BAB II TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Penelitian ini didasari dari beberapa penelitian terdahulu yang dijabarkan melalui Tabel 2.1 sebagai berikut

Tabel 2.1 Penelitian terdahulu

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
1.	Implementasi Metode COCOMO II untuk Estimasi Biaya Pengembangan Perangkat Lunak di CV. Profile Image Studio[11]	Memperkirakan sumber daya pengembangan perangkat lunak menggunakan metode COCOMO II, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak pada 2 aplikasi sedangkan penelitian yang akan dilakukan hanya 1 aplikasi saja.	Urutan dalam implementasi metode COCOMO II yang dilakukan tidak urut seperti pada bagan yang terlampir dan ada tahapan yang tidak dilampirkan	Penelitian ini mengimplementasikan metode COCOMO II untuk memperkirakan sumber daya pengembangan perangkat lunak sama seperti (Amru dkk,2019) agar selaras dengan tujuan penelitian untuk menghasilkan estimasi sumber daya pada	Selisih perhitunggan menggunakan COCOMO II dengan alokasi biaya yang dianggarkan pada CV. Profile Image Studio, untuk biaya 37,24%,waktu 52,94% dan SDM 40%

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
					pengembangan perangkat lunak.	
2.	Analisis Estimasi Biaya Pembuatan Perangkat Lunak Menggunakan Metode COCOMO II di Inagata Technosmith (Studi Kasus : Sistem Informasi Monitoring dan Evaluasi Penerimaan Beasiswa Santri Berprestasi UIN Malang)[14]	Memperkirakan sumber daya pengembangan perangkat lunak menggunakan metode COCOMO II, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak pada Sistem Informasi Monitoring dan Evaluasi Penerimaan Beasiswa Santri Berprestasi UIN Malang sedangkan penelitian yang akan dilakukan pada aplikasi	Gambar yang dilampirkan dan beberapa tabel yang dilampirkan kurang jelas sehingga susah untuk dibaca.	Penelitian ini mengimplementasikan metode COCOMO II untuk memperkirakan sumber daya pengembangan perangkat lunak sama seperti (Della dkk,2017) agar selaras dengan tujuan penelitian untuk menghasilkan estimasi sumber daya pada pengembangan perangkat lunak.	Hasil analisis estimasi biaya menggunakan metode COCOMO II menunjukkan bahwa waktu yang diperlukan untuk menyelesaikan Sistem Informasi Monitoring dan Evaluasi Penerimaan Beasiswa Santri Berprestasi UIN Malang lebih lama dibandingkan

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
			pemilihan ketua OSIS.			dengan metode Perkiraan yang digunakan oleh Inagata Technosmith
3.	Evaluasi Biaya Pengembangan Perangkat Lunak Dengan Menggunakan Metode Cocomo II (Studi Kasus: PT DOT Indonesia)[15]	Memperkirakan sumber daya pengembangan perangkat lunak menggunakan metode COCOMO II, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak pada 3 aplikasi sedangkan penelitian yang akan dilakukan hanya 1 aplikasi saja.	Gambar yang dilampirkan dan beberapa tabel yang dilampirkan kurang jelas sehingga susah untuk dibaca.	Penelitian ini mengimplementasikan metode COCOMO II untuk memperkirakan sumber daya pengembangan perangkat lunak sama seperti (Megha dkk,2018) agar selaras dengan tujuan penelitian untuk menghasilkan estimasi sumber daya pada	Selisih estimasi biaya,waktu dan SDM pada PT.DOT Indonesia dengan menggunakan metode COCOMO II memiliki kesamaan skala proyek dengan perangkat A,B dan C

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
					pengembangan perangkat lunak.	
4.	Estimasi Usaha Pengembangan Perangkat Lunak Aplikasi Cerdas Desaku Menggunakan Metode *Use Case Points* dan COSMIC[16]	Memperkirakan sumber daya pengembangan perangkat lunak pada suatu aplikasi, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak menggunakan 2 metode sedangkan penelitian menggunakan metode COCOMO II	Hanya memperkirakan usaha pengembangan perangkat lunak pada aplikasi cerdas desaku	Penelitian ini mengestimasikan usaha pengembangan perangkat lunak dari suatu aplikasi sama seperti penelitian (Amelia dkk, 2021) agar selaras dengan penelitian estimasi usaha pengembangan perangkat lunak dari suatu aplikasi.	Hasil perhitungan estimasi diperoleh menggunakan metode *use case points* yaitu 637,300 *man/hour* dan metode COSMIC diperoleh hasil 4.765,92 *man/hour*
5.	Estimasi Biaya Proyek Perangkat Lunak Menggunakan *Use*	Memperkirakan sumber daya pengembangan perangkat lunak	Memperkirakan sumber daya perangkat lunak menggunakan	Bagan yang dilampirkan susah dibaca. Alur dan proses	Penelitian ini mengestimasikan usaha pengembangan perangkat lunak dari	Penggunaan metode *use-case based effort estimation* lebih

