PENGEMBANGAN *FUNCTION POINT COMPLEXITY WEIGHT* DENGAN *FUZZY LOGIC* METODE MAMDANI

Galih Dian Hutama [1] Ristu Saptono [2] Hasan Dwi Cahyono [3]

[1) 2) 3)]Informatika, FMIPA Universitas Sebelas Maret
Jalan Ir. Sutami 36A Kentingan, Surakarta 57126 Indonesia
email : [1)]galihdhutama@student.uns.ac.id, [2)]ristu.saptono@staff.uns.ac.id,
[3)]hasandc@staff.uns.ac.id

Abstract

Measuring a software is the first step to do to build a software efficiently, effectively and give a good result. One of the part of measuring a software is by counting the software size trough Lines of Code (LOC). There are several methods for counting LOC and Function Point Analysis (FPA) is one of the method commonly used for counting LOC of a software. FPA method need to combine with another method to get a better accuracy. In this paper, fuzzy logic used to develop the weight in Function Point Complexity Table. The data is get from several applications and used as a simulation for LOC counting. LOC from FPA and fuzzy logic is being compared with real LOC of the application to get the deficit LOC. Mean Absolute Error (MAE) used as a error measuring method. The result are MAE for FPA method is 154.82 lines and MAE for FPA with fuzzy logic is 93.44 lines. This prove fuzzy logic can make LOC counting with FPA give less error which means is a better accuracy.

Keywords *— Lines of Code, Function Point Analisys, Fuzzy Logic.*

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. (Tunali, 2014)

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). 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.

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), *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*(Balaji et al., 2013). Pengelompokan *weight* tersebut sesuai dengan tabel *Function Point Complexity Weight* yang dibuat oleh Allan Albrecht(Xia et al., 2008).

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 memilki 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.

Dalam penelitian ini, *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(Elamvazuthi et al., 2009). 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* di penelitian ini 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(Kaur and Kaur, 2012). Output dalam proses *defuzifikasi* metode Mamdani berupa nilai ganda atau *range* sehingga memungkinkan metode Mamdani untuk menghasilkan output yang mempunyai toleransi tinggi karena mempertimbangkan nilai-nilai didekatnya. Penerapan *fuzzy logic* metode Mamdani dalam proses mendapatkan nilai weight pada tabel *Function Point Complexity Weight* membuat nilai *weight* yang didapat lebih akurat dan lebih sesuai.

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 dan efisien dari estimasi yang didapat menggunakan FPA dengan memperbaharui *weight* pada tabel *Function Point Complexity Weight* menggunakan *fuzzy logic* metode Mamdani.

2 DASAR TEORI

2.1 *Software Size Estimation*

Software Size Estimation merupakan pembelajaran untuk memperkirakan ukuran suatu perangkat lunak yang akan dikembangkan atau dibuat. Pengukuran perangkat lunak penting agar dalam pengerjaan perangkat lunak tidak membuang percuma resource yang ada dan menggunakannya semaksimal mungkin untuk mendapatkan proses pengerjaan yang efektif dan efisien. Ada beberapa metode dalam mengukur perangkat lunak, diantaranya :

- *Lines of Code* (LOC)
- *The Blitz*
- *Wideband Delphi*
- *Function Points*
- *Feature Points*
- *Object Points*
- *Number of Boxes on a Data Flow Diagram*
- *Number of Classes in a Design Diagram*

Meskipun ada banyak metode pengukuran perangkat lunak tetapi sulit untuk mendapatkan tingkat akurasi sebesar 100 persen untuk setiap metode. Diperlukan perpaduan metode untuk mendapatkan tingkat akurasi pengukuran perangkat lunak yang besar.

2.2 *Function Point Analysis*

Function Point Analysis (FPA) merupakan metode untuk mengukur pengembangan atau pembuatan perangkat lunak berdasarkan jumlah fungsionalitas dan komplesitas dari pandangan user. FPA diperkenalkan pada tahun 1986 oleh *International Function Point User Group* (IFPUG). IFPUG juga membuat *Counting Practices Manual* (CPM) sebagai standar industri dalam menggunakan FPA (Tunali, 2014).

