Makara Journal of Technology

Volume 6 | Issue 2 Article 1

8-2-2002

Analogy Method Development for Cost Estimation of Software Design

Riyanarto Sarno
1Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Kampus ITS Sukolilo, Surabaya, info@riyanarto.com

Joko Lianto Buliali
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Kampus ITS Sukolilo, Surabaya

Siti Maimunah
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Al Falah Jl. Sulawesi No. 1 Surabaya

PENGEMBANGAN METODE *ANALOGY* UNTUK ESTIMASI BIAYA RANCANG BANGUN PERANGKAT LUNAK

Riyanarto Sarno[1], Joko Lianto Buliali[1], Siti Maimunah[2]

[1]Jurusan Teknik Informatika, Fakultas Teknologi Informasi,
Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya

[2]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Al Falah
Jl. Sulawesi No. 1 Surabaya

info@riyanarto.com

Abstrak

Salah satu aspek penting pada perencanaan dan manajemen proyek rancang bangun perangkat lunak adalah mengestimasi biaya sebuah proyek. Beberapa metode telah digunakan untuk mengestimasi biaya suatu proyek perangkat lunak, dan metode *Analogy* merupakan metode baru yang menghasilkan estimasi relatif akurat. Makalah ini menunjukkan hasil studi penyempurnaan metode *Analogy*, yang meliputi penentuan model matematis untuk memilih proyek sejenis sebagai acuan estimasi, serta penentuan estimasi *effort* dan biaya. Studi ini menghasilkan penyempurnaan teknik estimasi biaya dengan parameter biaya lebih lengkap sehingga hasil estimasi relatif lebih baik daripada metode *Analogy* baku.

Abstract

The important aspect of planning and managing software development project is to estimate the cost of a project. There are several methods for estimating the cost of a software development project, and the *Analogy* method is a method which gives relatively better estimates. This paper shows that the modified Analogy method selects a closer project reference, estimates more accurate project effort and cost. This study enhances the cost estimate technique by including valid and complete cost parameters, therefore the estimate of a project cost is better than the result of the standard Analogy method.

Keywords: *Analogy method, similarity, software development, cost estimate, project management.*

Pendahuluan

Alokasi sumber daya dan penentuan biaya sebuah proyek marupakan salah satu aspek yang penting dalam perencanaan dan manajemen proyek pengembangan perangkat lunak [1]. Dengan demikian akurasi estimasi biaya suatu proyek merupakan salah satu penentu keberhasilan suatu proyek.

Sebelum menentukan estimasi biaya, terlebih dulu harus diestimasi besar *effort*, karena *effort* merupakan komponen biaya utama yang akan berpengaruh pada hampir semua obyek biaya [2]. Dengan demikian teknik estimasi biaya meliputi metode estimasi *effort*. Beberapa metode yang digunakan untuk estimasi *effort* dibahas pada referensi [2, 3].

Saat ini terdapat beberapa teknik estimasi biaya, diantaranya adalah COCOMO, SLIM, masing-masing mempunyai kelebihan dan kekurangan, namun tidak ada satupun teknik yang terbaik [4].

Teknik-teknik tersebut dapat dikelompokkan dalam tiga kategori berdasarkan metode yang digunakan [2], yaitu: *expert judgement, algorithmic models, Analogy.*

Berdasarkan observasi yang dilakukan oleh Snell, ternyata ada beberapa kelemahan pada metode algoritmik, antara lain penambahan *waktu* kalibrasi

sistem ke lingkungan lokalnya, semakin banyak waktu yang digunakan untuk proses kalibrasi akan semakin akurat estimasi yang dihasilkan. Kemerer telah mengadakan suatu studi yang menunjukkan bahwa beda antara estimasi dengan nilai aktual bervariasi antara 85 – 610 %. Kalibrasi model dapat meningkatkan akurasi, namun model-model ini tetap menghasilkan *error* antara 50 – 100 % [5].

Meskipun banyak penelitian menghasilkan estimasi biaya perangkat lunak mengacu pada metode algoritmik, namun metode *expert judgement* lebih banyak digunakan [5]. Metode ini didasarkan pada pengalaman manajer pada proyek yang serupa. Di sini akurasi prediksi didasarkan pada kompetensi, pengalaman, obyektivitas dan persepsi dari *experts* [1].

Pada tahun 1989 Myers memperkenalkan metode *Analogy* [6]. Metode ini dapat diaplikasikan ketika terdapat proyek-proyek lain dalam domain aplikasi serupa yang telah dikerjakan. Biaya proyek baru diestimasi melalui analogi dengan proyek-proyek yang telah ada tersebut. Selanjutnya Martin Shepperd dan Chris Schofield (1997) berusaha mengadakan penelitian terhadap metode ini dan sampai sekarang masih dalam penelitian [2]. Metode ini dipandang sebagai bentuk sistematis dari *expert judgement* karena para ahli sering mencari situasi yang analog untuk menjelaskan opini mereka. Metode ini meliputi pengenalan ciri (nilai parameter) dari proyek baru yang akan diestimasi. Ciri-ciri tersebut selanjutnya digunakan sebagai dasar pencarian proyek lama yang serupa. Nilai *effort* proyek serupa (analog) tersebut kemudian digunakan sebagai acuan untuk mengestimasi *effort* proyek baru.

Shepperd telah menganalisa nilai *Mean Magnitude of Relative Error (MMRE*), ditunjukkan Persamaan (1), terhadap enam dataset dengan menggunakan metode *Analogy*, metode regresi linier dan *stepwise*, hasilnya ditunjukkan pada Tabel 1.

$$MMRE = 100/n \sum_{i=1}^{i=n} \left(\frac{\left(\left| E_{pred} - E_{act} \right| \right)}{E_{act}} \right)_i \quad (1)$$

dimana:

Epred	:	nilai estimasi *effort*
Eact	:	nilai *effort* aktual
n	:	jumlah data *effort* aktual

Tabel 1. Nilai MMRE pada estimasi *effort* [2]

Dataset	Analogy	Regresi Linier	Regresi Stepwise
Albrecht	62 %	85 %	90 %
Atkinson	39 %	57 %	45 %
Desharnais	64 %	66 %	66 %
Finnish	62 %	133 %	101 %
Kemerer	62 %	107 %	107 %
Mermaid	78 %	252 %	252 %

Tabel 1 menunjukkan akurasi estimasi dari ketiga metode. Estimasi *effort* menggunakan metode *Analogy* menghasilkan *error* terkecil. Secara umum estimasi metode *Analogy* sama atau lebih baik daripada kedua metode yang lain.

