Scientific Journal of Informatics
Vol. 2, No. 1, Mei 2015
p-ISSN 2407-7658 http://journal.unnes.ac.id/nju/index.php/sji e-ISSN 2460-0040

Peningkatan Akurasi Estimasi Ukuran Perangkat Lunak dengan Menerapkan Logika Samar Metode *Mamdani*

Ristu Saptono[1], Galih Dian Hutama[2]

[1,2]Program Studi Informatika, FMIPA, Universitas Sebelas Maret Surakarta
Email: [1]ristu.saptono@staff.uns.ac.id, [2]galihdhutama@student.uns.ac.id

Abstrak

Salah satu kunci sukses dari pengembangan perangkat lunak adalah perencanaan. Untuk membuat perencanaan yang baik diperlukan estimasi ukuran perangkat lunak yang akan dibangun. Ukuran perangkat lunak biasanya disajikan dalam bentuk *Lines of Code* (LOC). Metode *Function Point Analysis* (FPA) merupakan metode yang paling sering digunakan untuk memperkirakan ukuran perangkat lunak dalam satuan LOC. Akurasi dari FPA bisa ditingkatkan dengan cara penyesuaian nilai bobot pada tabel *Function Point Complexity*. Metode yang digunakan untuk penyesuaian nilai bobot adalah metode logika samar *Mamdani*. Evaluasi dilakukan dengan cara membandingkan galat relatif antara hasil pengukuran FPA murni, FPA modifikasi *Mamdani* model 1, FPA modifikasi *Mamdani* model 2 serta nilai tengah antara model 1 dan model 2, dengan nilai LOC sebenarnya dari perangkat lunak. Model 1 dan model 2 dibedakan oleh nilai himpunan *fuzzy* pada proses fuzzifikasi. Sebanyak 13 perangkat lunak digunakan untuk pengujian. Hasilnya estimasi menggunakan FPA nilai tengah memberikan hasil terbaik dengan galat terkecil yaitu 1,6% dibandingkan FPA modifikasi *Mamdani* model 1 (2%), FPA model 2 (3,2%) dan FPA Murni (3,4%). Perbedaan galat relatif tersebut mempunyai tingkat kepercayaan secara statistika sebesar 76%.

Kata Kunci: Estimasi ukuran perangkat lunak, *Function Point Analysis*, Metode *Mamdani*

1. PENDAHULUAN

Salah satu faktor sukses dalam mengerjakan sebuah proyek, termasuk proyek perangkat lunak adalah estimasi dan perencanaan yang akan dilakukan. Proyek tidak dapat dikendalikan jika proyek tersebut tidak mempunyai estimasi dan perencanaan. Estimasi dan perencanaan dalam proyek perangkat lunak bisa diawali dengan pengukuran dari perangkat lunak yang akan dibuat atau dikembangkan [1].

Ada beberapa metode dalam melakukan pengukuran perangkat lunak, salah satu yang paling banyak digunakan adalah *Function Point Analysis* (FPA). FPA merupakan metode pengukuran perangkat lunak yang paling banyak digunakan di seluruh dunia. FPA pertama kali dikenalkan oleh Allan Albrecht pada tahun 1979 dan sekarang terus diperbaharui oleh *International Function Point User Group* (IFPUG) [2]. Terdapat beberapa metode pengukuran perangkat lunak selain FPA, sebagai contoh: *Lines of Code* (LOC) dan *wideband delphi*. Kedua metode pengukuran perangkat lunak tersebut mempunyai beberapa *issue* atau permasalahan yang membuat kedua metode pengukuran perangkat lunak tersebut tidak digunakan sehingga jarang ada pengembangan yang bisa membuat kedua metode pengukuran perangkat lunak tersebut berfungsi dengan lebih baik [3].

