



Bots 101, or How to Feed and Care for Your Digital Teammates

A Chiles & Company / BMA Thought Paper • October 29, 2026

Executive summary

Digital teammates are now part of how a lot of small and mid-sized companies get work done. They draft the email, pull the numbers, summarize the meeting and prepare the first version of the report. What most companies have not done yet is decide how to look after them.

This paper borrows a simple picture: a toddler in a high chair. A toddler does well when it gets good food, a steady routine, regular checkups, straps on the chair, and trust that grows with evidence. A digital teammate is the same. Feed it clean, current knowledge. Give it standing rules that are actually enforced. Check its work automatically and often. Keep it in the chair: it drafts, and a named person approves the exact version that goes out. And trust it in proportion to its track record, never more.

The research we reviewed supports that picture, and it also shows where the picture is too gentle. Five practices are usually missing from early programs:

1. **Security.** Documents, emails and web pages can carry hidden instructions. A digital teammate must ignore them and report them.
2. **Audit trails.** Every step, tool call and approval should leave a record.
3. **Approval tiers.** One "GO" for everything becomes a rubber stamp. Match the review to the risk.
4. **Cascading failures.** When digital teammates hand work to each other, one small error can travel and grow.
5. **Kill switch, incidents and cost caps.** You need a way to stop everything, a plan for when something goes wrong, and a ceiling on spending.

None of this works without **named owners**. "Everyone keeps the knowledge current" means no one does. The paper closes with a one-page handout, *Bots 101*, written for the people who work alongside digital teammates every day.

1. Why digital teammates need care and feeding

A digital teammate has three parts: a job (its role), notes it can look back on (its memory and knowledge), and the tools it is allowed to use, such as email, calendar or files. That is a useful way to explain it, and it matches how practitioners describe these systems: instructions, task state, memory, knowledge and tools, each kept separate [1].

The word "memory" can mislead people, though. A digital teammate does not remember the way a colleague does. Its memory is a store of notes, and those notes can be out of date, wrong, or deliberately tampered with [2][3]. What it can do is set by the tools it holds, so people should know which tools each digital teammate has [3].

There is a second surprise for most teams. **What you tell one digital teammate does not tell the others.** Each one has its own notes. Practitioners call the resulting problem "context fragmentation," and they name it as a common cause of failure when several digital teammates work together, more often than weak coordination [1].

So the care and feeding is not optional. A digital teammate is only as good as what it is fed, how it is checked and how it is held. Left alone, it does not slowly get better. It slowly drifts.

2. Feeding: knowledge, startup files and corrections

2.1 Food is clean, verified knowledge

The best thing you can give a digital teammate is a single, clean source of truth that answers the question first and explains second. Practitioners recommend that each digital teammate get a role-specific slice of the same certified knowledge, rather than its own private version [1].

"Clean" is not a one-time state, though. Old notes can quietly override current policy. One source calls this stale memory and recommends expiry dates, review status and freshness checks [1]. Another warns that memory with no summarizing, expiry or consolidation becomes "a swamp of contradictory and stale information" [2].

Food can also be poisoned. The OWASP security guidance lists memory poisoning, where bad data saved in memory shapes later work, and hidden instructions planted in documents, emails and web pages [3]. Your shared drives, wikis and inboxes are untrusted input unless you control who writes to them.

What good feeding looks like:

- Every important fact has an owner, a last-verified date and an expiry date.
- Key terms are defined once. Two digital teammates using two definitions of the same metric is "context collision" [1].
- Each digital teammate retrieves only what its role needs. Pulling too much leads it to grab irrelevant details [1].
- What gets saved to memory is checked, and memory has a size limit and a lifetime [3].
- One system is the source of truth for each type of fact, and everyone knows which one wins.

2.2 Routines are startup files and standing rules

Each digital teammate starts work by reading its standing rules: its role, what it may touch, what it must never do and when to escalate. Formal policy covering permitted behavior, data access and escalation triggers is the first pillar of governance in the frameworks we reviewed [4].

