BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Sebelumnya

Berdasarkan hasil dari beberapa penelitian yang sudah pernah dilakukan, metode COCOMO II sangat membantu dalam melakukan analisa estimasi perangkat lunak.

Penelitian tentang aspek atribut *analysis capability* sebagai *cost driver* pada model COCOMO II dan dikombinasikan dengan model *People-CMM* (Palacios 2007) menjelaskan bahwa faktor manusia adalah salah satu relevansi yang terpenting dan aspek yang krusial pada manajemen pengembangan perangkat lunak. Tujuannya adalah perbaikan kinerja untuk menyelesaikan perangkat lunak pada suatu organisasi.

Dalam penelitian mengenai analisis sistem penyelarasan projek pembangunan menggunakan COCOMO II (Ibrahim 2008) menjelaskan bahwa tujuan dari penelitian tersebut adalah untuk melakukan estimasi perangkat lunak dengan menggunakan tingkatan *early design* model COCOMO II. Hasil akhirnya adalah tingkatan *early design* model COCOMO II dapat diterapkan dalam memprediksi biaya (usaha dan jadwal) perangkat lunak.

Penelitian mengenai aspek PMAT (Process Maturity) sebagai *scale factor* COCOMO II (Yahya 2009) untuk mengetahui pengaruhnya terhadap jadwal perangkat lunak. Tujuan penelitian ini adalah untuk mendapatkan nilai baru dari faktor skala yang telah disesuaikan dengan lingkungan lokal mereka. Dengan 40

dataset yang diujikan didapatkan nilai PMAT baru yang diberi nama *Ideal Scale Factor* (ISF PMAT).

Hasil penelitian tersebut menyatakan bahwa ISF-PMAT berhasil mengestimasi jadwal lebih dekat ke jadwal aktual dibandingkan dengan estimasi jadwal yang didapat dari model COCOMO II. ISF-PMAT memberikan perbaikan akurasi model COCOMO II dengan PRED 20%, yaitu 13% untuk CMMI level satu (Lower Half), 12% untuk CMMI level satu (Upper Half), 37% untuk CMMI level dua, 50% untuk CMMI level tiga, dan 38% untuk CMMI level empat.

Penelitian tentang aspek proses kematangan (PMAT) sebagai *scale factor* model COCOMO II (Vishal Sharma 2010) untuk mengetahui pengaruhnya pada estimasi usaha. Hipotesa kerja yang disajikan yaitu bahwa satu set nilai PMAT bar di bawah CMMI akan meningkatkan daya prediksi model COCOMO II dan membuatnya tepat untuk diterapkan dalam organisasi pembangunan perangkat lunak yang mengadopsi CMMI.

Akurasi estimasi biaya pengembangan perangkat lunak sangat penting dalam perencanaan dan penganggaran proyek yang efektif dalam kendali manajemen proyek. salah satu input yang paling penting untuk estimasi biaya perangkat lunak menggunakan COCOMO II adalah proses PMAT. Hasil penelitian tersebut didapatkan nilai PMAT baru (ISF-PMAT) yang lebih mencerminkan pada dampak proses CMMI terhadap usaha pengembangan perangkat lunak. Nilai baru tersebut menghasilkan perbaikan akurasi dalam model COCOMO II dalam hal PRED sebesar 12% untuk CMMI level satu, 13% untuk

CMMI tingkat atas, 37% untuk CMMI level dua, 50% untuk CMMI tingkat tiga, dan 25% untuk CMMI tingkat empat.

Penelitian terhadap model COCOMO II selanjutnya ditunjukkan pada penelitian mengenai pengembangan *cost driver model* COCOMO II dengan memodifikasi nilai atribut *analysis capability* untuk estimasi usaha perangkat lunak (Sri Andayani 2012). Tujuan dari penelitian tersebut adalah untuk memodifikasi nilai atribut *analysis capability* pada aspek *cost driver* COCOMO II serta membandingkan akurasi hasil estimasi usaha pembangunan perangkat lunak yang menggunakan model COCOMO II dengan hasil estimasi usaha pembangunan perangkat lunak yang menggunakan model COCOMO modifikasi serta mengevaluasi performansi hasil COCOMO modifikasi.

Hasil dari penelitian tersebut diperoleh bahwa hasil modifikasi dari nilai atribut ACAP untuk model COCOMO II memberikan prediksi 50% dan akurasi 37%, sedangkan model COCOMO II umum memberikan prediksi 40% dan akurasi 33% terhadap estimasi usaha perangkat lunak. Hasil tersebut menunjukkan peningkatan hasil prediksi sebesar 10% dan akurasi sebesar 4%. Kinerja dari model COCOMO II modifikasi relatif lebih baik dari model COCOMO II.

2.2 Perkiraan Biaya Perangkat Lunak

Perkiraan biaya perangkat lunak adalah proses memprediksi biaya yang diperlukan dalam pengembangan sistem perangkat Lunak (Leung 2001). Perkiraan biaya perangkat lunak mengacu pada prediksi jumlah biaya, waktu, dan

staffing level yang diperlukan untuk membangun sistem perangkat lunak (Saliu 2003).

Perkiraan biaya perangkat lunak dapat diklasifikasikan kedalam metode pemodelan algoritmik dan pemodelan non-algoritmik. Pemodelan algoritmik diturunkan dari analisis statistikal dari data proyek terdahulu (Saliu 2003). Contoh dari pemodelan algoritmik adalah model linear, model multiplikatif, dan model *power-function*. Sedangkan pemodelan non-algoritmik berdasarkan pengalaman pakar dan perbandingan dengan proyek sebelumnya yang dianalogikan (Dickson 2007). Contoh yang paling umum adalah *expert judment* dan *analogy costing*.

2.3 COCOMO II

Model COCOMO II merupakan bagian dari rangkaian *Constructive Cost Model*s. Model ini merupakan upaya untuk memperbarui dan mengembangkan model estimasi biaya yang terkenal, yaitu COCOMO (Constructive Cost Model) yang merupakan suatu model estimasi biaya perangkat lunak yang diterbitkan di *Software Engineering Economics* oleh Barry Boehm pada tahun 1981. Model ini berfokus pada isu-isu seperti model proses non-sekuensial dan *rapid-development*, menggunakan kembali pendekatan dengan paket *commercial-off-the-shelf* (COTS), rekayasa ulang, komposisi aplikasi, dan efek kematangan proses perangkat lunak, serta kualitas estimasi *process-driven*.

