# METODE ESTIMASI *EFFORT* PERANGKAT LUNAK *USE CASE POINTS* DENGAN PEMILIHAN BOBOT KOMPLEKSITAS *USE CASE* MENGGUNAKAN OPTIMASI METAHEURISTIK

## LAPORAN

### SKEMA PENELITIAN DASAR


**Tim Peneliti**

| | | |
|---|---|---|
| Ardiansyah | NIDN 0523077902 | Ketua |
| Mulki Indana Zulfa | NIDN 0408128602 | Anggota |
| Ali Tarmuji | NIDN 0014107301 | Anggota |

**PROGRAM STUDI S1 INFORMATIKA**
**FAKULTAS TEKNOLOGI INDUSTRI**
**UNIVERSITAS AHMAD DAHLAN**

**2023**

# DAFTAR ISI

# DAFTAR GAMBAR

# DAFTAR TABEL

# ABSTRAK

Estimasi effort merupakan suatu kegiatan penting dalam proyek pengembangan perangkat lunak. Estimasi dilakukan untuk mengetahui seberapa banyak sumber daya yang akan digunakan dan berapa lama waktu yang diperlukan untuk menyelesaikan sebuah proyek. Salah satu metode estimasi effort perangkat lunak yang dapat digunakan adalah *Use Case Points* (UCP). Metode estimasi effort *Use Case Point* (UCP) memiliki kelebihan mudah diimplementasikan untuk mengetahui estimasi awal proyek, prosedur perhitungannya didefisinikan dengan baik, performanya lebih baik dibanding penilaian pakar dan tidak selalu membutuhkan data histori proyek untuk estimasi. Sedangkan kelemahannya adalah berupa ketidakpastian faktor biaya dan penentuan klasifikasi yang memiliki perbedaan nilai pengali yang cukup tinggi. Oleh Karena itu, penelitian ini akan menggunakan algoritman *Grey Wolf Optimizer* (GWO) untuk menemukan bobot kompleksitas use case sehinggan diharapkan bisa meningkatkan performa akurasi estimasi UCP.

Dalam penelitian ini dataset yang akan digunakan adalah dataset Silhavy, Radek (2017), yang terdiri dari 71 data proyek yang telah diselesaikan. Teknik validasi yang digunakan adalah *Leave One Out Cross Validation* (LOOCV). Sedangkan teknik evaluasi yang akan digunakan adalah *Mean Absolute Error* (MAE). Penelitian ini melakukan optimasi pada pembobotan kompleksitas *use case* pada metode estimasi *effort* perangkat lunak Use Case Points (UCP) menggunakan algoritma Grey Wolf Optimizer (GWO) yang diberi nama UCP+GWO. Kompleksitas *use case* yang dioptimasi adalah *simple, average*, dan *complex* yang masing-masing memiliki rentang nilai yaitu [5.00, 7.49], [7.50, 12.49], dan [12.5, 15.00]. Berdasarkan eksperimen diperoleh hasil bahwa MAE terbaik diperoleh oleh UCP+GWO yaitu sebesar 1070.65, sedangkan yang diperoleh UCP standar yaitu 1806.55. Berdasarkan uji statistik Wilcoxon didapatkan bahwa terdapat perbedaan signifikan antara kedua model UCP dan UCP+GWO. Sehinga bisa disimpulkan bahwa UCP+GWO berhasil meningkatkan performa akurasi metode estimasi *effort* Use Case Points.

**Kata Kunci:** *estimasi effort, use case points, grey wolf optimizer, optimasi, dataset, kompleksitas use case*

# BAB I

# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Perangkat lunak merupakan abstraksi fisik yang dimana memungkinkan kita untuk berbicara dengan mesin perangkat keras. Tanpa adanya perangkat lunak, maka perangkat keras yang telah diciptakan tidak akan dapat berguna atau berfungsi dengan optimal. Dalam pengembangan perangkat lunak tidak cukup membutuhkan waktu yang singkat karena memiliki tahapan yang harus dilalui. Untuk menghasilkan perangkat lunak yang berkualitas dalam pengembangannya perlu adanya manajemen proyek dengan mengikuti pola *Software Development Life Cycle* (SDLC) yang bertujuan untuk membangun suatu perangkat lunak agar berjalan sesuai dengan apa yang terjadi dari awal hingga akhir seperti perencanaan, analisis sistem, desain, implementasi dan pemeliharaan sistem. Salah satu kegiatannya melakukan estimasi [1].

Estimasi Effort merupakan suatu kegiatan yang paling penting untuk melakukan pengembangan perangkat lunak dalam memprediksi mengenai pengaturan sumber daya dan berapa lama waktu yang diperlukan untuk menyelesaikan sebuah proyek sesuai dengan apa yang telah ditargetkan [2].

Estimasi effort perangkat lunak merupakan salah satu aspek penting untuk menentukan keberhasilan suatu proyek perangkat lunak, dimana di dalamnya termasuk estimasi waktu pengerjaan, biaya dan sumber daya. Banyak proyek perangkat lunak yang gagal dikarenakan manajemen proyek yang buruk, termasuk dalam perhitungan estimasi perangkat lunak yang buruk. Estimasi yang tepat sangat berpengaruh dalam menentukan keberhasilan suatu proyek. Jika estimasi terlalu rendah dari seharusnya maka akan berakibat pada tidak terselesaikannya proyek perangkat lunak karena tidak sesuai dengan yang telah ditargetkan baik waktu maupun biaya [3].

Selama ini ada beberapa metode estimasi effort perangkat lunak yang dapat digunakan seperti *Function Points Analysis* (FPA), *Agile Software Development* (ASD) dan *Use Case Point* (UCP)[1]. *Function Points Analysis* (FPA) adalah metode standar untuk mengukur pengembangan perangkat lunak dari sudut pandang pengguna[4]. *Agile Software Development* (ASD) adalah metode manajemen untuk pengembangan dalam sebuah proyek untuk peningkatan berkelanjutan dalam perangkat lunak[5]. *Use Case Point* (UCP) adalah sebuah estimasi awal berdasarkan dari use case diagram yang dapat digunakan untuk memahami permasalahan yang mungkin terjadi pada perangkat lunak, estimasi ukuran proyek dan arsitektur umum sistem [3].

Pada penelitian ini *Use Case Points* (UCP) yang akan digunakan. Estimasi dalam metode UCP dimulai dengan melakukan pengukuran kompleksitas dari *actor (Unadjustes Actor Weight)* dan *Use Case (Unadjustes Use Case Weight)* yang ada di dalam sistem. Selanjutnya dilakukan pengukuran terhadap kompleksitas teknis *(Techinical Complexity Factor)* dan *environtment (Environtmental Factor)* pada sistem yang digunakan untuk pengembangan proyek perangkat lunak. Setelah itu baru dilakukan pengukuran terhadap *Use Case Point* (UCP), Faktor Produktifitas dan Estimasi Effort [3].

*Use Case Points* (UCP) memiliki kelebihan yaitu mudah diimplementasikan untuk mengetahui estimasi usaha di awal proyek, prosedur perhitungannya didefinisikan dengan baik, performanya lebih baik dibanding penilaian pakar, dan tidak membutuhkan data histori proyek untuk estimasi [6]. Sedangkan kelemahannya berupa ketidakpastian faktor biaya dan penentuan klasifikasi yang dimana memiliki perbedaan nilai pengali yang cukup tinggi. Keterbatasan atau kelemahan itu cukup mempengaruhi keakuratan estimasi yang dihasilkan. Terdapat perbedaan nilai pengali cukup tinggi dari level kompleksitas tiap *Use Case*. Dengan kata lain, untuk menghitung faktor kompleksitas *Use Case* dan menggunakan *Neural Network* untuk dapat memetakan data input berupa *Use Case* dan *Actor* yang terlibat[1]. Oleh karena itu, dilakukan usaha lanjutan dalam menyesuaikan optimasi bobot kompleksitas *Use Case* yang tepat agar dapat meningkatkan performa akurasi estimasi effort perangkat lunak UCP

menjadi lebih baik. Dimana untuk melakukan pendekatan yang bisa diterapkan untuk memilih optimasi bobot kompleksitas yang tepat dengan menggunakan algoritma optimasi metaheuristik [7].

Algoritma *Grey Wolf Optimizer* (GWO) merupakan algoritma metaheristik baru yang mampu memberikan hasil kompetitif, dimana eksplorasi *search* spesiesnya yang lebih luas dan juga dapat untuk menghindari terjebaknya lokal optimum[8]. Oleh karena itu, penelitian ini bertujuan untuk mencari nilai optimasi bobot kompleksitas *Use Case* yang tepat pada metode estimasi effort perangkat lunak *Use Case Points* untuk bisa meningkatkan performa akurasinya [7].

## 1.2 Rumusan Masalah

Berdasarkan uraian latar belakang diatas adalah bagaimana peningkatan akurasi estimasi *Use Case Points* (UCP) setalah dilakukan optimasi terhadap parameter bobot kompleksitas *Use Case* dengan menggunakan *Grey Wolf Optimizer* (GWO).

## 1.3 Tujuan Penelitian

Tujuan penelitian ini tentang optimasi bobot kompleksitas *Use Case* pada estimasi effort perangkat lunak menggunakan *Grey Wolf Optimizer* (GWO) adalah untuk mengoptimalkan penggunaan bobot yang tepat dalam metode Estimasi *Use Case Points* (UCP). Serta dalam penelitian ini bertujuan untuk meningkatkan akurasi estimasi usaha perangkat lunak yang mempertimbangkan kompleksitas masing – masing *Use Case* secara lebih efektif. Berikut ini beberapa tujuan khusus yang dapat diidentifikasi dalam penelitian ini:

1. Menentukan setting parameter yang tepat untuk algoritma *Grey Wolf Optimizer* (GWO).
2. Membuat program untuk mengimplementasikan algoritma *Grey Wolf Optimizer* (GWO) pada metode *Use Case Points* (UCP).
3. Mengevaluasi performa *Grey Wolf Optimizer* (GWO) dengan *tuning* parameter yang berbeda.

