Prosights Agentic Engineering Index: a briefing for executives

Your engineering teams now work with coding agents: AI tools that read a repository, change files and run commands for a developer. Your teams have rules for those agents. You cannot yet see which of the rules hold. This briefing shows how the Prosights Agentic Engineering Index answers that, how to read a result and what you must provide. AEI Scan is the program that gathers the evidence on your machines.

In brief

- **What you learn.** Which of your rules for agents are only written, which are checked after the agent has acted and which are enforced before it acts.
- **What you receive.** A level from 0 to 4 and a score from 0 to 100, overall and for each of 16 dimensions (a dimension is a part of the work with agents), and at most 10 actions in order.
- **What it costs you.** A machine with Python 3, and a person who answers 18 survey questions and can show an assessor the artifacts the interview asks for.
- **What stays with you.** The scan opens no source file and sends nothing unless you turn on its upload option.
- **What it does not tell you.** Whether agents make your teams faster. No study read for the index links a level to speed or to code quality (section 7).

1. What it grades, and why that matters to you

A rule for an agent usually starts as a sentence in a file the agent reads. An agent can ignore a sentence. The index tells you 3 things:

1. Which of your team's practices exist only as words, which a check catches after the agent has acted, and which a mechanism stops before the agent acts.
2. Where that leaves you in each of 16 dimensions, from the first written requirement to measurement, governance, cost, the guards around your process and how each model you use misbehaves.
3. What to change first.

The index grades evidence a second person can open: a file, a setting, a record. A team's description of itself is not enough to reach a level.

Why control, and not speed? Because the research on speed does not agree. Of the 4 studies below, 1 found developers slower with AI, 1 found a gain that was not statistically significant, and 2 found a cost in security or code quality.

Study	Published	What was done	What was found
Becker, Rush, Barnes, Rein (METR), 2025	Preprint by the authors on arXiv, July 2025	Randomized controlled trial: 16 developers and 246 issues in their own repositories, each issue assigned by coin flip to AI allowed or not allowed	Issues with AI allowed took 19% longer. Afterwards the developers estimated they had been 20% faster
Paradis and others (Google), 2024	Preprint by the authors on arXiv, October 2024	Randomized controlled trial: 96 Google software engineers, 1 set task	96 minutes with AI against 114 without. The adjusted estimate, about 21% faster, was not statistically significant ($p = 0.086$)
Perry, Srivastava, Kumar, Boneh, 2023	ACM CCS '23, a peer-reviewed conference	47 participants, mostly students, randomly given an AI assistant or none; 5 security-related tasks	The group with the assistant wrote less secure code on 4 of 5 tasks, and rated its own answers as more secure

Study	Published	What was done	What was found
He, Miller, Agarwal, Kästner, Vasilescu, 2025	MSR '26, a peer-reviewed conference	806 GitHub repositories that adopted the Cursor agent, compared with 1,380 matched repositories that did not	Lines of code added rose about 28.6%, significant only in the first 2 months. Static-analysis warnings rose 30.3% and stayed up

No study read for the index shows that a higher level brings faster delivery or better code. So the index measures whether your controls exist and can be shown. Speed is yours to measure (dimension 11 below).

2. 3 kinds of rule, 5 levels

Every practice the index looks at is 1 of 3 kinds. The kind decides how far you can rely on it.

Kind	What it is	What proves it
Written	A sentence in an instruction file or a document. Nothing checks that the agent follows it	The file
Checked	A mechanism reads the result after the agent has acted, and refuses to accept the change, the merge or the release	The check itself, which stops with an error
Enforced	A mechanism refuses the action before it happens	A hook that can deny the action, a permission rule, a sandbox setting

Evidence reaches a level. A self-rating never does. A result is written as the short name of the index, the score and the level, for example "AEI 79, Level 1", and a level is written "AEI Level 3".

