Charting Uncertain Waters

A Socio-Technical Roadmap for Sustaining Open Source Communities in the Age of GenAI

Zixuan (Steve) Feng
OSU → VCU

Reed Milewicz
Sandia National Labs

Emerson Murphy-Hill
Microsoft

Tyler Menezes [1]
CodeDay

Alexander Serebrenik,
Eindhoven TU

Igor Steinmacher
Northern Arizona University

Anita Sarma
Oregon State University

[1] Presenter

Download This Presentation

doi.org/10.69924/20640.1608



This material is based upon work supported by the National Science Foundation under Grant Numbers 2303612, 2247929, 2303042, 2347311, and 2303043.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

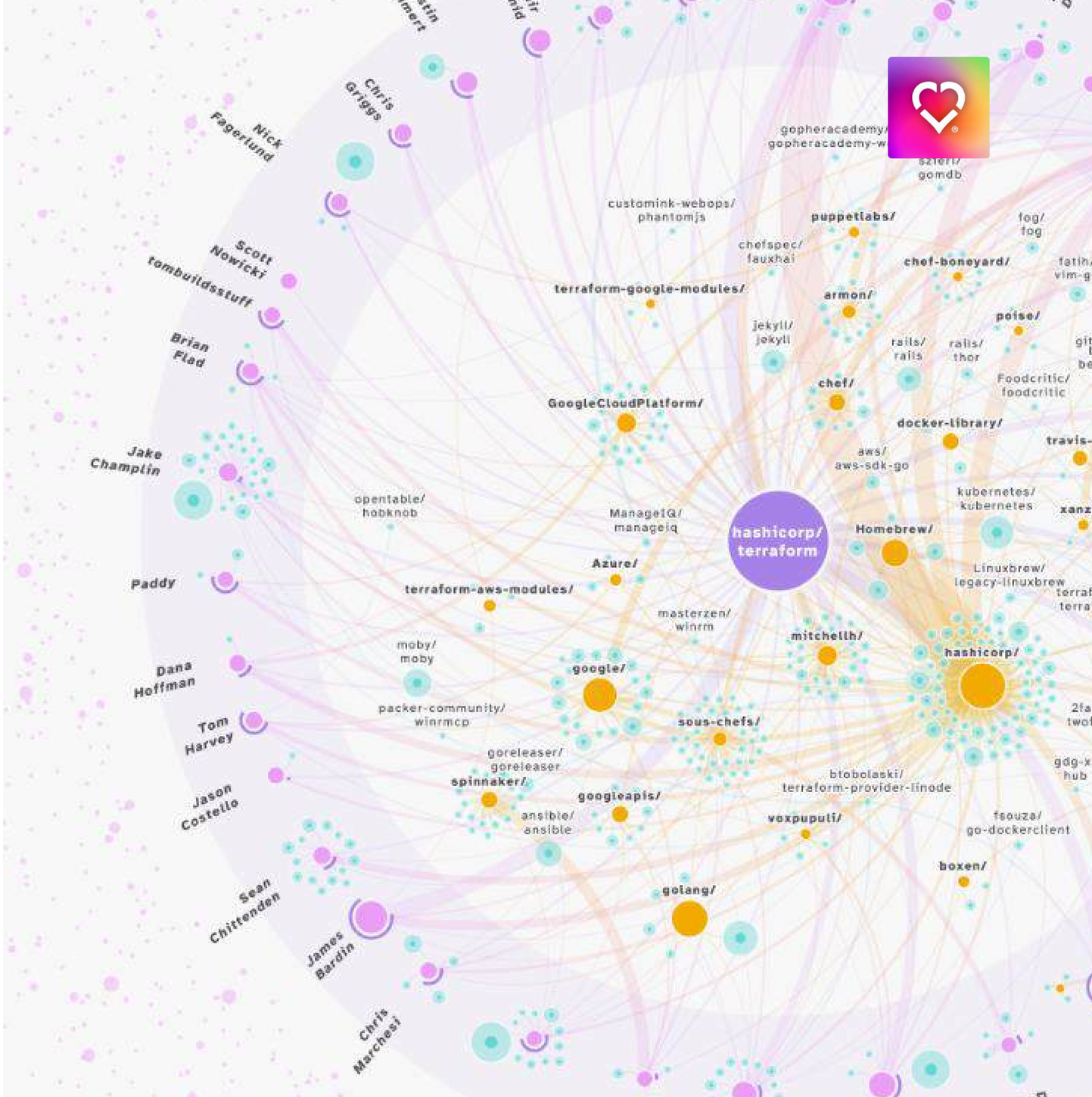
Background & Motivation

BACKGROUND & MOTIVATION

OSS powers the modern world, producing $8.8T in value.

