INTERFACE TECNOLÓGICA

EFFICIENCY IN WRITING SOFTWARE WITH VIM

EFICIÊNCIA NA ESCRITA DE SOFTWARE COM O VIM

Bruno Coimbra de Oliveira – b-coimbra@homail.com
Faculdade de Tecnologia (Fatec) – Taquaritinga – SP – Brasil

Jederson Donizete Zuchi – jederson.zuchi@fatec.sp.edu.br
Faculdade de Tecnologia (Fatec) – Taquaritinga – SP – Brasil

DOI: 10.31510/infa.v17i2.1066
Data de publicação: 18/12/2020

ABSTRACT

Modal editing is a, nowadays rare, concept that offers multiple interaction modes which are optimized for specific types of actions and interactions within a text editor. This study aimed to provide insight on the pros and cons of using a nowadays considered “outdated” piece of software called *Vim* for daily usage, and how its rationale helps its users to realize their objectives in an efficient and powerful manner. Furthermore, a test case with several subjects was put into place to gauge exactly how *Vim* compares to a standard editor on a particular scenario with timed comparison, following up on an analysis that was made based on the results of that experiment. Ultimately, it was concluded that, even for an old-fashioned tool, when certain criteria were met, *Vim* still posed as a strong contender for automating repetitive tasks, performing considerably faster than its adversaries. This experiment also served as a motive to demonstrate what might be gained from a tool with such a difficult learning curve.

Keywords: Text editors. Modal Editing. Vim. IDE.

RESUMO

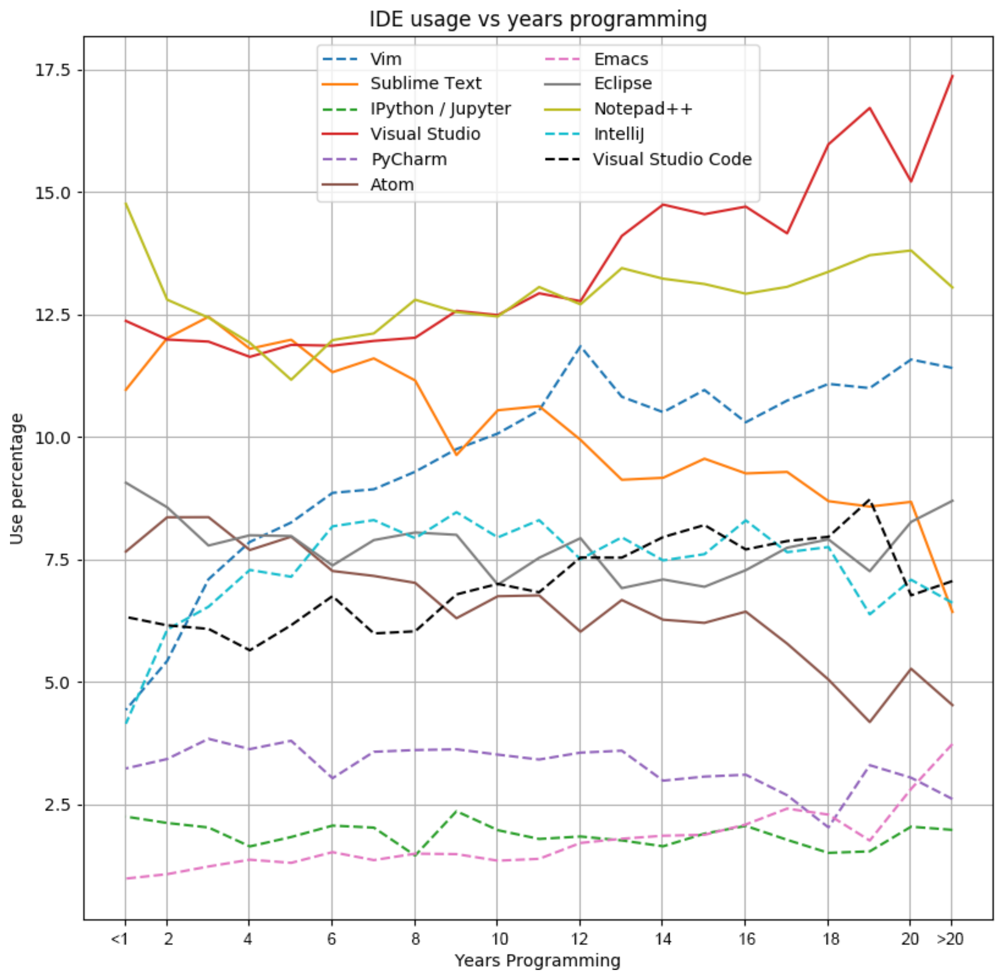
A edição modal é um conceito, hoje raro, que oferece vários modos de interação que são otimizados para tipos específicos de ações e interações dentro de um editor de texto. Este estudo teve como objetivo fornecer uma visão sobre os prós e contras do uso de um software hoje considerado “desatualizado” chamado Vim para uso diário, e como sua filosofia ajuda seus usuários a realizarem seus objetivos de maneira eficiente e poderosa. Além disso, um experimento com vários indivíduos foi conduzido para avaliar exatamente como o Vim se compara a um editor padrão em um cenário específico com comparação cronometrada, seguindo uma análise que foi feita com base nos resultados desse experimento. Por fim, concluiu-se que, mesmo para uma ferramenta antiquada, quando certos critérios eram atendidos, o Vim ainda se apresentava como um forte candidato à automação de tarefas repetitivas, com desempenho consideravelmente mais rápido que seus adversários. Esse experimento também serviu como razão para demonstrar o que pode ser ganho com uma ferramenta que possui uma curva de aprendizado difícil.

Keywords: Editores de Texto. Edição Modal. Vim. IDE.

1 INTRODUCTION

With the surge of new text editing software, their latest, modern features brought a decline to the popularity of lesser-known text editors, including “modal” editors, derivatives of the *Vi* editor et al. Although the concept of a modal editor was not completely diminished, according to *Stack Overflow*'s yearly survey (*Developer Survey*), the data shows that *Vim* has been hovering between the mark of 20% to 25% in terms of popularity when compared to other text editors such as *Notepad++* and *Visual Studio Code*, which rated at around 30% to 50% respectively (DEVELOPER SURVEY RESULTS, 2019).

Figure 1 - Graph comparing the popularity relevance of several text editors over the span of 20 years.



Source: Stack Overflow’s Developer Survey (2017).

The appeal that has made *Vim* stuck around for nearly 30 years since its initial release stems from its origins, and most importantly, from its distinct philosophy when compared to standard editors. Although it looks difficult to learn at first, there are only two main ideas that need to be grasped to really understand it – they are "modal editing" and operators. Ultimately, the purpose of *Vim* is to enable you to edit text effectively (ILIC, 2017). With that in mind, *Vim* does not operate in the process of inserting text per se, but instead, it mainly operates on editing and navigating through text. According to Ilic (2017, p. 16), when programmers are writing code, they likely are not spending most of their time typing, but rather moving around through existing code and editing them.

Nowadays, most developers are content with writing code in their *modeless* text editors, or are oblivious to an alternative way to edit text. The term *modeless* refers to the fact that each interface element has only one function, whereas a modal interface performs a different action based on context (OSIPOV, 2018). In itself, developers utilizing *modeless* text editors as their main tool for writing code are not frowned upon. Nonetheless, as McDonnell (2014, p. 1) states: "How you go about writing text is up to you, but the tools you use will determine how efficient you are at carrying out certain tasks. The more efficient you are, the quicker you'll complete that task [...]".

The reason why modal editing is so neglected is seemingly due to a big compromise for some developers, which is the exclusive use of the keyboard instead of the traditional conjunction with the keyboard and mouse (THE VALUABLE DEV, 2019). Although the idea of limiting yourself to the keyboard might seem inconvenient, this article will strive to prove it otherwise – whilst encompassing some of the main characteristics that make *Vim* such a unique tool, and ultimately, what makes it so powerful once mastered.

2 TEXT EDITORS AND WORD PROCESSORS

There are many different circumstances under which one will have to work with text. As such, it is important to recognize the uses for which different text-based pieces of software are intended. These programs generally fall under two categories: word processors and text editors (IDRH, 2014).

2.1 Text Editors

As Finseth (1991, p. 13) defines it: “In its most general form, text editing is the process of taking some input, changing it, and producing some output.”, in that regard, a text editor is simply a medium between sending input to the computer and that input being shown on the screen. Moreso, as the name implies, it offers a plethora of tools for the sole purpose of editing text. As with most text editors, they come with the most basic tools such as copying, replacing and “cutting” text. These are the main components of editing text.

Additionally, as (COTTRELL, 1999) describes it: “A text editor has the usual apparatus of pull-down menus and/or clickable icons for functions like opening and saving files, searching and replacing, checking spelling and so on. But it has no typesetting functionality.”, this statement makes it clear that text editors are inept of creating visual representations of text beyond *syntax highlighting* and making use of the features that a font provides (such as making characters appear in bold, italic, underlined etc.), which he then further adds: “The text you type in text editors appears on screen in a clear visual representation, but with no pretense at representing the final printed appearance of the document”.

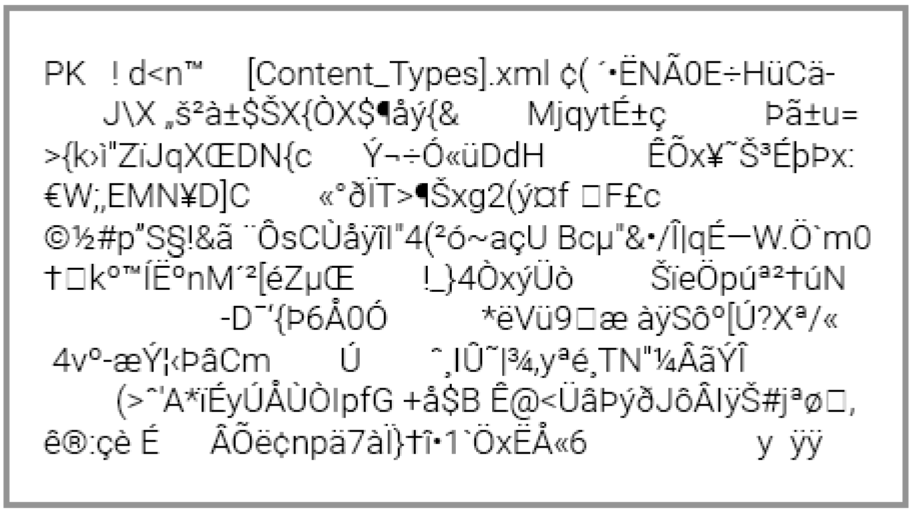
2.2 Word Processors

A word processor, on the other hand, is essentially any program capable of formatting text and other types of media (e.g. images) for physical or electronic printing. In contrast to a text editor, a word processor gives the user extensive control over the visual qualities of the document (IDRH, 2014).

To summarize, despite the fact that text editors and word processors are used for distinct purposes, they are similar in concept – both are tools for manipulating characters on the screen. Typically, writing code and writing english words are quite different activities. Compared to writing paragraphs to formulate sentences, when programming, you spend more time reading, switching files, navigating and editing code. This validates why different types of programs for writing code versus English words exist (e.g. Visual Studio Code versus Microsoft Word).

Furthermore, to demonstrate the distinction between a file created by a text editor and a word processor on a rudimentary level, below is a shortened view of the contents of a file created by a word processor once opened by an arbitrary text editor:

Figure 2 - Example of the contents of a Word document opened with a text editor.



```
PK  ! d<n™    [Content_Types].xml ¢( ´•ËNÃ0E÷HüCä-
     J\X „š²à±$ŠX{ÒX$¶åý{&      MjqytÉ±ç        Þã±u=
>{k›ì"ZïJqXŒDN{c     Ý¬÷Ó«üDdH          ÊÕx¥˜Š³ÉþÞx:
€W;,EMN¥D]C       «°ðÏT>¶Šxg2(ý¤f □F£c
©½#p"S§!&ã ¨ÔsCÙåÿîl"4(²ó~açU Bcµ"&•/Î|qÉ—W.Ö`m0
†□k°™ÍË°nM´²[éZµŒ       !_}4ÒxýÜò       ŠïeÖpúª²†úN
           -D¯'{Þ6Å0Ó         *ëVü9□æ àÿSô°[Ú?Xª/«
 4vº-æÝ¦‹ÞâCm       Ú       ˆ¸IÛ˜|¾,yªé¸TN"¼ÂãÝÎ
      (>ˆ'A*ïÉyÚÅÙÒIpfG +å$B Ê@<ÜâÞýðJôÂlÿŠ#jªø□,
ê®:çè É     ÂÕëçnpä7àÏ}†î•1`ÖxËÅ«6               y  ÿÿ
```

Source: IDRH (2014).

The figure above shows that while the contents of a file created by a text editor will appear the same to other text editors, a document created by a word processor contains special metadata used for representing the visual formatting of the text contents (COTTRELL, 1999).

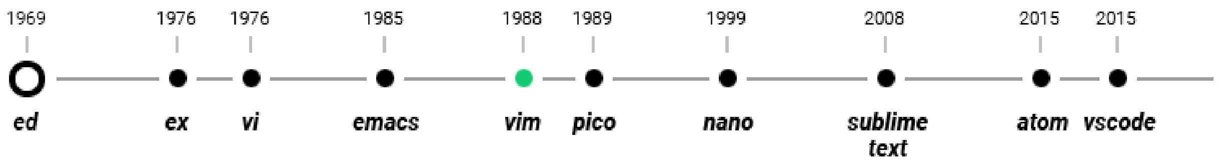
3 BRIEF HISTORY OF TEXT EDITORS

Preceding the invention of the *Vi* editor or any other screen editor for that matter, people communicated with computers on printing terminals, which worked rather primitively compared to nowadays' screens. These environments were once the norm for interactive computing, and it gave way to the existence of the so-called "line editors". Since line numbers were a way to discern a part of a file to be worked on – line editors evolved to edit those files (LAMB, 1994, ch. 5). Those line editors were available in closely identical form on nearly every UNIX system, which reaffirms (KLEENE, 2020)'s assertion that "[...] text files are the *lingua franca* of computers", despite having no practical use nowadays (unless support for a screen editor on a particular terminal is lacking), line editors are considered to be a relic of an earlier age in computing.

Subsequently, it took several years of incremental improvements for modern editors such as we have today to come to fruition. To understand how they came to be, it is vital to understand where they came from and how they evolved throughout the decades (KING, 2020).

Illustrated below is an exemplified timeline of the evolution of text editors (N.B. many editors were omitted since they branched from *vi*, but had relatively low popularity relevance).

Figure 3 - Timeline of popular text editors.



Source: Author (2020).

4 THE VI EDITOR

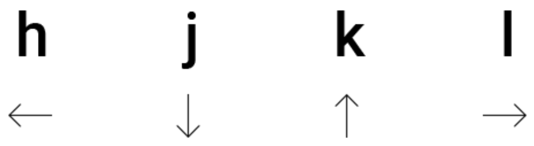
The *vi* editor is the original root of the *Vim* family tree. It was created by Bill Joy in 1976 for an earlier version of the BSD (Berkeley Software Distribution) operating system. The name “vi” is an abbreviation of “visual in ex”, which, as the name indicates, was an extension of the most common editor back then, the “ex” editor. At the time, *vi* was just a command that started the *ex* editor in one of its modes – the visual mode (SCHULZ, 2007, p. 8).

Vi was one of the first editors to introduce the concept of modality – which editors like *Vim* have inherited. It has several limitations, however, that its derivatives have aimed to solve. According to (SCOSALES, 1991), for the early versions of *vi*, those limitations were as follows:

- The inability to edit binary files (such as compiled programs);
- Only supporting a maximum line length of 1024 characters;
- Having a maximum file size limit of 250,000 lines at most;
- Maximum of 14 characters in a filename;

On the other hand, these limitations were put into place due to the fact that *vi* was developed at a time when Unix systems were considerably less stable than they are today, with severe constraints such as limited system resources, slow networks, no digital displays and no mouse. The *vi* user of yesteryear had to be prepared for the system to crash at arbitrary times (ROBBINS, p. 6, 2008). Due to the hardware limitations posed by those early machines, *vi* was designed to fit those needs accordingly. Nowadays, we can still find relics of those design decisions in *vi*, such as the navigation keys (CASSEL, 2020).

Figure 4 - Directional keys of the *Vim* editor.



Source: Author (2020).

The letters shown in the figure above, for instance, were used for navigation keys instead of the conventional cursor keys for the reason that Bill Joy (the creator of *vi*) used a video terminal with built-in keyboard with no cursor keys to develop *vi*. This also accounted for why a single-letter key on the keyboard performed various actions, as *vi* lacked support for a mouse (CASSEL, 2020).

All things considered, Bill Joy considers *vi* a tool that was designed for a world that is now extinct. According to him, those decisions were not exactly superior to what we use today, just a necessity of that particular time period, as stated by him in an interview: "People don't know that *vi* was written for a world that doesn't exist anymore." (CASSEL, 2020).

5 THE VIM EDITOR

The *Vim* editor is one of the many derivatives of the original "*vi*" editor (SCHULZ, 2007, p. 8). It was created by Bram Moleenar in 1988 by using the existing STEVIE *vi* clone as a starting point, and is broadly considered to be almost a proper superset of *vi*. Therefore, everything that is in *vi* is available in *Vim,* hence why it stands for "Vi IMproved" (TWO-BIT HISTORY, 2018).

Vim earned its place as the most popular editor amid the Linux community back in 2006 (TWO-BIT HISTORY, 2018). As of now, according to Stack Overflow's 2020 Developer Survey, *Vim* is the most popular text-mode (i.e. terminal emulator) editor, particularly amongst 40% of SysAdmin/DevOps people, and 25.8% for "average" software developers (DEVELOPER SURVEY RESULTS, 2020).

As with most things, an effort must be placed to achieve the desired knowledge. This motto is especially true for *Vim*'s learning curve, as it heavily relies on muscle memory and touch typing abilities (the ability to use muscle memory to find keys fast, without using the sense of sight), as (NEIL, 2012, p. 1) prefaces his book: "*Vim* traces its ancestry back to the classic Unix editors. These predate the mouse and all of the point-and-click interfaces that came with it. In *Vim*, everything can be done with the keyboard. For the touch typist, that means *Vim* does everything faster".

On the other hand, this dependence on muscle memory excludes a substantial amount of potential users. The average productivity levels of a person tend to go down once when they are introduced to a new tool that replaces their previous one. The reason for this, according to (ILIC, p. 11-14, 2017) is that they have to relearn a whole slew of new key-bindings and memorize where things are located inside the program from scratch, essentially rebuilding their muscle memory from anew. In contrast to the previous statement, Bram Moleenar, the creator of *Vim* himself, analogically wrote: “Learning to drive a car takes effort. Is that a reason to keep driving your bicycle? No, you realize you need to invest time to learn a skill. Text editing isn’t different. You need to learn new commands and turn them into a habit” (ILIC, p. 11, 2017).

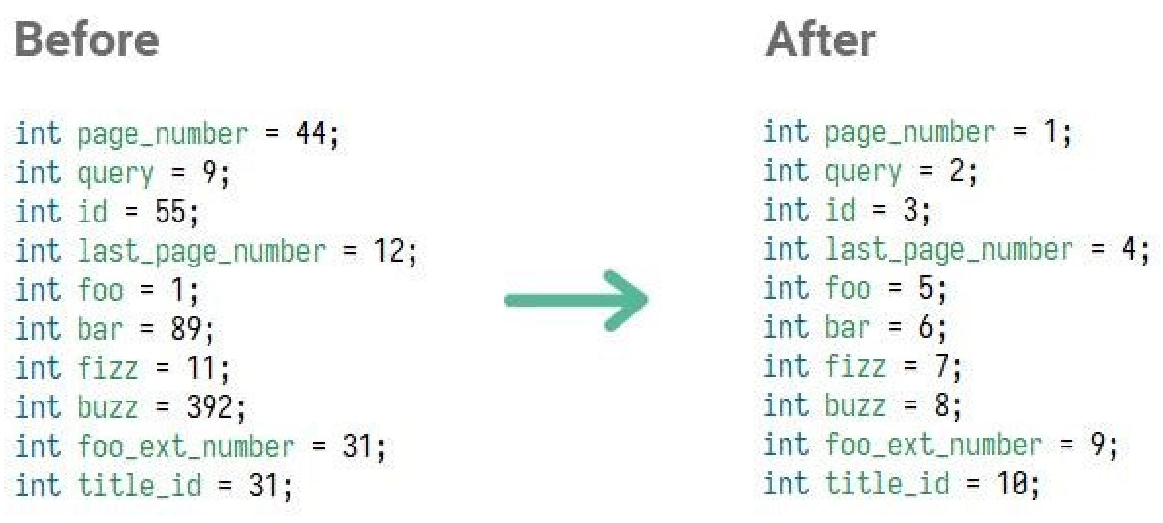
Vim is often described as a “language for text editing”, similarly to how a programming language would work. This similarity is observed on how the vast number of very specific commands can be combined in useful ways to cause an action to occur (MOSER, 2015). NEIL, Drew (p. 18, 2012) compares this mechanism to playing chords, notes and melodies on a piano. Even more, there is an element of strategic and intentional placing of those commands on the keyboard, all with the goal of accomplishing the greatest amount of work with the least number of keystrokes (MOSER, 2015).

6 METHOD

For the sake of providing a solid comparison between *Vim* and other text editors, an experiment was conducted with five anonymous subjects, all of which are software developers with roughly the same experience. The data collected from the results of the experiment was used to study the effectiveness of using *Vim* as the primary tool for editing text from a software developer standpoint. Furthermore, to better understand how the results were achieved for *Vim*’s case, this given study was primarily based on the bibliographic review of books, articles and online surveys (see Figure 1).

The subjects were asked to refactor a file with a list of unordered numbers, without changing the order of the lines, into an ordered list (increasing), as illustrated by the image below.

Figure 5 - Scenario taken by the subjects.



Before

```
int page_number = 44;
int query = 9;
int id = 55;
int last_page_number = 12;
int foo = 1;
int bar = 89;
int fizz = 11;
int buzz = 392;
int foo_ext_number = 31;
int title_id = 31;
```

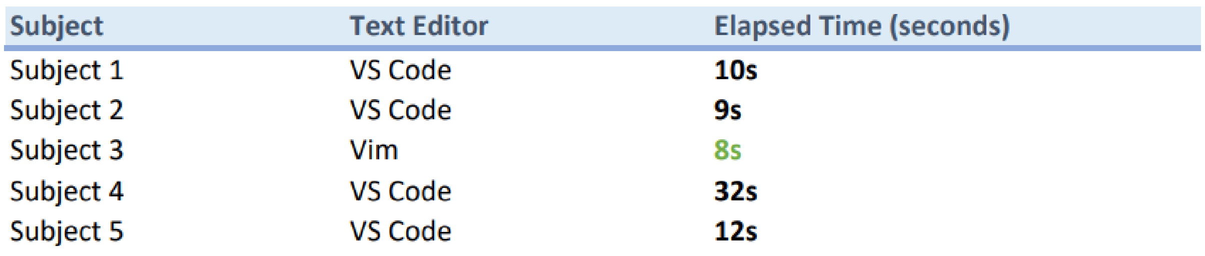
After

```
int page_number = 1;
int query = 2;
int id = 3;
int last_page_number = 4;
int foo = 5;
int bar = 6;
int fizz = 7;
int buzz = 8;
int foo_ext_number = 9;
int title_id = 10;
```

Source: Author (2020).

After the subjects performed the scenario above, the results were as follows:

Figure 6 - Test case results.



