Innovations In Software Configuration: Introducing A Data Comparison Tool Based On The Myers Diff Algorithm

Lorentzo Augustino [1], Adhi Kusnadi [2*], Marlinda Vasty Overbeek [3]

[1,2,3] Informatic Department, Faculty Engineering and Informatic,
Universitas Multimedia Nusantara, Tangerang, Indonesia
[*]Corresponding Author:
Email: adhi.kusnadi@umn.ac.id

Abstract.

In the realm of software project development, it's not uncommon for changes to emerge at various stages of a project's lifecycle. Such alterations can manifest in virtually every facet of the software development process, from conceptual design decisions down to the minutiae of the source code. To proficiently manage and track these dynamic changes, professionals turn to a specialized toolset known as Software Configuration Management (SCM). One of the standout features that SCM tools bring to the table is the 'diff' capability. This functionality allows developers to identify and review the disparities between two versions of source code. Recognizing the importance and utility of this feature, the primary objective of this research is to create an advanced diff application. This application, by leveraging the Myers Diff algorithm, is meticulously designed to pinpoint and showcase differences in characters between two sets of text-based data. Moreover, it accentuates these differences by visually highlighting the contrasting characters between the two datasets. To ensure the reliability and accuracy of this newly developed tool, we undertook a series of validation tests. We juxtaposed the results from our application against those from a comparable existing tool. Impressively, the discrepancies in results were minimal, with a marginal difference of just 1%. This suggests not only the utility but also the precision of our application in real-world software development scenarios.

Keywords: *Data Comparison Tools, Diff, Myers Diff, and Software ConfigurationManagement (SCM).*

I. INTRODUCTION

Software development, by its intrinsic nature, is a dynamic endeavor. Regardless of the specific phase of the software project, changes and modifications are not just probable but are often a certainty. As [1] astutely observed, the propensity to introduce changes to systems remains consistent throughout the software's developmental trajectory. Such modifications can span a diverse spectrum, including but not limited to shifts in business processes, adjustments to technical requirements, and even wholesale transformations in software models and their foundational source code.Typically, the collaborative nature of software projects, often involving teams comprising two or more developers, brings to the fore the imperative need for a meticulous mechanism to chronicle and monitor the sea of changes happening within the project. The primary motivation behind such a mechanism isn't just to maintain transparency. It seeks to provide a clear roadmap for every team member, detailing the changes and signposting tasks that need attention in the subsequent developmental stages. Moreover, this system of change management aims to seamlessly integrate the myriad contributions from multiple developers, ensuring that the final software product hews as closely as possible to the predetermined requirements.To address this need, the discipline of Software Configuration Management (SCM) or, as it is alternately termed, Revision Control, takes center stage. SCM as a process tailored to pinpoint and record system configurations at particular moments in time [2].

This meticulous documentation aims to judiciously control configuration changes, all the while ensuring the integrity of the configurations is maintained throughout the software development cycle. Fundamentally, SCM offers an invaluable service: it meticulously logs changes to the software, thus ensuring traceability and minimizing the risk of errors that could compromise the harmonious functionality of the software. Furthermore, its strategic importance is underlined when one considers the challenges faced by development teams, often composed of multiple members, in coherently managing the source code, whether originating from their efforts or those of their colleagues.Within the expansive architecture of SCM, there exist several integral mechanisms, each designed to work in tandem, offering a holistic suite of tools that aid developers in their software creation journey. Of these mechanisms, version control and data

comparison are particularly noteworthy. The latter, data comparison, emerges as an indispensable tool, designed to identify and highlight differences between two distinct data sets. For software developers, this tool becomes a beacon, allowing them to discern variations between successive software versions. This obviates the need for developers to embark on the tedious process of manually trawling through each line of source code to isolate and understand the changes.Interestingly, this domain of study and its significance was spotlighted in [3], the goal of this study is to compare 12 existing state-of-the-art and commercial XML diff algorithms by applying them to JATS documents in order to extract and evaluate changes between two versions of the same academic article.

