PERBANDINGAN MODEL ALORITMIK DAN NON ALGORITMIK UNTUK ESTIMASI BIAYA PERANGKAT LUNAK

Martini Ganantowe Bintiri[1], Azhari SN[2], Rocky Y. Dillak[3]
[1]Jurusan Teknik Sipil Fakultas Teknik Universitas Sintuwu Maroso
Jl. Pulau Timor No. 1 Poso Sulawesi Tengah 94615
[2]Jurusan Ilmu Komputer Fakultas MIPA Universitas Gadjah Mada
Bulaksumur, Yogyakarta 55281, Telp (0274) 588688, Fax (0274) 565223
[3]Jurusan Teknik Informatika, AMIKOM
Jl. Ring Road Utara Condong Catur, Sleman-Yogyakarta
E-mail: gana75mart@yahoo.com, arisn.softcom@yahoo.com, rocky_dillak@yahoo.com

ABSTRAK
Pemilihan metode estimasi biaya adalah proses yang sangat penting dalam kesuksesan pelaksanaan suatu proyek perangkat lunak. Hal ini dikarenakan kesesuaian penggunaan metode dengan proyek yang akan dikerjakan dapat menghasilkan estimasi biaya yang akurat dan dapat dipercaya. Estimasi biaya merupakan bagian penting dalam tahap pengembangan perangkat lunak dimana manajer dapat menentukan besarnya biaya, waktu dan usaha yang harus dikeluarkan atas pelaksaaan sebuah proyek. Untuk dapat memilih metode estimasi yang sesuai dengan proyek yang akan dikerjakan diperlukan pemahaman yang jelas tentang metode-metode estimasi biaya yang ada salah satunya mengetahui kelemahan dan kelebihan dari masing-masing metode tersebut. Dalam penelitian ini dikaji dua kelompok besar metode estimasi biaya perangkat lunak yakni model algoritmik dan non algoritmik untuk membandingkan kelebihan dan kekurangan beberapa metode yang sering digunakan dalam kedua kelompok model tersebut, sehingga dapat menjadi rekomendasi bagi para manajer serta pembaca dalam melakukan penelitian tentang estimasi biaya perangkat lunak maupun dalam pemilihan metode estimasi.

Kata kunci: perbandingan, estimasi biaya perangkat lunak, model algoritmik, model non algoritmik

1. PENDAHULUAN

Estimasi biaya perangkat lunak adalah proses yang sangat penting dalam pengembangan perangkat lunak. Estimasi biaya perangkat lunak sangat penting untuk mengontrol dan mengatur efisiensi pada seluruh proses yang dilakukan dalam pengembangan perangkat lunak. Estimasi biaya perangkat lunak sangat dibutuhkan dalam membuat usulan proposal, negosiasi kontrak, penjadwalan, kontrol dan monitoring. Untuk itu akurasi biaya perangkat lunak sangat dibutuhkan karena (Leung & Fan, 2001) :

a. Dapat membantu untuk mengklasifikasi dan menetukan prioritas pengembangan proyek sehubungan dengan perencanaan bisnis secara keseluruhan.
b. Dapat digunakan untuk menentukan *resources* yang sesuai dengan proyek yang dikerjakan dan seberapa baik *resources* ini akan digunakan.
c. Dapat digunakan untuk menilai pengaruh terhadap perubahan dan dukungan dalam perencanaan ulang.
d. Dapat memudahkan untuk mengatur dan mengontrol proyek ketika *resources* benar-benar sesuai dengan yang dibutuhkan.
e. Dapat memenuhi harapan c*ustomer* agar biaya pengembangan yang aktual dapat sejalan dengan biaya pengembangan yang diestimasi.

Umumnya, estimasi *effort* dan biaya sangat sulit dilakukan dalam proyek perangkat lunak karena proyek perangkat lunak bersifat dinamis dan kesulitan menemukan proyek yang sangat mirip dengan proyek tersebut sebelumnya. Selain itu, sulit untuk menentukan metode estimasi yang cocok untuk proyek tersebut.

