Towards LLMs as Operating Systems*. arXiv:2310.08560, 2023. arxiv.org/abs/2310.08560

15. LofiSu. *deepseek-super-agent: Open-source implementation of the DeepSeek Super-agent architecture*. GitHub repository, 2026. github.com/LofiSu/deepseek-super-agent

16. thefullgc. *opensuperagent: Community implementation of event-driven Super-agent*. GitHub repository, 2026. github.com/thefull-gc/opensuperagent

17. Weiser, Mark. *The Computer for the 21st Century*. Scientific American 265(3): 94–104, September 1991; doi:10.1038/scientificamerican0991-94. doi.org/10.1038/scientificamerican0991-94

18. Weiser, Mark, and Brown, John Seely. *Designing Calm Technology*. PowerGrid Journal 1.01, July 1996 (reprinted in Beyond Calculation: The Next Fifty Years of Computing, pp. 75–85, Springer, 1997). calmtech.com/papers/designing-calm-technology

19. Dey, Anind K. *Understanding and Using Context*. Personal and Ubiquitous Computing 5(1): 4–7, 2001; doi:10.1007/s007790170019. doi.org/10.1007/s007790170019

20. Horvitz, Eric. *Principles of Mixed-Initiative User Interfaces*. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '99), pp. 159–166, ACM, 1999; doi:10.1145/302979.303030. doi.org/10.1145/302979.303030

21. Hennessy, John L., and Patterson, David A. *Computer Architecture: A Quantitative Approach (6th Edition)*. Morgan Kaufmann, Cambridge, MA, 2017; ISBN 978-0-12-811905-1.

22. Apple Inc. *ActivityKit: Displaying Live Data with Live Activities*. Apple Developer Documentation. developer.apple.com/documentation/activitykit

23. Apple Inc. *Human Interface Guidelines*. Apple Developer Documentation. developer.apple.com/design/human-interface-guidelines/

24. SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE Standard J3016_202104, April 2021. www.sae.org/standards/content/j3016_202104/

25. Saltzer, Jerome H., and Schroeder, Michael D. *The Protection of Information in Computer Systems*. Proceedings of the IEEE 63(9): 1278–1308, September 1975; doi:10.1109/PROC.1975.9939. doi.org/10.1109/PROC.1975.9939

iPhone, iPad, Mac, Apple Watch, Siri and Apple Pencil are trademarks of Apple Inc., registered in the U.S. and other countries and regions. Piko is an independent product of Prysai Lab and is not affiliated with Apple Inc.

DOI: 10.5281/zenodo.24429625 · CC BY 4.0