## 1.4 Manfaat Penelitian

Penelitian ini tentang optimasi bobot kompleksitas *Use Case* pada estimasi effort perangkat lunak menggunakan *Grey Wolf Optimizer* (GWO) yang memiliki beberapa manfaat yang potensial, termasuk:

1. Meningkatkan akurasi estimasi effort dengan optimasi bobot kompleksitas *Use Case* dalam memberikan pemahaman tentang pengaruh terhadap keakuratan estimasi effort perangkat lunak.
2. Menganalisis untuk mengetahui pengaruh dalam pemilihan optimasi bobot kompleksitas *Use Case* pada estimasi effort perangkat lunak UCP menggunakan algoritma GWO.
3. Mengetahui hasil evaluasi perbandingan untuk pemilihan *Use Case* dengan optimasi bobot kompleksitas *Use Case* dalam memperhitungkan kompleksitas masing – masing *Use Case.*
4. Dan bisa digunakan dalam penelitian selanjutnya untuk mengetahui optimasi bobot kompleksitas *Use Case.*

# BAB II

# TINJAUAN PUSTAKA

## 2.1 Kajian Penelitian Terdahulu

Penelitian ini tidak terlepas dari penelitian – penelitian terdahulu yang pernah dilakukan yang tentu saja berkaitan dengan penelitian yang dikerjakan saat ini.

Penelitian pertama yang dilakukan oleh Habib Rahman M. Akbar Putra, Widhy Hayuhardika Nugraha Putra, Djoko Pramono tentang Estimasi Biaya Perangkat Lunak Menggunakan Metode *Use Case Points* dalam Studi Kasus PT. PLN (Persero) Area Malang. Penelitian ini dilakukan untuk mencari perhitungan estimasi waktu yang belum terstandar dengan baik membuat beban biaya proyek bisa menjadi lebih besar ataupun lebih kecil. Dan penggunaan metode *Use Case Points* dapat dijadikan salah satu standarisasi untuk perhitungan estimasi waktu sebuah proyek, yang dimana dapat menghasilkan tentang estimasi waktu yang dibutuhkan dalam pengembangan sebuah proyek. Sehingga dapat mendapatkan nilai total estimasi biaya yang dibutuhkan [9].

Penelitian kedua yang dilakukan oleh Puguh Jayadi, Alim Citra Aria Bima, Yoga Prisma Yudha, Kelik Sussolaikah tentang *End User Development* Pada *Use Case Points* Untuk Peningkatan Estimasi Perangkat Lunak. Penelitian ini dilakukan untuk menentukan jumlah waktu dan orang yang terlibat dalam suatu proyek perangkat lunak. Yang dimana hasil penelitian pengguna (*End User)* memiliki peran penting dalam pengembangan perangkat lunak untuk membantu meningkatkan estimasi pengembangan perangkat lunak dengan mengambil kualitas teknis EUD_TCF dan lingkungan EUD_ECF sebagai faktor tambahan dalam perhitungan estimasi. Dimana pembobotan serta perhitungan estimasi dalam pengembangan proyek system PIAUD dengan metode AUCP *(Advance Use Case Points)* menghasilkan nilai akurasi hamper sama dengan *Effort Of Hour* sebenarnya [10].

Penelitian selanjutnya yang dilakukan oleh Rika Priyanti Manik, Andi Reza Perdana Kusuma, Mochamad Chandra Saputra tentang Evaluasi Biaya Perangkat Lunak Menggunakan

Metode *Fuzzy Use Case Point. Software House* menciptakan bermacam – macam perangkat lunak secara terus – menerus dalam waktu dan biaya yang terbatas. Oleh karena itu, dibutuhkan sebuah perencanaan anggaran yang baik, yaitu dengan melakukan estimasi biaya. Penelitian ini dilakukan untuk membandingkan hasil perhitungan estimasi biaya yang dilakukan *Software House* menggunakan metode *Use Case Points* dan metode *Fuzzy Use Case Points* [11].

Penelitian selanjutnya yang dilakukan oleh Dhya Fairuzu Zahiroh, Mochamad Chandra Saputra, Admaja Dwi Herlambang tentang Perbandingan Evaluasi Biaya Pengembangan Sistem Antrian RSUD Dr Soetrasno Rembang Menggunakan Metode *Use Case Points* Dan *Function Points* Studi Kasus CV Pabrik Teknologi. Penelitian ini dilakukan untuk membagi ruang lingkup menggunakan *Gantt Chart* serta perhitungan estimasi biaya menggunakan metode *Use Case Points* dan metode *Function Points.* Untuk menerjemahkan kebutuhan fungsional perangkat lunak yang dapat dilakukan perhitungan estimasi effort dan biaya total. Menggunakan perhitungan *Use Case Points* dalam pendekatan yang baik untuk memperkirakan aktivitas apa saja yang terjadi selama pengembangan *Software* menggunakan *Use Case Diagram* serta *Use Case Scenario* sehingga menghasilkan *Hours Of Effort* dan biaya total. Sedangkan *Function Points* menggunakan pendekatan berorientasi pada fungsionalitas *Software* dalam menganalisi estimasi usaha *Software* yang kemudian digunakan untuk menganalisis estimasi biaya menggunakan *Data Flow Diagram* [12].

Penelitian selanjutnya dilakukan oleh Vidiyanti Lestari, Ahmad Kamsyakawuni, Kiswara Agung Santoso tentang Implementasi Algoritma *Grey Wolf Optimizer* (GWO) Di Toko Citra Tani Jember. Penelitian ini dilakukan terkait optimasi didefinisikan sebagai proses untuk menentukan nilai minimum atau maksimum yang bergantung pada fungsi dan tujuannya. Dan banyak ditemukan permasalahan yang menyangkut permasalahan optimasi. Maka diusulkan algoritma *Grey Wolf Optimizer* (GWO) untuk menyelesaikan masalah *knapsack.* Dan disimpulkan algoritma *Grey Wolf Optimizer* (GWO) dapat menyelesaikan permasalah *knapsack* yang dimana sebagai pembanding dengan metode simpleks hasilnya [8].

Berdasarkan kajian – kajian penelitian terdahulu diatas yang telah dijelaskan. Penelitian saat ini adalah Optimasi Bobot Kompleksitas *Use Case* pada Estimasi Effort Perangkat Lunak *Use Case Points* Menggunakan *Grey Wolf Optimizer.* Diharapkan dengan adanya penelitian ini dapat menghasilkan nilai optimasi bobot kompleksitas *Use Case* yang tepat pada metode estimasi effort perangkat lunak *Use Case Points* untuk bisa meningkatkan performa akurasinya.

Dari kajian–kajian terdahulu di atas, maka ditampilkan tabel perbandingan penelitian yang dapat dilihat ditabel 2.1 yang berisi rangkuman dari beberapa paper jurnal penelitian terdahulu.

*Tabel 2. 1 Kajian Penelitian Terdahulu*

| **Penelitian** | **Dataset** | **Variable** | **Metode** | **Hasil** |
|---|---|---|---|---|
| [9] | Dataset PLN Area Malang | Waktu, Sumber Daya dan Biaya | *Use Case Points* | Penelitian ini menghasilkan kesimpulan bahawa proyek Pengembangan Sistem Informasi Konstruksi Tiang PT. PLN (Persero) Area Malang menghasilkan nilai estimasi waktu selama 411,04 jam atau 2,5 bulan. Pada alokasi aktual (Guesstimate) yang dilakukan perusahaan mendapatkan nilai 640 jam atau 4 bulan. Sehingga didapatkan nilai selisih estimasi waktu selama 228,96 jam atau 1,5 bulan. Sedangkan hasil estimasi biaya yang dihitung menghasilkan nilai estimasi biaya total sebesar Rp20.383.062,50;. Pada alokasi aktual |

| | | | | |
|---|---|---|---|---|
| | | | | (Guesstimate) yang dilakukan perusahaan mendapatkan biaya total sebesar Rp27.324.500,00;. Sehingga didapatkan nilai selisih estimasi biaya total sebesar Rp6.941.000,00;. Hasil estimasi tersebut didapat dikarenakan berbagai parameter yang berbeda seperti, perbedaan alokasi sumberdaya antara kondisi aktual dengan penerapan dari metode UCP ini, hal tersebut terjadi dikarenakan pada penerapannya tidak mencakup mengenai pengalokasian sumberdaya yang diperlukan, kemudian perbedaan standar gaji yang digunakan peneliti dan standar gaji yang digunakan perusahaan yang otomatis berdampak pada hasil estimasi biaya. |

| | | | | |
|---|---|---|---|---|
| [10] | Dataset PIAUD | Waktu dan Sumber Daya | *Use Case Points* | Sesuai dengan hasil penelitian, pengguna *(End User)* memiliki peran penting dalam pengembangan perangkat lunak. Dimana pengguna memiliki kemampuan untuk mengembangkan perangkat lunak sendiri, yang dikenal sebagai *End User Development* (EUD). EUD yang melibatkan pengguna dalam pengembangan perangkat lunak dengan memberikan masukan dan perubahan pada perangkat lunak yang sedang dikembangkan. Hal ini membantu meningkatkan estimasi pengembangan perangkat kualitas teknis EUD_TFC dan lingkungan EUD_ECF sebagai faktor tambahan dalam perhitungan estimasi. Pembobotan serta perhitungan estimasi dalam pengembangan proyek system PIAUD dengan |

| | | | | |
|---|---|---|---|---|
| | | | | metode *Advance Use Case Points* menghasilkan nilai akurasi hamper sama dengan *Effort Of Hour* sebenarnya. |
| [11] | Dataset PT. ABC | Waktu, Sumber Daya dan Biaya | *Fuzzy Use Case Points* | Berdasarkan hasil penelitian bisa disimpulkan estimasi waktu yang menggunakan metode Fuzzy Use Case Point pada SIPAS adalah 91,7 jam, pada SMTPX adalah 129,84 jam. Kemudian estimasi biaya menggunakan metode Fuzzy Use Case Point pada SIPASadalah sebesar Rp 46.493.500,pada SMTPX adalah Rp 76.618.750. Maka hasil penjadwalan SIPAS dan SMTPX berdasarkan WBS membutuhkan alokasi 26 staf, waktu 91,87 jam, dan biaya sebesar Rp 46.493.500. Sedangkan pada SMTPX, membutuhkan alokasi 26 staf, waktu 129,84 jam, dan biaya sebesar Rp Rp 76.618.750. |

| | | | | |
|---|---|---|---|---|
| Dhya Fairuzu Zahiroh, Mochamad Chandra Saputra, Admaja Dwi Herlambang | Dataset CV Pabrik Teknologi | Waktu, Sumber Daya dan Biaya | *Use Case Points dan Function Points* | Hasil penjadwalan pengerjaan proyek system antrian RSUD Dr. Soetrasno Rembang menggunakan Gantt Chart berdasarkan Work Breakdown Structure (WBS) menghasilkan Gantt Chart untuk Guestimate dengan tiga fase: 1) Define Project Goal; 2) Plan Project; 3) Execute Project Plan. Dan mendapatkan hasil perbandingan yang dimana hasil penerapan metode Use Case Point menghasilkan keluaran berupa hours of effort selama 8.950 jam kerja dan biaya total sebesar Rp 157.843.750,00. Sedangkan hasil penerapan metode Function Point menghasilkan keluaran berupa estimasi effort (usaha) yaitu 24 orang dengan durasi pengerjaan selama 9 bulan dan biaya total sebesar Rp 200.650.000,00. |

