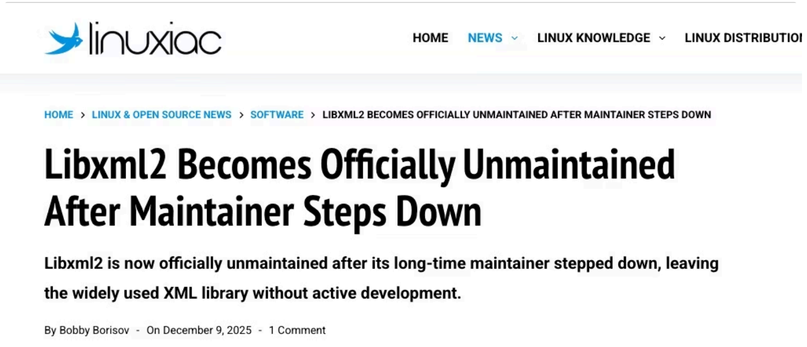
Estimating the Maintenance Burden of Rust vs C Open Source Libraries

Blake Kell and Kudzai Dhewa

Motivation


linuxiac
HOME
NEWS
LINUX KNOWLEDGE
LINUX DISTRIBUTIOI
HOME › LINUX & OPEN SOURCE NEWS › SOFTWARE › LIBXML2 BECOMES OFFICIALLY UNMAINTAINED AFTER MAINTAINER STEPS DOWN

Libxml2 Becomes Officially Unmaintained After Maintainer Steps Down

Libxml2 is now officially unmaintained after its long-time maintainer stepped down, leaving the widely used XML library without active development.

By Bobby Borisov - On December 9, 2025 - 1 Comment

Libxml2 is an essential open source XML parser implemented in C that has lost all of its maintainers. It comes preinstalled on most UNIX-based systems: Linux, Android, macOS, ChromeOS.

Our expert is considering assembling a team to migrate the library from C to Rust. Our goal: help him make an informed, data-driven decision.

Research Questions

1

Primary Research Question

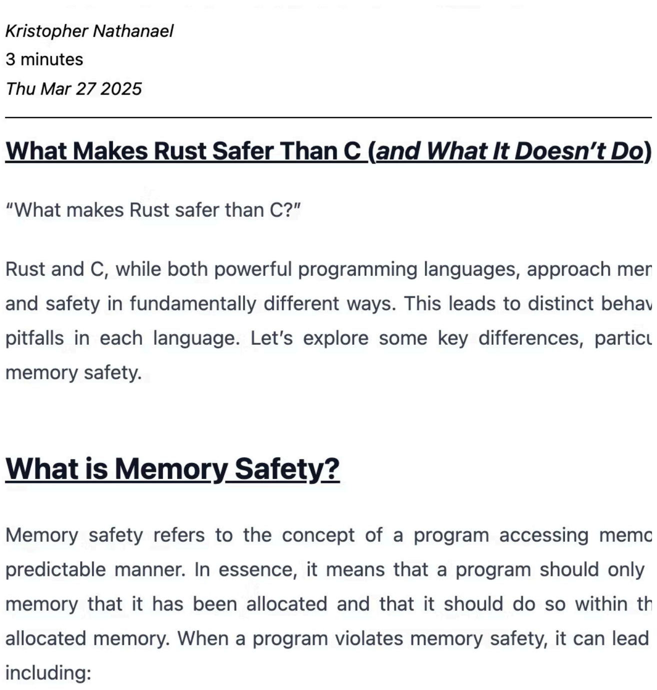
Is the maintenance burden of open-source libraries lower for projects written in Rust compared to those written in C?

2

Secondary Research Question

What maintenance activities are most common in C vs Rust open source libraries?

Rust vs C Overview



Kristopher Nathanael
3 minutes
Thu Mar 27 2025

What Makes Rust Safer Than C (*and What It Doesn’t Do*)

“What makes Rust safer than C?”