Level	Name	What it means
0	Unrecorded	No evidence was found for this part of the work
1	Written	The practice is written down; nothing checks that the agent follows it
2	Checked	A check catches a miss after the agent has acted
3	Enforced	A mechanism stops the action before it happens
4	Measured	Records are kept and read, so you can see whether the controls work

Terms used in this briefing:

- **Instruction file**: a file of rules the agent reads when it starts work, for example `CLAUDE.md` or `AGENTS.md`.
- **Hook**: a small program the agent's tool runs at a set moment, for example before a command runs. On some platforms a hook can refuse the action.
- **Sandbox**: a setting that limits which files and which network addresses the agent's commands can reach.
- **Session record**: the log an agent tool keeps of a single working session.
- **Platform**: the coding agent your team uses. The index covers 7: Claude Code, Codex CLI, Cursor, Antigravity, Grok Build, VS Code with Copilot, and Xcode.

3. The 16 dimensions

Each dimension is a part of the work, and each gets its own level and score. The last column is the highest level its checks reach. A dimension at its highest level does not hold your overall level back.

#	Dimension	The question it answers	Highest level
1	Requirements and recording	Is the work written down before an agent starts, so there is something to check the result against?	3
2	Decisions	Are choices recorded, and is it written which choices a person makes and which an agent may make?	2

#	Dimension	The question it answers	Highest level
3	Expected behavior and design	Is what a person expects written before the build, and is each expectation tied to a test?	2
4	Agent briefing and model policy	Does the agent get written instructions, and is the model it runs on fixed by version?	3
5	Build	Does each agent work in its own environment, with limits on its commands and on the size of a task?	3
6	Test and evidence	Does a check prove the work passes, in place of the agent saying so?	4
7	Review and merge	Does a reviewer other than its author (a person, or a model run as a separate reviewer) review every agent change, is it written which changes need a person, and do checks run at each merge?	4
8	Release	Does a gate stand before the step that reaches your customers, and is its result kept?	4
9	Safety controls	Do a sandbox, permission rules and hooks stop a harmful action before it happens?	4
10	Memory and continuity	Is what the agent must know in its next session written down, and does each agent record what it learned where the next agent reads it?	2
11	Measurement	Do you keep and read records of what agents did, what it cost and how quality moved?	4
12	Governance	Is it written which AI tools are used on which data and who approves, and can you show which rules are enforced?	3
13	Token and cost	Is it written which model does which work, do settings hold the cost of model use down, and are usage records read?	4
14	Harness depth	How much of the agent tool's own machinery is in use: a hook before file changes, connections to outside tools (MCP servers), agents with limited tools, a working copy per agent?	3
15	Process guards	Has the team built its own guards: deletes through 1 guarded script, builds and tests under time limits, checks that keep credentials and personal identifiers out?	3
16	Model conduct	Does the team know how each model it uses misbehaves (breaks a written rule, stops early or asks for a permission it already has, reports work as done when it is not, invents a reason), and does it act on that?	4

Dimension 16 asks about the model, where the others ask about the team's process. Vendors measure these behaviors in their own models. Anthropic's system cards for Claude Opus 5.5 and for Claude Fable 5.2 and Claude Mythos 5.2 score each model on false claims that a task is complete, and its system card for Claude Sonnet 4.5 says that its evaluations do not show how often such shortcuts happen in real use. A peer-reviewed study of 1,642 runs of multi-agent systems reports how prevalent each of 14 kinds of failure was: 11.8% for an agent disobeying the task it was given, and 6.2% for an agent stopping early (Cemri and others, NeurIPS 2025).

No source read gives the rate in a team's own work. So the dimension asks whether you keep a record for each model, whether someone other than the agent checks what it reports, and whether you count.

4. How a rating is produced

97 checks produce the rating. Each has a level, the evidence it reads and a rule for passing. Evidence reaches a check in 1 of 3 ways:

- **Scan (35 checks).** AEI Scan runs on a developer's machine, reads the files and settings where the platform's vendor documents them, and decides without a person.
- **Survey (18 checks).** Someone on your team answers a question, for example whether a reviewer other than its author reviews every agent change before it merges.
- **Interview (44 checks).** An assessor asks to be shown the thing itself, for example the release gate or the record of the last 2 releases, and grades what is shown.

