BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

Estimasi Pengembangan Perangkat Lunak Dengan Use Case Size Point

Puguh Jayadi*, Juwari, Muh.Nur Luthfi Azis, Kelik Sussolaikah
Teknik Informatika, Fakultas Teknik, Universitas PGRI Madiun, Jawa Timur
Email: [1*]puguh.jayadi@unipma.ac.id, [2]juwari@unipma.ac.id, [3]nur.azis@unipma.ac.id, [4]kelik@unipma.ac.id
Email Penulis Korespondensi: puguh.jayadi@unipma.ac.id

Abstrak− Dalam kegiatan pengembangan perangkat lunak salah satu tahap yang perlu diketahui oleh manager proyek dan klien adalah estimasi effort. Dari kegiatan estimasi digunakan sebagai acuan dalam penentuan waktu dan personil yang dilibatkan dalam pengembangan perangkat lunak. Ada beberapa metode yang digunakan dalam estimasi perangkat lunak seperti Function Point, COCOMO, Use Case Point hanya saja masih terdapat beberapa kekurangan seperti adanya kecenderungan subyektif dalam pengukuran kompleksitas dan kurangnya perhitungan pada faktor teknis maupun faktor lingkungan. Estimasi yang digunakan pada penelitian ini menggunakan metode Use Case Size Point yang merupakan pengembangan dari metode Use Case Point. Hasil dari Use Case Size Point yang digunakan memberikan hasil evaluasi effort yang lebih akurat dengan 0,01 Mean Magnitude of Relative Error (MMRE) dan 0,01 Mean Magnitude of Error Relative (MMER).

Kata Kunci: Estimasi; Use Case Point; Use Case Size Point; Mean Magnitude of Relative Error; Mean Magnitude of Error Relative

Abstract− In software development activities one of the stages that need to be known by the project manager and the client is the estimated effort. The estimation activity is used as a reference in determining the time and personnel involved in software development. There are several methods used in software estimation such as Function Point, COCOMO, and Use Case Point. However, there are still some shortcomings such as the subjective tendency to measure complexity and lack of calculation on technical factors and environmental factors. The estimation used in this study uses the Use Case Size Point method which is the development of the Use Case Point method. The results of the Use Case size Point were used to provide a more accurate evaluation of effort with 0.01 Mean Magnitude of Relative Error (MMRE) and 0.01 Mean Magnitude of Relative Error (MMER).

Keywords: Estimation; Use Case Point; Use Case Size Point; Mean Magnitude of Relative Error; Mean Magnitude of Error Relative.

1. PENDAHULUAN

Sebelum suatu perangkat lunak dibangun atau dikode diperlukan adanya tahapan estimasi yang digunakan untuk mengetahui effort atau upaya yang dibutuhkan dalam pengembangan perangkat lunak tersebut [1]. Pendekatan sistematis dalam estimasi pengukuran perangkat lunak menjadi penting untuk manajemen dan perencanaan perangkat lunak [2], [3]. Estimasi yang dimaksud dijadikan dasar oleh manager proyek perangkat lunak untuk mengetahui scope (besaran), biaya, jangka waktu serta personil yang dibutuhkan [4], [5]. Jika semuanya jelas, maka pihak manager dan personil yang terlibat dalam proses pengembangan akan lebih mudah menyusun tahapan pekerjaan sesuai waktu yang telah ditentukan. Dari pihak klien juga akan lebih mudah mengawasi waktu pengerajaan perangkat lunak tersebut sesuai biaya yang dibayarkan.

Estimasi biaya dapat digunakan untuk menentukan kinerja suatu proyek. Estimasi biaya yang akurat mengarah ke proyek yang sukses, dan estimasi yang tidak akurat menghasilkan kegagalan proyek[3]. Ada beberapa teknik atau metode yang menawarkan cara untuk estimasi pengembangan perangkat lunak seperti Analogy Based Estimation (ABE) yang menggunakan model regresi berganda dan teknik penilaian dari para ahli [3], Case Based Reasoning yang didasari dengan asumsi setiap proyek perangkat lunak yang memiliki jumlah effort yang sama [6], Function Point yang dihitung dengan besaran suatu fungsional perangkat lunak [7], [8], COCOMO II yang menggunakan UFP, Tabel Scale Driver, Table Effort Mulplier [9], [10]. Dari berbagai penelitiaan yang sudah dilakukan masih ada kecenderungan yang bersifat subjektif dan adanya ketidakpastian karena tidak adanya standar perhitungan yang pasti [11].

Penelitian yang dilakukan ini menggunakan metode Use Case Size Point yang merupakan perbaikan dari metode Use Case Point yang lebih difokuskan pada struktur internal Use Case. Proyek pengembangan perangkat lunak yang digunakan pada penelitian ini adalah dari pengembangan portal dan sistem informasi anggota dari Perkumpulan Pendidikan Islam Usia Dini (PIAUD). Proyek perangkat lunak tersebut dikembangkan berbasis website dengan lama pekerjaan sekitar 1228 jam. Dari hasil akhir perhitungan perhitungan Use Case Size Point memberikan nilai 1241 jam. Jika diukur akurasinya mendekati dengan nilai aktual yang dihitung dengan MRE menghasilkan 0,01 dan dihitung dengan MER menghasilkan 0,01.