Constructive Cost Model II (COCOMO II) adalah salah satu teknik estimasi pemodelan biaya algoritmik yang merupakan pembaharuan dari versi COCOMO 81. COCOMO II memiliki beberapa sub model diantaranya, yaitu :

1. *Application Composition*, digunakan ketika perangkat lunak terdiri dari bagian-bagian yang sudah ada (pembuatan prototipe).
2. *Early design* (desain awal model), digunakan ketika persyaratan tersedia, tetapi jika desain belum ada tidak bisa di mulai.
3. *Post-Architechture design model*, digunakan pada saat proyek siap untuk dikembangkan, proyek tersebut harus memiliki arsitektur siklus hidup yang memberikan informasi yang lebih akurat pada masukan-masukan penggerak biaya (cost drivers) dan memungkinkan estimasi biaya untuk lebih akurat.

Pada tingkat *Post-Architechture* terdapat dua aspek penting, yaitu 5 faktor skalar (scale factor) yang merupakan faktor penentu eksponen yang digunakan dalam *effort equation* (persamaan usaha) dan 17 atribut enggerak biaya (cost driver) yang merupakan faktor pengali yang menentukan usaha yang diperlukan untuk menyelesaikan proyek perangkat lunak.

2.3.1 Effort Estimation

Dalam COCOMO II *effort* dinyatakan sebagai *person months* (PM). Satu *person month* adalah jumlah waktu yang digunakan seseorang untuk mengerjakan proyek pengembangan perangkat lunak selama satu bulan. COCOMO II memproses *person hours per person month* (PH/PM) sebagai faktor yang dapat disesuaikan dengan nilai nominal 152 jam per Person-Month. Angka ini mengecualikan waktu yang dikhususkan untuk liburan, vakansi, dan libur akhir pekan. Angka dari *person-months* berbeda tergantung dari waktu yang dibutuhkan untuk menyelesaikan sebuah proyek; hal ini disebut sebagai jadwal pengembangan atau *Time to Develop*, TDEV. Sebagai contoh, sebuah proyek

mungkin dapat diperkirakan membutuhkan 50 PM *effort* namun memiliki jadwal sebanyak 11 bulan. Jika Anda menggunakan nilai yang berbeda untuk PH/PM – katakanlah, 160 alih-alih 152 – COCOMO II akan menyesuaikannya (pada kasus ini, menguranginya sebesar 5%). PM yang dikurangi ini akan berakhir pada jadwal pengembangan yang estimasinya lebih kecil.

Perhitungan *effort estimation* dalam COCOMO II menggunakan persamaan sebagai berikut :

$$Effort = Ax[Size]^{B} x\prod_{i=1}^{17} EffortMultiplier_i$$

Persamaan 2.1

Dimana :

A : konstanta (pada COCOMO II, A=2,94)

Size : jumlah baris program (dalam KLOC)

B : eksponen faktor skalar

Nilai eksponen faktor skalar didapatkan dari rumus berikut :

$$B = E + 0.01x\sum_{i=1}^{5} ScaleFactor_i$$

Persamaan 2.2

Dimana :

E : eksponen dengan nilai 0,92

2.3.2 Scale Factor

Eksponen E pada Persamaan 2.2 merupakan agregasi dari 5 *scale factor* (SF) yang menghitung ekonomis dan disekonomis relatif dari skala yang

ditemukan untuk proyek perangkat lunak dengan ukuran yang berbeda [Banker et al. 1994]. Jika E<1.0, berarti proyek tersebut menunjukkan skala ekonomis. Jika ukuran sebuah proyek dibuat menjadi dua kali lipat, maka effortnya kurang dari dua kali lipat. Produktivitas proyek ini meningkat seiring dengan meningkatnya ukuran proyek. Skala ekonomis beberapa proyek dapat diperoleh dengan *tools* khusus untuk proyek (mis. simulasi, testbed), tapi pada umumnya hal ini sulit diperoleh. Untuk proyek kecil, biaya *start-up* tetap seperti *tool tailoring* dan *setup* untuk laporan standar dan administratif biasanya menjadi sumber untuk skala ekonomis.

Jika E=1.0, berarti skala ekonomis dan disekonomis seimbang. Model linear ini sering digunakan untuk memperkirakan biaya dari proyek kecil.

Jika E>1.0, berarti proyek tersebut menunjukkan skala disekonomis. Umumnya ini disebabkan oleh 2 faktor utama, membesarnya *overhead* komunikasi interpersonal dan membesarnya *overhead* integrasi dari sistem besar. Proyek yang lebih besar akan memiliki personil yang lebih banyak, dan oleh karena itu lebih banyak arah komunikasi interpersonal yang memakan *overhead*. Mengintegrasikan produk-produk kecil sebagai bagian dari produk yang lebih besar, tidak hanya membutuhkan *effort* untuk mengembangkan produk yang kecil, tapi juga *overhead effort* tambahan untuk mendesain, *maintenance*, mengintegrasi, dan mengetes interface-nya dengan sisa produk yang lain.

Persamaan 2.2 mendefinisikan eksponen E, yang digunakan pada persamaan 2.1. Tabel 2.1 memberikan level rating untuk faktor skala COCOMO II. Pemilihan faktor skala didasarkan pada pemikiran bahwa mereka adalah sumber

yang signifikan untuk variasi eksponensial pada *effort* sebuah proyek atau variasi produktivitas. Setiap faktor skala memiliki sebuah ruang lingkup untuk level rating, dari Sangat Rendah sampai Sangat Tinggi. Setiap level rating memiliki berat. Nilai khusus untuk berat ini disebut sebagai sebuah *scale factor* (SF).

