STUDI KASUS PENGUKURAN SISTEM INFORMASI MENGGUNAKAN *FUNCTION POINT* (FP)

Dewi Khairani

Program Studi Teknik Informatika Fakultas Sains dan Teknologi
Dosen Teknik InformatikaUIN Syarif Hidayatullah Jakarta
Jl. Ir H. Juanda No.95 Ciputat 15412
Telp. (62-21) 7493606, 7493547 Fax.: (62-21) 7493315
dewi.khairani@uinjkt.ac.id

ABSTRAK

Pengukuran perangkat lunak, merupakan salah satu aktivitas yang dilakukan untuk menilai sebuah perangkat lunak menggunakan sebuah pendekatan kuantitatif. Banyaknya metode pengukuran dan belum ada standar dalam penilaian perangkat lunak menjadikan proses pengukuran ini sering menjadi proses yang diabaikan, meskipun perannya sangat strategis dalam pengembangan perangkat lunak. Tulisan ini bermaksud menjabarkan sebuah pembelajaran pengukuran perangkat lunak dengan menggunakan metode *Function Point* (FP) dengan studi kasus sistem informasi penerimaan mahasiswa baru http://spmb.uinjkt.ac.id di UIN Syarif Hidayatullah Jakarta untuk menilai *value* sistem informasi tersebut. *Function Point (*FP*)* merupakan salah satu metode kuatitatif yang banyak digunakan dalam pengukuran perangkat lunak karena memiliki set kapabilitas untuk memprediksi nilai perangkat lunak dari berbagai segi objek pengukuran. Hasil pengukuran dengan menggunakan FP dengan mempertimbangkan pembobotan kompleksitas berdasarkan 14 nilai karakteristik. Dalam penelitian ini, didapatkan nilai FP untuk Sistem Informasi Penerimaan Mahasiswa Baru adalah sebesar 108.12. Dalam Penggunaannya, FP dapat digunakan sebagai basis untuk menilai biaya hingga jumlah sumber daya yang dibutuhkan dalam pembuatan perangkat lunak yang dimaksud.

Kata Kunci: *pengukuran, Function Point, rekayasa, metrik*

I. PENDAHULUAN

Pengukuran perangkat lunak menjadi salah satu kebutuhan saat ini disaat meningkatnya pertumbuhan perangkat lunak yang beredar dan kemajuan akses terhadap internet ini membuat pembuat aplikasi *mobile* berlomba-lomba untuk membuat perangkat lunak yang dapat digunakan baik secara gratis ataupun berbayar pada perangkat komunikasi.

Pertumbuhan secara massif ini, membuat sebuah pengukuran perangkat lunak menjadi sangat perlu dilakukan sebagai sarana untuk menilai kualitas sebuah perangkat lunak. Walaupun hingga kini belum ada regulasi yang terstruktur dan dispesi-fikasikan untuk mengukur kualitas perangkat lunak yang beredar agar memenuhi standar kualitas tertentu, tidak membuat pengukuran perangkat lunak menjadi suatu hal yang kalah penting dibandingkan proses pembuatan dan pengujian perangkat lunak tersebut, karena fungsinya sangat krusial dalam penentuan kualitas perangkat lunak.

Kualitas perangkat lunak sendiri merupakan sebuah kombinasi perhitungan yang kompleks yang melibatkan beberapa variasi faktor dan kebutuhan dari pengguna yang bermacam-macam. Gabungan antara kebutuhan pengguna perangkat lunak dan faktor – faktor lain akan menghasilkan kualitas sebuah perangkat lunak [1].

Pengukuran perangkat lunak merupakan hal yang sangat mendasar dalam pembuatan perangkat lunak dan daur/siklus hidup perangkat lunak tersebut. Pengukuran perangkat lunak dan metrik telah banyak digunakan untuk memutuskan bentuk perangkat lunak versi selanjutnya, dalam bentuk pengurangan ataupun penambahan fitur produk. Selain itu Pengukuran Perangkat lunak juga dapat digunakan untuk mengukur faktor kebutuhan dan *budget* dalam sebuah proyek perangkat lunak, seperti sumber daya manusia dan biaya.