But this value is inseparable from the people: volunteers, maintainers, mentors, and the norms of altruism, freedom, and collaboration.

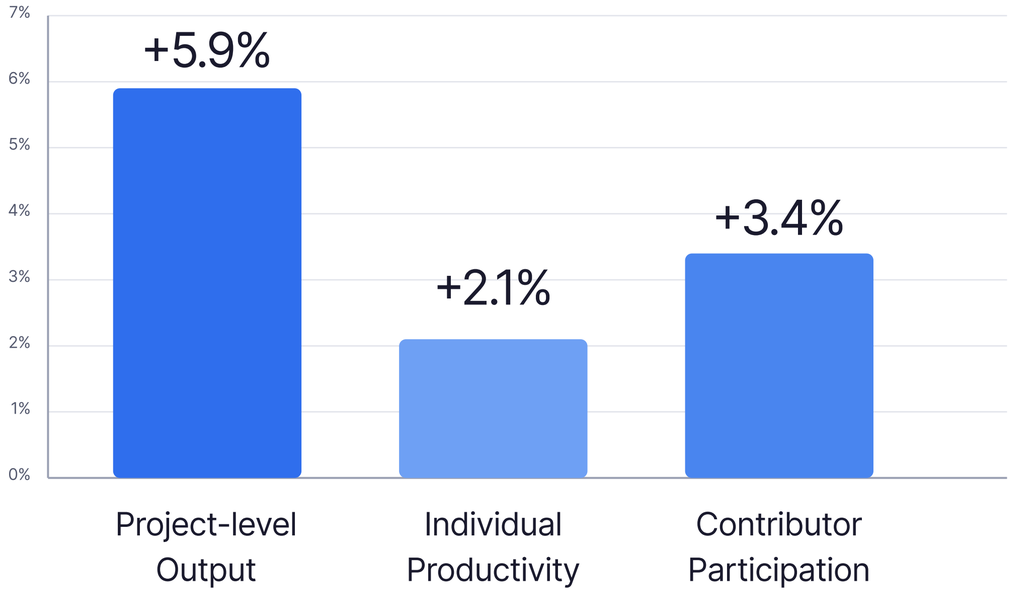
[Visualization from Bremer 2025]



