Comparison Analysis of Weight Value Changing in Function Point Analysis Between *Fuzzy* Inference System Mamdani and Tsukamoto for Software Size Estimation

Sambodo Wisnu Murti
Informatika, Fakultas MIPA
Universitas Sebelas Maret
Jl. Ir. Sutami No. 36 A Surakarta
Sambodo.wisnu@student.uns.ac.id

Ristu Saptono
Informatika, Fakultas MIPA
Universitas Sebelas Maret
Jl. Ir. Sutami No. 36 A Surakarta
ristu.saptono@staff.uns.ac.id

Esti Suryani
Informatika, Fakultas MIPA
Universitas Sebelas Maret
Jl. Ir. Sutami No. 36 A Surakarta
esti.suryani@staff.uns.ac.id

ABSTRACT

Key of successful software development is planning. To make a good planning is needed estimate the size of the software to be built. The size of the software is usually presented in the form of Lines of Code (LOC). Function Point Analysis (FPA) is the most common method used to estimate the size of the software in units LOC. Accuracy of the FPA can be improved by adjusting the weight values in the Function Point Complexity table. Previous research does the adjustment of the weight values of fuzzy logic Mamdani, and this study was conducted to analyze the accuracy of the result comparison between Mamdani method from previous studies with Tsukamoto method. The evaluation is done by comparing the relative error between the measurement of pure FPA, FPA with Mamdani modification, Tsukamoto modification, and with the actual value of the software LOC. A total of 13 software was used for testing. The result, FPA gives the best results with the smallest error that FPA with Mamdani modification (2.3%), FPA with Tsukamoto modification (2.75%) and the pure FPA (3.51%). The average yield obtained processing time are: the smallest is with pure FPA method 1.1 ms, while the Tsukamoto method obtained processing time 1.9 ms and the largest is the Mamdani method is 3 ms.

Keywords: *Function Point Analysis, Mamdani, Software Size Estimation, Tsukamoto*

1. PENDAHULUAN

Pengukuran perangkat lunak atau *Software Size Estimation* sangat dibutuhkan perusahaan atau individu sebagai tolok ukur sumber daya yang harus dikeluarkan ketika akan membangun suatu aplikasi. Terdapat beberapa metode pengukuran perangkat lunak, sebagai contoh yaitu: *Algorithm Model, Expert Judgments, Estimation by Analogy*. Metode Algoritma didesain untuk mendapatkan estimasi ukuran perangkat lunak menggunakan persamaan matematika. Persamaan matematika ini berdasarkan penelitian dan data yang telah dilakukan, salah satunya *Source Lines of Codes* (SLOC)[1], jumlah fungsi dalam perangkat lunak, maupun driver yang digunakan dalam perangkat lunak. Metode *Expert Judgement* menggunakan perhitungan berdasarkan pengalaman seorang pakar maupun sekelompok ahli dalam melakukan perhitungan. *Metode Estimation by Analogy* menggunakan perhitungan berdasarkan perangkat lunak sejenis yang pernah dibuat[1].

Function Point Analysis (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]. Keunggulan dari FPA sendiri adalah FPA mudah digunakan dan terdapat standarisasi untuk semua aplikasi yang ada di dunia ini.

Metode FPA menggunakan nilai bobot sebagai nilai untuk setiap fungsi yang ada didalam aplikasi. Nilai bobot tersebut bersifat teratur berdasarkan pada tabel *Function Point Complexity Weight*. Dalam penelitian ini, logika *fuzzy* digunakan untuk memodifikasi nilai bobot pada tabel Function Point Complexity Weight. Logika *fuzzy* menghasilkan output berupa himpunan *fuzzy*. Dan untuk membandingkan metode logika *fuzzy* yang paling akurat, maka akan digunakan metode Mamdani dan Tsukamoto. Setelah mendapatkan nilai Function Point (FP) dari sebuah perangkat lunak, maka dapat diukur size dari perangkat lunak tersebut dengan menghitung Lines of Code (LOC). Oleh karena itu, pengerjaan proyek perangkat lunak akan berjalan lebih efektif dalam segi waktu

dan efisien dari segi biaya yang dikeluarkan dari estimasi yang didapat menggunakan FPA.