| | | | | |
|---|---|---|---|---|
| Vidiyanti Lestari, Ahmad Kamsyakawuni, Kiswara Agung Santoso | Dataset Toko Citran Tani | Konstanta populasi dan konstanta jumlah iterasi | *Grey Wolf Optimizer (GWO)* | Solusi terbaik yang didapatkan oleh algoritma Grey Wolf Optimizer (GWO) dalam menyelesaikan permasalahan multiple constraints bounded knapsack adalah sebesar Rp 2.704.700,-. Solusi terbaik tersebut diperoleh dari profit paling maksimal hasil akhir simulasi data dengan parameter populasi 1000, parameter maksimal iterasi 10000 dan rata – rata iterasi saat Z maksimum local 9877,6. Parameter populasi dan maksimal iterasi dalam penyelesaian masalah ini memiliki pengaruh yang sama untuk mendapatkan hasil yang optimal, dimana semakin besar nilai dari parameter tersebut maka hasil yang didapatkan juga semakin mendekati nilai optimal. |

| | | | | |
|---|---|---|---|---|
| Penelitian saat ini | | | *Use Case Points dan Grey Wolf Optimizer* (GWO) | |

## 2.2 Landasan Teori

### 2.2.1 Estimasi Effort Perangkat Lunak

Estimasi effort merupakan salah satu aspek penting untuk menentukan keberhasilan suatu tahapan perencanaan mengembangkan perangkat lunak. Untuk memperkirakan estimasi waktu pengerjaan, biaya dan sumber daya untuk menyelesaikan pengembangan perangkat lunak diperlukanlah estimasi effort untuk mendapatkan hasil akurasinya [3].

### 2.2.2 Use Case Points

*Use Case Points* (UCP) adalah estimasi perangkat lunak yang digunakan untuk mengevaluasi perangkat lunak diseluruh proyek pengembangan perangkat lunak. *Use Case Points* (UCP) digunakan sesuai dari kebutuhan dari sistem yang dijelaskan dalam sebuah *Use Case*, yang mana itu merupakan bagian dari Teknik pemodelan UML. Ukuran perangkat lunak dihitung berdasarkan elemen dari *Use Case* yang dipertimbangkan dari aspek teknis dan lingkungan. Untuk proyek perangkat lunak *Use Case Points* (UCP) dapat digunakan untuk menghitung estimasi usaha untuk proyek tersebut. Umumnya untuk perencanaan dan implementasi proyek tergantung pada kompleksitasnya *Use Case* dan waktu penyelesaian proyek mungkin terpengaruh oleh beberapa hal berikut:

1. Jumlah langkah yang diperlukan untuk menyelesaikan *Use Case.*
2. Jumlah dan kompleksitas transaksi pada *Actor*.
3. Persyaratan teknis dari *Use Case* seperti konkurensi, keamanan dan kinerja.
4. Faktor lingkungan seperti pengalaman dan pengetahuan tim [3].

#### 2.2.2.1 Unadjusted Use Case Point (UUCP)

Perhitungan *Unadjusted Use Case Point* (UUCP) didapatkan dari penjumlahan kompleksitas *Unadjusted Use Case Weights* (UUCW) dengan *Unadjusted Actor Weights* (UAW).

### a. Unadjusted Actor Weights (UAW)

Langkah pertama untuk mendapatkan nilai UAW dilakukan pengklasifikasian *actor* menjadi beberapa kategori yaitu *simple, average,* dan *complex.* Tabel klasifikasi *actor* dapat dilihat pada tabel 2.2 dibawah.

*Tabel 2. 2 Klasifikasi Actor*

| Kategori | Deskripsi | Bobot |
|---|---|---|
| *Simpel* | *Actor* berinteraksi dengan sistem menggunakan (API) | 1 |
| *Average* | *Actor* berinteraksi dengan sistem melalui protocol, seperti TCP/IP | 2 |
| *Complex* | *Actor* berinteraksi dengan sistem menggunakan GUI atau halaman internet | 3 |

Total nilai *Unadjusted Actor Weights* (UAW) didapat dari menghitung berapa banyak (total) *actor* dari masing-masing tipe (tingkat kompleksitas) dikali dengan bobot masing masing tipe sesuai dengan tabel.

$$\boldsymbol{UAW = Total\ Actor\ x\ Bobot\ Actor} \quad \textbf{..... (1)}$$

### b. Unadjusted Use Case Weights (UUCW)

Langkah pertama untuk menentukan kategori *Use Case* apakah *simple, medium* dan *complex.* Tergantung dari jumlah transaksi yang dilakukan dalam deskrispsi *Use Case* pada tabel 2.3 dibawah.

*Tabel 2. 3 Kompleksitas Use Case*

| Kategori | Jumlah Transaksi | Bobot | Bobot Algen |
|---|---|---|---|
| *Simple* | 1 sampai 3 transaksi | 5 | 5-7.49 |
| *Average* | 4 sampai 7 transaksi | 10 | 7.5-12.49 |
| *Complex* | 8 sampai atau lebih | 15 | 12.5-15 |

Total nilai *Unadjusted Use Case Weights* (UUCW) didapat dari menghitung berapa banyak (total) *use case* dari masing-masing tipe (tingkat kompleksitas) dikali dengan bobot masing-masing tipe sesuai dengan tabel.

$$\boldsymbol{UUCW = Total\ Use\ Case * Bobot} \quad \textbf{..... (2)}$$

Setelah nilai UAW dan UUCW didapatkan, selanjutnya melakukan perhitungan UUCP. Rumus perhitungan UUCP dibawah ini[2].

$$\boldsymbol{UUCP = UAW + UUCW} \quad \textbf{..... (3)}$$

### 2.2.2.2 Technical Complexity Factor (TCF)

*Technical Complexity Factor* (TCF) adalah salah satu faktor yang diterapkan pada perkiraan ukuran perangkat lunak untuk memperhitungkan pertimbangan teknis sistem. *Technical Complexity Factor* (TCF) ditentukan dengan menetapkan nilai antara 0 (faktor tidak relevan) sampai 5 (faktor penting) untuk masing-masing dari 13 faktor teknis yang tercantum dalam tabel 2.4.

*Tabel 2. 4 Technical Complexity Factor*

| Technical Factor | | Bobot |
|---|---|---|
| 1. | Sistem Terdistribusi | 2 |
| 2. | Performa/Kinerja | 1 |
| 3. | Efisiensi Pengguna | 1 |
| 4. | Pemrosesan Internal Yang Kompleks | 1 |
| 5. | Penggunaan Kembali Kode | 1 |
| 6. | Mudah Untuk Di Install | 0.5 |
| 7. | Mudah Untuk Digunakan | 0.5 |
| 8. | Mudah Digunakan Diberbagai Platform | 2 |
| 9. | Mudah Untuk Diperbaiki | 1 |

| | | |
|---|---|---|
| 10. | Pararel Proses | 1 |
| 11. | Fitur Keamanan Khusus | 1 |
| 12. | Menerima Akses Dari Pihak Ketiga | 1 |
| 13. | Diperlukan Fasilitas Pelatihan Pengguna | 1 |

Nilai-nilai pada *Technical Factor* dikalikan dengan bobot masing-masing, kemudian dijumlah untuk mendapatkan Total *Technical Factor* (TF), lalu digunakan untuk mendapatkan nilai *Technical Complexity Factor* (TCF) dengan rumus dibawah ini[2].

$$\boldsymbol{TCF = 0.6 + (0.01\ x\ TF)} \quad \textbf{..... (4)}$$

### 2.2.2.3 Environmental Complexity Factor (ECF)

*Environmental Complexity Factor* (ECF) adalah faktor lain yang diterapkan untuk melakukan perhitungan pertimbangan faktor lingkungan dari sistem. *Environmental Complexity Factor* (ECF) ditentukan dengan menetapkan nilai antara 0 (tidak ada pengalaman) sampai 5 (ahli) untuk masing-masing dari 8 *Environmental Factor* yang tercantum dalam tabel 2.5.

*Tabel 2. 5 Environmental Factor*

| **Environmental Factor** | | **Bobot** |
|---|---|---|
| 1. | Familiar Dengan UML | 1.5 |
| 2. | Pengalaman Dengan Aplikasi | 0.5 |
| 3. | Pengalaman Dengan *Object Oriented* | 1 |
| 4. | Kemampuan Memimpin Analisi | 0.5 |
| 5. | Motivasi | 1 |
| 6. | Stabilitas Kebutuhan | 2 |
| 7. | Pekerja Yang Paruh Waktu | -1 |
| 8. | Tingkat Kesulitan Bahasa Pemrograman | -1 |

Nilai-nilai pada enviromental factor tersebut dikalikan dengan bobot masing-masing, kemudian dijumlah untuk mendapatkan Total *Enviromental Factor* (EF), lalu digunakan untuk mendapatkan *Enviromental Complexity Factor* (ECF) dengan rumus dibawah ini[2].

$$\boldsymbol{ECF = 1.4 + (-0.03\ x\ EF)} \quad \textbf{..... (5)}$$

Setelah mendapatkan nilai dari UUCP, TCF dan ECF, maka nilai UCP dapat dihitung dengan menggunakan rumus dibawah ini.

$$\boldsymbol{UCP = UUCP \times TCF \times ECF} \quad \textbf{..... (6)}$$

### 2.2.3 Algoritma Grey Wolf Optimizer

*Grey Wolf Optimizer* (GWO) merupakan metode optimasi berbasis *Swarm Intellingence* yang terinspirasi oleh perilaku berburu serigala di alam atau hierarki social dalam mekanisme perburuan dari sekumpulan serigala abu-abu *(Grey Wolf).* Hierarki serigala abu-abu memiliki dominan social yang tinggi dan terbagi menjadi 4 kategori yang dimulai dari hierarki teratas yaitu alpha, beta, delta dan omega. Hierarki ini menjadi pengaruh dalam pengambilan keputusan akhir untuk perburuan. Algoritma *Grey Wolf Optimizer* memiliki kelebihan yang dibandingkan dengan algoritma lain yaitu lebih cepat dalam mencapai nilai konvergen dan mendapatkan hasil yang kualitatif[8].


