**PERHITUNGAN LINE OF CODE DAN CODE CLONING PADA PERANGKAT LUNAK**

**STUDI KASUS APLIKASI PARADISE HOTEL**

Mata Kuliah : Evolusi Perangkat Keras



Mu’ammar Asaduddin

1695114061

**S1 TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS HASYIM ASY’ARI**

**TEBUIRENG - JOMBANG**

**2019**

**BAB I**

**PENDAHULUAN**

1. **Latar Belakang**

Proses pengembangan perangkat lunak fase implementasi adalah fase realisasi dari fase desain. Programmer adalah orang yang bertanggung jawab atas pembuatan kode program. Dalam proses pembuatan kode, programmer sering menggunakan kembali kode program yang sudah pernah dibuat dengan cara melakukan copy dan paste. Proses penduplikasian kode dengan atau tidak disertai dengan modifikasi disebut sebagai proses cloning. Proses kloning kode merupakan hal yang biasa dalam proses pembuatan kode program. Pembahasan yang berkaitan dengan kode kloning telah dilakukan oleh beberapa peneliti. Berbagai macam metode dalam proses pendeteksian kloning pada sebuah kode sumber (source code) telah dibahas. Begitu juga perbandingan tools atau alat pendeteksian kode kloning juga sudah dilakukan.

Kloning dipercaya dapat dapat menurunkan tingkat perawatan (maintainability) sebuah software. Hal ini disebabkan oleh adanya perilaku yang tidak konsistensi melakukan perubahan pada kloning kode. Sebagi conoh, kloning dilakukan pada beberapa modul dalam sebuah perangkat lunak, apabila satu dilakukan perubahan maka yang lain juga harus mendapatkan perlakuan yang sama. Perlakuan yang tidak sama dapat menyebabkan kondisi yang tidak konsisten. Kodisi ini dapat

menumbuhkan cacat (defect) pada perangkat lunak yang dikembangkan. Pada beberapa penelitian diungkapkan bahwa cacat perangkat lunak juga dipengaruhi oleh kompleksitas kode program. Kompleksitas kode program diukur dengan beberapa matrik yaitu LOC (Line of Code), McCabe’s Cyclomatic Complexity, dan Halstead’s Volume. Kompleksitas kode program dijadikan sebagai salah satu acuan dalam melihat kualitas program dan dengan menggunakan metode klasifikasi dapat dijadikan menjadi sebuah model dalam pendeteksian cacat pada perangkat lunak sedini mungkin.