Pernah dilakukan penelitian dengan modifikasi nilai bobot function point complexity weight pada FPA dengan logika *fuzzy* metode Mamdani untuk mendapatkan tingkat keakurasian yang lebih baik[3], sedangkan pada penelitian ini bertujuan untuk menganalisa perbandingan tingkat akurasi metode Mamdani dengan metode Tsukamoto yang digunakan untuk memodifikasi nilai bobot pada FPA dan untuk mendapatkan hasil rata-rata kesalahan relatif (Mean Relative Error) terkecil dari kedua metode, dengan demikian dapat disimpulkan metode mana yang memiliki tingkat akurasi yang lebih baik, sehingga dapat dipakai untuk perhitungan ukuran perangkat lunak dikemudian hari.

2. METODOLOGI PENELITIAN

2.1. Pengumpulan Data

Pengumpulan data didapatkan dari sistem informasi yang dibuat dalam Kerja Praktek Mahasiswa S1 Informatika UNS angkatan 2011. Dalam perancangan sistem informasi tersebut dihitung jumlah kelompok RET dan DET pada 5 komponen yang terdapat pada metode FPA, yaitu ILF, EIF, EI, EO, dan EQ agar bisa diolah menggunakan metode FPA dan didapatkan estimasi size dari sistem informasi yang akan dibuat, sehingga pengerjaan sistem informasi tersebut dapat dikerjakan dengan efektif dan efisien.

2.2 Perhitungan Estimasi Ukuran Perangkat Lunak

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

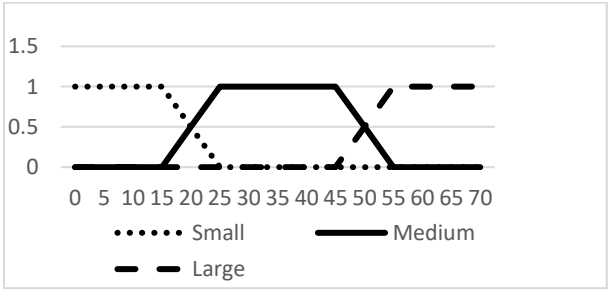
Tabel 1. Complexity Matrix for ILF and EIF[4].

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 bobot seperti terlihat pada Tabel 2.Tabel 2. Function Point Complexity Weight[4].

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 teratur sehingga akan diubah dengan logika *fuzzy* metode Mamdani dan Tsukamoto. Pertama untuk proses *fuzzy*fikasi adalah membuat input *fuzzy* sets untuk DET dan RET. Didefinisikan sebagai small, medium, large. Logika *fuzzy* digunakan untuk merubah nilai bobot. 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). *Fuzzy* set proses *fuzzy*fikasi untuk kedua model seperti terlihat pada Gambar 1 sampai dengan Gambar 6 dan setiap gambar menggambarkan nilai bobot *small, medium*, dan *large* seperti pada persamaan 1, 2, dst.



Gambar 1. Diagram *Fuzzy*fikasi DET pada ILF dan EIF

Persamaan bobot pada gambar 1 seperti terlihat pada persamaan 1 sampai 3 berikut.
Small:

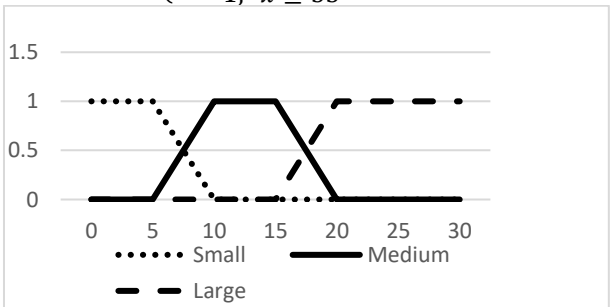
$$\mu(x)small = \begin{cases} 1; & x \leq 15 \\ \frac{25-x}{25-15}; & 15 \leq x \leq 25 \\ 0; & x \geq 25 \end{cases}$$