Penelitian tentang kemampuan dan pentingnya metode-metode estimasi biaya serta pengaruhnya terhadap kesuksesan proyek sangat dibutuhkan dalam proyek perangkat lunak agar manajer proyek perangkat lunak mengetahui dan yakin dengan kemampuan dari metode-metode tersebut, sehingga para manajer dapat memilih metode mana yang akan digunakan untuk estimasi biaya bagi proyek perangkat lunak yang akan dikerjakan (Khatibi & Jawawi, 2010).

Tujuan utama dari penelitian ini adalah mengkaji kemampuan dari beberapa metode estimasi biaya perangkat lunak dan membandingkan metode-metode tersebut sehingga dapat menjadi referensi yang tepat dalam pemilihan metode estimasi biaya perangkat lunak. Penelitian ini ditulis dengan sistematika sebagai berikut : bagian kedua, setelah pendahulan, menggambarkan secara singkat tentang estimasi biaya perangkat lunak. Bagian ketiga menggambarkan perbandingan dari metode-metode yang ada, dan terakhir, kesimpulan dan saran diilustrasikan pada bagian empat.

2. TEKNIK-TEKNIK ESTIMASI

Secara umum, terdapat banyak metode estimasi biaya perangkat lunak. Metode-metode tersebut dibagi dalam dua kelompok : Model Algoritmik dan Non Algoritmik. Dalam bagian ini, dijelaskan beberapa metode estimasi biaya perangkat lunak berdasarkan kelompok model algoritmik dan non algoritmik.

2.1 Model Algoritmik

Model Algoritmik didasarkan pada model matematika yang menghasilkan estimasi biaya sebagai fungsi dari sejumlah variabel, yang dianggap sebagai *cost factor* yang utama. *Cost factor* adalah komponen-komponen biaya yang diperhitungkan dalam *cost estimation model*. Semua model algoritmik menggunakan model persamaan sebagai berikut (Leung & Fan, 2001) :

$$Effort = f(x1, x2, \ldots\ldots, x_n) \quad (1)$$

Dimana, (x1, x2,.....x_n) adalah vektor dari *cost factor*.

Perbedaan antara metode-metode algoritmik yang ada yaitu dalam memilih *cost factor* dan fungsi. Berikut adalah *cost factor* yang digunakan dalam model algoritmik (Khatibi & Jawawi, 2010) :

- Product factors
- Computer factors
- Personnel factors
- Project factors

Menghitung *cost factors* sangat sulit untuk dilakukan bahkan beberapa *cost factor* diabaikan dalam beberapa proyek perangkat lunak (Khatibi & Jawawi, 2010). Hal ini karena komponen-komponen tersebut sangat bergantung dari jenis dan lingkungan implementasi suatu proyek. Dalam penelitian ini dipilih beberapa metode algoritmik yang dianggap popular dan telah dipublikasikan dalam beberapa paper dalam-tahun terakhir ini (Ali et al., 2010; Bia, Munoz, & Gomez, 2011; Jeng, Yeh, Wang, Chu, & Chen, 2011; Khatibi & Jawawi, 2011; Tan & Zao, 2006; Zia, Rashid, & Zaman, 2011).

2.1.1 Model CoCoMo II

CoCoMo II dikembangkan dari CoCoMo terdahulu, dimana model ini terdiri dari beberapa sub model (Sommerville, 2011) :

a. An Aplication-compotition Model

Model ini dipakai untuk estimasi effort dalam pengembangan proyek prototipe dan proyek yang dikomposisi dari beberapa komponen. Rumus untuk menghitung *effort* bagi *prototype system* :

$$PM = (NAP\ X\ (1\text{-}\%reuse/100)) / PROD \quad (2)$$

Dimana PM adalah estimasi *effort* dalam *person-month*, NAP adalah jumlah total *application point*, *%reuse* adalah jumlah estimasi dari *reused code* dalam pengembangan system, PROD adalah *subapplication point productivity* dalam satuan NAP/*month*.