**BAB II**

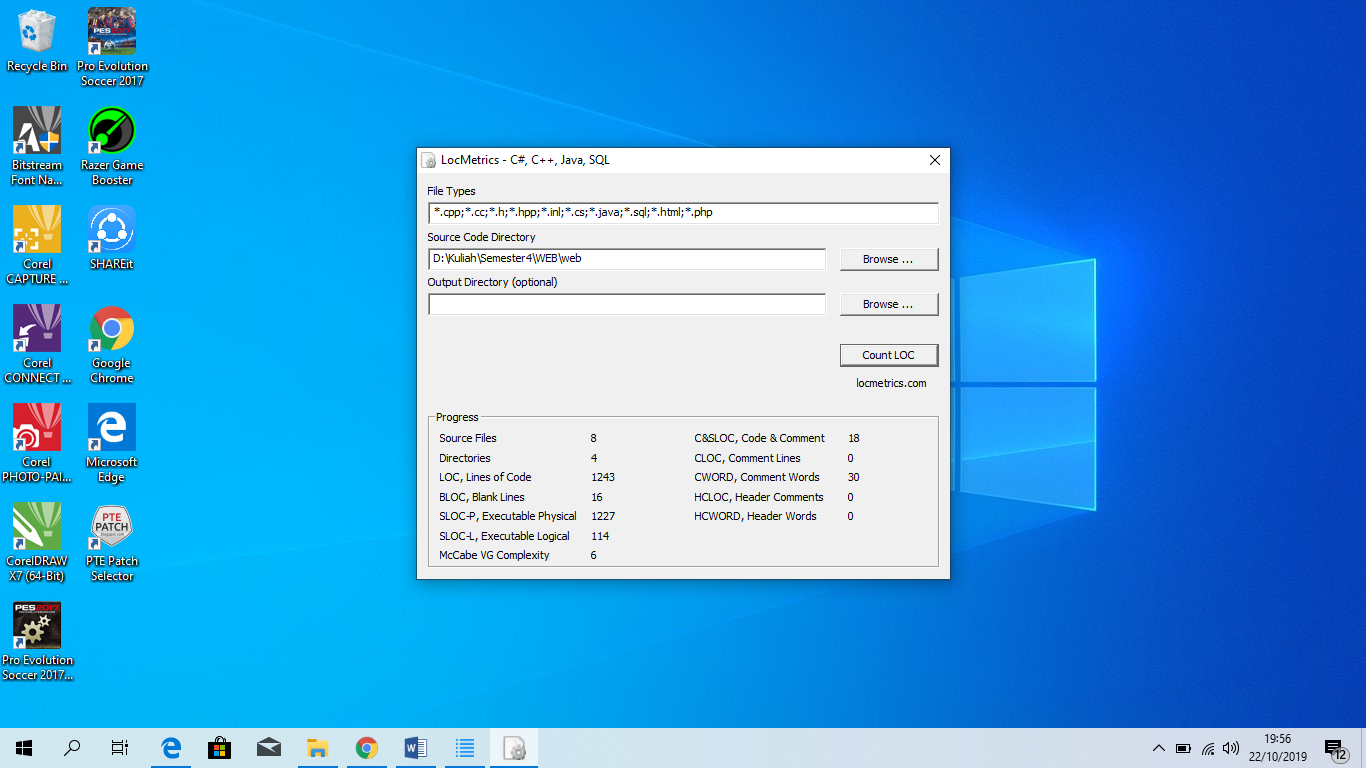
**PEMBAHASAN**

1. **Aplikasi PARADISE HOTEL**

Dalam laporan ini, kasus yang diambil adalah sebuah aplikasi berbasis web. Sebuah sistem Reservasi Hotel, aplikasi ini secara spesifik dibangun untuk memudahkan para traveller untuk mencari hotel dan informasi-informasi mengenai hotel termasuk fasilitas dan juga harga.

Melalui aplikasi ini, para traveller tidak perlu mencari hotel saat sudah tiba di lokasi namun sebelum berada dilokasi para traveller sudah bisa memboking kamar sesuai keinginan dan untuk memudahkan para traveller saat melakukan liburan.

Aplikasi ini dibangun dengan Bahasa PHP dan HTML. Ukuran asli dari proyek ini adalah 1243 baris kode dalam Bahasa PHP dan HTML dari total 59 file. Termasuk baris komentar pada kode juga dihitung dikarenakan komentar sangat penting. Gambar 1 menampilkan metrik kode sumber yang dikumpulkan dengan perangkat lunak *LocMetrics*



Gambar 1. Metrik Kode Sumber

1. **Penerapan FPA pada Proyek yang Dibangun**

Pertama-tama, *User Function* (UF) dari proyek harus dibuat dan diklasifikasikan sesuai tingkat kompleksitasnya. Dalam metode FPA terdapat 5 fungsi sebagai parameter pengukuran sebuah perangkat lunak, yaitu *External Input (EI), External Output (EO), Internal Logical File (ILF), External Interface File (EIF) dan External Inquiry (EQ).*

* *External Input* (EI) adalah proses dasar yang memproses data dan informasi kontrol yang datang dari luar batasan aplikasi.
* *External Output* (EO) adalah sebuah proses dasar dimana hasil data dilewatkan dari dalam ke keluar dari batasan aplikasi.
* *Internal Logical File* (ILF) adalah kelompok data atau kelompok informasi kontrol yang digunakan dalam aplikasi.
* *External Interface File* (EIF) adalah kelompok data berelasi atau informasi kontrol yang dirujuk oleh aplikasi, tapi dipelihara oleh aplikasi lain.
* *External Inquiry* (EQ) fungsi utamanya adalah menyediakan informasi ke user melalui pengambilan/pemrosesan data atau informasi kontrol dari ILF/EIF.

