Message Passing in Rust Applications

Author: Mare Pegels - M.A.Pegels@student.tudelft.nl | **Responsible Professor:** Andreea Costea | **Supervisor:** Ruben Backx

1. Introduction

Why Rust?

- Modern systems programming language
- Memory safety without garbage collection
- Widely used for concurrent applications

Research Gap

Limited empirical evidence on message passing in Rust applications

This study investigates message passing in Rust applications through repository classification and manual analysis.

2. Research Questions

RQ1 **Which message-passing primitives are used?**

RQ2 **Which architectural patterns appear?**

RQ3 **What roles does message passing serve?**

3. Methodology

Repository Selection

▼

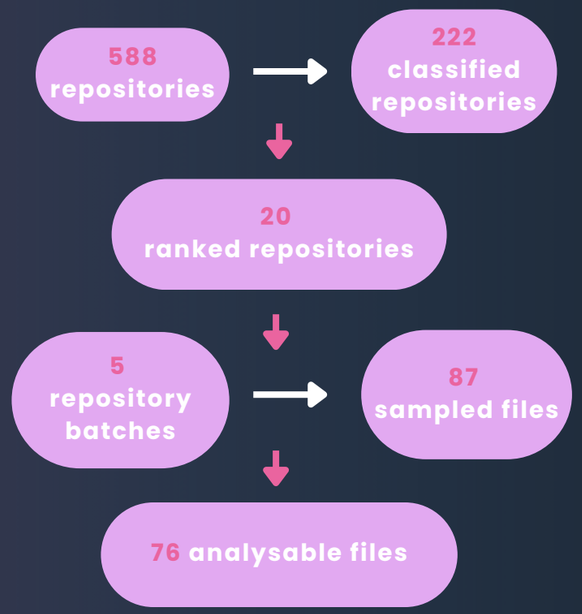
LLM Classification & Ranking

▼

File Sampling

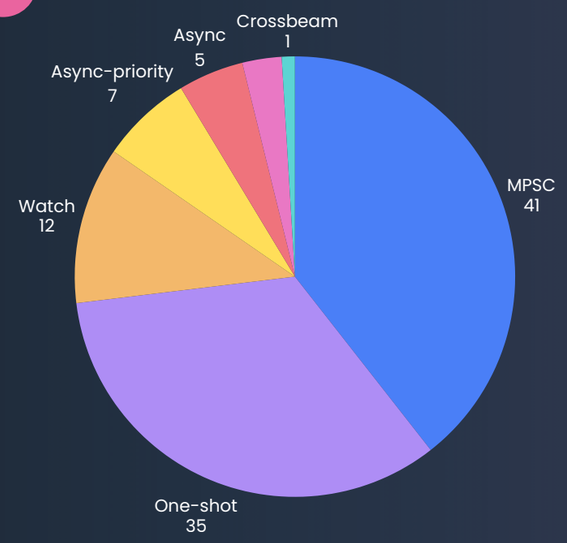
▼

Template-guided Manual Analysis

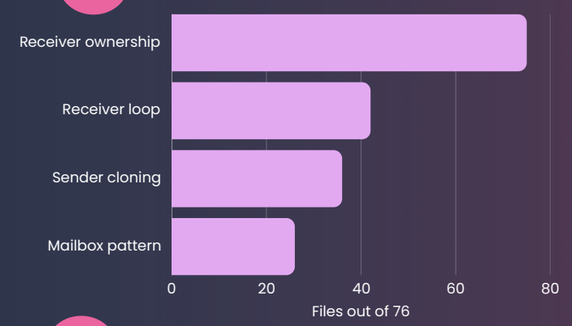


4. Results

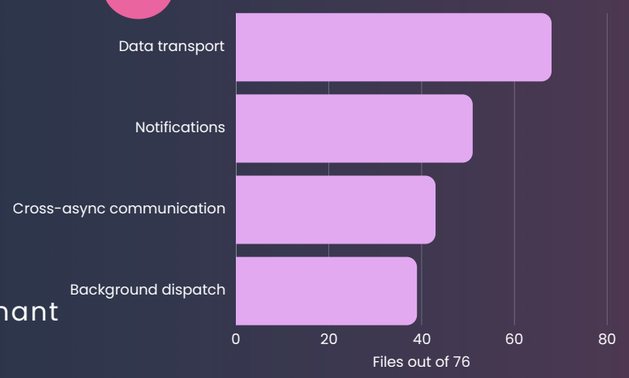
RQ1 Primitive creation sites in sampled analysable files



RQ2 Communication Structures



RQ3 Message passing roles



Key findings

RQ1: MPSC and one-shot are the dominant message-passing primitives.

RQ2: Receiver ownership is the predominant architectural pattern.

RQ3: Message passing primarily supports data transport, notifications, and cross-task coordination

5. Conclusions

- Rust applications use diverse message-passing primitives and communication structures.
- Message passing provides flexible coordination between concurrent components.

6. Future work

- Expand the repository dataset to validate the observed patterns.
- Compare Rust with other concurrent languages.
- Refine and evaluate the manual analysis framework.