BAB II
TINJAUAN PUSTAKA

Pada bab ini akan menjelaskan mengenai penelitian terdahulu dan dasar teori yang menjadi acuan atau landasan dalam pengerjaan tugas akhir ini.

2.1 Penelitian Terdahulu

Pada penelitian yang dilakukan oleh Habib Rahman dan kawan-kawan dengan judul "Estimasi Biaya Perangkat Lunak menggunakan Metode Use Case Point (Studi Kasus: PT. Pln(Persero) Area Malang)" hasil dari penelitian ini ini menunjukkan bahwa proyek Pengembangan Sistem Informasi Konstruksi Tiang PT. PLN (Persero) Area Malang menghasilkan estimasi waktu sebesar 411,04 jam atau setara dengan 2,5 bulan. Namun, dalam alokasi aktual yang dilakukan oleh perusahaan, estimasi waktu menjadi 640 jam atau setara dengan 4 bulan. Ini mengakibatkan selisih estimasi waktu sebesar 228,96 jam atau 1,5 bulan. Selain itu, estimasi biaya proyek juga diteliti, dan hasilnya menunjukkan bahwa estimasi biaya total sebesar 34% antara estimasi biaya proyek dan alokasi aktual Perusahaan [11]

Penelitian oleh Krisdian dan kawan-kawan dengan judul "Estimasi Biaya Pengembangan Perangkat Lunak Menggunakan Metode Fuzzy Use Case Point (Studi Kasus: Dinas Komunikasi dan Informatika Kota Batu)" hasil dari penelitan ini menyatakan bahwa estimasi waktu dan biaya menggunakan metode Fuzzy Use Case Point menghasilkan nilai yang lebih kecil dibandingkan dengan kondisi aktual, dengan rata-rata percent error sebesar 36,62% untuk durasi pengerjaan dan 69,99% untuk biaya total. Hal ini disebabkan oleh pendekatan yang lebih terstruktur dalam metode Fuzzy Use Case Point, yang tidak diterapkan oleh DISKOMINFO Kota Batu. Penelitian ini menyarankan agar DISKOMINFO menyempurnakan SOP untuk proses pengembangan aplikasi, termasuk pemisahan antara pengembangan website dan aplikasi berbasis web, serta menggunakan acuan dari PMBOK untuk penjadwalan proyek. Untuk penelitian selanjutnya, disarankan agar parameter seperti nilai staff-hour

dan distribusi usaha diperbarui, dan lebih banyak studi kasus serta jenis sistem diperluas untuk meningkatkan akurasi estimasi [12].

Penelitian oleh Manik dan kawan-kawan dengan judul "Evaluasi Biaya Perangkat Lunak Menggunakan Metode Fuzzy Use Case Point" mendapatkan hasil penelitian, estimasi waktu menggunakan metode Fuzzy Use Case Point untuk SIPAS adalah 91,7 jam dan untuk SMTPX adalah 129,84 jam. Estimasi biaya untuk SIPAS sekitar 61% dari total biaya aktual, dan untuk SMTPX sekitar 70%. Penjadwalan membutuhkan alokasi 26 staf, di mana biaya estimasi mewakili sekitar 61% dari total biaya aktual untuk SIPAS dan sekitar 70% untuk SMTPX [13]

2.2 Perangkat Lunak

Perangkat lunak adalah sebuah program yang berfungsi untuk memanipulasi data yang ada di dalamnya, dilengkapi dengan instruksi-instruksi yang menghadirkan fitur atau fungsi spesifik sehingga data yang dimanipulasi dapat ditampilkan dalam bentuk virtual [14].

Perangkat lunak, atau sering disebut juga sebagai software, merupakan bagian tak terpisahkan dari sistem komputer yang berfungsi untuk mengatur dan mengelola data serta proses-proses yang berjalan di dalamnya. Secara lebih rinci, perangkat lunak terdiri dari serangkaian instruksi atau kode yang ditulis dalam bahasa pemrograman tertentu. Instruksi-instruksi ini dirancang untuk menjalankan berbagai operasi dan fungsi yang memungkinkan pengguna untuk melakukan tugas-tugas spesifik sesuai dengan kebutuhan mereka.