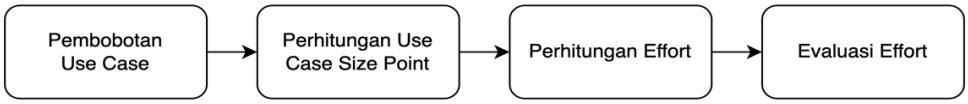
2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Langkah-langkah yang diterapkan pada penelitian ini seperti yang ada pada Gambar 1. Tahapan pertama dimulai dari pembobotan *Use Case* dalam proyek perangkat lunak. Penelitian ini menggunakan data proyek dari pengembangan sistem manajemen dan portal Pendidikan Islam Anak Usia Dini (PIAUD). Metode yang diusulkan

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

pada penelitian ini adalah menggunakan *Use Case Size Points*. Semua data yang digunakan dalam penelitian ini berasal dari wawancara kepada programmer serta manager proyek perangkat lunak PIAUD. Dari hasil pembobotan *Use Case* kemudian dihitung dengan metode *Use Case Size Point* yang nantinya didapatkan nilai *effort*. Setelah nilai *effort* sudah dihitung kemudian dilakukan tahapan evaluasi untuk mengetahui tingkat akurasi dengan nilai *effort* aktual (asli)



Gambar 1. Tahap Penelitian

2.2 Metode Use Case Size Point

Use Case adalah cara yang biasa digunakan untuk merepresentasikan seperti apa sebuah sistem berinteraksi dengan aktor yang ada di lingkungannya. *Use Case Points* pertama kali dikembangkan oleh Gustav Karner di tahun 1993 [12]. Use Case Points berasal dari sebuah Use Case Diagram yang dibobotkan dengan beberapa rumus yang digunakan untuk mengetahui *effort* atau upaya dalam satuan jam yang dibutuhkan dalam mengembangan perangkat lunak [13]. Penggunaan Use Case Point terdapat beberapa kelemahan yang didapatkan dari beberapa penelitian di tahun sesudahnya seperti adanya ketidakpastian pada faktor biaya dan kurang detail pada penentuan klasifikasinya [14]. Untuk menangani kekurangan dari *Use Case Point* maka dikembangkannya suatu metode *Use Case Size Point* yang lebih menjabarkan perhitungan dalam beberapa metrik yang lebih detail dengan memperhitungkan faktor teknis dan lingkungan ketika pengembangan sistem [15]. Tahapan-tahapan pada *Use Case Size Point* dijelaskan pada sub berikutnya.

2.1.1 Total Kompleksitas Aktor (*Total Complexity of Actor*) - TPA

Setiap aktor yang ada dalam Use Case memiliki kompleksitas (CA) yang ditentukan sesuai dengan transaksi data atau informasi yang diminta dan diterima dari Use Case. Detail dari bobot kompleksitas aktor terdapat pada Tabel 1. Total Kompleksitas Aktor (TPA) dihitung dengan persamaan 1.

Tabel 1. Tabel Kompleksitas Aktor

Kompleksitas (*CA*)	Jumlah Transaksi	Bobot
Sederhana (*Simple*)	<= 5	2
Rata-rata (*Average*)	6 - 10	4
Komplek (*Complex*)	> 10	6

$$TPA = \sum_{i=1}^{n} CA_i \quad (1)$$

2.1.2 Total Kompleksitas Prekondisi (*Total Complexity of the Preconditions*) - TPPrC

Setiap kondisi dari Use Case mempunyai tingkat kompleksitas yang diperoleh dari total ekspresi logika yang harus ada sebelumnya [15]. Contohnya seperti setiap pengguna yang akan login haruslah pengguna yang memiliki identitas dan password, pengguna yang bisa mengubah data profil adalah pengguna yang sudah berhasil login. Detail dari bobot kompleksitas prekondisi terdapat pada Tabel 2. Total Kompleksitas Prekondisi (TPPrC) dihitung dengan persamaan 2.

Tabel 2. Tabel Kompleksitas Prekondisi

Kompleksitas (*CPrC*)	Jumlah Transaksi	Bobot
Sederhana (*Simple*)	1 logika	1
Rata-rata (*Average*)	2 – 3 logika	2
Komplek (*Complex*)	> 3	3

$$TPRrC = \sum_{i=1}^{n} CPrC_i \quad (2)$$

2.1.3 Kompleksitas Skenario Utama (*Main Scenario Classification*) – TPCP

Kompleksitas pada skenario *Use Case* diidentifikasi menurut total tahapan yang diperlukan untuk mencapai skenario utama dalam suatu entitas (PCP). Sebagai contoh pengguna memilih menu profil kemudian sistem menampilkan data profil sesuai pengguna dan akhirnya pengguna bisa melakukan update data profil. Entitas yang dipakai pada skenario ini adalah pengguna. Detail dari bobot kompleksitas skenario utama terdapat pada Tabel 3 [15]. Total Kompleksitas Skenario Utama (TPCP) dihitung dengan persamaan 3.