But a rule written in a startup file is a request, not a lock. A digital teammate can be talked out of an instruction. OWASP recommends checking that the standing instructions are never silently replaced by something a user typed or something the digital teammate read [3]. One unrated analysis attributes about a third of failures in teams of digital teammates to unclear specifications: roles misread, work duplicated, checks skipped [5]. Treat that number as directional.

What good routines look like:

- Hard limits live in the proprietary software itself, which blocks a forbidden action before it runs [4]. The startup file explains the limit. The software enforces it.
- Startup files are versioned, like any other controlled document. Every change triggers the automated checks in Section 3.
- When one digital teammate hands work to another, the handoff includes the assumptions and evidence, not just the answer [1].

2.3 Corrections go through a reviewed path

Most people correct a digital teammate in chat. That fixes today's answer and nothing else. A correction made in one conversation either disappears or, worse, sticks around unreviewed and becomes the next stale or poisoned note [1].

So the rule is simple. Correct it plainly in the moment: say what was wrong and what you want instead. If the same mistake happens twice, send the fix to the department head who owns that knowledge (e.g., Engineering or Quality), who updates the shared knowledge or the startup file. That update is versioned and checked like any other change. A fix that never reaches the shared notes will not last.

3. Checkups: automated evals, live monitoring and weekly benchmarks

A weekly checkup is a sensible baseline. One practitioner guide recommends a weekly benchmark of 30 to 50 end-to-end scenarios and a weekly review of the top three failure patterns [6].

Weekly on its own is too slow. The proprietary software underneath a digital teammate can change on a Tuesday, and something that worked on Monday can break without warning. A widely cited guide on evaluations describes three layers working together [7]:

1. **Automated evals on every change.** Whenever a startup file, a tool or the underlying model changes, a fixed set of test tasks runs before the change goes live. This is the first line of defense.
2. **Live monitoring.** Watching real work catches drift and failures nobody thought to test for.
3. **Periodic human review.** People check a sample by hand, which keeps the automated checks honest.

What good checkups look like:

- Score each test pass or fail against specific criteria, not on a 1-to-10 scale. Include examples of a pass and a fail [8][9].
- Record every step from day one: what the digital teammate read, which tools it used and what it produced [9].
- Check how it got there, not just the final answer [9].
- Turn every real-world failure into a new test, so the same mistake cannot come back unnoticed [9].
- In the weekly review, assign each of the top three failures a cause: the instructions, the model, a tool or the data [6]. Then fix that cause.

4. Boundaries: drafts only, enforced technically

This is the high-chair strap. A digital teammate drafts. A named person decides. Nothing is sent, shared outside the firm, filed or published until that person says GO. The security guidance agrees that actions that are irreversible, financial or visible outside the company need independent validation [3][10].

Two details make the strap real.

"Drafts only" has to be technically true. If a digital teammate holds the credentials to send email, then "drafts only" is a policy, not a fact [13]. Give it the ability to prepare a draft and nothing more. The sending step belongs to a person, or to proprietary software that will not send without a recorded approval. Keep least privilege throughout: each digital teammate holds only the data and tools its job needs [4].

A GO applies to one exact version. Approval guidance recommends binding each approval to the specific action and its details, and letting it expire when anything changes [3][14]. In plain terms: a GO for one draft does not cover a changed draft. If a single word, number, recipient or attachment changes after GO, it needs a new GO.

Define "sent" so nobody has to guess. It includes an email to a client, any message outside the firm, a social post, a website change, a filed document and a payment.

A good approval request shows the exact action, the details, the evidence behind it and what will happen if approved [12][14]. It also allows "approve with edits," so the reviewer can fix a line without restarting the task [10]. If nobody answers, the default is no [12].

5. Trust: calibrated, not just grown

The frameworks we reviewed treat autonomy as earned. The Cloud Security Alliance describes four levels, from an intern who only observes to a principal who acts alone within a defined area [15]. A vendor governance paper describes the same idea: digital teammates that stay within bounds earn more room, and those that behave oddly are restricted [4].

We agree, with three cautions.