Didalam FPA terdapat 5 fungsi pembeda yang menjadi pengitungan awal mendapatkan nilai *weight.* 5 fungsi tersebut adalah (Pradani, 2013) :

- *Internal Logical Files* (ILF) merupakan data atau informasi kontrol yang perlu dipelihara melalui pengaksesan *insert, update,* dan *delete* di dalam sistem aplikasi.
- *External Interface Files* (EIF) merupakan data atau informasi kontrol yang diperlukan oleh sistem aplikasi, tetapi disimpan atau diplehara diluar sistem oleh sistem lain.
- *External Input* (EI) merupakan proses elementer yang memproses data dari luar sistem dan menyimpan hasilnya ke dalam sistem.
- *External Output* (EO) merupakan proses elementer yang memproses data dan menyajikan informasi ke luar sistem dengan disertai proses komputasi atau kalkulasi ketika menyajikan informasi.
- *External Inquiry* (EQ) merupakan proses elementer yang menyajikan data ke luar sistem tanpa melakukan proses pengolahan data.

Penghitungan nilai ILF, EIF, EI, EQ, dan EO juga dipengaruhi oleh tingkat kompleksitas komponen-komponen tersebut. Pengkategorian komponen didasarkan pada penghitungan DET, RET, dan FTR yang rinciannya sebagai berikut (Pradani, 2013) :

- *Data Element Type* (DET) adalah elemen data yang dikenal oleh user dan merupakan *non repeatable data field.* Contoh : Nama Pelanggan, Nomor Pelanggan, Tanggal Lahir, dan lain-lain.
- *Record Element Type* (RET) adalah *subgroup* data yang dikenal oleh user. Contoh: data pelanggan terdiri dari biodata, data finansial, dan data tanggungan keluarga, dan lain-lain.
- *File Type Reference* (FTR) adalah ILF atau EIF yang dibaca atau diakses oleh proses elementer, yaitu EI, EQ, dan EO.

Dengan melakukan penghitungan ILF, EIF, EI, EQ, dan EO maka didapat nilai dari *Unadjusted Function Point* (UFP). Terdapat 14 faktor yang berfungsi sebagai hasil perkalian untuk menjadikan UFP menjadi *Function Point* (FP). 14 Faktor tersebut adalah (Pradani, 2013) :

- *Data Communications* merupakan tingkat kebutuhan komunikasi langsung antara aplikasi dengan processor.
- *Distributed Functions* merupakan tingkat kebutuhan transfer data antara komponen-komponen aplikasi.
- *Performances Objectives* merupakan tingkat response time dan throughput yang perlu dipertimbangkan dalam pengembangan aplikasi.
- *Heavily Used Configuration* merupakan tingkat kebutuhan dimana setting konfigurasi komputer berpengaruh terhadap pengembangan aplikasi.
- *Transaction Rate* merupakan tingkat transaksi bisnis yang berpengaruh terhadap pengembangan aplikasi.
- *On Line Data Entry* merupakan tingkat kebutuhan input data secara interaktif.
- *End User Efficiency* merupakan tingkat kemudahan penggunaan aplikasi.
- *On Line Update* merupakan tingkat kebutuhan ILF di update secara interaktif.
- *Complex Processing* merupakan tingkat kesulitan logika proses yang mempengaruhi proses development.
- *Reusability* merupakan tingkat kebutuhan aplikasi dan kode program dirancang dan dikembangkan untuk bisa digunakan pada aplikasi lain.
- *Installation Ease* merupakan tingkat kemudahan konversi ke sistem baru yang berpengaruh pada proses development.
- *Operational Ease* merupakan tingkat kemudahan aplikasi dalam aspek-aspek operasional, seperti startup, backup, dan proses recovery.
- *Multiple Sites* merupakan tingkat kebutuhan aplikasi dapat dioperasionalkan pada lingkungan hardware dan software yang berbeda-beda.
- *Facilitate Change* merupakan tingkat kemudahan aplikasi untuk modifikasi logika proses maupun struktur data.