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
	Case-Based Effort Estimation[17]	pada suatu aplikasi, sama seperti penelitian yang dilakukan.	metode *Use Case-Based Effort Estimation* sedangkan penelitian menggunakan metode COCOMO II	perhitungan metode tidak dijelaskan secara detail.	suatu aplikasi sama seperti penelitian (W.J.N Putra dan A.R Perdanakusuma, 2020) agar selaras dengan penelitian estimasi usaha pengembangan perangkat lunak dari suatu aplikasi.	akurat dibandingkan dengan metode *non-algoritmic*
6.	Effort Estimation Menggunakan Metode *Use Case Point* untuk Pengembangan Perangkat Lunak (Studi Kasus Sistem Inventory	Memperkirakan sumber daya pengembangan perangkat lunak pada suatu aplikasi, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak menggunakan metode *Use Case Point* sedangkan penelitian	Tidak dituliskan permasalahan yang mendasar dari penelitian yang akan dilakukan malah menjelaskan	Penelitian ini mengestimasikan usaha pengembangan perangkat lunak dari suatu aplikasi sama seperti penelitian (R. Adhitama dan C. Kartiko, 2018) agar	Hasil perhitungan nilai EF adalah 20 jam untuk setiap UCP dimana resiko proyek pengerjaan tergolong rendah dan *effort*

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
	Peminjaman Alat Laboratorium) [18]		menggunakan metode COCOMO II	adanya sebuah aplikasi yang akan diteliti saja.	selaras dengan penelitian estimasi usaha pengembangan perangkat lunak dari suatu aplikasi.	*estimation* dari pengerjaan adalah 1340 jam pengerjaan.
7.	Pengembangan Aplikasi Perhitungan Estimasi Biaya Proyek Sistem Informasi Menggunakan Metode *Use Case Point, Extended Use Case Point,* dan COCOMO 2[19]	Memperkirakan sumber daya pengembangan perangkat lunak pada suatu aplikasi, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak menggunakan 3 metode sedangkan penelitian menggunakan metode COCOMO II	Tidak dituliskan permasalahan yang mendasar dari penelitian yang akan dilakukan malah menjelaskan metode-metode yang akan digunakan dalam penelitian.	Penelitian ini mengestimasikan usaha pengembangan perangkat lunak dari suatu aplikasi sama seperti penelitian (Ananta dkk, 2019) agar selaras dengan penelitian estimasi usaha pengembangan perangkat lunak dari suatu aplikasi.	Hasil pengujian *User Acceptance Testing* didapatkan nilai penerimaan sebesar 84% yang berarti sistem diterima oleh pengguna

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
8.	*Software Engineering Cost Estimation using COCOMO II Model*[20]	Memperkirakan sumber daya pengembangan perangkat lunak menggunakan metode COCOMO II, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak pada suatu aplikasi di perusahaan sedangkan penelitian yang dilakukan pada aplikasi pemilihan ketua OSIS.	Tidak dituliskan permasalahan yang mendasar dari penelitian yang akan dilakukan	Penelitian ini mengestimasikan usaha pengembangan perangkat lunak dari suatu aplikasi sama seperti penelitian (Khin dkk, 2019) agar selaras dengan penelitian estimasi usaha pengembangan perangkat lunak dari suatu aplikasi.	Hasil dari perhitungan dikatakan akurat dan dapat membantu pengembilan keputusan manajemen yang baik dalam menentukan waktu,biaya dan sumber daya manusia.
9.	*Project Productivity Evaluation in Early Software*	Memperkirakan sumber daya pengembangan perangkat lunak pada suatu	Memperkirakan sumber daya perangkat lunak menggunakan metode *Use*	Tidak dituliskan permasalahan yang mendasar dari penelitian	Penelitian ini mengestimasikan usaha pengembangan perangkat lunak dari suatu aplikasi sama	Hasil penelitian ini yaitu menggunakan variabel ukuran *Use Case Point*