Tabel 2.1. Nilai *Scale Factor*, SF_i, untuk Model COCOMO II

Scale Factors	**Very Low**	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
PREC SF_i	Thoroughly unpreceden-ted 6.20	Largely unpreceden-ted 4.96	Somewhat unprecedent-ed 3.72	Generally Familiar 2.48	Largely familiar 1.24	Thoroughly familiar 0.0
FLEX SF_i	Rigorous 5.07	Occasional relaxation 4.05	Some relaxation 3.04	General conformity 2.03	Some conformi-ty 1.01	General goals 0.00
RESL SF_i	Little (20%) 7.07	Some (40%) 5.65	Often (60%) 4.24	Generally (75%) 2.83	Mostly (90%) 1.41	Full (100%) 0.00
TEAM SF_i	Very difficult interactions 5.48	Some difficult interactions 4.38	Basically cooperative interactions 3.29	Largely cooperative 2.19	Highly cooperat-ive 1.10	Seamless interactions 0.00
PMAT SF_i	The Estimated Equivalent Process Maturity Level (EPML) or					
	SW-CMM Level 1 Lower 7.80	SW-CMM Level 1 Upper 6.24	SW-CMM Level 2 4.68	SW-CMM Level 3 3.12	SW-CMM Level 4 1.56	SW-CMM Level 5 0.00

Dua faktor skala, *Precedentedness* dan *Flexibility* menangkap perbedaan yang besar antara mode *Organic*, *Semidetached*, dan *Embedded* dari model COCOMO asli [Boehm 1981]. Tabel 2.2 dan Tabel 2.3 dirombak [Boehm 1981; Tabel 6.3] untuk memetakan fitur-fitur proyeknya pada skala *Precedentedness* dan *Development Flexibility*. Tabel-tabel ini dapat digunakan sebagai penjelasan yang lebih mendalam untuk skala rating PREC dan FLEX yang diberikan pada Tabel 2.1.

2.3.2.1 Precedentedness (PREC)

Jika sebuah produk mirip dengan beberapa proyek yang dikembangkan sebelumnya, maka precedentedness-nya tinggi.

Tabel 2.2. PREC Rating Level

Feature	Very Low	Nominal / High	Extra High
Organizational understanding of product objectives	General	Considerable	Thorough
Experience in working with related software systems	Moderate	Considerable	Extensive
Concurrent development of associated new hardware and operational procedures	Extensive	Moderate	Some
Need for innovative data processing architectures, algorithms	Considerable	Some	Minimal

2.3.2.2 Development Flexibility (FLEX)

Tabel 2.3. FLEX Rating Level

Feature	Very Low	Nominal / High	Extra High
Need for software conformance with pre-established requirements	Full	Considerable	Basic
Need for software conformance with external interface specifications	Full	Considerable	Basic
Combination of inflexibilities above with premium on early completion	High	Medium	Low

Faktor skala PREC dan FLEX sangat hakiki bagi sebuah proyek dan tidak dapat dikontrol. Tiga faktor skala berikutnya mengidentifikasikan manajemen yang dapat dikontrol melalui proyek mana yang dapat mengurangi skala disekonomis dengan mengurangi sumber dari turbulensi, *entropy*, dan *rework* sebuah proyek.

2.3.2.3 Architecture / Risk Resolution (RESL)

Faktor ini menggabungkan dua faktor skala pada Ada COCOMO ("Design Thoroughness by Product Design Review (PDR)" dan "Risk Elimination by PDR" [Boehm-Royce 1989; Gambar 4 dan 5]. Tabel 2.4 menguatkan rating Ada COCOMO untuk membentuk definisi yang lebih komprehensif untuk level rating RESL COCOMO II. Ini juga menghubungkan level rating ke MBASE/RUP *Life Cycle Architecture* (LCA) dan juga batu pijakan *waterfall* PDR. Rating RESL merupakan rata-rata subjektif dari karakteristik yang dirinci.

Tabel 2.4. RESL Rating Level

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Risk Management Plan identifies all critical risk items, establishes milestone for resolving them by PDR or LCA.	None	Little	Some	Generally	Mostly	Fully
Schedule, budget, and internal milestone through PDR or LCA compatible with Risk Management Plan.	None	Little	Some	Generally	Mostly	Fully
Percent of development schedule devoted to establishing architecture, given general product objectives.	5	10	17	25	33	40
Percent of required top software architects available to project.	20	40	60	80	100	120
Tool support available for resolving risk items, developing and verifying architectural specs.	None	Little	Some	Good	Strong	Full
Level of uncertainty in key architecture drives : mission, user interface, COTS, hardware, technology, performance.	Extreme	Signific-ant	Consider-able	Some	Little	Very Little
Number and criticality of risk items.	>10 Critical	5-10 Critical	2-4 Critical	1 Critical	>5 Non-Critical	<5 Non-Critical

2.3.2.4 Team Cohesion (TEAM)

Team Cohesion (TEAM) adalah salah satu atribut dalam *scale factor* yang digunakan untuk mengetahui tingkat kohesi dari tim proyek karena adanya kesulitan untuk sinkronisasi antar *stakeholder* dalam proyek, yaitu pengguna, pelanggan, pengembang, pengelola, dan lain-lain. Faktor skala TEAM menghitung sumber turbulensi dan entropi proyek karena kesulitan dalam sinkronisasi stakeholder proyek: pengguna, pelanggan, pengembang, pengelola, interfacers, lain-lain. Kesulitan-kesulitan tersebut timbul dari perbedaan tujuan dan budaya dari para *stakeholder*, seperti kesulitan dalam menyatukan tujuan serta kurangnya pengalaman *stakeholder* dan keakraban dalam operasi sebagai sebuah tim. Tabel 2.5 memberikan definisi rinci untuk rating level TEAM secara keseluruhan.

Tabel 2.5. TEAM Rating Levels

Characteristic	Very Low	Low	Nominal	High	Very High	Extra High
Consistency of stakeholder objectives and cultures	Little	Some	Basic	Consider-able	Strong	Full
Ability, willingness of stakeholders to accommodate other stakeholders objectives	Little	Some	Basic	Consider-able	Strong	Full
Experience of stakeholders in operating as a team	None	Little	Little	Basic	Consider-able	Extensi ve
Stakeholder teambuilding to achieve shared vision and commitments	None	Little	Little	Basic	Consider-able	Extensi ve

2.3.2.5 Process Maturity (PMAT)

Prosedur untuk menentukan PMAT diselenggarakan oleh *Software Engineering Institute's Capability Maturity Model* (CMM). Periode waktu untuk rating PMAT adalah waktu proyek dimulai. Untuk mengukur PMAT dapat dilakukan dengan dua cara, yang pertama adalah dengan menyimpulkan hasil evaluasi berdasarkan CMM seperti yang ditampilkan pada Tabel 2.6.