b. The Early Design Model

Model ini digunakan pada tahap awal pengembangan sebuah proyek, sebelum desain arsitektur system secara detail dibuat. Rumus untuk menghitung *effort*-nya :

$$PM = 2.94\ x\ Size^{(1.1\ -\ 1.24)}\ x\ M \quad (3)$$

$$M = PERS\ x\ RCPX\ x\ RUSE\ x\ PDIF\ x\ PREX\ x\ FCIL\ x\ SCED \quad (4)$$

M adalah faktor pengali dari 7 atribut yang mempengaruhi hasil estimasi. Atribut yang digunakan dalam *early design model* adalah *product reliability and complexity* (RCPX), *reuse required* (RUSE), *platform difficulty* (PDIF), *personal capability* (PERS), *personal experience* (PREX), *schedule* (SCED) dan *support facilities* (FCIL). Nilai dari atribut-atribut ini menggunakan skala 6 dimana 1 = sangat rendah dan 6 = sangat tinggi.

c. The Reuse Model

Model ini digunakan untuk estimasi *effort* yang diperlukan untuk mengintegrasikan *reusable* atau *generated code* dengan system yang ada. Rumus yang digunakan adalah :

$$ESLOC = ASLOC\ x(1\text{-}AT/100)\ x\ AAM \quad (5)$$

Dimana ESLOC adalah banyak *line of code* yang sama dengan *reusable code*, ASLOC adalah jumlah *line of code* dari *reusable code* termasuk kode yang *degenerate* otomatis, AT (*adaptate code*) adalah persentase kode yang diadopsi dari *reusable code* yang di *generate*.

d. The Post-Architecture Level

Digunakan untuk estimasi *effort* bagi sistem setelah desain arsitektur awal telah dibuat, jadi struktur subsistem diketahui. Dengan demikian maka hasil estimasi perangkat lunak dalam sub model ini diharapkan lebih akurat.

Persamaan yang dipakai sama dengan persamaan yang dipakai pada *early desain model* sebagai berikut :

$$PM = A\ x\ Size^{B}\ x\ M \quad (6)$$

Dimana A = 2.94, B = 1.17, *Size = jumlah line of code* dalam KSLOC, M = PERS x RCPX x RUSE x PDIF x PREX x FCIL x SCED.

2.1.2 Model Dicomo

Digitization cost model (DiCoMo) adalah sebuah model estimasi proyek perangkat lunak untuk *digital content production* (Bia, 2006). Metode ini adalah metode estimasi biaya yang mempunyai persamaan

estimasi mirip dengan *intermediate CoCoMo*, tetapi dengan beberapa perbedaan (Bia, Munoz, & Gomez, 2011) :

- *Size-independent overhead* (SIO), merupakan waktu untuk persiapan sebelum melakukan digitasi.
- *The size is known beforehand*, merupakan jumlah halaman (P) dari dokumen yang akan didigitasi.
- *Time is cost*, keseluruhan waktu yang digunakan untuk proses digitasi, yang secara langsung dikonversi menjadi biaya perangkat lunak menggunakan beberapa *cost factor* (jumlah uang yang dibayarkan perjam dalam setiap pekerjaan).

Formula dasar DiCoMo untuk menghitung waktu dalam *T* :

$$T = a.P^b + SIO \quad (7)$$

Dimana T adalah waktu yang dipakai untuk menyelesaikan tugas, P adalah jumlah dokumen yang didigitasi, nilai a dan b diperoleh dengan menyesuaikan kurva dengan data historis menggunakan metode *least-square*, SIO adalah waktu persiapan sebelum proses digitasi *content.*

2.1.3 Model Entity Relationship

Model entity relationship adalah model yang dikembangkan dari diagram *entity relationship* untuk estimasi ukuran perangkat lunak menggunakan regresi linear berganda (Tan & Zhao, 2006).

