ATAPIC

PICK ON THE ATTACK

Missions: Autonomous Go-to-Market Agents Under Human Authority

An architecture and product paper on ATAPIC's Agent Center

Enahoro Omoarukhe

Founder & Lead AI Engineer, ATAPIC

TECHNICAL WHITE PAPER • v1.0 • SEPTEMBER 2026

ABSTRACT

Most AI sales tools automate a step: they write an email, score a lead, or book a meeting. The work that actually decides whether a company grows is the loop around those steps. Somebody has to decide who to target, find the right people, learn what they care about, reach out at the right moment, follow up, notice who answered, and adjust when a tactic is not working. That loop is where small teams lose their week. This paper describes Missions, the system behind ATAPIC's Agent Center. A user states an outcome in plain language, for example "book ten meetings with finance leaders at fintech companies this month." A strategist model turns it into a written plan, and each part of the plan becomes a scheduled agent on the same runtime that powers the live assistant. A supervisor measures progress against the plan's own targets and adapts it. The user chooses how much the system may do alone, and that choice is enforced in the engine, not requested in a prompt. We describe the architecture, the policies that keep an autonomous system honest, the failure modes we met in production and how each was closed, and why we believe the right unit of automation for revenue work is a goal with a plan, not a task.

1. Motivation: automating the loop, not the step

A capable assistant can already do almost every individual step of outbound selling. It can search for people, read their public activity, draft a message, and send it. Yet teams that adopt these tools often find that their week looks the same. The reason is that the steps were never the bottleneck. The bottleneck is the loop that connects them, and the loop is run by a person:

- deciding who is worth the time this week, and why;
- turning that decision into searches, lists, and messages;
- remembering to follow up, and remembering to stop when someone replies;
- noticing that one channel works and another does not, and changing the plan.

Every one of these is a judgement call made daily. Tools that automate a single step leave the person running the loop, so the time saved on each step is spent operating the tool. The obvious alternative, a fully autonomous agent pointed at a vague goal, fails differently: it runs, but nobody can say what it is doing, why, or whether it is working, and it has no notion of when to stop and ask.

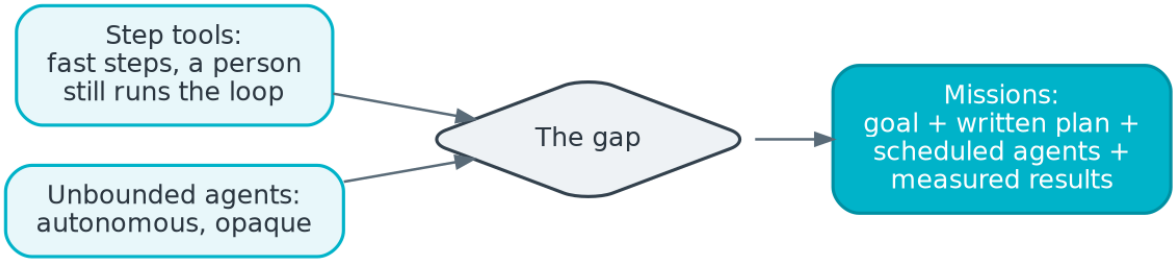


Figure 1. Missions fill the gap between tools that automate a single step and unbounded agents that nobody can inspect.

Missions sit between these. The loop is automated, but it is automated against a plan that a person can read, approve, and change, and every action it takes is recorded with its reason.

2. What a Mission is

A Mission is a goal with four things attached to it.

1. **A plan.** Who to target, how to position the offer, which plays to run, the numbers to hit, and the risks to watch. It is written in plain language and shown to the user before anything runs.
2. **Agents.** Each scheduled play in the plan becomes its own agent with its own clock: one prospects every morning, one researches the best candidates, one works LinkedIn, one runs email, one writes content.
3. **A lead list.** Every Mission owns a list. Everything its agents find is attached to it, so the Mission's pipeline is always a concrete set of people, not a statistic.
4. **A supervisor.** A recurring process that measures the Mission against its targets, records what happened, finishes it when its window ends, and proposes changes to the plan when a play is not working.

