PART II

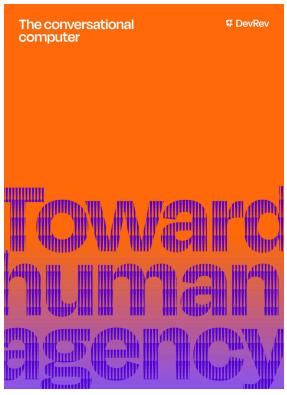
The collaborative computer

Toward team intelligence



DevRev

When agency becomes collective



Part I:
The conversational computer

If conversational agents mark the computer's progression toward human-level agency, then the following question is unavoidable: what does agency look like beyond the individual? Human intelligence rarely operates in isolation; it compounds in teams—through shared context, accumulated decisions, and institutional memory. Yet while we are rapidly formalizing skills and reasoning for AI agents, we have largely left team intelligence implicit and fragmented. Here, we define team intelligence as being a combination of trusted answers and safe actions, built around shared memory and multiplayer AI agency.

Today's multi-player collaboration tools treat knowledge as transient and siloed, assuming teams mostly think locally and synchronously; this assumption breaks at scale. Just as agents require structured memory and executable context to act with agency, teams require shared memory to enable them to think and act collectively. Here in Part 2, we explore how adopting a true knowledge graph over passive, unstructured collaboration systems enables the path to team intelligence.

The multi-player necessity

Team intelligence only becomes real when the enterprise is represented as it actually exists: as a hierarchy of teams, leaders, and cross-functional groups, not a flat list of users. That structure extends beyond the company itself to include the customer organization on the other side. Modeling both hierarchies explicitly is what allows the system to understand ownership, relationships, escalation paths, and the broader context in which work happens.

Post-COVID, remote work and globally distributed offices have gained traction, significantly cutting down those water cooler moments. And since that ChatGPT watershed moment, we aren't just collaborating with people—LLMs and agents are playing an important role being there with us, all the time.

- Knowledge discovery is more computationally mediated than ever before
- We increasingly begin collaborative work with question–answer and directive command sessions with agents, followed by the inclusion of additional participants who expand upon the chains of reasoning that emerge.
- Agents need to understand "when to chime in"—being overly eager creates more noise than insights

The productivity imperative

Knowledge workers now toggle between more than 10 tools daily [1], and the cognitive load has become unsustainable. We use the term *articulation work overload* to draw attention to a pattern we repeatedly encounter and believe merits better support: the invisible labor of stitching together fragments of work across systems to make the collaborative work happen.

Teams spend 30–40% of time on coordination overhead rather than on producing new outcomes:

- Finding decisions that were made
- Debating decisions that were already decided (because people don't know)
- Re-communicating decisions to new team members
- Waiting for information scattered across channels



The problem: fragmented collaboration

The new abnormal

Contemporary hybrid work is marked by a constantly shifting patchwork of tools, channels of communication, and agents that repeatedly tear at the threads through which collaboration is woven. The work of communicating, knowledge sharing, coordinating, sense-making, and deciding is distributed across countless chat channels, documents, tickets, dashboards, and AI assistants. As a result, this work is scattered, overlapping, and only partially visible to any one participant. What is missing is not data, but the means to assemble these fragments into shared, durable forms of work. The fundamental problem isn't information scarcity, but rather the absence of structure that binds communication, coordination, and knowledge into coherent, cooperative artifacts. This fragmentation unfolds across all levels of work, giving rise to interrelated challenges for both individuals and organizations alike.

Organizing around channels

With a patchwork of tools in play, today's teams often plan and track work in one set of tools and communicate about the work in another. As a result, decisions often remain buried in transient conversations instead of being captured in the shared objects that people rely on to coordinate day to day, and through which practices become part of organizational knowledge and accountability.

On an individual level, this leads to increased coordination and cognitive load. Workers must continuously reconstruct "what's going on" by piecing together scattered messages, boards, and documents.

On a team level, this fragmentation fuels misalignment across individuals, slows decision-making, and complicates sense-making for workers and managers, as people are forced to interpret noisy, partial cues about what others are doing and what the current priorities really are. The reality lies somewhere between DMs, emails, and scattered notes.