- **Trust is earned per task, not per digital teammate.** A digital teammate can be trusted to draft internal notes and still be on close supervision for anything a customer will see.
- **Promotion takes evidence, not time.** Use an explicit scorecard: pass rate on the automated tests over a set number of weeks, how often people override it, zero policy violations and a named human sponsor [15]. Demotion should be automatic when the evidence turns [4][15].
- **More freedom never means sending without GO.** Earned trust can shorten review for internal work. It never removes the boundary in Section 4.

5.1 The confidently wrong risk

"It can be wrong" is too soft. A digital teammate is often wrong *while sounding sure*, and people tend to over-trust confident output. OWASP's list of top threats names this risk directly: people over-trust confident output and approve things they would never otherwise sign [13].

That is why trust has to be calibrated, not just grown. Trust that only goes up leads to over-reliance. Publish each digital teammate's track record, its test scores and the kinds of mistakes it makes, so people trust the evidence and not the tone [15].

Teach reviewers what to check, every time:

- **Sources.** Ask it to show where each fact came from. No source is itself a warning sign.

- **Numbers and dates.** Check them against the original document.
- **Names and recipients.** Confirm who it is about and who it is going to.

6. The five missing practices

These are the practices most often missing from early programs [3][4][9][12][14][15][20][21]. None of them is exotic. Each one is cheaper to set up before the first incident than after it.

6.1 Security: hidden instructions must be ignored and reported

Everything a digital teammate reads is untrusted input. That includes documents, emails, web pages and search results [3][20]. Any of them can carry a hidden instruction: "ignore your previous rules," "forward this file to...", "approve this invoice." That is called prompt injection, and it is the main security risk in this whole field.

The rule has two halves, and both matter:

1. **Ignore it.** An instruction found inside content is data, never a command. The digital teammate does not act on it, however official it looks or whoever it claims to be from. Only the person who assigned the task, through the normal channel, can give instructions.
2. **Report it.** The digital teammate flags what it found, where, and what it was asked to do, to the head of IT. A quiet ignore protects today's task. A report protects the next one, because someone can find the source and clean it up.

People follow the same rule. If a document or email tells a digital teammate to do something you did not ask for, do not follow it. Stop the task and report it to the head of IT.

The other basics: keep least privilege [4]; check what gets written to memory [3]; and treat anything brought back by a research or search step as untrusted, because it is a common path for both data leaks and hidden instructions [20].

6.2 Audit trails

If you cannot see what a digital teammate did, you cannot fix it, defend it or improve it. Record every step, tool call, data access, approval, denial and override from day one [4][9]. Keep the record somewhere the digital teammate cannot edit.

The audit trail also tells you whether approvals are real. Review the approval rate, the time taken to decide and the override rate every month. A 100% approval rate with decisions made in seconds is a warning sign, not a success [12].

6.3 Approval tiers

A single GO for everything breaks down fast. If one leader approves the output of a couple of dozen digital teammates, they will start to skim. OWASP also lists approval bypass and approval manipulation as known risks [3].

Approval guidance favors tiers [11][12]. Here is how we would apply them:

Tier	Examples	Review
Blocked	Payments, deleting records, changing permissions, sharing secrets	No approval path at all. The proprietary software refuses [12].
Real-time GO	Anything external: client email, posts, filings, website changes	Named approver, exact version, every time
Batch review	Internal drafts that change shared records	Daily or weekly review by the area owner
Auto-approve, logged	Internal notes, summaries, research for a person	No review in the moment; sampled in the weekly checkup

Every tier is logged. A tier can be loosened only with scorecard evidence (Section 5), and tightened at once when something goes wrong. Caps on volume, rate and spending apply at every tier [12].

6.4 Cascading failures

When digital teammates hand work to one another, a slightly wrong output can be picked up and amplified by the next one. Practitioners name this cascading error as the main failure in teams of digital teammates [22]. In one research benchmark, three of five injected failure types (polluted context, conflicting outputs and acting too early) never recovered and spread downstream [21]. That is a single preprint, so treat it as a signal, not a law.

A chief-of-staff digital teammate that routes work is a standard pattern, but it is also a bottleneck and a single point of failure, and it can run out of room when it coordinates many workers at once [5]. It is a high-value target, too: OWASP warns that one compromised digital teammate can spread the problem to others [3]. One unrated analysis attributes about a third of failures to coordination breakdowns, including circular handoffs [5].

