Understanding the Impact of Research Software Engineers on Scientific Software Development: A Mixed-Methods Study

Thomas Bock, Will Sutherland, Curtis Atkisson, Anissa Tanweer, James D. Herbsleb, Bogdan Vasilescu

Motivation

Science depends heavily on software:
Scientific software makes science possible, scalable, and reproducible

Scientists

Lack formal training of software-engineering practices

Scientific output valued over software quality and sustainability

Research Software Engineer (RSE)

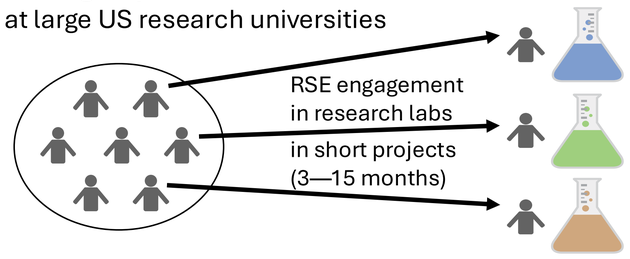
Lacks domain knowledge of scientific research

Bridges gap between software development and scientific practice

What's the impact of RSEs on scientific software development and scientific productivity?

Setting and Approach

Focal group: *Centralized RSE teams* at large US research universities



Mixed-methods approach:

Qualitative component:
Approximately 80 semi-structured interviews with RSEs, RSE managers, and scientists

Quantitative component:
Repository data analysis,
heuristics to identify RSEs, measuring impact on software practices and scientific output

Research Questions:

RQ1: What **challenges** do RSEs and scientists face in effective collaboration?

RQ2: How does RSE collaboration impact the software **development process** and culture?

RQ3: What is the **technical impact** on code quality and productivity?

RQ4: What is the **scientific impact** on scholarly output and software reuse?

Challenges

Challenges in assessing the impact of RSEs:

- Who is an RSE?
- Do **RSEs' backgrounds** influence the impact?
- Do **scientists' backgrounds** influence the impact?
- Does the **scope of a scientific project** influence the impact?
- Does the **maturity of a scientific project** influence the impact?

Some dimensions of difference in RSE work:

Scope and maturity of science
Less stabilized science ⟷ *More stabilized science*

RSE role and background
RSE professionalized as Scientist ⟷ *RSE professionalized as Software Engineer*

Work environment
RSE embedded in a research lab ⟷ *RSE mobile across projects*

What other dimensions matter?
How are the dimensions related to each other?
What are the challenges and benefits of various combinations of dimensions?

Qualitative Component

Preliminary results from the interviews:
17 interviews so far (8 with RSEs, 9 with RSE managers)

Benefits and challenges of long-term vs. short-term engagement

Long-term: Time to develop domain knowledge

Short-term: Ability to change projects

Long-term: Understand research group's practices

Short-term: Learn new tools regularly

Different understanding of RSE role

Teach scientists software engineering practices

Focus on software engineering work and let scientists do programming

Next steps: explore these and other contrasts, to understand how they might be systematic or structural

Quantitative Component

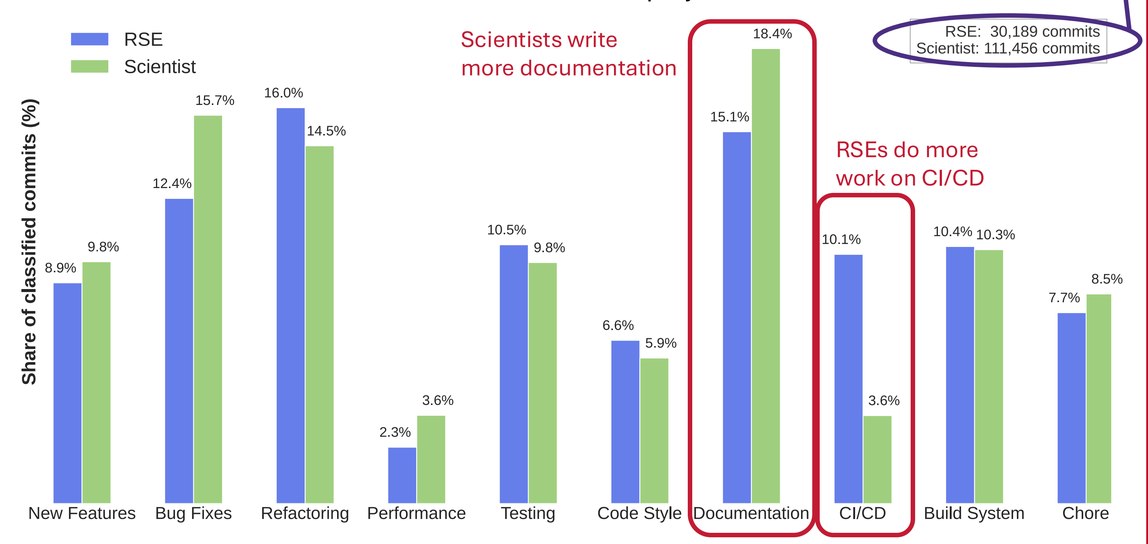
Preliminary results from repository analysis:
based on 54 scientific projects with RSE involvement

Per-project statistics	Min	Median	Max
Commits	17	2 626	31 035
Files	14	755	10 215
Project time period	0.1 years	2.1 years	18.4 years
RSEs with commits	0	2	5
Scientists with commits	0	4	204

Questions yet to be answered:

- What happened in these **long-lasting** projects **before the establishment of the RSE role**?
- How many of these non-RSEs are **really scientists**? Are there quasi-RSEs (e.g., students, etc.)?

Commit types of RSEs and scientists using Conventional Commits Specification (CCS):
based on 39 RSEs and 1018 scientists in 54 scientific projects with RSE involvement



Outlook

How does RSEs' work change with the rise of Generative AI usage?
RSEs' reflections from our interviews:

Scientists come with more sophisticated problems to RSEs as AI can solve the simpler ones now

"you're kind of sticking your head in the sand at this point if you're not at least trying"

RSEs get more AI slop to review from scientists and other RSEs

SERS #23

We acknowledge funding by: Schmidt Sciences, Alfred P. Sloan Foundation

Extended Abstract: