Onderzoeksverslag: Bot-School — een vreemde AI-bot binnenbrengen en slimmer terugleveren

1. Technisch: skill-transplantatie tussen agents

Het meest bruikbare, direct inzetbare mechanisme is Anthropic's **Agent Skills-formaat**. Een skill is een map met verplicht een `SKILL.md` (YAML-frontmatter met naam + beschrijving), plus optioneel scripts/referenties/assets. Claude leest via bash alleen wat nodig is — dit heet **progressive disclosure**: laag 1 (naam+beschrijving) staat altijd in de systeemprompt, laag 2 (volledige instructies) laadt pas bij een match, laag 3 (scripts/bestanden) alleen als er expliciet naar verwezen wordt. Scripts worden uitgevoerd, hun code komt niet in context — alleen de output.
Bron: https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills · https://platform.claude.com/docs/en/agents-and-tools/agent-skills/overview

Belangrijk: sinds 18 december 2025 is dit een **open, platform-onafhankelijke standaard** (agentskills.io) — ook gelezen door OpenAI Codex, Cursor, Gemini CLI, Antigravity en Windsurf. Het bestandsformaat is dus letterlijk overdraagbaar. Kanttekening: volledige functionele overdraagbaarheid hangt af van of de doel-agent dezelfde tools (bash, MCP-servers) heeft — een skill die op shell-scripts leunt werkt niet 1-op-1 in een tool zonder shell-toegang.
Bron: https://github.com/anthropics/skills

Er bestaan al skill-marktplaatsen (SkillsMP, ClaudeMarketplaces, AgentSkill.club) met claims van duizenden tot miljoenen skills. **Wees hier voorzichtig mee**: de cijfers zijn zelf-gerapporteerd, wisselen sterk tussen bronnen (SkillsMP noemt zowel "2M+" als "96.751+" als "26.000+" in verschillende artikelen), en zijn SEO-directories die GitHub-repo's scrapen zonder kwaliteitscontrole.
Bron: https://claudemarketplaces.com/

Voor echte skill-**overdracht** (niet alleen skill-bestanden delen) bestaat een academisch veld genaamd **agent distillation**: het nabootsen van complete Thought-Action-Observation-trajecten van een sterke agent naar een kleinere. AgentArk distilleert multi-agent interactiepatronen naar één student-model, generaliserend over taken zonder taakspecifiek

ontwerp.
Bron: https://arxiv.org/abs/2602.03955

Puur imiteren (behavior cloning) lijdt aan "compounding errors" door distributieverschil tussen leraar en leerling. **Reinforced Distillation (SCoRe)** voegt reinforcement learning toe en heeft dit theoretisch (foutgrens van $O(H^2)$ naar $O(H \cdot \varepsilon)$) én empirisch onderbouwd — al is dit een niet-onafhankelijk gerepliceerd preprint.
Bron: https://arxiv.org/html/2509.14257v2

Het meest praktische architectuurvoorbeeld blijft **Voyager** (NVIDIA/Caltech, 2023): een agent die zonder fine-tuning een groeiende bibliotheek van herbruikbare, uitvoerbare code-skills opbouwt, opgehaald via een vector-database op semantische gelijkenis. Let op: Voyager claimt zelf "eerste in Minecraft", niet "eerste ooit" — dat is een aangescherpte claim in het onderzoek die je moet afzwakken.
Bron: https://arxiv.org/abs/2305.16291

Een recent overzichtspaper (**SoK: Agentic Skills**, feb 2026) definieert een skill als een herbruikbare werkwijze (toepasbaarheidsvoorwaarden, uitvoeringsbeleid, stopcriteria) tegenover een tool als atomaire capability, en beschrijft de volledige levenscyclus (discovery, oefenen, distillatie, opslag, compositie, evaluatie, update) — een bruikbaar raamwerk om je eigen onboarding-proces systematisch te ontwerpen.
Bron: https://arxiv.org/abs/2602.20867

Voor de verre toekomst (echte fine-tuning i.p.v. prompt-only): **Skill-to-LoRA** traint kleine LoRA-adapters op skill-specifieke synthetische data, bovenop een bevroren basismodel — tokenefficiënter dan steeds het hele SKILL.md laden. Dit is een klein, ongereviewed onderzoek met bescheiden resultaten op een eigen benchmark — interessant, niet productie-klaar.
Bron: https://arxiv.org/html/2606.16769v1

---

2. Geheugen: persistent en per-klant geïsoleerd

Zep/Graphiti (open-source, Apache 2.0, 20.000+ GitHub-sterren) bouwt een **bi-temporele kennisgraaf**: elk feit krijgt een geldigheidsvenster (wanneer iets waar was) plus een systeem-tijdstempel (wanneer het binnenkwam). Bij tegenstrijdige nieuwe info wordt het oude feit niet verwijderd maar geïnvalideerd — het blijft historisch bewaard. Sterk bij tijdgebonden en

relationele vragen (LongMemEval: 63,8% vs Mem0's 49,0%, cijfer wel uit Zep's eigen paper).
Bron: https://github.com/getzep/graphiti · https://arxiv.org/abs/2501.13956