*Gambar 2. 1 Hierarki Grey Wolf Optimizer (GWO)*

Lalu dibawah ini adalah uraian dari model matematis hierarki social, pelacakan, pengepungan dan penyerangan mangsa.

a. Hierarki Sosial

Untuk memodelkan hierarki sosial serigala abu-abu secara matematis dalam proses GWO ditentukan bahwa solusi terbaik (*fttest solution*) sebagai alpha (α), solusi terbaik kedua dan ketiga diberi nama beta (β) dan delta (ð). Lalu kandidat solusi lainnya disebut omega (ω). *Grey Wolf Optimizer* (GWO) melakukan optimasi yang dipandu oleh (α), (β), (ð), dan (ω) mengikuti di belakang ketiganya.

b. Mengelilingi Mangsa

Seperti yang sudah dijelaskan jika serigala abu-abu mengelilingi mangsanya saat berburu sehingga mangsa tersebut tidak bisa bergerak. Maka disimulasikan kedalam rumus dibawah ini.

$$\boldsymbol{D = \left| C . X_p(t) - X(t) \right|} \quad \textbf{..... (7)}$$

$$\boldsymbol{X(t+1) = X_p(t) - A . D} \quad \textbf{..... (8)}$$

Dari rumus diatas dimana t menunjukan iterasi saat ini, A dan C adalah vektor koefisien, Xp adalah vektor posisi mangsa, dan X menunjukkan vektor posisi serigala abu-abu.

Vektor A dan C dihitung sebagai berikut:

$$\boldsymbol{A = 2_a . r_1 - a} \quad \textbf{..... (9)}$$

$$\boldsymbol{C = 2 . r_2} \quad \textbf{..... (10)}$$

Dimana komponen a menurun secara linear dari 2 menjadi 0 selama iterasi dan r1, r2 adalah vektor acak dalam [0,1].

c. Berburu

Perburuan biasanya dipimpin oleh alpa, kemudian beta dan delta serta juga sesekali berpartisipasi dalam pemburuan. Dan untuk mensimulasikan secara matematis perilaku berburu serigala abu-abu, disini alpha dianggap (kandidat solusi terbaik) serta beta dan dan delta memiliki pengetahuan lebih baik tentang lokasi potensial mangsa. Ada tiga solusi terbaik yang diperoleh sejauh ini dan mewajibkan pencarian lain (termasuk omega) dimana untuk memperbarui posisi mereka untuk

posisi agen pencarian terbaik. Gambaran rumus model matematisnya seperti di bawah ini [13].

$$\boldsymbol{D_a = |C_1.X_a - X| D_\beta = |C_2.X_\beta - X| D_\delta = |C_3.X_\delta - X|} \text{ .....(11)}$$

$$\boldsymbol{X_1 = X_a - A_1.(D_a), X_2 = X_\beta = A_2.(D_\delta), X_3 = X_\delta - A_3.(D_\delta)} \text{ .....(12)}$$

$$\boldsymbol{\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}} \text{ .....(13)}$$

Pelacakan, mengejar, mendekati dan mengacau mangsa sampai berhenti bergerak lalu menyerang mangsa. Adalah detail perilaku serigala abu-abu dalam berburu yang dibagi kedalam beberapa tahapan yaitu melingkari mangsa, berburu, menyerang mangsa. Seperti yang dijelaskan diatas, serigala abu-abu mengelilingi mangsanya selama berburu[8].

Bagaimana *Grey Wolf Optimizer* (GWO) mampu menyelesaikan masalah optimasi, ada beberapa poin yang harus diperhatikan:

1. Hirarki sosial yang ditawarkan untuk membantu GWO untuk solusi yang terbaik yang didapatkan sejauh ini selama iterasi.
2. Mekanisme melingkari mendefinisikan lingkungan berbentuk lingkaran di sekitar solusi yang dapat diperpanjang dimensi yang lebih tinggi sebagai h*yper-sphere.*
3. Parameter acak A dan C membantu kandidat solusi untuk memiliki *hyper*-spheres dengan jari-jari acak yang berbeda.
4. Metode perburuan yang diusulkan memungkinkan solusi kandidat untuk mencari posisi mangsa.
5. Eksplorasi dan eksploitasi dijamin oleh nilai adaptif a dan A.
6. Nilai adaptif parameter a dan A memungkinkan GWO untuk transisi yang mulus antara eksplorasi dan eksploitasi.
7. Dengan menurunkan A, setengah dari perulangan dikhususkan untuk eksplorasi ($|A| \geq 1$) dan setengah lainnya didedikasikan untuk mengeksploitasi ($|A| < 1$).
8. *Grey Wolf Optimizer* hanya memiliki dua parameter utama yakni a dan C.

Lalu ada kemungkinan untuk memadukan mutase dan operator evolusi lainnya untuk meniru seluruh siklus hidup serigala abu-abu. Dan *Pseudo Code* dituliskan seperti dibawah ini[13].

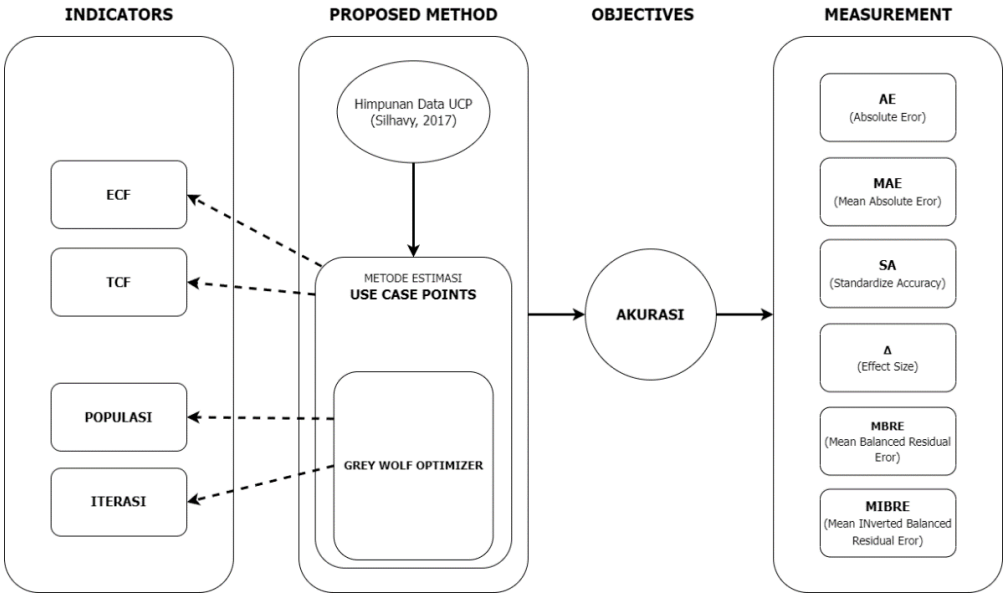
```
Inisiasi populasi awal grey wolf Xi (i = 1,2,…,n);
Initialize a, A, and C
Calculate the fitness of each search agent
Xα = the best search agent
Xβ = the second best search agent
Xδ = the third best search agent
While (1 max number of iterations)
   for each search agent
      Update the position of the current search agent by equation
(13)
   end for
   update a, A, and C
   calculate the fitness of all search agents
   update Xα, Xβ and Xδ
   t=t+1
end while
return Xα
```

# BAB III

# METODOLOGI PENELITIAN

## 3.1 Kerangka Pemikiran Penelitian

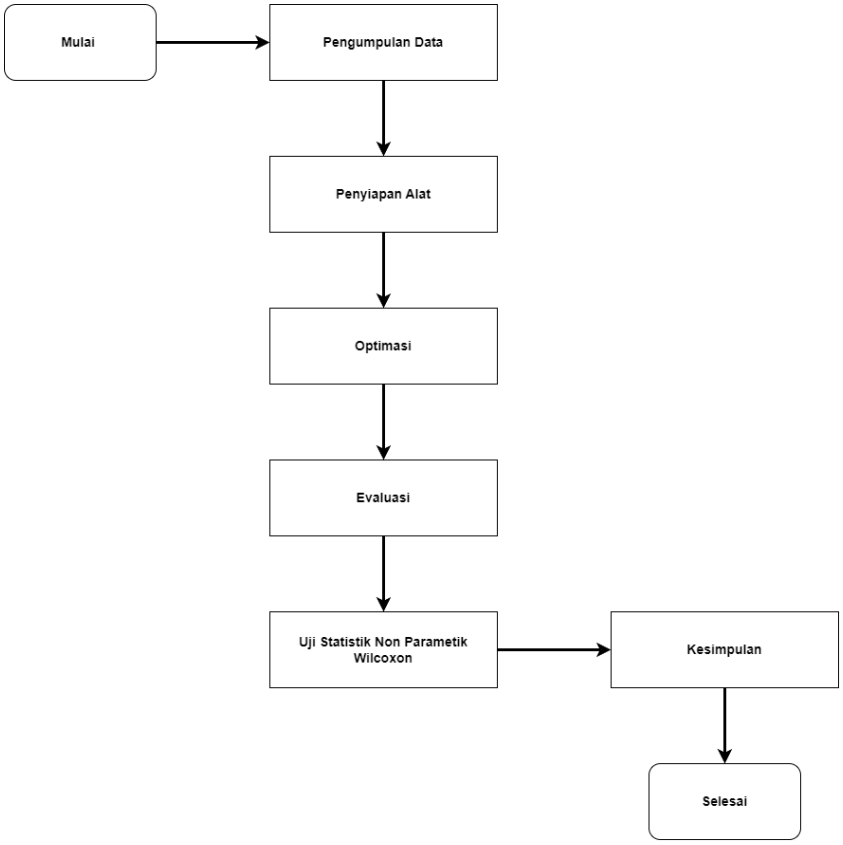
Kerangka Pemikiran Penelitian adalah suatu diagram yang menjelaskan secara garis besar alur logika berjalannya suatu penelitian. Kerangka pemikiran penelitian ini dibagi menjadi empat bagian yaitu *Indicators, Proposed Method, Objectives dan Measurement*. Pada penelitian ini metode yang diusulkan *(Proposed Method)* menggunakan metode *Use Case Points,* dimana pada proses optimasi bobot kompleksitas *Use Case* dibantu dengan menggunakan *Grey Wolf Optimizer.* Lalu indikator *(Indicators)* yang diobservasi adalah optimasi bobot kompleksitas *Use Case.* Tujuan *(Objectives)* pada penelitian ini adalah peningkatan akurasi estimasi effort pada model, dimana pengukuran peningkatan akurasi estimasi *(Measurements)* akan menggunakan *Mean Absolute Error* (MAE), *Mean Balanced Residual Error* (MBRE), *Standardize Accuracy* (SA), *Absolute Error* (AE), *Mean Inverted Balanced Residual Error* (MIBRE) dan *Effect Size* (Δ). Untuk gambaran kerangka pemikiran penelitian dapat dilihat pada gambar 3.1 berikut.




