McMaster University

ENGINEERING
Computing & Software

Investigation of Abbreviation Functions

Drasil – Generate All the Things!

Harmanpreet Singh Sagar, Dr. Jacques Carette, Dr. Spencer Smith
Department of Computing and Software, McMaster University, Hamilton, ON, Canada

Introduction

The Problem of Information Duplication:

- Prone to errors. [1]
- Hard to achieve traceability. [1]
- Reduces maintainability of software. [1]

The Solution:

- A software framework (**Drasil**) that captures data once and duplicates it as needed to generate all software artifacts, including code, requirements documentation, build scripts, etc. [1]

What is Drasil?

- A framework to generate all the software artifacts from a stable knowledge base. [1]
- Currently focuses on Scientific Software. [1]
- Common knowledge recipes are written in Domain Specific Languages (DSLs) embedded in Haskell. [1]
- Drasil has grown organically for almost 10 years and is now in need of refactoring.

Purpose / Objectives

- Analyze and resolve abbreviation function abuses in Drasil.
- Reduce duplication to prevent errors in Drasil.
- Improve traceability within the Drasil framework.
- The functions being analyzed are as follows:

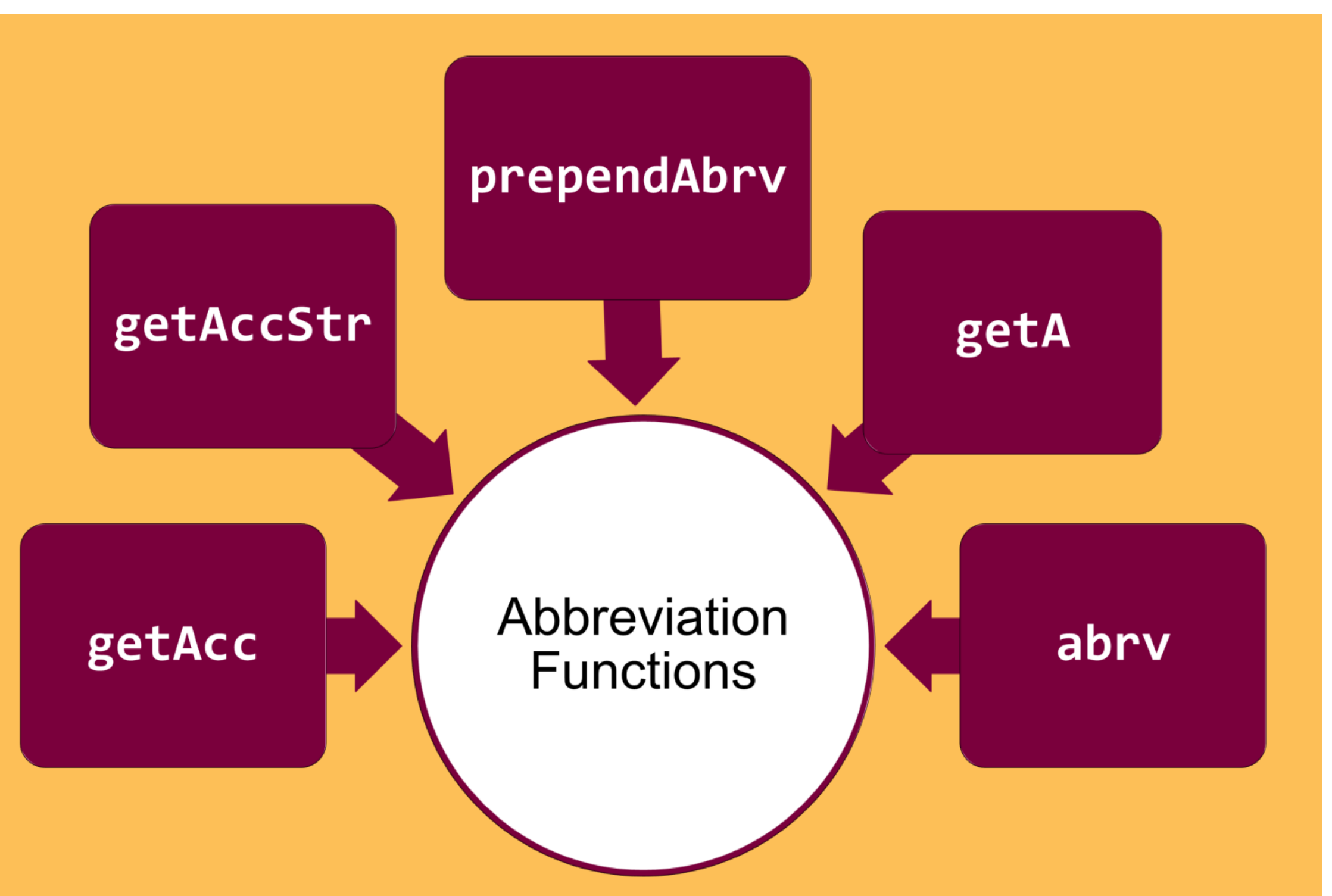


Figure 1: A mind map showing the various abbreviation functions in Drasil. (Source: Author's Own Creation)