Hasil studi Shepperd melatarbelakangi penelitian ini, yaitu mengembangkan metode *Analogy* untuk mengestimasi *effort*, dan selanjutnya mengembangkan teknik estimasi biaya. Pengembangan metode *Analogy* dilakukan dengan memperbaiki perhitungan pemilihan data proyek serupa dan menambah parameter sehingga estimasi *effort* lebih akurat. Kemudian dilakukan perbaikan terhadap kelengkapan komponen biaya, sehingga estimasi biaya juga lebih akurat.

Metode Penelitian

Metode *Analogy* adalah metode yang menyimpan hasil observasi pada proyek-proyek yang telah lalu, seperti: *effort* yang dibutuhkan untuk mengembangkan proyek, *platform* pemrograman dan sebagainya. Ketika terdapat proyek baru, proyek tersebut diidentifikasi berdasarkan nilai masing-masing parameternya. Selanjutnya dicari kemiripannya dengan proyek-proyek lama. Proyek lama yang mirip dianggap sebagai proyek analog dan estimasi *effort* dilakukan dengan cara menghitung *effort* rata-rata dari *effort* proyek-proyek analog.

Beberapa tahapan estimasi dengan menggunakan metode *Analogy*, adalah:

1. Pengukuran parameter-parameter dari proyek baru sesuai dengan basis data (dataset) yang akan digunakan.
2. Penentuan similaritas antara proyek baru dengan proyek-proyek lama dalam dataset.
3. Pemilihan proyek analog (mirip) dengan proyek baru.
4. Estimasi *effort* proyek baru.

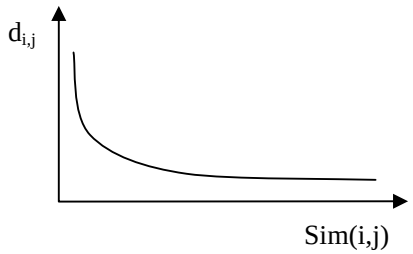
Pada tahap pertama, pengukuran parameter dilakukan sesuai dengan parameter proyek dari dataset. Karena dataset yang digunakan dalam penelitian ini adalah Albrecht dataset, maka digunakan teknik pengukuran *Function Points* yang mengandung parameter *input*, *output*, *inquiry*, *interface* dan *file* [4]. Dengan dua parameter tambahan, yaitu *Source Lines of Code* (*SLOC*) dan *Effort.*

Untuk meningkatkan akurasi *effort*, ditambahkan lagi dua parameter pada dataset yaitu *transition* dan *transformation* sehingga menjadi *3D Function Points* [5].

Tahap kedua, dalam menentukan similaritas digunakan metode *Nearest Neighbour Algorithm*. Metode ini merupakan algoritma umum yang didasarkan pada pengukuran jarak tiap-tiap parameter. Semakin kecil jarak

tiap-tiap parameter berarti semakin besar nilai similaritas karena jarak dianggap sebagai penyimpangan.

Jika jarak antara dua proyek ke-i dan ke-j diberi simbol $d_{i,j}$ dan similaritas antara dua proyek ke-i dan ke j diberi simbol Sim(i,j), maka korelasi antara jarak dan similaritas dapat dilihat pada Gambar 1.



Gambar 1. Korelasi antara Jarak ($d_{i,j}$) dengan Similaritas (*Sim(i,j)*)

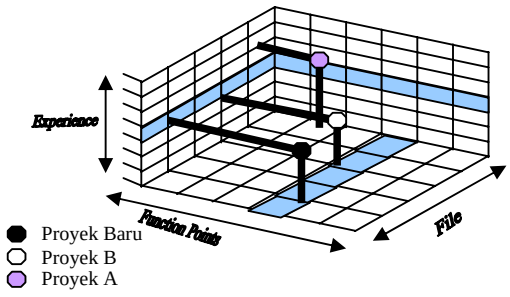
Untuk menentukan similaritas, data-data yang ada harus dinormalisasi terlebih dahulu dengan menggunakan Persamaan (2) agar mempunyai bobot yang sama, yaitu antara 0 dan 1.

$$C_{ij} = \frac{f_{ij} - \min(f_j)}{\max(f_j) - \min(f_j)} \qquad (2)$$

dimana:

Cij	:	nilai normalisasi parameter ke *j* proyek ke *i*
i	:	nomor proyek
j	:	nomor parameter dalam dataset
f_{ij}	:	nilai proyek ke *i* parameter ke *j*
$max(f_j)$	:	nilai maksimum dari parameter ke *j*
$min(f_j)$	:	nilai minimum dari parameter ke *j*

Setelah semua data dinormalisasi, kemudian masing-masing data proyek diletakkan pada area dimensi n (n adalah jumlah parameter proyek). Gambar 2 adalah contoh area 3 dimensi dengan parameter *experience*, *file* dan *function points*. Pada gambar tersebut proyek B lebih mirip daripada A, karena jaraknya lebih dekat.



Gambar 2. Contoh area dimensi 3 dengan parameter *experience, function points* dan *file*

Suatu algoritma umum yang diberikan oleh Aha [3] untuk menghitung *similarity* dinyatakan sebagai berikut,

$$SIM(C_1, C_2, F) = \frac{1}{\sqrt{\sum_{j \in F} Feature\_dissimilarity(C_{1j}, C_{2j})}} \qquad (3)$$

dimana

F	:	set dari n parameter
C_1 dan C_2	:	proyek ke *1 dan 2*

$$Feature\_dissimilarity(C_{1j}, C_{2j}) \begin{cases} (C_{1j} - C_{2j})^2 ; feature = numeric \\ 0 \quad ; feature = kategoris, C_{1i} = C_{2i} \\ 1 \quad ; feature = kategoris, C_{1i} \neq C_{2i} \end{cases}$$

Feature dalam persamaan di atas adalah parameter proyek. Contoh *feature* = *kategoris* adalah jika parameter ke i mempunyai nilai nama bahasa pemrograman. $C_{1i} = C_{2i}$ jika proyek 1 dan proyek 2 menggunakan bahasa pemrograman yang sama, misalnya C++.