Setelah didapatkan nilai FP, dapat dilakukan pengukuran pula terhadap *size, effort,* dan *resource* yang dibutuhkan dari perangkat lunak yang akan dikembangkan atau dibuat.

2.3 *Fuzzy Logic*

Fuzzy logic adalah peningkatan dari logika Boolean yang berhadapan dengan konsep kebenaran sebagian. Saat logika Boolean menyatakan bahwa segala hal dapat di ekspresikan dalam istilah biner (0 atau 1, hitam atau putih, ya atau tidak), *fuzzy logic* menggantikan kebenaran Boolean dengan tingkat kebenanaran (Naba, 2009).

Himpunan *fuzzy logic* pertamakali diperkenalkan oleh Lotfi A. Zadeh pada tahun 1965 sebagai cara matematis untuk merepresentasikan ketidakpastian linguistik. Berdasarkan konsep *fuzzy logic*, faktor-faktor dan kriteria-kriteria dapat diklasifikasikan tanpa batasan yang mengikat. *Fuzzy logic* sangat berguna untuk menyelesaikan banyak permasalahan dalam berbagai bidang yang biasanya memuat derajat ketidakpastian (Gokmen et al., 2010). Logika benar dan salah dari logika Boolean tidak dapat mengatasi masalah gradasi yang berada pada dunia nyata. Tidak seperti logika boolean, *fuzzy logic* mempunyai nilai yang berkelanjutan. *Fuzzy logic* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama.

Fuzzy logic merupakan generalisasi dari logika Boolean yang hanya memiliki dua nilai keanggotaan yaitu 0 dan 1. Dalam *fuzzy logic* nilai kebenaran suatu pernyataan berkisar dari sepenuhnya benar sampai dengan sepenuhnya salah (Meimaharani et al., 2014). *Fuzzy logic* berhubungan dengan ketidakpastian yang telah menjadi sifat alamiah manusia. *Fuzzy logic* memungkinkan nilai akhir berupa benar dan salah atau hitam dan putih, tetapi melibatkan area abu-abu.

2.4 *Fuzzy Inference System* Mamdani

Metode Mamdani sering juga dikenal dengan nama Metode Max-Min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan output diperlukan 4 tahapan (Hermawan and Putri, 2014), yaitu:

- Pembentukan Himpunan *Fuzzy* (Fuzzifikasi)

Himpunan *fuzzy* merupakan suatu pengembangan lebih lanjut tentang konsep himpunan dalam matematika. Himpunan *fuzzy* adalah rentang dari nilai-nilai. Masing-masing nilai mempunyai derajat keanggotaan (*membership*) antara 0 sampai dengan 1. Ungkapan logika Boolean menggambarkan nilai-nilai "benar" atau "salah". *Logika fuzzy* menggunakan ungkapan misalnya: "sangat lambat", "sedang", "sangat cepat" dan lain-lain untuk mengungkapkan derajat intensitasnya.

Logika *fuzzy* menggunakan satu set aturan untuk menggambarkan perilakunya. Aturan-aturan tersebut menggambarkan kondisi yang diharapkan dan hasil yang diinginkan dengan menggunakan fungsi IF... THEN. Suatu himpunan *fuzzy* A dalam semesta pembicaraan dinyatakan dengan fungsi keanggotaan (*membership function*) mu A, yang nilainya berada dalam interval [0,1]. Secara matematika hal ini dinyatakan dengan:

$$\mu_A : \cup \rightarrow [0,1]...(1) \quad (1)$$

Himpunan *fuzzy* A dalam semesta pembicaraan U biasa dinyatakan sebagai sekumpulan pasangan elemen u dan besarnya derajat keanggotaan (*grade of membership*) elemen tersebut sebagai berikut:

$$A = (\upsilon, \mu_A(\upsilon))/\upsilon \in \cup ...(2) \quad (2)$$

Tanda "/" digunakan untuk menghubungkan sebuah elemen dengan derajat keanggotaannya.