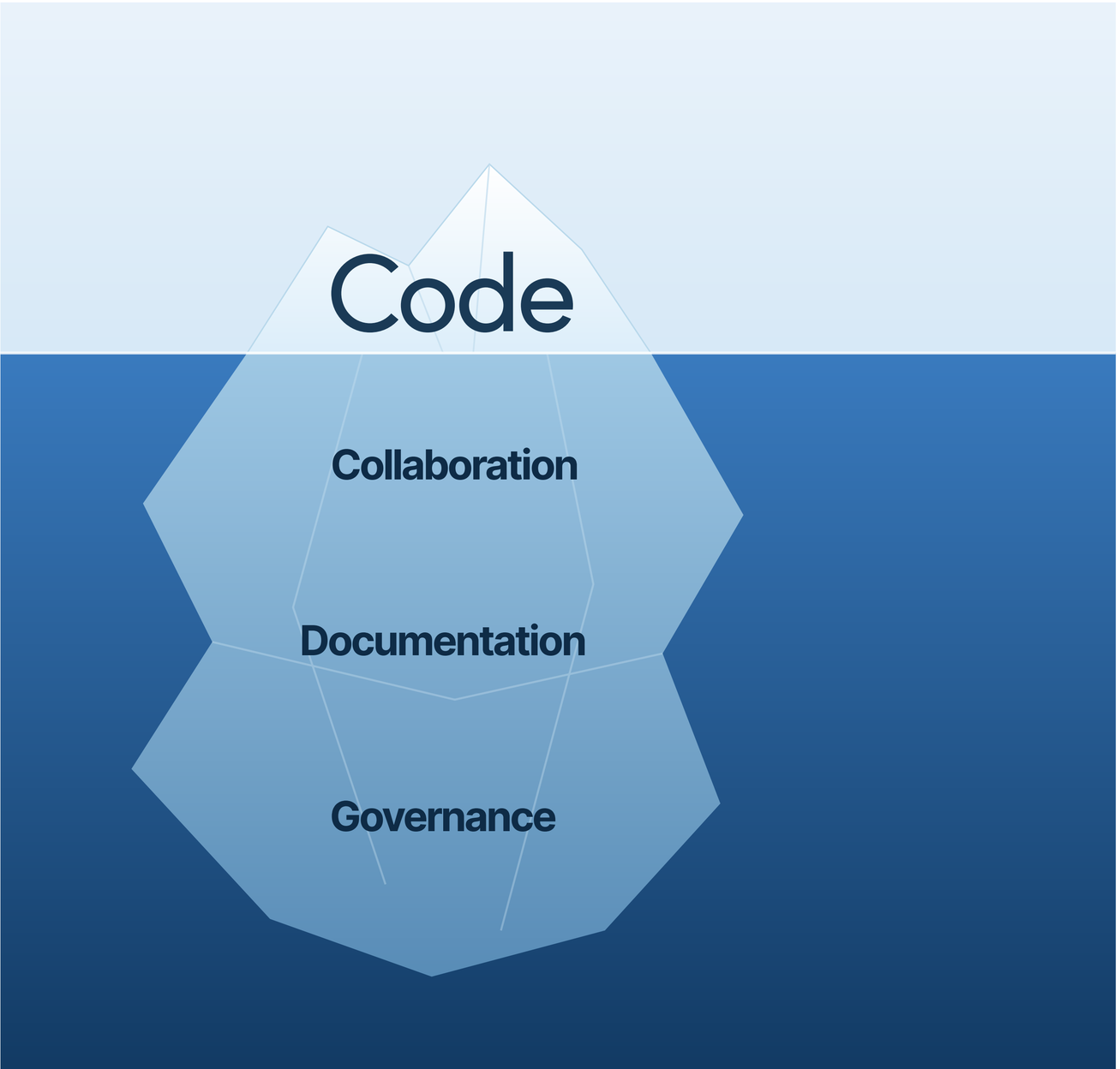
GenAI has arrived in OSS.

94% of OSS organizations report using GenAI for automation, documentation, and code generation
[Lawson, 2024]

But AI contributions skew toward maintenance and refinement, with fewer novel features.
[Yeverechyahu 2026]



[Song 2025]



GenAI is reshaping the socio-technical foundations of OSS.

The code at the heart of OSS is shaped by hidden values, governance, mentorship, and collaborative relationships. GenAI is now reconfiguring all of these.

Our guiding questions and contributions:

Guiding Question 1

What ways of working does GenAI enhance, obsolesce, retrieve, or reverse?

Contribution 1

Structured scenarios of GenAI's socio-technical impacts.

Guiding Question 2

How do we make GenAI work for OSS?

Contribution 2

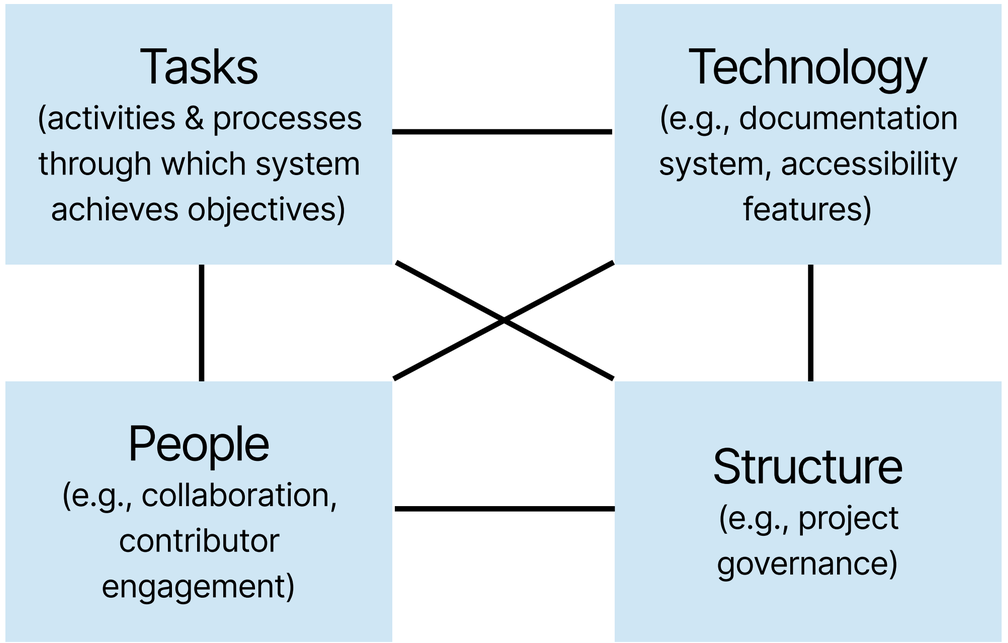
A two-stage research agenda.

