Perbandingan Hasil Analisis Aplikasi dengan Metode COCOMO I dan II

Habib Muhammad Baqir, Anita Hidayati
Teknik Informatika dan Komputer
Politeknik Negeri Jakarta,
Depok, Indonesia
muhbaq@gmail.com, anita.hidayati@tik.pnj.ac.id

Diterima: 7 September 2015. Disetujui: 9 Oktober 2015. Dipublikasikan: November 2015

Abstrak - Ada beberapa metode yang sudah dikembangkan untuk melakukan perkiraan upaya yang dibutuhkan oleh seseorang atau sebuah tim dalam mengembangkan sebuah perangkat lunak. Diantaranya adalah metode analisis *Function Point*, *Line of Code*, dan *Constructive Cost Model* (COCOMO). Artikel ini berfokus kepada model COCOMO I dan II. Untuk mencari perbedaan COCOMO I dan II, dilakukan estimasi pada sebuah aplikasi .NET yang menggunakan *Windows Forms*. Pada COCOMO I, terjadi peningkatan hampir 100% terhadap usaha yang dibutuhkan untuk mengembangkan aplikasi yang dilakukan di fase *basic* dan *intermediate*. Hal tersebut karena *cost driver* yang mempengaruhi tingkat usaha penyelesaian dapat dikenali sehingga dapat mengubah perencanaan untuk meningkatkan pemanfaatan sumber daya yang telah diberikan. Pada COCOMO II fase *Application Composition*, hasil estimasi yang didapatkan tidak berbeda jauh dengan hasil estimasi dengan menggunakan *basic COCOMO I*, karena tidak memperhitungkan faktor atau hal-hal yang dapat mempengaruhi hasil estimasi yang dilakukan pada fase lanjutan. Hasil estimasi fase *Early Design* mirip dengan hasil perhitungan *Intermediate COCOMO* karena ada tujuh *cost driver* baru yang teridentifikasi. Pada fase *Post-Architectural* hasil estimasi meningkat hampir dua kali lipat dari perhitungan fase sebelumnya. Ini menunjukkan aplikasi yang dianalisis membutuhkan usaha yang tinggi untuk melakukan pengembangan lebih lanjut.

Kata Kunci: COCOMO, COCOMO I, COCOMO II, model analisis, estimasi.

I. PENDAHULUAN

Pendorong dasar di dalam perancangan model analisis COCOMO pada tahun 1981 oleh Boehms adalah keinginan untuk mengetahui apa saja kebutuhan-kebutuhan (*cost*) yang diperlukan pada perancangan sebuah aplikasi sehingga orang-orang yang berkepentingan dapat mengalokasikan sumber daya yang sesuai untuk menjamin berhasilnya sebuah proyek pengembangan perangkat lunak.

Seiring dengan berjalannya waktu, praktik-praktik pemrograman pasti mengalami perubahan yang mengakibatkan keteringgalan model lama sehingga estimasi dengan menggunakan model lama tersebut tidak dapat memberikan tingkat ketepatan yang tinggi. Oleh karena itu, COCOMO II dirancang untuk beradaptasi kepada perubahan praktik-praktik pemrograman dengan ikut mempertimbangkan hal-hal seperti *code reusability* [1], dimana banyak *developer* yang menggunakan kembali atau menyadur kode yang sudah pernah ditulis pada proyek sebelumnya untuk digunakan pada proyek baru tersebut sehingga mengurangi usaha yang dikeluarkan. Hal seperti perancangan arsitektur program juga dipertimbangkan, tidak seperti COCOMO I yang tidak memasukkan *code maintenance* pada estimasi usaha tersebut.

II. TINJAUAN PUSTAKA

Albakri & Qureshi [2] menyimpulkan bahwa COCOMO II dengan *cost driver* dan model yang lebih dipersiapkan untuk praktik *programming* modern dapat memberikan estimasi yang lebih baik daripada COCOMO I. Model analisis COCOMO I terdiri dari

tiga tingkatan sub-model: *Basic COCOMO*; *Intermediate COCOMO*; dan *Detailed COCOMO*.

Basic COCOMO digunakan untuk menghitung upaya dan biaya yang diperlukan dengan menggunakan ukuran program [3]. *Basic COCOMO* biasanya digunakan untuk melakukan perhitungan biaya cepat, tetapi kurang detail karena tidak memperhitungkan hal-hal subyektif seperti kemampuan personil. Besarnya sebuah program ditentukan dengan seribu baris kode yang ditulis (*KLOC*). *COCOMO* sendiri memiliki tiga mode proyek yang dapat dianalisis, yaitu *Organic*, *Semi-Detached*, dan *Embedded* [4].