- Komposisi Aturan

Tidak seperti penalaran monoton, apabila sistem terdiri dari beberapa aturan, maka inferensi diperoleh dari kumpulan dan korelasi antar aturan. Ada 3 metode yang digunakan dalam melakukan inferensi sistem *fuzzy,* yaitu: *max, additive* dan *probabilistik* OR (probor).

- Penegasan (Defuzzifikasi)

Input dari proses defuzzifikasi adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy,* sedangkan output yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut. Sehingga jika diberikan suatu himpunan *fuzzy* dalam *range* tertentu, maka dapat diambil suatu nilai *crisp* tertentu sebagai output. Dalam metode Mamdani digunakan fungsi *Center of Gravity* (COG) untuk pengambilan nilai *crisp* tersebut. Formulasi untuk COG seperti dibawah ini:

$$Z_{cog} = \frac{\int_z^1 \mu_A(z) z dz}{\int_z^1 \mu_A(z) dz} \quad (3)$$

dimana mu A(z) adalah nilai *agregat* berdasarkan *rule* yang telah dibuat dari nilai input.

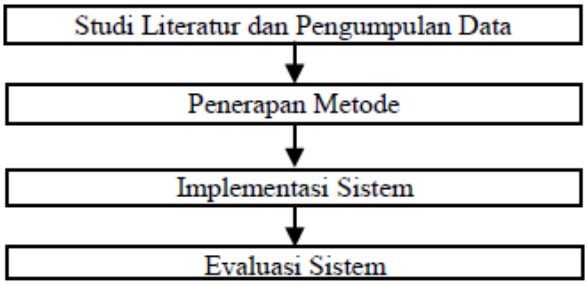
2.5 *Mean Absolute Error* (MAE)

Metode *Mean Absolute Error* (MAE) digunakan untuk menghitung *error* tiap metode. Semakin kecil nilai MAE yang didapat maka semakin kecil pula *error* yang dihasilkan metode tersebut. Formulasi metode MAE seperti dibawah ini:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| \quad (4)$$

dimana n adalah banyak item yang dihitung, x merupakan nilai prediksi dan y adalah nilai real.

3 METODOLOGI PENELITIAN



Gambar 1: Metodologi Penelitian

3.1 Studi Literatur dan Pengumpulan Data

Studi literatur dilakukan dengan membaca sejumlah jurnal baik nasional maupun internasional. Jurnal yang dibaca adalah jurnal dengan metode yang berkaitan dengan FPA dan *fuzzy logic.* Studi literatur juga bisa dilakukan dengan mempelajari modul-modul perkuliahan yang berkaitan dengan FPA dan *fuzzy logic* dari berbagai Universitas.

Survey dan pengumpulan 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.

3.2 Penerapan Metode

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*) terlihat pada Tabel 1.

Table 1: *Complexity Matrix for ILF and EIF* (Jones, 2008)

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.

Table 2: *Function Point Complexity Weight* (Jones, 2008)

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

Proses pengelompokan tersebut bersifat *crisp* sehingga akan diubah dengan *fuzzy logic* metode Mamdani. Pertama untuk proses fuzifikasi adalah membuat input *fuzzy sets* untuk RET/FTR dan DET. Didefinisikan sebagai *small, medium, large.* Didalam penelitian ini 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.* Terdapat 9 *rule* seperti yang terlihat pada Tabel 3.

Table 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 proses *defuzifikasi* dengan metode Mamdani menggunakan fungsi *Center of Gravity* (COG). Hasil dari proses defuzifikasi 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.

Kemudian mencari nilai *Total Degree of Influence* (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. 0 jika faktor tersebut tidak menimbulkan efek apapun dan 5 jika faktor tersebut sangat penting di perangkat lunak yang diukur. 14 faktor tersebut seperti terlihat pada Tabel 4.