As employees struggle to keep up with the invisible work of reassembling a coherent picture, the stress and burnout eventually hampers organizational productivity.

Many existing collaborative technologies were built to enable communication at scale and make the work of sharing, annotating, and linking information easier. The intent was great: you can share documents, send emails to whomever, and set up countless message threads. But the reality of contemporary knowledge work tells a different story. Instead of creating clarity, these tools have fragmented work, buried decisions, and shifted the sense-making burden onto the people doing the work. Multi-player ends up multiplying complexity.

The fundamental problems with unstructured channels

Collaboration exists everywhere—not in a destination

Unstructured channels assume collaboration happens within a specific message thread, email, or document. But work happens across entities:

- **Accounts:** Sales + Support + Engineering + Finance all have discussions tied to an account
- **Opportunities:** Sales, Product, and Technical Sales collaborate on deal strategy
- **Tickets:** Support, Engineering, and Product discuss solutions
- **Incidents:** On-call, Engineering, Product, and Comms coordinate response
- **Issues:** Engineering and Product plan work
- **Pull Requests:** Code review, architecture discussion, team knowledge

While such channels over-index on a specific type of streamlined engagement, modern work demands more:

1. **Embedded collaboration:** Collaboration should happen where work is (inside the CRM, inside the ticketing system, inside the project management tool)
2. **Cross-tribe visibility:** Your customer (B2C/scaled B2B) needs to see collaboration context without joining your internal channels
3. **Unified knowledge:** A single source of truth, not siloed conversations with artificially constrained context

And of course there's a matter of scale: if each entity—thousands of accounts, tickets, issues, and so on—defaults to a basis of communication, you need potentially millions of "channels". Channels can't scale this way; for instance, you can't invite 50,000 customers to your #customer-feedback channel, you can't tag everyone in the document, and CC'ing everybody on an email doesn't solve the problem.



Syntactic friction

By encoding collaboration knowledge into naming conventions and tribal memory, unstructured channels introduce a persistent discovery tax: inferring intent from naming conventions. Is #product-eng-sync for product managers coordinating with engineering, or engineers discussing product requirements? Does #tmp-mcp-9-9 still matter, or was it abandoned after the September 9th? When #customer-success and #cs-team both exist, which one is canonical?

This friction cannot be eliminated through reorganization—every attempt to rationalize structure simply creates new conventions that require tacit organizational knowledge to navigate. Both creators and contributors must maintain constant alignment: "Did you post this in the right place? Should we move this discussion? Are we duplicating a conversation happening elsewhere?". Without properly scoped boundaries, every convention leads to compromise.

All of this breeds invisible human labor—the institutional initiation required to know where information lives, the constant disambiguation of overlapping spaces, the reconciliation of redundant discussions. New employees aren't learning your business, they're learning conventions. This is tribal knowledge masquerading as information architecture, and it scales linearly with every person and every handoff.

Syntactic friction means compounding friction: every team member wastes time on tribal knowledge lookup rather than actual work. New team members onboard into chaos and, most importantly, reference-worthy knowledge gets forgotten or archived.

Arcane search

Search has been the antidote to this century's data explosion — on the internet, at least. And while conversations in unstructured channels may appear similar to internet content, there's a large contextual gap: no links, no document object model, no page tree. With channels, time is the only organizing principle.

Moreover, with businesses operating fluidly between tools, context has a tendency of either fading or evolving, and your unstructured channels cannot compensate for this context drift. There's serious implications here:

- **Context is invisible:** "We're moving to PostgreSQL" and "PostgreSQL adoption" and "database migration Q3" are semantically identical but textually different
- **Re-ranking is absent:** Search returns volume (all messages mentioning "PostgreSQL"), not relevance (what decision matters to my work right now?)
- **No semantic clustering:** Related discussions scattered across channels (#database, #backend, #infrastructure) never surface as a coherent topic

Information exists but is inaccessible — a form of data death. The cost of finding answers exceeds the value of having them. And this, in turn, hurts the quality of collaboration within the organization.

By contrast, today's search has advanced well beyond this into query understanding, personalized answers, and next-best-action suggestions. But to achieve this, modern search expects an alternative strategy for multi-player interaction.