How much the scan can read depends on your platform, because vendors document different amounts. Where a vendor documents no location, the tool does not guess. The check becomes a question for a person.

Platform	Scan checks the tool reads	Scan checks a person grades
Claude Code	35	0
Codex CLI	22	13
Cursor	21	14
VS Code with Copilot	19	16
Antigravity	10	25
Grok Build	3	32
Xcode	3	32

Evidence outranks description, in 4 ways:

1. For each scan check the survey also asks what you believe is in place. Where the scan can read the evidence, the scan result stands and your answer never changes it.
2. If you said yes and the scan found nothing, the check fails and the report lists it under "Said but not found". The number of such checks is your **claim gap**.
3. If you said no and the scan found it, the check passes and the report lists it under "In place but not known to the team".
4. A "yes" with nothing shown counts in full only for a level 1 check. At level 2 or higher it is marked "Claimed": it earns half credit in the score and does not count toward reaching a level.

5. What the report contains, and how to read it

You receive 1 report in 2 formats (a page that opens in a browser with no connection, and text) and the result as a data file. The report has 3 parts. Page 1 is written for you.

Part	What it holds
Page 1: Summary	Your result in a line, then the overall level, the overall score, the coverage and the claim gap, and what the level means in a sentence
Page 1: What to do next	The first 5 actions, each with the dimension, a sentence that says what to put in place, and where it goes or who confirms it
Page 1: What the result rests on	How many checks were graded by the scan, by your answers and by what a person was shown
The 16 dimensions	A single table: a level, the highest level possible, a score and the checks graded for each
Appendix: all next actions	At most 10 actions in order, each as the sentence page 1 uses. Then every other open scan check, with the file, the key or the hook event that the vendor of your coding agent documents
Appendix: what passed, by kind of control	Every practice in place, each stated as a sentence and sorted into the 3 kinds, so you see which rules depend on the agent choosing to follow them

Part	What it holds
Appendix: where answers and scan disagree	"Said but not found": the answer was yes and the scan found nothing. "In place but not known to the team": the scan passed the check and the answer denied it
Appendix: model conduct	Your own counts for each model, when you give them
Appendix: what was read, and every check	Every location the scan looked at, and every check with its result, what the check reads and what was found, so the result can be repeated
Appendix: the levels, and what the result does not show	The limits in section 7 of this briefing

In the browser, each part of the appendix is closed until you select its title, and every part prints open.

The 4 numbers on the first page:

- **Level (0 to 4).** The kind of control your work with agents rests on. To hold a level, at least 75% of the rated dimensions must be at that level or at their own highest level, and safety controls (dimension 9) must be there too, up to level 3. Level 4 also needs measurement (dimension 11) at level 4.
- **Score (0 to 100).** The share of the weighted evidence that was found, where a check's weight is its level. It is not a percentile and it compares you with no one.
- **Coverage.** How many of the 97 checks could be graded. A check that could not be established is left out of the score and lowers coverage.
- **Claim gap.** The number of checks where the answer was yes and the scan found nothing.

2 words qualify a result. A dimension is **not rated** when fewer than half of its checks could be graded. The overall level is **provisional** when at least 1 dimension is not rated.

An example: our own result. Prosights ran the tool on the repository of Kyberna, its own product, on 2026-11-16. Most of the checks were drawn from that product's build process, so read it as an example and not as validation (section 7, limit 6). Its first line reads "AEI 79, Level 1":

Level	Score	Coverage	Claim gap
1, Written	78.7	95 of 97 checks (98%)	0

#	Dimension	Level	Score	Checks graded
1	Requirements and recording	3, Enforced	100	4 of 4
2	Decisions	2, Checked	100	3 of 3
3	Expected behavior and design	2, Checked	100	2 of 2
4	Agent briefing and model policy	3, Enforced	100	6 of 6
5	Build	3, Enforced	100	4 of 4
6	Test and evidence	2, Checked	56	6 of 6
7	Review and merge	1, Written	47	7 of 7
8	Release	4, Measured	100	3 of 3
9	Safety controls	4, Measured	86	8 of 8
10	Memory and continuity	1, Written	78	6 of 6
11	Measurement	1, Written	53	5 of 5
12	Governance	1, Written	25	2 of 4
13	Token and cost	3, Enforced	88	12 of 12
14	Harness depth	3, Enforced	100	7 of 7

#	Dimension	Level	Score	Checks graded
15	Process guards	3, Enforced	100	11 of 11
16	Model conduct	1, Written	27	9 of 9

How to read it. All 16 dimensions were rated, so the level is not provisional. 2 checks were not graded: 12.1 and 12.4.