Tulisan ini mengupas metode pengukuran perangkat lunak dengan menggunakan *Function*

Penelitian ini dibiayai oleh Penelitian Dasar Pusat Penelitian dan Penulisan Tahun Anggaran 2015 UIN Syarif Hidayatullah Jakarta.

*Point*s (FP) pada Sistem Informasi penerimaan mahasiswa baru di Universitas Islam Negeri Syarif Hidayatullah Jakarta.

II. TINJAUAN PUSTAKA

2.1 Pengukuran Perangkat Lunak

Menurut *Institute of Electrical and Electronics Engineers* (IEEE), Pengukuran merupakan ukuran tingkat kuantitatif dari sebuah sistem, komponen, atau proses yang memiliki atribut tertentu. Sedangkan mengukur adalah mengindikasikan kuantitatif dari luasan, jumlah, dimensi, dan kapasitas. Pada dasarnya pengukuran merupakan kegiatan penentuan angka bagi suatu objek secara sistematik. Penentuan angka ini merupakan usaha untuk menggambarkan karakteristik suatu objek.

Setiap pengukuran yang dilakukan membutuhkan tersedianya suatu ukuran kuantitatif yang disebut metrik. Istilah ukuran, pengukuran, dan metrik sering digunakan secara bergantian. Pressman [2] sendiri membagi metrik ke dalam dua kategori seperti berikut:

1. *Direct Metric*, yaitu metrik yang berhubungan langsung dengan objek perangkat lunak seperti misalnya: perhitungan jumlah Line Of Code (LOC), kecepatan eksekusi, ukuran memori dan kesalahan yang ditemui dalam suatu periode waktu.
2. *Indirect Metric*, yaitu metrik yang didapatkan karena berhubungan dengan interaksi perangkat lunak denga lingkungannya, seperti: fungsionalitas, kualitas, efisiensi, reliabilitas, kompleksitas, reliabilitas, dan lain sebagainya.

Pengukuran secara langsung (*Direct Metric*) biasanya lebih mudah dilakukan karena hasil dapat diperoleh secara langsung. Sedangkan, pengukuran secara tidak langsung (*Indirect Metric*) harus melalui proses yang lebih kompleks dalam perhitungan dan pengumpulan informasi yang dibutuhkan.

2.2 *Line of Code*

Salah satu cara primitif yang digunakan dalam mengukur ukuran program adalah dengan menghitung baris kode (LOC), yaitu dalam satuan ribuan dari LOC (KLOC). Meskipun menghitung baris kode terdengar sederhana, setelah dipelajari lebih lanjut, terungkap bahwa ada beberapa masalah yang harus dijawab sebelum memulai untuk mengukur dengan menggunakan KLOC. Pertanyaan yang paling penting adalah yang berhubungan dengan definisi LOC yang sebenarnya.

Ada beberapa definisi yang digunakan untuk menjawab pertanyaan tersebut. Namun hal terpenting yang harus diperhatikan adalah bagaimana penggunaan metrik untuk mendefinisikan LOC ini dapat digunakan secara konsisten dan jika memungkinkan, dilakukan secara otomatis.

Pada implementasinya, metode pengukuran dengan menerapkan jumlah baris koding atau *Line of Codes* (LOC) yang dihasilkan oleh seorang *programmer* sering digunakan sebagai pedoman pengukuran besar/volume sebuah perangkat lunak dan terkadang, produktivitas. Hal ini tentu saja kadang menjadi tidak relevan, karena akan sangat bergantung kepada subjektivitas individu dalam menuliskan kode.

LOC juga merupakan sebuah output yang dapat diukur setelah tahap implementasi dalam sebuah daur hidup perangkat lunak, oleh karena itu, penggunaan LOC sebagai basis pengukuran, terbatas hanya pada '*after project*' dan tidak dapat digunakan sebagai bahan pertimbangan dalam perencanaan proyek perangkat lunak untuk menentukan sumber daya dan *budget* yang dibutuhkan.

Namun, di samping ambiguitas dan kerugian dalam penggunaannya, LOC berguna karena merupakanbanyaknya tersedia alat pendukung yang baik dan bahan referensi yang banyak dalam penggunaannya.

