|  |  |
| --- | --- |
|  | Title is in english max 15 words left alignment bold arial 14 sentence case |

[](https://crossmark.crossref.org/dialog/?doi=10.22219/kinetik.v4i3.841&domain=pdf)

First Author\*1, Second Author2, Third Author3

1, First Afiliation, Country

2, Second Afiliation, Country

3, Third Afiliation, Country

|  |  |
| --- | --- |
| Article Info | Abstract |
| Keywords:  Internet of Things Platform, Internet of Things, Message Queuing Telemetry Transport, MQTT Broker Server  Article history:  Received 17 August 2018  Revised 15 February 2019  Accepted 4 April 2019  Available online 4 April 2019  Cite:  Wardana, A., Rakhmatsyah, A., Minarno, A., & Anbiya, D. (2019). Internet of Things Platform for Manage Multiple Message Queuing Telemetry Transport Broker Server. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control, 4(3). doi:<http://dx.doi.org/10.22219/kinetik.v4i3.841>  \* Corresponding author.  Corresponding Author  E-mail address:  author@email.ac.id | *Code clones merupakan segmen kode di dalam suatu source files yang identik atau memiliki kemiripan antara satu sama lain. Semakin besar dan kompleks suatu sistem, semakin banyak kebutuhan sumber daya dan biaya yang diperlukan untuk mendeteksi code clones. Salah satu cara untuk meringankan sumber daya pada proses mendeteksi code clones adalah dengan otomasi proses mendeteksi code clones. Pada penelitian ini, proses mendeteksi code clones secara otomatis yang akan dirancang menggunakan pendekatan token. Penelitian ini bertujuan untuk mengevaluasi sebuah sistem pendeteksi code clones otomatis menggunakan pendekatan token untuk mengobservasi keakuratan hasil dan mengamati efisiensi waktu sistem tersebut.* |
|  |  |

1. Introduction

Menurut McCall, maintainability adalah besarnya usaha yang diperlukan untuk melokalisasi dan membetulkan kesalahan-kesalahan yang dapat ditemukan dalam program. Sistem perangkat lunak yang dapat melakukan operasi maintenance dengan mudah memiliki kualitas maintainability yang bagus. Menurut observasi, 60%-70% dari seluruh life cycle perangkat lunak digunakan untuk pemeliharaan perangkat lunak dalam sisi sumber daya, waktu, biaya dan usaha [2]. Usaha untuk meningkatkan software maintainability cukup sulit dan dapat menghabiskan sebagian besar biaya total proyek.

Code clones merupakan segmen kode di dalam suatu source files yang identik atau memiliki kemiripan antara satu sama lain [3], [4]. Semakin banyak code clones pada suatu sistem perangkat lunak maka semakin banyak Line of Codes pada suatu sistem perangkat lunak yang harus dipelihara oleh seorang developer. Oleh karena itu salah satu faktor yang dapat menyebabkan performansi maintainability rendah pada suatu sistem perangkat lunak adalah code clones. Ketika sebuah sistem mempunyai code clone sub-system yang diciptakan oleh duplikasi kode dengan sedikit modifikasi, seorang developer harus berhati-hati memodifikasi seluruh sub-system yang lain ketika software fault ditemukan [5]. Clones didalam source codes menyebabkan kesulitan dalam memodifikasi program secara konsisten. Code clones dapat diciptakan akibat penggunaan kembali kode dengan “copy-and-paste” atau dengan mengulangi porsi kode secara sengaja oleh seorang developer [4], [6].

Terdapat banyak tipe code clone yang dapat dideteksi dalam suatu source files yaitu Clones dari salinan yang sangat identik tanpa modifikasi (type-1), clones dari salinan identik secara sintaksis (type-2) dan clones dari salinan kode dengan sedikit modifikasi (type-3) [4]. Tahap yang penting untuk menghilangkan dan mengkoreksi code clones yaitu tahap mendeteksi code clones. Mendeteksi code clones dapat membantu mencari bugs, menentukan perbaikan bug yang tidak konsisten dan menemukan redundansi di dalam kode [3].

Mendeteksi code clones merupakan kegiatan yang memakan waktu banyak karena seorang developer harus mengkonfirmasi atau membandingkan seluruh baris kode pada source files secara manual [5]. Diperlukan usaha dan sumber daya yang banyak untuk mendeteksi code clones sangatlah penting. Semakin besar dan kompleks suatu sistem, semakin banyak kebutuhan sumber daya dan biaya yang diperlukan untuk mendeteksi code clones.