Dalam metode FPA terdapat 5 fungsi sebagai parameter pengukuran sebuah perangkat lunak, yaitu *Internal Logical File* (ILF), *External Interface File* (EIF), *External Input* (EI), *External Output* (EO), dan *External Inquiry* (EQ). Pengukuran perangkat lunak dilakukan dengan mengambil data untuk 5 fungsi tersebut kemudian di kelompokan berdasarkan banyak *Data Element Types* (DETs), *File Type References* (FTRs), dan *Record Element Types* (RETs) yang disebut *weight* menjadi 3 kelompok, yaitu *low, average* dan *high* [4]. Pengelompokan *weight* tersebut sesuai dengan tabel *Function Point Complexity Weight* yang dibuat oleh Allan Albrecht [5].

Output dari *weight* yang dikelompokan berdasarkan tabel *function point complexity weight* bersifat himpunan *crisp* sehingga memunculkan *output* yang tidak sesuai antara *weight* yang berdekatan atau berjauhan. Sebagai contoh untuk penghitungan ILF, sebuah aplikasi A memiliki jumlah DET 50 dan RET 3, masuk ke kelompok *average.* Aplikasi B memiliki jumlah DET 20 dan RET 3, masuk ke kelompok *average.* Sedangkan aplikasi C memiliki jumlah DET 19 dan RET 3, masuk ke kelompok *low.* Dengan jumlah RET yang sama untuk semua aplikasi, aplikasi A memiliki beda jumlah DET sebanyak 30 dengan aplikasi B dan kedua aplikasi tersebut masuk ke dalam kelompok yang sama, yaitu *average.* Sedangkan aplikasi B dan aplikasi C memiliki beda jumlah DET hanya 1 tetapi aplikasi C tidak masuk ke dalam satu kelompok dengan aplikasi B, aplikasi C masuk ke dalam kelompok *low.* Ilustrasi tersebut menunjukan kelemahan himpunan *crisp.* Berdasarkan pemikiran sebagai manusia, pengelompokan *weight* ke dalam tabel *function point complexity weight* kurang benar jika menggunakan himpunan *crisp* atau nilai tunggal.

Fuzzy logic digunakan untuk memperbaharui nilai *weight* pada tabel *function point complexity weight*. *Fuzzy logic* menghasilkan *output* berupa himpunan *fuzzy*. Pemikiran manusia bersifat *fuzzy* atau samar, begitu juga dengan kejadian *real* di dunia. *Fuzzy logic* adalah logika yang menyatakan perkiraan bukan kepastian, yang dimana mirip dengan pemikiran manusia dan kejadian *real* di dunia. Oleh karena itu *fuzzy logic* tidak sama seperti dengan logika pada umumnya [6]. Dengan *fuzzy logic* memungkinkan adanya toleransi dalam pengelompokan nilai *weight* agar memunculkan *output* yang lebih rasional. Sehingga *output* yang didapat menjadikan pengukuran perangkat lunak lebih akurat. Metode yang digunakan dalam *fuzzy logic* adalah metode *Mamdani*. Metode *Mamdani* telah diterima secara luas untuk digunakan para ahli dalam bidang *soft computing*. Metode *Mamdani* bekerja lebih intuitif dan mempunyai tingkat toleransi keputusan berdasarkan pola pikir manusia yang tinggi [7]. *Output* dalam proses defuzzifikasi metode *Mamdani* berupa nilai ganda atau *range* sehingga memungkinkan metode *Mamdani* untuk menghasilkan *output* yang mempunyai toleransi tinggi karena mempertimbangkan nilai-nilai didekatnya.

2. METODE

2.1. Pengumpulan Data

Data diambil dari beberapa aplikasi yang dibuat dalam kerja praktek mahasiswa S1 Informatika UNS angkatan 2011 dan aplikasi yang dipakai oleh Pemerintah Kabupaten Surakarta. Dalam aplikasi tersebut diambil data berupa komponen-

komponen penghitungan metode FPA sehingga dapat menentukan *size* dari aplikasi berdasarkan metode FPA.

2.2. Penghitungan Estimasi Ukuran Perangkat Lunak

Langkah pertama melakukan perbaikan nilai *weight* dengan *fuzzy logic* metode *Mamdani*. Nilai *weight* dalam metode FPA didapat dari menghitung banyak RET/FTR dan DET yang kemudian masuk ke salah satu kelompok (*low, average, high*) seperti terlihat pada Tabel 1.