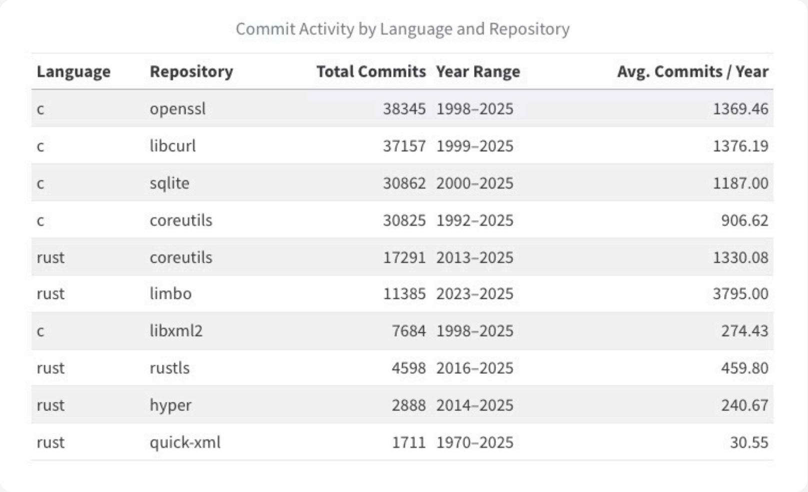
Rust and C, while both powerful programming languages, approach mer
and safety in fundamentally different ways. This leads to distinct behav
pitfalls in each language. Let’s explore some key differences, particu
memory safety.

What is Memory Safety?

Memory safety refers to the concept of a program accessing memc
predictable manner. In essence, it means that a program should only
memory that it has been allocated and that it should do so within th
allocated memory. When a program violates memory safety, it can lead
including:

- **C** is a mature systems language from the 1970s
- **Rust** is a newer systems language from the 2010s
- Rust is widely regarded as more memory safe
- Rust's compiler enforces strict memory rules at compile time
- C allows unrestricted memory access—errors are the programmer's responsibility
- As a result, programmers generally make **fewer memory-related errors in Rust** than in C

> **Hypothesis:** Rust requires less maintenance than C because of higher memory safety.



Commit Activity by Language and Repository

Language	Repository	Total Commits	Year Range	Avg. Commits / Year
c	openssl	38345	1998–2025	1369.46
c	libcurl	37157	1999–2025	1376.19
c	sqlite	30862	2000–2025	1187.00
c	coreutils	30825	1992–2025	906.62
rust	coreutils	17291	2013–2025	1330.08
rust	limbo	11385	2023–2025	3795.00
c	libxml2	7684	1998–2025	274.43
rust	rustls	4598	2016–2025	459.80
rust	hyper	2888	2014–2025	240.67
rust	quick-xml	1711	1970–2025	30.55