Dari persamaan di atas dapat dilihat bahwa *similarity* berbanding terbalik dengan akar jarak antar parameter dari dua proyek (*feature dissimilarity*).

Mengestimasi *effort* proyek baru dapat dilakukan dengan mencari proyek serupa sebagai acuan, untuk itu dapat digunakan persamaan $Y = ax_1 + bx_2 + cx_3$. Y adalah nilai estimasi *effort* dan $x_1, x_2, x_3, \ldots, x_n$ adalah parameter-parameter proyek; misalnya : *input*, *output*, *inquiry*, *file*.

Tabel 2 adalah contoh data set yang memenuhi $Y = ax_1 + bx_2 + cx_3$. Pada tabel ini nilai minimum x_1, x_2, x_3 adalah 0. Sedangkan nilai maksimum $x_1 = 3$, $x_2 = 6$, $x_3 = 1$ dan Y =300. Tabel 3 adalah hasil normalisasi dari Tabel 2 berdasarkan Persamaan (2).

Tabel 2. Fungsi $Y = 100x_1 + 50x_2 + 200x_3$

No.Data	Y	X_1	X_2	X_3
1	50	0	1	0
2	100	1	0	0
3	100	0	2	0
4	150	1	1	0
5	150	0	3	0
6	190	2	0	0
7	190	1	2	0
8	250	0	5	0
9	300	0	6	0
10	300	3	0	0

Tabel 3. Hasil normalisasi Tabel 2

No.Data	Y	X_1	X_2	X_3
1	0,17	0,00	0,17	0,00
2	0,33	0,33	0,00	0,00
3	0,33	0,00	0,33	0,00
4	0,50	0,33	0,17	0,00
5	0,50	0,00	0,50	0,00
6	0,67	0,67	0,00	0,00
7	0,67	0,33	0,33	0,00
8	0,83	0,00	0,83	0,00
9	1,00	0,00	1,00	0,00
10	1,00	1,00	0,00	0,00

Jika diberikan data baru $x_1 = 1$, $x_2 = 0$ dan $x_3 = 0$, nilai similaritas antara data baru dengan data di atas berdasarkan Persamaan (3) dapat dilihat pada Tabel 4.

Tabel 4. Nilai similaritas antara data baru dengan data dalam Tabel 2 berdasarkan persamaan (3).

No.Data	Y	X_1	X_2	X_3	Sim()
1	50	0	1	0	2,68
2	100	1	0	0	∞
3	100	0	2	0	2,12
4	150	1	1	0	6,00
5	150	0	3	0	1,66
6	190	2	0	0	3,00
7	190	1	2	0	3,00
8	250	0	5	0	1,11
9	300	0	6	0	0,87
10	300	3	0	0	1,22
Baru		1	0	0	

Dari Tabel 4, data analog dengan nilai similaritas tertinggi adalah No.Data 2, 4, 6 dan 7. Selanjutnya nilai estimasi Y dihitung dari nilai rata-rata data analog dengan similaritas tertinggi. Nilai estimasi Y adalah 162,5, sedangkan nilai sebenarnya adalah 100.

Hasil dan Pembahasan

Ada beberapa kelemahan dari metode *Analogy* yang diajukan oleh Shepperd, yang telah dibahas pada subbab 2. Pertama, batasan nilai similaritas dari data set yang diacu masih ditentukan secara subyektif. Kedua, jika data tidak lengkap, kemungkinan terjadi kesalahan estimasi cukup besar karena nilai $max(f_j)$ untuk normalisasi belum diketahui. Hal ini berakibat pada kesalahan penentuan similaritas dan pemilihan data analog. Ketiga, penentuan nilai estimasi berdasarkan nilai rata-rata data analog akan mengalami permasalahan jika data baru berada di luar area basis data yang ada. Kelemahan keempat yaitu hasil pengukuran similaritas tidak konsisten. Nilai similaritas antara dua proyek A dan B yang diukur dari A ke B akan berbeda dengan hasil pengukuran dari B ke A.

Metode Estimasi *Effort*

Pada Persamaan (3) terlihat bahwa nilai similaritas antara dua proyek berbanding terbalik dengan jarak antara dua proyek tersebut. Gonzales juga menyatakan bahwa untuk mencari kemiripan suatu pola vektor dengan pola vektor yang lain dapat dilihat dari jarak minimum dari pola vektor tersebut. Jarak dihitung berdasarkan jarak *Euclidean*, yaitu menjumlahkan perbedaan masing-masing vektor dari pola yang dicari kemiripannya. Jarak minimum mewakili dua buah pola yang paling mirip [7]. Dengan demikian untuk mencari data analog dapat digunakan jarak antar proyek. Karena proyek dengan beberapa parameternya dalam metode ini dianggap sebuah titik dalam dimensi n (n adalah jumlah parameter proyek), maka proyek yang nilai parameternya tidak lengkap (nilai parameter yang perlu diestimasi) dapat dianggap sebagai sebuah garis atau bidang, tergantung berapa parameter yang perlu dicari nilainya. Jarak antara titik dengan garis atau bidang adalah panjang garis normal [8].

