Patterns and Pitfalls: Empirical Insights into CAN Bus Bugs in Open-Source Repositories

Michael Gharbieh, Dr. Foyzul Hassan, Dr. Bruce Maxim
College of Engineering and Computer Science
University of Michigan-Dearborn

SURE EXP+ Undergraduate Research

INTRODUCTION

- CAN (Controller Area Network) protocol is the dominant intra-vehicle communications protocol in automotive applications[1].
- While the protocol itself is standardized by bodies like ISO[2] (International Standards Organization) and SAE[3] (Society of Automotive Engineers), development tools vary wildly.
- Lack of standard development tools can lead to confusion when working with different software vendor APIs (Application Protocol Interfaces) or ECU (Electronic Control Unit) Suppliers
- Resources to unify different development tools may increase productivity and foster innovation through market competition by increasing accessibility and ease of development for CAN systems

PURPOSE STATEMENT

The purpose of this project was to develop a static analysis tool to boost productivity for development and debugging of CAN related applications in various usage scenarios, irrespective of APIs or development tools used.

SKILL: TECHNOLOGY

- Developed new automation tools while still utilizing existing tools and methods to boost productivity.
- Studied existing development technologies to understand how they can be leveraged for use in future technologies.

METHODS

1. Open-source projects with the tag “can bus” were scraped from GitHub and GitLab.
2. Multi-level filtering was applied on the commits of collected repositories. First with automatic filtering, discarding anything not written in C or C++, then searching for commits that (attempted to) fix bug(s). Those results were then further filtered automatically to restrict to commits that contained CAN related fixes.
3. A random sample was applied on the remaining repositories for manual verification. The random sample was then categorized by the type of bug.
4. ASTs (Abstract Syntax Trees) were generated on the source code that the commits involved to be used for common pattern prediction.
5. The ASTs were examined to look for contextual elements that could be used to map parts of code to different patterns

SCAN HERE TO LEARN MORE **SCAN HERE TO LEARN MORE**

Results

- 7,017 repositories in total were scraped from GitHub and GitLab. Of the original repositories, 1,882 contained useful commits
- 250,902 total commits were analyzed with 39,361 commits being related to a bugfix (of any kind, not exclusive to CAN)
- From the 39,361 bugfix commits, 1,311 were bugfixes that dealt specifically with CAN implementation and/or development from 128 repositories
- A random sample of 300 commits was selected for manual verification, resulting in 95 different (relevant) bugs spanning 7 different categories and 4 subcategories (Fig. 2).
- A total of 122 ASTs were generated for use in pattern prediction

Conclusions

- Results suggest most CAN related issues fit into one of the 7 categories defined in Figure 2, indicating similar issues regardless of standardization.
- Preliminary examination of selected ASTs show similar contextual patterns for bugs of the same category.
- Contextual patterns indicate potential for static analysis tools to increase productivity and decrease development time.
- Future work is required to explore the viability of utilizing contextual trends with static analysis tools for bugs relating to CAN.

FIGURES

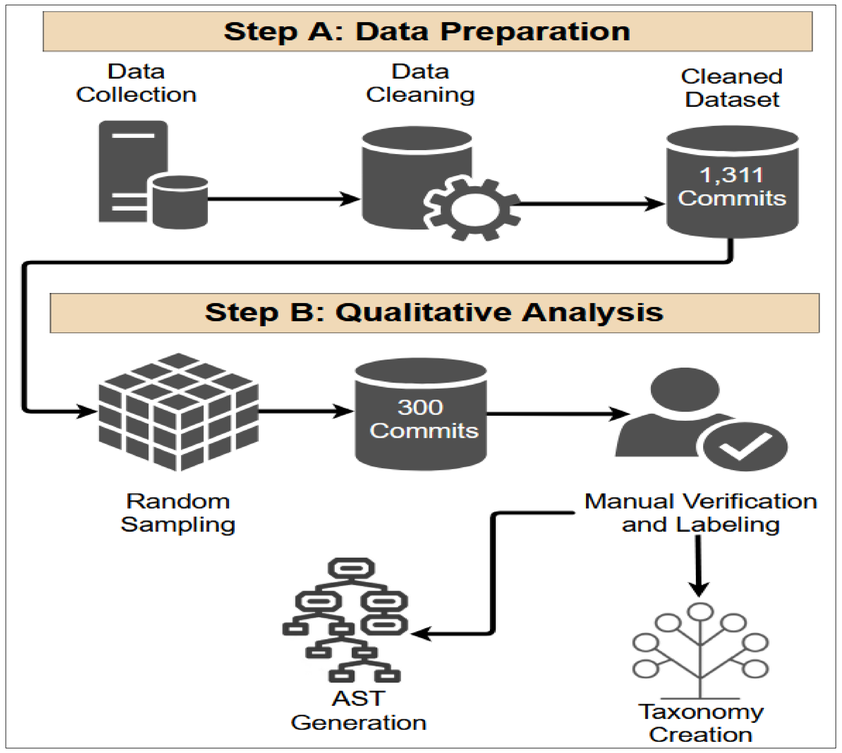

Figure 1: Approach Overview for Pattern Extraction

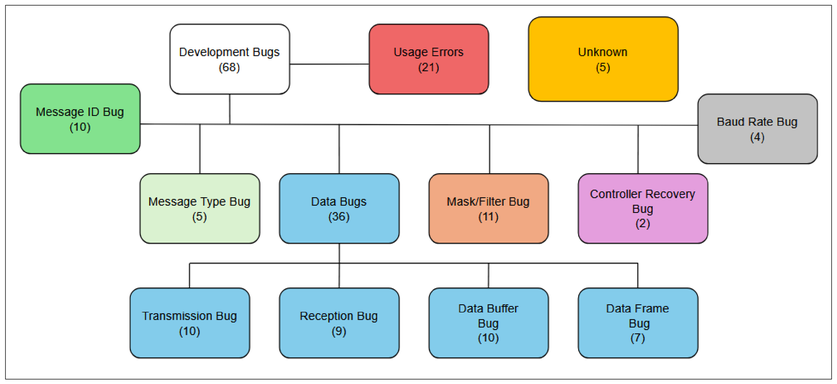

Figure 2: Taxonomy of bug categories with bug count per category

REFERENCES

1. https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/controller-area-network--can--overview.html?srsltid=AfmBOorKEd5Qlz_YtJ_5vTBd-zO9C4jf6WamtYoeP_-IHzGqsyo0bknL
2. https://www.iso.org/standard/86384.html#lifecycle
3. https://www.sae.org/publications/collections/content/j1939_dl/