Approach

McLuhan's Tetrad meets Socio-Technical Systems.

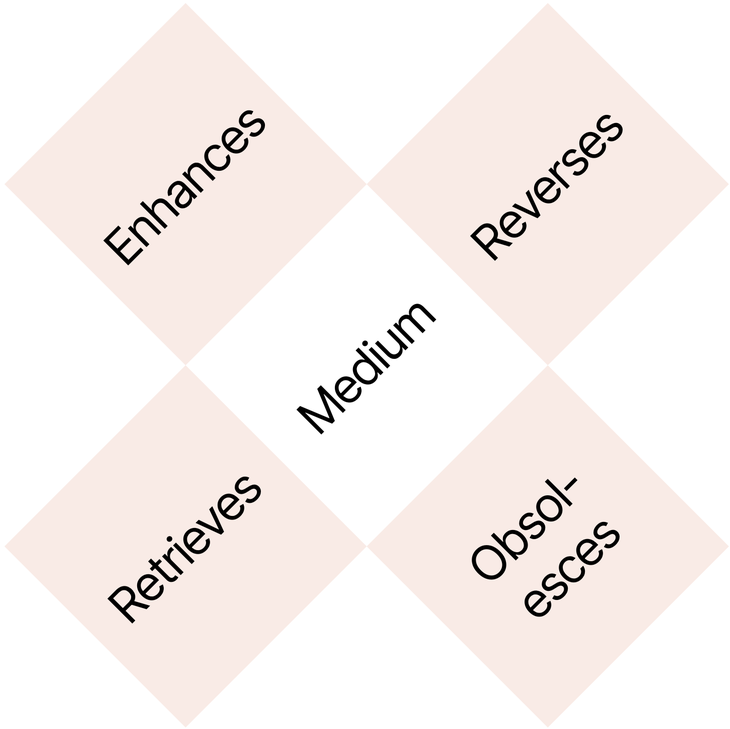
STS Framework Components

for each



McLuhan's Tetrad

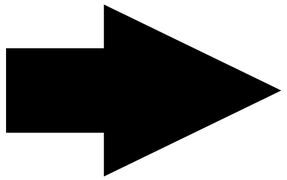
apply



Development & validation:

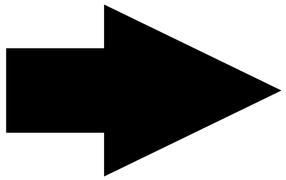
Individual Ideation

- Avoid social loafing
- Reduce normative pressure



Collaborative Synthesis

- Combined, merged and categorized ideas



Expert Validation

- Did not introduce new results
- 30+yr OSS researcher & OSS practitioner

APPROACH

Team composition & positionality:

Academic Authors

- 4 SE researchers: study OSS ecosystems and STS
- Focus: sustainability, inclusivity, mentoring, and contributor onboarding

Industry Authors

- 1 researcher from Microsoft: studies GenAI developer experience
- 1 national lab researcher: studies improving processes & tools across software development lifecycle
- 1 CodeDay executive: studies & implements OSS mentoring programs