Entity Relationship (ER) diagram adalah sebuah *graph* tidak berarah yang dengan mudah dapat digambar menjadi *graph* berarah dengan memandang *entity* sebagai *vertex* dan *relationship* sebagai *edge* dari *graph*. Menurut Ali et al. (2010) *effort* perangkat lunak dipengaruhi oleh karakteristik dari ER diagram yang digunakan sebagai *metric* dalam model estimasi :

- NOE : jumlah *entity* dalam ER diagram.
- NOR : jumlah *relationships* dalam ER diagram.
- NOA : jumlah atribut dalam ER diagram.
- NOP : jumlah kompleksitas *path* dalam ER diagram.

Krakteristik Diagram ER dimanfaatkan sebagai *metric* melalui regresi linear berganda diperoleh persamaan model ER sebagai berikut :

$$Y = \beta_0 + \beta_1 NOE + \beta_2 NOR - \beta_3 NOA - \beta_4 NOP \quad (8)$$

Dimana β_i ($1< i \leq 4$) adalah koofisien yang akan ditentukan (Tan & Zhao, 2006).

$$NOP = \sum i \in G \ \ Pi \quad (9)$$

Dimana NOP adalah kompleksitas *path* dalam ER diagram, Pi adalah kompleksitas *path* dalam *vertex*, i adalah jumlah *path* dari vertex yang saling terhubung dengan *vertex* lainnya dalam *graph* yang sama yang dihitung menggunakan algoritma untuk *search path* (Ali, dkk. 2011).

$$Pi = \sum_{i \neq j} l/n \sum_{i \neq j} l_{ij} \quad (10)$$

Dimana vertex ke i ≠ *vertex* ke j, Pi adalah kompleksitas *path* dari *vertex* ke i, n jumlah path yang dilalui *vertex* ke i yang dapat mengakses *vertex* ke j.

2.1.4 Function Point

Fuction Point (FP) *metric* semula dikembangkan sebagai alternative SLOC untuk mengukur produktivitas pengembangan perangkat lunak untuk tahap selanjutnya (Zia, dkk. 2011). Namun, menurut Albrecht (1983) model FP juga dapat menjadi *tool* yang efektif untuk estimasi biaya pada tahap awal pengembangan perangkat lunak dari suatu siklus pengembangan perangkat lunak.

Untuk *FP metric* disediakan lima kategori dari f*unction count weighting factors* : *external input, external output, internal file, external interface, external inquiry* seperti ditunjukan pada Tabel 1.

Derajat kompleksitas fungsi untuk setiap indikator diberi nilai 1 untuk low, 2 untuk average, dan 3 untuk high sedangkan nilai bobot untuk masing-masing indikator antara 3 dan 15 (Khatibi & Jawawi, 2010).

Secara umum, persamaan untuk menghitung *unadjusted function point* (UFP) sebagai berikut :

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} W_{ij} X_{ij} \quad (11)$$

Dimana W_{ij} adalah *weight* untuk baris i, kolom j, dan Xij adalah *function count* dalam sel i,j.

Tabel 1. Function count weighting factors

	Low	*Average*	*High*
External input	__ x 3	__ x 4	__ x 6
External output	__ x 4	__ x 5	__ x 7
Internal file	__ x 7	__ x 10	__x 15
External interface	__ x 5	__ x 7	__ x 10
External inquiry	__ x 3	__ x 4	__ x 6

UFP kemudian digunakan untuk menghitung *adjusted Function Point* (AFP) dengan persamaan sebagai berikut :

$$AFP = UFP \ X \ VAF \quad (12)$$

Dimana VAF adalah *value adjustment factor* dihitung sebagai berikut :

$$VAF = 0.65 + 0.01 \sum_{i=1}^{14} c_i \quad (13)$$

Dimana c_i adalah jumlah *degree of influence* dari karakteristik system (C1-C14) seperti yang

ditunjukkan pada Tabel 2. Range VAF antara 0.65 jika seluruh c_i=0 dan 1.35 jika seluruh c_i=5 (Khatibi, & Jawawi, 2010).