A high score and a low level can therefore sit side by side.

6. What you provide, and what stays on your machine

You provide:

- A machine that holds the repository and has Python 3. Nothing is installed. The tool was written and tested on Python 3.9.6 on macOS. Its tests give it the folders, environment variables and file formats that each vendor documents for Linux and Windows. On 2026-11-14 the test suite passed on GitHub's Linux and Windows machines; 3 tests that need a POSIX file system are skipped on Windows. No person has run the program on a Linux or Windows workstation with real agent files.
- A person who answers 3 opening questions, 18 survey questions and 35 questions on what the team believes is in place, and names the folder of your decision records if you keep them. Any question can be skipped.
- For the interview, a person who can show the 44 artifacts the interview checks ask for, and the scan checks the tool cannot read on your platform.
- Your choice on 2 options that are off unless you turn them on: counting the agent's session records, and adding the machine's name, the user's name and the repository's full path to the results.

What stays on your machine:

- The scan sends nothing unless you turn on its upload option. It writes 3 files into a folder you name, and you decide whether to send them to anyone.
- With the upload option, the tool sends the results as data, 1 file for each coding agent assessed, to the results site. It first shows what it will send and the address. You need no account to upload; you register on the site to see the result, and after that only your account can read or delete it.
- It reads whether the documented files exist, the settings the checks name, the names of hook events (never a hook's command), the text of the agent's instruction files (searched for headings, command lines and a few words, and counted in lines, with only the result kept), and how many files the agent's memory folder holds.
- It never opens source code, memory files or session records, and never copies a setting's value into its output. With your permission it counts session records and reads their dates.
- With the benchmark option, the tool reads the figures the results site publishes and sends no part of your result.
- On the results site, a result is counted in the benchmark only after you tick a box that is empty until you tick it. The site then uses the scores, the coding agent and 4 answers you may give (industry, how many people write or review software, who owns the organization and, for a venture-capital-backed one, the stage of its latest financing round), and publishes only medians and quartiles of groups of at least 11 results. You can turn it off again.
- Before anything is read, a trial mode lists every location the scan would look at, and stops.
- The developer briefing names the test that backs each of these statements.

7. What the result is, and what it is not

A result states which controls on agent work exist and can be shown, for 1 coding agent, in 1 repository, on the day of the run. A repository that holds files of 2 coding agents gets 2 results from the same run. It has 9 limits:

1. **No proven link to outcomes.** No study read shows that a level predicts speed or quality.
2. **Presence, not effectiveness.** A registered hook passes whether it guards 1 command or 100. The tool does not test that a control works.
3. **Platforms.** On Grok Build and Xcode the scan reads only the 3 checks that concern the repository itself; every other scan check is a question for a person.

4. **Word matching.** 3 checks look for headings, command lines and a few words in an instruction file. A file that covers the same ground in other words fails them.
5. **Small counts.** Several dimensions have 1 or 2 checks at a level, so 1 result can decide that level.
6. **Circularity.** 78 of the 97 checks come from the process used to build Kyberna, and for 32 of them that process is the only source. A result for Kyberna itself tests the arithmetic and is not independent validation.
7. **Judgment.** 44 checks rest on a person judging what is shown, and 2 assessors may judge it differently.
8. **Self-report.** Whether each change is reviewed before it merges is a survey answer; the scan cannot confirm it.
9. **Dimensions 13 to 16 rest on fewer sources.** All 11 checks of dimension 15 come from the process of Kyberna alone, and 5 scan checks in dimensions 13 to 15 are read on Claude Code only. The 9 checks of dimension 16 come from the corrections 1 person made in 2 work sessions, from vendors' reports on their own models and from 1 study.