*Gambar 3. 1 Kerangka Pemikiran Penelitian*

## 3.2 Tahapan Penelitian

Penulis penelitian ini menggunakan tahapan penelitian secara sistematis dan urut yang dimulai dari level pengumpulan data, penyiapan alat, optimasi, evaluasi dan uji statistik non parametik wilcoxon dan kesimpulan. Dibawah ini adalah bentuk flowchart tahapannya:




*Gambar 3. 2 Flowchart Tahapan Penelitian*

### 3.2.1 Pengumpulan Data

Pada pengumpulan data penelitian ini usaha yang dilakukan untuk memperoleh data atau dokumen yang dibutuhkan untuk bahan penelitian. Metode yang dilakukan untuk pengumpulan data ini menggunakan metode dokumen. Dokumen adalah tulisan atau catatan kejadian masalalu yang berbentuk lisan, gambar atau karya-karya dari seseorang. Pada penelitian ini dokumen yang akan digunakan adalah dataset publik. Dataset publik yang akan digunakan adalah dataset Silhavy, Radek (2017) yang terdiri dari 71 data proyek yang sudah diselesaikan. Yang meliputi perhitungan *Effort Multipiler* (EM), *Scale Factor* (SF), *Line Of Code* (LOC) dan atribut *Actual Effort* dari proyek yang

diselesaikan. Kemudian data tersebut akan dibandingkan dengan hasil perhitungan menggunakan algoritma *Grey Wolf Optimizer* untuk mendapatkan nilai *Person Month* (PM).

### 3.2.2 Penyiapan Alat

Salah satu faktor yang mendukung dalam berjalannya penelitian ini adalah penyiapan alat. Ada dua komponen utama yang harus tersedia pada penyiapan alat yaitu perangkat keras dan perangkat lunak. Kebutuhan perangkat keras dan perangkat lunak pada penelitian ini adalah sebagai berikut:

a. Perangkat Keras

Spesifikasi perangkat keras yang digunakan adalah sebagai berikut:

1. Laptop HP Pavilion Gaming 15 dengan spesifikasi sebagai berikut:
   - Prosesor AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
   - RAM 8 GB DDR4
   - ROM 512 GB SSD
   - Monitor 15.6” Full HD Slim Bezel
   - NVIDIA GeForce GTX 1650
2. Mouse

b. Perangkat Lunak

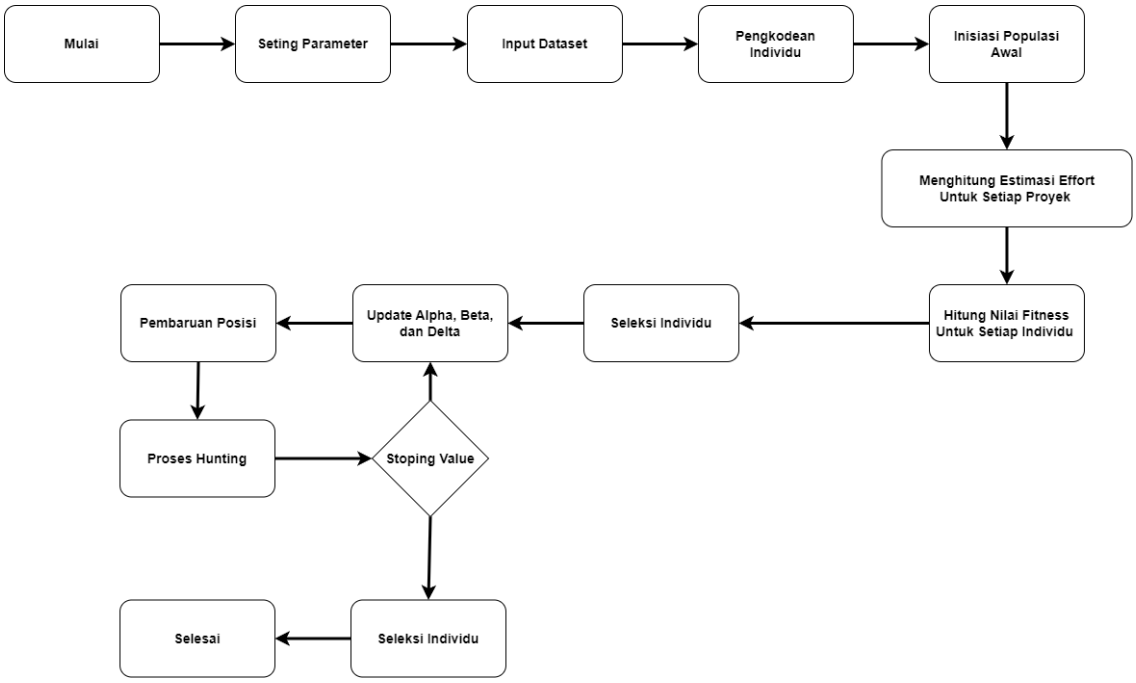
Spesifikasi perangkat lunak yang digunakan adalah sebagai berikut:

1. Windows 10
2. Browser Google Crome
3. Microsoft Office 2019
4. Microsoft Excel
5. Visual Studio Code
6. Mendeley Reference Manager
7. Draw.io

8. SPSS
9. Python

### 3.2.3 Optimasi

Data yang akan digunakan pada penelitian ini adalah data proyek yang didapatkan oleh Silhavy, Radek (2017) yang akan meliputi perhitungan *Effort Multipiler* (EM), *Scale Factor* (SF), *Line Of Code* (LOC) dan atribut *Actual Effort* dari proyek yang telah diselesaikan. Lalu setelah data yang didapat akan dimasukkan pada proses perhitungan menggunakan *Grey Wolf Optimizer.* Nilai yang akan dihitung menggunakan *Grey Wolf Optimizer* adalah nilai *Use Case Points*, sebelum mendapatkan nilai *Use Case Points* mencari terlebih dahulu nilai *Unadjusted Use Case Point* (UUCP), nilai *Technical Complexity Factor* (TCF) dan nilai *Environmental Complexity Factor* (ECF) setelah itu dijumlahkan untuk mendapatkan nilai *Use Case Points.* Dibawah ini adalah tahapannya.




*Gambar 3. 3 Flowchart Tahapan Optimasi*

### 3.2.4 Evaluasi

Evaluasi merupakan proses penilaian terhadapat suatu metode estimasi yang digunakan untuk menghasilkan performa yang diharapkan atau tidak. Terdapat berbagai

macam teknik evaluasi yang bisa digunakan, dan pada tahap evaluasi yang akan digunakan adalah *Mean Absolute Eror* (MAE), *Mean Balanced Eror* (MBRE) dan *Mean Inverted Balanced Residual Eror* (MIBRE). MAE, MBRE dan MIBRE akan digunakan karena bisa memberikan akurasi yang berbeda-beda satu sama lain. Dan ketiga teknik evaluasi ini bisa menilai seberapa bagus suatu model untuk memprediksi secara efektif. Sehingga mendapatkan model akurasi terbaik yang memliki nilai minimum.

### 3.2.5 Validasi

Validasi adalah proses memverifikasi atau mengonfirmasi kebenaran, kualitas, atau kemampuan prediktif model statistik atau algoritma pada data yang tidak terlihat saat pelatihan. Teknik validasi model yang dipakai dalam penelitian ini adalah *Leave One Out Cross Validation* (LOOCV). LOOCV adalah teknik validasi model yang menggunakan satu data pada setiap iterasi sebagai data uji dan sisanya sebagai data latih. Jika terdapat n data dalam dataset, maka akan dilakukan n iterasi. Pada setiap iterasi, satu data digunakan sebagai data uji dan sisanya sebagai data latih. LOOCV digunakan untuk menghitung akurasi model dengan menghitung rata-rata dari semua iterasi. Keuntungan teknik LOOCV adalah dapat mengevaluasi model dengan semua observasi sebagai data uji, sehingga dapat memberikan estimasi yang lebih akurat.

### 3.2.6 Uji Statistik Non Parametrik Wilcoxon

Pada Uji Statistik Non Parametrik Wilcoxon ini adalah untuk menguji hasil optimasi yang diperoleh, dengan melakukan perbandingan dengan cara menggunakan aplikasi SPSS. Pengujian ini dilakukan untuk mengetahui ada atau tidaknya perbedaan rata-rata antara dua kelompok data yang saling berhubungan. Dimana data yang digunakan untuk pengujian yaitu nilai *Absolute Eror* (AE) dari estimasi *Use Case Points* standar dan nilai *Absolute Eror* (AE) dari estimasi *Use Case Points* dengan algoritma *Grey Wolf Optimizer*.

# BAB IV

# HASIL DAN PEMBAHASAN

## 4.1 Hasil Pengumpulan Data

Eksperimen pada metode UCW+PSO menggunakan data historis proyek dari tiga perusahaan *perangkat lunak*. Himpunan data terdiri dari 71 proyek yang dikumpulkan oleh [80][132], berasal dari berbagai domain seperti asuransi, pemerintahan, perbankan, dan lain sebagainya (lihat Lampiran 1). Himpunan data ini dipilih karena berasal dari dunia industri yaitu tiga perusahaan perangkat lunak sehingga hasilnya bisa digeneralisasi [84]. Himpunan data berisi 38 atribut: *Simple Actors, Average Actors, Complex Actors, Simple UC, Complex UC, T1-T13, Env1-Env8, Sector, Language, Methodology, Application Type, UAW, UUCW, TCF, ECF, Real_P20, Real_Effort_Person_Hours,* dan *Data Donator*. Untuk keperluan eksperimen, dari ke-38 atribut yang digunakan hanya sebanyak 7 atribut saja, yaitu *Simple UC, Average UC, Complex UC, UAW, TCF, ECF*, dan *Actual Effort*. Sebagian besar proyek ditulis menggunakan Bahasa Java dan C#. Adapun ringkasan statistik deskriptif himpunan data histori proyek ini ditunjukkan oleh Tabel 4.1.