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

Tabel 3. Tabel Skenario Utama

Kompleksitas (*CP*)	Langkah + Entitas	Bobot
Sangat Sederhana (*Very Simple*)	≤ 5	4
Sederhana (*Simple*)	6 – 10	6
Rata-rata (*Average*)	11 – 15	8
Komplek (*Complex*)	16 – 20	12
Sangat Komplek (*Very Complex*)	> 20	16

$$PCP = \sum_{i=1}^{n} CP_i \quad (3)$$

2.1.4 Kompleksitas Skenario Alternatif (*Alternative Scenario Classification*) – TPCA

Kompleksitas skenario alternatif sama dengan skenario utama yang membedakan pada langkah dan entitas yang digunakan tidak harus ada pada pada Use Case. Detail dari bobot kompleksitas skenario alternatif terdapat pada Tabel 4 [15]. Total Kompleksitas Skenario Utama (TPCP) dihitung dengan persamaan 4.

Tabel 4. Tabel Skenario Alternatif

Kompleksitas (*CA*)	Langkah + Entitas	Bobot
Sangat Sederhana (*Very Simple*)	≤ 5	4
Sederhana (*Simple*)	6 – 10	6
Rata-rata (*Average*)	11 – 15	8
Komplek (*Complex*)	16 – 20	12
Sangat Komplek (*Very Complex*)	> 20	16

$$TPCA = \sum_{i=1}^{n} CA_i \quad (4)$$

2.1.5 Total Komplesitas Penanganan Kesalahan (*Total Complexity Exception*) – TPE

Setiap mekanisme penanganan kesalahan (*exception handling*) yang ada pada *Use Case* harus dianalisis menurut kompleksitasnya dan ditentukan oleh total ekspresi logika yang dicek/diuji untuk mendeteksi terjadinya kesalahan. Contohnya seperti saat pengguna melakukan login dengan mengisi identitas atau password salah maka sistem akan menampilkan peringatan kesalahan authentikasi. Detail dari bobot kompleksitas penanganan kesalahan (CE) terdapat pada Tabel 5 [15]. Total Kompleksitas Penanganan Kesalahan (TPE) dihitung dengan persamaan 5.

Tabel 5. Tabel Komplesitas *Exception*

Kompleksitas (*CE*)	Ekspresi yang diuji	Bobot
Sederhana (*Simple*)	1 ekspresi logika	1
Rata-rata (*Average*)	2 – 3 ekspresi logika	2
Komplek (*Complex*)	> 3 ekspresi logika	3

$$TPE = \sum_{i=1}^{n} CE_i \quad (5)$$

2.1.6 Total Komplesitas Pascakondisi (*Total Complexity of Postconditions*) - TPPoC

Kompleksitas Pascakondisi (CPoC) ditentukan menurut jumlah entitas terkait dari sebuah Use Case dan langkah-langkah yang ada sebagai respon dari sistem terhadap tindakan dari pengguna. Contoh dari kompleksitas pasca Pascakondisi adalah ketika pengguna yang sudah berhasil login dan mengubah profil maka akan ada notifikasi berhasil dan data pada entitas pengguna berubah. Detail dari bobot kompleksitas pascakondisi (*CPoC*) terdapat pada Tabel 6 [15]. Total Kompleksitas *Postconditions* (*TPPoC*) dihitung dengan persamaan 6.

Tabel 6. Tabel Komplesitas Pascakondisi

Kompleksitas (*CPoC*)	Jumlah Entitas + Langkah	Bobot
Sederhana (*Simple*)	≤ 3	1
Rata-rata (*Average*)	4 – 6	2
Komplek (*Complex*)	> 6	3

$$TPPoC = \sum_{i=1}^{n} CPoC_i \quad (6)$$

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

2.1.7 *Unadjusted Use Case Size Point* - UUSP

Unadjusted Use Case Size Point (UUSP) diperoleh dari penjumlahan Total Kompleksitas Aktor (TPA), Total Kompleksitas Prekondisi (TPPrc), Total Kompleksitas Skenario Utama (PCP), Total Kompleksitas Skenario Alternatif (TPCA), Total Kompleksitas Pascakondisi (TPPoC) dan Total Kompleksitas *Penanganan Kesalahan* (TPE). *Unadjusted Use Case Size Point* (UUSP) dihitung dengan persamaan 7.

$$UUSP = TPA + TPPrC + PCP + TPCA + TPE + TPPoC \quad (7)$$

2.1.8 Faktor Penyesuaian Teknis (*Technical Adjustment Factor*) - FTA

Perhitungan *Use Cse Size Point* (UUSP) juga melibatkan faktor teknis serta faktor lingkungan. Pada faktor teknis didapatkan dari seperti apa kesulitan dalam proyek yang akan dikembangkan. Masing-masing nilai faktor teknis berkaitan dengan karakteristik teknis pada pengembangan perangkat lunak yang memiliki nilai bobot 0 hingga 5. Nilai 0 mewakili faktor tidak berpengaruh pada proses pengembangan dan 5 mewakili faktor sangat berpengaruh pada proses pengembangan perangkat lunak [15]. Detail dari bobot kompleksitas faktor teknis (TF) terdapat pada Tabel 7 [15]. Total penyesuaian teknis (FTA) dihitung dengan persamaan 8.