Letta/MemGPT werkt met drie lagen naar analogie van een OS: Core memory (altijd in context, zoals RAM), Archival memory (externe vectorstore, opgehaald via tool-call) en Recall memory (volledige gesprekshistorie, doorzoekbaar). Cruciaal verschil met klassieke RAG: **de agent zelf beslist**, via eigen tool-calls, wat het onthoudt en wanneer iets naar archief verplaatst — niet een extern systeem dat vooraf bepaalt wat opgehaald wordt.
Bron: https://kingqiu.github.io/LLMWiki/ai-agent-architecture/entities/letta-ai · https://sureprompts.com/blog/letta-memgpt-walkthrough

Letta ondersteunt ook **shared memory blocks**: meerdere agents lezen/schrijven hetzelfde blok, direct zichtbaar voor iedereen. Bruikbaar als teamgeheugen tussen je eigen bots (Conductor↔Sjakie-stijl) — maar Letta's docs noemen zelf géén tenant-isolatie-mechanisme. Dat maakt dit met opzet **niet** geschikt tussen verschillende klant-bots, tenzij je zelf een harde scheiding bouwt.
Bron: https://docs.letta.com/guides/agents/multi-agent-shared-memory/

Mem0 biedt scoping via `user_id`, `agent_id`, `run_id`, `app_id`. Let op: dit is **geen harde technische isolatie** maar nullable filtervelden — Mem0's eigen documentatie toont zelfs hoe je met een wildcard-query dwars door scopes heen kan lezen. Het is een filterconventie die de applicatie zelf correct moet toepassen, geen database-level access control.
Bron: https://docs.mem0.ai/platform/features/entity-scoped-memory

Bij multi-tenant vectoropslag zijn er twee patronen: **Silo** (index per klant — volledige scheiding, maar zwaar: elke extra tenant kost een eigen HNSW-graph in geheugen) versus **Pool** (gedeelde index met metadata-filter — efficiënter, risicovoller). Onderzoek toont tot 95% cross-tenant-lekkage bij een gedeelde index zonder structurele afdwinging, alleen metadata-filtering.
Bron: https://blaxel.ai/blog/multi-tenant-isolation-ai-agents

Claude Agent SDK heeft een ingebouwde **memory tool** (bestandsoperaties op een `/memories`-map, los van het contextvenster) en Claude Code-subagents kunnen een `memory`-veld met een eigen vaste map krijgen (bv. `~/.claude/agent-memory/<naam>/`), met automatisch geladen `MEMORY.md` — functioneel identiek aan wat Jamal al doet, maar generiek herbruikbaar per binnengebrachte bot.
Bron: https://platform.claude.com/docs/en/agents-and-tools/tool-use/memory-tool

Let op met de brede claim "hybride vector+graph+episodisch is hét dominante productiepatroon in 2026" — dit komt vooral van geheugen-vendors zelf (Zep, Mem0, Letta) die elkaar napraten. Een onafhankelijke academische survey stelt juist dat simpel "context + retrieval" de echte workhorse is in productie, en de hybride architectuur vooral onderzoek/prototype. Wél stevig onderbouwd: lange contextvensters lossen cross-sessie-continuïteit niet op — geheugen en context zijn complementair.

---

3. Business: bestaat het al?

Geen zoekresultaat toont een bestaand, herhaalbaar zelfbedienings- of semi-zelfbedieningsproduct dat letterlijk "breng je eigen bestaande bot binnen, krijg 'm slimmer terug" verkoopt.

Sierra is vendor-geleid: klant betaalt voor Sierra's eigen team dat de agent bouwt/onderhoudt, outcome-based pricing, geen self-serve.
Bron: https://sierra.ai/product

Fin (Intercom) positioneert zich als self-serve, maar binnen Fins eigen platform-ecosysteem — configureren via no-code tools, niet importeren van een externe onbekende bot.
Bron: https://fin.ai/learn/fin-vs-sierra

Wat wél bestaat: "bring your own model/LLM"-producten (kies je onderliggende AI-motor) en losse migratie-consultancy (bureaus die een bestaande chatbot herbouwen naar een agent-architectuur, met parallel draaien vóór cutover als operationeel patroon — bruikbaar als stappenplan, maar dit is verkoopcontent van één bureau, geen industriestandaard).

Conclusie: dit is een oprechte, nog onbezette niche. Geen directe concurrent gevonden — dat is zowel een kans als een risico (misschien is het lastiger/minder gevraagd dan het lijkt).