Organic COCOMO biasanya digunakan pada sebuah proyek yang memiliki skala kecil, dikerjakan oleh tim pengembang berukuran kecil yang memiliki pengalaman di dalam lingkungan pengembangan perangkat lunak tersebut. Proyek tersebut biasanya memiliki *KLOC* sebesar 2 sampai 50. *Semi-Embedded COCOMO* merupakan mode *COCOMO* yang digunakan pada proyek skala sedang yang memiliki tim berukuran sedang dan memiliki pengalaman sistem yang berbeda-beda pada lingkungan pengembangan tersebut. Biasanya proyek tersebut berukuran 50 sampai 300 *KLOC*. *Embedded COCOMO* digunakan untuk proyek-proyek skala besar yang membutuhkan personil dan sumber daya banyak. Aplikasi yang dirancang biasanya memiliki lebih dari 300 *KLOC*. *Basic COCOMO* biasanya hanya digunakan untuk perhitungan cepat perancangan perangkat lunak, tetapi masih belum memperhitungkan hal-hal lain seperti keterbatasan perangkat keras, kualitas personil, dan lain-lain.

Intermediate COCOMO memiliki komponen yang sama dengan *basic COCOMO* tetapi ditambahkan dengan sebuah set *cost driver*, yang merupakan sebuah penilaian subyektif dari produk, perangkat keras, personil, dan atribut proyek. Setiap *cost driver* tersebut memiliki sub-Set atribut yang ada. *Detailed COCOMO* merupakan *sub-model COCOMO* ketiga, yang memiliki semua karakteristik *Intermediate COCOMO* dengan menambahkan penilaian dampak *cost driver* pada fase-fase perancangan [5].

COCOMO II memiliki empat *sub-model*, yaitu: *Application Composition*, *Early Design*, *Reuse*, *Post Architectural* [6]. *Sub-Model Application Composition* adalah perhitungan upaya yang diperlukan untuk melakukan *prototipe* sistem menggunakan skrip, pemrograman basis data. *Sub-Model Early Design*

menghitung upaya yang diperlukan dengan menggunakan kebutuhan sistem dan keputusan-keputusan rancangan. *Sub-Model Reuse* menghitung upaya untuk mengintegrasikan komponen yang dapat digunakan kembali. *Sub-Model Post Architectural* menghitung upaya yang diperlukan dengan kebutuhan rancangan sistem.

III. RANCANGAN ANALISIS

Pada artikel ini analisis dilakukan pada aplikasi *.NET* yang memiliki 25 komponen. Dengan dilakukannya analisis menggunakan model-model estimasi *COCOMO*, bisa dicari tahu model *COCOMO* apa yang lebih tepat untuk melakukan analisis estimasi aplikasi *.NET*. Mode *COCOMO I* yang digunakan pada analisis ini adalah *Organic COCOMO*, karena proyek yang dianalisis merupakan proyek kecil yang memiliki *KLOC* kurang dari 50. Untuk model *COCOMO II*, sub-Model yang akan digunakan adalah keempat sub-Model yang ada untuk menggambarkan setiap fase pengembangan aplikasi.

IV. ANALISIS DENGAN COCOMO I

Sub-Model yang digunakan: *Basic COCOMO I*

Mode yang digunakan: *Organic*

Rumus perhitungan:

$$Effort Applied (E): E = 2.4 \times (KLOC)^{1.05}$$

$$Development Time (D): D = 2.5 \times (E)^{0.38}$$

$$People Required (T): T = \frac{E}{D} \quad (1)$$

Total perhitungan LOC, 25 komponen terdapat pada Tabel 1.

TABEL 1. JUMLAH LOC 25 KOMPONEN

Nama Komponen	Lines of Code
About	239
Admin	767
Circle	7
Config	6
FormAttendance	491
FormContract	1471
FormEditProfile	303
FormEmployee	1378
FormInventory	917
FormLogin	294
FormMainMenu	447
FormPayroll	1362
FormPettyCash	258
FormUserMainMenu	749

Hardworker	28
HardWorkerEvent	4
Input	7
Program	3
Settings	22
SplashScreen	116
Tools	1005
ToolsPingPong	128
ToolsSnake	168
ToolsTicTacToe	296
User	1
TOTAL	10467 LOC (10.467 KLOC)

Perhitungan untuk *Effort Applied* (E):

$$E = 2.4 \times (10.467)^{1.05}$$

$$E = 28.25 \text{ MM (Man Month)}$$

Development Time (D):

$$D = 2.5 \times (28.25)^{0.38}$$

$$D = 8.898 \text{ Month}$$

People Required (T):

$$T = 28.25 / 8.898$$

$$T = 3.175$$

$$T = 3 \text{ People}$$

Sub-Model yang digunakan: *Intermediate COCOMO I*

Mode yang digunakan: *Organic*

Rumus yang digunakan:

$$E = EAF \times 3.2 \times (KLOC)^{1.05}$$

$$D = 2.5 \times (E)^{0.38} \quad (2)$$

Dimana EAF (*Effort Adjustment Factor*) adalah hasil perkalian nilai faktor-faktor *cost driver*.