Tabel 7. Tabel Technical Adjustment

Faktor (TF)	Faktor	Bobot
TF1	Merepresentasikan komunikasi data yang ada pada sistem (*Data communication*)	0-5
TF2	Proses terdistribusi dengan sistem atau perangkat lain (*Distributed processing*)	0-5
TF3	Kehandalan sistem terhadap permintaan dan tanggapan data atau instruksi (*Performance*)	0-5
TF4	Kelengkapan sistem untuk menunjang kebutuhan pengguna (*Equipment utilization*)	0-5
TF5	Kapasistas Transaksi (*Transaction Capacity*)	0-5
TF6	Data yang disimpan dan diproses oleh sistem secara online (*On-line input of data*)	0-5
TF7	Sistem dapat meningkatkan efisiensi pengguna (*User efficiency*)	0-5
TF8	Komponen pada sistem yang dapat diperbarui secara online (*On-line update*)	0-5
TF9	Kode dapat digunakan kembali pada module atau menu lain (*Code reuse*)	0-5
TF10	Sistem terdapat proses yang rumit (*Complex processing*)	0-5
TF11	Instalasi yang mudah (*Easiness of deploy*)	0-5
TF12	Kemudahan operasi/penggunaan pada sistem maupun basisdata *(Easiness operation)*	0-5
TF13	Sistem dapat dikembangkan multi platform *(Many places)*	0-5
TF14	Mudah untuk diubah (*Facility of change*)	0-5

$$FTA = 0.65 + (0.01 * \sum_{i=1}^{n} TF_i) \quad (8)$$

2.1.9 Faktor Penyesuaian Lingkungan - *Environtment Adjustment Factor* (FAA)

Faktor lingkungan dihitung dari seperti apa kondisi yang ada selama pengembangan perangkat lunak [4]. Setiap nilai faktor lingkungan berkaitan dengan biaya pengembangan perangkat lunak memiliki bobot antara 0 hingga 5. nilai 0 bermakna faktor tidak ada pengaruh pada biaya pengembangan dan 5 bermaksna faktor sangat berpengaruh pada biaya pengembangan perangkat lunak [15]. Detail dari bobot faktor lingkungan (EF) terdapat pada Tabel 8 [15]. Total penyesuaian lingkungan (FAA) dihitung dengan persamaan 9.

Tabel 8. Tabel Environtment Factor

Faktor (EF)	Faktor	Bobot
EF1	Proses pengembangan sistem yang diterapkan (*Formal development process existence*)	0-5
EF2	Pengalaman pengembangan aplikasi/sistem (*Application Experience*)	0-5
EF3	Pengalaman menggunakan bahasa pemrograman (*Experience of the team with the used Programming Language*)	0-5
EF4	Kemampuan analis mengumpulkan kebutuhan pengguna yang harus ada pada sistem (*Presence on an experienced analyst*)	0-5
EF5	Kebutuhan pengguna yang stabil atau tidak berubah-ubah (*Stable Requirements*)	0-5

$$FAA = (0.01 * \sum_{i=1}^{n} EF_i) \quad (9)$$

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

2.1.10 *Use Case Size Point* (USP)

Nilai akhir dari *Use Case Size Point* dihitung dari perkalian *Unadjusted Use Case Size Point* (UUSP) dengan pengurangan dari nilai *Technical Adjustment Factor* (FTA) dan *Environtment Adjustment Factor* (EAF). USP dihitung dengan persamaan 10.

$$USP = UUSP * (FTA - FAA) \quad (10)$$

2.1.11 Effort Estimasi

Dari nilai USP digunakan untuk menghitung *effort* dengan satuan jam dengan cara nilai *Use Case Size Point* (USP) dibagi dengan besaran nilai produktifitas. Nilai produktitas yang dipakai 0,38 [15]. Perhitungan *effort* didapatkan dengan persamaan 11.

$$Effort = USP \ / \ Productivity \quad (11)$$

2.2 Evaluasi Effort

Pada penelitian ini menggunakan 2 metode untuk mengevaluasi nilai effort, yaitu *Mean Magnitude of Relative Error* (MMRE) yang merupakan kriteria yang sangat umum digunakan untuk mengevaluasi model estimasi *effort* perangkat lunak [16] [17], [18]. Ketika menggunakan MMRE untuk evaluasi, hasil yang bagus ditunjukkan dengan nilai yang rendah. Nilai MMRE merupakan nilai rata-rata dari sejumlah nilai MRE yang didapat dengan rumus pada persamaan 12.