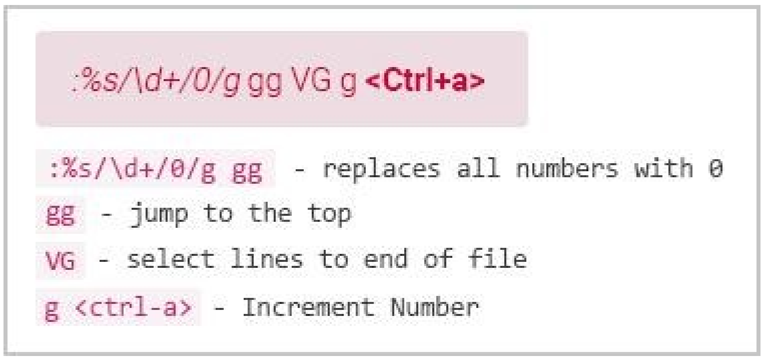
Subject	Text Editor	Elapsed Time (seconds)
Subject 1	VS Code	**10s**
Subject 2	VS Code	**9s**
Subject 3	Vim	**8s**
Subject 4	VS Code	**32s**
Subject 5	VS Code	**12s**

Source: Author (2020).

Upon analysis of the results, the discrepancy between the results from standard editors and the *Vim* subject is evident. The sample used for this test case is considerably small, however, considering that in real world scenarios there would be a need to refactor even larger file sizes, the elapsed time would grow exponentially for the standard editors, whereas for *Vim*’s case, it would remain constant.

Additionally, the series of keystrokes performed by the *Vim* subject can be viewed in the image below.

Figure 7 - Commands used by the *Vim* editor to reorder a list of numbers.



```
:%s/\d+/0/g gg VG g <Ctrl+a>

:%s/\d+/0/g gg  - replaces all numbers with 0
gg  - jump to the top
VG  - select lines to end of file
g <ctrl-a>  - Increment Number
```

Source: Author

7 CONCLUSION

An ordinary programmer will come across many different tools to aid him during his career. Plenty of those tools will be aimed at manipulating code, such as text editors and integrated development environments (IDEs). According to (ZUKERMAN, 2011) we often reject ancient pieces of software with the mindset that “newer is better”, even though they are still actively used by plenty of users. If we take a look at how Microsoft Word (first released in 1990) progressed over the decades, we can observe that it has not degraded or remained stale functionality-wise throughout the years, but instead, that it has evolved due to the maturity that it possesses. The same can be said about *Vim*, it is based on a much older editor called *Vi*, and continues to preserve its longevity with the help of its devout community, ensuring that *Vim* continues to catch up with modern technologies for manipulating text in the future.

To conclude, given that one of the core roles of a software developer is to write code, text editing itself poses a big importance in their lives, thus, it is worth turning a critical eye toward how well our tools facilitate this specific activity.

REFERENCES

CASSEL, David. **A Look at Vim, a Text Editor for the Ages**. Available at: <https://thenewstack.io/a-look-at-vim-a-text-editor-for-the-ages/>. Accessed in: aug. 31st, 2020.

CERUZZI, Paul E. **A History of Modern Computing.** 2nd ed. London, UK: The MIT Press, p. 15, p. 47, apr. 2003.

COTTRELL, Allin. **Word Processors**. jun. 29th, 1999. Available at: <http://ricardo.ecn.wfu.edu/~cottrell/wp.htm>. Accessed in: sep. 17th, 2020.

DEVELOPER SURVEY RESULTS 2017. Stack Overflow. Available at: <https://insights.stackoverflow.com/survey/2017>. Accessed in: sep. 13th, 2019.

DEVELOPER SURVEY RESULTS 2019. Stack Overflow. Available at: <https://insights.stackoverflow.com/survey/2019>. Accessed in: jul. 27th, 2019.

FINSETH, A. Graig. **The Craft of Text Editing:** A Cookbook for an Emacs. St. Paul Minnesota: Springer-Verlag & Co, feb. 1991.

GNU. **The GNU ed line editor.** feb. 20th, 2020. Available at: <http://www.gnu.org/software/ed/manual/ed_manual.html>. Accessed in: oct. 6th, 2020.