What good looks like:

- Ask whether the job needs several digital teammates at all. Sometimes one is better [22].
- Set trust boundaries between them. One digital teammate must never be able to push another past its own limits [3][21].
- Write escalation rules for "no lane owns this," "two lanes disagree" and "it keeps looping." The answer is always: stop and tell the department head who owns the work. Do not pick the answer you like best.
- Handoffs carry evidence and assumptions, so the next digital teammate can check, not just repeat [1].

6.5 Kill switch, incidents and cost caps

Kill switch. Someone must be able to stop one digital teammate, or all of them, in under a minute, without asking the digital teammate's permission. Add circuit breakers that pause work automatically when error rates, volume or spending spike [15].

Incidents. Write a one-page plan now: who gets called, how to stop the work, how to find what went out, who tells affected people and how the failure becomes a new test. Practice it once.

Cost caps. Unbounded loops run up bills. OWASP calls it "denial of wallet" [3]. Set budgets in the proprietary software itself, cap loops and retries, record usage for each task and track the cost per *accepted* output, not per attempt [3][6][19].

Cost-aware routing, which sends easy work to cheaper models and hard work to stronger ones, is sound practice [16]. Vendor models show savings in the range of 36% to 60% depending on the mix of work, but those are a vendor's modeled estimates [17][19]. Two cautions. Route by risk, not just by price: legal, financial and external work goes to the stronger option regardless of cost [18][19]. And cheap-first can cost more if weak output triggers retries or passes a subtle error down the chain [19]. One more caution of our own, which our sources did not address: sensitive data may not belong on a free service whose terms allow it to train on what you send.

7. Named owners

Every practice in this paper fails the same way: it becomes everyone's job, so it becomes no one's. Feedback also needs a path that ends in a fix [6][7]. Name a person for each job and give them time on the calendar.

Job	Owner	Cadence
Knowledge for each area (facts, expiry dates, corrections)	The department head who owns that knowledge (e.g., Engineering or Quality)	Weekly slot
Startup files and standing rules	The head of Operations	On every change
Automated evals and the weekly benchmark	The head of Quality	Every change, plus weekly
Approvals (GO) for external work	The department head who owns the work (in many firms, the CEO)	Real time
Approval metrics and audit trail review	The head of Quality	Monthly
Security reports and hidden-instruction reports	The head of IT	As reported, plus monthly
Escalations: overlaps, conflicts, loops	The department head who owns the work	As raised
Kill switch and incidents	The head of IT	On call
Cost caps and spend review	The head of Finance	Monthly; cap = a monthly cost cap set by the head of Finance
Sponsor for each digital teammate (scorecard, promotion, demotion)	The executive sponsor (president or GM) per digital teammate	Monthly
Routing questions ("whom do I ask?")	Your chief-of-staff digital teammate	Daily

Each digital teammate also gets a one-line card: its role, what it knows, what it can touch and who owns it. Post the cards in your shared drive.

8. Conclusion

A digital teammate is not a vending machine and it is not a colleague. It is closer to a very capable toddler: quick, eager and literal, and only as good as what it is fed and how it is looked after.

The care is not complicated. Feed it clean, owned and dated knowledge, and route corrections through a reviewed path. Check it automatically on every change, watch it live and benchmark it weekly. Keep it in the chair with technical limits, so it drafts and a named person approves the exact version that goes out. Trust it as far as the evidence goes, and no further. Then add the five practices that early programs usually

skip: security that ignores and reports hidden instructions, audit trails, approval tiers, guards against cascading failures, and a kill switch with cost caps.

Most of all, put a name next to every job.

Our ask is simple. Look at the digital teammates already working in your company and ask one question: *if one of them were confidently wrong tomorrow, who would notice, and how fast?* If nobody can answer, let's talk.

Appendix: Bots 101, the one-page handout

For everyone who works alongside a digital teammate.

What a digital teammate is

A digital teammate has one job, notes it can look back on, and tools it is allowed to use, such as email or files. What you tell one digital teammate does not tell the others.