$$MRE = \frac{|Actual\ Effort - Predicted\ Effort|}{Actual\ Effort} \quad (12)$$

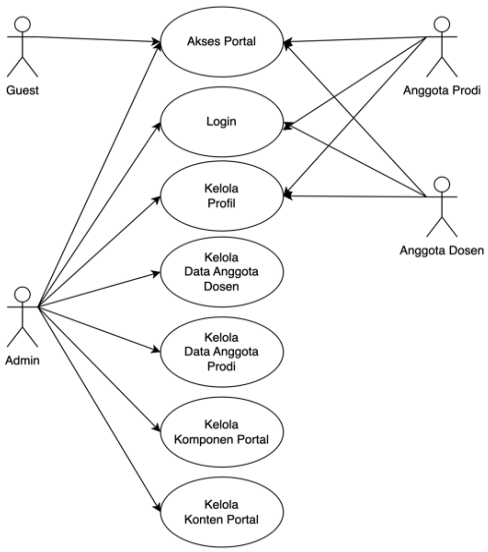
Metode kedua adalah *Mean Magnitude of Error Relative* (MMER) sebagai alternatif metode lain untuk mengevaluasi model estimasi effort perangkat lunak [16]–[18]. Nilai MMER merupakan nilai rata-rata dari sejumlah nilai MER yang didapat dengan rumus pada persamaan 13.

$$MER = \frac{|Actual\ Effort - Predicted\ Effort|}{Predicted\ Effort} \quad (13)$$

3. HASIL DAN PEMBAHASAN

3.1 Analisis Uji Validitas dan Reliabilitas

Skenario *Use Case Diagram* yang digunakan dalam pengembangan sistem PIAUD adalah yang ada pada gambar 2. Dalam *Use Case* tersebut terpadat 3 aktor yang berhubungan dengan 7 entitas dalam sistem. *Use Case* yang sudah dibuat kemudian dianalisis untuk mengetahui bobot dari setiap aktor dan entitasnya. Dengan data yang berasal dari wawancara ke programmer dan manager proyek perangkat lunak PIAUD maka dilakukan perhitungan *effort.*



Gambar 2. Use Case sistem PIAUD

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

Tahapan pertama adalah menghitung TPA. Perhitungan tersebut berdasarkan pada jumlah transaksi data yang diminta dan diterima oleh aktor dari masing masing *Use Case* [4], [15]. Hasil perhitungan TPA terdapat pada tabel 9 dan dihitung dengan persamaan 1. Aktor yang dilibatkan dalam *Use Case* adalah 3 dan masing-masing aktor memiliki jumlah transaksi diatas 10 terhadap setiap *Use Case*.

Tabel 9. Tabel Hasil TPA

Kompleksitas	Jumlah Transaksi	Bobot	Aktor	Jumlah Transaksi Data	Bobot x Jumlah Aktor
Sederhana (*Simple*)	<= 5	2	Admin	> 10	6
Rata-rata (*Average*)	6 – 10	4	Anggota Dosen	> 10	6
Komplek (*Complex*)	> 10	6	Anggota Prodi	> 10	6
		Total			18

Tahapan kedua adalah menghitung TPPrC. Setiap kondisi dari *Use Case* memiliki kompleksitas yang ditentukan oleh jumlah ekspresi logika yang harus diuji sebelumnya [4], [15]. Ekspresi yang diuji bernilai akhir benar atau salah. Hasil perhitungan TPA terdapat pada tabel 10 dan dihitung dengan persamaan 2. Contoh dari kompleksitas prekondisi yang termasuk kategori *Simple* adalah: Anggota Prodi mendaftar pada sistem harus memasukkan data yang lengkap dan benar sesuai dengan yang telah ditentukan. Contoh dari kompleksitas prekondisi yang termasuk kategori *Averange* adalah: Anggota Prodi yang akan mencetak sertifikat adalah anggota yang sudah berhasil login dan merupakan Anggota Prodi yang diterima atau disetujui.

Tabel 10. Tabel Hasil TPPrC

Kompleksitas	Langkah + Entitas	Bobot	Jumlah	Bobot x Jumlah
Sederhana (*Simple*)	1	1	44	44
Rata-rata (*Average*)	2	2	8	16
Komplek (*Complex*)	3	3	0	0
	Total			60

