The Programming Languages of Vim and Emacs: Advantages and Disadvantages

Long Ling

Abstract

Vim and *Emacs* are two legendary text editors that have gained a cult following for their extensive customization capabilities. A significant factor contributing to their flexibility and extensibility is the choice of programming languages for configuring the editors. This paper explores the differences between these two editors' configuration languages, namely *Emacs Lisp (Elisp)* and *Vimscript*, as well as *Lua*, the language used in modern versions of *Neovim (nvim)*. It compares their advantages and disadvantages in terms of configuring the editors and discusses how these languages have shaped the ecosystems surrounding Vim and Emacs.

ACM Reference Format:
Long Ling. 2018. The Programming Languages of Vim and Emacs: Advantages and Disadvantages. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

When it comes to configuring *Emacs* and *Vim*, three languages stand out: *Vimscript*, *Elisp*, and *Lua*. Each language has its own characteristics and considerations.

Vimscript is known for its lightweight nature, making it easy for programmers to configure. However, its lightweight nature also poses a challenge when it comes to comprehensive configuration. While it allows for quick and straightforward customization, achieving a fully-fledged configuration can be difficult.

On the other hand, *Elisp* supports extensive configuration capabilities, resembling that of an operating system. It provides a comprehensive set of tools for customization. However, one drawback of *Elisp* is its age, which makes it less approachable for modern programmers. Its syntax and concepts can sometimes be obscure and challenging to grasp, requiring a significant learning curve.

Lua, another language used for configuring *Vim*, offers a balance between ease of learning and comprehensive configuration support. It provides a simpler syntax and is relatively easy to pick up, making it accessible to a wide range of programmers. Additionally, *Lua* can handle more advanced configurations when needed. However, one downside of *Lua* is its tendency to be less stable and undergo frequent updates and iterations.

2 key idea

2.1 Emacs Lisp (Elisp)

EmacsLisp, a dialect of *Lisp*, is the primary configuration language for *Emacs*.You can find detailed information in the following website:https://www.gnu.org/software/emacs/manual/html_node/eintr/

2.1.1 Extensive Configuration Capabilities. *Elisp*, the Emacs Lisp language, is specifically designed for configuring and extending the Emacs text editor. It provides a wide range of features and APIs that allow users to customize almost every aspect of Emacs, making it highly versatile and powerful for comprehensive configuration. Its capabilities go beyond simple customization and enable users to create complex workflows, define new commands, and extend Emacs functionalities to suit their specific needs.

2.1.2 System-like Configuration. *Elisp*'s configurability resembles that of an operating system, where users can define their own commands, keybindings, modes, and even create custom packages. This system-like configuration capability gives Emacs users unparalleled control over their editing environment, making it a favored choice among power users and developers who require extensive customization options.

2.1.3 Learning Curve for Modern Programmers. *Elisp*'s age and historical context can make it challenging for modern programmers to learn. Its syntax, concepts, and programming style may differ significantly from more modern programming languages, making it less intuitive for newcomers. The initial learning curve can be steep, requiring programmers to familiarize themselves with Emacs-specific conventions and functional programming concepts.

2.1.4 Obscurity and Complexity. *Elisp*'s extensive capabilities and flexibility can sometimes lead to complex and obscure code. Emacs, being a highly extensible editor, allows

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

users to modify its core behavior extensively. This flexibility, combined with the historical accumulation of features and customizations, can result in codebases that are difficult to understand and maintain, especially for programmers not well-versed in Elisp.

2.2 Vimscript

Vimscript, a scripting language specifically designed for *Vim*, is examined in this section.You can find the detailed information for *Vimscript* in the following website:https://learnvimscriptthehardway.stevelosh.com/

2.2.1 Simplicity and Minimalism. *Vimscript* is designed to be simple and minimalist, focusing on providing basic functionality and customization options. Its lightweight nature allows programmers to quickly grasp the language and make simple configurations without much complexity.

2.2.2 Lack of Advanced Data Structures. *Vimscript* lacks advanced data structures like arrays, dictionaries, or objects that are commonly found in other programming languages. This limitation makes it challenging to organize and manipulate data in a more comprehensive and efficient manner, hindering complex configurations that require sophisticated data handling.

2.2.3 Limited Control Flow Constructs. *Vimscript* has a relatively basic set of control flow constructs, such as conditionals and loops. While they are sufficient for simple configurations, they may become less expressive and restrictive when attempting to achieve more intricate and comprehensive configurations.

2.2.4 Modularity and Scripting. *Vimscript* encourages modularity by allowing the creation of small, reusable scripts called "vimrc" files. This modularity makes it easy for programmers to configure specific aspects of Vim to their liking. However, combining multiple configurations into a cohesive and comprehensive setup can be challenging due to the lack of a standardized module system or dependency management.

2.2.5 Lack of Standard Libraries. Unlike many programming languages that come with extensive standard libraries, *Vimscript* has a limited set of built-in functions and commands. This lack of comprehensive standard libraries can make it difficult to handle complex tasks, requiring programmers to either rely on external plugins or write custom functions from scratch.

2.3 Lua in Neovim

Lua, a configuring language in *neovim*.You can find detailed information in the following website: https://www.lua.org/

2.3.1 Simplicity and Ease of Learning. *Lua* is designed to be a lightweight and easy-to-learn scripting language. It has a clean and concise syntax, with a minimalistic set of features and a small core API. Its simplicity makes it accessible to programmers of various skill levels, including beginners, who can quickly grasp the language and start configuring applications.

2.3.2 Versatile Configuration Capabilities. *Lua*'s flexibility allows it to be used for a wide range of configurations, including in applications like text editors (such as Vim). It provides mechanisms for defining custom behaviors, extending functionality, and configuring various aspects of the application. *Lua*'s lightweight nature and straightforward syntax make it well-suited for customization tasks, allowing developers to tailor the configuration to their specific needs.

2.3.3 Instability and Rapid Updates. *Lua* has a reputation for being a dynamic language that undergoes frequent updates and iterations. The *Lua* development team actively introduces new features, enhancements, and bug fixes, which can lead to changes in the language and its ecosystem. While these updates improve the language's capabilities and address issues, they can also introduce compatibility concerns and require developers to adapt their configurations when migrating to newer versions.

2.3.4 Evolution of Lua Versions. *Lua* has evolved through different major versions, such as *Lua*5.1, *Lua*5.2, *Lua*5.3, and *Lua*5.4. Each version introduces new features and improvements, but it also brings potential incompatibilities with previous versions. This evolution, while necessary for progress, can make it challenging for developers to maintain and update existing *Lua* configurations, especially if they heavily rely on specific features or libraries that may have changed.

2.3.5 Community and Ecosystem Dynamics. *Lua* has a vibrant community that actively contributes to its development, libraries, and frameworks. However, this dynamic nature can result in a diverse and sometimes fragmented ecosystem. Different *Lua* libraries may have different levels of compatibility with various *Lua* versions or specific configurations. This fragmentation can make it harder to ensure stability and consistency when building and maintaining comprehensive *Lua* configurations

3 Conclusion

In summary, this paper explores the configuration languages used in *Vim*, *Emacs*, and *Neovim*. It compares *EmacsLisp*(*Elisp*), *Vimscript*, and *Lua* in terms of their advantages and disadvantages for configuring the respective editors. The paper also discusses the impact of these languages on the ecosystems surrounding *Vim* and *Emacs*.