

Agent Skills Panel: Summary and Key Themes

CASS Software Ecosystem Working Group · Oct 5, 2026

At a glance

The panel's shared conclusion: agent skills are useful, but they are not the hard part. Five library and tool leads agreed that agents already write, test, and document scientific code well. The bottleneck has moved to code review, CI capacity, and trust.

The panel met Monday, October 5, 2026, 12–1 pm ET, as the first webinar in the working group's fall–spring series. Mike Heroux moderated. Each panelist gave a short introduction against five prompts, followed by about ten minutes of open discussion. Jim Willenbring and Carol Woodward also contributed from the audience.

Panelist	Project · Institution	Main message
Ross Bartlett	Trilinos · Sandia	Skills are a context-management strategy, "nothing magical." Testing and trust are the hard problems.
Ann Almgren	AMReX · LBNL	Her agent learned how she works without a single hand-written skills file. The pace of change cannot be handled incrementally.
Cody Balos	SUNDIALS · LLNL	Skills are a software product that encodes expertise. Code review is now the bottleneck.
Satish Balay	PETSc · Argonne	One review skill changed the merge-request workflow and overloaded CI in the process.
Berk Geveci	VTK / ParaView · Kitware	Writing code is largely solved. Effort now goes to domain knowledge bases and applications for non-developers.

What agent skills are

An agent skill is a way to give a model extra context only when it needs it. Ross Bartlett opened with this primer so the rest of the panel had a common definition.

A skill is a named Markdown file. Its header holds a name and a short description. Its body holds the context and tool definitions for one kind of task. Skills usually live in a project directory such as `.agents`, or in a global location. The format began at Anthropic and is now a widely adopted standard.

How a skill gets used:

1. At startup, the agent harness puts only each skill's name and short description into the context window.
2. The user gives a prompt. The model decides a skill looks relevant and asks for it.
3. The harness injects that skill's full Markdown file and calls the model again.
4. The model requests a tool described in the skill. The harness runs it in the user's shell and returns the result.

The payoff is a small context window: faster and cheaper runs, a model that stays sharp, and fewer compactions. Ross contrasted this with loading everything up front, which buries the relevant 10% in a haystack and makes the model "dumber" and more expensive.

Ross and others named four limits:

- **Catalog bloat.** Jim Willenbring found that even dozens of short skill descriptions add up to significant context. Ross suggested a handful of category skills that disclose finer-grained ones, for perhaps one-tenth the context. Cody Balos noted that plugins bundle skills so whole groups can be switched on or off.
- **Triggering.** Models often ignore a skill even when its description clearly applies. Users may need to name the skill or force-load it.
- **Testing.** A skill that works with one harness and model may fail with another. Nothing agentic can be trusted without targeted evaluation suites, and building them is hard.
- **Supply chain.** A downloaded skill is text injected into your context, which makes it a prompt-injection vector. Ross urged treating skills like system software: version control, source verification, checksums. "We're not really doing that right now."

Responses by prompt

Panelists received five prompts and were told they need not answer every one. Their remarks are grouped here by prompt, in speaking order.

1. What have you tried?

- **Ross Bartlett (Trilinos).** Coding agents and skills for automated refactoring and test generation. Much of his last six months went to evaluation suites for agentic workflows.
- **Ann Almgren (AMReX).** Agents across the whole cycle: design, coding, debugging, tests, PR review and documentation. "Somebody's agent writes it, my agent reviews it," and after a couple of iterations, the result is usually good. The agent is more diligent than she is about tests and coverage. It also works as a thinking partner: it caught edge cases, ran an unrequested convergence study, and pointed to another code using the same algorithm variant.
- **Cody Balos (SUNDIALS).** Two families of skills. For developers: building the code, creating tests, updating answer files, addressing PR review comments, and a code-review skill distilled by an agent from 700+ past PR reviews. For users: an "A to Z" guide skill that goes from a problem

description to an auto-tuned SUNDIALS implementation, and a skill for writing custom extensions.

- **Satish Balay (PETSc).** One review skill with three entry points: review a branch, review a merge request, or review a merge request and post suggested fixes that can be applied directly. The skill tells the agent what to focus on and what to ignore, so it flags only what reviewers care about. He also uses agents to rewrite messy commit history into clean commits and to find old bugs and code/documentation mismatches. Teammates run an experimental documentation MCP server with limited use so far.
- **Berk Geveci (VTK / ParaView).** Little reliance on skills for coding. Thousands of public VTK and ParaView scripts mean agents already produce working code, though not always the filter an expert would pick. Kitware built a VTK knowledge base for agents and AI-enabled applications that users steer in plain language. Colleague Patrick O'Leary built domain-specific applications for nuclear energy and groundwater in about three hours.