Cost driver:

1. *Product Attributes*:
 - RELY – *Required Software Reliability*
 - DATA – *Database Size*
 - CPLX – *Product Complexity*
2. *Computer Attributes*
 - TIME – *Execution Time*
 - STOR – *Main Storage*
 - VIRT – *Virtual Machine Volatility*
 - TURN – *Computer Turnaround Time*
3. *Personal Attributes*
 - ACAP – *Analyst Capability*
 - AEXP – *Application Experience*
 - PCAP – *Programmers Capability*

- VEXP – *Virtual Machine Experience*
- LEXP – *Programming Language Experience*

4. *Project Attributes*

- MODP – *Usage of Modern Programming Practices*
- TOOL – *Use of Software Tool*
- SCED – *Required Development Schedule*

TABEL 2. ESTIMASI NILAI COST DRIVER

	<i>Very Low</i>	<i>Low</i>	<i>Nominal</i>	<i>High</i>	<i>Very High</i>	<i>Extra High</i>
RELY					1.4	
DATA		0.94				
CPLX		0.85				
TIME				1.11		
STOR			1.00			
VIRT		0.87				
TURN			1.00			
ACAP		1.19				
AEXP			1.00			
DCAP	1.21					
VEXP		1.1				
LEXP			1.00			
MODP				0.91		
TOOL					0.83	
SCED		1.08				

Nilai *cost driver* yang ada pada Tabel 2 akan dikalikan untuk menghasilkan nilai EAF yang akan dimasukkan kepada perhitungan *cost Intermediate COCOMO*.

$$EAF = 1.4 \times 0.94 \times 0.85 \times 1.11 \times 1.00 \times 0.87 \times 1 \times 1.19 \times 1.00 \times 1.21 \times 1.1 \times 1.00 \times 0.91 \times 0.83 \times 1.08$$

$$EAF = 1.395$$

Perhitungan untuk *Effort Applied* (E):

$$E = 1.395 \times 3.2 \times (10.467)^{1.05}$$

$$E = 52.545 \text{ Man Month}$$

Development Time (D):

$$D = 2.5 \times (52.545)^{0.38}$$

$$D = 11.265 \text{ Month}$$

People Required (T):

$$T = \frac{52.545}{11.265}$$

$$T = 4.664 \text{ People}$$

$$T = 4 \text{ atau } 5 \text{ People}$$

Sub-Model Detailed COCOMO I tidak digunakan pada estimasi aplikasi ini karena Detailed COCOMO I biasanya digunakan untuk proyek besar dan tidak memenuhi persyaratan proyek yang bisa mengikuti proses rekayasa perangkat lunak Barry.

V. ANALISIS DENGAN COCOMO II

Sub-Model yang digunakan: *Application Composition*. Rumus yang digunakan:

$$\text{Object Point (OP)} = EC \times CF \quad (3)$$

EC: *Estimated Count*, CF: *Complexity Factor*

$$\text{New Object Point (NOP)} = \frac{OP}{\text{Productivity}}$$

$$\text{Cost (NOP)} = \frac{\text{Labor Rate}}{\text{Productivity}}$$

Total Project Cost =

$$\frac{\text{Cost}}{\text{NOP}} \times \text{NOP of current project} \quad (4)$$

TABEL 3. PARAMETER PROYEK

	EC	CF		
		Simple	Average	Complex
Screens	22	1	2	3
Reports	9	2	5	8
Components	15	1	1	10
Productivity	13			

Perhitungan berdasarkan Tabel 3 seperti berikut:

$$\text{Object Point (OP)} = (22 \times 3) + (9 \times 8) + (15 \times 1) = 153$$

Aplikasi .NET yang ada menggunakan 30% kode yang sudah ada pada proyek sebelumnya.

$$\text{New Object Point (NOP)} = 153 \times \frac{100 - 30}{100}$$

$$\text{NOP} = 107.1 \approx 107 \text{ NOP}$$

$$\text{Effort (Man Month)} = \frac{107}{13}$$

$$\text{Effort} = 8.2307 \approx 8 \text{ Man Month}$$

Telah diasumsikan bahwa bayaran rata-rata untuk seorang *programmer* adalah 1500 USD.

$$\text{Cost (NOP)} = \frac{1500}{13}$$

$$\text{Cost (NOP)} = 115 \text{ USD}$$

$$\text{Total Project Cost} = 115 \times 107$$

$$\text{Total Project Cost} = 11235 \text{ USD}$$

Sub-Model yang digunakan: *Early Design*. Rumus yang digunakan:

$$\text{Effort} = A \times \text{Size}^B \times M \quad (5)$$

Dimana:

A = 2.94, sebagaimana yang sudah diusulkan oleh Boehm.