Tabel 4.1. Statistik deskriptif himpunan data Silhavy ($N$ = 71)

| Variable | Mean | StDev | Skewness | Kurtosis | Max | Min |
|---|---|---|---|---|---|---|
| SimpleUC | 2,7 | 2,9 | 3,29 | 17,658 | 20 | 0,00 |
| AverageUC | 15,84 | 5,37 | 0,296 | 0,140 | 30 | 3,00 |
| ComplexUC | 14,29 | 4,45 | 0,191 | -0,290 | 27 | 5,00 |
| UAW | 10,49 | 5,01 | 0,803 | -1,264 | 19 | 6,00 |
| TCF | 0,92 | 0,114 | -0,269 | -1,019 | 1,12 | 0,71 |
| ECF | 0,86 | 0,117 | -0,556 | 0,861 | 1,09 | 0,51 |
| Actual Effort | 6558,72 | 664,24 | 0,574 | -0,922 | 7970 | 5775 |

Berdasarkan Tabel 4. tersebut dapat diketahui bahwa variabel AverageUC, ComplexUC, dan TCF berdistribusi normal dengan *skewess* yang mendekati nol. Sedangkan SimpleUC tidak berdistribusi normal dengan *skewness* yang jauh dari nol. Menariknya, variabel ComplexUC cenderung melebar dengan nilai kurtosis -0,29, dan SimpleUC membentuk kurva leptokurtosis dengan nilai kurtosis sebesar 17,658. Apabila nilai kurtosis lebih besar dari tiga, maka variabel cenderung kebal terhadap pencilan. Dari tabel tersebut bisa ditemukan bahwa hanya SimpleUC saja yang kurtosisnya lebih besar dari tiga. Sehingga variabel yang lain cenderung memiliki pencilan. Selain itu, dapat diketahui pula bahwa sebagian besar *use case* proyek

adalah *average*. Hal ini menunjukkan bahwa nilai rerata dan maksimum AverageUC lebih luas dibanding SimpleUC dan ComplexUC.

### 4.2 Hasil dan Pembahasan Implementasi Metode Use Case Points

Perhitungan estimasi UCP diimplementasikan dalam sebuah *class* bernama `UseCasePoints` yang ditunjukkan oleh gambar berikut. *Class* `UseCasePoints` terdiri dari *method* `estimatingUCP` dengan parameter `singleTupleTestData`, `productivityFactor`, dan `useCaseWeight`.

Pada *method* `estimating`, langkah pertama adalah mendeklarasikan variabel untuk menampung seluruh indeks dari `singleTupleData`. Langkah kedua adalah menghitung `uucw` dengan menggunakan persamaan (2), dilanjutkan `uucp` yang menggunakan persamaan (3), `ucp` berdasarkan persamaan (6), `estimatedEffort`, dan `absoluteError`. Kembalian dari `estimating` berupa *array* yang terdiri dari dua elemen yaitu `estimatedEffort` dan `absoluteError`. Kedua elemen tersebut disertakan karena `estimatedEffort` untuk digunakan ketika membuat model grafik perbandingan antara `actualEffort` dan `estimatedEffort`. Sedangkan `absoluteError` digunakan untuk menghitung nilai *fitness*.

```
class UseCasePoints:

    def estimatingUCP(singleTupleTestData, productivityFactor,
useCaseWeights):
        simpleTestData = singleTupleTestData[0]
        averageTestData = singleTupleTestData[1]
        complexTestData = singleTupleTestData[2]
        uawTestData = singleTupleTestData[3]
        tcfTestData = singleTupleTestData[4]
        ecfTestData = singleTupleTestData[5]
        actualEffortTestData = singleTupleTestData[6]
        simpleWeight = useCaseWeights[0]
        averageWeight = useCaseWeights[1]
        complexWeight = useCaseWeights[2]

        uucw = (simpleTestData * simpleWeight) + (averageTestData *
averageWeight) + (complexTestData * complexWeight)

        uucp = uawTestData + uucw

        ucp = uucp * tcfTestData * ecfTestData

        estimatedEffort = ucp * productivityFactor
```

```
        absoluteError = abs(actualEffortTestData - estimatedEffort)

        return [estimatedEffort, absoluteError]
        # output: [6752.52, 977.52]
```

Gambar 4.2 Implementasi Use Case Points

### 4.3 Hasil dan Pembahasan Implementasi UCP+GWO

Implementasi UCP+GWO tersusun atas *class* `GreyWolfOptimizer` dengan empat method yaitu `isPositioningOutOfRange`, `fitnessFunction`, `preyHunting`, dan `runGWO`. `isPositioningOutOfRange` berfungsi untuk memeriksa apakah posisi baru yang dihasilkan melebihi batasan rentang yang ditentukan atau tidak, `fitnessFunction` untuk menghitung nilai *fitness*, `preyHunting` untuk memperbarui posisi tiap *wolf* dan `runGWO` berfungsi pembaruan solusi dan mendapatkan solusi terbaik. Secara rinci keempat *method* tersebut dibahas satu per satu sebagai berikut.

*Method* `isPositionOutOfRange` memiliki dua parameter yaitu `index` dan `newPosition`. *Method* ini akan memeriksa apakah posisi baru yang dihasilkan lebih kecil dari batas bawah rentang variabel desain atau lebih besar dari batas atas variabel desain. Kembalian *method* ini berupa nilai boolean `True`. Method `fitnessFunction` hanya menerima parameter input `objectiveValue` yang selanjutnya mengembalikan nilai di bawah satu. Method ini digunakan ketika tiap solusi (*wolf*) akan dievaluasi nilai fitnessnya.

*Method* `preyHunting` terdiri dari input parameter `population`, `wolfType`, `C`, dan `A`. Tiap `wolf` dalam `population` akan diperbarui posisinya dengan menghitung nilai `D` sesuai persamaan (11), (12).

```
class GreyWolfOptimizer:

    def __init__(self, designVariableRanges, singleTupleDataset,
productivityFactor):
        self.designVariableRanges = designVariableRanges
        self.singleTupleDataset = singleTupleDataset
        self.productivityFactor = productivityFactor
        self.estimatedEffortIndex = 0
        self.absoluteErrorIndex = 1
```

```
    def isPositionOutOfRange(self, index, newPosition):
        for i in range(len(self.designVariableRanges)):
            if i == index and (newPosition <
self.designVariableRanges[i]['lowerBound'] or newPosition >
self.designVariableRanges[i]['upperBound']):
                return True

    def fitnessFunction(self, objectiveValue):
        smallNumber = 0.0001
        return 1 / (objectiveValue + smallNumber)

    def preyHunting(self, population, wolfType, C, A):
        for wolf in population:
            for i in range(len(wolfType['wolfPositions'])):
                D = abs(C * wolfType['wolfPositions'][i] -
wolf['wolfPositions'][i])
                newPosition = wolfType['wolfPositions'][i] - A * D

                if self.isPositionOutOfRange(i, newPosition):
                    newPosition = wolfType['wolfPositions'][i]

                wolf['wolfPositions'][i] = newPosition
                objectiveValues =
UseCasePoints.estimatingUCP(self.singleTupleDataset,
self.productivityFactor, wolf['wolfPositions'])
                wolf['estimatedEffort'] =
objectiveValues[self.estimatedEffortIndex]
                wolf['absoluteError'] =
objectiveValues[self.absoluteErrorIndex]
                wolf['fitnessValue'] =
self.fitnessFunction(objectiveValues[self.absoluteErrorIndex])
        return populatio
```

Gambar 4.3 Implementasi UCP+GWO (a)

Method `runGWO` terdiri dari parameter `populationSize` dan `maxIter`. Pertama-tama dilakukan pembangkitan populasi awal sebanyak `populationSize`. Tiap posisi kandidat solusi ditentukan dengan membangkitkan nilai acarak di antara rentang nilai variabel desain. Tiap kandidat solusi dikemas dalam *array* asosiatif terdiri dari `wolfPositions`, `estimatedEffort`, `absoluteError`, dan `fitnessValue`. Pada tahap kedua, dilakukan kalkulasi nilai objektif dan *fitness* tiap kandidat solusi yang diakhiri dengan pengurutan populasi secara menurun (*descending*). Berdasarkan populasi yang telah terurut selanjutnya ditentukan *wolf* `alpha`, `beta` dan `delta`.

Langkah ketiga adalah menghitung nilai `a`, `A`, dan `C` yang akan digunakan pada tahapan keempat yaitu *encircling prey*. Pada tahap ini tiap ini, tiap posisi dari *wolf* `alpha`, `beta` dan `delta` dimasukkan ke persamaan (13) untuk mendapatkan posisi baru yang berarti memperbarui kandidat solusi sebelumnya. Proses dari tahap satu hingga empat ini terus berulang hingga iterasi berakhir yang pada selanjutnya diperoleh solusi terbaik.