Table 4: *Value Adjusment Factor*(Jones, 2008)

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 :

$$TCA = 0.65 + 0.01 * TDI \tag{5}$$

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

$$TCA = 0.65 + 0.01 * TDI \tag{6}$$

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 (Tunali, 2014).

Table 5: *Productivity Factor Programming Language*

Programming Language	Productivity
SQL	37
PHP	53
HTML/Javascript	58

Penghitungan LOC dalam penelitian ini menghasilkan 2 nilai. Nilai pertama adalah LOC yang didapat dari metode FPA dan nilai kedua adalah LOC yang didapat dari metode FPA dengan *fuzzy logic.* Hasil LOC dari masing-masing metode akan dijadikan nilai estimasi size suatu perangkat lunak yang diukur.

Dari 2 nilai LOC yang dihasilkan dihitung *Mean Absolute Error* (MAE) untuk setiap metode. Langkah pertama dihitung terlebih dahulu selisih LOC yang dihasilkan oleh setiap metode dengan LOC asli dari

aplikasi yang dihitung dengan *software SLOC Metrics* 3.0. Dari nilai selisih tersebut kemudian dapat dicari *Mean Abosulute Error* (MAE) untuk setiap metode.

3.3 Implementasi Sistem

Program dalam penelitian ini dibangun dengan bahasa pemrograman PHP. Didalam program terdapat 2 metode perhitungan, yaitu *fuzzy logic* metode Mamdani dan *Function Point Analysis* (FPA).

Untuk *fuzzy logic* metode Mamdani dimulai dari input jumlah atau banyak RET/FTR dan DET dalam perangkat lunak yang diukur. Setelah data diolah dengan *fuzzy logic* metode Mamdani muncul angka yang menjadi nilai *weight.*

Dari nilai *weight* tersebut dilakukan penghitungan metode FPA. Hasil dari penghitungan metode FPA tersebut adalah nilai FP yang bisa digunakan untuk mengukur size dari perangkat lunak yang berupa *Lines of Code* (LOC).

3.4 Evaluasi Sistem

Simulasi yang dilakukan adalah mengukur beberapa aplikasi dari hasil Kerja Praktek Mahasiswa S1 Informatika UNS angkatan 2011 dan aplikasi yang dipakai oleh Pemerintah Kabupaten Surakarta. Dari hasil simulasi kemudian didapat nilai *Function Point* (FP) yang bisa digunakan untuk mengukur *Lines of Code* (LOC) perangkat lunak. Dari hasil estimasi yang didapat bisa diketahui error terhadap hasil dari kejadian real. Nilai *error* tersebut yang berfungsi sebagai bahan evaluasi sistem.

4 PEMBAHASAN

Penelitian ini menggunakan data penelitian berupa aplikasi hasil Kerja Praktek mahasiswa jurusan Informatika FMIPA UNS angkatan 2011 dan aplikasi yang dipakai oleh Pemerintah Kabupaten Surakarta. Terdapat 6 aplikasi yang akan dijadikan sebagai data penelitian, yaitu: Akomodasi, *Avenger, Catering, IMB, iSpeedy,* dan *Meetingroom.* Data yang digunakan dari aplikasi adalah berupa *Data Element Types* (DETs), *Record Element Types* (RETs), *File Type References* (FTRs), dan *General System Characteristic* yang terdiri dari 14 faktor. Setelah semua data yang dibutuhkan terkumpul maka dilakukan metode *Function Point Analysis* (FPA) dan *fuzzy logic* untuk mendapatkan *Lines of Code* (LOC) aplikasi.

4.1 Hasil*Lines of Code*(LOC)

Sebelum dilakukan penghitungan LOC menggunakan FPA dan *fuzzy logic*, dilakukan terlebih dahulu penghitungan LOC *real* dari aplikasi sebagai pembanding hasil yang didapat dari metode FPA dan *fuzzy logic* menggunakan *software SLOC Metrics* 3.0. Hasil dari *SLOC Metrics* 3.0 berupa LOC *real* dari aplikasi dapat dilihat pada tabel 6.