The future of OSS in the era of GenAI

Theme	Enhances	Obsolesces	Retrieves	Reverses
AI-Enhanced Testing Rigor	Iteration speed and test coverage: AI accelerates development by generating test scaffolds, suggesting edge cases, and enabling rigorous testing workflows.	TDD and manual test scaffolding: Traditional reliance on hand-written tests, like Test-Driven Development, is replaced by automated pipelines.	Ad hoc experimentation culture: Contributors can prototype quickly, relying on AI to generate test prototypes.	False assurance and blurred accountability: Over-reliance on AI tests may lead to unexamined approvals, missed edge cases, and weakened quality.
AI-Enhanced Code Quality Tenets	Baseline code quality: AI tools like Copilot ensure cleaner, readable code by flagging poor patterns and bugs across experience levels.	Boilerplate coding and superficial reviews: Repetitive tasks like scaffolding and shallow style checks are offloaded to AI.	Unix craftsmanship mindset: Revives simplicity, modularity, and maintainability principles.	Vulnerability and false confidence: Blind reliance on AI code may introduce fragile or vulnerable code, approved without scrutiny.
AI-Driven Technical Debt Management	Technical debt awareness: AI tools scan codebases to identify outdated or redundant code, aiding remediation.	Manual code audits: Periodic reviews and manual debt assessments are replaced by AI automation.	Regular inspection and cleanup: Reinforces norms like removing dead code and enhancing codebase structure.	Shallow fixes: Contributors may address flagged issues superficially, neglecting root causes and long-term design quality.
AI-Augmented Code Review and Feedback	Review efficiency and velocity: AI tools assist OSS reviews by generating comments and identifying issues, speeding up feedback.	Long feedback loops: Reviewers avoid minor style checks, and contributors get faster feedback.	In-depth design discussions: AI frees contributors to focus on architectural trade-offs and mentorship.	Loss of community interaction: Generic AI comments may make reviews transactional, reducing human engagement and community bonds.
AI-Automated Maintenance and Workflow Management	Maintenance tasks: AI automates refactoring, formatting, and documentation updates, reducing maintainer fatigue.	Low-level tedious work: AI alleviates minor bug fixes and code hygiene tasks.	Fast self-healing OSS workflows: Enables rapid issue fixes, supporting "release early, release often."	Prioritization loss: AI-generated commits may bury important issues, reducing maintainer visibility.

Task (software development practices)

Highlights from Task (software development practices)

Enhances

Testing rigor, code quality baselines, technical debt visibility, review speed

Reverses Into

False assurance, blurred accountability, shallow fixes, buried priorities

Obsolesces

Manual test scaffolding, deep audits, human-enforced style review

Retrieves

Ad hoc experimentation

Theme	Enhances	Obsolesces	Retrieves	Reverses
AI-Supported Code Comprehension and Community Contextual Learning	Quick comprehension: AI tools enable rapid understanding of unfamiliar OSS codebases and workflows via contextual explanations and Q&A.	Exhaustive documentation reliance: Reduces the need to study lengthy READMEs, wikis, or discussion threads.	Mentorship and contextual learning: Provides guidance to newcomers, reducing mentor burden and fostering inclusive onboarding.	Misleading guidance and fragmented understanding: Over-reliance on AI may lead to vague or incorrect explanations, reducing documentation quality.
AI-Driven Documentation Maintenance	Documentation maintenance: AI syncs OSS documentation with code changes, easing updates in fast-moving projects.	Manual documentation upkeep: Reduces the burden of revising documentation manually, ensuring alignment with code.	Open contribution philosophy: Simplifies documentation, encouraging collaborative updates and broader participation.	Silent divergence: AI-generated documentation may miss critical context, causing drift between code and its meaning.
AI-Enhanced Code Explanation Practices	Code clarity: AI tools encourage clearer comments, docstrings, and commit messages in OSS projects.	Manual comment quality enforcement: AI flags unclear or missing explanations, reducing style-based rejections.	Inline documentation culture: Revives context-rich, meaningful comments explaining code intent.	Loss of human explanatory skills: Over-reliance on AI-generated summaries may weaken contributors' ability to articulate code reasoning.
AI-Enabled Multilingual Accessibility	Multilingual accessibility: GenAI translation tools make OSS documentation accessible to non-English speakers.	English-only defaults and manual translation: Reduces reliance on English and human translators with dynamic language support.	Globalization ideals: Enables global, inclusive OSS communities with fewer linguistic barriers.	Translation-induced errors: AI translations may introduce errors or awkward phrasing, misaligning technical or community-specific terms.