Tabel 1. *Complexity matrix for ILF and EIF* [8]

ILF/EIF	**DET**		
RET	1-19	20-50	51+
1	*Low*	*Low*	*Average*
2-5	*Low*	*Average*	*High*
6+	*Average*	*High*	*High*

Kemudian untuk tiap kelompok di tiap komponen memiliki nilai yang berbeda yang menjadi nilai *weight* seperti terlihat pada Tabel 2.

Tabel 2. *Function point complexity weight* [8]

Component	***Low***	***Average***	***High***
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiris	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

Proses pengelompokan tersebut bersifat *crisp* sehingga akan diubah dengan *fuzzy logic* metode *Mamdani*. Pertama untuk proses fuzzifikasi adalah membuat *input fuzzy sets* untuk RET/FTR dan DET. Didefinisikan sebagai *small, medium, large.* Selanjutnya akan digunakan 2 model *fuzzy sets.* Kemudian membuat *rule* untuk semua model *fuzzy logic* dimana *input* RET/FTR dan DET dihubungkan dengan "AND" dan menghasilkan *output*.

Kemudian proses defuzzifikasi dengan metode *Mamdani* menggunakan fungsi *Center of Gravity* (COG). Hasil dari proses defuzzifikasi tersebut merupakan nilai *weight* yang baru dan digunakan untuk penghitungan metode FPA.

Setelah mendapat nilai *weight* yang baru dengan *fuzzy logic* metode *Mamdani* maka dilakukan penghitungan pengukuran perangkat lunak dengan FPA. Diawali dengan mendapatkan *Unadusted Function Point* (UFP) dengan cara mengalikan nilai *weight* dengan jumlah fitur yang ada di setiap fungsi yang berbeda. Kemudian hasil yang didapat setiap fungsi dijumlahkan dan didapat nilai UFP.

Rule yang dipakai ada sembilan seperti yang terlihat pada Tabel 3.

Tabel 3. *Fuzzy logic rule set*

Rule	**DET**	**RET/FTR**	***Output***
1	*Small*	*Small*	*Low*
2	*Small*	*Medium*	*Low*
3	*Small*	*Large*	*Average*
4	*Medium*	*Small*	*Low*
5	*Medium*	*Medium*	*Average*
6	*Medium*	*Large*	*High*
7	*Large*	*Small*	*Average*
8	*Large*	*Medium*	*High*
9	*Large*	*Large*	*High*

Kemudian mencari nilai *Total Degree of Infulence* (TDI) dengan cara menjumlahkan banyak pengaruh terhadap perangkat lunak yang diukur dari 14 faktor yang ada sebagai *Value Adjusment Factor*. Tiap faktor diberi nilai dari 0 sampai 5.0 jika faktor tersebut tidak menimbulkan efek apapun dan 5 jika faktor tersebut sangat penting di perangkat lunak yang diukur. Faktor tersebut ada 14, seperti terlihat pada Tabel 4.

Tabel 4. *Value Adjusment Factor* [8]

ID	***Factors***
C1	*Data communications*
C2	*Distributed function*
C3	*Performance objectives*
C4	*Heavily used configuration*
C5	*Transaction rate*
C6	*On-line data entry*
C7	*End-user efficiency*
C8	*On-line update*
C9	*Complex processing*
C10	*Reusability*
C11	*Installation ease*
C12	*Operational ease*
C13	*Multiple sites*
C14	*Facilitate change*

Setelah mendapat nilai TDI maka bisa mendapatkan nilai *Technical Complexity Adjusment* (TCA). Formulasi untuk mendapatkan TCA adalah sebagai berikut.

$$TCA = 0.65 + 0.01 * TDI \quad (1)$$

Setelah mendapat nilai TCA maka bisa mendapatkan nilai *Function Point* (FP). Formulasi untuk mendapatkan nilai FP adalah sebagai berikut.