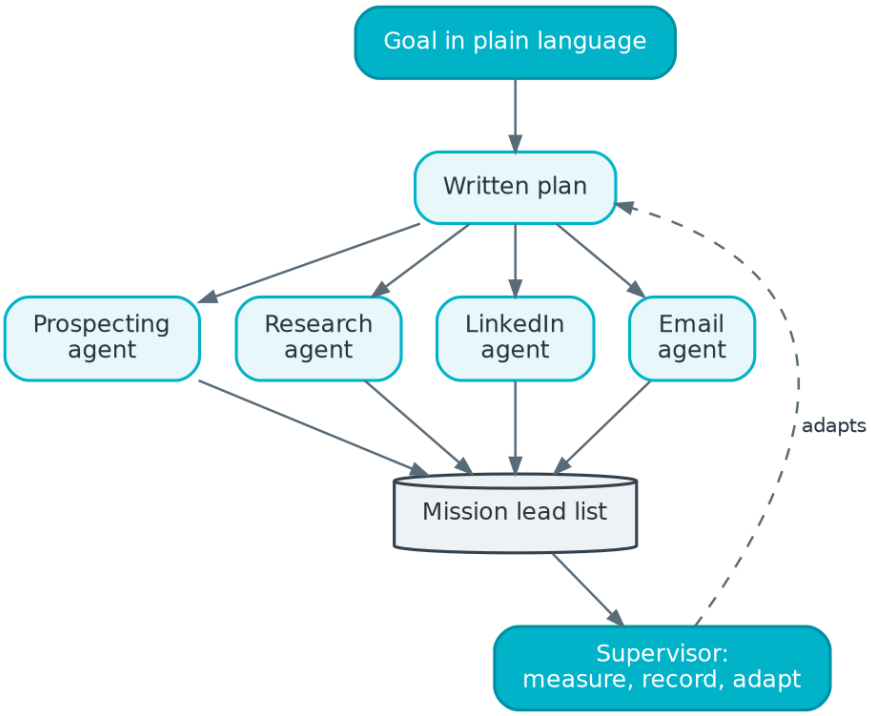


Figure 2. Anatomy of a Mission: a goal, a written plan, one scheduled agent per play, a lead list, and a supervisor that measures and adapts.

The important property is that nothing in a Mission is hidden. The plan is a document. The agents are real, listed, and can be switched off individually. The list is a real list. The supervisor's reasoning is written into the Mission's own activity log.

3. From goal to plan

3.1 The strategist

The user's goal, the channels they allow, the time window, and what the company sells (read from its profile) are given to a strategist model. It returns a structured plan: an ideal customer profile, positioning, three to seven plays, targets for the key numbers, and the risks it sees. The strategist is tried across an ordered chain of frontier models. A model that refuses or errors is set aside for an hour and the next one is tried. Every attempt is recorded, and the Mission stores which model actually wrote its plan, so "who planned this?" always has a visible answer.

3.2 Rules decide what may run, not the prompt

A model's plan is a proposal, not an instruction. Before anything is saved, a set of deterministic, unit-tested rules decides what of the plan may run:

- plays on channels the user switched off are dropped, whatever the model wrote;
- schedules, run times, and lanes are normalized to values the scheduler understands;
- the number of plays is capped;
- only known target metrics are kept, so a dashboard never shows a number nobody measures;
- instructions are trimmed to a safe length.

If no model answers at all, the Mission is not left empty. A starter plan, written by people and clearly labelled as such, is used instead. We call this "never return zero": the user always gets something they can act on and an honest label saying where it came from.