Jika $Q(x_1, y_1, z_1)$ adalah titik dalam garis, posisi normal $n = (a, b, c)$ dan titik awal terletak pada Q. Sebagaimana diilustrasikan pada Gambar 3, jarak d sama dengan panjang proyeksi orthogonal dari $\overrightarrow{QP_0}$ pada n [11]. Dengan demikian,

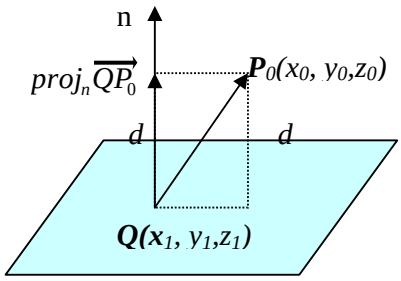
$$d = \left\| proj_n \overrightarrow{QP_0} \right\| = \frac{\left| \overrightarrow{QP_0}.n \right|}{\|n\|} \qquad (4)$$

dimana

$$\overrightarrow{QP_0} = (x_0 - x_1, y_0 - y_1, z_0 - z_1) \qquad (5)$$

$$\overrightarrow{QP_0}.n = a(x_0 - x_1) + b(y_0 - y_1) + c(z_0 - z_1) \qquad (6)$$

$$\|n\| = \sqrt{a^2 + b^2 + c^2} \qquad (7)$$



Gambar 3. Jarak antara titik dengan bidang

dengan demikian

$$d = \frac{\left| a(x_0 - x_1) + b(y_0 - y_1) + c(z_0 - z_1) \right|}{\sqrt{a^2 + b^2 + c^2}} \qquad (8)$$

Sebagai contoh pada Tabel 5, nilai konstanta a = 100, b = 50 dan c = 200. Jarak (d) antara F dengan A, B, C, D, E ditunjukkan pada kolom terakhir.

Tabel 5. Contoh data baru dan jarak (d)

No.Data	Y	X_1 (x)	X_2 (y)	X_3 (z)	d
1 (A)	100	0	2	0	3,33
2 (B)	150	1	1	0	3,11
3 (C)	250	0	1	1	2,67
4 (D)	300	0	6	0	2,44
5 (E)	400	1	6	0	2,00
Baru (F)		4	1	2	

Berdasarkan jarak terdekat, maka E lebih mirip dengan F dan dapat dianggap sebagai data analog untuk dijadikan acuan estimasi.

Sebelum Persamaan (8) dipakai untuk menghitung jarak antara proyek baru dengan proyek-proyek yang ada

dalam basis data terlebih dulu konstanta dan nilai maksimum tiap-tiap parameter dihitung menggunakan metode *Analogy*.

Berikut ini cara mendapatkan konstanta dan nilai maksimum tiap-tiap parameter. Tabel 6 dihitung berdasarkan Tabel 5 yang ditambah dengan kolom normalisasi dan kolom perbedaan nilai Y ternomalisasi dengan total nilai x_i ($|Y - \sum x_i|$).

Tabel 6. Penentuan nilai data baru untuk mencari nilai konstanta x_i

Data					Normalisasi				
No.Data	Y	X_1	X_2	X_3	Y	X_1	X_2	X_3	$\|Y - \sum X_i\|$
1	100	0	2	0	0,250	0,000	0,333	0,000	0,083
2	150	1	1	0	0,375	1,000	0,167	0,000	0,792
3	250	0	1	1	0,625	0,000	0,167	1,000	0,542
4	300	0	6	0	0,750	0,000	1,000	0,000	0,250
5	400	1	6	0	1,000	1,000	1,000	0,000	1,000
Baru		1	0	0					

Untuk mendapatkan nilai maksimum x_i, maka x_i data baru diberi nilai 1 dan x yang lain diberi nilai 0, pemberian nilai 1 ini dilakukan untuk semua x_i. Untuk masing-masing x_i, jika jarak terdekat berada pada baris j, maka konstanta untuk masing-masing x_i dapat diperoleh dengan Persamaan (9),

$$Const(i) = \sum X_{iNormBaru} / \sum X_{iNorm,\,j} * Y_j \qquad (9)$$

dimana:

Const(i) : nilai konstanta ke i
Y_j : nilai Y ke j
$X_{iNormBaru}$: nilai normalisasi x_i data baru
X_{iNormj} : nilai normalisasi x_i data ke j

Nilai maksimum dari masing-masing x_i dapat diperoleh dengan Persamaan (10),

$$Max(i) = \frac{Max(Y)}{const(i)} \qquad (10)$$

dimana:

Max(Y) : nilai maksimum Y dari semua data
const(i) : konstanta dari masing-masing x_i

Nilai konstanta dan nilai x maksimum tergantung pada kelengkapan data yang ada. Jika data yang tersedia tidak lengkap, maka nilai konstanta dan nilai x maksimum yang dihasilkan tidak stabil. Jika demikian harus dilakukan iterasi beberapa kali untuk mencapai kestabilan nilai. Sebelum dilakukan iterasi, data normalisasi harus di-*update* terlebih dahulu dengan nilai maksimum yang baru. Nilai x maksimum dan konstanta dinyatakan stabil jika nilai $|Y - \sum x_i|$ pada iterasi berikutnya tidak mengalami perubahan.
Setelah stabil, diperoleh konstanta a = 100, b = 50 dan c = 200 serta nilai maksimum $x_1 = 4$, $x_2 = 8$ dan $x_3 = 2$. Data pada Tabel 6 akan berubah menjadi Tabel 7.

Tabel 7. Normalisasi dengan nilai maksimum $x_1 = 4$, $x_2 = 8$ dan $x_3 = 2$

Data					Normalisasi				$\sum X_i$	$\|Y - X_i\|$	*d()*
No. Data	Y	X_1	X_2	X_3	Y	X_1	X_2	X_3			
1	100	0	2	0	0,25	0,00	0,25	0,00	0,25	0,00	0,05
2	150	1	1	0	0,37	0,25	0,12	0,00	0,37	0,00	0,03
3	250	0	1	1	0,62	0,00	0,12	0,50	0,62	0,00	0,35
4	300	0	6	0	0,75	0,00	0,75	0,00	0,75	0,00	0,05
5	400	1	6	0	1,00	0,25	0,75	0,00	1,00	0,00	0,16
Baru		1	0	0		0,25	0,00	0,00	0,25	*Min(d()) :*	0,03

Tabel 7 dapat digunakan untuk mencari data analog (data dengan jarak minimum) dan nilai estimasi *effort* dapat diperoleh dengan menggunakan Persamaan (11).

$$Y_{Baru} = \sum X_{iNormBaru} / \sum X_{iNorm,\,j} * Y_j \quad manmonth \quad (11)$$

dimana:

Y_{Baru} : nilai Y data baru
Y_j : nilai Y ke j
$X_{iNormBaru}$: nilai normalisasi x_i data baru
X_{iNormj} : nilai normalisasi x_i data ke j