Technology (docs & accessibility)

Highlights from Technology (documentation and accessibility):

Enhances

Code comprehension for newcomers, always-current docs, multilingual access

Reverses Into

Silent drift between code and docs, translation errors, atrophy of contributors' ability to articulate design intent

Obsolesces

Exhaustive READMEs, manual translation, English-only defaults

Retrieves

Contextual learning, open contribution, globalization

Theme	Enhances	Obsolesces	Retrieves	Reverses
AI-Supported Contributor Retention/Engagement Management	Retention awareness: AI automates onboarding, offboarding, and check-ins, maintaining consistent OSS contributor engagement.	Manual retention efforts: Personalized onboarding and engagement tracking are replaced by AI-driven workflows.	Newcomer community care: Revives OSS values of inclusive, proactive support for contributors.	Trust issues and engagement erosion: Inaccurate or impersonal AI messages may confuse contributors, eroding trust and participation.
AI-Facilitated Cross-Lingual Collaboration	Cross-lingual communication: AI translation tools enable seamless collaboration across language barriers in OSS.	English-dominated gatekeeping: AI reduces the need for English fluency or human translators.	Multilingual participation: Revives OSS goals of global, localized knowledge access.	Cultural miscommunication: AI mistranslations or awkward phrases may reduce credibility and cause confusion.
AI-Enhanced Respectful, Inclusive Communication	Community tone: AI identifies toxic language and promotes respectful, inclusive OSS communication.	Manual moderation: AI reduces the need for human moderators to monitor tone.	Code of Conduct ideals: Reinforces inclusive, welcoming OSS spaces for diverse contributors.	Over-filtering and suppressed dialogue: AI's excessive moderation may silence valid perspectives, weakening trust.
AI-Augmented Learning and Mentorship	24/7 knowledge access: AI acts as an always-available mentor, helping OSS contributors navigate issues and workflows.	Traditional mentoring: Fixed-time office hours and formal mentoring are less necessary.	Peer-driven learning: Revives informal collaboration and mentorship in OSS.	Shallow learning and silos: Over-reliance on AI may lead to copy-paste habits and skipped foundational understanding.
AI-Supported Interdisciplinary Collaboration	Interdisciplinary collaboration: AI provides tailored explanations, easing cross-expertise collaboration in OSS.	Long feedback loops: AI reduces delays from domain misunderstandings.	Inclusive collaboration: Revives OSS's vision of diverse, non-specialist contributions.	Siloed understanding: Over-reliance on AI translations may hinder direct engagement, risking disconnected teams.

People (community & collab)

Highlights from People (community engagement and collaboration)

Enhances

Onboarding, cross-lingual collaboration, 24/7 AI mentorship, respectful communication.

Reverses Into

Impersonal messages, eroding trust, shallow learning, weakened belonging, disengagement

Obsolesces

Manual engagement efforts, English-dominated gatekeeping, traditional mentoring, long feedback loops

Retrieves

Multilingual participation, peer-driven learning



Theme	Enhances	Obsolesces	Retrieves	Reverses
AI-Supported Governance Docs	Governance drafting: AI aids in drafting comprehensive governance documents like Codes of Conduct and contribution guidelines.	Delays in governance documentation: Reduces delays in creating foundational documents, especially for new OSS projects.	Governance scaffolding: Establishes norms that support contributor participation in smaller projects.	Governance as placeholder: AI-generated documents may become superficial, rarely read or enforced.
AI-Assisted Licensing and Legal Guidance	License support: AI validates license compatibility and provides 24/7 Q&A, improving legal awareness.	Compliance checks: Eliminates delays in legal reviews, offering instant guidance for licensing issues.	—	Compliance noise: AI may produce inaccurate advice or false positives.
AI-Driven Policy Enforcement	Policy enforcement: AI flags violations of project policies, ensuring consistent enforcement in OSS.	Human-only moderation: Reduces moderator burden and supports newcomers in reporting issues.	Inclusive contribution: Reinforces safe, accountable OSS environments for all contributors.	Surveillance discomfort: Overuse of AI may make contributors feel monitored, eroding trust.

Structures (governance)





Highlights from Structures (governance practices)

Enhances

Drafting codes of conduct, license compatibility checks, consistent policy enforcement

Reverses Into

Unread governance documents, compliance noise, surveillance discomfort, eroded trust

Obsolesces

Delays in creating documents or checking compliance

Retrieves

Governance scaffolding, inclusive contribution

Takeaways

What can practitioners do now?