HINZ, Marco; BROWN, George; et al. **Vim Galore**. Everything you need to know about Vim. Available at: <https://github.com/mhinz/vim-galore/>. Accessed in: sep. 3rd, 2020.

IDRH. **Text Editors and Word Processors**. may 20th, 2014. Available at: <https://idrh.ku.edu/text-editors-and-word-processors>. Accessed in: sep. 15th, 2020.

ILIC, Jovica. **Mastering Vim Quickly.** Berlin, Germany: Jolesoft, p. 11-16, nov. 2017.

KING, Brian. **The History of Modern Text Editors.** jul. 20th, 2020. Available at: <https://blog.grio.com/2020/07/the-history-of-modern-text-editors.html>. Accessed in: sep. 24th 2020.

KLEENE, Robin. **The Era of Visual Studio Code**. sep. 21st, 2020. Available at: <https://blog.robenkleene.com/2020/09/21/the-era-of-visual-studio-code/>. Accessed in: sep. 25th, 2020.

LAMB, Linda. **Learning the vi Editor.** 5th ed. aug 1994. Available at: <https://www.cs.ait.ac.th/~on/O/oreilly/unix/vi/index.htm>. Accessed in: oct. 6th, 2020.

MCDONNELL, Mark. **Pro Vim**. 1st ed. Apress, nov. 2014.

MOSER, R. Gary. **Some thoughts on Vim**. mar. 17th, 2015. Available at: <https://blog.revolutionanalytics.com/2015/03/some-thoughts-on-vim.html>. Accessed in: oct. 10th, 2020.

NEIL, Drew; POPE, Tim; KEPPLER, Kay. **Practical Vim:** Edit Text at the Speed of Thought. Dallas, Texas: The Pragmatic Bookshelf, oct. 2012.

ORST, Andrey. **Text Editors.** Available at: <https://andreyorst.gitlab.io/posts/2020-04-29-text-editors/>. Accessed in: aug. 24th 2020.

OSIPOV, Ruslan. **Mastering Vim:** Build a software environment with Vim and Neovim. Birmingham, UK: Packt Publishing Ltd, p. 6-7, nov. 2018.

PETERSON, Mickey. **Mastering Emacs**. Blurb Incorporated, p. 13-16, may 23rd, 2015.

ROBBINS, Arnold; HANNAB, Elbert; LAMB, Linda. **Learning the VI and VIM Editors:** Text Processing at Maximum Speed and Power. 7th ed. Sebastopol, CA: O'Reilly Media, Inc., jul 2008.

SCHULZ, Kim. **Hacking Vim**: A cookbook to get the most out of the latest Vim editor. Birmingham, UK: Packt Publishing Ltd, may 2007.

SCOSALES. **The editor vi and its limitations.** 1992. Available at: <https://www.scosales.com/ta/kb/103472.html>. Accessed in: mep. 2nd, 2020.

THE VALUABLE DEV. **Is Vim Really Not For You? A Beginner's Guide.** oct. 28th 2019. Available at: <https://thevaluable.dev/vim-for-beginners/>. Accessed in: sep. 3rd, 2020.

TWO-BIT HISTORY. **Where Vim Came From.** aug. 5th, 2018. Available at: <https://twobithistory.org/2018/08/05/where-vim-came-from.html>. Accessed in: sep. 14th, 2020.

WILLIAMS, Al. **Editor Wars**. jul. 26th, 2016. Available at: <https://hackaday.com/2016/07/26/editor-wars/>. Accessed in: oct. 7th, 2020.

WINCENT. **Modal Editor**. mar. 12nd, 2020. Available at: <https://wincent.com/wiki/Modal_editor>. Accessed in: oct. 7th, 2020.

ZUKERMAN, Erez. **The Top 7 Reasons To Give The Vim Text Editor A Chance.** may 12th, 2011. Available at: <https://www.makeuseof.com/tag/top-7-reasons-to-give-the-vim-text-editor-a-chance/>. Accessed in: oct. 13rd, 2020.