Tabel 2.6. PMAT Rating untuk *Estimated Process Maturity* Level (EPML)

PMAT Rating	Maturity Level	EPML
Very Low	CMM Level 1 (lower half)	0
Low	CMM Level 1 (upper half)	1
Nominal	CMM Level 2	2
High	CMM Level 3	3
Very High	CMM Level 4	4
Extra High	CMM Level 5	5

Sedangkan cara yang kedua adalah dengan menggunakan 18 *Key Process Area* (KPA) dari *SEI Capability Maturity Model* [Paulk et al. 1995]. Prosedur untuk menentukan PMAT adalah dengan menentukan persentase kepatuhan untuk masing-masing KPA (dengan skala Likert pada Tabel 2.7).

Tabel 2.7. KPA Rating Level

Key Process Areas (KPA)	Almost Always'	Frequently'	About Half'	Occasionally'	Rarely if Ever'	Does Not Apply'	Don't Know'
Requirements Management • System requirements allocated to software are controlled to establish a baseline for software engineering and management use. • Software plans, products, and activities are kept consistent with the system requirements allocated to software.	.	.	.	.	.	.	.
Software Project Planning • Software estimates are documented for use in planning and tracking the software project. • Software project activities and commitments are planned and documented. • Affected groups and individuals agree to their commitments related to the software project.	.	.	.	.	.	.	.
Software Project Tracking and Development • Actual results and performances are tracked against the software plans • Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans. • Changes to software commitments are agreed to by the affected groups and individuals.	.	.	.	.	.	.	.
Software Subcontract Management • The prime contractor selects qualified software subcontractors. • The prime contractor and the subcontractor agree to their commitments to each other. • The prime contractor and the subcontractor maintain ongoing communications. • The prime contractor tracks the subcontractor's actual results and performance against its commitments.	.	.	.	.	.	.	.

Tabel 2.8. KPA Rating Level (Lanjutan)

Key Process Areas (KPA)	Almost Always'	Frequently'	About Half'	Occasionally'	Rarely if Ever'	Does Not Apply'	Don't Know'
Software Quality Assurance (SQA) • SQA activities are planned • Adherence of software products and activities to the applicable standards, procedures, and requirements is verified objectively. • Affected groups and individuals are informed of software quality assurance activities and results. • Noncompliance issues that cannot be resolved within the software project are addressed by senior management.	.	.	.	.	.	.	.
Software Configuration Management (SCM) • SCM activities are planned • Selected work products are identified, controlled, and available. • Changes to identified work products are controlled. • Affected groups and individuals are informed of the status and content of software baselines.	.	.	.	.	.	.	.
Organization Process Focus • Software process development and improvement activities are coordinated across the organization. • The strengths and weaknesses of the software processes used are identified relative to a process standard. • Organization level process development and improvement activities are planned.	.	.	.	.	.	.	.
Organization Process Definition • A standard software process for the organization is developed and maintaned. • Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.	.	.	.	.	.	.	.

Tabel 2.9. KPA Rating Level (Lanjutan)

Key Process Areas (KPA)	Almost Always[1]	Frequently[2]	About Half[3]	Occasionally[4]	Rarely if Ever[5]	Does Not Apply[6]	Don't Know[7]
Training Program • Training activities are planned • Training for developing the skills and knowledge needed to perform software management and technical roles is provided. • Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.	.	.	.	.	.	.	.
Integrated Software Management • The project's defined software process is a tailored version of the organization's standard software process. • The project is planned and managed according to the project's defined software process.	.	.	.	.	.	.	.
Software Product Engineering • The software engineering tasks are defined, integrated, and consistently performed to produce the software. • Software work products are kept consistent with each other.	.	.	.	.	.	.	.
Intergroup Coordination • The customer's requirements are agreed to by all affected groups. • The commitments between the engineering groups are agreed to by the affected groups. • The engineering groups identify, track, and resolve intergroup issues.	.	.	.	.	.	.	.
Peer Reviews • Peer review activities are planned • Defects in the software work products are identified and removed.	.	.	.	.	.	.	.

Tabel 2.10. KPA Rating Level (Lanjutan)

Key Process Areas (KPA)	Almost Always[1]	Frequently[2]	About Half[3]	Occasionally[4]	Rarely if Ever[5]	Does Not Apply[6]	Don't Know[7]
Quantitative Process Management • The quantitative process management activities are planned. • The process performance of the project's defined software process is controlled quantitatively. • The process capability of the organization's standard software process is known in quantitative terms.	▪	▪	▪	▪	▪	▪	▪
Software Quality Management • The project's software quality management activities are planned. • Measurable goals of software product quality and their priorities are defined. • Actual progress toward achieving the quality goals for the software products is quantified and managed.	▪	▪	▪	▪	▪	▪	▪
Defect Prevention • Defect prevention activities are planned. • Common causes of defects are sought out and identified. • Common causes of defects are prioritized and systematically eliminated.	▪	▪	▪	▪	▪	▪	▪
Technology Change Management • Incorporation of technology changes are planned. • New technologies are evaluated to determine their effect on quality and productivity. • Appropriate new technologies are transferred into normal practice across the organization.	▪	▪	▪	▪	▪	▪	▪
Process Change Management • Continuous process improvement is planned. • Participation in the organization's software process improvement activities is organization wide. • The organization's standard software process and the project's defined software processes are improved continuously.	▪	▪	▪	▪	▪	▪	▪

Sebuah tingkat kematangan proses (EPML) yang setara dihitung sebagai lima kali tingkat kepatuhan rata-rata semua nilai *n* dalam KPA dalam satu proyek tunggal, dimana *does not apply* dan *don't know* tidak dihitung sehingga terkadang membuat nilai *n* kurang dari 18. Setelah masing-masing KPA diberi peringkat, 100% untuk *Almost Always*, 75% untuk *Frequently*, 50% untuk *About Half*, 25% untuk *Occasionally*, dan 1% untuk *Rarely if Ever*, maka EPML dihitung dengan menggunakan persamaan 2.3.

$$EPML = 5x\left(\sum_{i=1}^{n}\frac{KPA\%_i}{100}\right)x\frac{1}{n}$$

Persamaan 2.3

2.4 Effort Multiplier

2.4.1 Post-Architechture Cost Drivers

Model ini adalah model yang paling rinci dari seri COCOMO. Model ini digunakan dalam pengembangan dan pemeliharaan produk perangkat lunak dalam Generator Aplikasi , Sistem Integrasi , atau sektor Infrastruktur (Boehm et al . 2000).