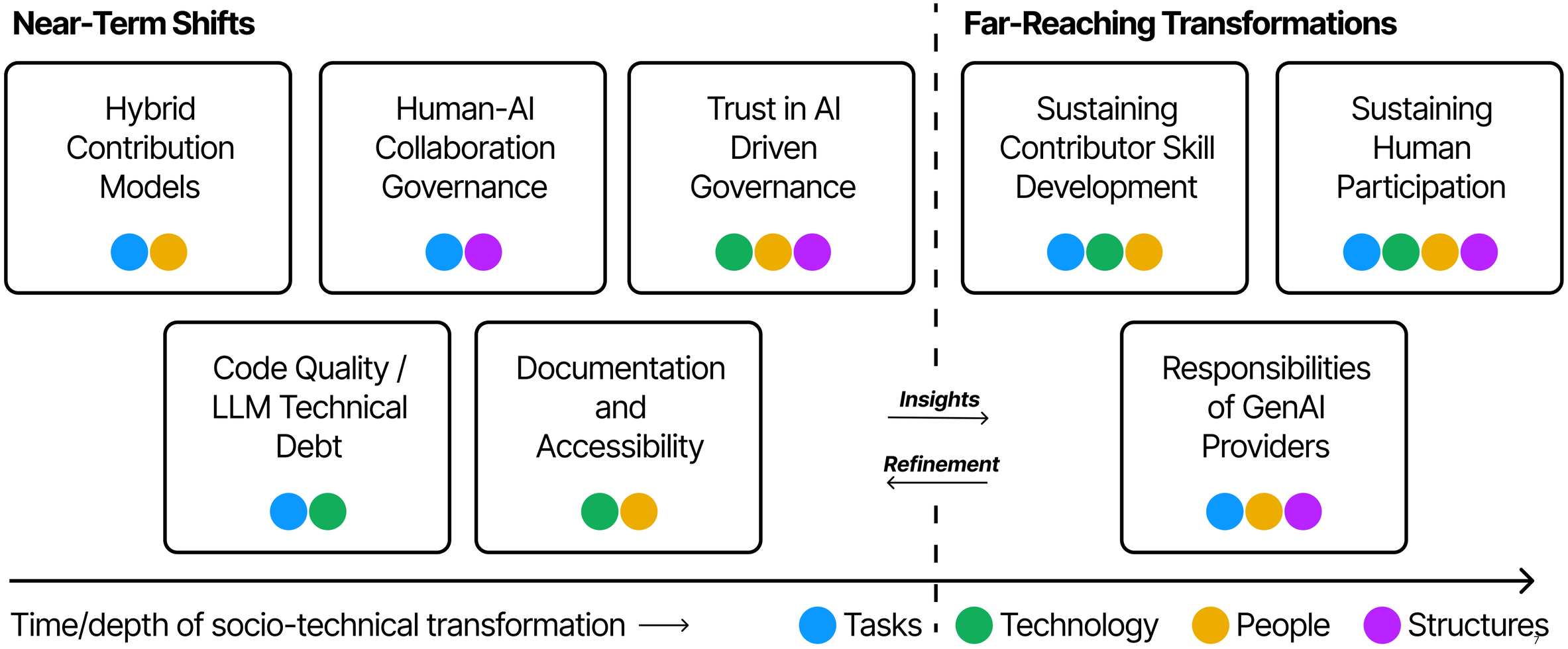
Maintainers
Use the Tasks table as a sprint retrospective checklist. Ask "which workflows are trading reflection for speed?"

