# TINJAUAN PUSTAKA

## Penelitian Terkait

Penelitian terkait dengan sistem pendeteksi *code clone* merupakan studi yang masih aktif diteliti. Salah satunya adalah sistem pendeteksi *code* *clones* secara otomatis dengan model pendekatan Baker menggunakan metode pembandingan token [6]. *Source code* akan direpresentasikan menjadi urutan token dari baris-baris kode yang sudah diubah menjadi token dengan analisis leksikal. Sistem ini akan membandingkan urutan token secara baris antar baris dengan menggunakan *suffix tree* untuk mendeteksi *code clone*. Pendekatan ini dapat mendeteksi *code clone* meskipun *syntax*-nya berbeda.

Sistem pendeteksi *code clones* otomatis dengan model pendekatan Kamiya dkk. juga merepresentasikan *source code* dengan urutan token [5]. Namun, model pendekatan Kamiya membandingkan token antar token tanpa menghiraukan baris kode. Pada model pendekatan Kamiya mentransformasikan urutan kode sesuai peraturan yang telah mereka definisikan dan membuat metrik *clones* untuk evaluasi pasangan *code* *clones* sebagai teknik optimasi efisiensi komputasi.

Model pendekatan Ducasse dkk. membandingkan seluruh baris antar baris pada *source code* secara tekstual [7]. Model pendekatan ini tidak melalui proses parsing sehingga model pendekatan ini *language independent*. Untuk menambah efisiensi komputasi, baris-baris kode pada *source code* dipartisi menggunakan fungsi *hash* *strings*. Hasil proses pendeteksi *code* *clones* divisualisasikan dengan *dotplot*, setiap titik merupakan baris yang terdeteksi sebagai pasangan *code clones*.

## *Software Maintainability*

*Maintainability* memiliki banyak definisi yang ambigu di dalam konteks kualitas perangkat lunak. *Maintainability* dapat didefinisikan sebagai kemampuan suatu perangkat lunak untuk memperbaiki kesalahan-kesalahan dan dalam memodifikasi atau yang dapat ditemukan dalam program [1]. Dengan kata lain *maintainability* mengukur kemudahan perangkat lunak dalam beradaptasi untuk memenuhi kebutuhan dan ekspektasi pengguna.

ISO 25010 menyebutkan bahwa *maintainability* merepresentasikan derajat efektivitas dan efisiensi suatu sistem dapat dimodifikasi untuk memperbaharui, mengkoreksi atau mengadaptasi [8]. Terdapat beberapa sub-faktor kualitas pada faktor kualitas *maintainability* untuk dijadikan sebagai aspek penilaian kualitas :

* ***Modularity***, ukuran dari suatu komponen program yang dapat diubah dengan memiliki efek minim kepada komponen lain pada program.
* ***Reusability***, aset-aset yang dapat digunakan pada sistem lain pada suatu program.
* ***Modifiability***, efektivitas suatu perangkat lunak dalam penambahan dan perbaikan fitur tanpa mengurangi kualitas dan menambah *error*/*bug*.
* ***Testability***, efektivitas suatu perangkat lunak dalam ketersediaan kriteria *testing* yang dapat dijalankan oleh sistem.

## *Code Clones*

*Code clones* merupakan segmen kode di dalam suatu *source* *files* yang identik atau memiliki kemiripan antara satu sama lain [3]. *Code clones* adalah redundansi dalam kode yang dapat menambah beban pekerjaan *maintenance* [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*. *Figure* 2‑1 Contoh Code Clonesmerupakan contoh dari *code clones*.

Figure 2‑1 Contoh Code Clones

*Code clones* memiliki tiga tipe yang dapat dibandingkan berdasarkan basis teks program yang disalin [4]. Tipe-tipe *code clones* tersebut yaitu :

* **Tipe 1** merupakan salinan yang sama dari segmen kode tanpa ada modifikasi, kecuali perbedaan spasi dan komentar.
* **Tipe 2** merupakan salinan dengan struktur sintaks, hanya ada perbedaan pada nama variabel, tipe dan *function identifier*.
* **Tipe 3** merupakan salinan dari segmen kode dengan sedikit modifikasi struktur.

Akar munculnya *code clones* adalah dari *programmer*s yang melakukan *forking*, *templating* dan *customization* [4]. *Forking* adalah menduplikasi yang digunakan untuk bootstrap pengembangan solusi serupa, dengan harapan bahwa evolusi kode akan terjadi secara independen, setidaknya dalam jangka pendek. *Templating* digunakan sebagai metode untuk langsung menyalin perilaku dari kode yang ada, tetapi mekanisme abstraksi yang sesuai tidak tersedia dalam jangka pendek. *Customization* terjadi ketika kode yang ada saat ini tidak cukup memenuhi serangkaian persyaratan yang baru.

## *String Matching Algorithm*

Konsep suatu algoritma *string matching* adalah mencari satu atau beberapa pola *string* yang terdapat pada suatu teks yang banyak [10]. Algoritma *string matching* memindai teks menggunakan *window* teks sebagai pemeriksa kemunculan pola dengan membandingkan karakter *window* dengan karakter pola. Setelah mencapai pola yang cocok atau setelah tidak cocokan, *windows akan* bergeser ke pola selanjutnya. Proses mencocokkan dan mengeser *windows* akan terus diulang hingga akhir ujung teks.

### *Knuth-Morris (KMP) Algorithm*

Algoritma ini mencari kemunculan dari pola dari suatu teks dari kiri ke kanan dengan menggunakan pengamatan bahwa ketika terjadi ketidaksesuaian, memaksimalkan pergeseran untuk menghindari redundansi perbandingan, sehingga mendapatkan keuntungan dari karakter yang cocok sebelumnya. Algoritma ini memberikan keuntungan bahwa *pointer* pada teks tidak akan decremented.

## Metode Perbandingan Token

Metode perbandingan token membandingkan kode dengan menggunakan token [4]. Metode ini membandingkan secara baris per baris dari kode yang sudah ditransformasikan menjadi urutan-urutan token menggunakan sebuah *token parser*. Token parser yang digunakan bergantung dengan bahasa pemrograman yang dijadikan model [6]. Metode ini dianggap kelas berat karena memerlukan *language parser* yang khusus untuk bahasa pemrograman tertentu.

Tujuan dari transformasi kode menjadi urutan token adalah untuk mendeteksi *code clones* yang memiliki *syntax* berbeda tetapi memiliki makna yang sama [5]. Mewakili kode sebagai urutan token memungkinkan untuk mendeteksi *code clones* dengan perbedaan struktur baris, yang tidak dapat dideteksi metode perbandingan baris.