$$FP = UFP * TCA \quad (2)$$

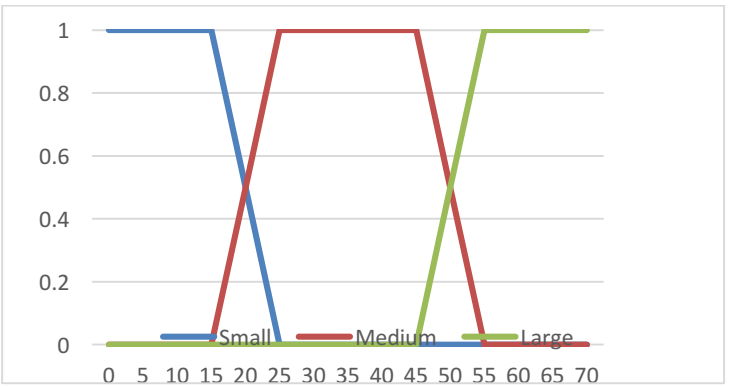
Dengan nilai FP bisa digunakan untuk mengukur LOC perangkat lunak, *cost, effort, resource* yang dibutuhkan dan lama pengerjaan perangkat lunak. Untuk mengukur

LOC dari FP cukup dengan hanya mengalikan nilai FP dengan nilai faktor produktivitas bahasa pemrograman yang dibuat oleh Capers Jones [1].

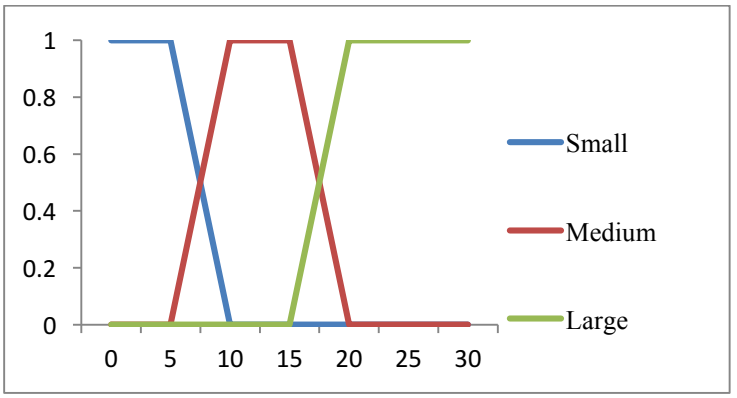
Tabel 5. *Productivity Factor Programming Language* [14]

Programming Language	***Productivity Factor***
SQL	37
PHP	53
HTML / Javascript	58

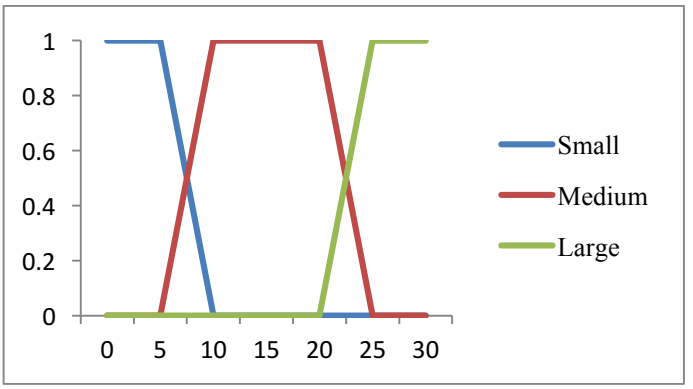
Fuzzy logic digunakan untuk mengembangkan nilai *weight.* Nilai DET dan RET/FTR yang ada dimasukan ke dalam kelompok *small, medium,* atau *large* sesuai dengan *rule* yang telah dibuat. Kemudian model *fuzzy set* dibuat berdasarkan data *weight* dari tabel *complexity matrix.* Nilai DET dan RET/FTR masuk kedalam proses fuzzifikasi. *Fuzzy set* dalam proses fuzzifikasi berdasarkan dari nilai *complexity matrix* untuk setiap fungsi pembeda (ILF, EIF, EI, EQ, EO). Terdapat dua model *fuzzy set,* perbedaan kedua model tersebut terletak pada nilai RET/FTR, sedangkan untuk nilai DET adalah sama untuk kedua model. *Fuzzy set* proses fuzzifikasi untuk kedua model seperti terlihat pada Gambar 1 sampai dengan Gambar 9.



