Coding-Fähigkeiten im Agenten-Kontext (Hermes Agent / "hermes"-Tool-Calling)

Vorab kurz zur Einordnung, weil das für die Bewertung wichtig ist: "Hermes" taucht bei dir in zwei Bedeutungen auf, die beide relevant sind:

1. **Hermes Agent** (Nous Research) – ein Open-Source-Agenten-Harness (Nachfolger von OpenClaw) mit Skill-Learning, Memory, Tool-Gateway etc. Läuft mit praktisch jedem Provider (OpenRouter, Bedrock, vLLM, Ollama, …).
2. Der **"hermes"-Tool-Call-Parser** in vLLM/llama.cpp – das ChatML-`<tool_call>`-Format, das ursprünglich für Nous-Hermes-Modelle entwickelt wurde und heute auch von Qwen2.5/3, Llama 3.x, Mistral, DeepSeek u.a. genutzt wird. Modelle mit nativer Unterstützung dieses Formats lassen sich in einem Agenten-Loop deutlich zuverlässiger steuern (weniger "Tool-Call als Fließtext"-Fehler).

Für die Bewertung zählt daher nicht nur der reine SWE-bench-Score, sondern auch **Tool-Calling-Zuverlässigkeit, Kontextlänge, Kosten/Token und Ausführungsgeschwindigkeit** in einem Multi-Step-Agenten-Loop.

Gesamtübersicht (Stand: Juli 2026)

Modell	SWE-bench Verified (ca., je nach Harness)	Architektur / Aktive Params	Kontext	Tool-Calling-Eignung im Agenten-Loop	Einschätzung
Qwen3-Coder-480B-A35B	~55% (mini-SWE-agent)	MoE, 35B aktiv	256K (bis 1M ext.)	Nativ stark, explizit für Agentic Coding trainiert	Starkes Flaggschiff-Open-Model, aber inzwischen von kleineren Nachfolgern teils überholt

Modell	SWE-bench Verified (ca., je nach Harness)	Architektur / Aktive Params	Kontext	Tool-Calling-Eignung im Agenten-Loop	Einschätzung
Qwen3-Coder-30B-A3B (Flash)	~50–60% (Werte variieren stark je Quelle/Update)	MoE, nur 3B aktiv	256K	Sehr gut, „out-of-the-box" starke Tool-Use-Scores	Bestes Preis/Leistungs-Verhältnis für lokale/günstige Agenten, oft nahe an 480B-Modell
Qwen3-Coder-Next	>70% (SWE-Agent-Scaffold)	Hybrid (Attention+SSM), 3B aktiv/80B total	256K+	Für Agenten-Workflows und lokale Entwicklung optimiert	State-of-the-Art unter den Effizienzmodellen (Feb. 2026) – exzellent für Hermes lokal via vLLM/Ollama
Codestral 2508	~52%	Dense, ~22B	256K	Eher FIM/Autocomplete-fokussiert, weniger als reiner Agent gedacht	Gut für IDE-Vervollständigung; für vollen Agenten-Loop ist Mistrals **Devstral** (53–62%) die bessere Wahl
KAT-Coder-Pro V2	73–80% (nähert sich Claude Opus 4.6)	Proprietär, günstig (Kwaipilot)	256K	Explizit für agentische Coding-Tasks gebaut, Top-Terminal-Bench-Werte	Einer der stärksten Nicht-Reasoning-Agenten-Coder aktuell, ~12x günstiger als Claude
GPT-5.1-Codex-Mini	solide, aber unter Codex-Max	Proprietär	400K	Für Codex-Workflows gebaut, zuverlässiges Tool-Calling	Guter Kompromiss Kosten/Leistung bei OpenAI, aber klar unter GPT-5.1-Codex(-Max)
DeepSeek V3 (0324)	Basis-Niveau (älter)	MoE 671B/37B aktiv	128K	Ordentlich, aber unter V3.1/V3.2	Nur noch als Baseline relevant
DeepSeek V3.1 / Terminus	verbessert ggü. V3	MoE	128K	Function Calling im Non-Reasoning-Modus deutlich verbessert	Guter Zwischenschritt, durch V3.2 abgelöst
DeepSeek V3.2 / V3.2-Exp	**70%** (high reasoning)	MoE, mit Sparse-Attention (DSA)	128K+	Sehr stark in Agent-/Terminal-Bench-Evals	Aktuell eines der stärksten offenen Reasoning-Coding-Modelle, nahe an Gemini 3 Pro