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
	Effort Estimation[21]	aplikasi, sama seperti penelitian yang dilakukan.	*Case Point* sedangkan penelitian menggunakan metode COCOMO II	yang akan dilakukan	seperti penelitian (M. Azzeh dan A. B. Nassif, 2018) agar selaras dengan penelitian estimasi usaha pengembangan perangkat lunak dari suatu aplikasi.	untuk memprediksi upaya lebih akurat daripada menggunakan produktivitas
10	*Estimation of Software Development Effort: A Differential Evolution Approach*[22]	Memperkirakan sumber daya pengembangan perangkat lunak pada suatu aplikasi, sama seperti penelitian yang dilakukan.	Memperkirakan sumber daya perangkat lunak menggunakan 2 metode sedangkan penelitian menggunakan metode COCOMO II	Urutan langkah penggunaan metode kurang dijelaskan dengan detail.	Penelitian ini mengestimasikan usaha pengembangan perangkat lunak dari suatu aplikasi sama seperti penelitian (Prerna dkk, 2020) agar selaras dengan penelitian estimasi usaha pengembangan	Hasil perhitungan menyatakan algoritma DE lebih akurat dibandingkan dengan metode lainnya.

No	Judul	*Comparing*	*Contrasting*	*Criticize*	*Synthesize*	*Summarize*
					perangkat lunak dari suatu aplikasi.	

Berdasarkan kesepuluh literatur maka penelitian ini akan mengambil metode COCOMO II karena pada literatur 11 terdapat kelebihan dari metode COCOMO II yaitu dapat menilai suatu karakteristik proyek yang dikerjakan, pada literatur 14 menyarankan metode COCOMO II dapat digunakan pada objek lainnya dan pada literatur 20 metode COCOMO II memiliki kelebihan yaitu memudahkan dalam memprediksi usaha proyek IT yaitu biaya,waktu dan sumber daya manusia.

2.2 Dasar Teori

Dasar teori menjelaskan mengenai landasan teori berserta komponen pendukung dalam penelitian estimasi sumber daya pengembangan sistem informasi PEMILOS menggunakan COCOMO II.

2.2.1 Estimasi Sumber daya Proyek IT

Estimasi sumber daya proyek IT merupakan suatu kegiatan pengelolaan sumber daya dalam mencapai tujuan dan sasaran dari proyek IT, agar proyek tersebut dapat berjalan sesuai dengan tahapan dan sasaran yang diinginkan[10].

Estimasi sumber daya proyek IT adalah proses dimana memperkirakan biaya yang digunakan saat melakukan pengembangan perangkat lunak, yang diprediksi yaitu jadwal,SDM dan biaya dikeluarkan dalam penyelesaian proyek pengembangan *software*[11].

2.2.2 Aplikasi Pemilihan Ketua OSIS

Aplikasi Pemilihan Ketua OSIS merupakan suatu bentuk pembelajaran pemilihan ketua OSIS yang dikemas melalui *website* seperti pada pemilihan legislatif ataupun pilpres dan pilkada[23]. Sikap demokrasi yang tertanam sejak awal akan berdampak positif kepada siswa dalam kehidupannya pada masa yang akan datang, karena para siswa nantinya menjadi penerus bangsa dan dapat memberikan hak politiknya dengan penuh[7].

Pemilihan Ketua OSIS merupakan cerminan pemilihan umum, melalui PEMILOS siswa dan siswi SMA/SMK dapat mengerti pentingnya pemilihan umum, mengerti tahapan pemilihan umum,mengetahui bagaimana menggunakan hak pilih, dan mengetahui tata cara pemungutan suara dalam pemilu[6].

2.2.3 COCOMO II

COCOMO II atau *Constructive Cost Model* II merupakan suatu pemodelan algoritmik yang dapat membantu dalam menghitung estimasi biaya,usaha dan waktu dalam proyek pengembangan perangkat lunak, COCOMO II hasil pengembangan dari COCOMO 81 yang dikembangkan oleh Boehm pada 1981 dan dipublikan pada tahun 1997[11].

COCOMO II memberikan dukungan terkini untuk perangkat lunak perusahaan, perangkat lunak berorientasi objek, perangkat lunak yang baru dikembangkan, dan perangkat lunak komersial. COCOMO II dapat digunakan tidak hanya pada tahap awal desain proyek, tetapi juga pada proyek yang menggunakan arsitektur yang sudah ada[12].

Tahap dalam mengimplementasikan metode COCOMO II dijabarkan sebagai berikut[12] :

1. Analisis *Data Flow Diagram*

   Pada tahap analisis DFD digunakan untuk menggambarkan kompleksitas dari sistem perangkat lunak sehingga dapat mempermudah dalam menentukan nilai *Unadjasted Function Point* (UFP).

2. Menghitung *Unadjusted Function Point*

   Tahap ini digunakan untuk mendapatkan nilai UFP yang berasal dari DFD berdasarkan 5 komponen function point yaitu *External Input* (EI),*External Output* (EO),*External Inquiry* (EQ),*Internal Logical FIle* (ILF),*External Interface File* (EIF). Setelah 5 komponen *function point* dianalisis maka selanjutnya dihitung nilai UFP berdasarkan *Data Element Type* (DET),*Record Element Type* (RET), dan *Files Type References* (FTR) lalu dikalikan nilai kompleksitas dari perangkat lunak.