Tahapan ketiga adalah perhitungan TPCP. Setiap skenario utama dalam *Use Case* memiliki beberapa jumlah langkah dan entitas yang harus dilalui [4], [15]. Hasil perhitungan PCP terdapat pada tabel 11 dan dihitung dengan persamaan 3. Contoh dari kompleksitas skenario utama yang termasuk kategori *Simple* adalah: Admin ketika akan memvalidasi data Anggota Prodi yang sudah terdaftar maka admin harus melakukan login terlebih dahulu. Kemudian admin membuka menu Anggota Prodi yang dilanjutkan memilih data Anggota Prodi tertentu. Sistem akan menampilkan data Anggota Prodi yang dipilih lengkap dengan data-data yang menjadi detailnya seperti kelengkapan anggota prodi, akreditasi prodi, kerjasama prodi, sarana prodi, jurnal prodi, karya prodi. Sistem memberikan kelengkapan data tersebut berasal dari beberapa entitas. Jika admin sudah memeriksa kelengkapan data Anggota Prodi maka bisa disetujui dengan mengganti status yang pilih. Sistem menyimpan status tersebut dan memberi tahukan Anggota Prodi yang bersangkutan bahwa telah diterima.

Tabel 11. Tabel Hasil TPCP

Kompleksitas	Langkah + Entitas	Bobot	Jumlah	Bobot x Jumlah
Sangat Sederhana (*Very Simple*)	≤ 5	4	4	16
Sederhana (*Simple*)	6 - 10	6	46	276
Rata-rata (*Average*)	11 - 15	8	1	8
Komplek (*Complex*)	16 - 20	12	0	0
Sangat Komplek (*Very Complex*)	> 20	16	0	0
	Total			300

Pada penelitian ini tidak terdapat perhitungan TPCA karena semua langkah dan entitas termasuk dalam skenario Utama. Maka hasil TPCA yaitu 0.

Tahapan selanjutnya adalah perhituangan TPE. Hasil perhitungan TPE terdapat pada tabel 12 dan dihitung dengan persamaan 5. Contoh dari kompleksitas penanganan kesalahan yang termasuk kategori *Simple* diantaranya: Anggota Prodi mendaftar pada sistem tapi data yang dimasukkan salah atau tidak lengkap maka sistem akan memberikan notifikasi sebelum data tersebut disimpan. Contoh yang lain adalah ketika Admin menghapus kategori postingan pada komponen website tapi ada peringatan dari sistem karena data yang dihapus masih memiliki relasi dengan data di tabel lain.

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

Tabel 12. Tabel Hasil TPE

Kompleksitas	Langkah + Entitas	Bobot	Jumlah	Bobot x Jumlah
Sederhana (*Simple*)	1	1	38	38
Rata-rata (*Average*)	2 - 3	2	0	0
Komplek (*Complex*)	> 3	3	0	0
	Total			38

Tahapan kelima adalah menghitung perhitungan TPPoC. Penghitungan total kompleksitas pascakondisi diperoleh dengan cara menentukan detail entitas dan memperhitungkan juga jumlah entitasnya di dalam Use Case [4], [15]. Contoh dari kategori *Simple* seperti skema Admin login membutuhkan entitas User, skema admin mengubah data profil membutuhkan entitas User. Hasil setelah pengguna yang sudah mengganti profil maka akan pemberian dari sistem. Sedangkan untuk contoh kategori *Complex* seperti skema Admin menyetujui Anggota Dosen membutuhkan entitas dari Anggota Dosen, Keahlian Dosen, Pengalaman Dosen, Buku Dosen, Penelitian Dosen, Pengabdian Dosen, HKI Dosen. Setelah Admin menyetujui Anggota Dosen, maka Anggota Dosen dapat mencetak sertifikat. Hasil perhitungan TPPoC terdapat pada tabel 13 dihitung dengan persamaan 6.

Tabel 13. Tabel Hasil TPPoC

Kompleksitas	Jumlah Entitas	Bobot	Jumlah	Bobot x Jumlah
Sederhana (*Simple*)	Jumlah entitas < 3	1	44	44
Rata-rata (*Average*)	Jumlah entitas 2 - 3	2	4	8
Komplek (*Complex*)	Jumlah entitas > 6	3	4	12
	Total			64

Tahapan keenam adalah menghitung *Unadjusted Use Case Size Point* (UUSP) diperoleh dari penjumlahan Total Kompleksitas Aktor (TPA), Total Kompleksitas Prekondisi (TPPrc), Total Kompleksitas Skenario Utama (PCP), Total Kompleksitas Skenario Alternatif (TPCA), Total Kompleksitas Pascakondisi (TPPoC) dan Total Kompleksitas Penanganan Kesalahan (TPE) [4], [15]. Hasil perhitungan UUSP menggunakan persamaan 7.

$$UUSP = TPA + TPPrC + PCP + TPCA + TPE + TPPoC$$
$$= 18 + 60 + 300 + 0 + 38 + 64$$
$$= 480$$

Tahapan ketujuh adalah menghitung FTA. Setiap nilai faktor teknis berkaitan dengan karakteristik teknis pada perangkat lunak yang memiliki nilai antara 0 sampai 5 dengan nilai 0 berarti tidak berpengaruh pada proses pengembangan dan 5 berarti sangat berpengaruh pada proses pengembangan perangkat lunak [15]. Hasil perhitungan FTA terdapat pada tabel 14 dihitung dengan persamaan 8.