Medium:

$$\mu(x)medium = \begin{cases} 0; & x \leq 15 \\ \frac{x-15}{25-15}; & 15 \leq x \leq 25 \\ 1; & 25 \leq x \leq 45 \\ \frac{55-x}{55-45}; & 45 \leq x \leq 55 \end{cases}$$

Large:

$$\mu(x)large = \begin{cases} 0; & x \leq 45 \\ \frac{x-45}{55-45}; & 45 \leq x \leq 55 \\ 1; & x \geq 55 \end{cases}$$



Gambar 2. Diagram *Fuzzy*fikasi DET pada EI

Persamaan bobot pada gambar 2 seperti terlihat pada persamaan 4 sampai 6 berikut.

Small:

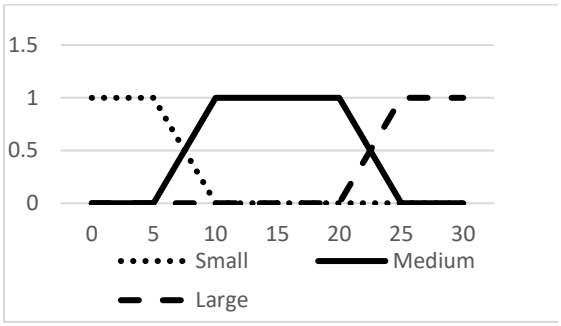
$$\mu(x)small = \begin{cases} 1; & x \leq 5 \\ \frac{10-x}{10-5}; & 5 \leq x \leq 10 \\ 0; & x \geq 10 \end{cases}$$

Medium:

$$\mu(x)medium = \begin{cases} 0; & x \leq 5 \\ \frac{x-10}{10-5}; & 5 \leq x \leq 10 \\ 1; & 10 \leq x \leq 15 \\ \frac{20-x}{20-15}; & 15 \leq x \leq 20 \end{cases}$$

Large:

$$\mu(x)large = \begin{cases} 0; & x \leq 15 \\ \frac{x-15}{20-15}; & 15 \leq x \leq 20 \\ 1; & x \geq 20 \end{cases}$$



Gambar 3. Diagram *Fuzzy*fikasi DET pada EO dan EQ

Persamaan bobot pada gambar 4.3 seperti terlihat pada persamaan 7 sampai 9 berikut.

Small:

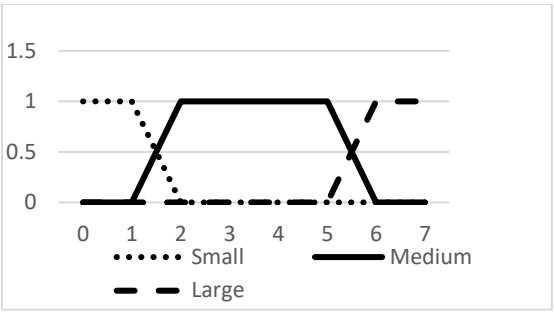
$$\mu(x)small = \begin{cases} 1; & x \leq 5 \\ \frac{10-x}{10-5}; & 5 \leq x \leq 10 \\ 0; & x \geq 10 \end{cases}$$

Medium:

$$\mu(x)medium = \begin{cases} 0; & x \leq 5 \\ \frac{x-10}{10-5}; & 5 \leq x \leq 10 \\ 1; & 10 \leq x \leq 20 \\ \frac{25-x}{25-20}; & 20 \leq x \leq 25 \end{cases}$$

Large:

$$\mu(x)large = \begin{cases} 0; & x \leq 20 \\ \frac{x-20}{25-20}; & 20 \leq x \leq 25 \\ 1; & x \geq 25 \end{cases}$$



Gambar 4. Diagram *Fuzzy*fikasi RET pada ILF dan EIF

Persamaan bobot pada gambar 4 seperti terlihat pada persamaan 10 sampai 12 berikut.