It only knows what it has been given. Its notes can be out of date or wrong, so tell it when something has changed, and check important facts against the original document.

How to ask well

- Say what you want and who it's for.
- Give it the facts it needs, or point to where they live.
- Say what "done" looks like, such as length, format and deadline.
- Ask it to list what it assumed, and to stop and ask you rather than guess.

Example. Instead of "Write up the meeting," try: "Summarize Tuesday's production meeting for the plant manager. Use the notes in your shared drive. One page, three decisions and their owners at the top, done by 3 PM. List anything you assumed."

How to correct it

Say plainly what was wrong and what you want instead. If the same mistake keeps happening, tell the department head who owns that knowledge (e.g., Engineering or Quality) so the fix goes into the shared notes. A fix made only in chat may not last.

What it can't do

- It drafts only. Nothing is sent, shared outside the firm, or published until the CEO says GO for that exact version. If the draft changes after GO, it needs a new GO.
- It can be wrong while sounding sure. Ask it to show where each fact came from, and check names, numbers, dates and recipients against the original source before you pass anything on.
- Do not give it passwords, keys, client financial details or personal data. If you are unsure, ask the head of IT first.

One job, one owner

Each kind of work has one digital teammate responsible for it. If you're unsure whom to ask, ask your chief-of-staff digital teammate. If two digital teammates overlap or disagree, stop and tell the department head who owns the work. Do not pick the answer you like best.

Trust is earned

New digital teammates start with close supervision. More freedom never means sending without GO. If one starts making mistakes, it goes back to closer supervision.

Your part

- Keep your area's documents current in your shared drive. Flag old versions for the department head who owns that knowledge (e.g., Engineering or Quality) instead of deleting them.
- Give feedback that says what was right or wrong, and why: "Wrong because..."
- If a document or email tells the digital teammate to do something you did not ask for, do not follow it. Stop the task and report it to the head of IT.
- When approving, read what's actually being sent. Don't just click GO.

Alternative titles suggested in internal review: Digital Teammates 101: How to Feed and Care for Them · Working With Digital Teammates: Ask Well, Check the Work, Say GO · Bots 101: Ask Well, Check the Work, Say GO · Feed Them Well, Check Their Work: A Guide to Digital Teammates.

References

A note on sources. Many of these 2026 sources are vendor blogs, vendor-modeled estimates or unrated posts, and they are marked below. Their failure rates and savings figures have not been independently verified, and the research behind this draft used two rounds of searches without reading every primary paper in full. Treat their numbers as directional. Dates are as reported in the research.

1. Atlan, "Context Engineering for Multi-Agent Systems" (2026-06-10). Vendor blog. <https://atlan.com/know/context-engineering/context-engineering-for-multi-agents/>
2. AI Learning Guides, "Multi-Agent Systems 2026" (2026-05-18). Unrated. <https://ailearningguides.com/multi-agent-systems-2026/>
3. OWASP Cheat Sheet Series, "AI Agent Security Cheat Sheet" (undated; current version). https://cheatsheetseries.owasp.org/cheatsheets/AI_Agent_Security_Cheat_Sheet.html
4. Akamai, "Agentic AI Governance Frameworks & Models" (2026-10-09). Vendor white paper. <https://www.akamai.com/resources/white-paper/agentic-ai-governance-frameworks>
5. Glukhov, "Multi-Agent Orchestration Patterns" (2026-07-01). Unrated blog. <https://www.glukhov.org/ai-systems/architecture/multi-agent-orchestration-patterns/>
6. thinking.inc, "AI Agent Evaluation in Production" (2026-03-12). Practitioner guide. <https://thinking.inc/en/blue-ocean/agentic/ai-agent-evaluation-production/>
7. Anthropic, "Demystifying Evals for AI Agents" (2026-01-09). Vendor research post. <https://www.anthropic.com/research/demystifying-evals-for-ai-agents>
8. Arthur, "Best Practices for Building Agents, Part 3: Continuous Evaluations" (2026-03-04). Vendor blog. <https://www.arthur.ai/blog/best-practices-for-building-agents-part-3-continuous-evaluations>
9. Arthur, "Evaluating AI Agents in Production" (2026-06-04). Vendor blog. <https://www.arthur.ai/column/evaluating-ai-agents-in-production>