Data Collection

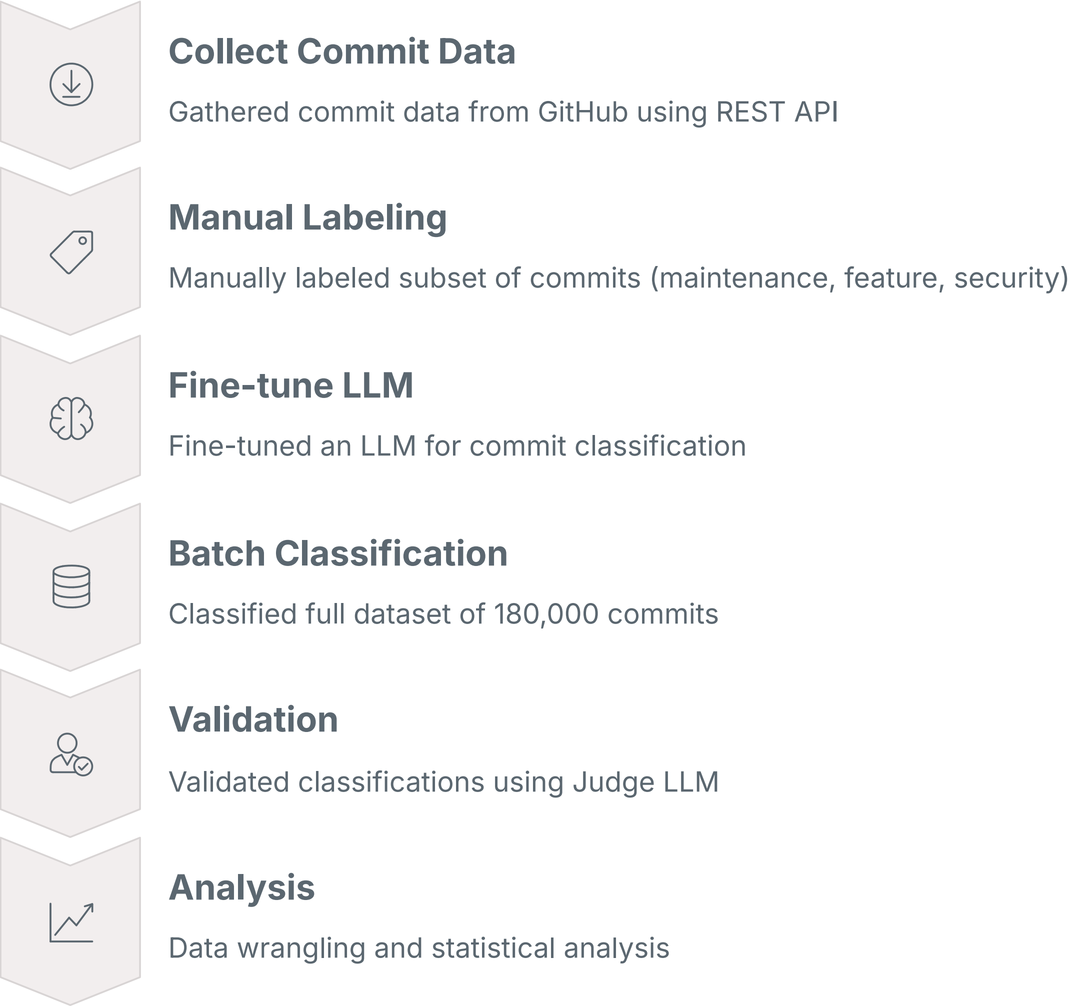
Library Selection

Sampled ten widely used open-source libraries: 5 Rust and 5 C, using pairwise matching in similar functional domains (XML parsing, Database, HTTP, System Utils, Cryto/TLS.

Selection Criteria

Projects in maintenance phase rather than early development, validated through expert consultation.

Data Acquisition and Processing



Collect Commit Data

Gathered commit data from GitHub using REST API

Manual Labeling

Manually labeled subset of commits (maintenance, feature, security)