[4], They developed a tool, based on a simple "text diff" algorithm, to detect drastic code changes in student code progressions, and to point instructors to possible cheating cases. In contrast, the current research endeavor adopts a more panoramic view, embracing all genres of source code and eschewing bias towards any specific programming language. Additionally, a key differential in the current study is the chosen algorithmic foundation. This research is predicated on the merits of the Myers Diff algorithm. The choice of the latter is underpinned by its proven proficiency in parsing changes between two text-based files. Its enduring relevance is further underscored by its continued deployment in numerous data comparison tools, a testament to its speed, reliability, and robustness, despite its extended tenure in the realm of algorithms.Given the aforementioned complexities and the imperative for software developers to concentrate on the core tasks of software innovation, this research proposes the development of a novel tool. This endeavor culminates in the project aptly titled, "Innovations in Software Configuration: Introducing a Data Comparison Tool Based on the Myers Diff Algorithm," which promises to be a pivotal resource for the software development community.

II. METHODS

The research methodology employed in this study is outlined as follows:

Literature Review

An extensive review of existing sources such as journals, reports, applications, and previously published articles by other researchers was conducted. This review focused on topics related to differential analysis, the longest common subsequence problem, software configuration management, the Myers Diff algorithm, and other pertinent subjects that contribute to the development of a data comparison application for this research.

Design Phase

The design phase involved conceptualizing a data comparison application. During this stage, the Myers Diff algorithm was incorporated into a web-based program, in alignment with the pre-established objectives and parameters of the study.

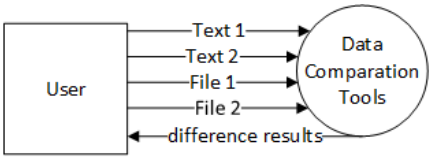

Fig 1. Context Diagram

Figure 1 illustrates the Level 0 Data Flow Diagram (DFD) or context diagram for the Data Comparison Tools application. Users have two distinct methods available for examining the differences between two sets of data:

1. Direct Text Input: Users can manually enter the data into the application by copying and pasting it into the designated input field provided within the platform.
2. File Upload: Alternatively, users can upload the data intended for comparison in the form of a file. It's imperative that the uploaded file be text-based.

Once the data, either in text form or as a file, is introduced into the application, users can initiate the comparison process by pressing the corresponding button within the interface. Upon completion of this comparative analysis, the application will then present the disparities between the two datasets to the user.

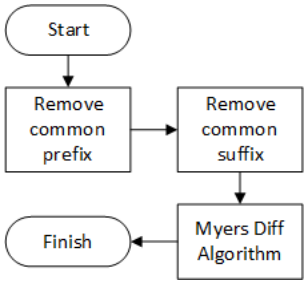

Fig 2. Method Flowchart

Figure 2 presents the flow chart for the "Identify Differences" process. The procedure commences with the elimination of the common prefix, which refers to the identical string segment present at the beginning of both Text 1 and Text 2. The process subsequently advances to the removal of the common suffix, a shared string segment located at the termination of both texts. These preliminary steps — the elimination of common prefix and common suffix — are strategically incorporated to enhance the application's performance efficiency. The data, once refined through these initial steps, is then channeled into the Myers Diff Algorithm for further processing.

Implementation

The previously designed algorithm and framework were operationalized and integrated into the program, culminating in the construction of the data comparison application.

Myers Diff Algorithm

The Myers Diff Algorithm, a cornerstone in the realm of text comparison and difference detection, was introduced by Eugene Myers as an innovative solution to compute the differences, or "diff," between two sequences. Rather than simply identifying shared subsequences or modifications, Myers' approach pinpoints the most efficient way to transform one sequence into another [5]. This paper dissects the core mechanics of the Myers Diff Algorithm, delineating its operational framework and methods.

To enhance understanding of the Myers Diff Algorithm, let's elucidate its mechanics using a concrete example. Consider two sequences, A and B, with A being "ABCABBA" and B being "CBABAC". Our objective is to discern the sequence of edits to transform A into B using Myers' method.