Table 6: LOC *Real* Aplikasi

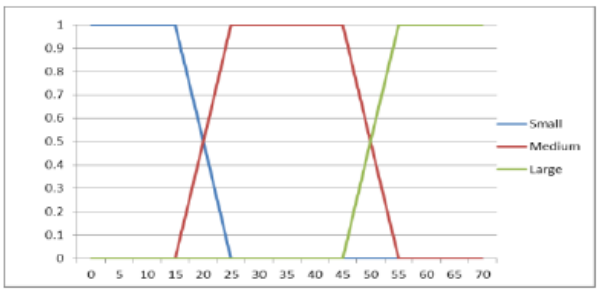
Aplikasi	**LOC (*lines*)**
Akomodasi	4089
Avenger	2468
Catering	4172
IMB	3341
iSpeedy	5310
Meetingroom	3520

Selanjutnya dilakukan penghitungan LOC dengan menggunakan metode FPA. Hasil LOC dari metode FPA seperti pada tabel 7.

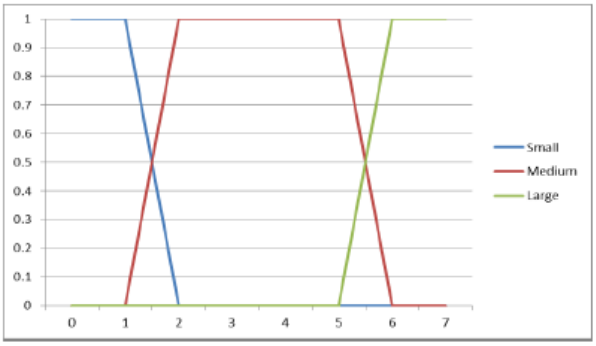
Table 7: LOC dengan Metode FPA

Aplikasi	**LOC (*lines*)**
Akomodasi	3891.6
Avenger	2340.9
Catering	3707.36
IMB	3290.4
iSpeedy	5296.5
Meetingroom	5296.5

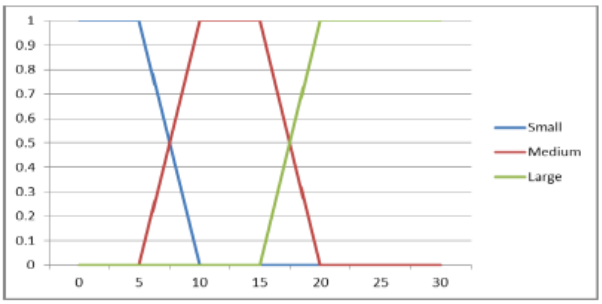
Untuk menggunakan metode FPA dengan *fuzzy logic* sebagai pengembangan nilai *weight* pada tabel *Function Point Complexity Weight* dilakukan terlebih dahulu membuat fuzifikasi sesuai dengan *fuzzy sets* yang telah dibuat. Fuzifikasi terhadap nilai DET, RET, dan FTR dengan menggunakan fuzzy logic seperti pada beberapa gambar dibawah ini.



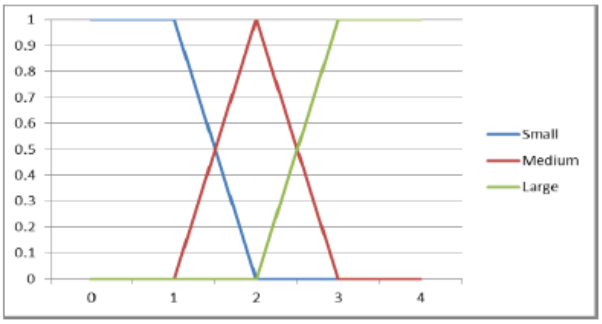
Gambar 2: Diagram Fuzifikasi DET pada ILF dan EIF