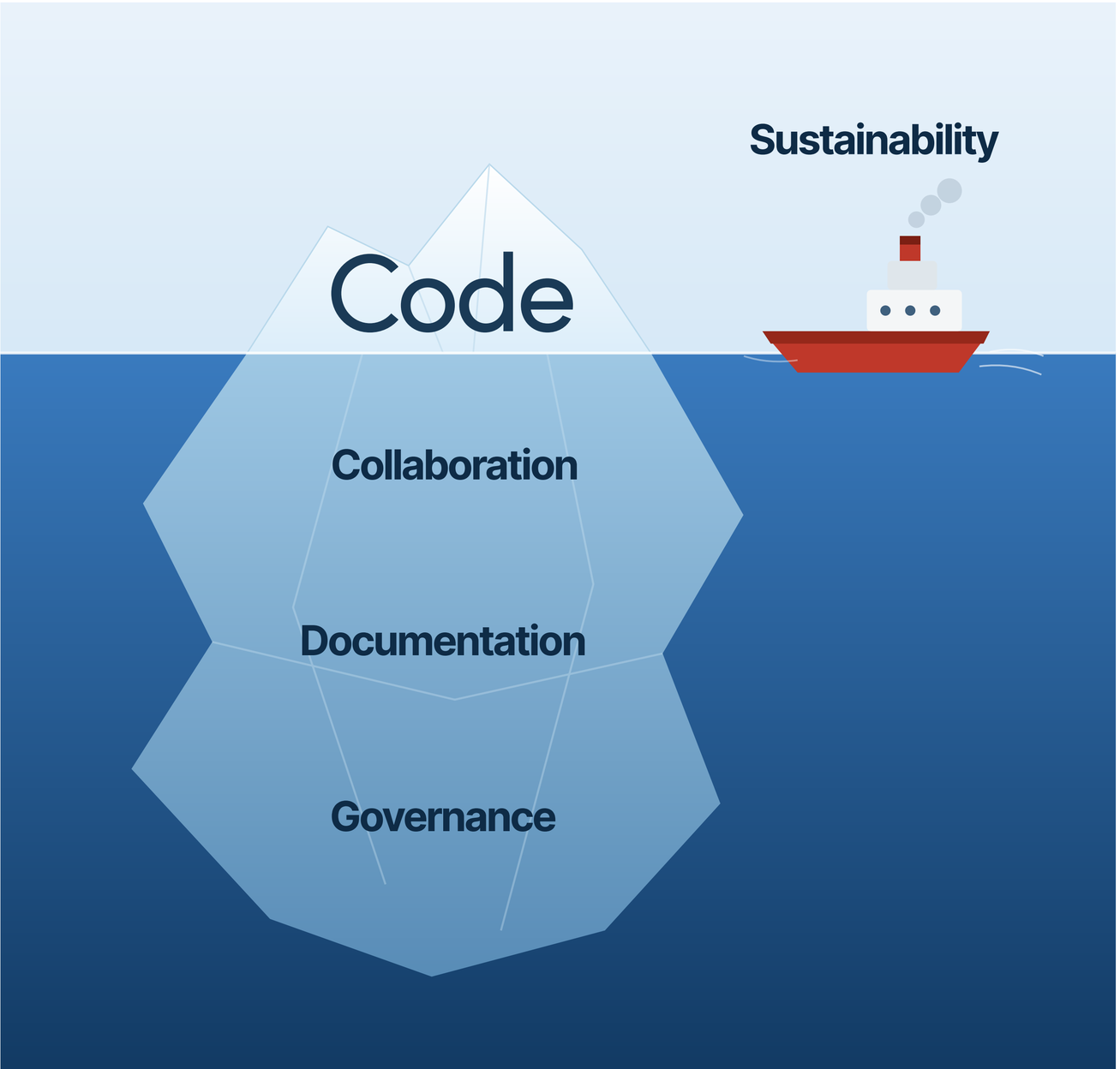
Project Leaders
Apply the Structures table as a checklist to decide what to automate and what requires human oversight.

Documentation Leads
Use the People table to map where human context must be preserved.

Community Managers
Use the People table to guide monthly discussions on mentoring and engagement.

What needs further study?





GenAI's impact on OSS is much deeper than code.

It is reconfiguring collaboration, knowledge, governance, and the conditions for community survival.

If we wait for AI vendors to solve everything, we risk losing control of the principles that make OSS resilient and inclusive.

Download the Paper

doi.org/10.1145/3789210

Contact Me

linkedin.com/in/tylermenezes
tylermenezes@codeday.org

References

Bremer, N. (2025). Visualizing the Top Contributor Network of Any GitHub Repo. Visual Cinnamon. Retrieved July 6, 2026, from https://www.visualcinnamon.com/2025/01/github-top-contributor-network/

Song, F., Agarwal, A., & Wen, W. (2023). The Impact of Generative AI on Collaborative Open-Source Software Development: Evidence from GitHub Copilot (SSRN Scholarly Paper No. 4856935). Social Science Research Network. https://doi.org/10.2139/ssrn.4856935

Lawson, A., Hendrick, S., Rausch, N., Sica, J., & Gerosa, M. (2024). Shaping the Future of Generative AI. The Linux Foundation.

Yeverechyahu, D., Mayya, R., & Oestreicher-Singer, G. (2026). The Impact of Large Language Models on Open-source Innovation: Evidence from GitHub Copilot (arXiv:2409.08379). arXiv. https://doi.org/10.48550/arXiv.2409.08379