2.3. *Model COCOMO*

COCOMO adalah sebuah model yang didesain oleh Barry Boehm untuk memperoleh perkiraan dari jumlah orang-bulan yang diperlukan untuk mengembangkan suatu produk perangkat lunak. Satu hasil observasi yang paling penting dalam model ini adalah bahwa motivasi dari tiap orang yang terlibat ditempatkan sebagai titik berat. Hal ini menunjukkan bahwa kepemimpinan dan kerja sama tim merupakan sesuatu yang penting, namun demikian poin pada bagian ini sering diabaikan.

Model COCOMO dapat diaplikasikan dalam tiga tingkatan kelas:

1. Proyek organik, adalah proyek dengan ukuran relatif kecil, dengan anggota tim yang sudah berpengalaman, dan mampu bekerja pada permintaan yang relatif fleksibel.
2. Proyek sedang (*semi-dettached*), adalah proyek yang memiliki ukuran dan tingkat kerumitan yang sedang, dan tiap anggota tim memiliki tingkat keahlian yang berbeda.
3. Proyek terintegrasi (*Embedded*), adalah proyek yang dibangun dengan spesifikasi dan operasi yang ketat.

Model COCOMO

Type	Effort	Schedule
Organic	PM=2.4 (KDSI) $^{1.05}$	TD=2.5 (PM) $^{0.38}$
Semi-detached	PM=3.0 (KDSI) $^{1.12}$	TD=2.5 (PM) $^{0.35}$
Embedded	PM=3.6 (KDSI) $^{1.20}$	TD=2.5 (PM) $^{0.32}$

Dimana:
PM = *person-month* (man-month)
KDSI = *delivered source instructions*, dalam ribuan
TD = *number of months estimated for software development*

Hirarki model Boehm berbentuk sebagai berikut:

- Model 1: Model COCOMO Dasar menghitung usaha pengembangan perangkat lunak (dan biaya) sebagai fungsi dari ukuran prgram yang diekspresikan dalam baris kode yang diestimasi.
- Model 2: Model COCOMO *Intermediate* menghitung usaha pengembangan perangkat lunak sebagai fungsi ukuran program dan serangkaian "pengendali biaya" yang menyang-kut penilaian yang subyektif terhadap produk, perangkat keras personil, dan atribut proyek.
- Model 3: Model COCOMO *advanced* menghubungkan semua karakteristik versi intermediate dengan penilaian terhadap pengaruh pengendali biaya pada setiap langkah (analisis, perancangan, dan lain-lain) dari proses rekayasa perangkat lunak.

Pengembangan model COCOMO adalah dengan menambahkan atribut yang dapat menentukan jumlah biaya dan tenaga dalam pengembangan perangkat lunak, yang dijabarkan dalam kategori dan subkategori sebagai berikut:

1. Atribut produk
   a. Reliabilitas perangkat lunak yang diperlukan
   b. Ukuran basis data aplikasi
   c. Kompleksitas produk
2. Atribut perangkat keras
   a. Performa program ketika dijalankan
   b. Memori yang dipakai
   c. Stabilitas mesin virtual
   d.Waktu yang diperlukan untuk mengeksekusi perintah
3. Atribut Sumber Daya Manusia
   a. Kemampuan analisis
   b. Kemampuan ahli perangkat lunak
   c. Pengalaman membuat aplikasi
   d. Pengalaman menggunakan mesin virtual
   e. Pengalaman dalam menggunakan bahasa pemrograman
4. Atribut proyek
   a. Menggunakan perangkat lunak tambahan
   b. Metode rekayasa perangkat lunak
   c. Waktu yang diperlukan

Masing-masing subkategori diberi bobot antara 0 (sangat rendah) sampai 6 (sangat tinggi), dan kemudian dijumlahkan. Dari pengembangan ini diperoleh persamaan:

$$\mathbf{E=}\boldsymbol{ai}\mathbf{(KLOC)}\boldsymbol{(b)i}\mathbf{.EAF}$$

2.4 *Function Point*

Pendekatan dengan menggunakan *Function Point*ini mencoba untuk menghilangkan beberapa kelemahan dari LOC dengan menurunkan ukuran program tidak lagi dari banyaknya baris kode, melainkan ditentukan oleh fungsionalitas yang dirasakan oleh pengguna. Hal ini menyebabkan metrik ini dapat dikategorikan sebagai metrik yang independen dari bahasa pemrograman dan teknologi yang digunakan. Dengan demikian, FP dapat digunakan untuk menormalkan dan membandingkan hasil dari beberapa lingkungan perangkat lunak yang berbeda.