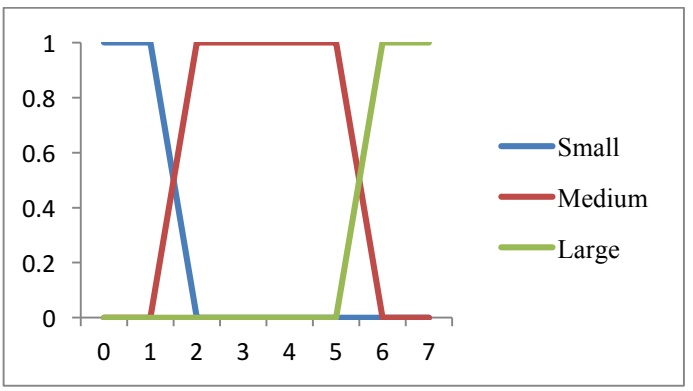
Gambar 1. DET ILF & EIF



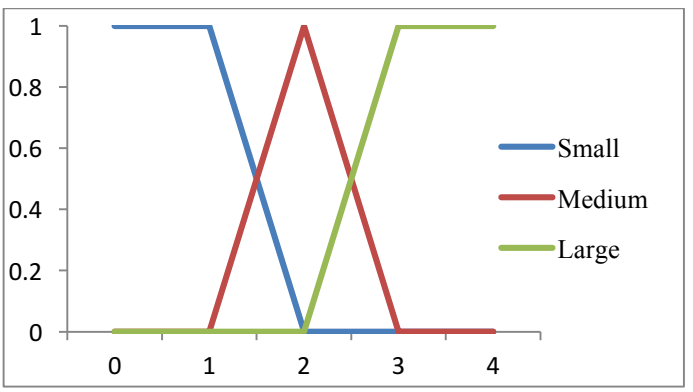
Gambar 2. DET EI



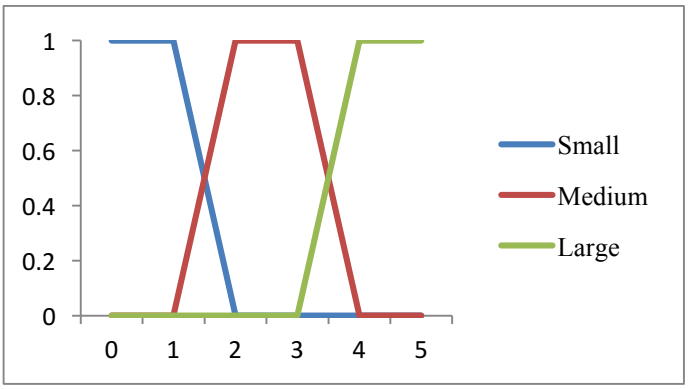
Gambar 3. DET EO & EQ



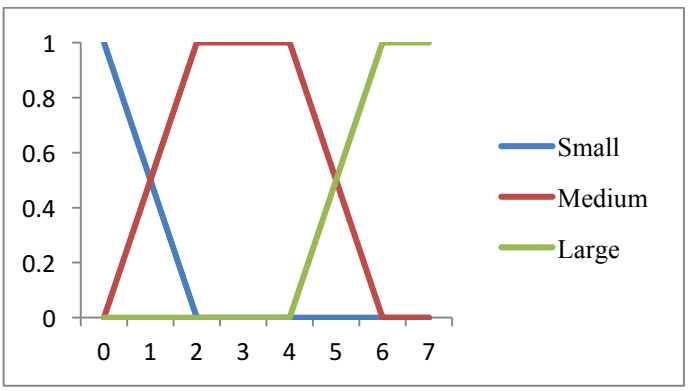
Gambar 4. RET ILF & EIF *fuzzy logic* model pertama