Size adalah KSLOC.

B adalah koefisien peningkatan usaha yang dibutuhkan dengan berkembangnya proyek yang dikerjakan, dengan rentang nilai dari 1.1 sampai 1.24. M adalah koefisien yang merupakan hasil perkalian dari tujuh *cost driver* yang diasosiasikan dengan Sub-Model *Early Design* seperti pada Tabel 4.

Cost Drivers:

1. RCPX – *Product Reliability and Complexity*
2. RUSE – *Reuse Required*
3. PDIF – *Platform Difficulty*
4. PERS – *Personnel Capability*
5. PREX – *Personnel Experience*
6. SCED – *Schedule*
7. FCIL – *Support Facilities*

TABEL 4. HASIL COST DRIVER EARLY DESIGN

	Extra Low	Very Low	Low	Nominal	High
PERS				1	
RCPX			0.83		
RUSE				1	
PDIF				1	
PREX			1.22		
FCIL		1.30			
SCED		1.43			

Koefisien M merupakan hasil perkalian *multiplier* yang didapatkan dari *cost driver*.

$$M = 1 \times 0.83 \times 1 \times 1 \times 1.22 \times 1.30 \times 1.43 = 1.8824$$

Nilai B adalah menengah, sama dengan 1.15 karena aplikasi .NET yang dirancang akan mengalami perkembangan seiring dengan berjalannya waktu karena adanya penambahan jumlah *user* yang menggunakan *system* dan ukuran *database* yang meningkat. Maka,

$$\begin{aligned} \text{Effort}(\text{Man Month}) &= 2.94 \times (10.467)^{1.15} \times 1.8824 \\ &= 82.3862 \text{ Man Month} \end{aligned}$$

Perhitungan *Sub-Model Reuse* tidak digunakan di dalam artikel ini dikarenakan aplikasi .NET yang dianalisis tidak menggunakan kode yang diterjemahkan dari bahasa lain.

Sub-Model yang digunakan: *Post-Architectural*

$$\text{Effort} = A \times \text{Size}^B \times M \quad (6)$$

Model ini digunakan ketika proyek sudah mencapai tingkat *development* yang sudah mumpuni. Model *Post-Architectural* digunakan untuk *maintenance*. Hasil estimasi *cost driver* ditunjukkan pada Tabel 5.

Cost Driver detail:

1. *Product Factors*
 - a. RELY – *Required Software Reliability*
 - b. DATA – *Database Size*
 - c. CPLX – *Product Complexity*
 - d. RUSE – *Developed for Reusability*
 - e. DOCU – *Documentation Match to Life-Cycle Needs*
2. *Platform Factors*
 - a. TIME – *Execution Time Constraint*
 - b. STOR – *Main Storage Constraint*
 - c. PVOL – *Platform Volatility*
3. *Personnel Factor*
 - a. ACAP – *Analyst Capability*
 - b. PCAP – *Programmer Capability*
 - c. PCON – *Personnel Continuity*
 - d. APEX – *Applications Experience*
 - e. PLEX – *Platform Experience*
 - f. LTEX – *Language and Tool Experience*

4. Project Factors

- a. TOOL – *Use of Software Tools*
- b. SITE – *Multisite Development*
- c. SCED – *Required Development Schedule*

TABEL 5. HASIL ESTIMASI COST DRIVER

	Very Low	Low	Nominal	High	Very High
RELY			1.00		
DATA		0.90			
CPLX		0.87			
RUSE			1.00		
DOCU				1.11	
TIME			1.00		
STOR			1.00		
PVOL			1.00		
ACAP	1.42				
PCAP	1.34				
PCON				0.90	
APEX	1.22				
PLEX		1.09			
LTEX	1.20				
TOOL	1.17				
SITE				0.93	
SCED	1.43				

$$\begin{aligned} M &= 1.00 \times 0.90 \times 0.87 \times 1.00 \times 1.11 \times 1.00 \times 1.00 \\ &\quad \times 1.00 \times 1.42 \times 1.34 \times 0.90 \times 1.22 \\ &\quad \times 1.09 \times 1.20 \times 1.17 \times 0.93 \times 1.43 \\ M &= 3.6956 \end{aligned}$$

$$\begin{aligned} \text{Effort} &= 2.94 \times (10.467)^{1.15} \times 3.6956 \\ \text{Effort} &= 161.7438 \text{ Man Month} \end{aligned}$$

