GitHub

Best practices for a collaborative software development culture

Collaboration, transparency, and code reuse aren't just found in open source communities. Successful software developers build together the same way at work—an approach known as "innersource." These innersource best practices will help your team create a developer culture that is effective and collaborative.

Be transparent

Transparency is the baseline for any innersource culture, just as it is for the wider open source community.

- Share projects across your organization as broadly as possible. Exceptions can be made for projects that require a higher level of security on a case-by-case basis.
- Ensure that your team has the tools and processes to communicate openly and build consistently—and that those tools and processes are being used. If they're not, find out why and make adjustments until you establish a solid routine.
- Confirm that your engineering initiatives are well-resourced and supported by leadership.
- Give developers enough autonomy to contribute to projects outside their immediate teams.

Distribute workloads

To effectively adopt innersource practices, contributors need to be able to work easily across teams and organizational silos.

- Make approvals and reviews more effective by distributing control across a smaller group of participants. A small, cross-functional team of decision-makers can also help teams stick to quality standards and get support from leadership.
- Spread heavy workloads across geographically distributed developers or even with non-developer team members.
- Keep in mind that non-developers are a valuable asset to your innersource community.

  Project managers can quickly assess the state of the project, and changes made to project timelines can be tied directly to the code in question.

  Technical writers should be encouraged to work alongside project engineers, precisely documenting code features where they originate.

  Security engineers can review code for compliance and security-related weaknesses, preventing issues earlier in the workflow.

Encourage communication and ongoing review

Openly sharing work and feedback brings more ideas into your innersource community—and more code.

- Make ample use of your version control system's commenting, tagging, and review features.
- Use forks and branches so developers can try different changes without affecting original projects. These tools also make it easy to start with existing solutions, instead of writing new code each time.
- Encourage developers to share their latest work early on and often by tagging others in their code commits.
- If someone finds a bug, they should create an issue describing the problem. Tag that same contributor when you update the code so they know the bug has been fixed.
- Make each step in the software development lifecycle (such as testing and deployment) visible to everyone.

Don't forget the documentation

As innersource projects become more complex, in-code documentation keeps work moving and consistent.

- Document by default, not as an afterthought. Effective documentation makes your decision process clear to outside contributors and avoids wasted resources and duplication.
- Add README and CONTRIBUTING guides to all repositories to help everyone understand basic usage and project best practices.
- Use documentation to help onboard and acclimate new contributors and other participants into your process.
- Update in-code documentation regularly to keep user guides and manuals in sync with the latest version of the code.

Want to try innersource with your team?

Contact us:

github.com/enterprise
experts@github.com