Modell	SWE-bench Verified (ca., je nach Harness)	Architektur / Aktive Params	Kontext	Tool-Calling-Eignung im Agenten-Loop	Einschätzung
GLM-4.5-Air	~57–58%	MoE, leichtgewichtig (~12B aktiv)	128K	Höchste Tool-Calling-Erfolgsrate in der GLM-Familie (90,6%)	Sehr gute leichte Agenten-Option, günstig selbst zu hosten
MiniMax M2	Basis-Niveau, günstig	MoE	200K	Gut, textbasiert	Solide günstige Einstiegsoption
MiniMax M2.1	>67% (über mehrere Harnesses)	MoE	200K	Multilingual verbessert	Deutlicher Sprung ggü. M2
MiniMax M2.5	**75,8%** (einer der schnellsten auf 80%+)	MoE	200K	Sehr günstig (~$0,07/Task laut swebench.com)	Top Preis/Leistung, konkurriert mit Claude Opus 4.6
MiniMax M2.7	weiter verbessert	MoE	200K	Text-only, aktueller OAuth-Default in Hermes Agent	Aktuell Standard-Empfehlung für MiniMax in Hermes
MiniMax M3	Flaggschiff (Juni 2026)	MoE, multimodal	~1M	Anthropic-Messages-kompatibel, Bild/Video	Stärkster MiniMax, aber (noch) nicht überall Standard-Default
MiniMax M2-her	–	–	–	–	Keine belastbaren Quellen gefunden – vermutlich Verwechslung/Community-Fine-Tune, kein offizielles MiniMax-Release
Qwen3-235B-A22B-Instruct-2507	solide, oberhalb Kimi K2/Claude 4 Opus in AA-Index	MoE, 22B aktiv	256K	Guter Generalist mit Agentic-Fähigkeiten	Starker Allrounder, für reines Coding aber unter den Coder-Varianten
Qwen3-235B-A22B-Thinking-2507	Top-Reasoning (92,3 im AA-Reasoning-Index)	MoE, 22B aktiv	256K	Stark bei komplexer Logik/Multi-Step-Planung	Beste Wahl, wenn Agent tiefes Reasoning vor Tool-Calls braucht
Qwen3-Next-80B-A3B	wettbewerbsfähig, sehr effizient	Hybrid, nur 3B aktiv	256K, 10x Durchsatz	Für agentisches Coding & Reasoning positioniert	Extrem effizient – gute Balance aus Geschwindigkeit und Fähigkeit

Modell	SWE-bench Verified (ca., je nach Harness)	Architektur / Aktive Params	Kontext	Tool-Calling-Eignung im Agenten-Loop	Einschätzung
gpt-oss-120b	~62% (mit Tool-Zugriff)	MoE, ~5B aktiv	128K	Explizit auf Function-Calling/Agentic trainiert (OpenAI)	Bestes offenes OpenAI-Modell für Agenten, Apache-2.0
gpt-oss-20b	~37%	MoE, kleiner	128K	Ebenfalls gutes Tool-Calling, aber klar schwächer	Für sehr limitierte Hardware/leichte Tasks
GPT-5 Mini	~56–60% (je nach Scaffold)	Proprietär	groß	Solides Tool-Calling	Guter Mittelweg, deutlich günstiger als GPT-5
GPT-5 Nano	~35%	Proprietär	groß	Funktioniert, aber schwach bei komplexen Multi-Step-Tasks	Nur für einfache/hochfrequente Aufgaben sinnvoll
Hermes 4 70B	eher mittel (kein spezialisiertes Coding-Modell)	Dense, Llama-3.1-70B-Basis	128K	**Nativ auf das "hermes"-Tool-Call-Format trainiert**, hybrides Reasoning (Think-Mode)	Beste rohe *Kompatibilität* mit Hermes-Agent/vLLM-Parser, aber bei reinen SWE-bench-Coding-Metriken hinter den spezialisierten Coder-MoEs
Llama 4 Maverick	~15–21% (stark scaffold-abhängig, teils widersprüchliche Angaben)	MoE, 17B aktiv/400B total	1M (nominell)	Community-Feedback zur Coding-Qualität eher negativ	Nicht empfehlenswert als primärer Coding-Agent-Backend
Llama 4 Scout	~9% (teils höhere Einzelwerte in anderen Benchmarks, uneinheitlich)	MoE, 17B aktiv/109B total	10M (nominell)	Schwach bei komplexem Agentic Coding	Eher für lange Kontext-Aufgaben als für Coding-Agenten geeignet

(Werte schwanken teils stark je nach Scaffold/Datum der Messung – SWE-bench-Zahlen für dasselbe Modell können sich zwischen mini-SWE-agent, OpenHands, Claude Code, Droid etc. um 10–20 Prozentpunkte unterscheiden. Als Trend/Ranking sind sie aber verlässlich.)