3. Menghitung *size*

   Menghitung nilai *size* sistem perangkat lunak yaitu dengan cara menghitung *Source Line Of Code* (SLOC) yang dikonversikan ke dalam *Kilo Source Line Of Code* (KSLOC). Langkah dalam tahap ini yaitu :

   1) Masukkan nilai total UFP sistem perangkat lunak

2) Kemudian kalikan dengan standar konversi SLOC sesuaikan dengan bahasa pemrograman yang digunakan dalam pembuatan perangkat lunak yang hasilnya berupa SLOC

3) Lalu bagi nilai SLOC dengan 1000 agar menjadi nilai KSLOC

4. Menghitung *Scale Factor*

   Menentukan nilai *Scale Factor* berasal dari parameter yang telah dikumpulkan dari kuesioner yang sebelumnya diisi oleh pengembang sistem yang terdiri dari 5 atribut yaitu *Precendentness* (PREC),*Development Flexibility* (FLEX),*Risk Resolution* (RESL),*Team Cohesian* (TEAM),*Process Maturity* (PMAT).

   Setelah mendapatkan nilai dari setiap 5 atribut dari lembar kuesioner kemudian tambahkan setiap skala faktor sehingga menghasilkan total penilaian skala faktor yang nantinya digunakan untuk menghitung faktor eksponen, sehingga nilai dari faktor eksponen nanti dapat digunakan dalam persamaan *effort estimation*.

5. Menghitung *Effort Multiplier*

   Pada tahap ini nilai *Effort Multipliers* didapatkan dari 17 parameter dari effort multipliers melalui kuesioner yang diisi oleh pengembang sistem, kemudian menghasilkan nilai rerata penilaian *effort multiplier* yang nantinya dimasukkan ke dalam perhitungan *effort estimation*.

6. Menghitung *Effort Estimation*

   Pada tahap ini akan dihitung nilai estimasi proyek pengembangan sistem informasi berupa estimasi waktu,biaya dan sumber daya manusia.

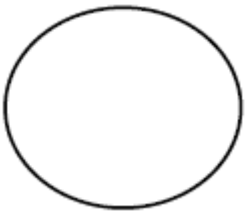
2.2.3.1 *Data Flow Diagram*

Data Flow Diagram (DFD) adalah suatu diagram yang menggunakan notasi-notasi untuk menggambarkan aliran data dari suatu sistem baik aliran data masuk,proses dan keluar guna membantu memahami sistem dengan jelas[24]. Terdapat dua teknik dasar dalam penggambaran simbol DFD yang umum dipakai, yang pertama adalah Gane and Sarson sedangkan yang kedua adalah Yourdan and De Marco[25]. Kegunaan masing-masing notasi pada *Data Dlow Diagram* (DFD) yaitu [25] :

a. *External Entity* (kesatuan luar), digunakan untuk menyatakan suatu departemen atau divisi dalam perusahaan tetapi diluar sistem yang dikembangkan.
b. *Data Flow* (arus data), digunakan untuk menunjukan arus data yang berupa masukan untuk sistem ataupun hasil dari proses sistem.
c. *Process* (proses), digunakan untuk menunjukan kegiatan atau kerja yang dilakukan oleh orang, mesin atau komputer dari hasil arus data yang masuk kedalam proses.
d. *Data Store* (simpanan data), digunakan untuk menunjukan simpanan dari data yang dapat berupa suatu file atau database.

Penggambaran notasi pada Data Flow Diagram menurut Yourdan and De Marco dijabarkan pada Tabel 2.2 sebagai berikut[24] :

Tabel 2.2 Notasi DFD[24]

Keterangan	Notasi
External Entity	
Data Flow	
Process	
Data Store	

2.2.3.2 *Unadjusted Function Point* (UFP)

Unadjusted Function Point (UFP) merupakan model pengukuran sekaligus satuan ukuran perangkat lunak dengan cara mengkuantifikasi fungsionalitas

perangkat lunak yang disediakan untuk pengguna berdasarkan pada desain logik[8]. *Function point* digunakan untuk mengatasi masalah yang berhubungan dengan baris kode sebagai ukuran suatu perangkat lunak dan membantu dalam mengembangkan mekanisme estimasi usaha yang terkait dengan pengembangan perangkat lunak. Setiap komponen *Function point* diklasifikasikan tingkat kompleksitasnya, berikut komponen *function point* yang dijabarkan pada Tabel 2.3[8].

