# PENDAHULUAN

## Latar Belakang

Menurut McCall, *maintainability* adalah besarnya usaha yang diperlukan untuk melokalisasi dan membetulkan kesalahan-kesalahan yang dapat ditemukan dalam program [1]. 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 leksikaldan 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.

## Rumusan Masalah

Dari latar belakang yang sudah dipaparkan diatas, masalah yang akan dibahas pada penelitian ini adalah :

1. Seberapakah akurat sistem pendeteksi *code clones* otomatis menggunakan perbandingan token dalam mendeteksi *code clones*?
2. Bagaimana performasi komputasi sistem otomatisasi pendeteksi *code clones* menggunakan pendekatan perbandingan token dalam sisi waktu?

## Tujuan

Penelitian ini bertujuan untuk membuat sebuah sistem pendeteksi *code clones* otomatis menggunakan metode perbandingan tokenuntuk mengobservasi keakuratan dan mengamati efisiensi komputasi metode tersebut.