Fine-tune LLM

Fine-tuned an LLM for commit classification

Batch Classification

Classified full dataset of 180,000 commits

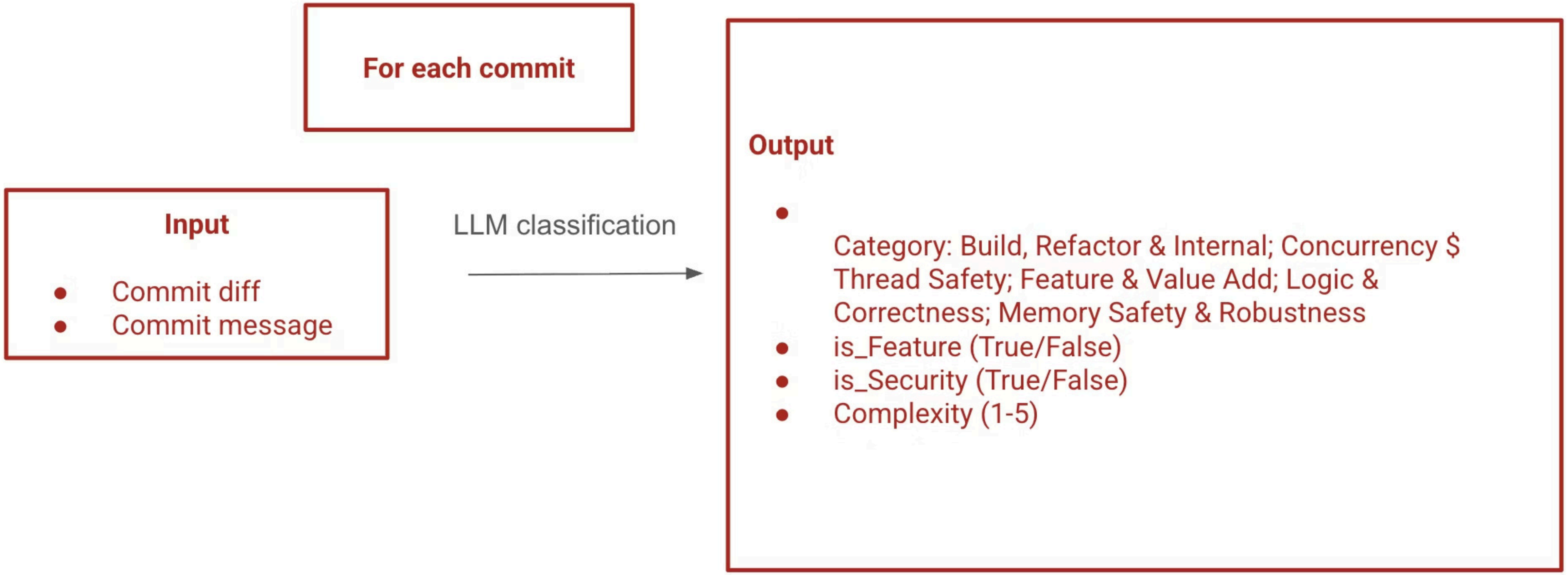
Validation

Validated classifications using Judge LLM

Analysis

Data wrangling and statistical analysis

LLM Classification



For each commit

Input

- Commit diff
- Commit message

LLM classification

Output

- Category: Build, Refactor & Internal; Concurrency $ Thread Safety; Feature & Value Add; Logic & Correctness; Memory Safety & Robustness
- is_Feature (True/False)
- is_Security (True/False)
- Complexity (1-5)

Results

Maintenance Burden by Domain

Maintenance Consistently Higher in C

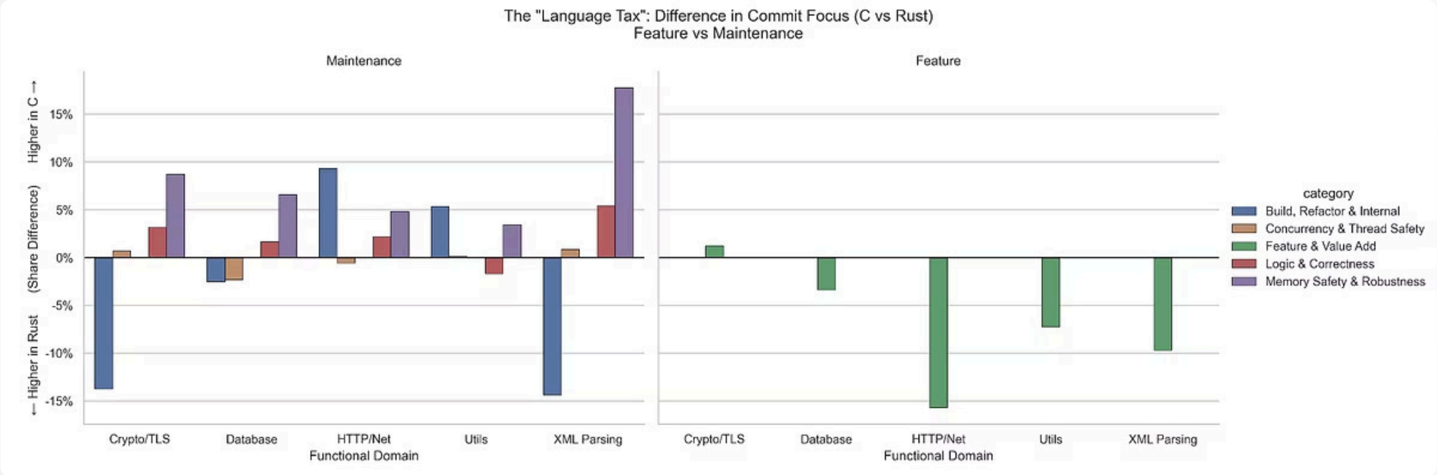
Across all domains, maintenance work is positive except refactoring in two domains, indicating C requires more maintenance than Rust.