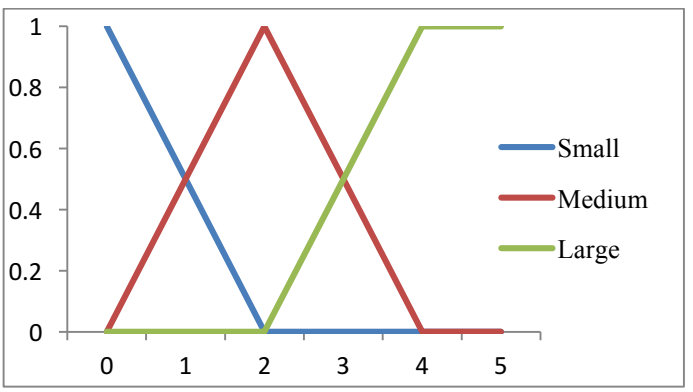
Gambar 5. FTR EI *fuzzy logic* model pertama



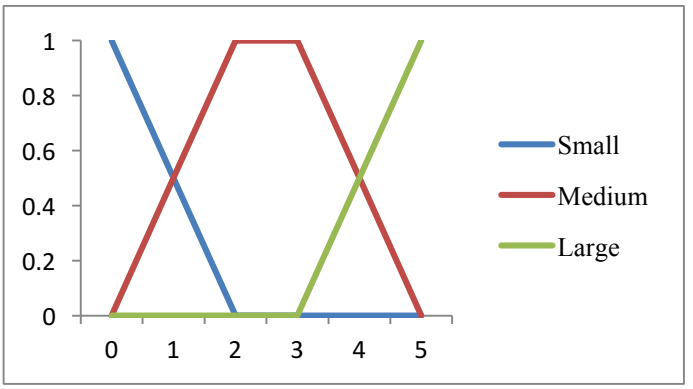
Gambar 6. FTR EO & EQ *fuzzy logic* model pertama



Gambar 7. RET ILF & EIF *fuzzy logic* model kedua

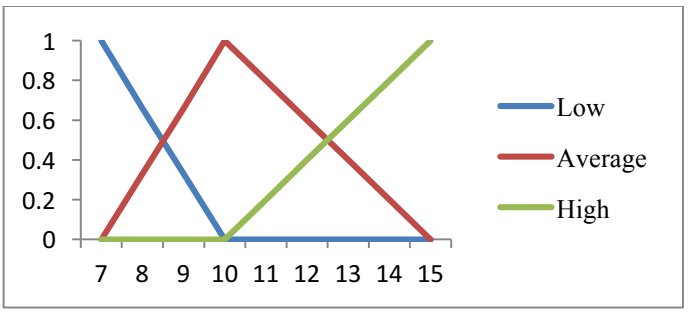


Gambar 8. FTR EI *fuzzy logic* model kedua

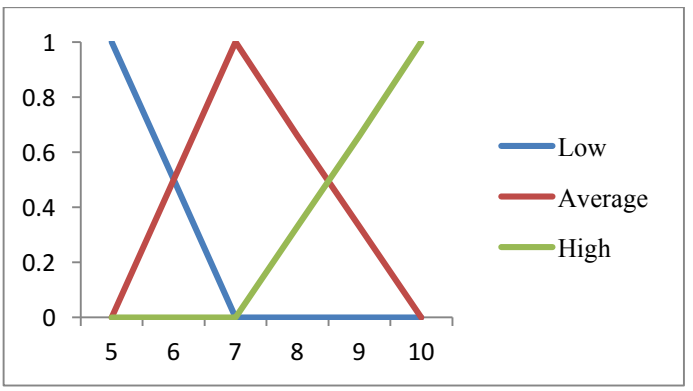


Gambar 9. FTR EO & EQ *fuzzy logic* model kedua

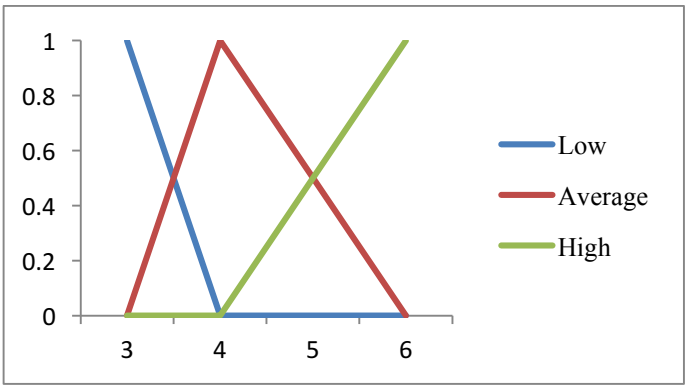
Selanjutnya adalah *fuzzy set* untuk proses defuzzifikasi. *Fuzzy set* ini didapat berdasarkan dari nilai *Function Point Complexity Weight*. *Fuzzy set* untuk proses defuzzifikasi sama untuk kedua model seperti terlihat pada Gambar 10 sampai dengan Gambar 13.