Table 2. Technical complexity factor components

C_i	*Karakteristik*	*DI*	C_i	*Karakteristik*	*DI*
C1	*Data comunications*		C8	*Online update*	
C2	Distributed functions		C9	*Complex processing*	
C3	*Performance*		C10	*Reusability*	
C4	*Heavily used configuration*		C11	*Instalation easy*	
C5	*Transaction rate*		C12	*Operational easy*	
C6	*Online data entry*		C13	*Multiple sites*	
C7	*End user efficiency*		C14	*Facilitate change*	

DI adalah *degree of influence* yang nilainya ditentukan sebagai berikut (Albrecht, 1983) :
- *Not present, or no influence = 0*
- *Insignificant influence = 1*
- *Moderate influence = 2*
- *Average influence = 3*
- *Significant influence = 4*
- *Strong influence, throughout = 5*

2.2 Model Non Algoritmik

Model ini menggunakan informasi estimasi biaya dari proyek sebelumnya yang memiliki kemiripan dan proses estimasinya dilakukan dengan menganalisa dataset dari proyek sebelumnya.

Pada bagian ini dipilih tiga metode yang dianggap populer dan telah dipublikasikan dalam beberapa paper pada tahun-tahun terakhir ini (Attazadeh & Ow, 2009; Khatibi & Jawawi, 2011; Kaur, Singh, & Kahlon, 2008; Kusuma, Hermadi, & Ardiansyah, 2010; Malhotra & Jain, 2011; Setia, Hermadi, & Kusuma, 2008).

2.2.1 Analogy-Based Estimation

Metode ini memanfaatkan data proyek yang serupa yang telah diketahui biayanya, dan menggunakan biaya tersebut sebagai estimasi atas proyek yang baru (Kusuma, Hermadi, & Ardiansyah 2010). Metode analogy mempunyai prinsip dasar bahwa proyek yang serupa mempunyai biaya yang serupa, oleh karena itu dalam melakukan estimasi biaya sebuah proyek baru maka akan dicari proyek lama yang memiliki kesamaan yang tinggi (kedekatan jarak) dengan proyek yang baru menggunakan *similarity function*. Menurut Khatibi & Jawawi (2010) terdapat dua *similarity function* yang populer yang dapat digunakan dalam metode Analogy yaitu *Euclidean Similarity* (ES) dan *Manhattan Similarity* (MS). Sebagai contoh penelitian yang dilakukan oleh Kusuma, Hermadi, Ardiansyah (2010) menggunakan ES n dimensi sebagai *similarity function*, dimana setiap n dimensi mewakili satu atribut.

2.2.2 Expert Judgment

Metode ini didasarkan pada pendapat para pakar yang dikumpulkan dari para pakar yang berpengalaman dengan proyek yang mirip dengan proyek yang akan dikerjakan. Metode ini biasanya digunakan ketika data yang tersedia terbatas. Teknik yang umum digunakan dalam metode ini adalah teknik DELPH yang melibatkan berapa langkah sebagai berikut (Khatibi & Jawawi, 2010) :

a. Fomulir dibagikan kepada para pakar oleh koordinator.
b. Setiap pakar mengisi formulir tersebut menggunakan metode estimasi mereka (tidak diperbolehkan berdiskusi satu dengan yang lain)
c. Formulir dikumpulkan oleh koordinator lalu dihitung jumlah estimasi pada masing-masing formulir (Termasuk menghitung rata-rata dan median) dan proes estimasi dilanjutkan pada iterasi berikutnya.
d. Ulangi langkah b-c sampai diperoleh estimasi yang diharapkan atau disetujui.

2.2.3 Model Machine Learning

Metode *machine learning* dikategorikan kedalam dua metode utama yang dijelaskan sebagai berikut (Khatibi & Jawawi , 2010) :

a. Jaringan Saraf Tiruan

Jaringan saraf tiruan adalah suatu jaringan yang terdiri dari *neuron-neuron* yang saling terhubung oleh bobot. Jaringan saraf tiruan memiliki minimal tiga *layer* yang terdiri dari *layer input*, *hiden layer*, dan *layer output* . Data pertama kali diinput ke neuron-neuron pada *layer input* yang selanjutnya akan diteruskan ke *hiden layer* melalui bobot ke *neuron-neuron* di *hiden layer* seterusnya sampai pada *leyer output*.