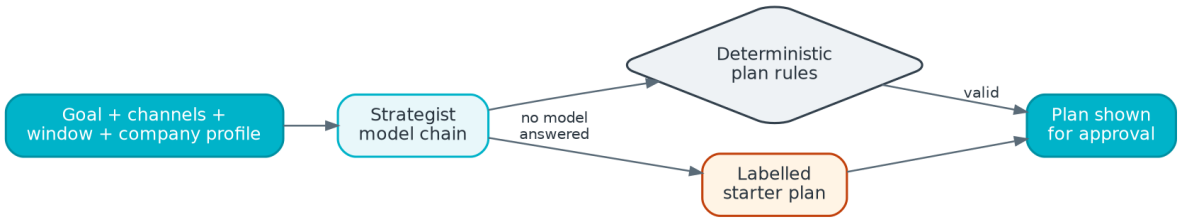


Figure 3. From goal to plan. A strategist model chain proposes; deterministic rules decide what may run; a labelled starter plan is used if no model answers.

4. From plan to agents

Each scheduled play becomes a real agent on the same engine that runs the chat assistant and hand-configured agents. This was a deliberate choice. We did not build a second execution system for autonomy. An agent created by a Mission can search, research, save people, create LinkedIn tasks, write emails, and post content with exactly the same tools, the same permissions, and the same safety checks as everything else in the product. When the assistant gets a new capability, Missions get it too.

Agents run on a clock set in the Mission's own time zone. A central tick runs every fifteen minutes and starts plays whose time has come. Plays that fall due together are started a couple of minutes apart, because we learned in production that a burst of simultaneous model calls is answered with rate limits. Rate-limit responses are retried with widening waits instead of being recorded as failures.

A Mission agent works its own list. Its instructions carry the list's identity, so a LinkedIn play asks for "the next people on this Mission's list who have not been contacted," creates their tasks, and moves on.

5. Autonomy as an enforced policy

The single most important design decision in Missions is that the amount of autonomy is chosen by the user and enforced by the engine. It is never delegated to the model's judgement.

Mode	Plan	Sending	Plan changes
Autopilot	Launches on its own	Sends	Applies the supervisor's changes itself
Co-pilot (default)	Approved once by the user	Sends once the plan is approved	Pauses apply themselves; rewrites and new plays ask first
Review all	Approved by the user	Never sends; drafts only	Every change asks first

A system prompt that says "do not send emails in review mode" is a request. A model can misread it, and a prompt can be edited. Instead, the autonomy of the running Mission travels with the execution itself, and every tool that could send, activate, or enrol is checked against it before it runs:

- in Review all, an email or newsletter send is converted to a preview, and activating or enrolling a sequence is refused with a reason the agent can read and act on;
- in Co-pilot and Autopilot, LinkedIn tasks are created already approved, so the browser extension carries them out without a second click, and personalised email campaigns are approved and handed to the send scheduler, which still respects the sending window and mailbox limits;
- if the engine cannot tell which Mission an agent belongs to, it assumes Review all. The safe default is to draft.

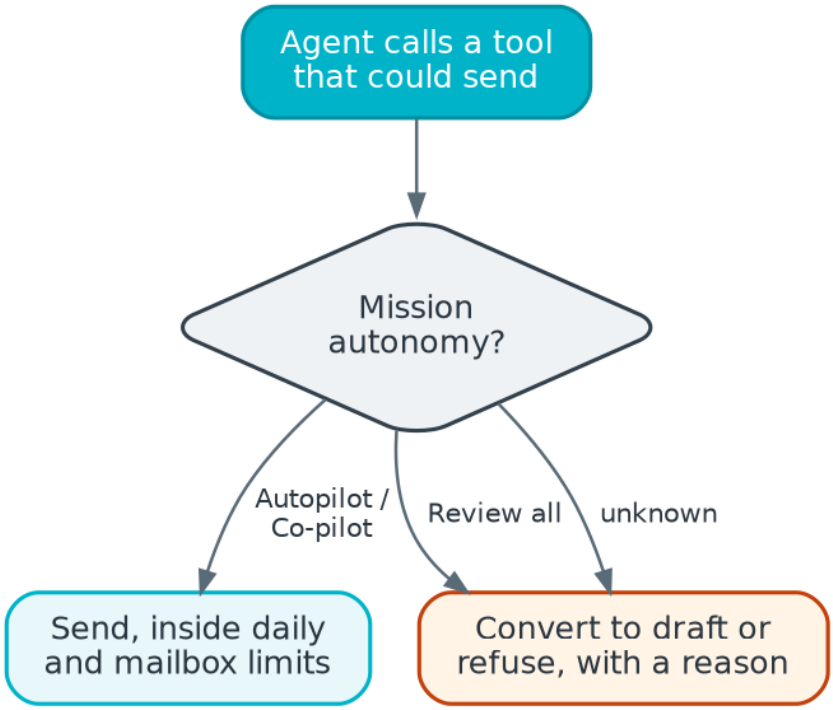


Figure 4. Autonomy is enforced in the engine. Every tool that could send is checked against the Mission's mode; an unknown mode drafts.

The policy function is pure and unit tested. The same inputs always produce the same decision, which is what lets us state the behaviour in a paper with confidence.

6. Quality before quantity: the lead gate

An autonomous system that saves the wrong people is worse than one that saves nobody, because nobody looks before the outreach starts. Our early Missions showed this clearly. Searches driven by public posts returned the authors of posts, company pages, interns, and people whose role had nothing to do with the target. When a person searches, they skim and discard these. An agent saved them.

Missions now search the way the product's best search does: a structured people source first (titles, seniority, and company facts), with the signal-driven source as the fallback when the first returns nothing. Then every result passes a gate before the model ever sees it:

- the role and company are separated out of free-text headlines;
- company pages and junior roles are removed;
- the role must match the plan's target roles by function (with acronyms expanded, so "CFO" matches "finance"), and by seniority when the target is senior.

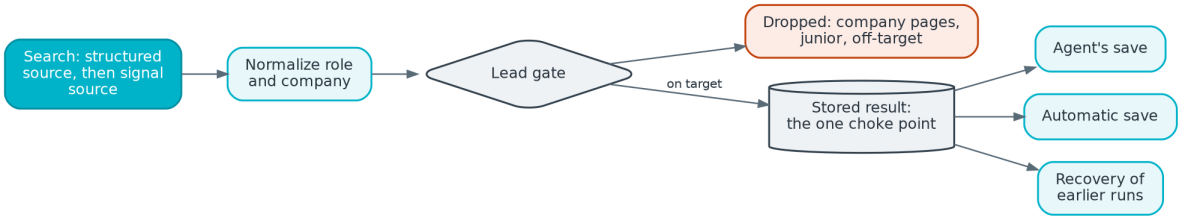


Figure 5. The lead gate sits at the one choke point every save path reads, so no unfiltered person can reach a Mission's list.

The gate sits at the one place every later path reads from, the stored search result. The agent's own save, the engine's automatic save, and the recovery of older runs all read the same filtered set, so there is no path by which an unfiltered person reaches a list. For people saved before the gate existed, a preview-first clean-up shows exactly what it would remove. Anyone with history (a message sent, a reply, a deal) is only taken off the Mission, never deleted.

7. Research before reaching out

Found is not the same as ready. For the best candidates on a list, a research play runs a deep dive: recent posts, company signals, and a short set of openers, each tied to its source. The research is attached to the person, so when an owner opens someone from the Mission's pipeline they see what the agent read, what it concluded, what it drafted, and the timeline of everything that has happened. We limit deep dives per run, because research costs time and money and a Mission should spend both on the people most likely to answer.

The reasoning that decides who goes first is the same reasoning the assistant uses in chat. First-party engagement (someone engaging with you) outranks their own posts about the problem, which outranks a recent role change, which outranks company events, which outrank a bare title. A message should be grounded in whichever signal was strongest.