1. Constructing the Edit Graph

Initiate an edit graph with A and B. The horizontal axis represents A and the vertical axis represents B. Each point (x, y) is an intersection of elements A[x] and B[y].

2. Traversal and Diagonal Moves

Start at the origin (0,0). The algorithm identifies a match between A[1] = B[1] = B, which allows a diagonal move. However, the next characters A[2] = A and B[2] = C don't match, preventing further diagonal progress.

3. Horizontal and Vertical Moves

Given the mismatch at A[2] and B[2], the algorithm opts for a vertical move, indicating a deletion from A or an addition to B. This is consistent with deleting the A from A or adding the C to B.

Moving further, there’s a diagonal progression again, with matches at "BCA" in A and "BCB" in B.

4. Bit-vector Implementation (Simplified)

Using bit-vectors, the paths are encoded as binary. If the path is possible, the position is marked as 1; otherwise, it's 0. The algorithm utilizes bitwise operations on these vectors for efficient path calculations.

5. Resulting Difference

6. From the algorithm's traversal, the difference between A and B can be summarized as:

Delete A from the 2nd position of A.

Add C to the 2nd position of B.

The sequences "BCA" from A and "BCB" from B match except for the last character.

Delete BA from the end of A.

Add C to the end of B.

III. RESULT AND DISCUSSION

System Implementation

The Data Comparison Tools application is architected as a web-based platform, utilizing PHP as the server-side scripting language and JavaScript on the client side. The interface is crafted using Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS). Additionally, specific frameworks and libraries are integrated to expedite the application development process.Figure 3 displays the primary user interface of the application. This view greets users upon accessing the Data Comparison Tools. Situated in the top-right corner of the application is a switch, allowing users to modify the input format based on their preference. By default, the selected input format is "TEXT," and users are presented with two text areas: the first for inputting the older or original source code, and the second for entering the newer version of the source code.

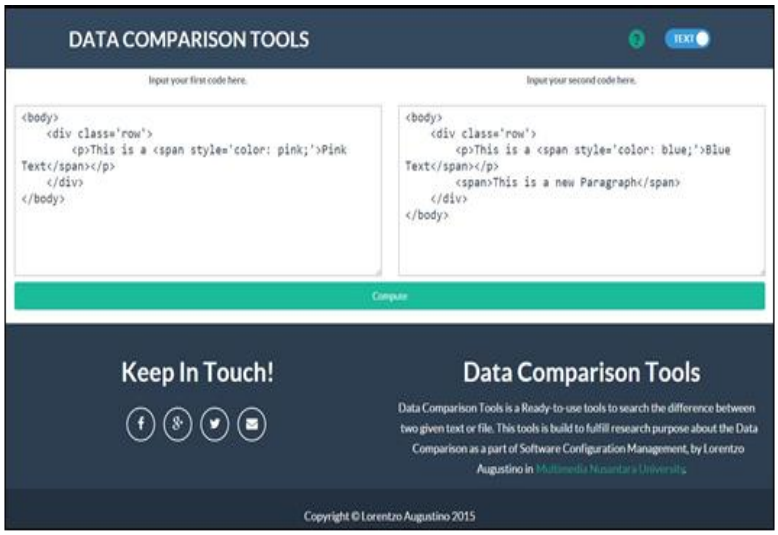

Fig 3. User Interface of Aplication

Testing

The research progressed by conducting tests on the application's response under various conditions [6], observing the reactions produced by the software.

1. Both Text1 and Text2 are empty.
2. Text1 is empty, while Text2 contains placeholder text.
3. Text1 contains placeholder text, and Text2 is empty.
4. Both File1 and File2 are left empty.
5. File1 is uploaded with an empty file (no content), and File2 is left empty.
6. File1 is left empty, while File2 is uploaded with an empty file (no content).
7. File1 is uploaded with a non-text-based file, and File2 is uploaded with a placeholder file.
8. File1 is uploaded with a placeholder file, and File2 is uploaded with a non-text-based file.