Tabel 2.3 Komponen *Function Point*[8]

Komponen	Keterangan
External Input (EI)	*Input* berasal dari luar sistem, baik dari user maupun sistem lainnya yang selanjutnya digunakan untuk mengupdate *Internal Logical Files.*
External Output (EO)	*Output* merupakan data yang ditampilkan pada aplikasi untuk menyediakan Informasi kepada user, baik dalam bentuk laporan, tampilan di layar, pesan error, dst.
External Inquary (EQ)	Inquiries eksternal didefinisikan sebagai input online yang memicu respon dari *software* untuk menghasilkan *output* online.
Internal Logical File (ILF)	File logika internal merupakan data yang dikelompokkan secara logis, disimpan secara internal dan didapat dari input eksternal.
External Interface File (EIF)	Data yang dikelompokkan secara logis namun berada diluar aplikasi yang menyediakan Informasi yang dibutuhkan aplikasi.

Setelah melakukan analisis komponen *function point*, setiap kompnen memiliki nilai perhitungan yang nantinya akan dikalikan dengan bobot kompleksitas *function*. Berikut bobot komponen *function* yang dijabarkan pada Tabel 2.4[26].

Tabel 2.4 Bobot Kompleksitas *Function Poin*t[26]

Tipe fungsi	Bobot Kompleksitas		
	Low	*Average*	*High*
Internal Logical File	7	10	15
External Interface File	5	7	10
External Input	3	4	6
External Output	4	5	7
External Inquary	3	4	6

Bobot kompleksitas ditentukan berdasarkan pada jumlah tipe elemen data tiap fungsi dan jumlah referensi tipe *file*. Ada 3 kategori dalam menentukan bobot kompleksitas yaitu DET,RET dan FTR, Tabel 2.5 menjelaskan DET,RET dan FTR[8].

Tabel 2.5 DET,RET,dan FTR[8]

Kategori	Keterangan
Data Element Type (DET)	*Field* yang tak berulang dan diidentifikasi *user* sebagai *field* yang unik
Record Element Type (RET)	Subgroup dari *element type* yang berada dalam *Internal Logical File (ILF)* atau *Eksternal Interface File (EIF)*
File Type Reference (FTR)	Sebuah jenis *file* yang dibaca oleh fungsi transaksional. Fungsi transaksional merepresentasikan fungsionalitas yang disediakan user untuk melakukan pemrosesan data menggunakan aplikasi

Setelah menganalisis DET,RET dan FTR, selanjutnya menghitung nilai DET,RET, dan FTR tersebut kedalam tingkat bobot kompleksitas UFP yang dijabarkan pada Tabel 2.6[26].

Tabel 2.6 Bobot kompleksitas UFP[26]

Untuk *Internal Logical Files* dan *External Interface Files*			
	Data Element Type		
Record Element Type	1-19	20-50	51+
1	*Low*	*Low*	*Avg.*
2-5	*Low*	*Avg*	*High*
6+	*Avg.*	*High*	*High*
Untuk *External Output* dan *External Inquiry*			
	Data Element Type		
File Type Reference	1-15	6-19	20+
0 atau 1	*Low*	*Low*	*Avg.*
2-3	*Low*	*Avg.*	*High*
4+	*Avg.*	*High*	*High*
Untuk *External Input*			
	Data Element Type		
File Type Reference	1-4	5-15	16+
0 atau 1	*Low*	*Low*	*Avg.*
2-3	*Low*	*Avg.*	*High*
3+	*Avg.*	*High*	*High*

2.2.3.3 Menghitung *Size*

Menghitung nilai *size* perangkat lunak yaitu dengan cara menghitung *Source Line Of Code* (SLOC) yang dikonversikan ke dalam *Kilo Source Line Of Code* (KSLOC)[8]. Total UFP dikonversikan ke dalam SLOC dengan mengalikan bahasa pemrograman yang digunakan sesuaikan QSM function point language table yang kemudian dibagi 1000 untuk menghasilkan KSLOC[24]. Tabel 2.7 menjabarkan bahasa pemrograman yang ada berdasarkan pada *Quantitative Software Management* (QSM) *Function Point Language Table*[8].

Tabel 2.7 *QSM Function Point Language Table*[8]

Bahasa	QSM SLOC/FP Data			
	Average	***Median***	***Low***	***High***
ABAP (SAP) *	28	18	16	60
ASP *	51	54	15	69
Assembler *	119	98	25	320
C *	97	99	39	333
C# *	50	53	25	80
HTML *	34	40	14	48
Java *	53	53	14	134
JavaScript *	47	53	31	63
SQL *	21	21	13	37
Visual Basic *	42	44	20	60

Angka yang digunakan untuk perhitungan yaitu median, dikarenakan level data tersebut merupakan indikator yang dinilai paling akurat dibandingkan dengan level indikator yang lain[8].