Dari Tabel 7, jarak terdekat dengan data baru adalah data ke 2, sehingga dengan Persamaan (11) diperoleh nilai estimasi dari Y_{baru} sebagai berikut

Y_{baru} = (0,250 / 0,375) * 150
= 100

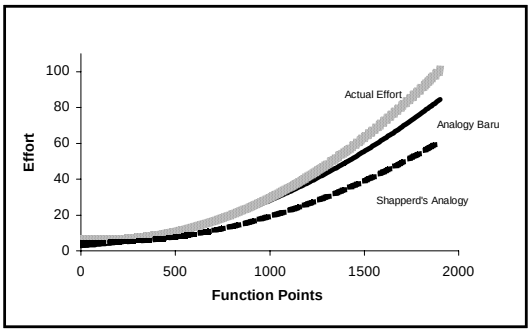
Modifikasi metode *Analogy* ini mampu mereduksi nilai *MMRE*, sehingga akurasi estimasi dapat ditingkatkan. Perbandingan nilai *MMRE* yang dihasilkan oleh metode yang dikembangkan oleh Shepperd dan metode yang telah dimodifikasi dapat dilihat pada Tabel 8.

Tabel 8, diperoleh dengan cara melakukan uji coba untuk mengestimasi nilai *effort* dengan menggunakan *Albrecht dataset*, yang mempunyai data 10 proyek. Secara bergantian satu data diambil untuk dijadikan *test set* untuk diestimasi nilai *effort*-nya. *Residual Error* (RE) adalah selisih antara nilai estimasi *effort* dibandingkan dengan nilai aktualnya. *MMRE* adalah nilai rata-rata dari *RE*. Pada Tabel 8 dapat dilihat bahwa nilai *MMRE* dari metode yang telah dimodifikasi lebih kecil dari pada nilai *MMRE* metode Shepperd.

Tabel 8. Perbandingan nilai *MMRE* estimasi shepperd dengan Metode Analogy modifikasi.

No.	Actual Effort	Estimasi Shepperd	RE	Analogy Modifikasi	RE
1	102,4	61,2	40%	97,65	5%
2	105,2	61,2	42%	100,29	5%
3	11,1	8	28%	11,67	5%
4	20,1	12,9	39%	20,09	0%
5	28,8	17,3	36%	28,34	2%
6	10	6,1	39%	10,63	6%
7	8	2,9	64%	8,75	9%
8	4,9	4,1	16%	5,83	19%
9	12,9	20,1	64%	13,36	4%
10	19	12,9	32%	19,11	1%
		MMRE	64%	MMRE	17%

Perbandingan estimasi *effort* yang dihasilkan metode *Analogy* Shepperd dengan metode *Analogy* yang telah dimodifikasi (*Analogy* baru) ditunjukkan pada Gambar 4.



Gambar 4. Perbandingan *Effort* aktual dengan Metode *Analogy* Shepperd dan *Analogy* baru

Teknik Estimasi Biaya

Sebelum menentukan teknik estimasi biaya suatu proyek pengembangan perangkat lunak terlebih dulu dilakukan tahapan berikut :

1. Penentuan nilai *3D Function Points* (*FP*)
   Identifikasi fungsi-fungsi sebagai parameter proyek disesuaikan dengan permintaan pemakai antara lain: *outputs, inquiries, inputs, files, interfaces, transformations, transitions*.
   Setelah masing-masing fungsi dikelompokkan dan dihitung, kemudian diberi bobot sesuai dengan tingkat kompleksitasnya. Nilai total seluruh fungsi disebut nilai *Un-adjusted Function Points* (*UFP*). Kemudian identifikasi karakteristik aplikasi (faktor kompleksitas teknis).
   Nilai *FP* dihitung dengan mengalikan nilai *UFP* dan nilai faktor kompleksitas teknis (*Adjusted Factor/AF*) .
   Selanjutnya nilai *FP* yang telah diketahui dapat dikonversi ke jumlah *Source Lines of Code* (*SLOC*) yang ekivalen. Konversi dapat dilakukan dengan menggunakan Tabel ekivalensi bahasa pemrograman (Tabel 9).
2. Kalkulasi penggunaan kembali (*reuse*) perangkat lunak yang ada dan komponen-komponen serta pustaka komersial [9,10].

Tabel 9. Lines of code per function points berdasarkan bahasa pemrograman [13]

Bahasa	SLOC per Function Points
C++ default	53
Cobol default	107
Delphi 5	17
HTML	14
Visual Basic 6	24
SQL default	13
Java 2 default	46

3. Estimasi *effort* (*E*) dengan menggunakan metode *Analogy* yang telah dimodifikasi.
4. Penentuan waktu yang diperlukan (t_d)
   Setelah nilai *effort* didapatkan, waktu yang diperlukan dapat dihitung dengan menggunakan Persamaan (12).
   $$t_d = SM * (E)^{0,33} \quad (12)$$
   dimana:
   t_d : waktu yang diperlukan (*months*)
   SM : Schedule Multiple yang dapat dilihat nilainya pada Tabel 10
   E : *effort*

Tabel 10. Faktor untuk Konversi Effort [12]

Project Type	Schedule Multiple
COCOMO II Default	3.67
Embedded Development	4.00
E-Commerce Development	3.19
Web Development	3.10
Military Development	3.80

5. Penentuan biaya proyek
   Biaya proyek dapat dihitung dengan Persamaan (13) dan (14).
   $$BiayaProduksi = B_{FS}+B_{FD}+B_{MB}+B_T+B_M+B_D \quad (13)$$
   $$Estimasi\ biaya = BiayaProduksi*(1+Pajak) \quad (14)$$
   dimana:
   B_{FS} : biaya studi kelayakan
   B_{FD} : biaya desain fungsi
   B_{MB} : biaya pemrograman
   B_T : biaya training
   B_M : biaya pemeliharaan
   B_D : biaya dokumentasi