VI. KESIMPULAN

Pada model estimasi COCOMO I dapat dilihat bahwa pada model *Intermediate* yang menggunakan *cost driver* dapat memberikan gambaran yang lebih detail terhadap kondisi pengembangan proyek sehingga terjadi perubahan hasil estimasi. Pada fase *basic*, didapatkan bahwa usaha yang dibutuhkan untuk mengembangkan aplikasi tersebut adalah 28.25 *Man Month* dan ketika dilakukan estimasi yang lebih detail pada fase *intermediate*, ditemukan bahwa usaha yang dibutuhkan meningkat, menjadi 52.545 *Man Month*. Terjadi peningkatan sebesar 24.295 *Man Month* karena telah ditemukannya faktor-faktor yang mempengaruhi tingkat usaha yang dibutuhkan untuk menyelesaikan pengembangan proyek aplikasi.

Dapat dilihat pada fase *Intermediate*, *cost driver* yang mempengaruhi tingkat usaha penyelesaian

dapat dikenali sehingga desainer sistem dapat mengubah perencanaan untuk meningkatkan pemanfaatan sumber daya yang telah diberikan.

Setelah menerapkan model *COCOMO II* kepada estimasi biaya pengembangan sebuah aplikasi, penulis mendapati bahwa pada fase *Application Composition*, hasil estimasi yang didapatkan adalah 8 *Month*, hasil tersebut tidak berbeda jauh dengan hasil estimasi dengan menggunakan *basic COCOMO I*, karena kedua *Sub-Model* tersebut merupakan *Sub-Model* yang digunakan untuk melakukan estimasi biaya pengembangan kasar, tidak memperhitungkan faktor atau hal-hal yang dapat mempengaruhi hasil estimasi yang dilakukan pada fase lanjutan.

Pada fase lanjutan seperti *Early Design*, ada tujuh *cost driver* baru yang teridentifikasi yang sudah dituliskan sebelumnya yang merupakan simplifikasi dari *cost driver* yang ada pada fase *Post-Architectural* yang berjumlah 17 *cost driver* yang lebih mendetil.

Hasil estimasi fase *Early Design* adalah 82.3862 *Man Month*, dapat dilihat bahwa setelah faktor-faktor yang dapat mempengaruhi estimasi akan mempengaruhi hasil hingga meningkatkan hasil estimasi lebih dari sebelumnya, mirip dengan hasil perhitungan *Intermediate COCOMO*. Ketika perhitungan sudah mencapai fase *Post-Architectural*, fase dimana semua faktor *cost driver* sudah diketahui secara mendetil, hasil estimasi meningkat menjadi

161.7348 *Man Month*. Hampir dua kali lipat daripada perhitungan fase sebelumnya. Ini menunjukkan aplikasi yang dianalisis membutuhkan usaha yang tinggi untuk melakukan pengembangan lebih lanjut.

REFERENSI

- [1] Ratnesh Litoriya, Narendra Sharma, Abhay Kothari, "Incorporating Cost driver substitution to improve the effort using Agile COCOMO II", 2012 CSI Sixth International Conference on Software Engineering (CONSEG), 2012, Pages: 1-7.
- [2] Albakri, M. M., Qureshi M. R. J. "Empirical Estimation of COCOMO I and COCOMO II Using a Case Study." 2-5 (2010).
- [3] Richard E. (Dick) Fairley, "The influence of COCOMO on software engineering education and training", Journal of Systems and Software, Volume 80, Issue 8, August 2007, Pages 1201-1208.
- [4] Barry W. Boehm; Ricardo Valerdi, "Achievements and Challenges in Cocomo-Based Software Resource Estimation", IEEE Journals & Magazines, Vol. 25, Issue: 5, 2008, Pages: 74-83.
- [5] M. Madheswaran; D. Sivakumar, "Enhancement of prediction accuracy in COCOMO model for software project using neural network", Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2014, Pages: 1 - 5.
- [6] Al Yahya M., Ahmad R., Lee S., "Impact of CMMI Based-Software Process Maturity on COCOMO II's Effort Estimation", Intl. Arab Journal of IT Vol. 7, No. 2, 129-137 (2010).