Tujuh belas atribut *Effort Multipliers* (EM) pada model *Post-Architechture* digunakan untuk menyesuaikan upaya nominal (nominal effort) dan Orang-Bulan (Person-Month). Tingkat nominal selalu memiliki *effort multiplier* 1,00 , artinya tidak ada perubahan pada *effort* yang diperkirakan. Peringkat *Off-nominal* umumnya mengubah *effort* yang diperkirakan.

Model COCOMO II dapat digunakan untuk memperkirakan usaha dan jadwal untuk seluruh proyek atau untuk sebuah proyek yang terdiri dari beberapa modul. Ukuran dan peringkat *cost driver* dapat berbeda untuk setiap modul, dengan pengecualian dari atribut *Required Development Schedule* (SCED) dan faktor skala.

2.4.1.1 Product Factors

Product factors menjelaskan variasi dalam upaya yang diperlukan untuk mengembangkan perangkat lunak yang disebabkan oleh karakteristik produk dalam pengembangan. Sebuah produk yang kompleks memiliki persyaratan keandalan yang tinggi atau bekerja dengan *database* yang besar, sehingga akan membutuhkan usaha yang lebih untuk menyelesaikan produk tersebut. Terdapat lima atribut dalam *product factors* dan atribut CPLX memiliki pengaruh yang kuat pada estimasi usaha.

2.4.1.1.1 Required Software Reliability (RELY)

RELY adalah ukuran sejauh mana perangkat lunak harus melakukan fungsi yang ditujukan selama periode waktu tertentu. Jika efek dari kegagalan perangkat lunak hanya sedikit, hanya berupa ketidaknyamanan maka RELY sangat rendah. Jika kegagalan perangkat lunak berisiko terhadap kehidupan manusia maka RELY sangat tinggi. Tabel 2.11 memberikan skema penilaian untuk RELY.

Tabel 2.11. RELY *Cost Driver*

RELY Descriptors :	Slight inconven-ience	Low, easily recoverable losses	Moderate, easily recoverable losses	High financial loss	Risk to human life	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0.82	0.92	1.00	1.10	1.26	n/a

2.4.1.1.2 Database Size (DATA)

Atribut *cost driver* ini mencoba untuk menangkap efek persyaratan data uji yang besar terhadap pengembangan produk. Peringkat ditentukan dengan menghitung D/P, rasio *byte* dalam *database* pengujian untuk SLOC dalam program. Alasan ukuran *database* penting untuk dipertimbangkan adalah karena usaha yang diperlukan untuk menghasilkan data uji yang akan digunakan untuk melaksanakan program. Dengan kata lain, DATA adalah menangkap usaha yang diperlukan untuk merakit dan memelihara data yang dibutuhkan untuk menyelesaikan tes program melalui IOC seperti yang dapat dilihat pada tabel 2.12.

Tabel 2.12. DATA *Cost Driver*

DATA* Descriptors :		Testing DB bytes/Pgm SLOC < 10	10 ≤ D/P < 100	100≤ D/P < 1000	D/P ≥ 1000	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0.90	1.00	1.14	1.28	n/a

DATA dinilai sebagai rendah jika D / P kurang dari 10 dan sebaliknya bernilai sangat tinggi jika lebih besar dari 1000. P diukur dalam baris sumber setara kode (SLOC), yang mungkin melibatkan titik fungsi atau konversi penggunaan kembali.

2.4.1.1.3 Product Complexity (CPLX)

Kompleksitas dibagi menjadi lima area, yaitu *control operation*, *computational operational*, *device-dependant operations*, *data management operations*, dan *interface management operations*. Ciri untuk masing-masing area dapat dilihat pada tabel 2.13 sampai tabel 2.15, sedangkan rating kompleksitas dari peringkat area yang dipilih dapat dilihat pada tabel 2.16.

Tabel 2.13. Komponen CPLX Rating Level

	Control Operations	Computational Operations	Device-dependant Operations	Data Man. Operations	User Interface Man. Operations
Very Low	Straight-line code with a few non-nested structured programming operators. Dos, CASEs, IFTHENELSEs. Simple module composition via procedure calls or simple scripts.	Evaluation of simple expressions : e.g., A=B+C*(D-E)	Simple read, write statements with simple formats.	Simple arrays in main memory. Simple COTS-DB queries, updates.	Simple input forms, report generators.
Low	Straightforward nesting of structured programming operators. Mostly simple predicates.	Evaluation of moderate-level expressions : e.g., D=SQRT(B**2-4.*A*C)	No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level.	Single file subsetting with no data structure changes, no edits, no intermediate files. Moderately complex COTS-DB queries, updates.	Use of simple graphic user interface (GUI) builders.

Tabel 2.14. Komponen CPLX Rating Level (Lanjutan)

	Control Operations	Computational Operations	Device-dependant Operations	Data Man. Operations	User Interface Man. Operations
Nominal	Mostly simple nesting. Some intermodule control. Decision tables. Simple callbacks or message passing, including middleware-supported distributed processing.	Use of standard math and statistical routines. Basic matrix/vector operations.	I/O processing includes device selection, status checking and error processing.	Multi-file input and single file output. Simple structural changes, simple edits. Complex COTS-DB queries, updates.	Simple use of widget set
High	Highly nested structured programming operators with many compound predicates. Queue and stack control. Homogeneous, distributed processing. Single processor soft real-time control.	Basic numerical analysis : multivariate interpolation, ordinary differential equations. Basic truncation, roundoff concerns.	Operations at physical I/O level (physical storage address translations; seeks, reads, etc.). Optimized I/O overlap.	Simple triggers activated by data stream contents. Complex data restructuring.	Widget set development and extension. Simple voice I/O, multimedia.