The application responds as expected. In addition to condition testing, the Data Comparison Tools application was also assessed for the validity of its results. The validation test was conducted using comparable applications, and the outcomes were then juxtaposed with the results produced by the developed Data Comparison Tools application.

Validation

Process validation is the analysis of data gathered throughout the design and manufacturing of a product in order to confirm that the process can reliably output products of a determined standard [7][8]. The validation process is carried out by inputting each predetermined data pair into the Data Comparison Tools application developed in this research, as well as into other commercial applications. TortoiseGit (v1.8.11.0), which is already installed on the personal computer that also serves as the server where the Data Comparison Tools from this research are tested, was used to help verify the validity of the application developed in this study [9]. Evidence can be seen from the difference in percentage change calculated using either Application 1 or Application 2, with a gap of less than 1% for each data pair.

Acceptance Testing

Acceptance testing conducted at the site at which the product is developed and performed by employees of the supplier organization, to determine whether a component or system satisfies the requirements, normally including hardware as well as software [10][11]. Acceptance testing was conducted to determine whether the method of displaying differences is both effective and easily readable. Two criteria were used: "Functional Correctness and Completeness" and "Accuracy". Acceptance testing was carried out by surveying all members of the population, who are employees at PT Teltics Media. There were 8 (eight) respondents in total. It can be concluded that for each user within the predetermined population, there is at least one output format considered good and easy to view. From these results, it's inferred that the Data Comparison Tools application effectively presents character differences found between the two texts and does so in an easily viewable manner. However, each user has their own preference regarding the optimal output format. Of the three output formats provided by the application, the second format (Side by Side) was the most preferred by respondents, with 4 (four) out of 8 (eight) respondents choosing it. Additionally, this second format (Side by Side) was deemed the best and easiest to read by the respondents, with 7 out of the 8 respondents answering "Yes".

Performance Testing

Performance testing is a testing measure that evaluates the speed, responsiveness and stability of a computer, network, software program or device under a workload [12]. Performance testing was conducted to determine the speed of the application developed in this research under various conditions. Performance measurements were made by observing the time the application took to identify differences between two sets of data. The testing process involved comparing five pairs of data using the Data Comparison Tools application and recording the time taken. Tests were conducted on these five pairs of data for each provided scenario. The scenarios are as follows:

1. The application was placed on a local server situated on the same machine used for performance testing. This condition was tested by accessing the localhost through a local network.
2. The application was hosted on a web server, and accessed using an internet connection with an average download speed of 425KB per second and an average upload speed of 48KB per second.
3. The application was hosted on a web server and accessed using an internet connection with an average download speed of 720KB per second and an average upload speed of 720KB per second.
4. All three scenarios were tested using the same machine and the same web browser, specifically Google Chrome version 43.0.2357.124m."

IV. CONCLUSION

The conclusion drawn from the research is that the Data Comparison Tools application has been successfully developed by implementing the Myers Diff algorithm to compare two sets of data provided. The algorithm utilized effectively identifies character differences between the two texts, be it additions or subtractions. This can be seen from the similarities in character changes identified using the developed Data Comparison Tools application in comparison to the changes identified using a predetermined commercial application. However, the developed application is less effective when given two sets of data that have very low similarities or no similarities at all.Furthermore, this research has successfully presented an output format that is clear and easily viewable.

This is evident from the respondents' feedback, with at least one output format they considered to be good and easy to read. The developed application also underwent performance testing to measure its efficacy in identifying differences between two sets of data provided. The application displayed relatively consistent execution times, showing minimal variance when run on a local network or over the internet."Suggestions for further research include the continued need for development to optimize the output format generated by the application. For instance, it could compare the existing source code line-by-line or word-by-word and format the differences in a manner that's more human-readable or semantically structured. Additionally, there's a need for a mechanism that can assist the application in identifying differences between two sets of data that have very low similarities

V. ACKNOWLEDGMENTS

Thank you to Multimedia Nusantara University for providing funding support and providing research space.