Memory Safety Gap

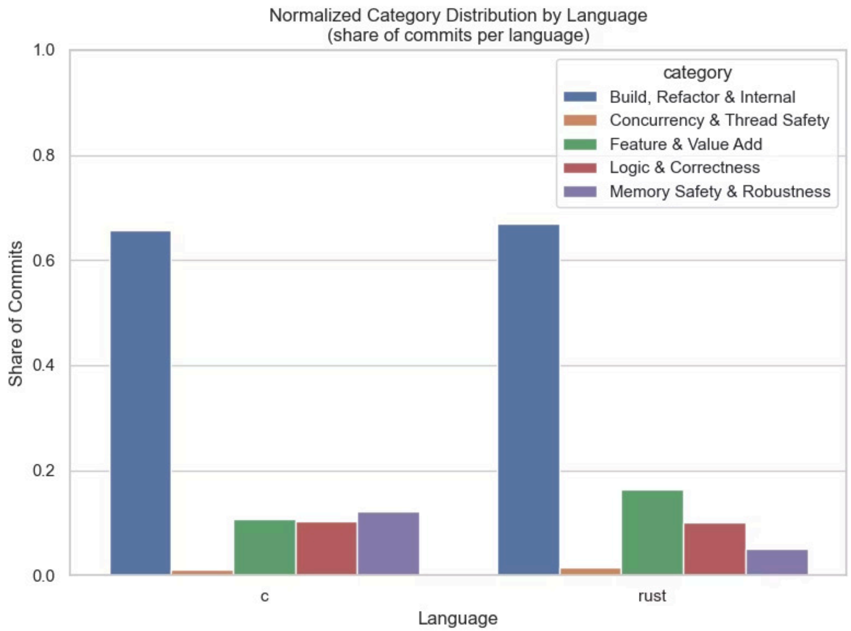
Memory-safety maintenance (purple bars) is consistently higher in C projects—nearly 20% in the XML comparison (libxml2 vs. quick-xml).

Rust Feature Development focus

Feature and value-add work (green bars) is consistently higher in Rust, suggesting reduced bug-fixing time enables innovation.



Memory Safety: 3x Difference



12%

C Memory Safety

Commits related to memory safety in C sample

4%

Rust Memory Safety

Commits related to memory safety in Rust sample

3x

Reduction Potential

Migrating to Rust could mean 3x fewer memory safety issues

C also shows a higher proportion of general logic and correctness fixes.