Selain itu, karena fungsi ini diturunkan dari spesifikasi, kita dimungkinkan untuk mendapatkan ukuran program sebelum proses *development* berlangsung. Perlu diingat meskipun, bahwa konversi antara FP dan LOC tidak bisa diharapkan menjadi linear, karena ukuran pelaksanaannya tidak hanya tergantung pada jumlah fungsi tetapi juga pada kompleksitasnya.

Function Point terdiri dari 5 buah [5] yaitu sebagai berikut:

- **Tipe Input**, merupakan *interface* yang melakukan pemasukan data ke aplikasi.
- **Tipe Output**, merupakan output yang dihasilkan aplikasi untuk pengguna/*user* yang dapat berupa laporan di-*print* atau yang ditampilkan pada layar.
- **Tipe Query/Search/View**, merupakan fungsi yang berkaitan dengan menindahan terhadap data yang tersimpan.
- **Tipe File/Tabel/*Database***, merupakan fungsi yang berkaitan dengan logic penyimpan data yang dapat berupa file atau semacam *database relational*.
- **Tipe Interface Eksternal**, merupakan fungsi yang berkaitan dengan komunikasi data pada perangkat/mesin yang lain.

Dalam perhitungan komponen pada *Function Point*, setiap tipe komponen tersebut diberikan bobot berdasarkan kompleksitasnya. Contoh pembobotan

yang tertera disesuaikan dengan *Function Point International User Group* (FPIUG).

Tabel 1. Tabel Pedoman Perhitungan Function Point

Tipe Komponen	Level Kompleksitas									TOTAL CFP
	Sederhana			Menengah			Kompleks			
	J	B	P	J	B	P	J	B	P	
Tipe Input	X	3	?	X	4	?	X	6	?	?
Tipe Output	X	4	?	X	5	?	X	7	?	?
Tipe Query/ Search/View	X	3	?	X	4	?	X	6	?	?
Tipe File/Table/ *Database*	X	7	?	X	10	?	X	15	?	?
Tipe Interface External	X	6	?	X	7	?	X	10	?	?
	TOTAL									?

2.1. Menghitung *Crude Function Points(*CFP*)*

Crude *Function Point*s (CFP) adalah untuk menghitung bobot nilai dari komponen-komponen *Function Point* yang dikaitkan dengan software yang akan dibuat.

Tabel 1 merupakan contoh formulir kosong yang dapat digunakan untuk menghitung bobot masing2 poin yang ada dalam sistem informasi.

2.2. Menghitung *Relative Complexity Adjustment Factor* (RCAF)

RCAF digunakan untuk menghitung bobot kompleksitas dari software berdasarkan 14 karakteristik.

Penilaian Komplesitas memiliki skala antara 0 dan 5

- 0 = Tidak Pengaruh
- 1 = Insidental
- 2 = Moderat
- 3 = Rata-rata
- 4 = Signifikan
- 5 = Essential

Berikut merupakan tabel yang dapat dijadikan pedoman dalam menghitung RCAF[6]:

Tabel 2. Contoh Tabel Perhitungan Function Point

NO	KARAKTERISTIK	BOBOT
1.	Tingkat kompleksitas Komunikasi Data	4
2.	Tingkat kompleksitas Pemrosesan Terdistribusi	2
3.	Tingkat kompleksitas Performance	3
4.	Tingkat kompleksitas Konfigurasi	3
5.	Tingkat Frekuensi Penggunaan *Software*	1
6.	Tingkat Frekuensii Input Data	3
7.	Tingkat Kemudaaan Pengunaan Bagi *User*	3
8.	Tingkat Frekuensi *Update* Data	2
9.	Tingkat Kompleksitas Prosesing Data	3
10.	Tingkat Kemungkinan Penggunaan Kembali/Reusable Kode Program	4
11.	11. Tingkat Kemudahaan Dalam Instalasi	4
12.	Tingkat Kemudahaan operasional *software* (*backup, recovery*, dsbny)	3
13.	Tingkat Software dibuat untuk multi organisasi/perusahaan/client	3
14.	Tingkat kompleksitas dalam mengikuti perubahaan/fleksibel	3
	TOTAL	**41**