Tabel 14. Tabel Hasil FTA

Faktor	Skor	Keterangan
TF1	4	Komunikasi data yang ada pada sistem memiliki pengaruh cukup tinggi terhadap pengembangan sistem.
TF2	3,25	Sistem atau perangkat lain berinteraksi dengan sistem yang dikembangkan memiliki pengaruh cukup tinggi terhadap pengembangan sistem.
TF3	4	Komplesitas kehandalan sistem terhadap permintaan dan tanggapan data atau instruksi memiliki pengaruh cukup tinggi terhadap pengembangan sistem.
TF4	3,75	Kelengkapan dari sistem yang dikembangkan memiliki pengaruh cukup tinggi terhadap pengembangan sistem.
TF5	4,25	Kapasistas transaksi pada sistem yang dikembangkan miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF6	4,5	Semua data yang disimpan dan diproses oleh sistem secara online miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF7	4,5	Sistem yang dibuat dapat meningkatkan efisiensi pengguna miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF8	3,5	Terdapat beberapa komponen pada sistem yang dapat diperbarui secara online miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF9	3,75	Kode yang digunakan kembali pada module atau menu lain pada sistem miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF10	3	Proses yang rumit pada sistem miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF11	3,5	Kemudahan instalasi sistem miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

TF12	3,5	Penggunaan atau operasi sistem maupun basisdata miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF13	3,75	Sistem dapat dikembangkan multi platform miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
TF14	4	Kemudahan sistem untuk diubah miliki pengaruh yang cukup tinggi terhadap pengembangan sistem.
Total	53,25	

FTA = *0.65 + (0.01 * Total)*
= *0.65 + (0.01 * 53.25)*
= *1.18*

Tahapan kedelapan adalah menghitung FAA. Setiap faktor lingkungan yang dihitung adalah kondisi sekitar selama pengembangan perangkat lunak [4]. Setiap nilai faktor lingkungan berkaitan dengan biaya pengembangan perangkat lunak memiliki nilai antara 0 sampai 5 dengan nilai 0 berarti tidak berpengaruh pada biaya pengembangan dan 5 berarti sangat berpengaruh pada biaya pengembangan perangkat lunak [15]. Hasil perhitungan FAA terdapat pada tabel 15 dihitung dengan persamaan 9.

Tabel 15. Tabel Hasil FAA

Faktor	Skor	Keterangan
EF1	3,6	Lebih dari setengah anggota programmer berpengalaman menggunakan metode pengembangan sistem
EF2	3,8	Lebih dari setengah anggota programmer memiliki pengalaman pengembangan sistem lebih dari 3 tahun
EF3	4,1	Lebih dari setengah anggota programmer memiliki pengalaman menggunakan bahasa pemrograman lebih dari 4 tahun
EF4	4	Sistem analis memiliki pengalaman mengumpulkan kebutuhan sistem dari pengguna
EF5	3,5	Kebutuhan sistem yang dikumpulkan mengalami sedikit perubahan selama proses pengembangan sistem
Total	19,00	

FAA = *0.01 * Total*
= *0.01 * 19,00*
= *0,19*

Langkah berikutnya adalah penghitungan *Use Case Size Point* (USP). Nilai USP didapatkan dari perkalian nilai UUSP dengan pengurangan dari FTA dan FAA. Hasil perhitungan USP dihitung dengan persamaan 10.

USP = *UUSP * (FTA* - FAA*)*
= *480 * (1.18 – 0.19)*
= *476,40*

Penghitungan berikutnya adalah mendapatkan nilai *Hours of Effort*, yaitu dengan nilai *Use Case Size Point* (USP) dibagi dengan nilai produktifitas (0,38) sesuai dengan persamaan 11 [15].

Hours of Effort = *USP / 0,3839*
= *476,40 / 0,3839*
= *1241*

Langkah terakhir adalah penghitungan evaluasi *effort* dengan MMRE yang ada pada persamaan 12 dan MMER yang ada pada 13. Hasil perhitungan evaluasi *effort* terdapat pada tabel 16.

Tabel 16. Tabel Evaluasi Effort

Effort		Evaluasi	
Aktual	USP	MER	MRE
1228	*1241*	*0,01*	*0,01*

BULLETIN OF INFORMATION TECHNOLOGY (BIT)
Vol 3, No 4, Desember 2022, Hal 332 - 340
ISSN 2722-0524 (media online)
DOI 10.47065/ bit.v3i1. 408
https://journal.fkpt.org/index.php/BIT

4. KESIMPULAN

Perhitungan estimasi pengembangan perangkat lunak pada proyek PIAUD dengan menggunakan metode Use Case Size Poins memberikan nilai akurasi hampir sama dengan effort aktual atau sebenarnya. Perbedaan akurasi sebesar 0,01 bila menggunakan perhitungan evaluasi MER dan 0,01 bila dengan perhitungan evaluasi MRE. Dari nilai tersebut bisa disimpulkan bahwa estimasi menggunakan metode Use Case Size Poinsm memberikan hasil lebih mendekati effort aktual

UCAPAN TERIMAKASIH

Terimakasih peneliti ucapkan kepada semua pihak yang telah mendukung dalam penelitian ini, harapannya hasil penelitian ini bisa menjadi bahan dasar dan acuan pembelajaran serta penelitian selanjutnya.