8. Replies change everything

Outreach systems usually treat a reply as a statistic. For a person, a reply is the moment the plan should stop and a human should take over. When someone answers, on email or LinkedIn, Missions:

- pause that person's sequences;
- cancel their queued or approved LinkedIn actions (the connections check is kept);
- move them to Responded;

• send a high-priority notification to the lead's owner.

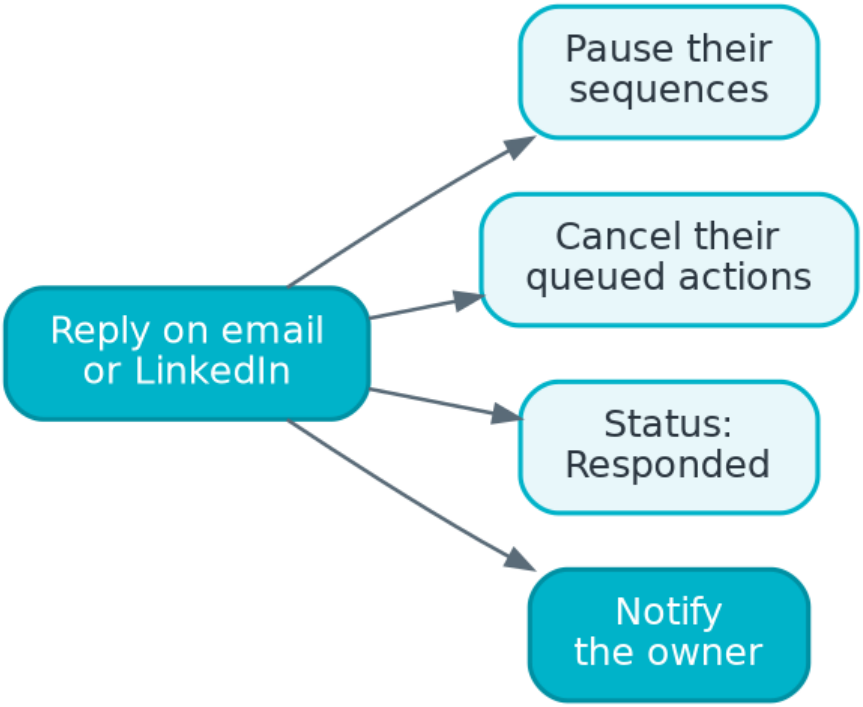


Figure 6. A reply stops the machine for that person and hands the conversation to a human.

LinkedIn replies arrive through the browser extension, which reads the inbox on a schedule. They are matched to a lead only by a unique exact name, recorded once, and never used to create new leads. An ambiguous match is ignored rather than guessed.

9. The pipeline tells the truth

Each person on a Mission's list sits at a stage: found, researched, hot, engaged, invited, connected, messaged, in a sequence, replied, won, or stopped. The rule for stages is strict: a stage is claimed only from a recorded fact. A queued connection request is not an invitation. A drafted email is not a message. A person is "invited" only when the extension reports the request as sent, and "connected" only when the connections check sees them.

This matters because the most damaging thing an autonomous system can do is report progress that did not happen. An owner who sees "100 contacted" and discovers that nothing left the building stops trusting every other number on the page.

10. The supervisor

Every fifteen minutes the supervisor visits each live Mission. It:

1. obeys the organization's emergency stop, which pauses every Mission and agent at once;
2. finishes Missions whose window has ended;
3. measures the Mission's numbers since it started, each from the table that already records that fact. The Mission keeps no copies, so a missing data source blanks only its own number;

4. records "target hit" moments;
5. starts due plays and releases queued work that the Mission's autonomy allows;
6. once a day, asks a model to review progress against the plan and propose changes.

Proposals follow the autonomy rules in Section 5. A question the supervisor cannot answer from data becomes a "Needs you" card for the owner. The owner's answer is added to every play as standing guidance, so a decision is made once, not re-asked daily.