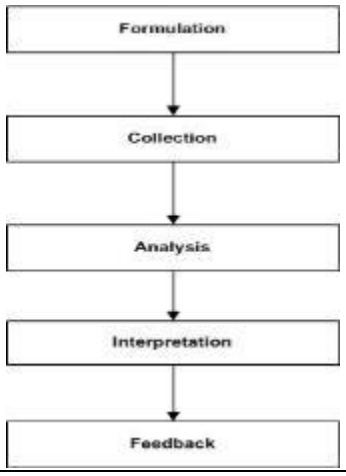
2.3. Menghitung *Function Point* (FP)

Selanjutnya untuk menghitung FP, maka digunakan rumus, sebagai berikut[6]:

$$FP = CFP \times (0.65 + 0.01 \times RCAF) \ldots\ldots (1)$$

Angka 0.65 dan 0.01 adalah ketetapan atau konstanta yang dibuat oleh *Function Point Internasional User Group* (IFPUG).

III. Metodologi Pengukuran



Gambar 1. Metode Pengukuran Menurut Roche

Prinsip dasar pengukuran menurut Roche[2], menunjukkan bahwa kegiatan pengukuran dapat dikategorikan berdasarkan lima kegiatan, di antaranya:

1. *Formulation* : menentukan cara perhitungan yang akan digunakan dalam mengukur suatu perangkat lunak dan metrik yang sesuai untuk diterapkan.
2. *Collection* : mekanisme yang digunakan untuk mengakumulasi data yang dibutuhkan untuk memperoleh perumusan metrik.
3. *Analysis* : perhitungan metrik dan penerapan rumus matematika.

4. *Interpretation* : evaluasi metrik yang menghasilkan pemahaman mengenai representasi kualitas.
5. *Feedback* : rekomendasi yang diperoleh dari interpretasi metrik sebuah produk yang ditransmisikan ke tim *software*.

Metode pengukuran ini akan penulis gunakan sebagai pedoman pengukuran perangkat lunak pada penelitian ini.

3.1 Prosedur dan Implementasi

3.1.1 *Formulation*

Dalam penelitian kali ini penulis menggunakan FP dalam pengukuran, untuk kasus yang berbeda, penulis telah melakukan penelitian dengan menggunakan metode GQM[4] sebagai metode pengukuran perangkat lunak.

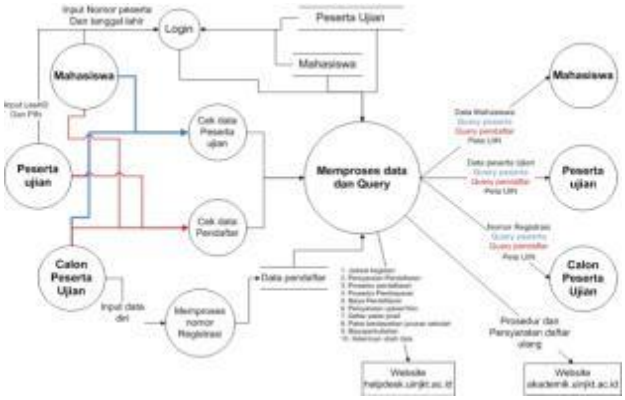
3.1.2 *Collection*

Dalam penelitian yang dilakukan kali ini, penulis menilai sebuah sistem informasi penerimaan mahasiswa baru di lingkungan UIN Syarif Hidayatullah http://spmb.uinjkt.ac.id. Berikut merupakan ruang lingkup dalam sistem informasi yang kami jadikan batasan dalam penelitian ini.

Gambar 2. Halaman Utama SPMB *Online* UIN Jakarta

Sistem ini digunakan sebagai objek penelitian mengingat fungsionalitasnya yang sangat terbatas dan ruang lingkup domain yang kecil, sehingga memudahkan penggunaannya sebagai bahan studi kasus.

Dari hasil analisa terhadap sistem web yang diekspos kepada publik, kami memetakan ruang lingkup fungsionalitas ke dalam alur proses yang dijabarkan dalam Gambar 3.