2.2.3.4 *Scale Faktor*

Scale Factor merupakan cara yang digunakan untuk menentukan usaha proyek dan karakteristik dari suatu proyek. Menentukan nilai *scale factor* mengguankan 5 parameter penilian setiap bobot yang ada pada *scale factor*, nilai *scale factor* diperoleh melalui kuesioner yang sebelumnya telah diisi oleh pengembang perangkat lunak[8]. Tabel 2.8 menjelaskan 5 parameter *scale factor* sebagai berikut[8]:

Tabel 2.8 Parameter *Scale Factor*[8]

Parameter *Scale Factor*	Deskripsi
Precedentness (PREC)	Faktor skala ini merupakan faktor skala yang menggambarkan pengalaman masa lalu organisasi dengan proyek-proyek sejenis.

Parameter *Scale Factor*	**Deskripsi**
Flexibility (FLEX)	Faktor skala ini merupakan faktor skala yang menggambarkan kemampuan klien dalam menentukan tujuan dan mengkomunikasikan kebutuhan perangkat lunak kepada tim pengembang.
Risk Resolution (RESL)	Faktor skala ini merupakan faktor skala yang menggambarkan seberapa sering perusahaan merencanakan manajemen resiko terhadap proyek yang dijalankan oleh perusahaan. Perencanaan manajemen resiko ini berguna untuk mengidentifikasi dan mencegah semua resiko yang mungkin terjadi.
Team Cohesion (TEAM)	Faktor skala ini merupakan faktor skala yang menggambarkan seberapa baik tim pengembangan bekerja sama.
Process Maturity (PMAT)	Faktor skala yang menggambarkan kematangan proses pengembangan perangkat lunak dalam organisasi. Hal ini didasarkan pada Model Kematangan Kemampuan Rekayasa Perangkat Lunak atau *Capability Maturity Model* (CMM).

Rumus persamaan *Scale Factor* yang digunakan untuk mencari nilai dari *Scale Factor* sebagai berikut[8] :

$$E = B + 0.01 \times SF \quad (2.1)$$

Dimana :

E = faktor eskponen

B = nilai koefisien 0.91 (untuk COCOMO II.2000)

SF = total nilai *Scale Factor*

Selanjutnya bobot penilaian dari setiap faktor skala yang telah didapatkan nantinya akan dihitung dan digunakan dalam perhitungan *effort estimation*. Nilai dari bobot *scale factor* dijelaskan pada Tabel 2.9[26].

Tabel 2.9 Bobot *Scale Factor*[26]

Parameter *Scale Factor*	***Very Low***	***Low***	***Nominal***	***High***	***Very High***	***Extra High***
Precedentness (PREC)	6.20	4.95	3.72	2.48	1.24	0.00
Flexibility (FLEX)	5.07	4.05	3.04	2.03	1.01	0.00
Risk Resolution (RESL)	7.07	5.65	4.24	2.83	1.41	0.00
Team Cohesion (TEAM)	5.38	4.38	3.29	2.19	1.10	0.00
Process Maturity (PMAT)	7.80	6.24	4.68	3.12	1.56	0.00

2.2.3.5 *Effort Multiplier*

Effort multiplier merupakan faktor penggerak biaya dalam menentukan usaha yang diperlukan dalam pengerjaan proyek perangkat lunak[11]. *Effort multiplier* terdapat 7 kategori dan 17 atribut pada submodel *post-architecture* model COCOMO II, nilai *effort multiplier* diperoleh melalui kuesioner yang sebelumnya telah diisi oleh pengembang perangkat lunak[8]. Tabel 2.10 menjelaskan deskripsi *effrort multiplier*[8]

Tabel 2.7 Deskripsi *Effrort Multiplier*[8]

NO	**Kategori**	**Atribut**
1.	*RCPX (Product Reliability and Complexity)*	Penilaian *cost driver* terkait beberapa faktor yaitu : 1. Sejauh mana perangkat lunak menjalankan aplikasi sesuai fungsinya selama periode waktu *(RELY).* 2. Ukuran database yang digunakan. Ukuran dapat dihitung menggunakan D/P *(DATA).* 3. Perangkat lunak dan perangkat keras dalam melakukan tugasnya seperti platform (arsitektur, sistem operasi, bahasa pemrograman dan antar muka yang terkait), sistem manajemen database,