Small:

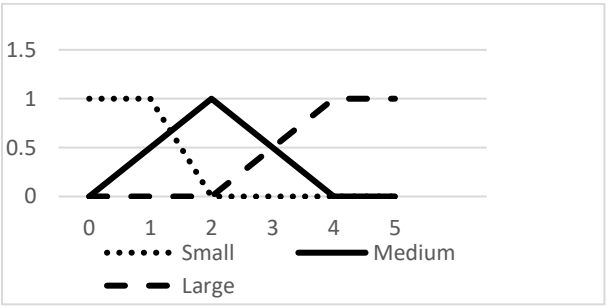
$$\mu(x)small = \begin{cases} 1; & x \leq 1 \\ \frac{2-x}{2-1}; & 1 \leq x \leq 2 \\ 0; & x \geq 2 \end{cases}$$

Medium:

$$\mu(x)medium = \begin{cases} 0; & x \leq 1 \\ \frac{x-1}{2-1}; & 1 \leq x \leq 2 \\ 1; & 2 \leq x \leq 5 \\ \frac{6-x}{6-5}; & 5 \leq x \leq 6 \end{cases}$$

Large:

$$\mu(x)large = \begin{cases} 0; & x \leq 5 \\ \frac{x-5}{6-5}; & 5 \leq x \leq 6 \\ 1; & x \geq 6 \end{cases}$$



Gambar 5. Diagram *Fuzzy*fikasi RET pada EI

Persamaan bobot pada gambar 5 seperti terlihat pada persamaan 13 sampai 15.

Small:

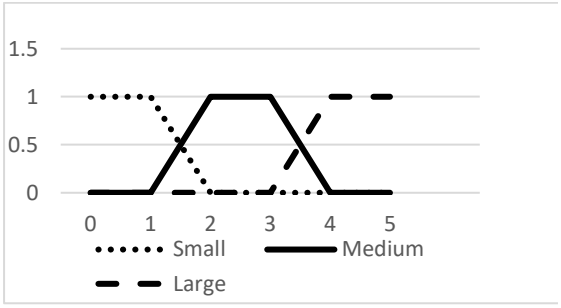
$$\mu(x)small = \begin{cases} 1; & x \leq 1 \\ \frac{2-x}{2-1}; & 1 \leq x \leq 2 \\ 0; & x \geq 2 \end{cases}$$

Medium:

$$\mu(x)medium = \begin{cases} 0; & x \leq 1 \\ \frac{x-1}{2-1}; & 2 \leq x \leq 1 \\ \frac{4-x}{4-2}; & 2 \leq x \leq 4 \end{cases}$$

Large:

$$\mu(x)large = \begin{cases} 0; & x \leq 2 \\ \frac{x-2}{4-2}; & 2 \leq x \leq 4 \\ 1; & x \geq 4 \end{cases}$$



Gambar 6. Diagram *Fuzzy*fikasi RET pada EO dan EQ

Persamaan bobot pada gambar 6 seperti terlihat pada persamaan 16 sampai 18 berikut.
Small:

$$\mu(x)small = \begin{cases} 1; & x \leq 1 \\ \frac{2-x}{2-1}; & 1 \leq x \leq 2 \\ 0; & x \geq 2 \end{cases}$$

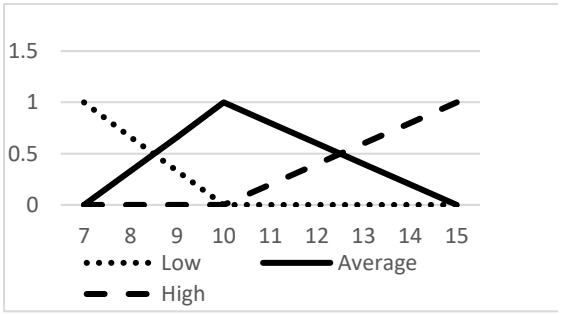
Medium:

$$\mu(x)medium = \begin{cases} 0; & x \leq 1 \\ \frac{x-1}{2-1}; & 1 \leq x \leq 2 \\ 1; & 2 \leq x \leq 3 \\ \frac{4-x}{4-3}; & 3 \leq x \leq 4 \end{cases}$$