Methods

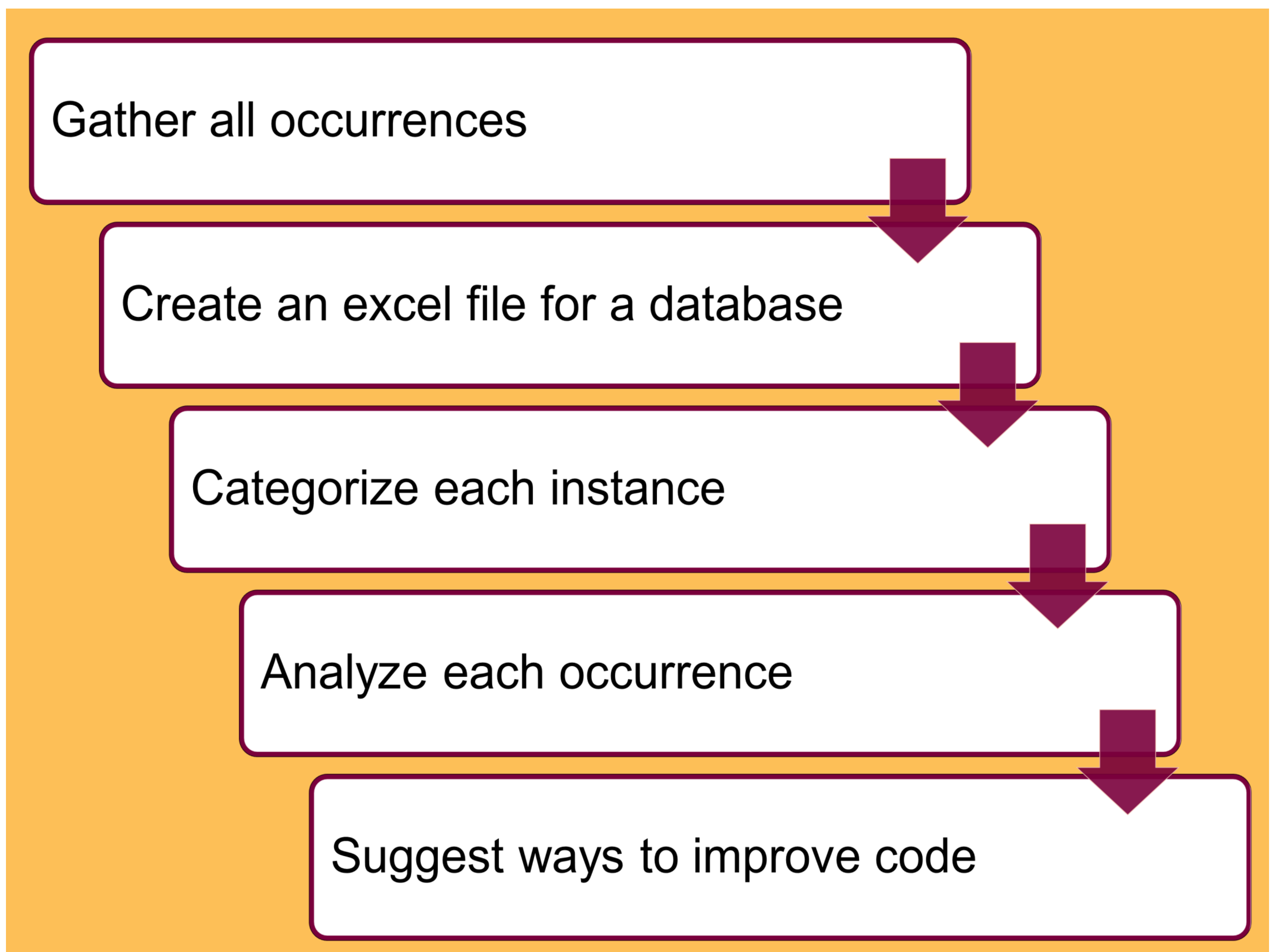


Figure 2: A flowchart showcasing the various steps taken to investigate the use of abbreviations within the Drasil framework. (Source: Author's Own Creation)