Perangkat lunak aplikasi adalah sekumpulan program yang dirancang untuk membantu bisnis dengan memproses data yang dibutuhkan untuk menjalankan operasinya secara detail.[14] .

Perangkat lunak aplikasi, sering kali disebut sebagai aplikasi bisnis atau software aplikasi, terdiri dari berbagai program yang dirancang khusus untuk mendukung berbagai aktivitas bisnis. Program-program ini berfungsi untuk

mengotomatisasi, mengelola, dan mengoptimalkan berbagai proses bisnis sehingga operasi sehari-hari dapat berjalan lebih efisien dan efektif.

2.3 Estimasi Biaya Perangkat Lunak

Estimasi biaya perangkat lunak adalah proses dalam memproyeksikan usaha yang dibutuhkan untuk membangun sistem perangkat lunak [15].Estimasi biaya perangkat lunak menjadi pondasi krusial dalam tahapan pengembangan perangkat lunak dan juga menjadi kunci untuk mengelola aspek yang kompleks seperti biaya, waktu, dan kualitas dalam sebuah proyek [16].

Estimasi biaya perangkat lunak melibatkan berbagai aktivitas untuk menentukan jumlah sumber daya, waktu, dan biaya yang diperlukan untuk menyelesaikan suatu proyek perangkat lunak. Proses ini mencakup analisis mendalam terhadap kebutuhan proyek, spesifikasi teknis, dan cakupan kerja yang akan dilakukan. Tujuannya adalah untuk memberikan gambaran yang akurat tentang upaya yang dibutuhkan, sehingga manajemen proyek dapat membuat perencanaan yang realistis dan dapat diandalkan.

Estimasi yang akurat memastikan proyek perangkat lunak berjalan lancar tanpa penundaan dan sesuai dengan anggaran yang telah ditetapkan. Namun, setiap kesalahan dalam memperhitungkan setiap aspek dan estimasi biaya bisa berpotensi mengakibatkan kegagalan proyek dalam hal jadwal, anggaran, atau pilihan yang diambil. [17].

Estimasi yang akurat dalam proyek perangkat lunak merupakan fondasi penting yang mendukung keberhasilan keseluruhan proyek. Dengan estimasi yang tepat, manajer proyek dapat merencanakan sumber daya, menetapkan target waktu, dan mengalokasikan anggaran dengan lebih efektif. Proses ini membantu dalam mengidentifikasi kebutuhan teknis dan fungsional sejak awal, sehingga tim dapat bekerja dengan panduan yang jelas dan tujuan yang terukur. Selain itu, estimasi yang tepat membantu dalam mengelola ekspektasi pemangku kepentingan dan memastikan

bahwa semua pihak yang terlibat memahami batasan dan potensi tantangan yang mungkin dihadapi.

2.4 Metode Estimasi Biaya

Ada beragam teknik yang digunakan untuk memperkirakan biaya perangkat lunak, namun teknik-teknik tersebut dapat dikelompokkan menjadi dua model: algoritmik dan non-algoritmik..

Model algoritmik menggunakan formula matematis dan algoritma untuk memperkirakan biaya berdasarkan parameter seperti ukuran perangkat lunak, kompleksitas, dan pengalaman tim. Teknik umum dalam model ini termasuk COCOMO, FPA, dan SLIM. Model ini menawarkan konsistensi dan kemungkinan analisis sensitivitas, namun memerlukan data input yang akurat dan kurang fleksibel terhadap perubahan.