Large:

$$\mu(x)large = \begin{cases} 0; & x \leq 3 \\ \frac{x-3}{4-3}; & 3 \leq x \leq 4 \\ 1; & x \geq 4 \end{cases}$$

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 7 sampai dengan Gambar 10 dan setiap gambar diagram memiliki persamaan nilai bobot untuk *low, average*, dan *high* seperti pada persamaan 19, dst..



Gambar 7. Diagram Deffuzyfikasi pada ILF

Persamaan bobot pada gambar 7 seperti terlihat pada persamaan 19 sampai 21 berikut.
Low:

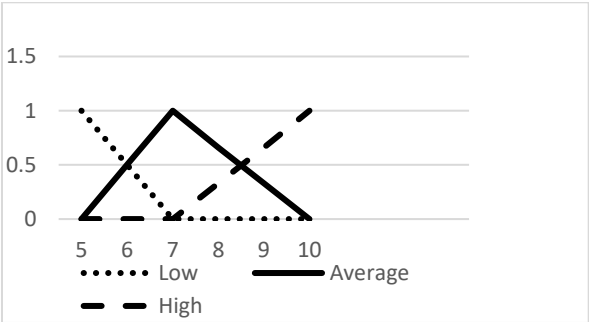
$$\mu(x)low = \begin{cases} 1; & x \leq 7 \\ \frac{10-x}{10-7}; & 7 \leq x \leq 10 \\ 0; & x \geq 10 \end{cases}$$

Average:

$$\mu(x)average = \begin{cases} 0; & x \leq 7 \\ \frac{x-7}{10-7}; & 7 \leq x \leq 10 \\ \frac{15-x}{15-10}; & 10 \leq x \leq 15 \end{cases}$$

High:

$$\mu(x)high = \begin{cases} 0; & x \leq 10 \\ \frac{x-10}{15-10}; & 10 \leq x \leq 15 \\ 1; & x \geq 15 \end{cases}$$



Gambar 8. Diagram De*fuzzy*fikasi pada EIF

Persamaan bobot pada gambar 8 seperti terlihat pada persamaan 22 sampai 24 berikut.
Low:

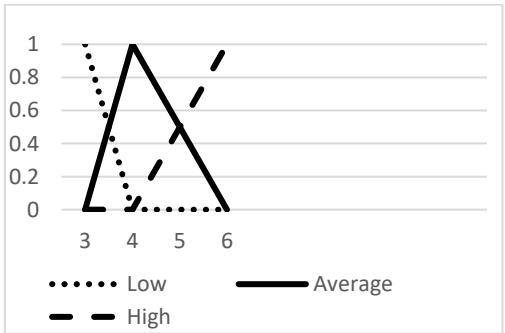
$$\mu(x)low = \begin{cases} 1; & x \leq 5 \\ \frac{7-x}{7-5}; & 5 \leq x \leq 7 \\ 0; & x \geq 7 \end{cases}$$

Average:

$$\mu(x)average = \begin{cases} 0; & x \leq 5 \\ \frac{x-5}{7-5}; & 5 \leq x \leq 7 \\ \frac{10-x}{10-7}; & 7 \leq x \leq 10 \end{cases}$$

High:

$$\mu(x)high = \begin{cases} 0; & x \leq 7 \\ \frac{x-7}{10-7}; & 7 \leq x \leq 10 \\ 1; & x \geq 10 \end{cases}$$



Gambar 9 Diagram De*fuzzy*fikasi pada EI dan EQ

Persamaan bobot pada gambar 9 seperti terlihat pada persamaan 25 sampai 27.