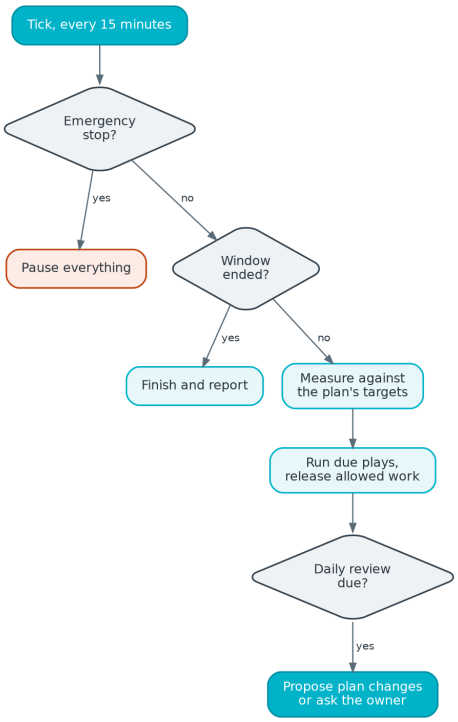


Figure 7. The supervisor's tick: obey the stop switch, finish expired Missions, measure, run due work, and review the plan daily.

11. What production taught us

Missions have run on real accounts. The failures we met were not exotic. Each was a place where an autonomous system looked healthy while doing nothing useful. We list them because they generalize.

Failure	What it looked like	The fix
Found is not saved	Runs reported "found 16" and ended green; nobody reached the list	The engine saves a Mission run's results itself if the agent did not, and recovers earlier runs from their stored search, marking them done only after a save that succeeded
Wrong people	Company pages, interns, and off-target roles were saved	Structured search first, then the lead gate at the single choke point every save reads
Silent duplication	An update failed on an over-long field, and every tick minted a new agent	Short summaries in bounded fields, full instructions where there is room, and a sweep for orphaned agents
Bursts	Simultaneous plays were answered with rate limits	Staggered starts and retries with widening waits
Approval debt	Hundreds of LinkedIn actions queued before the send policy existed waited for a click that never came; LinkedIn showed zero accepted	Sending Missions release their own queued work each tick, with duplicates removed first
Unexplained failures	A red run with no reason	Every failed step is recorded with its reason and shown on the play

The common lesson is that an autonomous system needs invariants, not just features. Examples: a stage is claimed only from a fact; every save reads one filtered source; the safe default is to draft. Each invariant closed a class of failures rather than one bug.

12. By the numbers

These are the design parameters that bound a Mission today. They are not measured outcomes. Outcome metrics (reply rate, meetings per hundred contacted, time saved) are being collected and will be reported in a future revision.

Parameter	Value
Plays per Mission	3 to 7
Live Missions per organization	5 by default
Supervisor cadence	Every 15 minutes
Plan review	Once a day by default
Plays falling due together	Started about 2 minutes apart
Rate-limit retries	Three, at widening intervals
Refused strategist model	Set aside for one hour
Deep dives per research run	Up to 8 people
Pipeline stages	11, each claimed only from a recorded fact
Default autonomy	Co-pilot: plan approved once, then it runs
Unknown autonomy	Treated as Review all: draft, never send

13. The research behind Missions

Missions combine several well-studied ideas. The contribution is in how they are combined and bounded, not in any single one.