Di sisi lain, model non-algoritmik mengandalkan pengalaman dan data historis dari proyek serupa untuk membuat estimasi biaya. Teknik umum dalam model ini meliputi Expert Judgment, Analogous Estimating, Delphi Technique, Top-Down Estimating, dan Bottom-Up Estimating. Model ini memberikan fleksibilitas dan memanfaatkan pengalaman praktis, tetapi rentan terhadap bias subjektif dan memerlukan data historis yang relevan.[18] .

2.5 Pengukuran Perangkat Lunak

Pengukuran perangkat lunak merupakan kegiatan yang terjadi sepanjang semua tahapan dalam proses pengembangan perangkat lunak. Dalam proses ini, berbagai produk perangkat lunak dievaluasi atau akhirnya diukur menggunakan metrik-metrik khusus untuk produk perangkat lunak. [19]. Metrik perangkat lunak terkait dengan pengukuran produk dan proses pengembangan perangkat lunak, serta berperan dalam mengarahkan dan menilai model serta alat yang digunakan dalam proses pengembangan [20].

Pengukuran perangkat lunak melibatkan penggunaan metrik-metrik yang telah ditentukan sebelumnya untuk mengevaluasi berbagai aspek produk perangkat lunak, termasuk kualitas, kinerja, keandalan, dan kompleksitas. Metrik-metrik ini membantu dalam memahami kemajuan proyek, mengidentifikasi potensi masalah, dan mengambil keputusan yang lebih tepat dalam pengembangan perangkat lunak. Selain itu, metrik perangkat lunak juga memberikan kerangka kerja untuk membandingkan produk atau proses yang berbeda, serta untuk meningkatkan praktik pengembangan perangkat lunak secara keseluruhan.

Keakuratan estimasi ukuran secara langsung dapat berdampak pada keakuratan estimasi biaya perangkat lunak [21]. Ketika pengukuran dilakukan dengan efektif, ini memberikan pemahaman yang memadai bagi organisasi pengembangan perangkat lunak untuk membuat estimasi yang dapat dipercaya dan mengidentifikasi kendala sebelumnya. Hal ini membantu dalam pengelolaan risiko dengan lebih baik dan meningkatkan kualitas produk dan proyek secara konsisten. [19].

Pengukuran yang tepat dan akurat dalam pengembangan perangkat lunak memainkan peran penting dalam menyediakan pandangan yang jelas tentang ukuran, kompleksitas, dan lingkup proyek. Dengan memahami ukuran proyek dengan baik, organisasi dapat membuat estimasi biaya yang lebih akurat, mengidentifikasi potensi risiko lebih awal, dan mengambil tindakan pencegahan yang tepat untuk mengelola risiko tersebut. Selain itu, pengukuran yang efektif juga membantu dalam mengarahkan sumber daya dengan lebih efisien, memastikan pemenuhan tenggat waktu, dan meningkatkan kualitas produk secara keseluruhan. Dengan demikian, pengukuran yang akurat memainkan peran krusial dalam kesuksesan proyek perangkat lunak dengan memberikan dasar yang kuat untuk pengambilan keputusan yang terinformasi dan pengelolaan risiko yang efektif.

2.5 Use Case Point

Metode Use Case Point (UCP) adalah teknik yang umum digunakan dalam industri perangkat lunak untuk mengestimasi biaya proyek. UCP digunakan untuk

melakukan estimasi yang lebih rinci pada proyek pengembangan perangkat lunak, berdasarkan kompleksitas dari setiap use case, sesuai hasil penelitian Gustav Karner pada tahun 1993 [22].Metode UCP memungkinkan analisis terhadap aktor, use case, serta berbagai aspek teknis dan lingkungan untuk digabungkan ke dalam sebuah persamaan.

Dengan pendekatan ini, estimasi yang dihasilkan cenderung lebih mendekati hasil yang diperoleh dari pengalaman praktisi atau pengembang perangkat lunak. UCP memungkinkan tim proyek untuk mempertimbangkan berbagai aspek yang relevan dengan perangkat lunak yang akan dikembangkan, termasuk kompleksitas fungsional, keterlibatan aktor, dan faktor teknis lainnya.