Low:

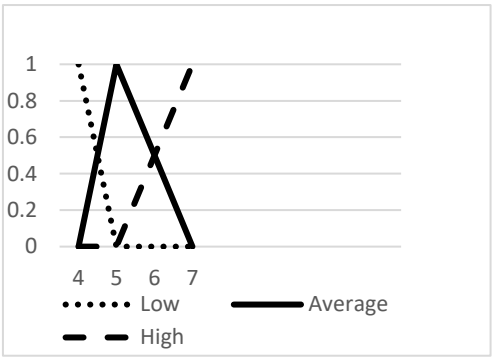
$$\mu(x)low = \begin{cases} 1;\ x \le 3 \\ \frac{4-x}{4-3}; 3 \le x \le 4 \\ 0;\ x \ge 4 \end{cases}$$

Average:

$$\mu(x)average = \begin{cases} 0;\ x \le 3 \\ \frac{x-3}{4-3}; 3 \le x \le 4 \\ \frac{6-x}{6-4}; 4 \le x \le 6 \end{cases}$$

High:

$$\mu(x)high = \begin{cases} 0;\ x \le 4 \\ \frac{x-4}{6-4};\ 4 \le x \le 6 \\ 1;\ x \ge 6 \end{cases}$$



Gambar 10 Diagram De*fuzzy*fikasi pada EO

Persamaan bobot pada gambar 10 seperti terlihat pada persamaan 28 sampai 30 berikut.

Low:

$$\mu(x)low = \begin{cases} 1;\ x \le 4 \\ \frac{5-x}{5-4}; 4 \le x \le 5 \\ 0;\ x \ge 5 \end{cases}$$

Average:

$$\mu(x)average = \begin{cases} 0;\ x \le 3 \\ \frac{x-4}{5-4}; 4 \le x \le 5 \\ \frac{7-x}{7-5}; 5 \le x \le 7 \end{cases}$$

High:

$$\mu(x)high = \begin{cases} 0;\ x \le 5 \\ \frac{x-5}{7-5};\ 5 \le x \le 7 \\ 1;\ x \ge 7 \end{cases}$$

Kemudian membuat rule untuk semua model logika *fuzzy* dimana input DET dan RET dihubungkan dengan "AND" dan menghasilkan output. Rule yang dipakai seperti pada Tabel 3.

Tabel 3. *Fuzzy* Logic Rule Set

Rule	DET	RET	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 proses defuzzifikasi dengan metode Mamdani menggunakan fungsi Centroid dan metode Tsukamoto menggunakan Center Average De*fuzzy*fier[5]. Hasil dari proses defuzzifikasi tersebut merupakan nilai bobot yang baru dan digunakan untuk penghitungan metode FPA. Setelah mendapat nilai weight yang baru dengan logika *fuzzy* metode Mamdani dan metode Tsukamoto maka dilakukan penghitungan pengukuran perangkat lunak dengan FPA. Diawali dengan mendapatkan Unadusted Function Point (UFP) dengan cara mengalikan nilai bobot dengan jumlah fitur yang ada di setiap fungsi yang berbeda. Kemudian hasil yang didapat setiap fungsi dijumlahkan dan didapat nilai UFP.

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 Adusment Factor. Tiap faktor diberi nilai dari 0 sampai 5. Jika faktor tersebut tidak menimbulkan efek apapun bernilai 0 dan 5 jika faktor tersebut sangat penting di perangkat lunak yang diukur. 14 faktor tersebut seperti terlihat pada Tabel 4.

Tabel 4. Value Adjusment Factor [6]

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 menghitung TCA seperti pada persamaan 31.

$$TCA = 0.65 + 0.01 * TDI \text{.....(31)}$$

Setelah mendapat nilai TCA maka bisa mendapatkan nilai *Function Point* (FP). Formulasi untuk menghitung nilai FP adalah seperti pada persamaan 32.

$$FP = UFP * TCA \text{.....(32)}$$

Dengan nilai FP bisa digunakan untuk mengukur LOC perangkat lunak, 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 yang dapat dilihat pada Tabel 5[7].