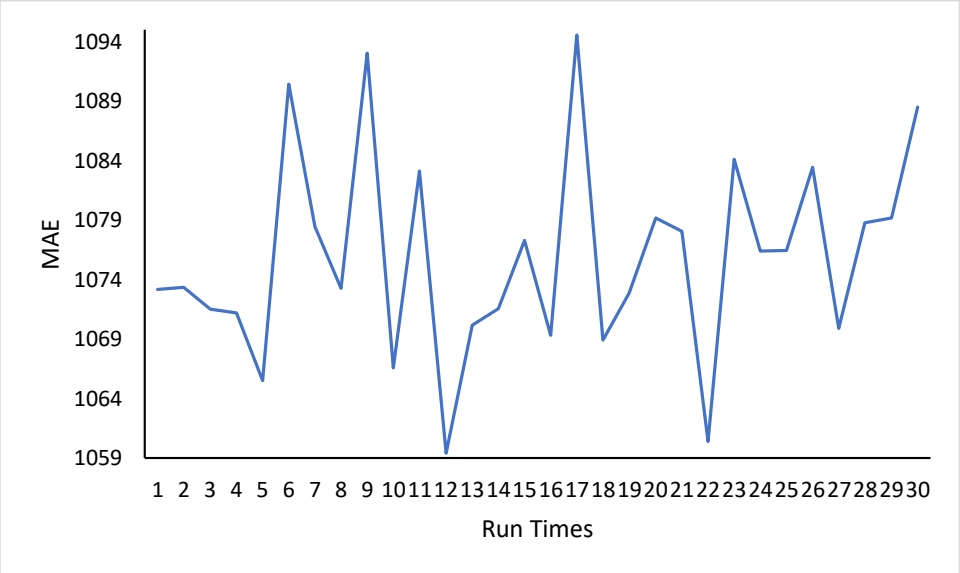
```
    def runGWO(self, populationSize, maxIter):

        # 1. Generate initial population
        wolfPositions = []; population = []
        bestTempFitnessValue = 0
        bestWolf = {'wolfPositions':wolfPositions,
'estimatedEffort':None, 'absoluteError':None, 'fitnessValue':0}
        alphaIndex = 0; betaIndex = 1; deltaIndex = 2

        for _ in range(populationSize):
            for designVariableRange in designVariableRanges:
                designVariableValue =
random.uniform(designVariableRange['lowerBound'],
designVariableRange['upperBound'])
                wolfPositions.append(designVariableValue)
                # print(designVariableValues)
                # output: [6.34, 8.99, 14.65]
            population.append({'wolfPositions':wolfPositions,
'estimatedEffort':None, 'absoluteError':None, 'fitnessValue':0})
            wolfPositions = []
        # print(population)
        # sys.exit()
        # output: [{'wolfPositions': [6.43, 9.99, 13.75],
'estimatedEffort': None, 'absoluteError': None, 'fitnessValue': 0},
{'wolfPositions': [6.92, 9.18, 14.40], 'estimatedEffort': None,
'absoluteError': None, 'fitnessValue': 0},.., {'wolfPositions': [6.58,
10.55, 13.29], 'estimatedEffort': None, 'absoluteError': None,
'fitnessValue': 0}]

        for iter in range(maxIter):

            # 2. Calculate objective and fitness values
            # TODO provide spefici function
            for wolf in population:
                objectiveValues =
UseCasePoints.estimatingUCP(self.singleTupleDataset,
self.productivityFactor, wolf['wolfPositions'])
                wolf['estimatedEffort'] =
objectiveValues[self.estimatedEffortIndex]
```

```
                wolf['absoluteError'] =
objectiveValues[self.absoluteErrorIndex]
                wolf['fitnessValue'] =
self.fitnessFunction(objectiveValues[self.absoluteErrorIndex])
            # print(population)
            # sys.exit()
            # output: [{'wolfPositions': [6.61, 12.42, 14.21],
'estimatedEffort': 5254.46, 'absoluteError': 2715.53, 'fitnessValue':
0.000190}, {'wolfPositions': [6.84, 11.23, 13.51], 'estimatedEffort':
4968.96, 'absoluteError': 3001.03, 'fitnessValue':
0.000201},..,{'wolfPositions': [6.83, 11.96, 13.54], 'estimatedEffort':
5072.88, 'absoluteError': 2897.11, 'fitnessValue': 0.000197}]

            # 3. Grouping hierarchy [alpha, beta, delta, omega]
            population = sorted(population, key=lambda x:
x['fitnessValue'], reverse=True)
            # print(population)
            # sys.exit()
            # output: [{'wolfPositions': [5.55, 7.60, 14.13],
'estimatedEffort': 4496.76, 'absoluteError': 3473.23, 'fitnessValue':
0.000222}, {'wolfPositions': [7.17, 8.02, 13.49], 'estimatedEffort':
4553.95, 'absoluteError': 3416.04, 'fitnessValue':
0.000219},..,{'wolfPositions': [5.77, 11.95, 14.24], 'estimatedEffort':
5128.55, 'absoluteError': 2841.44, 'fitnessValue': 0.000194}]

            # 4. Initializing a, A, and C
            coefficient = 2
            a = coefficient - (coefficient * iter) / maxIter
            A = (coefficient * a) * random.uniform(0,1) - a
            C = coefficient * random.uniform(0,1)

            # 5. Encircling Prey
            alphaWolf = population[alphaIndex]
            alphaPopulation = self.preyHunting(population, alphaWolf , C,
A)
            alphaWolfFitnessValue = alphaWolf['fitnessValue']

            if alphaWolfFitnessValue > bestTempFitnessValue:
                bestTempFitnessValue = alphaWolfFitnessValue
                bestWolf['wolfPositions'] = alphaWolf['wolfPositions']
                bestWolf['estimatedEffort'] =
alphaWolf['estimatedEffort']
                bestWolf['absoluteError'] = alphaWolf['absoluteError']
                bestWolf['fitnessValue'] = alphaWolf['fitnessValue']

            betaPopulation = self.preyHunting(population,
population[betaIndex], C, A)
```

```
            deltaPopulation = self.preyHunting(population,
population[deltaIndex], C, A)

            varLength = len(self.designVariableRanges)
            positions = []
            for i in range(populationSize):
                for j in range(varLength):
                    position = (alphaPopulation[i]['wolfPositions'][j] +
betaPopulation[i]['wolfPositions'][j] +
deltaPopulation[i]['wolfPositions'][j]) / varLength
                    positions.append(position)
                population[i]['wolfPositions'] = positions
                positions = []
        return bestWolf
```

Gambar 4.4 Implementasi algoritma UCP+GWO (b)

Gambar 4.5 menunjukkan hasil sementara estimasi *effort* UCP setelah pembobotan kompleksitas *use case*-nya dilakukan menggunakan GWO. Terlihat bahwa pada saat running ke-12 dan 22 UCP+GWO memperoleh MAE terkecil. Sebaliknya running ke-9 dan 17 memperoleh MAE terbesar.

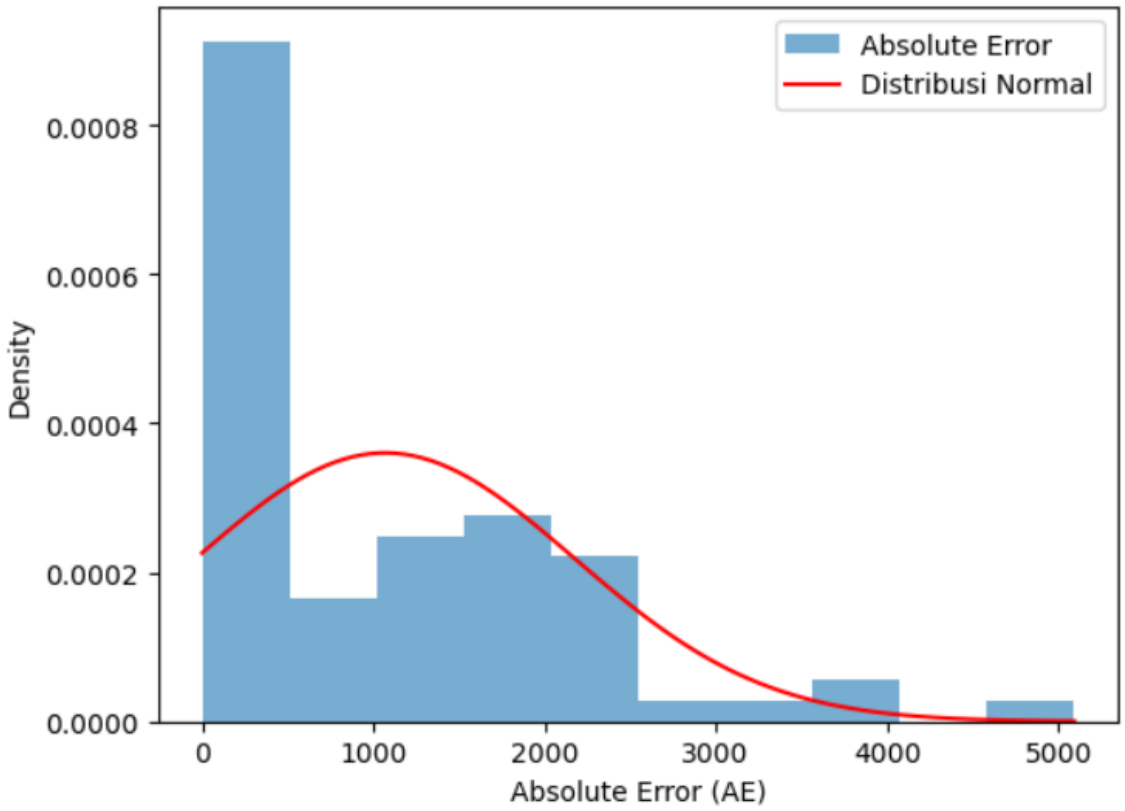



Gambar 4.5 Hasil implementasi UCP+GWO

### 4.4 Pengujian Statistik

Sebelum memutuskan metode statistik yang akan digunakan untuk pengambilan keputusan terhadap performa model, maka pdilakukan analisis data hasil *running* UCP+GWO sebelumnya. Gambar 4.6 menunjukkan bahwa data yang dihasilkan oleh UCP+GWO

menunjukkan tidak berdistribusi normal dengan *p*-value sebesar 0.0255. Karena datanya tidak berdistribusi normal, maka metode uji hipotesis yang digunakan adalah Wilcoxon. Hasil pengujian Wilcoxon menunjukkan nilai *p*-value sebesar 4.047233048463143e-13 yang berarti antara perbedaan signifikan antara sampel absolute error yang dihasilkan dari UCP dengan sampel absolute error yang dihasilkan oleh UCP+GWO.




Gambar 4.6 Histogram data beserta kurva distribusi normalnya

# BAB V

# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Penelitian ini melakukan optimasi pada pembobotan kompleksitas *use case* pada metode estimasi *effort* perangkat lunak Use Case Points (UCP) menggunakan algoritma Grey Wolf Optimizer (GWO) yang diberi nama UCP+GWO. Kompleksitas *use case* yang dioptimasi adalah *simple*, *average*, dan *complex* yang masing-masing memiliki rentang nilai yaitu [5.00, 7.49], [7.50, 12.49], dan [12.5, 15.00]. Berdasarkan eksperimen diperoleh hasil bahwa MAE terbaik diperoleh oleh UCP+GWO yaitu sebesar 1070.65, sedangkan yang diperoleh UCP standar yaitu 1806.55. Berdasarkan uji statistik Wilcoxon didapatkan bahwa terdapat perbedaan signifikan antara kedua model UCP dan UCP+GWO. Sehinga bisa disimpulkan bahwa UCP+GWO berhasil meningkatkan performa akurasi metode estimasi *effort* Use Case Points.

## 1.5 Saran

Beberapa aspek penting yang digunakan pada model estimasi yang dikembangkan pada penelitian ini yaitu setting parameter GWO sendiri. Untuk penelitian berikutnya perlu dilakukan misalnya dengan menggunakan teknik *chaotic* untuk membangkitkan nilai-nilai acak pada GWO. Selain itu perlu juga dilakukan modifikasi pada saat ekpsloitasi dengan algoritma menggunakan pencarian lokal seperti *tabu search* atau *simulated annealing*.

# DAFTAR PUSTAKA


[1] "Muhadi2015 - Fuzzy Use Case Point - 2015.pdf."