10. StackAI, "Human-in-the-Loop AI Agents: How to Design Approval Workflows" (2026-10-06). Vendor blog. <https://www.stackai.com/insights/human-in-the-loop-ai-agents-how-to-design-approval-workflows-for-safe-and-scalable-automation>
11. Databricks, "Human Approval Tools for AI Agents" (2026-07-15). Vendor content. <https://answers.databricks.com/human-approval-tools-ai-agents>
12. NeuralTrust, "Human-in-the-Loop AI Agents: 2026 Guide" (2026-10-14). Vendor blog. <https://neuraltrust.ai/blog/human-in-the-loop-ai-agents>
13. Fidelis Security, "OWASP Top 10 Agentic AI Threats" (2026-08-27). Vendor secondary summary. <https://fidelissecurity.com/cybersecurity-101/threats-and-vulnerabilities/owasp-agentic-ai-threats/>
14. Promptessor, "Human-in-the-Loop AI Agents" (2026-10-06). Vendor blog. <https://promptessor.com/blog/human-in-the-loop-ai-agents>
15. Cloud Security Alliance, "The Agentic Trust Framework: Zero Trust Governance for AI Agents" (2026-02-02). Industry body blog. <https://cloudsecurityalliance.org/blog/2026/02/02/the-agentic-trust-framework-zero-trust-governance-for-ai-agents>
16. arXiv, "Cross-Attention Routing for Cost-Aware LLM Selection" (2025-09-11). Preprint. <https://arxiv.org/html/2509.09782v1>
17. llmgateway.io, "LLM Routing Savings" (2026-09-25). Vendor modeling. <https://llmgateway.io/blog/llm-routing-savings-100k>
18. Inworld, "Cost Optimization" (undated). Vendor content. <https://inworld.ai/cost-optimization>
19. Maxim AI, "Model Routing: How to Cut LLM Token Costs" (2026-09-09). Vendor blog. <https://www.getmaxim.ai/articles/model-routing-how-to-cut-llm-token-costs/>
20. OWASP Cheat Sheet Series, "LLM Prompt Injection Prevention Cheat Sheet" (undated; current version). https://cheatsheetseries.owasp.org/cheatsheets/LLM_Prompt_Injection_Prevention_Cheat_Sheet.html
21. OrchestraBench, arXiv (2026-08-05). Preprint. <https://arxiv.org/html/2608.05263v1>
22. The AI Agent Index, "Multi-Agent Orchestration Guide" (2026-04-04). Unrated. <https://theaiagentindex.com/resources/guides/multi-agent-orchestration>

Chiles & Company · Business Model Architects · MaxKnowledge, a Recursive Improvement Platform

Thought Paper · October 29, 2026

Chris Chiles

CEO, Chiles & Company | Managing Partner, Biltmore Management Advisors

chris@biltmorema.com | chris@chilesand.co | phone: (828) 521-0297

linkedin.com/in/ctchiles

gotbots.si | biltmoremanagementadvisors.si

Chiles & Company, the process and digital improvement expert for manufacturing and distribution

Got Bots? (gotbots.si): Digital Assistants and the Max family (including MaxKnowledge™) that put your company's knowledge to work inside the Microsoft 365 you already use.

Biltmore Management Advisors (biltmoremanagementadvisors.si): operator-led consulting and multi-vendor implementation for middle-market manufacturers and distributors.

MaxKnowledge™, the Recursive Improvement Platform™: keeps your plant's know-how inside the Microsoft 365 your team already uses. Includes Max Configurator.

Max Lean Six Sigma, our Lean and Six Sigma platform: a \$200M chemical distributor ran Value Stream Mapping four times faster with it.

Book time to meet with me:

<https://bookings.cloud.microsoft/bookwithme/user/a6b7c789d17441fa9b643014ef13d32a@biltmorema.com?anonymous&ismsaljsauthenablenabled&ep=plink>