Bots: static automation cannot adapt

These same tools often offer bots as triggered functions waiting for invocations; they respond to slash commands, react to @mentions, and execute predefined workflows. But in the modern AI context of agency, they are massively limited:

- Bots aren't proactive: A bot cannot proactively attend to a conversation to surface relevant context or flag a decision
- Bots don't evolve: Skills are hardcoded, so expanding bot capabilities requires engineering.
- Bot can't learn from context: Every interaction is isolated, so bots don't develop an understanding of team dynamics, projects, or decisions

In short, they create false confidence in automation, when true multi-player needs adaptive intelligence. The resulting experience is often clumsy:

- Bots reminding you of a task three weeks after a decision that invalidates it
- Stale knowledge as to which actual quarter "Q2" is referencing to



Toward modern collaboration

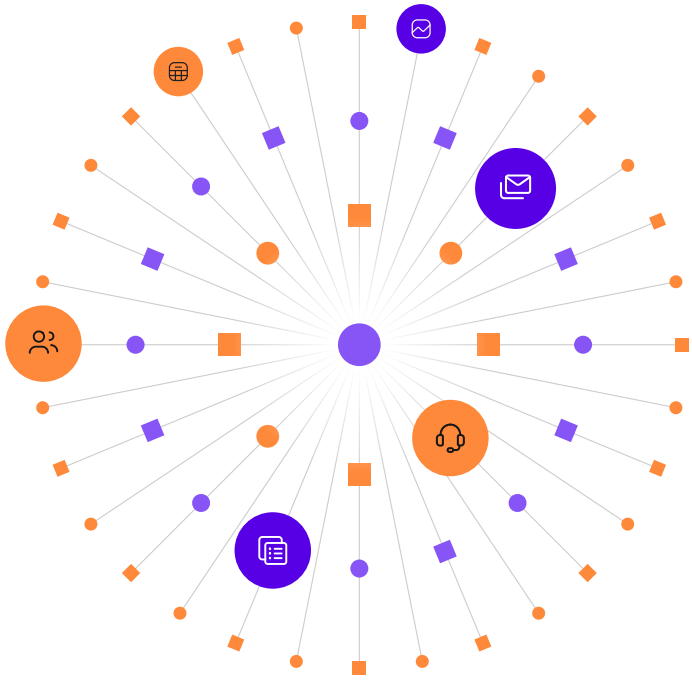
So how do you break the cycle? Through relationships and embeddedness, forming a knowledge graph.

Researchers have long shown that successful collaborative work depends on ongoing articulation work through which responsibilities, dependencies, and meanings are aligned [2, 3]. Without a knowledge graph, this articulation often takes place in transient channels, while the artifacts of work—tickets, code, and, dashboards—remain only loosely connected to the articulation that produced them. Given this articulation work is precisely what establishes context, the absence of durable links between articulation work and actual artifacts renders much of this context fragile and forgotten.

By contrast, the knowledge graph enables emergent mapping of relationships between work, product, people, and conversations. In the knowledge graph model, conversations are embedded in the system of record, allowing intent to be inferred from natural discussion within the broader work context. Moreover, implicit relationships between the work you do, the customers you serve, and the products you create are made explicit. No tribal knowledge required. No new syntax to learn. Just colleagues talking, with intelligence operating on the actual knowledge graph rather than shouting through API keyholes.

Comparison of collaboration modes

Key Elements of Modern Collaboration	Channel-Based	Knowledge Graph-Based
Coordination Substrate	Static, predefined containers	Dynamic, emergent relationships
Collaboration Mechanism	Channel-specific	Context-aware across entities
Information Discoverability	Socially mediated i.e."If you know, you know"	Semantic search and recommendations
Cognitive Scalability	Breaks at 50+ channels	Scales to millions of relationships
Agentic Scope	Executes commands	Understands context, acts proactively
Relationship to Work	Loose, decoupled	Explicit, embedded
Durability	Ephemeral, local	Persistent, organizational