Gambar 2. Ruang Lingkup fungsionalitas Sistem Informasi pada Studi Kasus

Berdasarkan analisa fungsionalitas pada Sistem Informasi Akademik diperoleh data berikut dan dapat diturunkan dalam bentuk formulir pembobotan sebagai berikut:

Tabel 3. Tabel Pengklasifikasian Fungsionalitas untuk SI http://spmb.uinjkt.ac.id

Fungsionalitas	Tipe	Level Kompleksitas
Melihat data peserta ujian	Output	Sederhana
Melihat data pendaftar ujian	Output	Sederhana
Melihat File Peta UIN	View	Sederhana
Melihat Informasi Jadwal Kegiatan	View	Sederhana
Melihat Informasi Persyaratan Pendaftaran	View	Sederhana
Melihat Informasi Prosedur Pendaftaran	View	Sederhana
Melihat Informasi Prosedur Pembayaran	View	Sederhana
Melihat Informasi Biaya Pendaftaran	View	Sederhana
Melihat Informasi Pilihan Program Studi	View	Sederhana
Melihat Informasi Persyaratan Upload Foto	View	Sederhana
Melihat Informasi Pilihan Paket	View	Sederhana
Melihat Informasi Biaya Perkuliahan Mahasiswa Baru	View	Sederhana
Melihat Informasi Ketentuan Pengubahan Data	View	Sederhana
Melihat Informasi Prosedur dan Persyaratan Daftar Ulang	View	Sederhana
Melihat Informasi Biaya Perkuliahan Tahun 2015	View	Sederhana
Mencari Data Peserta Ujian	Query	Sederhana
Mencari Data Pendaftar SPMB	Query	Sederhana
Mencetak Kartu Ujian	Output	Menengah
Mendapat No. Registrasi (Generate 11 Digit Nomor)	Output	Menengah
Login Peserta SPMB	Query	Kompleksitas

Fungsionalitas	Tipe	Level Kompleksitas
Login Pendaftaran Ulang	Query	Menengah
Memasukkan Data Peserta Ujian	Input	Sederhana
Memasukkan Data Peserta SPMB	Input	Sederhana
Melihat Pilihan Paket Prodi A	Output	Sederhana
Melihat Pilihan Paket Prodi B	Output	Sederhana
Melihat Pilihan Paket Prodi C	Output	Sederhana
Melihat Alur Pendaftaran	Output	Sederhana
Pembayaran *Online*	Interface External	Menengah

3.1.3 *Analysis*

Selanjutnya, dari hasil pemetaan yang dilakukan, kita masukkan hasilnya ke dalam tabel perhitungan FP yang sebelumnya telah kita buat.

Dari penurunan Tabel 3, didapat perumusan berikut yang menghasilkan nilai *Crude* FP.

Tabel 4. Tabel Perhitungan Function Point untuk SI http://spmb.uinjkt.ac.id

Tipe Komponen	Level Kompleksitas									TOTAL CFP
	Sederhana			Menengah			Kompleks			
	J	B	P	J	B	P	J	B	P	
Tipe Input	2	3	6	0	4	0	0	6	0	6
Tipe Output	6	4	24	2	5	10	0	7	0	34
Tipe Query/Search/View	15	3	45	1	4	4	1	6	6	55
Tipe File/Table/ *Database*	0	7	0	0	10	0	0	15	0	0
Tipe Interface External	0	6	0	1	7	7	0	10	0	7
	TOTAL									102

Menghitung *Function Point*

Selanjutnya, setelah mendapatkan CFP, kita dapat memasukkan nilai tersebut dalam menghitung *Function Point*, kita menggunakan pedoman penilaian pada Tabel 3:

FP = CFP ×(0.65 + 0.01 x RCAF)
= 102× (0.65 + (0.01 x 41))
= 102 × (0.65 + 0.41)
= 102 × 1.06
= 108.12

Sehingga, nilai *Functional Point* yang didapat dari hasil perhitungan pada *website* SPMB *Online* UIN Syarif Hidayatullah Jakarta adalah **108.12**.

3.1.4 *Interpretation*

Dalam hal penggunaan FP, kita dapat menjadikannya sebagai salah satu pedoman untuk mendapatkan nilai metrik atas sesuatu dengan mengkonversikan FP ke dalam metrik yang diinginkan.