Commit Distribution by Library

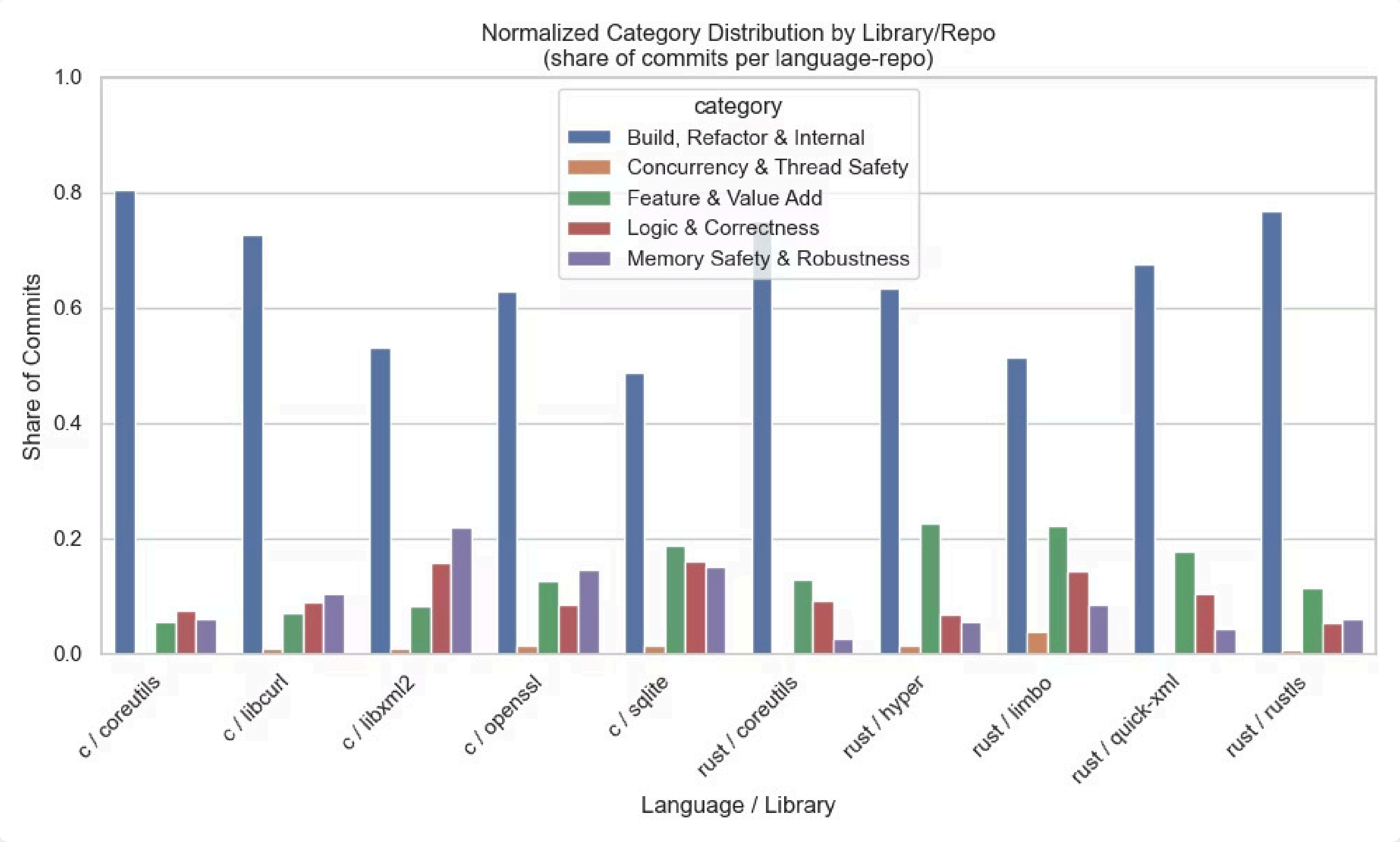
Structural Difference

C projects weighted toward maintenance work; Rust projects weighted toward features.