2.6 Fuzzy Use Case Point

Fuzzy use case points (FUCP) yang merupakan kombinasi antara logika fuzzy dan use case point, dapat digunakan untuk menutupi kelemahan dari use case point .Kontribusi dari penelitian ini adalah meningkatkan akurasi estimasi upaya tanpa memerlukan investasi tambahan dalam hal waktu dan biaya, sambil juga menyederhanakan metode estimasi tanpa mengurangi tingkat keakuratannya [23].

FUCP memanfaatkan logika fuzzy untuk menangani ketidakpastian dan ambiguitas yang sering terjadi dalam pengukuran dan estimasi perangkat lunak. Dengan pendekatan ini, FUCP dapat memberikan hasil estimasi yang lebih adaptif dan fleksibel, terutama dalam situasi di mana parameter-parameter yang diperlukan untuk estimasi tidak dapat diukur secara pasti. Dengan demikian, FUCP menjadi alternatif yang menjanjikan dalam meningkatkan keakuratan estimasi upaya dalam pengembangan perangkat lunak.

2.7 Tahap-Tahap Estimasi Use Case Point

Pada tahap penghitungan menggunakan metode use case point, langkah-langkah dilakukan sesuai dengan kerangka yang telah diuraikan oleh Gustav Carner seperti berikut:

2.7.1 Unadjusted Use Case Point

Unadjusted Use Case Point didapatkan dengan mencari Unadjusted Actor Weight (UAW) dan Unadjusted Use Case Weight (UUCW). Untuk mendapatkan UAW, kita harus mengidentifikasi jumlah aktor pada use case diagram dan mengklasifikasikannya berdasarkan tingkat kompleksitasnya, yaitu Simple, Average, atau Complex. Setelah itu, berikan bobot pada setiap aktor sesuai klasifikasinya.

Tabel 2. 1 *Unadjusted Actor Weight*

Kompleksitas	Definisi	Bobot
Simple	Aktor mempresentasikan sistem lain dengan API	1
Average	Aktor berinteraksi dengan sistem lain melalui protocol adalah manusia yang berinteraksi dengan line terminal	2
Complex	Jika aktor berinteraksi dengan melalui antarmuka grafis (GUI)	3

Tabel 2. 2 *Unadjusted Actor Weight*

Kompleksitas	Definisi	Bobot
Simple	1-3 Transaksi	5
Average	4-7 Transaksi	10
Complex	>7 Transaksi	15

Nilai dari setiap use case akan dikalikan berdasarkan bobot dari setiap kompleksitas use case weight. Setelah nilai UAW dan UUCW didapatkan maka dapat dijumlahkan untuk mendapat hasil dari **Unadjusted Use Case Point (UUCP)**.

2.7.2 Technical Complexity Factors (TCF)

Technical Complexity Factor (TCF) menilai kompleksitas teknis proyek perangkat lunak berdasarkan atribut seperti keandalan, ukuran, integrasi, kinerja, dan penggunaan perangkat khusus. Skor TCF membantu menyesuaikan estimasi upaya dan sumber daya, dengan proyek ber-TCF tinggi memerlukan lebih banyak waktu dan biaya. TCF mendukung perencanaan yang tepat, identifikasi risiko, dan alokasi sumber daya yang efisien[24].

Tabel 2. 3 *Technical Complexity Factor (TCF)*

TF	Technical Factor	Bobot
T1	Sistem Terdistribusi	2
T2	Kinerja	1
T3	Efisiensi Pengguna Akhir	1
T4	Pemrosesan Internal yang Kompleks	1
T5	Kemampuan Melakukan Penggunaan Kembali Kode	1
T6	Kemudahan Instalasi	0.5
T7	Kemudahan Penggunaan	0.5
T8	Dukungan Antar Platform	2
T9	Kemudahan untuk Mengubah	1
T10	Konkurensi	1
T11	Fitur Keamanan Khusus	1
T12	Pelatihan untuk staff	1
T13	Akses Pihak ketiga	1

Akumulasi dari perkalian bobot dengan nilai akan dilakukan untuk mendapatkan total TF, yang kemudian digunakan dalam menghitung Technical Complexity Factor (TCF).