2. Where are you headed?

- **Ann Almgren.** In the short term, paying down technical debt: latent bugs in 10- to 20-year-old code, unguarded edge cases, long-deferred features and test coverage. She called this valuable but "mundane in some sense."
- **Cody Balos.** A shared skills marketplace at Livermore, across computing zones and in open source, for skills that span libraries such as AMReX, SUNDIALS and hypre. Beyond skills, adapting an agent-heavy development workflow to scientific and numerical software.
- **Satish Balay.** Most effort stays on the merge-request process. A work in progress is coordinating multiple agents for application development, each with a bounded task that fits its context window.
- **Berk Geveci.** Moving up a level, from "compute the Q-criterion and extract an isosurface" to "I have a fluid problem and care about these features, give me an application." The means are curated knowledge bases built from papers and examples. The intended user is an engineer who thinks in domain terms, not VTK filters.

3. Where is the biggest opportunity?

- **Ann Almgren.** Freedom from tedium. "If you can tell me what you want something to look like, I can just do it on Monday." She cited a recent blog post comparing mathematicians to climbers passed by an airplane, and asked what we can see from the summit when we arrive rested.
- **Cody Balos.** Skills as a new way to encode expertise, alongside documentation, papers and code. They are not going away: vendors ship skills inside their own harnesses. The larger opportunity is redesigning workflow for the new velocity of code.
- **Berk Geveci.** Full interactive applications generated on request for a specific analysis, eventually extending to running simulations and managing data. Curated examples also raise code quality, because models cannot tell good VTK usage from bad in their training data.

4. What is the biggest open question or concern?

- **Ross Bartlett.** Testing and trust, "the two huge things."
- **Ann Almgren.** Three concerns. First, dependence: "we're in the cocaine phase," and access will be rationed or cut, as Genesis has already signaled. Second, we are close to taking ourselves out

of the loop, with unknown consequences. Third, the value of library code changes if a user can re-implement one algorithm in an afternoon instead of carrying a dependency. She also worries that many colleagues are not yet reeling from the pace.

- **Cody Balos.** How to test, evaluate, benchmark, version, and share skills. And how to reconcile agent-reviewed code with the lab expectation that a human reviews every line.
- **Satish Balay.** Knowing where effort pays off when tools improve so fast that skill files may become unnecessary. Each suggested fix needs a retest, and CI is overwhelmed.
- **Berk Geveci.** Loss of expertise. A PhD-candidate intern's agents produced strong in-situ analysis work with MFEM, delivered as a 100-page report his manager could not use. Skills every earlier student learned may vanish, and practice may freeze at today's state of the art.

5. How can CASS help?

- **Berk Geveci.** Encourage the building of curated knowledge bases, and keep convening discussions like this one.
- **Ann Almgren.** Work out how projects can set up layered PR review, with automated gates before a PR reaches the next level of review.

Open discussion: review, volume and throttling

The open discussion centered on one problem: agents generate pull requests faster than projects can review and test them. Carol Woodward framed it with two questions.

What has actually helped ensure the quality and correctness of generated PRs?

- **Ross Bartlett.** The weakest agent output is documentation for humans. It has too little detail or far too much, and sometimes just echoes his prompt. He spends about 80% of review time fixing it. Tests and code are decent.
- **Cody Balos.** Use a hierarchy of review based on risk. Agents review and fix low-risk changes, and a classifier routes high-risk changes to humans. In SUNDIALS, any change to an answer file would count as high risk. The classifier itself needs a quick way to be checked.
- **Ann Almgren.** The issue is volume, not authorship. People have always submitted flawed PRs, and agents are held to a higher standard than humans ever were. Having her agent review every incoming PR costs her tokens and will not scale. She wants a filter a PR must pass before reaching the next level of review.
- **Ann Almgren, on variation.** Different agents find different bugs, like colleagues with different perspectives. Full reproducibility should not be the goal: "They're not the computers, they're the colleagues."

Should projects throttle incoming PRs, as some journals and conferences are considering for submissions?

Carol put it this way: we can do anything now, but do we need to do everything?

