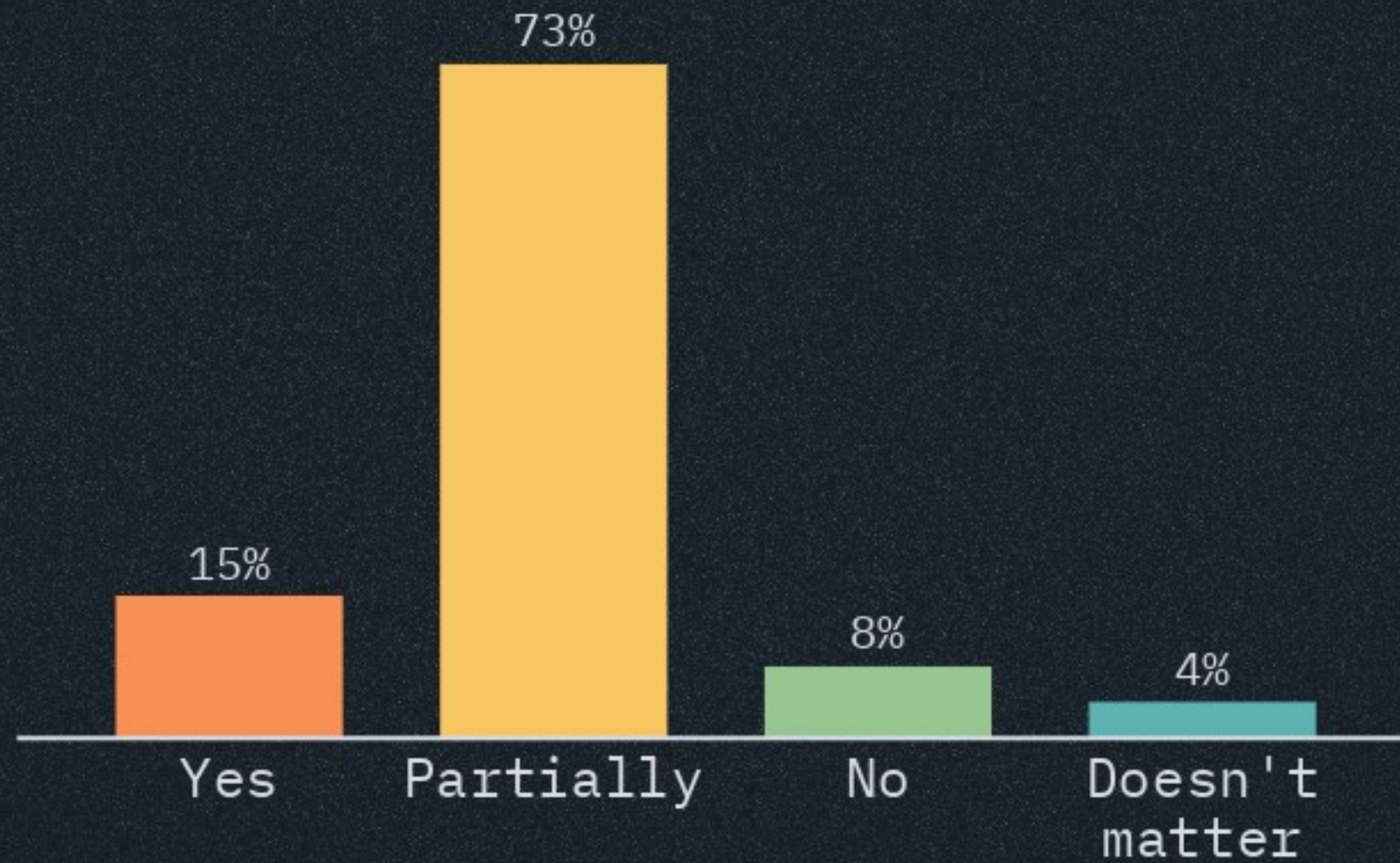


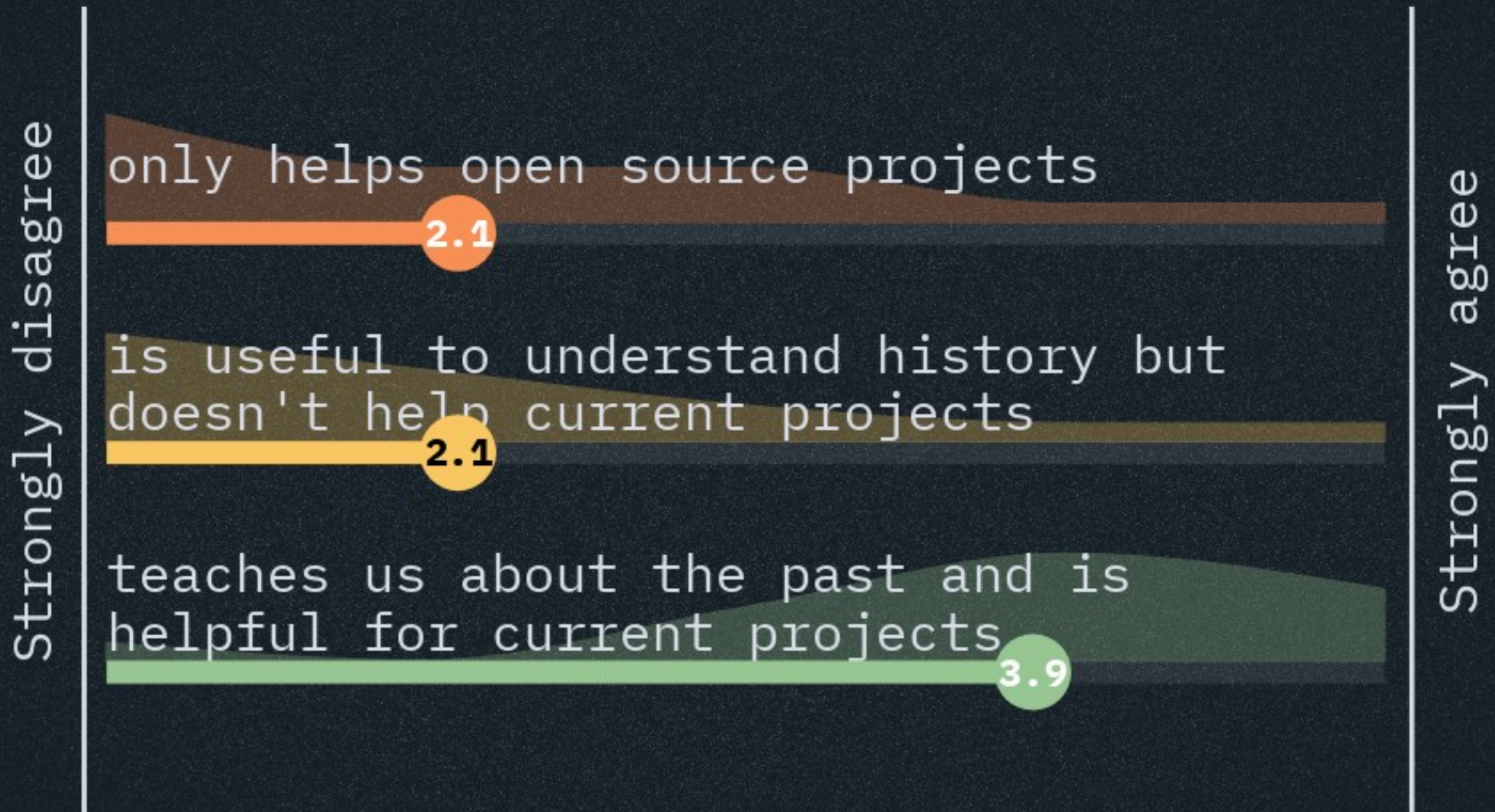
Is SE research addressing problems of multi-person multi-version development?



What areas of software engineering do you think should be studied more?

technology transfer
industrial software software ecosystem
software design process process management
regression testing developer communication human aspects
software evolution
software architecture
human factors tool and support building modern sqa practices
configuration-management continuous integration
end user development actual engineering work
design decisions software specifications
iot and cps development version management
collaboration knowledge management
defect reporting
software quality

Studying software ecosystems...



What questions about multi-person multi-version development are interesting for individual projects?

How can we ensure software quality?