REFERENCES

[1] W. W. Agresti, W. M. Evanco, and W. M. Thomas, “Models for Improving Software System Size Estimates during Development,” J. Softw. Eng. Appl., vol. 03, no. 01, pp. 1–10, 2010.
[2] A. Y. P. Putri, “Modifikasi Metode Function Point Dengan Menambahkan Kompleksitas Proses Bisnis Pada General System Characteristics Untuk Estimasi Biaya Perangkat Lunak,” Institut Teknologi Sepuluh Nopember, 2018.
[3] A. Kaushik, P. Kaur, N. Choudhary, and Priyanka, “Stacking regularization in analogy-based software effort estimation,” Soft Comput., vol. 26, no. 3, pp. 1197–1216, 2022.
[4] F. Ristanti, A. Dwi Herlambang, and M. C. Saputra, “Evaluasi Biaya Pengembangan Perangkat Lunak Dengan Menggunakan Metode Extended Use Case Point Dan Use Case Size Point,” 2018.
[5] N. Marcheta, “Effort Estimation Modeling Of E-Government Application Development Using Function Points Based On TOR And SRS Document,” J. Inf. Technol. Its Util., vol. 3, no. 1, p. 5, 2020.
[6] M. R. Barroek and D. E. Hadinata, “Proses Inferensi Dinamis Pada Software Effort Estimation Menggunakan Case Based Reasoning,” J. Komput. Terap., vol. 3, no. 2, pp. 77–94, 2017.
[7] S. Sariyanti, “Pengembangan Kakas Estimasi Perangkat Lunak Dengan Function Point Dan Use Case Point Untuk Praktikum Rekayasa Perangkat Lunak,” J. Sarj. Tek. Inform., vol. 6, no. 2, pp. 1–9, 2018.
[8] S. Sariyanti and Ardiansyah, “Kakas Estimasi Perangkat Lunak Menggunakan Function Point dan Use Case Point untuk Praktikum Rekayasa Perangkat Lunak di Universitas Ahmad Dahlan,” Annu. Res. Semin., vol. 3, no. 1, 2017.
[9] L. Indriyani, “Perbandingan Metode Cocomo II Dan Metode Analogy Untuk Estimasi Effort Pengembangan Software,” J. Tek. Komput. AMIK BSI, vol. 6, no. 2, pp. 174–180, 2020.
[10] O. S. Fatonah and Y. Afrizal, “Model Estimasi Biaya Perangkat Lunak Menggunakan COCOMO II (Studi Kasus Pt. X),” Informatika, pp. 1–7, 2016.
[11] M. C. Saputra et al., “Perbandingan Antara Metode Advance Use Case Point Dan Revised Use Case Point Untuk Evaluasi Biaya Pengembangan Sistem Informasi Reservasi Ruangan,” Jurnan Tek. Inform. Dan Sist. Inform., vol. 7, no. 1, 2020.
[12] G. Karner, “Resource estimation for objectory projects,” Object. Syst. SF AB, pp. 1–9, 1993.
[13] M. Hariyanto et al., “Estimasi Proyek Pengembangan Perangkat Lunak Dengan Fuzzy Use Case Points,” J. Softw. Eng., vol. 1, no. 1, pp. 54–63, 2015.
[14] M. Hariyanto and R. S. Wahono, “Estimasi Proyek Pengembangan Perangkat Lunak Dengan Fuzzy Use Case Points,” J. Softw. Eng., vol. 1, no. 1, pp. 54–63, 2015.
[15] M. R. Braz and S. R. Vergilio, “Software Effort Estimation Based on Use Cases,” in Proceedings - International Computer Software and Applications Conference, 2006, vol. 1, pp. 221–228.
[16] M. Azzeh, A. B. Nassif, and I. B. Attili, “Predicting Software Effort from Use Case Points: A Systematic Review Reputation Systems View project Capturing Emirati-Accented Speech Corpora for Applications of Speech Signal Processing View project Predicting Software Effort from Use Case Points: A Sys,” Sci. Comput. Program., vol. 204, 2021.
[17] V. Van Hai, H. L. T. K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, “Toward Improving the Efficiency of Software Development Effort Estimation via Clustering Analysis,” IEEE Access, vol. 10, no. June, pp. 83249–83264, 2022.
[18] S. A. Butt, S. Misra, G. Piñeres-Espitia, P. Ariza-Colpas, and M. M. Sharma, “A Cost Estimating Method for Agile Software Development,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2021, vol. 12955 LNCS, pp. 231–245.