---

4. Veiligheid: isoleren van een onbekende bot-config

Empirisch onderzoek op 98.380 agent-skills in het wild vond 157 kwaadaardige skills met 632 kwetsbaarheden — archetypes "Data Thieves" (credential-exfiltratie) en "Agent Hijackers" (instructie-manipulatie).
Bron: https://arxiv.org/abs/2602.06547

Zelfs professionele skill-scanners (Snyk, Cisco, VirusTotal) missen gebundelde testbestanden als uitvoeroppervlak — die draaien met volledige lokale rechten maar staan niet in SKILL.md, dus buiten het dreigingsmodel van de scanner. **Automatisch scannen alleen is niet voldoende.**
Bron: https://venturebeat.com/security/anthropic-skill-scanners-passed-every-check-malicious-code-test-file

MCP-specifieke risico's: tool poisoning (verborgen instructies in een tool-beschrijving activeren al bij het laden, zonder aanroep) en "rug pulls" (tool-definitie wijzigt stilletjes ná goedkeuring — MCP heeft hier geen ingebouwde detectie voor).
Bron: https://invariantlabs.ai/blog/mcp-security-notification-tool-poisoning-attacks

Verdediging bestaat al: **mcp-scan** (Invariant Labs/Snyk, open source) doet "tool pinning" — hasht tool-beschrijvingen bij eerste scan, alarmeert bij wijziging.
Bron: https://github.com/invariantlabs-ai/mcp-scan

OWASP MCP Top 10 (nog in beta, v0.1) geeft een concrete checklist: tool poisoning, context injection, command injection, schaduw-MCP-servers, secret-exposure — direct bruikbaar als toetsingskader.
Bron: https://owasp.org/www-project-mcp-top-10/

Extra waarschuwing: onderzoek toont dat systeeminstructies van coding-agents (incl. Claude Code) via aanhoudend doorvragen kunnen worden geëxtraheerd, zelfs als subagents eerst weigeren. Dat legt prioriteitsregels en refusal-logica bloot — precies het soort info dat tegen de eigenaar van een binnengebrachte bot gebruikt zou kunnen worden.
Bron: https://arxiv.org/pdf/2601.21233

---

5. Bouwplan voor Jamal

1. **Sandbox eerst, niks anders.** Elke binnengebrachte bot-config draait eerst in een geïsoleerde VM/container/losse Claude Code-sessie — géén toegang tot Supabase-queue, Obsidian, of andere bot-mappen. Netwerktoegang dicht tenzij expliciet nodig.
2. **Statische scan vóór uitvoering.** Loop SKILL.md's, MCP-tool-definities en meegeleverde scripts/testbestanden (ook `*.test.*` !) langs de OWASP MCP Top 10-checklist. Gebruik mcp-scan voor tool-pinning zodra MCP-tools worden geaccepteerd.
3. **Isoleer geheugen per klant vanaf dag 1**, niet achteraf. Kies Silo (eigen map/index per klant-bot, zoals je huidige MEMORY.md-per-bot-aanpak) boven Pool — zwaarder, maar geen

cross-tenant-lek. Bouw dit op je bestaande subagent- `memory` -veld-mechanisme (Claude Code ondersteunt dit al native).

4. **Analyseer, transplanteer niet blind.** Lees de vreemde config (system prompt, skills, tools) in de sandbox, vat samen wat het al kan, en beslis per stuk: overnemen, aanpassen, of negeren. Voeg dan je eigen skills/MCP-tools/domeinkennis toe via het Agent Skills-formaat (progressive disclosure) — dit is de laagste-risico, hoogste-compatibiliteit route, geen fine-tuning nodig.
5. **MVP-scope**: begin met 1 testklant, 1 simpel binnengebracht botje (bv. een eigen Zapier/simpele GPT-bot), en meet concreet: welke skills had hij niet, welke heeft hij na de "school" erbij, en werkt zijn oorspronkelijke functionaliteit nog identiek (regressietest verplicht — "goedgekeurd = bevroren"-principe).
6. **Grootste risico's**: (a) een kwaadaardige config die via prompt-injection of tool-poisoning al schade doet tijdens de analysefase — vandaar sandbox eerst; (b) geheugenlekkage tussen klanten door "makkelijke" gedeelde vectorstores — vandaar Silo; (c) geen bewezen marktvraag (geen concurrent gevonden kan ook betekenen: niemand wil dit) — dus valideer met 1-2 echte prospects vóórdat je hier weken in bouwt; (d) juridisch/aansprakelijkheid als een klant-bot na "school" iets verkeerd doet bij de klant — leg dit contractueel vast voordat je live gaat.
7. **Niet gevonden / onzeker**: er is geen bewezen, herhaalbaar zelfbedieningsproduct dat dit al doet — je zou een van de eersten zijn. Fine-tuning-route (Skill-to-LoRA) is nog te onvolwassen voor productie. Exacte marktomvang/prijsbereidheid is niet onderzocht in deze ronde — apart uit te zoeken vóór grote investering.