2.7.3 Environmental Complexity Factors (ECF)

Environmental Complexity Factor (ECF) digunakan dalam memproyeksikan proyek perangkat lunak, menggambarkan tingkat kompleksitas lingkungan di mana perangkat lunak akan diimplementasikan. ECF mempertimbangkan berbagai aspek, termasuk integrasi sistem, ketersediaan platform, keamanan, dan regulasi. Pemanfaatan ECF membantu dalam menyesuaikan perkiraan upaya dan sumber daya yang dibutuhkan, yang sangat penting dalam perencanaan dan manajemen efisien proyek perangkat lunak[24].

Tabel 2. 4 *Environmental Complexity Factors (ECF)*

EF	Environment Factor	Bobot
E1	Keakraban dengan metode pengembangan yang digunakan	
E2	Pengalaman Aplikasi	
E3	Pengalaman Berorientasi Objek	
E4	Menguasai Kemampuan Analisis	
E5	Motivasi	
E6	Kebutuhan yang Stabil	
E7	Pekerja Paruh Waktu	
E8	Bahasa Pemrograman yang Sulit	

Setelah mendapatkan nilai Environmental Complexity Factors (EF) untuk setiap faktor lingkungan, total dari hasil perkalian nilai bobot EF dengan nilai yang diberikan akan dijumlahkan. Dengan melakukan langkah-langkah perhitungan tersebut, peneliti akan mendapatkan nilai total EF dari hasil perhitungan sebelumnya. Proses perhitungan ini dilakukan dengan menggunakan rumus yang telah ditentukan sebelumnya.

2.7.4 Skala Nilai

Technical dan Environment akan di nilai berdasarkan skala antara 0-5 yang setiap nilai tersebut memiliki ketentuan masing-masing, berikut merupakan penjelasan setiap nilai untuk menghitung TF pada tabel 2.5.

Tabel 2. 5 Skala Nilai

Nilai	Deskripsi	Kriteria Setiap Faktor	Uraian
0	Not present or no influence	Tidak Memiliki Pengaruh dengan proyek	Ketika Faktor tidak ada hubungannya dengan proyek
1	Incidental influence	Memiliki Pengaruh yang kurang penting dalam proyek	Ketika Faktor Proyek ini ada hubungannya tetapi tidak terlalu berpengaruh secara significant
2	Moderate influence	Memiliki Pengaruh biasa pada proyek pengembangan	Ketika faktor proyek ini memiliki pengaruh tetapi tidak terlalu berpengaruh pada pengembangan

3	Average influence	Memiliki Pengaruh Rata-Rata pada proses pengembangan proyek	Ketika Faktor sering digunakan dalam pengembangan proyek
4	Significant influence	Memiliki Pengaruh yang cukup kuat pada proses pengembangan proyek	Ketika Faktor ini memiliki hal yang harus ada dan berperan cukup segnifikan
5	Strong influence throughout	Memiliki Pengaruh yang kuat pada proses pengembangan proyek	Ketika faktor memiliki hubungan yang kuat dengan pengembangan sistem

2.7.5 Use Case Point

Ketika nilai Technical Complexity Factor (TCF), yang menunjukkan kompleksitas teknis dari proyek perangkat lunak, dan Environmental Complexity Factor (ECF), yang menggambarkan kompleksitas lingkungan tempat perangkat lunak diterapkan, telah diperoleh maka sudah dapat di cari nilai dari *Use Case Point* proyek terserbut.