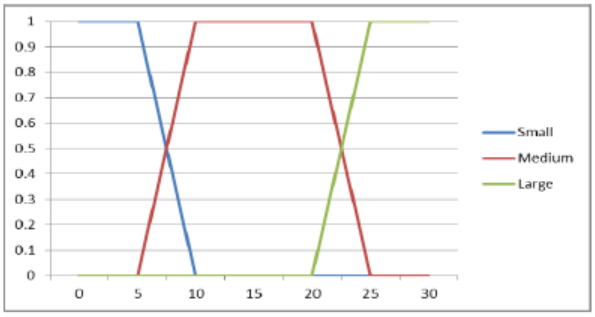
Gambar 3: Diagram Fuzifikasi RET pada ILF dan EIF



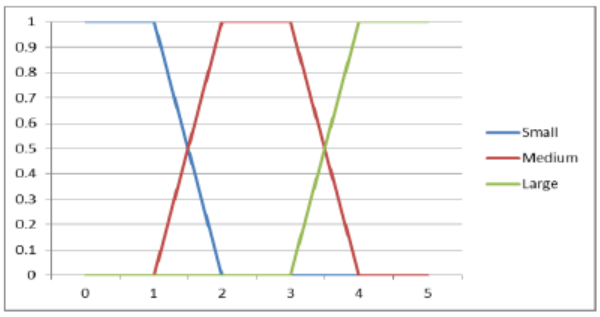
Gambar 4: Diagram Fuzifikasi DET pada EI



Gambar 5: Diagram Fuzifikasi RET pada EI

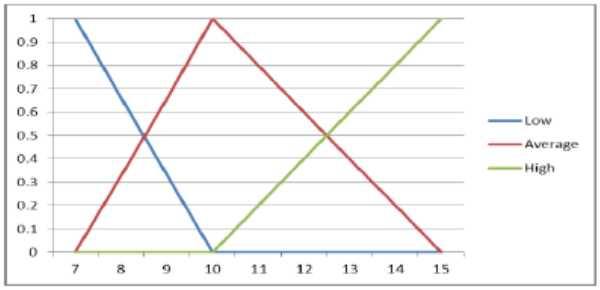


Gambar 6: Diagram Fuzifikasi DET pada EO dan EQ

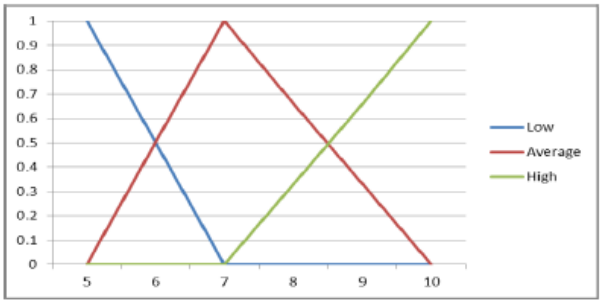


Gambar 7: Diagram Fuzifikasi RET pada EO dan EQ

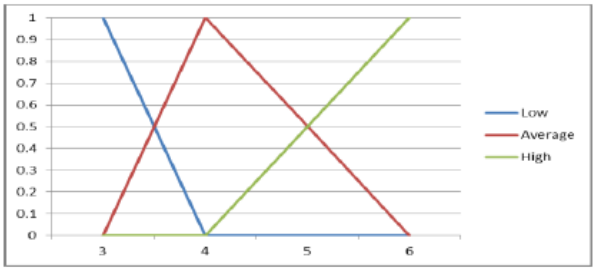
Dengan menggunakan metode Mamdani sebagai *Fuzzy Inference System* maka output dalam proses defuzzifikasi berupa nilai *range*, seperti yang ditunjukan oleh beberapa gambar dibawah ini.



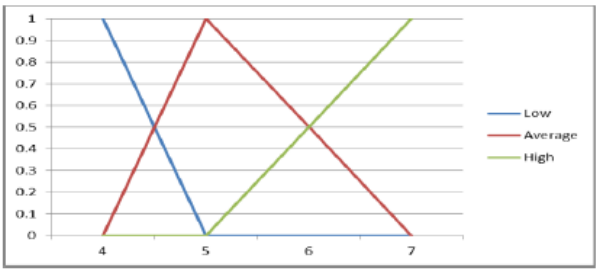
Gambar 8: Diagram Defuzifikasi pada ILF



Gambar 9: Diagram Defuzifikasi pada EIF



Gambar 10: Diagram Defuzifikasi pada EI dan EQ



Gambar 11: Diagram Defuzifikasi pada EO

Untuk hasil LOC dari metode FPA dengan *fuzzy logic* seperti terlihat pada tabel 8.