Gruppierte Einschätzung

□□ Top-Tier Open-Weight-Agentencoder (beste Wahl für Hermes, wenn Budget/Hardware es zulässt)

- **KAT-Coder-Pro V2** und **MiniMax M2.5/M2.7** liegen aktuell an der Spitze der Nicht-Reasoning-Agentencoder (75–80% SWE-bench Verified) bei sehr niedrigen Kosten – ideal für Hermes im Cloud-Betrieb.
- **DeepSeek V3.2** ist die stärkste Reasoning-Variante unter den offenen Modellen, sehr gut in Terminal-Bench/Agenten-Setups.
- **Qwen3-Coder-Next** ist der Effizienz-Champion: fast so gut wie 10–20x größere Modelle, exzellent lokal via Ollama/vLLM in Hermes.

□□ Solide Generalisten mit guter Agentic-Fähigkeit

- **Qwen3-235B-A22B** (Instruct für schnelle Antworten, Thinking für tiefe Planung), **Qwen3-Coder-480B**, **GLM-4.5-Air** – gute Allrounder, GLM-4.5-Air besonders attraktiv als leichtgewichtige, selbst hostbare Option.

□□ Beste lokale/Edge-Optionen

- **gpt-oss-120b/20b**, **Qwen3-Coder-30B-A3B**, **Qwen3-Next-80B-A3B** – alle mit nativer Tool-Calling-Unterstützung in vLLM/llama.cpp (Parser `hermes`/`qwen`), laufen gut auf einer einzelnen High-End-GPU oder Apple-Silicon-Mac und sind explizit für Agenten-Workflows trainiert.

□□ Proprietäre Cloud-Optionen mit Preis/Leistungs-Fokus

- **GPT-5.1-Codex-Mini**, **GPT-5 Mini** – brauchbare Mittelklasse, aber teurer und nicht klar besser als die günstigsten offenen Alternativen (KAT-Coder, MiniMax).
- **GPT-5 Nano**, **Codestral 2508** – eher für einfache/hochfrequente Aufgaben (Autocomplete, Klassifikation) als für komplexe Multi-Step-Agenten-Loops.

⚠ Weniger geeignet als primärer Coding-Agent-Backend

- **Llama 4 Maverick/Scout**: bei Release (April 2025) durchwachsenes Community-Feedback zur Coding-Qualität, klar hinter spezialisierten MoE-Coding-Modellen zurück.
- **Hermes 4 70B**: hervorragend bei **Tool-Use-Protokoll-Treue** (logisch, da vom selben Team wie der Hermes-Agent-Harness trainiert) und gutem hybridem Reasoning, aber als generalistisches Dense-Modell auf Llama-3.1-70B-Basis bei reinen SWE-bench-Coding-Metriken hinter den 2025/2026er-Spezialcoder-MoEs. Sehr gute Wahl, wenn dir **Zuverlässigkeit der Tool-Calls und Steuerbarkeit** wichtiger ist als rohe Benchmark-Zahlen bei riesigen Repos.
- **MiniMax M2-her**: konnte ich nicht verifizieren – vermutlich kein offizielles Release (evtl. Verwechslung mit M2.5/M2.7 oder ein Community-Fine-Tune).

Praktische Empfehlung für den Einsatz mit Hermes Agent

Szenario	Empfehlung
Maximale Coding-Power, Cloud, Budget egal	KAT-Coder-Pro V2 oder MiniMax M2.5/M2.7, alternativ DeepSeek V3.2 (Reasoning)
Bestes Preis/Leistungs-Verhältnis	MiniMax M2.5/M2.7, Qwen3-Coder-30B-A3B
Lokal/self-hosted (eine GPU/Apple Silicon)	Qwen3-Coder-Next, gpt-oss-120b, Qwen3-Coder-30B-A3B
Höchste Tool-Call-Zuverlässigkeit/„Hermes-nativ"	Hermes 4 70B (nativer Parser), Qwen3-Familie
OpenAI-Ökosystem, moderat	GPT-5.1-Codex-Mini oder GPT-5 Mini
Vermeiden für ernsthafte Agenten-Coding-Tasks	Llama 4 Maverick/Scout, GPT-5 Nano, Codestral 2508 (besser: Devstral)

Wenn du magst, kann ich als nächsten Schritt einen konkreten Kosten/Latenz-Vergleich (Tokens/Sekunde, $/Mio Tokens) für 3–4 deiner Favoriten im Hermes-Agent-Setup zusammenstellen, oder tiefer in ein einzelnes Modellpaar (z.B. KAT-Coder-Pro V2 vs. MiniMax M2.5) einsteigen.

Revision #1
Created 2026-07-08 02:34:14 UTC by art10m
Updated 2026-07-08 02:34:39 UTC by art10m