UCP = Use Case Point

UUCP = Unadjusted Use Case Point

TCF = Technical Factor

ECF = Environment Complexity Factor

2.7.6 Perhitungan Effort Rate

Saat memperkirakan usaha yang dibutuhkan untuk mengembangkan perangkat lunak, digunakan parameter yang disebut Effort Rate (ER) yang menentukan jumlah total jam kerja yang diperlukan untuk setiap use case point. Penelitian sebelumnya telah menetapkan ER sebesar 8,2 jam kerja per use case point, yang memegang peran penting dalam menilai biaya proyek dan menentukan usaha yang diperlukan untuk mengembangkan fitur-fitur dalam sistem perangkat lunak. Namun, penelitian terkini mengadopsi perhitungan dari Albrecht A, yang menetapkan ER sebesar 20 jam kerja per use case point, terutama digunakan untuk proyek baru atau yang tidak memiliki data historis. Biasanya, nilai ER berkisar antara 15 hingga 30 jam kerja per use case point.

2.8 Tahap-Tahap Fuzzy Use Case Point

Tahap-Tahap pada fuzzy use case point dalam perhitungan effort mengikuti tahap yang ada pada use case point yang sudah dihitung sebelumnya dan yang menjadi pembeda adalah nilai klasifikasi Use Case Weight yang dimodifikasi menggunakan fuzzy logic

2.8.1 Estimasi Fuzzy Use Case Point

Logika Fuzzy diperkenalkan oleh Prof. Zadeh dari Universitas California di Berkeley pada tahun 1965 [25]. Konsep Fuzzy Use Case Point pertama kali diusulkan oleh Hariyanto dan Wahono dalam penelitian berjudul "Estimasi Proyek Pengembangan Perangkat Lunak dengan Fuzzy Use Case Point" pada tahun 2015 [26]. Fuzzy Use Case Point merupakan sebuah inovasi dari metode Use Case Point, dimana pendekatan logika fuzzy digunakan sebagai pengganti nilai tegas dalam mengklasifikasikan tingkat kompleksitas use case.

Tabel 2. 6 *Unadjusted Fuzzy Use Case Weight* [26]

ΣTransaksi	Bobot
1	5
2	5
3	6.45
4	7.5
5	8.55
6	10
7	11.4
8	12.5
9	13.6
10	15

Unadjusted Fuzzy Use Case Point adalah hasil yang didapatkan setelah menjumlahkan Unadjusted Actor Weigjt dengan Unadjusted Fuzzy Use Case Weight

2.8.2 Klasifiasi Use Case

Klasifikasi use case melibatkan pengelompokkan use case berdasarkan karakteristik dan perannya dalam sistem. Use case utama merupakan fungsionalitas inti dari sistem yang memegang peranan penting dalam mencapai tujuan utama sistem. Sementara itu, supporting use case memberikan tambahan fungsionalitas yang mendukung dan melengkapi fungsionalitas utama. Selain itu, exceptional use case menangani skenario yang tidak terduga atau tidak biasa, sedangkan business rule use case mendefinisikan implementasi aturan bisnis yang harus diikuti oleh sistem. Subfunction use case membantu memecah fungsionalitas kompleks menjadi bagian-bagian yang lebih kecil dan terkelola, sementara extension use case memperluas fungsionalitas use case utama dengan menggambarkan skenario tambahan. Dengan memahami klasifikasi use case ini, tim pengembangan dapat merencanakan dan mengelola pengembangan sistem secara lebih terstruktur dan efektif[27].

2.8.3 Fuzzykasi

Fuzifikasi merupakan langkah penting dalam menerapkan konsep logika fuzzy, di mana nilai-nilai yang awalnya tajam atau pasti dikonversi menjadi nilai-nilai kabur atau ambigu dalam suatu sistem atau analisis. Dalam konteks penggunaan logika fuzzy, variabel dan kondisi tidak terbatas pada nilai biner saja, melainkan dapat memiliki rentang nilai yang beragam di antara ekstrem yang mungkin ada. Proses fuzifikasi

melibatkan pembentukan fungsi keanggotaan yang secara spesifik menentukan seberapa kuat atau seberapa dekat nilai pasti dapat dimasukkan ke dalam himpunan kabur yang sesuai. Dengan menggunakan teknik ini, sistem yang menerapkan logika fuzzy mampu menangani ketidakpastian dan ambiguitas dalam data masukan, memungkinkan pengambilan keputusan yang lebih adaptif dan responsif terhadap situasi yang kompleks serta bervariasi [28].