Real-world usage scenario

	Channels	Knowledge Graph
Navigation	You join #account-acme. Discussion about SLA. Linked ticket in Jira. Decision thread in email. You manually connect the dots.	You view the Account-ACME entity, which is connected to the current SLA, active tickets, decision history, related team members, and revenue impact. All in context, not scattered.
Facilitation	Channels are **flat**, so all conversations carry equal weight, and all context must be inferred. No proactive clustering, classification by type, routing based on context, or pattern analysis.	With semantic relationships, systems can: ▪ **Cluster** all "API deprecation" discussions and surface the consensus decision ▪ **Classify** conversations using NLP (e.g., flag a thread as a decision) ▪ **Route** work contextually (e.g., send a new ticket to the team with domain expertise) ▪ **Analyze** through graph traversal (e.g., identify decisions that caused the most downstream rework)
Organization	▪ Support ticket created in Zendesk ▪ Support team discusses in #support ▪ Links to Jira issue ▪ Engineering discusses in #backend ▪ Customer waits for email updates	▪ Ticket is a node in the graph ▪ All collaboration is embedded in the ticket, not a separate channel ▪ Supports both internal discussion and customer visibility (controlled) ▪ Customer sees progress in context without joining a channel

Unlocking team intelligence

Team intelligence is the institutional know-how that supports coordination across individuals, carried through shared understanding of decisions, priorities, and lessons embedded in ongoing work. It's the informal reasoning that helps synthesize, to make sense of it all.

While every organization possesses this collective intelligence, most collaboration systems fail to retain and support it. Unstructured channels treat knowledge as conversational and ephemeral, causing context to evaporate at every handoff, phase change, or organizational boundary. The decision remains, the rationale is forgotten.

The knowledge graph model, as built into Computer, stabilizes this informal reasoning into durable organizational knowledge. Communication is no longer separate from execution; conversations, decisions, and outcomes become embedded in a unified structure that supports coordination, accountability, and learning over time. This is how team intelligence comes alive—not locked in human memory, but encoded as Shared Memory. Accessible in both multi-player and single-player mode, leveraging hardened skills to power trusted answers and safe actions.

In fact, researchers have a term for this fragile synthesis layer that wraps together all the informal reasoning: transactive memory system (TMS). Computer by DevRev differs from other collaboration tools because it upholds the transactive memory that enable teams to memorialize who knows what and how to coordinate work accordingly.

TMS theory describes how to maintain meta knowledge of who knows what, distribute cognitive labor, and coordinate work based on one another's expertise. Recent work in this field extends this logic to AI-enabled teams, showing that intelligent agents can actively participate in encoding, retrieving, and allocating knowledge across the team [4, 5]. The big unlock isn't about breaking data silos, it's about eliminating expertise silos. It enables teams to think together across tools, time, and people.

And because every discussion is embedded directly into the tickets, accounts, and opportunities it affects, to name a few record categories, Computer can shape team intelligence. It can surface the reasoning, tradeoffs, and actors historically involved, for instance. It can proactively point teams toward relevant expertise, historical dependencies, and missing stakeholders. This approach aligns closely with TMS, which identifies such processes as markers of a hybrid group mind, or simply team intelligence. As Leonardi observed, team intelligence emerges when workers encounter persistent digital traces of others' communications [6]. The result is a system that simplifies coordination, speeds onboarding to projects, and supports more intelligent collaboration.

"This is how team intelligence comes alive—not locked in human memory, but encoded as Shared Memory. Accessible in both multi-player and single-player mode, leveraging hardened skills to power trusted answers and safe actions."

Integrated conversations, embedded in the business

Computer's shared memory also functions as what Dourish [7] describes as a technology of accountability, a system that renders organizational activity observable and reportable to the various stakeholders who depend on it. According to Dourish, traditional workflow technologies succeed not merely because they coordinate tasks, but because they provide a means for organizations to account for their work to others by presenting it in a structured, intelligible, and externally rational way.

By embedding conversations, decisions, rationales, and timelines into the entities where work unfolds, Computer's trusted answers can help surface what is being done, why it is being done, and how it relates to broader organizational commitments. These embedded representations may function as accounting resources through which organizations make their work more coherent, inspectable, and accountable to stakeholders. Examples here would be engineering teams seeking clarity, leadership evaluating decisions, or customers monitoring the status of their requests.