Pengkonversian Penggunaan *Function Point*

A. Biaya

Dimisalkan, setiap *Function Point*, dikenakan biaya Rp. 1.000.000/FP dan *Function Point* pada Sistem Informasi SPMB *Online* UIN Syarif Hidayatullah Jakarta adalah 108.12. Maka kita dapat diestimasikan biaya yang diperlukan untuk pembuatan Sistem Informasi yang dimaksud adalah:

Rp. 1.000.000 x 108.12 = Rp. 108.120.000

B. Mandays

Sedangkan untuk estimasi produksi, kita dapat menggunakan perhitungan seperti berikut:

2 Jam x **108.12** = 216 jam (pembulatan).
= 216/8
= 27*

*Angka ini sama dengan **27 hari kerja** (Asumsi dalam 1 hari bekerja selama 8 jam) .

C. Manajemen Proyek

FP juga dapat digunakan dalam perhitungan komponen dalam Manajemen Proyek, yang penjabaran dalam implementasinya ke anggaran SDM proyek, sebagaimana dijelaskan pada Tabel 5 di bawah berikut:

Tabel 5 Tabel Perhitungan Anggaran Proyek Sumber Daya Manusia

No	Jabatan	Jumlah Orang	Hari	Biaya/hari	Jumlah
1	Project Manager	1	27	Rp400,000	Rp10,800,000
2	System Analyst	1	27	Rp250,000	Rp6,750,000
3	Programmer	2	27	Rp200,000	Rp10,800,000
4	*Database* Admin	1	27	Rp250,000	Rp6,750,000
5	Graphic desainer	1	20	Rp250,000	Rp5,000,000
6	Tester	1	20	Rp200,000	Rp4,000,000
7	Trainer	1	5	Rp500,000	Rp2,500,000
TOTAL					**Rp46,600,000**

3.1.5 *Feedback*

Dalam halnya penyampaian *feedback*, atau umpan balik untuk penelitian ini, diberikan kepada pihak Pustipanda UIN Syarif Hidayatullah sebagai

pengelola sistem yang dijadikan sebagai objek penelitian.

IV. KESIMPULAN

Berdasarkan penelitian yang kami lakukan pada tulisan ini, kami menyimpulkan bahwa *Function Point* dapat digunakan untuk menilai/mengukur sebuah perangkat lunak. Dalam implementasinya, sebagai salah satu cara mengukur sebuah perangkat lunak, FP membutuhkan penilaian profesional karena melibatkan penilaian yang sangat subjektif. Untuk mengurangi faktor subjektifitas dalam penelitian ini, kami menggunakan perhitungan FP dengan menggunakan Pedoman dari *Function Point International User Group* (FPIUG). Dalam pengukuran Sistem Informasi SPMB *Online* UIN Syarif Hidayatullah Jakarta didapatkan nilai FP sebesar 108.12. Nilai ini kemudian dapat kita konversikan sehingga dapat diturunkan untuk beberapa fungsi pengukuran, seperti biaya dan waktu yang dibutuhkan untuk membangun perangkat lunak tersebut.

DAFTAR PUSTAKA

[1] Grady, Robert B., Practical Software Metrics for Project Management and Process Improvement, Prentice-Hall, Inc., 1992, ISBN: 0137203845

[2] Pressman, Roger S. 2010. "SOFTWARE ENGINEERING- A Practitioner's Approach - Seventh Edition", Penerbit McGraw-Hill Companies, New York: 889 hlm.

[3] DACS. 2005. "Software Acquistion Gold Practice Goal-Question-Metric (GQM) Approach", DACS Gold Practice Document Series GP-31 V 1.0

[4] Akbar, M.; Sukmana, H.T.; Khairani, D., "Models and software measurement using Goal/Question/Metric method and CMS Matrix parameter (Case study discussion forum)," in *Cyber and IT Service Management (CITSM), 2014 International Conference on* , vol., no., pp.34-38, 3-6 Nov. 2014 doi: 10.1109/CITSM.2014.7042171

[5] K.v.d. Berg, D. Ton, and O. Rogier, "Functional Size Measurement Applied to UML-base *User* Requirements", Retrievable from doc.utwente.nl, 2005

[6] Cah, "The *Function Point* Method", Pearson Education Limited, Retrievable from www.cs.nott.ac.uk, 2004

ACKNOWLEDGEMENT

Diberikan untuk seluruh peserta matakuliah *Software Metrics and Measurement* Tahun 2014-2015.