[2] P. A. Ningrum, J. S. Informasi, and F. T. Informasi, "Penentuan Nilai Effort Rate ( Er ) Pada Metode Use Case Point ( Ucp ) Untuk Estimasi Effort Proyek Pengembangan Perangkat Lunak Di Bidang Bisnis," *J. Tek. Pomits*, pp. 1–6, 1993.

[3] M. Nur Faiz, W. Adi Prabowo, and M. Fajar Sidiq, "Studi Komparasi Investigasi Digital Forensik pada Tindak Kriminal," *J. Informatics, Inf. Syst. Softw. Eng. Appl.*, vol. 1, no. 1, pp. 63–70, 2018, doi: 10.20895/INISTA.V1I1.

[4] S. A. Y. Anggara, Y. D. Chandra, M. S. D. Putra, and S. R. Wicaksono, "Analisis Pengukuran Kualitas Perangkat Lunak Point of Sale Nextar Menggunakan Metode Functional Point Analysis," *J. Ilmu Komput. dan Teknol.*, vol. 1, no. 1, pp. 12–14, 2020, doi: 10.35960/ikomti.v1i1.502.

[5] T. A. Pertiwi, N. T. Luchia, P. Sinta, R. Aprinastya, I. R. Fachrezi, and M. L. Hamzah, "Jurnal Testing dan Implementasi Sistem Informasi ABSENSI BERBASIS WEB MENGGUNAKAN METODE AGILE WEB-BASED ATTENTION INFORMATION SYSTEM DESIGN AND IMPLEMENTATION USING THE AGILE SOFTWARE DEVELOPMENT," vol. 1, no. 11, pp. 53–66, 2023.

[6] M. Azzeh, A. Bou Nassif, and I. B. Attili, "Predicting software effort from use case points: A systematic review," *Science of Computer Programming*, vol. 204. 2021. doi: 10.1016/j.scico.2020.102596.

[7] A. P. Pamungkas, "Estimasi Effort Perangkat Lunak COCOMO II Dengan Pemilihan Konstanta Multiplikatif Dan Eksponensial Menggunakan Algoritma Genetika," 2015.

[8] V. Lestari, A. Kamsyakawuni, and K. Agung Santoso Jurusan Matematika, "IMPLEMENTASI ALGORITMA GREY WOLF OPTIMIZER (GWO) DI TOKO CITRA TANI JEMBER (Implementation of the Grey Wolf Optimizer (GWO) Algorithm at Citra Tani Jember Store)," *Maj. Ilm. Mat. dan Stat.*, vol. 19, pp. 65–74, 2019, [Online].

Available: https://jurnal.unej.ac.id/index.php/MIMS/index

[9] H. R. M. A. Putra, W. H. N. Putra, and D. Pramono, "Estimasi Biaya Perangkat Lunak menggunakan Metode Use Case Point (Studi Kasus: PT. Pln (Persero) Area Malang)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, no. 6, p. 964X, 2019, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5520

[10] P. Jayadi, A. C. Aria Bima, Y. P. Yudha, and Kelik Sussolaikah, "End User Development pada Use Case Point untuk peningkatan Estimasi Perangkat Lunak," *Tematik*, vol. 10, no. 1, pp. 74–82, 2023, doi: 10.38204/tematik.v10i1.1289.

[11] R. P. Manik, A. R. Perdanakusuma, and M. C. Saputra, "Evaluasi Biaya Perangkat Lunak Menggunakan Metode Fuzzy Use Case Point," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 7, pp. 2649–2659, 2018.

[12] M. D. Newton, E. P. Boer, W. N. Lipscomb *et al.*, "Perbandingan Evaluasi Biaya Pengembangan Sistem Antrian RSUD Dr Soetrasno Rembang Menggunakan Metode Use Case Point dan Function Point ( Studi Kasus : CV Pabrik Teknologi )," *2010 Int. Conf. Logist. Syst. Intell. Manag. ICLSIM 2010*, vol. 2, no. 2, pp. 2367-, 2018.

[13] M. O. Okwu and L. K. Tartibu, "Grey Wolf Optimizer," *Stud. Comput. Intell.*, vol. 927, pp. 43–52, 2021, doi: 10.1007/978-3-030-61111-8_5.

**LAMPIRAN 1. Bobot Kompleksitas Use Case dan absolute error terbaik**

[7.396540252121788, 12.48360998948572, 14.99524598261434] 2484.0160241543035

[6.940387861317883, 8.689410593616861, 13.066896961971885] 12.345825719640743

[6.724638051000286, 11.75233199605632, 14.773984572419478] 1837.9604933750143

[7.005868787011458, 12.346831421374096, 14.99451459454596] 1393.6161236427415

[7.273667386039615, 12.488294260059149, 14.839368533224308] 1573.133023231555

[7.311884145751715, 12.329843302009897, 14.939044458840963] 101.19970494619338

[5.3148691939000825, 9.581130035635564, 13.094527125461164] 18.40493050540954

[7.404844436495, 12.277789869417056, 14.855776017508283] 1098.5211181208806

[7.22007479255982, 12.33506012553548, 14.77924877249064] 402.2134449638561

[5.051589934844927, 7.74929266974805, 13.710948610486664] 18.157897603015954

[7.362463425961901, 12.452067608173616, 14.74401213279271] 1787.3998142544406

[7.326501536885094, 12.295666039392747, 14.706923066853799] 1935.0395699643368

[7.4857421782245686, 12.459512854290136, 14.990631032339065] 1808.508446065427

[5.047058730713773, 8.079106970490649, 12.586517175613704] 412.8790425074876

[6.943270293424552, 12.191754796756086, 13.199669295357806] 28.90167169616143

[7.464060773425818, 12.461019963482746, 14.885969855324968] 2174.040301834574

[7.059635872018013, 12.159590368581863, 14.696607916302252] 1836.294924706759

[7.120146678583164, 12.048676077234589, 14.904730589299156] 874.8705050179296

[7.452929118036519, 12.436779574774286, 14.979956850316938] 2147.7008599681694

[6.77262082729312, 12.240812562849072, 14.538113339227813] 4.92450789371469

[7.2985998530519005, 12.334759158686003, 14.97816190097055] 2470.1275874892617

[5.077737370865054, 7.645139857274948, 12.555208620464322] 135.0828817749989

[7.417906168120805, 12.426355444082603, 14.995072908839282] 940.1434874769557

[6.903190765725708, 12.367240067260523, 14.941230940155243] 1731.1096097112058

[6.915079119224797, 12.227083920383393, 14.470996391674602] 1524.509353036081

[6.510870606768966, 9.358402539631173, 13.634788211449623] 7.574853858021925

[7.4790447266313995, 12.429891296563992, 14.979741459936394] 4010.4146993591194

[7.454747640528176, 12.46555334943016, 14.791552774459356] 1213.4614070459202

[6.00433188006706, 9.67693443637901, 14.748700962123502] 8.00517375813979

[7.4480958131696395, 12.39750468628825, 14.9844372213755] 1538.3309598527667

[7.120818702113029, 10.628529603036187, 13.089144229625262] 5.983745703051682

[6.476265964501124, 12.440076939440381, 14.9013791803995] 776.5976179508598

[7.288549504825392, 12.424986214730694, 14.709202121693277] 32.81261480279045

[5.015814719979463, 12.02879405623675, 13.681515197901623] 7.403553770955114

[7.44238318407553, 12.4890103117932, 14.945692088315374] 2395.215656602454

[7.482340087684814, 12.485248864988547, 14.782501414244427] 72.05967751420394

[7.371104599121581, 12.469154663106401, 14.625496109039997] 1118.0705785278087

[7.489331111541126, 12.37937100893617, 14.989441931724093] 3061.9913341404185

[7.4298780567212255, 12.285501808669503, 14.77495281388847] 7.560661510481623

[6.13211946856102, 10.670258213524471, 12.67118880531196] 5.440659351026625

[7.255106875447896, 12.393856788642031, 14.874450026712745] 3732.1732304463476

[7.406650659187476, 12.319560364990751, 14.877839258211452] 886.5283109480006

[5.3234633056202165, 8.727289361807385, 14.940722578715958] 1.024922812675868

[5.290649656668057, 7.532994296304306, 12.56819736127404] 25.716675687866882

[7.094988173590022, 12.447452265236961, 14.533802067748255] 2249.1954459793124

[6.702484616346914, 8.060597035678825, 12.677998788831143] 1093.2215040721067

[7.251519562965119, 12.10015715561906, 14.549444103237951] 523.5522449762111

[5.04293411429899, 7.5812098778280355, 12.639560519735204] 396.3046157292374

[7.459798682667951, 12.410991166344806, 14.97069728804443] 1301.0545680043342

[5.064805818399089, 7.686830185315311, 12.627623466259578] 336.7276136701239

[5.24724024177333, 7.501965582520182, 12.507199534307896] 4745.2154727704365

[5.03129141469462, 7.734146437961073, 12.657838056358647] 413.18417343132023

[5.554893281254138, 8.258222906548063, 14.869926447859568] 39.99896708677625

[6.957062296180954, 7.521364940819471, 13.311820285369105] 14.828807423602484

[5.339398645277582, 8.056184673585339, 13.052857903754983] 591.3651902765987

[5.02398209562147, 7.503253746112057, 12.50210048838164] 1257.1732395769232

[5.171080245269298, 12.37318515105169, 14.474463539618316] 318.0740481930861

[5.0147454093606925, 7.597495532460967, 12.519674789391724] 307.7369008283031

[5.21362620321301, 7.85337309412631, 13.099383996607846] 1095.7339238684835

[6.482630633800504, 12.474945702087501, 14.978431370912835] 2341.7535180612144

[5.5944654158802, 8.199875439303646, 12.517075870920024] 37.698984995984574

[6.892345011405102, 9.063750403015918, 14.356607343307678] 1.5769758118185564

[7.04713317322329, 11.53378576045365, 14.601226818071662] 742.5943352143677

[5.0211499562636375, 7.563783810186873, 12.548193767364928] 2719.3853780266854

[5.032366034737996, 7.509740789194969, 12.536520433450042] 59.74258773654037

[5.356217813202931, 7.749609696715763, 12.541632909594322] 1939.2663327395276

[5.104532997989665, 7.652100970042204, 12.63805672171638] 2048.966493730295

[5.0848271533328, 7.606920165761615, 12.62860013527172] 2034.9102361613213

[5.156649753706951, 7.7993712300225795, 12.605188818921777] 1294.1884670453546

[7.168889109755971, 12.464329773237992, 14.932541795284601] 194.11333932258185

[5.253297060047458, 7.5001913676682035, 12.875378821032678] 261.60932907935785

MAE: 1070.6569039030276