Tabel 5. Productivity Factor Programming[4].

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

Penghitungan LOC juga menghasilkan 2 nilai dikarenakan adanya 2 metode logika *fuzzy* yang digunakan. Nilai range LOC antara logika *fuzzy* metode Mamdani dan logika *fuzzy* metode Tsukamoto dan hasil LOC dari masing-masing model logika *fuzzy* akan dijadikan nilai estimasi size suatu perangkat lunak yang diukur.

2.3 Evaluasi Hasil

Dari hasil estimasi yang didapat, dicari rata-rata error relatif terhadap ukuran LOC real yang dihitung dengan aplikasi SLOC Matrics 3.0. Dengan rumus Mean Relative Error (MRE):

$$MRE = \frac{1}{n}\Sigma\left|\frac{x-y}{y}\right| \quad \text{.....(33)}$$

dimana n adalah banyak item yang dihitung dan x merupakan nilai estimasi, dan y merupakan nilai asli. Penelitian ini juga menganalisis perbandingan waktu proses yang dibutuhkan setiap metode untuk melakukan perhitungan, untuk mengetahui metode mana yang tercepat dalam melakukan perhitungan digunakan fungsi *microtime* sebagai berikut:

```
$start = microtime (true);

$finish = microtime (true);

$totaltime = $finish - $start;
```

3. HASIL DAN PEMBAHASAN

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

Tabel 6. Hasil LOC Real dan Estimasi Aplikasi

Aplikasi	LOC Real	Estimasi LOC		
		FPA	Mamdani	Tsukamoto
Akomodasi	4089	3892	4009	3993
Beasiswa	2468	2341	2442	2410
Profile	6587	6253	6406	6424
Meetingroom	7078	6931	7142	7253
Avenger	4172	3708	3856	4109
IMB	3341	3291	3375	3195
iSpeedy	5210	5297	5231	5105
Bidikmisi	4662	4536	4653	4609
Keluhan	3520	3445	3508	3757
Catering	5113	5172	5277	4963
Penundaan	3264	3266	3357	3359
Reg Bidikmisi	6223	5753	5894	6297
Wisuda	4486	4444	4292	4421

Berdasarkan Tabel 6 dicari Mean Relative Error (MRE) untuk masing-masing metode, sehingga diperoleh hasil yang ditunjukkan pada Tabel 7. Terlihat bahwa metode logika *fuzzy* Mamdani memiliki angka rata-rata error relatif sebesar (2.30%), metode Tsukamoto (2.75%) dan FPA (3.51%).

Tabel 7. Nilai Error Relatif Hasil Estimasi

Aplikasi	Relative Error		
	FPA	Mamdani	Tsukamoto
Akomodasi	0.0483	0.0198	0.0236
Beasiswa	0.0515	0.0105	0.0236
Profile	0.0508	0.0275	0.0248
Meetingroom	0.0209	0.0089	0.0247
Avenger	0.1114	0.0758	0.0153
IMB	0.0151	0.01	0.0437
iSpeedy	0.0166	0.0039	0.0202
Bidikmisi	0.0272	0.002	0.0115
Keluhan	0.0215	0.0035	0.0673
Catering	0.008	0.0121	0.0481
Penundaan	0.0004	0.0283	0.0289
Reg Bidikmisi	0.0756	0.0529	0.0117
Wisuda	0.0095	0.0432	0.0146
MRE	**0.0351**	**0.023**	**0.0275**

Untuk membandingkan skrip/metode mana yang lebih rumit, maka diukur juga waktu proses dari masing-masing metode yang yang dapat dilihat pada tabel 8.