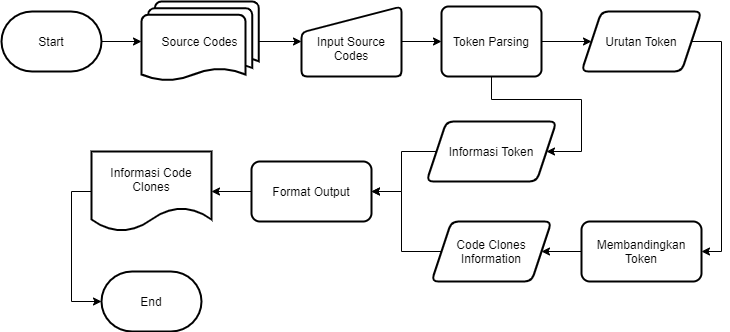
Salah satu cara untuk meringankan sumber daya pada proses mendeteksi code clones adalah dengan otomasi proses mendeteksi code clones [4]. Pada penelitian ini, sistem pendeteksi code clones yang akan dibuat menggunakan metode perbandingan token. Keunggulan metode perbandingan token yaitu metode ini dapat merepresentasi source code sebagai urutan token yang memungkinkan untuk mendeteksi clones dengan struktur baris berbeda [5]. Sistem ini akan membuat sebuah urutan token dari kode inputan melalui analisis leksikal dan menerapkan transformasi berbasis aturan pada urutan token. Kami berharap penelitian ini dapat berkontribusi untuk menambah pemahaman tentang software enggineering dan menambah kelebihan dan kekurangan dalam metode ini.

1. Research Method
   1. Datasets

Penelitian ini menggunakan hasil dari penelitian A Code Clone Oracle yang diterbitkan pada tahun 2014 oleh MSR 2014.

* 1. Perancangan Sistem

Sistem yang akan dirancang menerima inputan berupa source code dari satu atau lebih program. Sistem yang akan dirancang memiliki lima tahap utama yaitu input source codes, token parsing, string matching, format output dan evaluate result. merupakan alur dari sistem pendeteksi code clones.



* + 1. Input Source Code

Pada tahap ini dilakukan pembacaan file *source code* yang menjadi input pada sistem. *Source code* yang diinput berbahasa C. Sistem dapat menerima input dari beberapa *source codes* maupun satu *source code*. Jika sistem menerima banyak inputan *source codes* maka sistem akan menyisipkan ke satu buah file sehingga terbaca seperti satu buah *source code*.



* + 1. Token Parsing

Figure 3‑4 merupakan tiga tahap utama dalam mengubah inputan *source codes* menjadi sebuah *token sequence* dan informasi baris fungsi/prosedur dari token tersebut. Tahapan ini dibagai menjadi tiga bagian, yaitu *remove white space*, *shorten names*, dan *tokenize*. Hasil proses ini adalah hasil transformasi *token sequence* informasi seluruh fungsi dari *token sequence*.



* + - 1. Remove White Space And Comment

Tahap ini berfokus terhadap penghapusan indentasi di setiap baris source codes dan spasi tambahan yang tidak diperlukan. Tahap ini memiliki tujuan untuk menghapus data yang dapat merusak hasil dari string matching sehingga data mudah untuk diproses pada tahap string matching.



* + - 1. Tokenize

Pada tahap ini, sistem mengganti seluruh *identifier* variabel, tipe, dan konstanta menjadi token, kecuali *language construct* seperti lambang operator. Tujuan dari proses ini untuk mendeteksi porsi kode dengan variabel yang berbeda nama dapat terdeteksi sebagai *clone*.



* 1. String Matching

Sistem yang akan dirancang akan mendeteksi *code clone* pada level fungsi. Satu fungsi yang telah ditemukan akan dibandingkan dengan satu fungsi lain untuk diperiksa *code* *pattern* menggunakan algoritma *string matching*. Kedua fungsi yang memiliki *code* *pattern* yang mirip dengan fungsi lain disebut *clone pair.*

Sistem yang akan dirancang memiliki dua model algoritma *string matching* yang digunakan untuk menjadi eksperimen efisiensi mendeteksi *code clones*. Model pertama menggunakan algoritma *Knuth Morris Pratt pattern searching*. Sistem akan mencatat waktu yang dibutuhkan untuk menjalankan algoritma *string matching* yang dipakai. Hasil *clone pair* dan waktu berjalan setiap model algoritma merupakan *output* dari tahap ini.