- **Mike Heroux.** A thought experiment: decline large PRs and ask for plain-English specifications instead. The maintainer's own agent then generates the code, so its behavior is easier to understand.

- **Berk Geveci.** CI is already throttling. Some pipelines on the Frank systems take days, and a human still looks at every PR eventually. Nobody knows what happens to the growing backlog.
- **Cody Balos.** Google's code review guidance says a team ships only as fast as it can review together. The job now is raising review pace.
- **Jim Willenbring.** Build layers of filtering that reject what is egregiously wrong, or comment back so the PR can be revised. The aim is a better signal-to-noise ratio for human reviewers, not an AI verdict on which PRs are acceptable. The filters can learn over time from human reviews.

Key themes

Seven themes ran across the introductions and the discussion.

1. **Skills are context engineering, not magic.** Ross's framing held the whole session. A skill is text loaded on demand, and the hard work of testing and verifying still has to be done.
2. **Skills matter less for yourself and more for others.** Ann, Satish and Berk all see the need for hand-written guidance shrinking as tools improve. Skills earn their keep when they transfer expertise: Cody's user-facing skills, Satish's review priorities, Berk's curated examples.
3. **The bottleneck has moved from writing code to reviewing and testing it.** Cody, Satish, Berk, and Carol each described PR volume overwhelming reviewers and CI. No one reported having solved it.
4. **Layered, risk-based review is the emerging answer.** Cody, Ann and Jim converged on automated filters and agent review for low-risk changes, with human attention reserved for what matters most.
5. **The trust infrastructure for skills does not exist yet.** Evaluation suites, versioning, provenance, and protection against prompt injection all came up as unmet needs, especially for sharing skills across projects.
6. **Mature codes are getting better.** Agents are finding long-standing bugs, filling test gaps, reconciling documentation with code, and cleaning commit history. Ann called it paying off technical debt.
7. **The value of what we build is shifting.** Re-implementing an algorithm instead of adding a dependency, generating an application in hours, and accepting specifications instead of PRs all change what a library or a contribution is worth.

Panelists also named risks on the human side: expertise that is never learned (Berk), people leaving the loop, and dependence on AI access that may be rationed (Ann).

Where panelists differed

- **Are we leading?** Ann argued the labs are "riding the wave," not at its front, and waste time acting otherwise. Cody agreed on position but argued skills remain worth building.
- **Documentation.** Ann said documenting "has never been easier." Ross called agent-written documentation for humans "awful" and his main review burden.
- **Consistency.** Ross stressed that skills behave differently across harnesses and models and must be tested. Ann welcomed variation among agents as a source of different insights.

Open questions for the working group

These questions were raised and left open. They are candidates for the November working meeting.

- How does a project set up layered PR review in practice: which automated gates, which risk rules, and who checks the classifier?
- Where should skills that span several libraries live, and how would a CASS effort relate to marketplaces such as the one Livermore is building?
- What would shared practice for testing, versioning, and verifying the source of skills look like?
- Should projects limit or reshape contributions, for example, smaller PRs or specifications in place of code?
- What does "a human reviews every line" mean when agents write and review most code?
- How does a library's value change when users can re-implement a single algorithm in an afternoon?
- How should projects plan for AI access that becomes rationed or more expensive?
- How do early-career researchers build the skills that agents now cover for them?

Resources and next meetings

The conversation continues at Teatime on Oct 15, 2026, and at the working meeting on Nov 2, 2026.

Series	When	Next
Teatime	3rd Thursday monthly, 4–4:30 pm ET	Oct 15, 2026
Working meeting	1st Monday of odd months, 12–1 pm ET	Nov 2, 2026, with this summary on the agenda
Webinar	1st Monday of even months, 12–1 pm ET	December

To get invitations, join the [working group list](#) and the [Teatime list](#) on groups.io. The group's Slack channel is #wg-ecosystem on the CASS workspace.

Panelists pointed to these resources. Links for several are on their slides.

- The Agent Skills standard, which began at Anthropic (Ross Bartlett)
- A study of prompt-injection attacks delivered through skills (Ross Bartlett)
- A recent blog post on AI and mathematics, with the climbers-and-airplane analogy (Ann Almgren)
- An article on OpenAI's agent-driven software development workflow (Cody Balos)
- Google's code review guide (Cody Balos)
- A public benchmark site for agents on visualization tasks, which Berk Geveci believes comes from Livermore