NO	Kategori	Atribut
		browser yang sesuai digunakan dalam menjalankan aplikasi ini *(CPLX).* 4. Kesesuaian dokumentasi proyek terhadap kebutuhan siklus hidup perangkat lunak *(DOCU).*
2.	*PERS (Personel Capability)*	Penilaian *cost driver* terkait beberapa kemampuan personel yaitu : 5. Kemampuan personel dalam analisis dan desain, efisiensi dan ketelitian, serta kemampuan untuk berkomunikasi dan bekerja sama. Dalam hal ini, dapat dinilai dari sertifikasi yang sudah didapatkan personel atau pengalaman kerja tim dalam suatu proyek *(ACAP).* 6. Kemampuan programmer dalam efisiensi penulisan kode program, ketelitian dan kemampuan untuk berkomunikasi dan bekerja sama sebagai sebuah tim. Dengan kata lain, berapa banyak proyek dimana programmer tersebut terlibat *(PCAP).* 7. Pergantian personel tiap tahun pada proyek. Semakin sedikit pergantian maka semakin tinggi skala *(PCON).*
3.	*RUSE (Developed for Reusability)*	8. *Ruse* merupakan *cost driver* terkait tingkat upaya yang diperlukan untuk mengembangkan komponen yang dimaksudkan untuk digunakan kembali pada proyek-proyek yang sedang berjalan atau proyek di masa mendatang.
4.	*PDIF (Platform Difficulty)*	Penilaian *cost driver* terkait beberapa kendala dalam pengembangan yaitu :

NO	Kategori	Atribut
		9. Persentase kendala waktu eksekusi yang diharapkan dapat digunakan pada sistem perangkat lunak *(TIME).* 10. Persentase tingkat kendala penyimpanan utama yang dikenakan pada sistem perangkat lunak *(STOR).* 11. Perubahan yang terjadi pada *hardware* dan *software* dalam kurun waktu tertentu *(PVOL).*
5.	*PREX (Personel Experience)*	Penilaian *cost driver* terkait beberapa pengalaman yaitu : 12. Pengelaman kerja tim proyek pada suatu proyek pengembangan aplikasi sistem perangkat lunak atau subsistem *(APLEX).* 13. Pemahaman tim proyek dalam menggunakan *platform, interface database,* jaringan, *middleware (PLEX).* 14. Pengelaman tim proyek dalam pemrograman dengan bahasa tertentu dan pemanfaatan *CASE tool* dalam mengembangkan perangkat lunak *(LTEX).*
6.	*FCIL (Facilities)*	15. *TOOL* merupakan penilaian cost driver terkait penggunaan *CASE tool* dalam pengembangan perangkat lunak pada proyek, seperti dari mengubah kode yang sederhana menjadi terintegrasi. 16. *SITE* adalah bagaimana cara komunikasi yang digunakan dalam pengembangan perangkat lunak pada proyek.

NO	Kategori	Atribut
7.	*SCED (Required Development Schedule)*	17. *SCED* merupakan penilaian *cost driver* terkait tingkat persentase dari percepatan atau kemunduran jadwal terhadap jadwal suatu proyek yang telah ditetapkan sebelumnya.

COCOMO II submodel *Post Architecture* terdapat nilai tetap untuk masing-masing bobot. Tabel 2.11 menjelaskan nilai *effort multipliers*[26].

Tabel 2.8 Nilai *Effort multipliers*[26]

Scale Factor	***Very Low***	***Low***	***Nominal***	***High***	***Very High***	***Extra High***
RELY	*slight inconveni-ence*	*low, easily recoverable losses*	*moderate ,easily recovera ble losses*	*high financial loss*	*risk to human life*	
	0.82	0.92	1.00	1.10	1.26	
DATA		*Testing* DB *bytes/*Pgm SLOC < 10	10 ≤ D/P < 100	100 ≤ D/P < 1000	D/P ≥ 1000	
		0.90	1.00	1.14	1.28	
CPLX	*software and hardware not working properly*	*there was some error while running the software*	*the software runs as it should*	*software runs very well no error at all*	*the software runs and when testing there are no errors*	
	0.73	0.87	1.00	1.17	1.34	1.74

Scale Factor	***Very Low***	***Low***	***Nominal***	***High***	***Very High***	***Extra High***
RUSE		*none*	*across project*	*across program*	*across product line*	*across multiple product lines*
		0.95	1.00	1.07	1.15	1.24
DOCU	*Many lifecycle needs uncovered*	*Some lifecycle needs uncovered*	*Right-sized to life-cycle needs*	*Excessive for life-cycle needs*	*Very excessive for life-cycle needs*	
	0.81	0.91	1.00	1.11	1.23	
TIME			≤ 50% *use of available execution time*	70% *use of available execution time*	85% *use of available execution time*	95% *use of available execution time*
			1.00	1.11	1.29	1.63
STOR			≤ 50% *use of available storage*	70% *use of available storage*	85% *use of available storage*	95% *use of available storage*
			1.00	1.05	1.17	1.46