- *Biaya Studi Kelayakan (*B_{FS}*)*
  Beberapa komponen yang mempengaruhi biaya aktifitas studi kelayakan antara lain:
  - Waktu untuk studi kelayakan (t_{Feas})
    $$t_{Feas} = t_d / 4 \quad (15)$$
  - *Effort* untuk studi kelayakan (E_{Feas})
    $$E_{Feas} = MP_{Feas} * t_d/4 \quad (16)$$
    MP_{Feas} : jumlah orang untuk studi kelayakan
  - Biaya tenaga kerja untuk studi kelayakan (C_{FS})
    $$C_{FS} = E_{Feas} * UR \quad (17)$$
    UR : Upah Regional
  - Biaya listrik untuk studi kelayakan (C_{LFs}) dapat diperoleh dengan Persamaan,
    $$C_{LFs} = E_{Feas} * L_{Rp} \quad (18)$$
    $$L_{Rp} = L_{Komp} * J_H * H_B \quad (19)$$
    dimana:
    L_{Komp} : ongkos listrik per unit komputer lengkap
    L_{Rp} : ongkos listrik per unit komputer per bulan
    J_H : jumlah jam kerja per hari
    H_B : jumlah hari kerja per bulan
  - Biaya konsumsi untuk studi kelayakan (C_{KFs})
    $$C_{KFs} = E_{Feas} * K_{Rp} \quad (20)$$

$$K_{Rp} = K_H * H_B \quad (21)$$

dimana:

K_{Rp} : biaya konsumsi per orang per bulan
K_H : biaya konsumsi per orang per hari
H_B : jumlah hari kerja per bulan

- Biaya overhead untuk studi kelayakan (BO_{Fs})

$$BO_{Fs} = C_{Gfs}+C_{Dfs}+C_{Tfs}+C_{Afs} \quad (22)$$
$$C_{Gfs} = E_{Feas} * G_{Rp} \quad (23)$$
$$C_{Dfs} = E_{Feas} * D_{Rp} \quad (24)$$
$$C_{Tfs} = E_{Feas} * T_{Rp} \quad (25)$$
$$C_{Afs} = E_{Feas} * A_{Rp} \quad (26)$$

dimana:

C_{Gfs} : biaya gedung dan listrik
C_{Dfs} : biaya depresiasi mesin
C_{Tfs} : biaya telpon
C_{Afs} : biaya asuransi
G_{Rp} : biaya sewa gedung dan ongkos listrik per orang per bulan
D_{Rp} : biaya depresiasi mesin per bulan
T_{Rp} : biaya telpon per orang per bulan
A_{Rp} : biaya asuransi per orang per bulan

$$D_{Rp} = H_{Komp} / (U_D * 12) \quad (27)$$

dimana:

H_{Komp} : biaya satu unit komputer lengkap
U_D : umur depresiasi mesin (dalam tahun)

Jadi biaya total studi kelayakan adalah

$$B_{FS} = C_{FS} + C_{LFs} + C_{KFs} + BO_{Fs} \quad (28)$$

- Biaya Desain Fungsi (B_{FD})

Beberapa komponen yang mempengaruhi biaya aktifitas desain fungsi antara lain:

- Waktu yang diperlukan untuk desain fungsi (t_{FD})
$$t_{FD} = t_d / 3 \quad (29)$$
- *Effort* yang diperlukan untuk desain fungsi (E_{FD}) sebanding dengan estimasi *effort*; sesuai dengan SLOC [6].
- Jumlah orang yang diperlukan untuk desain fungsi (MP_{FD})
$$MP_{FD} = E_{FD} / t_{FD} \quad (30)$$
- Biaya tenaga kerja untuk desain fungsi (C_{FD})
$$C_{FD} = E_{FD} * UR \quad (31)$$
- Biaya listrik untuk desain fungsi (C_{LFD}) dapat diperoleh dengan Persamaan (17).
- Biaya konsumsi untuk desain fungsi (C_{KFD}) dapat diperoleh dengan Persamaan (20).
- Biaya overhead untuk desain fungsi (BO_{FD}) dapat diperoleh dengan Persamaan (22).

Jadi biaya total desain fungsi adalah

$$B_{FD} = C_{FD} + C_{LFD} + C_{KFD} + BO_{FD} \quad (32)$$

- Biaya Pemrograman & Implementasi (B_{MB})

Beberapa komponen yang mempengaruhi biaya aktifitas pemrograman antara lain:

- Jumlah orang yang diperlukan untuk pemrograman (MP)
$$MP = E / t_d \quad (33)$$
- Biaya tenaga kerja untuk pemrograman (C_{MB})
$$C_{MB} = UR * E \quad (34)$$
- Biaya listrik untuk pemrograman (C_{LMB}) dapat diperoleh dengan Persamaan (18).
- Biaya konsumsi untuk pemrograman (C_{KMB}) dapat diperoleh dengan Persamaan (20).
- Biaya overhead untuk pemrograman (BO_{MB}) dapat diperoleh dengan Persamaan (22).

Jadi biaya total pemrograman adalah

$$B_{MB} = C_{MB}+C_{LMB}+C_{KMB}+BO_{MB} \quad (35)$$

- Biaya Training (B_T)

Beberapa komponen yang mempengaruhi biaya aktifitas training antara lain:

- *Effort* yang diperlukan untuk training (E_T) *Effort* yang diperlukan untuk training diasumsikan sama dengan *effort* untuk pemrograman.
$$E_T = E \quad (36)$$
- Waktu yang diperlukan untuk training (t_T)
- Jumlah orang yang diperlukan untuk training (MP_T)
$$MP_T = E_T / t_T \quad (37)$$
- Biaya tenaga kerja untuk training (C_T) dapat diperoleh dengan menggunakan Persamaan (34) dengan penyesuaian nilai E_T dan t_T.
- Biaya listrik untuk training (C_{LT}) dapat diperoleh dengan Persamaan (18).
- Biaya konsumsi untuk training (C_{KT}) dapat diperoleh dengan Persamaan (20).
- Biaya *overhead* untuk training (BO_T) dapat diperoleh dengan Persamaan (22).