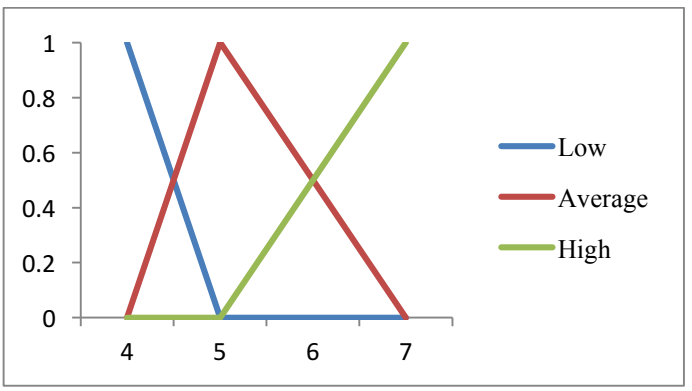
Gambar 10. Defuzzifikasi ILF



Gambar 11. Defuzzifikasi EIF



Gambar 12. Defuzzifikasi EI & EQ



Gambar 13. Defuzzifikasi EO

Penghitungan LOC menghasilkan 4 nilai dari beberapa metode. Nilai pertama adalah LOC yang didapat dari metode FPA. Nilai kedua adalah LOC yang didapat dari metode FPA dengan *fuzzy logic* model pertama. Nilai ketiga adalah LOC yang didapat dari metode FPA dengan *fuzzy logic* model kedua. Nilai keempat adalah LOC yang didapat dari nilai tengah antara metode FPA dengan *fuzzy logic* model pertama dan model kedua. Hasil LOC dari masing-masing metode akan dijadikan nilai estimasi *size* suatu perangkat lunak yang diukur.

2.3. Evaluasi Hasil

Evaluasi dilakukan dengan cara menghitung nilai *Mean Relative Error* (MRE) dari hasil estimasi baik FPA, *fuzzy* model 1, *fuzzy* model 2 maupun dengan nilai tengah terhadap nilai LOC sebenarnya.

3. HASIL DAN PEMBAHASAN

Hasil LOC sebenarnya dan hasil estimasi masing-masing model ditunjukkan pada Tabel 6.

Tabel 6. Hasil LOC *real* dan hasil estimasi

Aplikasi	LOC real	FPA	Fuzzy Model 1	Fuzzy Model 2	Nilai Tengah
Akomodasi	4089	3891.60	4008.23	4271.84	4140.03
Avenger	2468	2340.90	2441.97	2507.22	2474.59
Catering	4172	3707.36	3855.92	4216.15	4036.03
IMB	3341	3290.40	3374.34	3076.56	3225.45
iSpeedy	5310	5296.50	5217.81	5363.31	5290.56
Meetingroom	3520	3444.35	3507.79	3764.90	3636.34
Beasiswa	6587	6252.48	6405.69	6758.21	6581.95
Bidikmisi	7078	6930.24	7141.34	6972.50	7056.92
Penundaan	5213	5171.52	5217.88	5268.76	5243.32
Keluhan	4662	4535.52	4652.85	4708.52	4680.68
Profile	3264	3265.23	3356.53	3552.20	3454.36
RegBidikmisi	6223	5752.32	5893.98	6405.65	6149.81
Wisuda	4486	4443.54	4522.52	4482.90	4502.71

Berdasarkan Tabel 6 dicari *Mean Relative Error* (MRE) untuk masing-masing model, sehingga diperoleh hasil yang ditunjukkan pada Tabel 7. Terlihat bahwa estimasi nilai tengah memberikan hasil terbaik dengan MRE 1,6% dengan kata lain tingkat keberhasilannya 98,4%. Selanjutnya *fuzzy* model 1 (2%), *fuzzy* model 2 (3,2%) dan FPA (3.4%).