Scale Factor	***Very Low***	***Low***	***Nominal***	***High***	***Very High***	***Extra High***
PVOL		*Major change every 12 mo.; Minor change every 1 mo.*	*Major: 6 mo.; Minor: 2 wk.*	*Major: 2 mo.;Minor: 1 wk*	*Major: 2 wk.;Minor: 2 days*	
		0.87	1.00	1.15	1.30	
ACAP		35th *percentile*	55th *percentile*	75th *percentile*	90th *percentile*	
		1.42	1.00	0.85	0.71	
PCAP		35th *percentile*	55th *percentile*	75th *percentile*	90th *percentile*	
		1.34	1.00	0.88	0.76	
PCON	48% / *year*	24% / *year*	12% / *year*	6% / *year*	3% / *year*	
	1.29	1.12	1.00	0.90	0.81	
AEXP	≤ 2 *months*	6 *months*	1 *year*	3 *years*	6 *year*	
	1.22	1.10	1.00	0.88	0.81	
PEXP	≤ 2 *months*	6 *months*	1 *year*	3 *years*	6 *year*	
	1.19	1.09	1.00	0.91	0.85	

Scale Factor	***Very Low***	***Low***	***Nominal***	***High***	***Very High***	***Extra High***
LTEX	≤ 2 *months*	6 *months*	1 *year*	3 *years*	6 *year*	
	1.20	1.09	1.00	0.91	0.84	
TOOL	*edit, code,debug*	*simple, frontend, backend CASE, little integration*	*basic lifecycle tools, moderately integrated*	*strong, mature lifecycle tools, moderately integrated*	*strong, mature, proactive life-cycle tools, well integrated with processes, methods, reuse*	
	1.17	1.09	1.00	0.90	0.78	
SITE	*Somephone, mail*	*Individual phone, FAX*	*Narrow band email*	*Wideband electronic communication.*	*Wideband delect.comm., occasional video conf*	
	1.22	1.09	1.00	0.93	0.86	0.80
SCED	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
	1.43	1.14	1.00	1.00	1.00	

2.2.3.6 *Effort Estimation*

Effort estimation dinyatakan dalam *Person-Month* (PM), satu *Person-Month* merupakan jumlah waktu yang harus dihabiskan untuk mengerjakan proyek pengembangan sistem perangkat lunak selama satu bulan. Perhitungan estimasi usaha (*Person-Month*) submodel *post architrcture* menggunakan persamaan 2.2 sebagai berikut[12] :

$$PM = A \times (Size)^{e} \times \Pi_{i=1}^{i17} EMi \quad (2.2)$$

Dimana :

PM = *person-month*

A = nilai koefisien 2.94 (Untuk COCOMO II,2000)

Size = nilai KSLOC

e = nilai faktor eksponen

EM = *effort multiplier*

2.2.3.7 Estimasi Waktu,Sumber Daya Manusia dan Biaya

COCOMO II mempunyai kemampuan untuk memperkirakan waktu yang hampir sama dengan COCOMO 81. Persamaan ini digunakan untuk *Early-Design* dan juga *Post-Architecture* COCOMO II, berikut adalah persamaannya[8].

$$TDEV = C \times (PM)^{(D+0{,}2 \times (E-B))} \quad (2.3)$$

Dimana:

TDEV = *Time to Development*/jumlah waktu pengembangan

C = nilai koefisien 3,67 (untuk COCOMO II,2000)

D = nilai koefisien 0,28 (untuk COCOMO II,2000)

B = nilai koefisien 0,91 (untuk COCOMO II,2000)

E = nilai faktor eksponen

PM = *person per-month*

Selanjutnya menghitung jumlah sumber daya manusia menggunakan rumus berikut[8] :

$$Average\ staff = PM/TDEV \quad (2.4)$$

Dimana :

Average staff = jumlah pegawai dibutuhkan

PM = *person per month*

TDEV = *Time to Development*/jumlah waktu pengembangan

Selanjutnya menghitung biaya perbulan menggunakan persamaan sebagai berikut[8] :

$$Biaya\ per\ bulan = Average\ staff\ \times UMR \quad (2.5)$$

Dimana :

Biaya per bulan = biaya yang dikeluarkan tiap bulan

Average staff = jumlah pegawai dibutuhkan

UMR = Upah minimum regional daerah

Setelah diketahui nilai biaya perbulan, selanjutnya menghitung biaya total menggunakan persamaan sebagai berikut[8] :

$$Biaya\ total = biaya\ per\ bulan\ \times TDEV \quad (2.6)$$

Dimana:

Biaya total = biaya total proyek

Biaya perbulan = biaya yang dikeluarkan tiap bulan

TDEV = *Time to Development*/jumlah waktu pengembangan