- **Reasoning and acting.** Agents that interleave reasoning with tool use (ReAct) and that learn from their own mistakes in language (Reflexion) established that a language model can act, observe, and adjust. Every Mission agent runs this loop.
- **Beliefs, desires, intentions.** Classical agent architectures separate what an agent wants from the plan it commits to and the actions it takes. A Mission's goal, plan, and agents mirror that separation. Because the plan is an explicit, inspectable artifact, it can be reviewed and revised independently of execution.
- **Levels of automation.** Human-factors research has long argued that automation is not binary, and that the right level depends on the function and the cost of error. Autopilot, Co-pilot, and Review all are three levels, applied specifically to the functions that carry risk (sending, activating, enrolling), while low-risk functions such as searching and researching run freely.
- **Mixed-initiative interaction.** Well-designed systems decide when to act and when to ask, and they make asking cheap. The supervisor's "Needs you" questions, and the standing guidance their answers become, apply this directly.
- **Timing and trigger events.** Sales research and practice have long held that the moment of contact matters as much as the message. A role change or a public statement of a problem opens a window. Missions prioritise such signals over static attributes.

14. Safety and governance

- **Tenant isolation.** Every Mission, list, lead, task, and event is scoped to its organization. The Mission endpoints refuse another organization's records.
- **An emergency stop.** One organization-level switch pauses every Mission and agent.
- **Enforced autonomy.** Sending is gated in code by the Mission's mode, as described in Section 5.
- **Human pace on LinkedIn.** LinkedIn actions run from the user's own browser through the ATAPIC Companion extension, inside daily and hourly limits, and pause automatically when LinkedIn signals a restriction.
- **Honest records.** Every action and every failure is logged with its reason, and the pipeline reports only recorded facts.
- **Preview first.** Destructive operations (clean-ups, deletions) show what they will do before they do it, and anything with history is protected.

15. Why it is valuable

- **Time returned.** The daily loop of deciding, searching, writing, and following up is the work that consumed the week. It now runs on its own clock.
- **Consistency.** A plan runs the same way every day, which is what a repeatable revenue motion needs and what a busy person cannot provide.
- **Accessibility.** A founder with no sales team and no operations staff can run a multi-channel go-to-market motion by describing the outcome they want.
- **Trust.** Because the plan is readable, the autonomy is chosen, and the numbers are facts, the owner can hand over the work without handing over control.
- **Compounding.** Missions ride the same runtime as the assistant, so every new capability the assistant gains is immediately available to every Mission.

16. Limitations and roadmap

Missions are not a replacement for judgement in a live conversation. When someone replies, the system steps back on purpose. Plan quality depends on how well the company profile describes what it sells, and a vague profile produces a vaguer plan. LinkedIn work depends on the browser extension being installed and running. Hosted always-on workspaces remove that dependency and are rolling out separately.

Near-term work includes reporting measured outcome metrics per Mission, letting the supervisor compare plays against each other and shift effort toward what is working, richer meeting booking from replies, and shared Missions across a team.

Closing

The thesis of this paper is that the right unit of automation for revenue work is a goal with a plan. A goal alone is too vague to trust. A task alone is too small to matter. A goal with a written plan, scheduled agents, a list of real people, measured targets, and an autonomy level the owner chose and the engine enforces can be trusted with the loop that decides whether a company grows. Missions are ATAPIC's implementation of that idea. The owner says what they want, reads the plan, chooses how much the system may do alone, and gets their week back.

References

1. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. ICLR, 2023.
2. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language Agents with Verbal Reinforcement Learning. NeurIPS, 2023.
3. Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language Models Can Teach Themselves to Use Tools. NeurIPS, 2023.
4. Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative Agents: Interactive Simulacra of Human Behavior. UIST, 2023.
5. Rao, A. S., and Georgeff, M. P. BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS), 1995.
6. Parasuraman, R., Sheridan, T. B., and Wickens, C. D. A Model for Types and Levels of Human Interaction with Automation. IEEE Transactions on Systems, Man, and Cybernetics, Part A, 30(3), 2000.
7. Horvitz, E. Principles of Mixed-Initiative User Interfaces. Proceedings of CHI, 1999.
8. Amershi, S., et al. Guidelines for Human-AI Interaction. Proceedings of CHI, 2019.
9. Wang, L., et al. A Survey on Large Language Model based Autonomous Agents. Frontiers of Computer Science, 18(6), 2024.