Jadi biaya total desain fungsi adalah

$$B_T = C_T+C_{LT}+C_{KT}+BO_T \quad (38)$$

- Biaya Pemeliharaan (B_M)

Beberapa komponen yang mempengaruhi biaya aktifitas pemeliharaan antara lain:

- *Effort* yang diperlukan untuk pemeliharaan (E_M) *Effort* yang diperlukan untuk pemeliharaan diasumsikan sama dengan 15% dari *effort* untuk pemrograman sebab *effort* untuk pemeliharaan mempunyai range antara 12 sampai 17 persen dari *effort* pengembangan [12].
$$E_M = 0{,}15 * E \quad (39)$$
- Waktu yang diperlukan untuk pemeliharaan (t_M)
$$t_M = t_d$$
- Jumlah orang yang diperlukan untuk pemeliharaan (MP_M)
$$MP_M = E_M / t_M \quad (40)$$
- Biaya tenaga kerja untuk pemeliharaan (C_M) dapat diperoleh dengan Persamaan (35) dengan penyesuaian nilai E_M dan t_M.

- Biaya listrik untuk pemeliharaan (C_{LM}) dapat diperoleh dengan Persamaan (18).
- Biaya konsumsi untuk pemeliharaan (C_{KM}) dapat diperoleh dengan Persamaan (20).
- Biaya overhead untuk pemeliharaan (BO_M) dapat diperoleh dengan Persamaan (22).

Jadi biaya total desain fungsi adalah

$$B_M = C_M + C_{LM} + C_{KM} + BO_M \quad (41)$$

- Biaya Dokumentasi (B_D)

Jumlah halaman dokumentasi dari suatu proyek dapat diprediksi dengan menggunakan *effort* pengembangan sebagai input [10]. Persamaan yang digunakan untuk mendapatkan jumlah halaman dokumentasi (*Pages*) tersebut adalah:

$$Pages = \sum(A_i + B_i E^{Ci}) \quad (42)$$

dimana

A, B, C : konstanta penentuan jumlah halaman
i : macam dokumen yang harus dibuat.
Doc_{Rp} : biaya pembuatan dokumentasi per halaman

Jadi biaya dokumentasi proyek adalah

$$B_D = Doc_{Rp} * Pages \quad (43)$$

Dataset yang dipergunakan sebagai acuan untuk mengestimasi *effort* adalah *Albrecht Dataset*, sebagaimana yang dipergunakan Shepperd dalam studi kasusnya [11]. Untuk menyesuaikan dengan *3D Function Points*, ditambahkan dua kolom dalam dataset, yaitu kolom *transformations* dan *transitions* (10), dan data untuk kolom tersebut diisi dengan nilai nol karena untuk aplikasi sistem informasi secara umum nilai *Function Points* sama dengan nilai *3D Function Points*.

Tabel 11 adalah dataset Albrecht dan nilai maksimum sementaranya ada pada Tabel 12. Hasil normalisasinya dapat dilihat pada Tabel 13, sedangkan Tabel 14 dan 15 adalah nilai maksimum dan normalisasi akhir dari dataset Albrecht.

Tabel 11. *Albrecht dataset*

Actual Effort	Input	Output	File	Inquiries	Transformations	Transitions	Function Points	SLOC
4,1	7	12	8	13	0	0	199	40
0,5	15	15	3	6	0	0	199	3
6,1	12	15	15		0	0	260	29
4,9	17	17	5	15	0	0	289	30
10	13	19	23		0	0	283	28
8	34	14	5		0	0	195	35
2,9	33	17	5	8	0	0	224	22
8,9	27	19	6	24	0	0	400	52
3,6	25	28	22	4	0	0	500	15
7,5	28	41	11	16	0	0	417	24

Tabel 12 Nilai maksimum sementara *Albrecht dataset*

Actual Effort	Input	Output	File	Inquiries	Transformations	Transitions	Function Points	SLOC
105,2	193,0	150,0	60,0	75,0	0,0	0,0	1.902,0	317,0

Tabel 13. Normalisasi sementara *Albrecht dataset*

Actual Effort	Input	Output	File	Inquiries	Transformations	Transitions	Function Points	SLOC
0,04	0,04	0,08	0,13	0,17	1,00	1,00	0,11	0,13
0,00	0,08	0,10	0,05	0,08	1,00	1,00	0,10	0,01
0,06	0,06	0,10	0,25	0,00	1,00	1,00	0,14	0,09
0,05	0,09	0,11	0,08	0,19	1,00	1,00	0,15	0,09
0,09	0,07	0,13	0,38	0,00	1,00	1,00	0,15	0,09
0,08	0,17	0,09	0,08	0,00	1,00	1,00	0,11	0,11
0,03	0,17	0,11	0,08	0,11	1,00	1,00	0,12	0,07
0,08	0,14	0,13	0,10	0,32	1,00	1,00	0,20	0,16
0,03	0,13	0,19	0,37	0,05	1,00	1,00	0,26	0,05
0,07	0,14	0,27	0,17	0,20	1,00	1,00	0,22	0,07

Tabel 14 Nilai maksimum *Albrecht dataset*

Actual Effort	Input	Output	File	Inquiries	Transformations	Transitions	Function Points	SLOC
105,2	98341,74	76431,4	30572,56	38205,7	5,095427	5,095427	969150,2	161934,6

Tabel 15. Normalisasi *Albrecht dataset*

Actual Effort	Input	Output	File	Inquiries	Transformations	Transitions	Function Points	SLOC
0,0489	0,0001	0,0003	0,0004	0,0006	0,0020	0,0020	0,0004	0,0004
0,0047	0,0003	0,0003	0,0002	0,0003	0,0020	0,0020	0,0003	0,0000
0,0579	0,0002	0,0003	0,0008	0,0000	0,0020	0,0020	0,0004	0,0003
0,0466	0,0002	0,0004	0,0003	0,0007	0,0020	0,0020	0,0005	0,0003
0,0950	0,0002	0,0004	0,0013	0,0000	0,0020	0,0020	0,0005	0,0003
0,0760	0,0006	0,0003	0,0003	0,0000	0,0020	0,0020	0,0004	0,0004
0,0276	0,0006	0,0004	0,0003	0,0004	0,0020	0,0020	0,0004	0,0002
0,0846	0,0005	0,0004	0,0003	0,0011	0,0020	0,0020	0,0007	0,0005
0,0342	0,0004	0,0006	0,0012	0,0002	0,0020	0,0020	0,0008	0,0001
0,0713	0,0005	0,0009	0,0006	0,0007	0,0020	0,0020	0,0007	0,0002