Jaringan Saraf Tiruan (JST) Propagasi Balik merupakan metode yang sangat baik untuk masalah estimasi biaya perangkat lunak (Malhotra & Jain, 2011). Metode ini membutuhkan data pelatihan untuk mengajarkan pada JST agar menjadi pintar dan dapat melakukan estimasi biaya perangkat lunak.

b. Metode Fuzzy

Proses estimasi biaya menggunakan metode *fuzzy* adalah dengan mendefinisikan fungsi keanggotaan setiap variabel pada *cost driver* menggunakan prinsip *fuzzy* seperti yang disulkan oleh Zadeh (1996).

Setia, Hermadi, & Kusuma (2008) dalam penelitian estimasi biaya menggunakan *fuzzy* mengusulkan tidak semua faktor pada *Cost Driver* didefinisikan dalam *fuzzy zet* karena hanya merupakan deskripsi biasa. Faktor-faktor tersebut adalah RELY, CPLX, MODP, dan TOOL. Proses perhitungan estimasi biaya perangkat lunak menggunakan logika *fuzzy* sama dengan CoCoMo, tetapi *effort multipliers* yang digunakan berasal dari sebuah persamaan :

$$F\_C_{ij} = \sum_{j=1}^{k_i} \mu_{A_j}^{V_i}(P) x\ C_{ij} \quad (14)$$

Dimana F_C_{ij} adalah *effort multipliers* yang didapatkan dari himpunan *fuzzy*, $\mu_{A_j}^{V_i}$ adalah fungsi keanggotaan dari himpunan *fuzzy* A_j yang berasosiasi dengan *Cost Driver* V_i, P adalah urutan proyek dan K_i adalah jumlah *Cost Driver*.

3. PEMBAHASAN

3.1 Membandingkan Metode Estimasi

Berdasarkan pemaparan diatas, maka pada bagian ini dilakukan perbandingan metode berdasarkan kelebihan dan kekurangan seperti ditunjukkan pada Tabel 3 dan Tabel 4 sebagai berikut :

Tabel 3. Rincian perbandingan metode dalam model algoritmik

Metode	*Kelebihan*	*Kekurangan*
Cocomo II	Standar industry, Mudah menyediakan informasi yang mendalam, Proses kalibrasi jelas dan efektif	Didasarkan pada proses waterfall, Tidak semua *extensions* telah dikalibrasi;
DiCoMo	Ukuran pekerjaan diketahui terlebih dahulu, Mudah diaplikasikan pada proses *digital production* yang berbeda.	Hanya khusus digunakan untuk proyek digitasi, Hasil dipengaruhi oleh keahlian operator dan technology yang digunakan.
ER Model	Mudah diaplikasikan; Hasilnya akurat; Dilakukan pada tahap awal, Tidak bergantung pada bahasa pemrograman.	Tidak ada standar desain ER diagram sehingga desainnya bergantung pada desainer.
Function Point	Tidak bergantung pada bahasa pemrograman, Dilakukan pada tahap awal pengembangan proyek.	Mekanismenya sulit dilakukan.

Tabel 4. Rincian perbandingan metode dalam model non algoritmik

Metode	*Kelebihan*	*Kekurangan*
Expert Judgment	Pakar yang ahli dengan pengalaman yang relevan memberikan hasil estimasi yang akurat; Proses estimasi cepat.	Kesuksesan tergantung pada pakar; Memungkinkan hasil estimasi bias.
Analogy-Based Estimation	Berdasarkan pada data aktual dan data historis;	Proyek yang mirip sulit ditemukan; Data historis mungkin tidak akurat.
Jaringan Saraf Tiruan	Mampu belajar dari pengalaman/data historis	Tidak ada pedoman untuk mendesain arsitektur jaringannya; Kinerja bergantung pada besarnya data pelatihan
Fuzzy	Tidak membutuhkan perlatihan, *power of reasoning*	Sangat sulit menentukan derajat keyakinan *Fuzzy*

Sesuai dengan rincian perbandingan metode pada Tabel 3 dan Tabel 4 di atas dapat diketahui keuntungan dan kekurangan dari model algoritmik dan non-algoritmik seperti ditunjukkan pada Tabel 5.