Table 8: LOC dengan Metode FPA Menggunakan Fuzzy Logic

Aplikasi	**LOC (*lines*)**
Akomodasi	4008.23
Avenger	2441.97
Catering	3855.92
IMB	3374.34
iSpeedy	5217.81
Meetingroom	3507.79

4.2 Pembandingan Hasil

Setelah semua hasil LOC didapat dari metode FPA dan *fuzzy logic* maka dilakukan pembandingan hasil diantara semua metode tersebut, dengan cara mencari nilai *Mean Absolute Error* (MAE) dari selisih nilai LOC yang didapat dari metode FPA dan *fuzzy logic* dengan nilai LOC real yang didapat dari *software SLOC Metrics* 3.0. Hasil MAE untuk setiap metode seperti terlihat di tabel 9.

Table 9: LOC dengan Metode FPA Menggunakan Fuzzy Logic

Metode	**MAE (*lines*)**
FPA	154.82
Fuzzy Logic	93.437

5 PENUTUP

5.1 Kesimpulan

Metode *Software Size Estimation* tidak ada yang memberikan tingkat akurasi sebesar 100 persen, dengan penambahan metode *software engineering*.

5.2 Saran

Penggunaan metode *software engineering* pada FPA untuk memprediksi size sebuah aplikasi tidak terbatas pada *fuzzy logic*. Sebagai contoh, terdapat *Artificial Intelligent System* yang menggunakan data penghitungan sebelumnya sebagai nilai pembanding untuk hasil yang dikeluarkan.

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. Perlu dilakukan penelitian untuk mendapatkan nilai estimasi selain LOC dengan menggunakan FP.

REFERENSI

Balaji, N., Shivakumar, N., and Ananth, V. (2013). Software Cost Estimation using Function Point with Non Algorithmic Approach. *Global journal of computer science and technology*.

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*, 6.

Gokmen, G., Akinci, T. C., Tektas, M., Onat, N., Kocyigit, G., and Tektas, N. (2010). Evaluation of student performance in laboratory applications using fuzzy logic. *Procedia - Social and Behavioral Sciences*, 2(2):902–909.

Hermawan, L. and Putri, A. (2014). Penerapan Algoritma Fuzzy Mamdani untuk Mengatur Game Scoring pada Game Helitap.

Jones, C. (2008). *Applied Software Measurement: Global Analysis of Productivity and Quality*. McGraw-Hill, 3rd edition.

Kaur, A. and Kaur, A. (2012). Comparison of Mamdani-Type and Sugeno-Type Fuzzy Inference Systems for Air Conditioning System. (2):323–325.

Meimaharani, R., Teknik, D. F., Studi, P., Informatika, T., Kudus, U. M., Listyorini, T., Teknik, D. F., Studi, P., Informatika, T., and Kudus, U. M. (2014). Analisis Sistem Inference Fuzzy Sugeno dalam Menentukan Harga Penjualan Tanah Untuk Pembangunan Minimarket. 5(1):89–96.

Naba, D. E. A. (2009). *Belajar Cepat Fuzzy Logic Menggunakan MatLab*. I, 1st pub edition.

Pradani, W. (2013). Kajian Metode Perhitungan Metrik Function-Point dan Penerapannya pada Dua Perangkat Lunak yang Dipilih. 2(1):28–34.

Tunali, V. (2014). Software Size Estimation Using Function Point Analysis – A Case Study for a Mobile Application.

Xia, W., Capretz, L., Ho, D., and Ahmed, F. (2008). A New Calibration for Function Point Complexity Weights. *Information and Software Technology*, 50:670–683.