How the index stands beside other published work. On 2026-11-13, 19 published maturity models, assessments and scanners were read on their owners' pages. Among those 19, no page listed the agent's hooks, permission rules or sandbox settings among its inputs, and none combined a survey with a scan. Other entries do things this index does not: DORA's model rests on survey research with nearly 5,000 respondents, and this index has no survey population; Larridin's checklist has 84 checks of code hygiene that this index does not repeat; Factory's product offers automated fixes; VirtusLab publishes 240 practice guides. The 19 are the entries that were read, not the whole field.

8. After the rating

- **Act on page 1.** The report lists at most 10 changes in order: failed or claimed safety checks first, then in each other dimension the checks at the lowest level that holds it back. Page 1 shows the first 5. Each action is a sentence that says what to put in place. For a scan check it names where that goes on your platform; for a survey or interview check it says who confirms it.
- **Run the scan again.** A second run confirms a change to a scan check. Your answers are kept in a file, so a later run reuses them.
- **Compare.** The report can show your result beside a scan of 204 public open-source repositories and a new installation of your coding agent, which scores 0. It also shows the results other members chose to share, once a group of them holds 11 results; below that it shows the count. Each figure states how many results stand behind it. The shared results are not a sample of all teams.
- **Measure speed yourself.** Dimension 11 asks for the records: merge and revert rates of agent changes against your own baseline, usage or cost per agent or task, and a quality trend over 3 months.

9. For an investor: several companies, and one company before a deal

This section covers 2 uses. An investor in several companies can keep a portfolio. An investor who considers a deal can ask the company for a pre-deal assessment.

A member of the results site can keep a portfolio: a list of companies whose results that member reads side by side.

- **You invite, and the company decides.** You make a portfolio and an invitation link for each company, and you send the link yourself. A person at the company signs in, reads what will be shared and with whom, and agrees or declines. Nothing of a company's result reaches you before that. You cannot add a result for a company.
- **What you read.** For each company that agreed: its latest result with the overall score, the level, the 16 dimension scores, the coverage, the claim gap and the date, and its earlier runs. Where a group of shared results has at least 11 results, you read where the company's score stands in that group.
- **What you read for the portfolio.** In each dimension the lowest, the median and the highest score of your companies, and the 5 dimensions that are among the 3 lowest scores of the most companies.
- **What you do not read.** The checks, the reasons, the file names and the company's own counts for each model. A company can share those as a second, separate choice. You never read the email address of the company's account.
- **Withdrawal.** A company can withdraw at any time, and the portfolio then no longer shows its results. What you saved or printed before stays with you.

- **Exports.** You can save the portfolio table as a CSV file and print a summary.
- **The benchmark by ownership.** A member may say whether the organization is venture-capital-backed, private-equity-backed, publicly listed or otherwise privately held, and for a venture-capital-backed one the stage of its latest financing round. The benchmark shows the median of such a group once it has 11 results.

What the portfolio does not show. Each company runs the scan itself and gives its own answers; the results site recomputes the scores and does not verify the answers. The site does not check that the account that agreed belongs to the company you named. The limits of section 7 hold for every row. No study read for this index relates a score to a company's value, and this briefing claims no such relation.

Before a deal, you can ask a company for its results under a code name.