Tabel 5. Ringkasan perbandingan model algoritmik dan non-algoritmik

Metode	*Kelebihan*	*Kekurangan*
Algoritmik	Hasil estimasi akurat dan dapat dipercaya, Dapat menghasilkan estimasi berulang, Mudah untuk modifikasi data *input* dan menyesuaikan dengan formula, Dapat digunakan pada proyek skala besar atau kecil.	Kalibrasi untuk proyek yang lalu mungkin tidak mencerminkan kondisi saat ini, C*ost factor* tidak mudah diukur, Ukuran input dan *rating cost driver* yang tidak akurat menyebabkan hasil yang tidak akurat.
Non Algoritmik	Proses estimasi cepat, Mudah dipelajari, Dapat melakukan estimasi berdasarkan data historis dan penalaran.	Sulit mencari data historis yang mirip dengan proyek baru, Data historis terkadang tidak akurat, Membutuhkan banyak ahli yang pengalaman dalam bidangnya, Sulit mendesain arsitekturnya, Efisien untuk proyek skala kecil atau *particular project*.

Berdasarkan hasil perbandingan yang dilakukan pada Tabel 3, Table 4, dan Tabel 5 di atas dapat diketahui bahwa tidak ada metode yang benar-benar baik atau sempurna, masing-masing metode memiliki kelebihan dan kekurangan. Sesuai keyataan bahwa masing-masing metode memiliki kelebihan dan kekurangan maka metode-metode tersebut dapat saling melengkapi satu dengan yang lain sehingga dalam penggunaannya dapat digabungkan untuk menghasilkan estimasi yang lebih akurat dan dapat dipercaya.

Dalam memilih metode estimasi sebaiknya memperhatikan jenis, ukuran, dan kondisi proyek yang akan dikembangkan, ketersediaan data historis, ketersediaan pakar yang ahli dalam pengalaman yang relevan dengan proyek yang akan dikembangkan, kemiripan dari proyek yang akan dikembangkan dengan proyek sebelumnya, dll.

3.2 Bagaimana Mengevaluasi Hasil Dari Metode Estimasi

Proses evaluasi metode estimasi perangkat lunak adalah proses membandingkan akurasi dari biaya estimasi dengan biaya aktual dengan cara menghitung beberapa *metric* yang meliputi : RE

(Relative Error), MRE *(Magnitude of Relative Error)* dan MMRE *(Mean Magnitude of Relative Error)*. Persamaan *metric* tersebut sebagai berikut :

$$RE = (PM \text{ estimasi} - PM \text{ aktual}) / (PM \text{ aktual}) \quad (15)$$

$$MRE = |PM \text{ estimasi} - PM \text{ aktual}| / (PM \text{ aktual}) \quad (16)$$

$$MMRE = \sum MRE \quad (17)$$

Parameter lain yang digunakan untuk evaluasi kinerja adalah *PRED(Percentage of the Prediction)* dengan persamaan sebagai berikut :

$$PRED(X) = A / N \quad (18)$$

Dimana, *A* adalah jumlah proyek dengan *MRE* kurang dari atau sama dengan X dan *N* adalah jumlah dari proyek yang telah dikerjakan.

4. KESIMPULAN DAN SARAN

4.1 Kesimpulan

Berdasarkan pemaparan di atas maka disimpulkan bahwa penelitian mengenai perbandingan model algoritmik dan non-algoritmik sangat penting karena dapat menjadi referensi bagi pembaca maupun manajer proyek perangkat lunak dalam memilih metode yang sesuai dengan kebutuhan proyek. Dalam memilih metode sebaiknya dipertimbangkan beberapa hal : kelebihan dan kekurangan dari metode estimasi, ukuran, jenis, dan tipe proyek, tersedianya pakar yang ahli dalam pengalamannya yang relevan dengan proyek yang akan dikerjakan, ketersediaan data historis. Jika proyek yang dikerjakan meiliki data historis yang lengkap dan proyek berukuran kecil dapat menggunakan model non algoritmik dan jika tidak ada data historis dan proyek berukuran besar dapat digunakan model algoritmik.