Tabel 2.15. Komponen CPLX Rating Level (Lanjutan)

	Control Operations	**Computational Operations**	**Device-dependant Operations**	**Data Man. Operations**	**User Interface Man. Operations**
Very High	Reentrant and recursive coding. Fixed-priority interrupt handling. Task synchronization, complex callbacks, heterogeneous distributed processing. Single-processor hard real-time control.	Difficult but structured numerical analysis : near-singular matrix equations, partial differential equations. Simple parallelization.	Routines for interrupt diagnosis, servicing, masking. Communication line handling. Performance-intensive embedded systems.	Distributed database coordination. Complex triggers. Search optimization.	Moderately complex 2D/3D, dynamic graphics, multimedia.
Extra High	Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Distributed hard real-time control.	Difficult and unstructured numerical analysis : highly accurate analysis of noisy, stochastic data. Complex parallelization.	Device timing-dependant coding, micro-programmed operations. Performance-critical embedded systems.	Highly coupled, dynamic relational and object structures. Natural language data management.	Complex multimedia, virtual reality.

Tabel 2.16. CPLX *Cost Driver*

Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0,73	0,87	1	1,17	1,34	1,74

2.4.1.1.4 Developed for Reusability (RUSE)

Atribut *cost driver* ini menghitung upaya tambahan yang diperlukan untuk membangun komponen yang dapat digunakan kembali pada proyek-proyek saat ini atau masa depan. Upaya ini dilakukan dengan menciptakan desain *software* yang lebih generik, dokumentasi yang lebih rumit , dan pengujian yang lebih luas untuk memastikan komponen siap untuk digunakan dalam aplikasi lain. Nilai rating RUSE dapat dilihat pada tabel 2.17.

Tabel 2.17. RUSE *Cost Driver*

RUSE Descriptors		**none**	**across project**	**across program**	**across product line**	**across multiple product lines**
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0,95	1,00	1,07	1,15	1,24

Pengembangan usabilitas ini berpengaruh pada atribut RELY dan DOCU. Atribut RELY harus paling banyak satu tingkat di bawah rating RUSE, sedangkan rating DOCU setidaknya harus berada pada rating *Nominal* dengan rating *Nominal* / *High* untuk RUSE, atau berada pada rating *High* dengan rating *Very High* dan *Extra High* untuk RUSE.

2.4.1.1.5 Documentation Match to Life-Cycle Needs (DOCU)

Beberapa model biaya perangkat lunak memiliki *cost driver* untuk tingkat dokumentasi yang diperlukan. Dalam COCOMO II, skala rating untuk *cost driver* DOCU dievaluasi dalam hal kesesuaian dokumentasi proyek untuk kebutuhan siklus hidupnya. Skala rating mulai dari Sangat Rendah (banyak siklus hidup

perlu terbuka) hingga Sangat Tinggi (sangat berlebihan untuk kebutuhan siklus hidup) dapat dilihat Tabel 2.18.

Tabel 2.18. DOCU *Cost Driver*

DOCU Descriptors	**many life cycle needs uncovered**	**some life cycle needs uncovered**	**right-sized to life cycle needs**	**excessive for life cycle needs**	**very excessive for life cycle needs**	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	0,81	0,91	1,00	1,11	1,23	n/a

Mencoba untuk menghemat biaya melalui tingkat dokumentasi Sangat Rendah atau Rendah umumnya akan dikenakan biaya tambahan selama proses pemeliharaan siklus hidup proyek.

2.4.1.2 Platform Factors

Platform factors mengacu pada kompleksitas *target-machine* perangkat keras dan perangkat lunak dari infrastruktur yang digunakan. Beberapa faktor tambahan pada *platform factors* yang dipertimbangkan, seperti distribusi, paralelisme, *embeddedness*, dan operasi *real-time* telah diakomodasi oleh CPLX pada tabel 2.13 sampai tabel 2.15.

2.4.1.2.1 Execution Time Constraint (TIME)

Merupakan ukuran dari kendala waktu eksekusi yang dipaksakan pada sistem perangkat lunak. Peringkat tersebut dinyatakan dalam persentase dan diharapkan dapat digunakan oleh sistem atau subsistem dalam mengkonsumsi sumber daya waktu. Rating yang digunakan berkisari dari nominal, kurang dari

50% dari sumber daya waktu eksekusi yang digunakan, dan ekstra tinggi 95% dari sumber daya waktu eksekusi dikonsumsi. Untuk lebih jelasnya dapat dilihat pada tabel 2.19.

Tabel 2.19. TIME *Cost Driver*

TIME Descriptors			**50% use of available execution time**	**70% use of available execution time**	**85% use of available execution time**	**95% use of available execution time**
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1,00	1,11	1,29	1,63

2.4.1.2.2 Main Storage Constraint (STOR)

Peringkat ini merupakan tingkat kendala penyimpanan utama yang dikenakan pada sistem perangkat lunak atau subsistem. Mengingat peningkatan yang luar biasa dalam waktu eksekusi prosesor yang tersedia dan penyimpanan utama, seseorang dapat mempertanyakan apakah variabel kendala masih relevan. Namun, banyak aplikasi terus berkembang untuk mengkonsumsi sumber daya apapun yang tersedia, terutama dengan perkembangan COTS yang besar aplikasi yang besar dan berkembang produk COTS, membuat *cost driver* ini masih relevan. Rating berkisar dari nominal (kurang dari 50%), untuk ekstra tinggi (95%). Untuk lebih jelasnya dapat dilihat pada tabel 2.20.