REFERENCES

[1] E. H. Bersoff, V. D. Henderson, and S. G. Siegel, "Software configuration management," *ACM SIGSOFT Softw. Eng. Notes*, vol. 3, no. 5, pp. 9–17, 1978.

[2] S. Berczuk and B. Appleton, *Software configuration management patterns: effective teamwork, practical integration*. Addison-Wesley Professional, 2020.

[3] Mamangkey, J., Suryanto, D., et all (2021). Isolation and enzyme bioprospection of bacteria associated to Bruguiera cylindrica, a mangrove plant of North Sumatra, Indonesia, Biotechnology Reports, 2021, 30, e00617.

[4] M. Cuculovic, F. Fondement, M. Devanne, J. Weber, and M. Hassenforder, "Semantics to the rescue of document-based XML diff: A JATS case study," *Softw. Pract. Exp.*, vol. 52, no. 6, pp. 1496–1516, 2022.

[5] N. Alzahrani and F. Vahid, "Detecting Possible Cheating In Programming Courses Using Drastic Code Change," in *2022 ASEE Annual Conference & Exposition*, 2022.

[6] Harahap, P. Hrp, N.K.A.R. Dewi, Macrozoobenthos diversity as anbioindicator of the water quality in the River Kualuh Labuhanbatu Utara, ***International Journal of Scientific & Technology Research,*** 9(4),2020,pp.179-183.

[7] M. Sjölund, "Evaluating a Tree Diff Algorithm for Use in Modelica Tools," in *Modelica Conferences*, 2021, pp. 529–537.

[8] Harahap, Arman ,2018, Macrozoobenthos diversity as bioindicator of water quality in the Bilah river, Rantauprapat, Medan. ***J. Phys.***: Conf. Ser. 1116 052026.

[9] M. Banafaa *et al.*, "6G mobile communication technology: Requirements, targets, applications, challenges, advantages, and opportunities," ***Alexandria Eng. J.,*** vol. 64, pp. 245–274, 2023.

[10] W. Xu, "Toward human-centered AI: a perspective from human-computer interaction," *interactions*, vol. 26, no. 4, pp. 42–46, 2019.

[11] Harahap, A., et all (2021), Monitoring Of Macroinvertebrates Along Streams Of Bilah River ***International Journal of Conservation Sciencethis*** link is disabled, 12(1), pp. 247–258.

[12] B. G. Santoso, F. A. T. Tobing, and A. Kusnadi, "ERP Odoo Based Medical Reimbursement System Using Scrum Method:(Study Case: Group of Retail and Publishing Kompas Gramedia)," in *2023 20th* ***International Joint Conference on Computer Science and Software Engineering (JCSSE)***, IEEE, 2023, pp. 327–332.

[13] T. Haslwanter, "Useful Programming Tools," in *Hands-on Signal Analysis with Python: An Introduction*, Springer, 2021, pp. 197–204.

[14] Harahap, et, all, Macrozoobenthos diversity as anbioindicator of the water quality in the Sungai Kualuh Labuhanbatu Utara, AACL Bioflux, 2022, Vol 15, Issue 6.

[15] N. Nasir, "Acceptance Testing in Agile Software Development-Perspectives from Research and Practice." 2021.

[16] G. K. Suryatenggara and W. Istiono, "Enhancing Campus Communication and Collaboration: Design and Development of a Social Community Media Website for Universitas Multimedia Nusantara, Indonesia," ***Asian J. Res. Comput. Sci.***, vol. 16, no. 3, pp. 281–289, 2023.

[17] Harahap, Arman. 2020. Species Composition & Ecology Index Of The Family Gobiidae At The Mangrove Belawan Of Sicanang Island ***International Journal of Scientific & Technology Research*** Volume 9, Issue 04, April 2020.

[18] M. T. Hossain, R. Hassan, M. Amjad, and M. A. Rahman, "Web Performance Analysis: An Empirical Analysis of E-Commerce Sites in Bangladesh.," ***Int. J. Inf. Eng. Electron. Bus.***, vol. 13, no. 4, 2021.