Key Benefits	Example
Context is automatic	When you view an entity, you see all relevant conversations
History is preserved	Decisions and discussions are tied to known entities, not ephemeral chat
Discoverability improves	Search for an entity → find all related discussions automatically
New team members onboard faster	Instead of "read 6 months of channel history," they view the entity and see the story

You create knowledge, not threads

In a knowledge graph, collaboration is embedded in entities. So when a decision is made, it's tied to the related work—issue, opportunity, ticket, etc.—with semantic understanding of the decision rationale, ownership, timeline, and history. Moreover, this information can be clustered and annotated to allow topics to surface, further bolstering knowledge.

By contrast, conversations in unstructured channels are simply conversations; even if you mark something as important, it's still embedded and indexed within the thread context, as opposed to a deeper relationship with work, product, and customer.

So when an enterprise customer asks why their feature request isn't being prioritized, the discussion can simply embed itself into the feature request ticket; the access provisions of both customer and PM govern what they respectively see without requiring a separate channel. Team members can then get trusted answers that combine customer commentary with the priority and status of the request to take actions safely.



Your customer is in the team

This approach also helps bring the customer—including vendors and partners—into the team fabric:

- **Selective visibility:** External parties see only the entities relevant to them (their Account, their Tickets, their Opportunities)
- **Low overhead:** Visibility is controlled at the entity level, not by creating channels
- **Embedded collaboration:** Customers see discussion within their ticket, not in a separate channel they have to join

Your agent is in the team

Agents are inside the team, not outside the organization looking in; they're team members that understand context about your company. Enterprises function through hierarchies of teams that together capture the full structure of the company, as do your customers. An agent that understands teams, hierarchy, and the relationship between the enterprise and its customers can participate in work the way the organization actually operates.

Key Advantage	Detail	Examples
Proactive intelligence	Scan collaboration across entities and surface relevant context	You're working on API redesign. Here's the decision from 2 weeks ago on versioning strategy. This team member objected. Might be worth alignment.
Adaptive helpfulness	Understand what each team needs	Sales team asks about "budget" → agent knows this means "contract value" in this context, not "financial plan" Engineering team asks about "budget" → agent knows this means "complexity/scope"
Decision facilitation	Help teams make decisions by surfacing relevant context	3 teams have opinions on database choice. Let me summarize their constraints and trade-offs.
Knowledge evolution	Learn from feedback	Agent recommended a solution. Team said "tried that, didn't work." Agent remembers and won't recommend it again in similar contexts.
Gap identification	Spot when collaboration is missing	This architectural decision affects 5 teams. Only 2 have been consulted. Should I surface to the others?

Why this matters even more now

Understanding today's workplace

The end of synchronous privilege

Today's generation entering the workforce expects tools that don't privilege synchronous presence; they want meritocracy of ideas, not proximity to power. Tools that separate "where we talk" from "where we work" inherently favor whoever's online at the right moment. On the other hand, embedding conversations within work objects supports a more equitable form of collaboration. This transformation is especially vital in today's distributed, AI-augmented teams that demand conversation and work artifacts coexist.

Asynchronous-first work demands persistent context

You can't build a global company on "you had to be there" culture. Conversations need to live where decisions get executed, not in a separate stream that privileges whoever happened to be online. The shift is not simply toward remote work but toward the expectation of temporal equity: every team member, regardless of timezone or schedule, should have equal access to context and decision-making power.

The distributed memory crisis

Institutionally, distributed work has created another critical problem: institutional memory is evaporating. When your engineering team in Bangalore discusses a critical decision in a channel at 9 PM PST, your product team in SF wakes up to 200+ messages they'll never read. The context lives in someone's head or dies in a thread. Meanwhile, turnover has accelerated—the median tenure at tech companies is now under 2 years. When people leave, their channel conversations become archaeological artifacts disconnected from the work they reference. You're not just losing people; you're losing the why behind decisions.