Tabel 16. Studi kasus PON 1999

Business Function	Jumlah	Kompleksitas	Bobot	UFP
OUTPUT	69	sederhana	4	276
	39	rata-rata	5	195
	0	kompleks	7	0
SUB TOTAL:	108			461
INQUIRIES	140	sederhana	3	420
	0	rata-rata	4	0
	0	kompleks	6	0
SUBTOTAL:	140			420
INPUT	71	sederhana	3	203
	1	rata-rata	4	4
	0	kompleks	6	0
SUBTOTAL:	72			207
FILE	68	sederhana	7	476
	5	rata-rata	10	50
	0	kompleks	15	0
SUBTOTAL:	73			526
INTERFACE	47	sederhana	5	235
	0	rata-rata	7	0
	0	kompleks	10	0
SUBTOTAL:	47			235
TRANSFORMATION	0	sederhana	7	0
	0	rata-rata	10	0
	0	kompleks	15	0
SUBTOTAL:	0			0
TRANSITION	0	sederhana	N/A	0
	0	rata-rata	3	0
	0	kompleks	N/A	0
SUBTOTAL:	0			0
TOTAL UNADJUSTED FUNCTION POINTS (UFP):			1769	

Tabel 17. Kompleksitas Teknis

Komponen Teknis	Nilai
Data communication	5
Distributed function	5
Performance	5
Heavily used configuration	0
Transaction rates	4
Online data entry	5
Design for user efficiency	2
Online update	4
Complex processing	1
Reusability	3
Installation ease	2
Operational ease	5
Multiple sites	5
Facilitate change	4
Total Komponen Teknis *(∑Fi)*	**50**

Jumlah *Function Points* (FP) dapat dihitung sebagai berikut:

$$AF = 0{,}65 + 0{,}01\ (\sum Fi)$$
$$= 0{,}65 + 0{,}01\ (50)$$
$$= 1{,}15$$

$$FP = UFP \times AF$$
$$= 1769 \times 1{,}15$$
$$= 2049$$

menurut Tabel 11, kalau digunakan bahasa pemrograman VB, rata-rata *source lines per function points*-nya adalah 53. Jadi kalau terdapat *function points* sebanyak 75, maka rata-rata *source lines of code* (SLOC) adalah :

$$SLOC = 2049 * 53 = 114000$$

Dengan demikian data parameter yang dimiliki oleh sistem PON 1999 ini adalah :

Output	= 108
Inquiry	= 140
Input	= 72
File	= 73
Interface	= 47
Transformation	= 0
Transition	= 0
FP	*= 2049*
SLOC	*= 114000*

Tabel 18. Hasil estimasi biaya

Komponen Biaya	Estimasi	Nilai Aktual	Prosentase Perbedaan
Jumlah Halaman Dokumen	2.198		
Waktu Studi Kelayakan (bulan)	2,96		
Jumlah SDM Studi Kelayakan	1		
Waktu Desain Fungsi (bulan)	3,72		
Jumlah SDM Desain Fungsi	7		
Waktu Programming & Implementasi (bulan)	11,83		
Jumlah SDM Programming & Implementasi	11		
Waktu Training (bulan)	0,45		
Jumlah SDM Training	1		
Waktu Pemeliharaan (bulan)	1		
Jumlah SDM Pemeliharaan	2		
Biaya Studi Kelayakan	5.711.346,95	7.000.000,00	17,41%
Biaya Desain Fungsi	32.044.942,68		
Biaya Pemrograman & Implementasi	147.204.063,04		
Biaya Pemeliharaan	423.404,04		
Biaya Training	1.138.520,98		
Biaya Dokumentasi	2.197.151,22		
Sub total :	**178.729.429,91**		
Pajak 10 %	17.872.942,99		
Total Biaya Proyek :	**197.602.372,91**	**190.000.000,00**	**3,80%**

Kesimpulan

Studi ini telah menunjukkan bahwa modifikasi metode *Analogy* menghasilkan estimasi biaya proyek pengembangan perangkat lunak lebih akurat dibandingkan hasil dari metode *Analogy* Shepperd standard. Perbaikan metode *Analogy* dilakukan ditahap-tahap berikut:

- Perbaikan metode pengukuran jarak untuk menentukan proyek serupa sebagai acuan estimasi *effort*.
- Akurasi estimasi *effort* ditingkatkan dengan memasukkan parameter lebih lengkap.
- Akurasi estimasi biaya pengembangan perangkat lunak ditingkatkan dengan menambah kelengkapan komponen biaya.

Daftar Acuan

[1] S.L. Pfleeger, Software Engineering Theory and Practice, Prentice-Hall Inc., 1998, p.98.

[2] M.J. Shepperd, C. Schofield, IEEE Transaction on Software Enginering Vol. 23, 1997, p. 736.

[3] R. Meli, Risks, Requirements and Estimation of a Software Project, *http://www.escom.co.uk /conference1999/meli.pdf*, 1999.

[4] R.S. Pressman, Software Engineering, McGraw-Hill, Boston, 1997, p.741.

[5] International Function Points Users Group, *http://www.ifpug.org*, 1999.

[6] I. Sommerville, Software Engineering, 5th.ed., Addison-Wesley Publishing Company, Wokingham, 1996, p.590.

[7] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Prentice Hall, Upper Saddle, New Jersey, 2002, p.580.

[8] Gallery Math Online, Analytic geometry 2, *http://www.univie.ac.at/future.media/moe/galerie/geom2/geom2.html*, 1988.

[9] W. Roetzheim, Creating the Project Plan, *http://www.sdmagazine.com*, 2001.

[10] W. Roetzheim, Estimating Software Costs, *http://www.sdmagazine.com/documents/s=821/sdm0010d/*, 2000.

[11] M. Shepperd, The NAME Project Non-Algorithmic Methods of Estimating, *http://dec.bournemouth.ac.uk/staff/decind22/web/NAME.html*, 2000.