Setiap UF diklasifikasikan berdasarkan tingkat kompleksitas antara lain *height, medium* dan *low*. Nilai dari setiap UF pada masing masing kategori kompleksitas dikalikan dengan *complexity weight* yang telah ditentukan pada Tabel 1

Tabel 1. Complexity

|  |  |  |  |
| --- | --- | --- | --- |
| **User Function Types** | **Complexity Weight** | | |
| **Low** | **Medium** | **Height** |
| External Input | 3 | 4 | 6 |
| External Output | 4 | 5 | 7 |
| Internal Logical File | 7 | 10 | 15 |
| External Interface File | 5 | 7 | 10 |
| External Inquiry | 3 | 4 | 6 |

Terdapat 4 *External Input*, 2 *External Output*, 3 *Internal Logical File*, 0 *External Interface File* dan 5 *External Inquiry* dalam proyek ini yang tertera pada Tabel 2, 3, 4 dan 5.

Tabel 2. External Input

|  |  |  |
| --- | --- | --- |
| **No.** | **Deskripsi** | **Kompleksitas** |
| 1 | Halaman Login | Medium |
| 2 | Room Online | Height |
| 3 | Reservation | Height |
| 4 | Contact | Medium |

Tabel 3. External Output

|  |  |  |
| --- | --- | --- |
| **No.** | **Deskripsi** | **Kompleksitas** |
| 1 | Tampil Pilihan | Medium |
| 2 | Book Reservation | Height |

Tabel 4. Internal Logical File

|  |  |  |
| --- | --- | --- |
| **No.** | **Deskripsi** | **Kompleksitas** |
| 1 | Gambar Hotel | Medium |
| 2 | Database Tabel Reservation | Height |
| 3 | Database Tabel Login | Medium |

Tabel 5.External Inquiry

|  |  |  |
| --- | --- | --- |
| **No.** | **Deskripsi** | **Kompleksitas** |
| 1 | Proses Login | Medium |
| 2 | Proses Cek Room Online | Height |
| 3 | Proses Reservation | Height |
| 4 | Proses Input Pilihan | Height |
| 5 | Proses Input Contact | Height |

Selanjutnya penghitungan nilai UFP dilakukan dengan mengalian UF dan nilai kompleksitas masing-masing lalu menjumlahkan keseluruhan nilai yang menghasilkan angka 95. Perhitungan UFP dapat dilihat pada Tabel 6.

Tabel 6. Perhitungan Nilai UFP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User Function Types** | **Complexity Weight** | | | |
| **Low** | **Medium** | **Height** | **Total** |
| External Input (EI) | 3x0 | 4x2 | 6x2 | 20 |
| External Output | 4x0 | 5x1 | 7x1 | 12 |
| Internal Logical File | 7x0 | 10x2 | 15x1 | 35 |
| External Interface File | 5x0 | 7x0 | 10x0 | 0 |
| External Inquiry | 3x0 | 4x1 | 6x4 | 28 |
| Unadjusted Function Point (UFP) | | | | 95 |

Setelah menghitung UFP, selanjutnya adalah menghitung nilai *Total Degree of Influence* (TDI), hasil TDI bisa dilihat di Tabel 7. Ada 14 faktor yang dibuat mempengaruhi tingkat kesulitan yang berhubungan dengan implementasi sistem. Setiap faktor diberi nilai dari 0 sampai 5. Nilai 0 jika faktor tersebut tidak menimbulkan efek apapun dan 5 jika faktor tersebut sangat penting di perangkat lunak yang diukur. Dalam FPA, faktor tersebut disebut *Value Adjustment Factor* seperti terlihat pada Tabel 7.