The AI compliance imperative

Post-pandemic regulatory guidelines require audit trails showing who decided what and why. Channel-based systems lack the structured accountability that modern compliance and AI governance demand:

- Decisions are in channel conversations
- Contracts are in Salesforce
- Tickets are in Zendesk

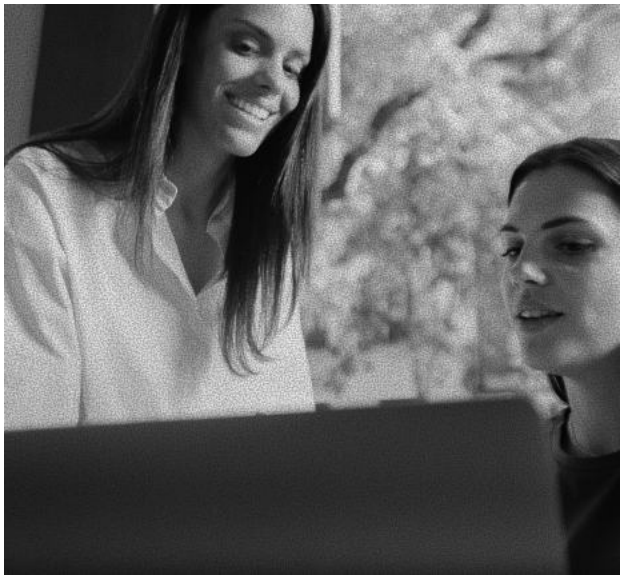
For SOC2, HIPAA, or financial audit compliance, you can't prove decisions were made correctly because the evidence is scattered. Knowledge graphs centralize this to provide trusted answers and safe actions.

The AI readiness imperative

Organizations are being asked to deploy AI agents while their knowledge infrastructure is fundamentally unsuited for it. Teams that structure work around channels can't leverage AI effectively:

- LLMs need structured context, not just chat transcripts.
- You can't ask "show me all decisions that caused rework" if decisions are scattered in channels
- You can't build predictive models without understanding entity relationships

Organizations that structure work around knowledge graphs are AI-ready; when conversations are tied to tickets tied to customers tied to outcomes, data is already in a form AI systems can reason about.



Conclusion

The core insight

Unstructured channels as a primary mode of collaboration struggle due to three failed assumptions:

- Information doesn't stay in one place—it's scattered across CRM, ticketing, code, email, and beyond
- Teams don't work in isolation—collaboration is cross-functional and fluid
- Users don't always know where to look, and tribal knowledge is a tax on productivity

Knowledge graphs align collaboration with how work actually happens — through entities (Accounts, Tickets, Opportunities, Code, Decisions) that have inherent relationships. When embedded in organizational practice, knowledge graphs and agents can reinforce team intelligence, the know-how that sustains coordination across teams.

The competitive advantage

Organizations that move to knowledge graph-based collaboration first will:

- Execute faster (less coordination overhead)
- Scale better (no left-nav explosion, no tribal knowledge)
- Retain talent (less friction, clearer context)
- Leverage AI (structured data ready for intelligent agents)

Knowledge graphs give teams a shared memory; the next step is making it useful at the speed of work by reimagining what enterprise search can be atop this memory. In the next chapter, we advocate for a more actionable search paradigm, one that sidesteps traditional retrieval results in favor of real answers you can trust—grounded, permission-aware, and ready to turn answers into action.

References

[1] Murty, R. N., Dadlani, S., & Das, R. B. (2022). How much time and energy do we waste toggling between applications?. Harvard Business Review.

[2] Schmidt, K., & Bannon, L. (1992). Taking CSCW seriously: Supporting articulation work. Computer supported cooperative work (CSCW), 1(1), 7-40.

[3] Duckert, M., & Bjørn, P. (2024). Revisiting Grudin's eight challenges for developers of groupware technologies 30 years later. i-com, 23(1), 7-31.

[4] Wegner, D. M. (1987). Transactive memory: A contemporary analysis of the group mind. In Theories of group behavior (pp. 185-208). New York, NY: Springer New York.

[5] Hopf, K., Nahr, N., Staake, T., & Lehner, F. (2025). The group mind of hybrid teams with humans and intelligent agents in knowledge-intense work. Journal of Information Technology, 40(1), 9-34.

[6] Leonardi, P. M. (2015). Ambient Awareness and Knowledge Acquisition: Using Social Media to Learn "Who Knows What" and "Who Knows Whom"1. MIS Quarterly, 39(4), 747–762. https://doi.org/10.25300/MISQ/2015/39.4.1

[7] Dourish, P. (2001). Process descriptions as organisational accounting devices: The dual use of workflow technologies. Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work - GROUP '01, 52. https://doi.org/10.1145/500286.500297



DevRev