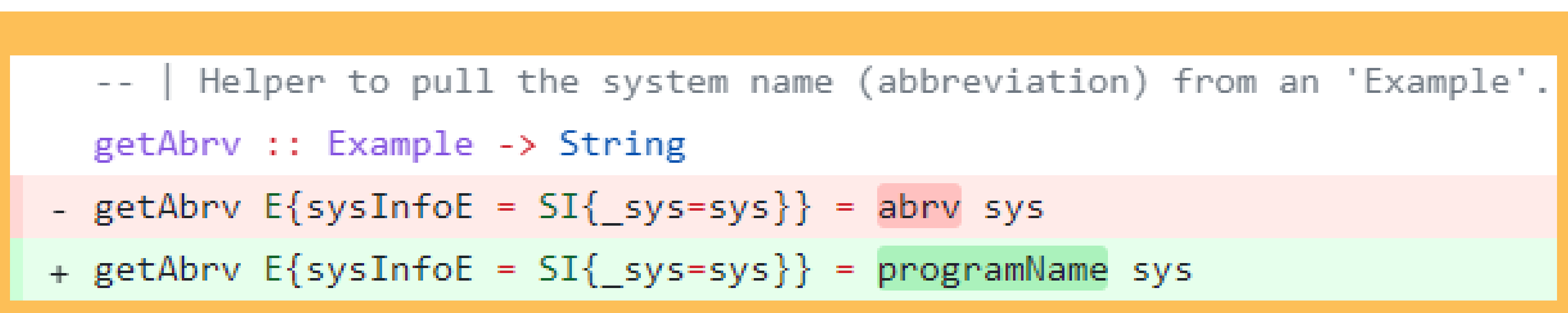
Methods Continued

Short Name	Long Name	Description
F	Fundamental	An acronym is required and cannot be replaced with the full word in the SRS
I	Incidental	The sentence would still make sense if the full phrase was used instead in SRS
U	Unknown	Unsure of the classification or of its purpose
T	Text	Used to form text
OI	Obtain Idea	Finds the Idea contained in x used to make y, and later is used to obtain acronym out of that idea.
OA	Obtain Acronym	Obtains the acronym from an Idea of a type
DF/RF	Domain / Reference Formation	Used in a function that is used to make a domain or a reference
OR	Obtain Reference	Used in a function that is used to obtain a reference.
NF	Name Formation	Used in a function that is used to make names

Figure 3: Table showcasing the various classifications. (Source: Author's Own Creation)

Results

- Abbreviation functions used in **146** lines of code
- No changes suggested for **`getA`** and **`getAcc`**.
- **`getAccStr`** was replaced with **`abrv`** due to same functionality.
- **`abrv`** replaced with **`programName`** in 19 different instances.
- Improved file naming by replacing spaces with underscores
  - `PD Controller` → `PD_Controller`.
- **`prependAbrv`** forms labels for `GenDefn`, `DataDefinitions`, `InstanceModel`, and `TheoryModel`. Further investigations are ongoing.
- Abbreviation functions misused as a side-effect of de-embedding the Drasil framework.
- **Refactored code** to improve traceability and documented findings in the Drasil wiki.



```
  -- | Helper to pull the system name (abbreviation) from an 'Example'.
  getAbrv :: Example -> String
- getAbrv E{sysInfoE = SI{_sys=sys}} = abrv sys
+ getAbrv E{sysInfoE = SI{_sys=sys}} = programName sys
```

Figure 4: A code block showcasing an example of a change made to the code base. (Source: Author's Own Creation)

Function	Location	Word	Classification
getAcc	drasil-docLang/lib/Drasil/Sections/Introduction.hs	Doc.SRS	F
getAcc	drasil-example/dblpendulum/lib/Drasil/DblPendulum/Assumptions.hs	twoD	F

Figure 5: A table showcasing different usages and classification of the **`getAcc`** function. (Source: Author's Own Creation)

Conclusion

This investigation highlighted the uses and abuses of abbreviation functions within Drasil. A lot of these abuses were a result of the de-embedding of Drasil. The misuse of these functions has been fixed to improve readability and traceability. Refactoring the code has also helped in following file naming conventions and has thus improved the quality of the artifacts that are generated by Drasil.

Next Steps

- Continue the investigation into the **`prependAbrv`** function to determine how it is used in reference formation.
- Refactor the code for the Drasil framework to effectively and efficiently utilize the **`prependAbrv`** function.
- Investigate functions that could be merged as a result of carrying out the same task.

Reference

[1] Daniel Szymczak, W. Spencer Smith, and Jacques Carette. Position paper: A knowledge-based approach to scientific software development. In *Proceedings of SE4Science'16 in conjunction with the International Conference on Software Engineering* (ICSE), Austin, Texas, United States, May 2016. In conjunction with ICSE 2016. 4 pp.

Acknowledgements

I would like to thank the McMaster Engineering Research Experience Award committee for funding this position and paper. I would also like to thank Dr. Spencer Smith and Dr. Jacques Carette for providing me with this valuable opportunity to learn and contribute to this project. Finally, I would like to thank Jason Balaci and Samuel Crawford for their support throughout the summer.

HARMANPREET SINGH SAGAR
Email: sagarh@mcmaster.ca
LinkedIn: harmanpreetssagar

View Our Work
https://jacquescarette.github.io/Drasil