4.2 Saran

Setiap manajer proyek perangkat lunak harus memiliki pemahaman prinsip dari berbagai metode estimasi biaya perangkat lunak termasuk kelebihan dan kekurangannya karena dapat menjadi referensi yang baik dalam memilih metode yang sesuai. Untuk itu perlu dilakukan banyak penelitian terhadap metode-metode estimasi yang ada baik model algoritmik maupun non-algoritmik dengan melakukan perbandingan, kolaborasi ataupun *improvement* agar dapat diperoleh metode estimasi yang lebih baik.

Mengingat adanya kelebihan dan kekurangan dari masing-masing metode yang ada maka disarankan untuk memakai lebih dari satu metode dalam melakukan estimasi dan membandingkan hasilnya agar diperoleh hasil estimasi yang akurat dan dapat dipercaya.

PUSTAKA

Albrecht, J., A., & Gaffney, E., J. (1983). Software function, source lines of code, and development effort prediction – A software science validation, *IEEE transactions on software engineering*, 1-17

Ali., A., Qadri, S., Muhammad, S., S., Abbas, J., Pervaiz, T.,M., & Awan, S. (2010). Software Cost Estimation through Entity Ralationship Model. *Journal of American Science, 6*(11), 47-51.

Attarzadeh, I., & Ow, H., S. (2009) Proposing a New High Performance Model for Software Cost Estimation. *Second International Conference on Computer and Electrical Engineering*, 112-116.

Bia, A. (2006). Estimating Digitization Costs in Digital Libraries Using Software Engineering Methods, *Centro de Investigacion Operative*, 1-16.

Bia, A., Munoz, R., & Gomez, J. (2010). DiCoMo : the digitization cost model. *Int J Digit Libr*, 141-153.

Jeng, D.,Yeh, D.,Wang, D., Chu,L., S., & Chen, M., C. (2011). A Specific Effort Estimation Method Using Function Point. *Journal of Information Science and Engineering 27*, 1363-1376.

Kaur, J., Singh, S., Kahlon, S., K. (2008). Comparative Analysis of the Software Effort Estimation Models. *Proceeding od World Academy of Science. Engineering and Technplogy*. *36*, 485-487.

Khatibi, V., & Jawawi, A.,N. (2010). Software Cost Estimation Methods : A Review. *Journal of Emerging Trends in Computingand Information Science*, *2*(1), 21-29.

Kusuma, A., W., Hermadi, I., & Ardiansyah. (2010). Analisis Metode Analogy Based Estimation untuk Perkiraan Biaya Perangkat Lunak. *Jurnal Ilmiah Ilmu Komputer*, 1-9.

Leung, H., & Fan, Z. (2001). *Software Cost Estimation*, Departemen of Computing, The Hongkong Polytechnic University, 1- 14.

Malhotra, R., & Jain., A. (2011) Software Effort Prediction using Statistical and Machine Learning Methods. *(IJACSA) International Journal of Advanced Computer Science and Applications, 2*(1), 145-151.

Setia, D., Hermadi, I., M., & Kusuma, A., W. (2008). Perkiraan Biaya Perangkat Lunak Menggunakan Logika Fuzzy. Diakses pada 28 maret 2011 dari http://repository.ipb.ac.id.

Sommerville, L. (2011). Estimation techniques, Software Engineering, nint edition, 633-645.

Tan, K., B., H., & Zhao, Y. (2006). Sizing data-Intensive System from ER Model. *IEICE Trans. Inf. & Syst.*, *89*(4), 1321-1326.

Zia, Z., Rashid, A., & uz Zaman, K. (2011). Software Cost Estimation for component-based fourt-generation-language software applications. *IET Software*, *5*(1), 103-110.

Zadeh, L. (1996). Fuzzy Logic : Computing With Words. *IEEE Transactions on Fuzzy System, 4*(2), 103-111.