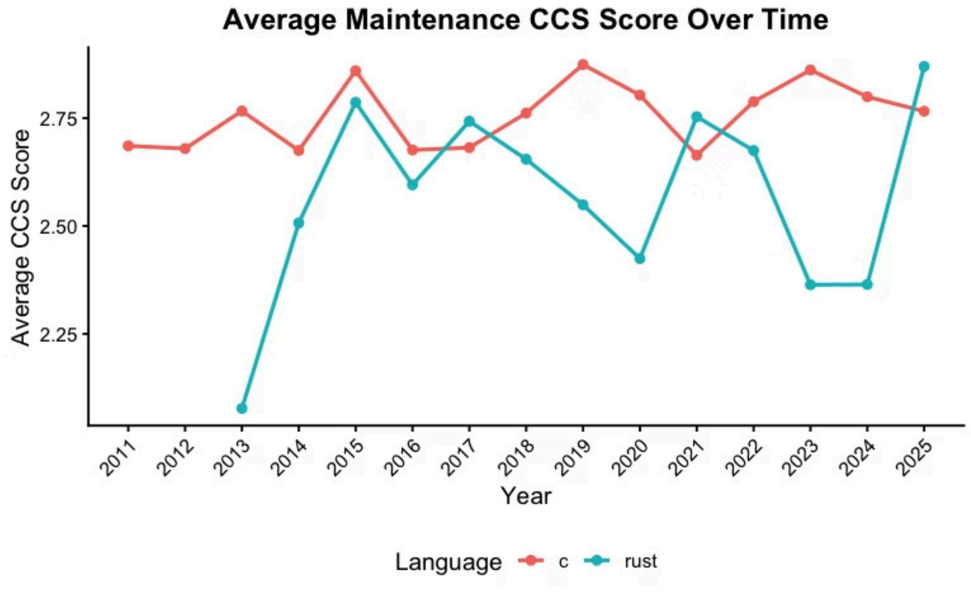
Purple and brown bars larger than green for all C libraries; vice versa for Rust.

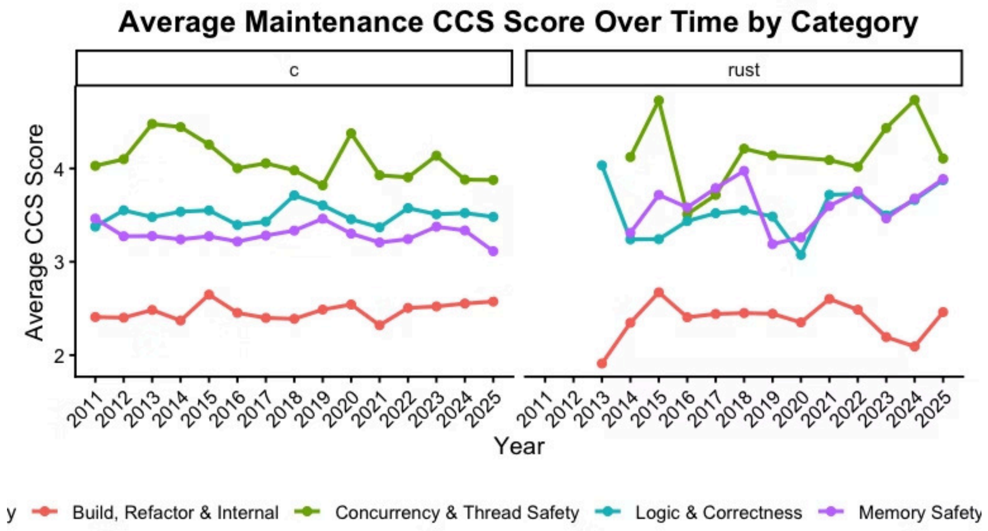
Libxml2

Dominated by logic and memory fixes, while Rust counterpart quick-xml focuses on features.



Commit Complexity Score





→ Lower Overall Complexity

Rust generally has lower commit complexity scores, indicating lower maintenance burden.

→ Memory Safety Complexity Trade-off

Memory safety commits for Rust are more complex than in C—an upfront investment that leads to less maintenance long-term.

Conclusion

01

Focus Shift

C forces focus on fixing past memory safety issues; Rust enables focus on growth and new features.

02

3x Reduction

Migrating libxml2 to Rust might lead to 3x reduction in frequency of memory-related maintenance events.

03

Reduction in memory safety maintenance seems to correlates with increase in feature development work.

04

Rust commits are more complex for memory safety, but this upfront investment results in efficient long-term maintenance.

Long story short….Migrating to Rust is a good idea

Limitations

Language Maturity

C is much more mature than Rust, making it challenging to find truly comparable libraries.

Statistical Validation

Regressions to validate correlation between reduction in memory safety maintenance and increase in feature work would strengthen findings.

Complexity Measurement

LLM-based complexity score might not be the most accurate measurement approach.

Thank you to our domain expert: Josh Aas, Shilad Sen and everyone for listening