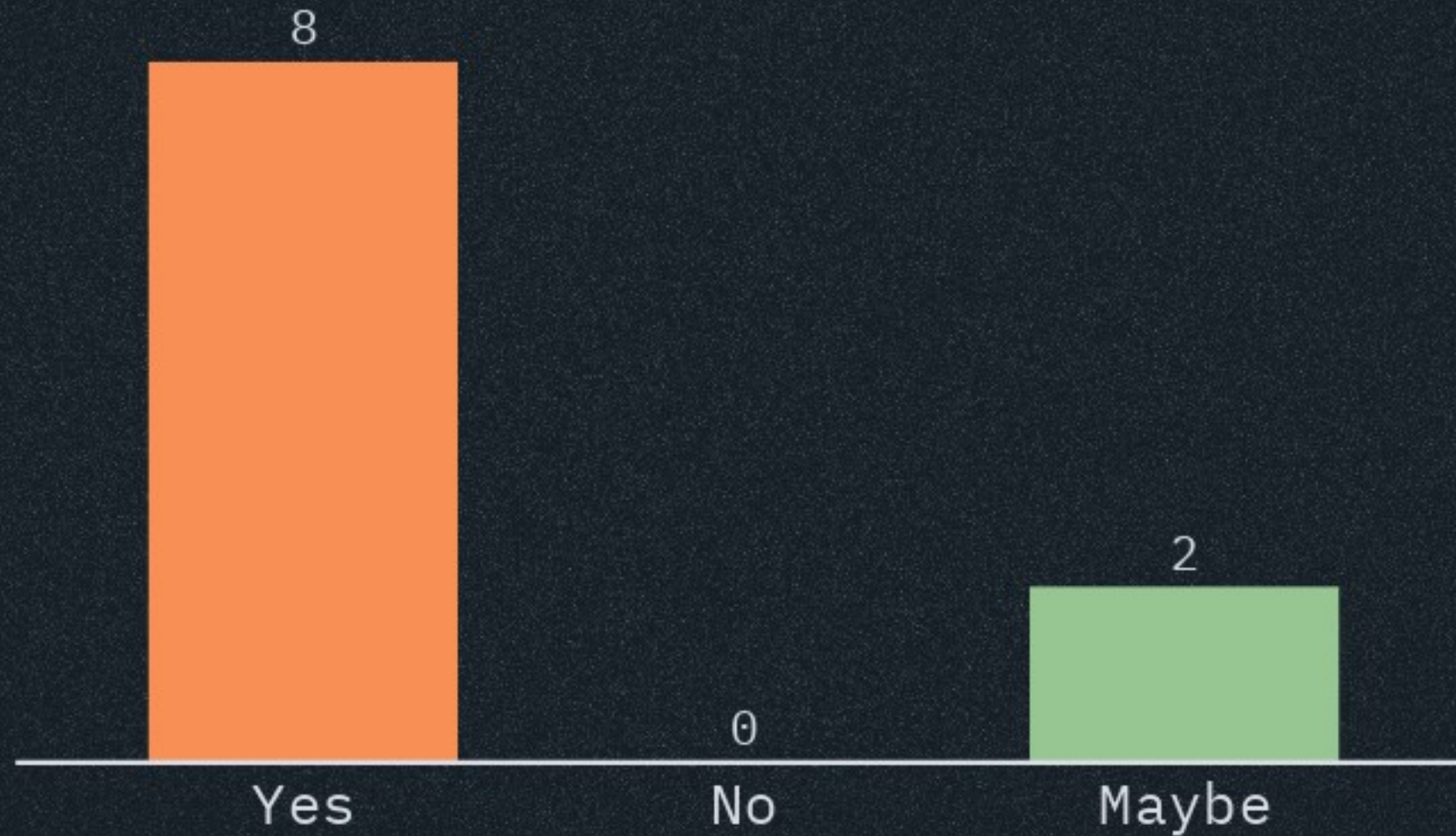
How do build up the most effective team/sub-teams and best support them?

What dependencies should you use? How to nudge developers of those dependencies to make changes if you need them? How to isolate yourself from their changes if you don't want them?

How to effectively share knowledge in development teams?

how the individual project's development process is effected by other projects or developers in the same ecosystem, in the context of multi-person multi-version development.

Can you think of interesting opportunities to automate parts of software development flow?



Is SE research addressing problems of multi-person multi-version development?