Tabel 2.20 STOR *Cost Driver*

STOR Descriptors			**50% use of available storage**	**70% use of available storage**	**85% use of available storage**	**95% use of available storage**
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	n/a	1,00	1,05	1,17	1,46

2.4.1.2.3 Platform Volatility (PVOL)

"Platform" yang digunakan di sini berarti kompleksitas perangkat keras dan perangkat lunak (OS, DBMS, dll) dari produk perangkat lunak. Jika perangkat lunak yang akan dikembangkan adalah sistem operasi maka *platform* adalah perangkat keras komputer. Jika sistem manajemen *database* yang akan dikembangkan maka *platform* adalah perangkat keras dan sistem operasi. Jika *browser teks* jaringan yang akan dikembangkan maka *platform* adalah jaringan, perangkat keras komputer, sistem operasi, dan informasi repositori yang didistribusikan. *Platform* ini mencakup setiap *compiler* atau perakit yang mendukung pengembangan sistem perangkat lunak. Peringkat ini berkisar dari rendah, di mana ada perubahan besar setiap 12 bulan, hingga peringkat yang sangat tinggi, di mana ada perubahan besar setiap dua minggu. Untuk lebih jelasnya dapat dilihat pada tabel 2.21.

Tabel 2.21 PVOL *Cost Driver*

PVOL Descriptors		**major change every 12 mo.; minor change every 1 mo**	**major 2 mo.; minor 2 wk.**	**major 2 mo.; minor 1 wk.**	**major 2 wk; minor 2 days**	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	n/a	0,87	1,00	1,15	1,3	n/a

2.4.1.3 Personnel Factors

Setelah ukuran produk, faktor orang memiliki pengaruh terkuat dalam menentukan jumlah usaha yang diperlukan untuk mengembangkan produk perangkat lunak. Rating untuk *personnel factors* ditujukan untuk menilai kemampuan dan pengalaman tim pengembangan, bukan individu. Peringkat ini memiliki kemungkina untuk berubah selama proyek berlangsung.

2.4.1.3.1 Analyst Capability (ACAP)

Analis adalah personel yang bekerja pada persyaratan, desain tingkat tinggi dan desain rinci. Atribut utama yang harus dipertimbangkan dalam penilaian ini adalah analisis dan kemampuan desain, efisiensi dan ketelitian, serta kemampuan untuk berkomunikasi dan bekerja sama. Peringkat tidak harus mempertimbangkan tingkat pengalaman analis, yang dinilai dengan APEX, LTEX, dan PLEX. Tim analis yang jatuh di persentil lima belas dinilai sangat rendah dan mereka yang jatuh dalam persentil sembilan puluh dinilai sebagai sangat tinggi. Untuk lebih jelasnya dapat dilihat pada tabel 2.22.

Tabel 2.22 ACAP *Cost Driver*

ACAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,42	1,19	1,00	0,85	0,71	n/a

2.4.1.3.2 Programmer Capability (PCAP)

Tren saat ini terus menekankan pentingnya kemampuan analis. Namun meningkatnya kompleksitas paket COTS, dan *leverage* produktivitas yang signifikan terkait dengan kemampuan *programmer* 'untuk menangani paket COTS, menunjukkan kecenderungan yang lebih tinggi akan pentingnya kemampuan *programmer*.

Evaluasi harus didasarkan pada kemampuan *programmer* sebagai tim bukan sebagai individu. Faktor utama yang harus dipertimbangkan dalam penilaian adalah kemampuan, efisiensi dan ketelitian, serta kemampuan untuk berkomunikasi dan bekerja sama. Pengalaman *programmer* tidak harus dipertimbangkan di sini, karena dinilai dengan APEX, LTEX, dan PLEX. Sebuah tim *programmer* dinilai sangat rendah pada persentil lima belas dan tim *programmer* dinilai sangat tinggi pada persentil sembilan puluh. Untuk lebih jelasnya dapat dilihat pada tabel 2.23.

Tabel 2.23 PCAP *Cost Driver*

PCAP Descriptors	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,34	1,15	1,00	0,88	0,76	n/a

2.4.1.3.3 Personnel Continuity (PCON)

Skala rating untuk PCON adalah dalam hal *turnover* pegawai tahunan, mulai dari 3% untuk kontinuitas sangat tinggi, hingga 48% untuk kontinuitas sangat rendah. Untuk lebih jelasnya dapat dilihat pada tabel 2.24.

Tabel 2.24 Contoh PCON *Cost Driver*

PCON Descriptors	48% / year	24% / year	12% / year	6% / year	3% / year	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,29	1,12	1,00	0,9	0,81	n/a

2.4.1.3.4 Applications Experience (APEX)

Rating untuk *cost driver* ini (sebelumnya berlabel AEXP) tergantung pada tingkat aplikasi pengalaman tim proyek pengembangan sistem atau sub sistem perangkat lunak. Peringkat tersebut didefinisikan setara dengan tingkat pengalaman tim proyek terhadap suatu aplikasi. Rating yang sangat rendah diberikan untuk pengalaman aplikasi yang kurang dari 2 bulan. Sedangkan rating

yang sangat tinggi diberikan untuk pengalaman 6 tahun atau lebih. Untuk lebih jelasnya dapat dilihat pada tabel 2.25.

Tabel 2.25 APEX *Cost Driver*

APEX Descriptors	**≤ 2 months**	**6 months**	**1 years**	**3 years**	**6 years**	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,22	1,1	1,00	0,88	0,81	n/a

2.4.1.3.5 Platform Experience (PLEX)

Model *Post-Architechture* memperluas pengaruh produktivitas terhadap pengalaman *Platform*, PLEX (sebelumnya berlabel PEXP), dengan mengakui pentingnya memahami penggunaan *platform* yang lebih kuat, termasuk antarmuka pengguna dengan grafis yang lebih, *database*, jaringan, dan kemampuan *middleware* terdistribusi. Untuk lebih jelasnya dapat dilihat pada tabel 2.26.

Tabel 2.26 PLEX *Cost Driver*

PLEX Descriptors	**≤ 2 months**	**6 months**	**1 years**	**3 years**	**6 years**	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,99	1,09	1,00	0,91	0,85	n/a

2.4.1.3.6 Language and Tool Experience (LTEX)

Atribut ini adalah ukuran tingkat pengalaman tim proyek terhadap bahasa pemrograman yang digunakan dalam pengembangan perangkat lunak, termasuk

penggunaan alat-alat pada saat melakukan persyaratan dan representasi desain dan analisis, manajemen konfigurasi, ekstraksi dokumen, manajemen perpustakaan, gaya dan format program, memeriksa konsistensi, perencanaan dan pengendalian, dan lain-lain. Selain pengalaman dalam bahasa pemrograman, pengalaman untuk mendukung *tool* set proyek juga mempengaruhi pengembangan usaha. Rating rendah diberikan untuk pengalaman kurang dari 2 bulan. Sedangkan rating yang sangat tinggi diberikan untuk pengalaman 6 tahun atau lebih. Untuk lebih jelasnya dapat dilihat pada tabel 2.27.