2.8.4 Fuzzy Inference System

Fuzzy Inference System (FIS), juga dikenal sebagai Fuzzy Inference System, adalah sistem yang memanfaatkan prinsip logika fuzzy untuk mengubah masukan yang kabur menjadi keluaran yang kabur pula. FIS terdiri dari tiga komponen utama, yaitu himpunan fuzzy, basis aturan, dan mekanisme inferensi. Himpunan fuzzy menggambarkan kategori variabel masukan dan keluaran serta menetapkan fungsi keanggotaan untuk setiap kategori tersebut, sementara basis aturan menghubungkan kondisi masukan dengan tindakan yang sesuai melalui aturan-aturan logika fuzzy. Mekanisme inferensi, yang menjadi inti dari FIS, mengolah masukan kabur melalui basis aturan untuk menghasilkan output dalam bentuk variabel kabur pula [29].

FIS menerima masukan dalam bentuk variabel kabur dan melalui proses yang melibatkan fuzzifikasi, penerapan aturan fuzzy, dan defuzzifikasi, menghasilkan keluaran yang sesuai dalam bentuk variabel kabur juga. Sistem ini digunakan dalam berbagai aplikasi seperti pengendalian otomatis, pengambilan keputusan, pengenalan pola, dan pemodelan sistem kompleks. Dengan memanfaatkan logika fuzzy, FIS mampu mengatasi ketidakpastian dan ambiguitas dalam masukan serta memberikan output yang lebih adaptif dan responsif terhadap kondisi yang kompleks .

2.8.5 Defuzzykasi

Defuzzifikasi merupakan tahap penting dalam penggunaan logika fuzzy yang melibatkan konversi keluaran kabur dari sistem logika fuzzy menjadi nilai tegas atau pasti yang dapat digunakan dalam aplikasi praktis. Dalam logika fuzzy, keluaran sistem sering kali dinyatakan dalam himpunan fuzzy yang menunjukkan tingkat keanggotaan

dari setiap nilai dalam variabel keluaran. Proses defuzzifikasi bertujuan untuk menghasilkan nilai yang lebih konkret dan jelas yang merepresentasikan output sebenarnya dari sistem fuzzy, memungkinkan pengambilan keputusan yang lebih akurat dan implementasi tindakan yang lebih tepat [30]

Metode-metode defuzzifikasi yang umum digunakan, seperti metode centroid, metode maksimum, dan metode rata-rata tertimbang, memiliki pendekatan yang berbeda dalam menentukan nilai tegas dari himpunan fuzzy keluaran. Metode centroid, sebagai contoh, menghitung titik tengah atau centroid dari area yang dianggap sebagai bagian dari himpunan fuzzy keluaran, sementara metode maksimum memilih nilai crisp yang memiliki tingkat keanggotaan tertinggi dalam himpunan fuzzy. Melalui proses defuzzifikasi, keluaran yang semula kabur diubah menjadi nilai konkret yang dapat dipahami dan digunakan dalam aplikasi praktis, menjembatani kesenjangan antara teori logika fuzzy dan implementasi di dunia nyata.

2.8.6 Perhitungan Fuzzy Use Case Point

Untuk Fuzzy Use Case Point juga sama seperti Use Case Point akan di kali semua nilai dari FUCP, TCF, dan ECF untuk mendapatkan nilainya. Setelah mendapatkan Use Case Point dan Fuzzy Use Case Point dari perhitungan menggunakan rumus maka selanjutnya akan dilanjutkan dengan menghitung effort rate