* 1. Format Output

Pada tahap ini, sistem mengeluarkan seluruh informasi fungsi yang ada pada *source codes*, *clone pairs*, dan waktu yang diperlukan untuk mendeteksi *code clones*. Penamaan function yang dikeluarkan sistem ditambah dengan nama file sebagi *prefix* yang dipisah oleh tanda “.”, misalkan fungsi A terdapat pada file bernama X sehingga penamaan fungsi menjadi “X.A”.

*Output* pada tahap ini berupa file berformat excel yang memiliki dua kolom, *function pairs* dan *clone pairs*.Kolom *function pairs* berisi pasangan dua fungsi yang ada pada *source codes* dan kolom *clone pairs* berisi nilai “yes” atau “no” tergantung kepada pasangan dua fungsi kolom *function pairs*. Kolom *clone pairs* bernilai “yes” jika pasangan dua fungsi merupakan *code clones* dan bernilai “No” jika pasangan dua fungsi bukan merupakan *code clones*. Informasi waktu yang diperlukan untuk mendeteksi *code clones* terdapat pada baris terakhir pada file excel yang dipisahkan dengan baris dengan nilai *function pairs* “End”. Tabel 3‑1 adalah contoh *output* dari tahap ini.

|  |  |
| --- | --- |
| Function Pairs | Code Clone |
| File A.Function A-File A.Function B | Yes |
| File A.Function A-File A.Function C | No |
| … | … |
| File X.Function X-File Y.Function Y | No |

* 1. Evaluasi Hasil

Hasil dari tahap format output akan dibandingkan dengan hasil dari penelitian “A Code Clone Oracle” oleh Krutz dan Le (Krutz & Le, 2014). Sistem akan menghitung jumlah prediksi clone pairs yang benar sesuai hasil penelitian Krutz dan Le dan menghitung akurasi dengam menggunakan Equation 3‑1 Perhitungan Akurasi Sistem Pendeteksi Code Clones. Untuk mengevaluasi performansi waktu, nilai waktu yang didapatkan pada format output akan dibandingkan dengan berapa Line Of Codes dari source codes, sehingga akan didapatkan nilai waktu yang diperlukan untuk menjalankan source codes dengan LOC tertentu.

**3. Results and Discussion**

**4. Conclusion**

**Notation**

The example of notation can be described with the following description:

n : the number of data

Mi : the median value of i-th class.

: the average value of data.

Fi : Frequency of i-th data.

Acknowledgement

Thank you to the Internet of Things Studio, Telkom University which has become a place for researchers to develop this journal research. Hopefully, this research can make a major contribution to the advancement of technology in Indonesia.

**References**

The minimal number of references used in manuscript must be 25 references using APA Style. The main references are reputed international journals or proceeding and doi hyperlink must be provided. All references should be to the most pertinent and up-to-date sources. In reference and citation are mandatory using reference apps such as Mendeley. Please use the reference writing procedure provided in this guide:

References

Al-Khamaiseh, K., & ALShagarin, S. (2014, August). A Survey of String Matching Algorithms. *International Journal of Engineering Research and Applications, 4*, 144-156.

Baker, B. S. (1992). A program for identifying duplicated code. *Computing Science and Statistics*.

Ducasse, S., Rieger, M., & Demeyer, S. (1999). A language independent approach for detecting duplicated code. *Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). 'Software Maintenance for Business Change' (Cat. No.99CB36360)*, 109-118.

Kamiya, T., Kusumoto, S., & Inoue, K. (2002, 08). CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *Software Engineering, IEEE Transactions on, 28*, 654-670. doi:10.1109/TSE.2002.1019480

Krutz, D. E., & Le, W. (2014). A Code Clone Oracle. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*. doi:10.1145/2597073.2597127

*Maintainability*. (t.thn.). Dipetik November 20, 2020, dari iso25000.com: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010/57-maintainability

Malhotra, R., & Anuradha, C. (2016). Software Maintainability: Systematic Literature Review and Current Trends. *International Journal of Software Engineering and Knowledge Engineering, 26*, 1221-1253. doi:10.1142/S0218194016500431

Mens, T., & Demeyer, S. (2010). Chapter 2 Identifying and Removing Software Clones. Dalam *Software evolution* (hal. 15-36). Berlin: Springer. doi:10.1007/978-3-540-76440-3

Onuoha, O. (2016). Detecting Code Clones: A review. *arXiv preprint arXiv:1605.02661*.

Pressman, R. S. (2012). BAB 14 Konsep-konsep Kualitas. Dalam R. S. Pressman, *Rekayasa Perangkat Lunak: Pendekatan Praktisi, Edisi 7* (hal. 482-504). Yogyakarta: Penerbit ANDI.