- **You ask, and the company decides.** You make a request, choose what you ask for (the scores, or the scores with the checks, the reasons and the file names) and set a last day. You send the request link yourself. The company runs the scan on its own machines, for as many repositories as it names, and uploads the results. You read nothing until a person at the company signs in and agrees.
- **The scope is the company's statement.** The company states in how many repositories its teams work with coding agents, which of them are in the results and which are left out. The report prints that statement beside the number of repositories uploaded and says that nobody checked it.
- **Every figure says how well it is evidenced.** Each graded check has 1 of 4 grades. Declared: the company said so. Scanned: the scan read it in a run nobody outside the company watched. Witnessed: the scan read it in a run an assessor watched. Verified by assessor: an assessor was shown the artifact and graded the check. The report gives the share of the score at each grade and the weakest grade.
- **The assessor.** A person whom Prosights has approved as an assessor, and whom you name for the request, can watch a run and grade the checks that need an artifact shown. A watched run shows the repository at that moment, and a team that is watched may behave differently.
- **The report.** The first page is written for an investment committee: the result, the level, the evidence, the coverage, the scope declared against the scope uploaded, and the 5 most serious findings. Then come the 16 dimensions with the spread across repositories, the findings by severity with the rule printed in full, the questions for management, and a remediation plan ordered by level and by kind of work. The rest follows as an appendix. The plan gives no hours and no cost, because none was measured.
- **Access ends.** You name each further reader. The company sees every name and every time the report is opened. On the last day, or sooner if the company withdraws or you end the request, the site deletes the results. If the company agrees, its results move into your portfolio as its first dated results.

The benchmarks page of the website links to a sample pre-deal report made from 3 made-up repositories.

What the pre-deal report does not show. A run that nobody outside the company watched can be staged. The severity rule is the site's own: no source read for it gives a scale for the findings of a technology due diligence. The site cannot tell the company from another person who holds the request link. The limits of section 7 hold for every figure. The report makes no claim about value, returns or price.

Sources

01, 03, 04, 05, 06, 10 to 12, 14, 15 and 17 are the files with those numbers in the `research` folder, except 06, which is `framework/06-framework.md`. "Tool" is the `tool` folder.

Section	Source
1	06 section 1 (the 3 things, evidence over self-description, no proven link). The 4 studies: 03 section 3, with Becker and others at https://arxiv.org/abs/2507.09089, Paradis and others at https://arxiv.org/abs/2410.12944, Perry and others at https://arxiv.org/abs/2211.03622, He and others at https://arxiv.org/abs/2511.04427
2	06 section 2 (kinds and levels); level meanings from tool `assessment/data/catalog.json`; 05 section 6 and 06 section 4.1 (the 7 platforms, hooks, session records); 03 section 1, Anthropic, https://www.anthropic.com/engineering/claude-code-sandboxing (what a sandbox limits)

Section	Source
3 and 4	06 sections 3, 4, 4.3 and 5.1 to 5.3; the questions in section 3 are drawn from the checks in 06 section 4.2, and for dimensions 13 to 15 from 10 section 3, 11 section 3 and 12 section 3; dimension 16 and the paragraph under the table: 15 parts B and C, with Anthropic's system cards at https://anthropic.com/claude-opus-5-5-system-card (pages 36 and 111) and https://www.anthropic.com/document/claude-fable-5-2-mythos-5-2-system-card (page 114), the system card for Claude Sonnet 4.5 (section 6), and Cemri and others at https://arxiv.org/abs/2503.13657; the table of platforms is generated from tool `assessment/data/platforms.json` by `briefings/build.py`
5 and 6	06 sections 4, 5.1, 5.3, 5.4, 6 and 7; tool `README.md`; the example is generated from tool `runs/kyberna/results.json` (`assessments/kyberna/paper-test.md`, "The tool's run")
7	06 section 8 (the 9 limits); 04 sections 4 and 5, gaps 1 and 5, and 17 (the 19 entries). DORA: 04 row 4, https://dora.dev/ai/capabilities-model/report. Larridin: 04 row 13, https://larridin.com/developer-intelligence/readiness-checklist/. Factory: 04 row 12, https://docs.factory.com/web/agent-readiness/overview. VirtusLab: 04 row 10, https://visdom-maturity-matrix.virtuslab.com/
8	06 section 6 (next actions); tool `README.md` (answers kept in a file); 06 section 4.2, checks 11.2 to 11.4
9	14 section 10 (sources, design choices and limits of the ownership answers and the portfolio); `portal/src/portfolio.js` and `portal/test/portfolio.test.js` (what the owner reads at each share level). The pre-deal assessment: 16 (sources, the 3 rules, what each person does, limits), with ISA 580 paragraph 4 and ISA 500 paragraphs A17 and A31 as cited there; `portal/src/predeal.js` and `portal/test/predeal.test.js`