Tabel 8. Hasil Perbandingan Waktu Proses Semua Metode

Aplikasi	Waktu (ms)		
	FPA	Mamdani	Tsukamoto
Akomodasi	0.8	3.4	1.4
Avenger	1	3.5	1.2
Beasiswa	0.9	3.5	2.5
Bidikmisi	1.1	3.2	2.6
Catering	1.2	3.6	1.4
IMB	0.7	2.3	1.6
iSpeedy	1.4	3.2	1.9
Keluhan	1.1	2.5	2.3
Meetingroom	1	2.2	2.1
Penundaan	1.6	2.5	2.1
Profile	1.2	2.7	1.5
Reg Bidikmisi	1.2	3.6	2.2
Wisuda	1.1	2.4	1.6
Rata-rata	1.1	3.0	1.9

Hasil waktu proses sesuai dengan tabel 8 didapatkan waktu proses yang paling kecil dengan metode FPA yaitu 1.1 ms sedangkan dengan metode Tsukamoto didapatkan waktu proses 1.9 ms dan yang terbesar yaitu dengan metode Mamdani yaitu 3 ms.

4. KESIMPULAN DAN SARAN

4.1. Kesimpulan

Metode software engineering seperti logika *fuzzy* dapat digunakan untuk meningkatkan akurasi metode FPA. Hal ini dapat dibuktikan dengan penelitian ini, dengan memodifikasi nilai function point complexity weight pada FPA dengan menggunakan kedua metode logika *fuzzy* yaitu metode Mamdani dan Tsukamoto terbukti memiliki nilai rata-rata error relatif (MRE) lebih kecil daripada

metode FPA tanpa modifikasi dengan logika *fuzzy*.

Dengan menghitung rata-rata error relatif pada ke-13 data yang digunakan dapat diketahui bahwa dengan memodifikasi FPA dengan menggunakan logika *fuzzy* metode Mamdani didapatkan rata-rata error relatif terkecil yaitu 2,3%, sedangkan dengan menggunakan metode Tsukamoto didapatkan MRE sebesar 2,75%, dan dengan menggunakan FPA tanpa modifikasi didapatkan MRE sebesar 3.51%. Hasil rata-rata waktu proses didapatkan waktu proses yang paling kecil dengan metode FPA yaitu 1.1 ms sedangkan dengan metode Tsukamoto didapatkan waktu proses 1.9 ms dan yang terbesar yaitu dengan metode Mamdani yaitu 3 ms.

4.2. Saran

Data yang memiliki standard sebagai data simulasi metode sebaiknya menggunakan data aplikasi yang dikerjakan oleh developer atau dengan menggunakan aplikasi-aplikasi yang dibuat secara profesional. Dari pengembangan function point complexity weight bisa dicoba dengan menggunakan metode yang lain agar tingkat akurasinya bisa meningkat.

Selain itu dengan dihasilkannya nilai FP maka mendapatkan estimasi perangkat lunak tidak hanya terbatas pada LOC. Dengan nilai FP bisa didapat pula biaya, waktu pengerjaan, dan sumber daya yang harus dikeluarkan agar proses pengerjaan sebuah proyek perangkat lunak akan berjalan efektif dan efisien.

5. DAFTAR PUSTAKA

[1] Balaji, N., Shivakumar, N., & Ananth, V. V. 2013. *Software Cost Estimation using Function Point with Non Algorithmic Approach*. Global Journal of Computer Science and Technology Software & Data Engineering.

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

[3] Saptono, Ristu., & Dian Hutama, Galih 2015. *Peningkatan Akurasi Estimasi Ukuran Perangkat Lunak dengan Menerapkan Logika Fuzzy Metode Mamdani.* Scientific Journal of Informatics.

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

[5] Kusumadewi, S. 2003. *Penentuan Tingkat Resiko Penyakit menggunakan Tsukamoto Fuzzy Inference System* .Yogyakarta:Teknik Informatika Universitas Islam Indonesia.

[6] Pratiwi, Dian. 2013. *Implementation of Function Point Analysis in Measuring the Volume Estimation of Software System in Object Oriented and Structural Model of Academic System.* International Journal of Computer Applications (0975 – 8887) Volume 70– No.10, May 2013

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