Tabel 7. *Value Adjustment Factors*

|  |  |  |
| --- | --- | --- |
| **ID** | **Factor** | **DI** |
| C1 | Data communications | 4 |
| C2 | Distributed function | 4 |
| C3 | Performance objectives | 5 |
| C4 | Heavily used configuration | 3 |
| C5 | Transaction rate | 4 |
| C6 | On-line data entry | 4 |
| C7 | End-user efficiency | 5 |
| C8 | On-line update | 4 |
| C9 | Complex processing | 5 |
| C10 | Reusability | 5 |
| C11 | Installation ease | 3 |
| C12 | Operational ease | 3 |
| C13 | Multiple sites | 3 |
| C14 | Facilitate change | 4 |
| Total Degree of Inluence | | 56 |

Ketika keseluruhan faktor dan skor masing-masing telah ditentukan, selanjutnya adalah menghitung nilai *Total Degree of Influence* (TDI) dengan menjumlahkan ke 14 faktor tersebut. Lalu TDI diubah menjadi nilai akhir *Technical Complexity Adjustment* (TCA) dengan formula (1).

TCA = 0.65 + 0.01 x TDI

TCA = 0.65 + 0.01 x 56 = 1.21 (1)

Akhirnya FP dari sistem didapatkan dengan mengalikan nilai UFP dengan TCA seperti pada formula (2).

FP = UFP x TCA

FP = 95 x 1.21 = 114.95 (2)

Untuk mendapatakan nilai estimasi ukuran dari sistem dalam *Lines of Codes* (LOC), maka nilai FP dikalikan dengan 53 yang merupakan *productivity factor* dari bahasa pemrograman PHP. *Productivity Factor* adalah jumlah kode logis per function point dan nilainya bervariasi untuk setiap bahasa pemrogramanan, seperti yang dibuat oleh Capers Jones pada Tabel 8.

Tabel 8. *Productivity Factor Programming Language*

|  |  |
| --- | --- |
| **Programming Language** | **Productivity Factor** |
| SQL | 37 |
| PHP | 53 |
| HTML/Javascript | 58 |

sehingga LOC didapatkan seperti pada formula (3).

LOC = FP x PF

LOC = 114.95 x 53 = 6092 (3)

Ukuran proyek telah diestimasi dengan nilai estimasi 6092 baris kode dengan menggunakan metode FPA. Setelah proyek selesai, ukuran asli dari proyek adalah 1243 baris kode. Jika nilai estimasi dibandingkan dengan nilai ukuran asli, maka perbedaannya adalah sekitar 0,08044%.

**BAB III**

**PENUTUP**

1. **KESIMPULAN**

Kode sumber pada proses pengembangan perangkat lunak memiliki atribut. Atribut tersebut antara lain adalah kloning, kompleksitas dan LOC. Ketiga atribut tersebut memiliki pengaruh terhadap kemungkinan terjadinya cacat pada perangkat lunak yang dikembangkan.

Nilai pengaruh masing-masing atribut kode tersebut dapat dihitung dengan menggunakan pendekatan statistika. Metode tersebut bernama regresi dan korelasi, dimana metode tersebut dapat mengetahui nilai pengaruh antara variabel bebas (yang mempengaruhi) dengan variabel terikat (yang dipengaruhi).

Metode FPA telah diaplikasikan untuk menghitung estimasi ukuran perangkat lunak pada aplikasi berbasis web yang dikembangkan oleh penulis serta hasil perhitungan telah dievaluasi.

Hasil perhitungan yang didapat yaitu 0,08044%. Bagaimanapun juga akan selalu didapatkan perbedan antara estimasi dan ukuran sesunguhnya dikarenakan proses identifikasi jenis fungsi dan klasifikasi kompleksitas adalah hal yang sulit. Sehingga perlu pendekatan yang lebih baik untuk mengantisipasi perbedaan antara nilai estimasi dan ukuran aslinya.