Tabel 2.27 LTEX *Cost Driver*

LTEX Descriptors	**≤ 2 months**	**6 months**	**1 years**	**3 years**	**6 years**	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,20	1,09	1,00	0,91	0,84	n/a

2.4.1.4 Project Factors

Project factors menghitung pengaruh pada usaha yang telah diperkirakan seperti penggunaan perangkat lunak modern, lokasi dari tim pengembang, dan kompresi dari jadwal proyek.

2.4.1.4.1 Use of Software Tools (TOOL)

Penggunaan *Software tools* telah meningkat secara signifikan sejak tahun 1970-an. Rating untuk TOOL berkisar dari mengedit sederhana dan kode untuk level sangat rendah, dan rating alat manajemen siklus hidup yang terintegrasi untuk level sangat tinggi. Rating Nominal pada atribut TOOL dalam

COCOMO'81 setara dengan rating Sangat Rendah pada atribut TOOL dalam COCOMO II. Skala rating TOOL dapat dilihat pada tabel 2.28.

Tabel 2.28 TOOL *Cost Driver*

TOOL Descriptors	**edit, code, debug**	**simple, frontened, 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**	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,17	1,09	1,00	0,9	0,78	n/a

2.4.1.4.2 Multisite Development (SITE)

Dengan meningkatnya frekuensi perkembangan *multisite*, dan indikasi efek pembangunan *multisite* yang signifikan, *cost driver* SITE telah ditambahkan dalam COCOMO II untuk menentukan peringkat *cost driver* ini penilaian diambil dari rata-rata dua faktor, yaitu *site collocation* (dari sepenuhnya *collocated* untuk distribusi internasional) dan *communication support* (dari *surface mail* dan beberapa akses telepon ke multimedia interaktif).

Sebagai contoh, jika sebuah tim sepenuhnya *collocated*, tidak perlu multimedia interaktif untuk mencapai rating Ekstra Tinggi, cukup menggunakan *E-mail narrowband*. Untuk lebih jelasnya dapat dilihat pada tabel 2.29.

Tabel 2.29 SITE *Cost Driver*

SITE Descriptors	**internati onal**	**multi-city and multi-company**	**multi-city or multi-company**	**same city or metro area**	**same building or complex**	**fully collocate d**
Communicat ions Descriptors	**some phone, mail**	**individu al phone, FAX**	**narrow band email**	**wideband electronic communi cation**	**wideba nd elect. Comm., occasio nal video conf.**	**interacti ve multime dia**
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,22	1,09	1,00	0,93	0,86	0,8

2.4.1.4.3 Required Development Schedule (SCED)

Peringkat ini mengukur kendala jadwal yang dikenakan pada tim proyek pengembangan perangkat lunak. Peringkat tersebut didefinisikan dalam hal persentase jadwal *stretch-out* atau percepatan sehubungan dengan jadwal nominal untuk sebuah proyek yang membutuhkan jumlah yang diberikan usaha. Jadwal yang dipercepat cenderung menghasilkan lebih banyak usaha dalam tahap awal untuk menghilangkan risiko dan memperbaiki arsitektur, lebih banyak usaha dalam tahap selanjutnya untuk mencapai pengujian lebih dan dokumentasi secara paralel. Pada tabel 2.30 jadwal kompresi 75% dinilai sangat rendah dan jadwal *stretch -out* dari 160% berperingkat sangat tinggi. *Stretch-out* tidak menambahkan atau mengurangi usaha, karena ukuran tim yang kecil umumnya seimbang dengan kebutuhan untuk menjalankan fungsi administrasi proyek selama jangka waktu yang lama.

SCED adalah satu-satunya *cost drivers* yang digunakan untuk menjelaskan pengaruh jadwal kompresi / ekspansi untuk keseluruhan proyek. Faktor skala juga digunakan untuk menggambarkan keseluruhan proyek. Rating untuk atribut SCED dapat dilihat pada tabel 2.30.

Tabel 2.30 SCED *Cost Driver*

SCED Descriptors	75% of nominal	85% of nominal	100% of nominal	130% of nominal	160% of nominal	
Rating Levels	Very Low	Low	Nominal	High	Very High	Extra High
Effort Multipliers	1,43	1,14	1,00	1,00	1,00	n/a

2.5 Evaluasi Model Perkiraan Biaya Perangkat Lunak

Proses evaluasi model perkiraan biaya perangkat lunak adalah dengan membandingkan akurasi dari biaya estimasi dengan biaya aktualnya (Boehm 1981 dalam Indri et al 2001). Dalam COCOMO II digunakan dua jenis nilai untuk mengukur keakuratan dari sebuah model perkiraan biaya perangkat lunak (Briand et all, 1988), yaitu *Relative Error* (RE) atau *Magnitude of Relative Error* (MRE) dan PRED (P) dengan persamaan sebagai berikut :

$$RE = \frac{(usaha_{\text{modifikasi}} - usaha_{aktual})}{usaha_{aktual}}$$

Persamaan 2.4

$$MRE = \frac{\left|Usaha_{\text{modifikasi}} - Usaha_{aktual}\right|}{Usaha_{aktual}}$$

Persamaan 2.5

$$PRED(P) = \frac{K}{N}$$

Persamaan 2.6

Dimana :

K : jumlah proyek dimana MRE kurang dari atau sama dengan P

N : jumlah proyek

Presentase prediksi P% terdiri dari 3 jenis, yaitu PRED (20%), PRED (25%), dan PRED (30%). Misalkan model yang diusulkan dievaluasi pada PRED (25%), maka yang harus dilakukan adalah menghitung jumlah MRE di persamaan yang kurang dari atau sama dengan 25% dan membaginya dengan jumlah proyek.