Tabel 7. Nilai *relative error* hasil estimasi

Aplikasi	**FPA**	***Fuzzy* 1**	***Fuzzy* 2**	**Nilai Tengah**
Akomodasi	0.048	0.020	0.045	0.012
Avenger	0.051	0.011	0.016	0.003
Catering	0.111	0.076	0.011	0.033
IMB	0.015	0.010	0.079	0.035
iSpeedy	0.003	0.017	0.010	0.004
Meeting room	0.021	0.003	0.070	0.033
Beasiswa	0.051	0.028	0.026	0.001
Bidikmisi	0.021	0.009	0.015	0.003
Penundaan	0.008	0.001	0.011	0.006
Keluhan	0.027	0.002	0.010	0.004
Profile	0.000	0.028	0.088	0.058
Reg Bidikmisi	0.076	0.053	0.029	0.012
Wisuda	0.009	0.008	0.001	0.004
MRE	0.034	0.020	0.032	0.016

Secara statistika perbedaan MRE tersebut mempunyai tingkat kepercayaan sebesar 76%, seperti pada analisa variansi uji beda *mean* pada Tabel 8.

Tabel 8. Uji beda *mean*

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.002960	3	0.00099	1.45	0.24	1.41
Within Groups	0.032658	48	0.00068			
Total	0.036	51				

4. SIMPULAN

Penggunaan metode *fuzzy logic* dapat meningkatkan tingkat akurasi estimasi ukuran perangkat lunak. Hal ini terbukti dengan menggunakan *fuzzy logic* sebagai metode untuk mengembangkan nilai *weight* pada tabel *Function Point Complexity Weight* maka tingkat akurasi estimasi LOC dari sebuah aplikasi akan lebih baik. Jika dengan meggunakan metode FPA saja didapat nilai MRE sebesar 3.4%, sedangkan jika ditambah dengan menggabungkan metode *fuzzy logic* didapat hasil MRE yang lebih baik, yaitu 2% untuk *fuzzy logic* model pertama dan 3,2% untuk *fuzzy logic* model kedua, bahkan hasil nilai tengah dari kedua model *fuzzy logic* tersebut menghasilkan nilai MRE yang lebih baik lagi, yaitu 1,5% yang menandakan tingkat akurasi menjadi lebih baik. Perbedaan tersebut mempunyai tingkat kepercayaan secara statistika sebesar 76% melalui uji beda *mean*.

5. REFERENSI

[1] Tunali, V. 2014. *Software Size Estimation Using Function Point Analysis – A Case Study for a Mobile Application*. Muhendisik ve Teknoloji Sempozyumu.

[2] Ferrucci, F., Gravino, C., and Sarro, F. 2014. *Conversion from IFPUG to COSMIC: within vs without-company equations*. Society for East Asian Archaeology.

[3] Shirmohammadi, S. 2009. *Software Size Estimation*. Ottawa.

[4] Balaji, N., Shivakumar, N., and Ananth, V. V. 2013. Software Cost Estimation using Function Point with Non Algorithmic Approach. *Global Journal of Computer Science and Technology Software & Data Engineering*. Vol 13(8): 1-5.

[5] Xia, W., Capretz, L. F., Ho, D., and Ahmed, F. 2008. A New Calibration for Function Point Complexity Weights. *Electrical and Computer Engineering Publications*. Vol. 50(7-8): 670-683.

[6] Elamvazuthi, I., Vasant P., and Webb, J. 2009. The Application of *Mamdani Fuzzy* Model for Auto Zoom Function of a Digital Camera. *International Journal of Computer Science and Information Security*. Vol. 6(3): 244-249.

[7] Kaur, A. and Kaur, A. 2012. Comparison of *Mamdani*-Type and Sugeno-Type *Fuzzy* Inference Systems for Air Conditioning System. *International Journal of Soft Computing and Engineering.* Vol. 2(2): 323-325.

[8] Jones, C. 2008. *Applied Software Measurement, Global Analysis of Productivity and Quality*, 3rd ed. McGraw-Hill, New York.