pISSN: 2087-8893
eISSN: 2527-3671

PERHITUNGAN UKURAN KOMPLEKSITAS FUNGSIONAL PERANGKAT LUNAK DENGAN METRIK *FUNCTION POINT*

Yuni Widyaningtyas[1)], Achmad Arwan[2)], dan Denny Sagita Rusdianto[3)]
[1,2,3)]Program Studi Informatika Fakultas Ilmu Komputer Universitas Brawijaya
Jl. Veteran No. 8 Malang, Jawa Timur, Indonesia 65145

e-mail: yuni.widyaningtyas@gmail.com[1)], arwan@ub.ac.id [2)], denny.sagita@ub.ac.id[3)]

ABSTRAK

Function Point (FP) adalah metrik yang digunakan untuk mengukur kompleksitas perangkat lunak berdasarkan fungsi perangkat lunak tersebut. Perhitungan metrik FP dengan menggunakan desain DFD (*Data Flow Diagram*) dan ERD (*Entity Relationship Diagram*) masih dilakukan secara manual. Dengan perhitungan manual tersebut terdapat kesalahan sebesar 90% yang dilakukan oleh siswa yang belum lulus dalam mengidentifikasi *logic file*. Berdasarkan kekurangan tersebut, maka dibuatlah program yang otomatis menghitung FP dengan harapan mengurangi kesalahan perhitungan FP. Sistem perhitungan FP berdasarkan desain DFD dan ERD dibuat berbasis web dengan menggunakan *framework* Codeigniter. Sistem yang telah dibuat diuji dengan menggunakan *white-box* dan *black-box testing*. Teknik yang digunakan dalam pengujian yaitu, *basis path testing*, pengujian dataset, dan pengujian validasi. Dari pengujian tersebut didapatkan bahwa kebutuhan fungsional dan non-fungsional telah berjalan sesuai dengan yang diharapkan pada tahap analisis kebutuhan, dan aplikasi ini telah sesuai dengan perhitungan manual yang telah dibuat untuk menghitung nilai FP.
Kata Kunci: DFD, ERD, *framework* Codeigniter, Function Point.

ABSTRACT

Function Point is a metric that used for measure software complexity based on software functional. FP metric computation using DFD and ERD design have done manually. There are 90% error when identify logic file in FP metric computation manually by an undergraduate student. Based on that fact, so this research is about developing a system that can be use to count the FP metric, indeed to decreasing error in FP computation. Function Points counting system is count the FP from DFD and ERD design. This system is developed in web base using CodeIgniter Framework. The system that has been developed is tested using white-box and black-box testing. Technique that is used in testing are basis path testing, dataset testing, and validation testing. From that tests, we know that the functional and non functional process has been meet with the functional and non functional requirement and the result of FP count using this system has the same with the FP count manually.
Keywords: Codeigniter *framework, DFD, ERD, Function Point.*

I. PENDAHULUAN

SEBUAH proyek perangkat lunak perlu adanya sebuah manajemen, karena rekayasa perangkat lunak profesional selalu mempersoalkan anggaran belanja organisasional dan batasan jadwal. Perencanaan proyek yang dibuat pada awal proyek digunakan untuk mengkomunikasikan kepada tim pengembang dan *customer*, bagaimana pekerjaan dapat selesai. Hal yang perlu dikomunikasikan tersebut antara lain adalah perkiraan sumber daya yang dibutuhkan dan penjadwalan pekerjaan.

Perkiraan sumber daya yang dibutuhkan berkaitan dengan ukuran perangkat lunak yang dibangun. Ukuran perangkat lunak yaitu suatu hal yang dapat dihitung dari sebuah proyek perangkat lunak. Ada dua pendekatan pengukuran, yaitu pengukuran langsung dan pengukuran tidak langsung. Pengukuran langsung dapat dihitung dengan menggunakan *Lines of Code* (LOC), sedangkan pengukuran tidak langsung dapat diwakili dengan *Function Point* (FP) [1]. Pengukuran LOC adalah pengukuran perangkat lunak berdasarkan bahasa pemrograman yang digunakan [2]. Hasil pengukuran dengan menggunakan LOC akan menghasilkan hasil yang berbeda untuk bahasa pemrograman yang berbeda. Hal ini membuat pengukuran menggunakan FP dinilai lebih dinamis karena FP mengukur berdasarkan fungsionalitas perangkat lunak.

FP adalah metode yang memperkirakan kompleksitas fungsional, dilihat dari sudut pandang pengguna. FP diperkenalkan oleh Allan J. Albrecht. Metode ini dikelola oleh IFPUG (*International Function Point Users Group*) dan dipakai sebagai standar ISO dan IEEE [3]. FP dapat diaplikasikan pada beberapa hal, yaitu studi banding, perkiraan biaya pengembangan, perkiraan biaya perawatan, kontrak *outsource*, perkiraan kualitas, pengukuran kualitas, dan lain-lain [4].

Penelitian yang pernah dilakukan oleh Jose Antonio Pow-Sang dan kawan-kawan dalam makalah *A Conversion Model and a Tool to Identify Function Point Logic using UML Analysis Class Diagrams* membahas mengenai identifikasi *logic file* dari sebuah *class diagram*. Dalam penelitian tersebut menyebutkan bahwa siswa yang belum lulus melakukan kesalahan perhitungan parameter *logic file* sebesar 90% untuk mengidentifikasi relasi antar tabel, 60% untuk identifikasi *Record Element Type* (RET), dan 80% pada identifikasi *Data Element Type* (DET). Angka tersebut didapatkan pada pengujian identifikasi *logic file* yang dilakukan oleh 10 orang praktisi. Bahkan kesalahan identifikasi *logic file* yang dilakukan oleh 17 orang mahasiswa yang belum

lulus mencapai angka 100% untuk identifikasi relasi antar tabel dalam *logic file*, 100% pada identifikasi RET, dan 70,5% pada identifikasi DET [5].

Berdasarkan kekurangan perhitungan FP secara manual yang telah disebutkan di atas, maka dibuatlah program yang dapat otomatis menghitung nilai FP dengan harapan kesalahan perhitungan FP dapat berkurang. Aplikasi yang akan dibuat akan menganalisa desain permodelan terstruktur yaitu *Data Flow Diagram* (DFD) dan *Entity Relationship Diagram* (ERD). Metode pengembangan yang dipakai dalam penelitian ini adalah berorientasi obyek. Hasil pengukuran kompleksitas fungsional perangkat lunak ini diharapkan dapat menjadi dasar perkiraan perencanaan proyek dan *user* dapat membangun sebuah proyek perangkat lunak yang lebih baik dengan mempergunakan sumber daya yang sesuai.

Tujuan penelitian ini untuk mengetahui proses mendapatkan nilai FP dari desain ERD dan DFD secara otomatis dan untuk mengetahui tingkat keberhasilan otomatisasi perhitungan ukuran kompleksitas fungsional perangkat lunak berdasarkan DFD dan ERD dengan metrik FP dari segi kecepatan dan ketepatan hasil yang didapatkan.

Batasan masalah yang dibahas dalam penelitian ini adalah DFD dan ERD yang dibuat harus menggunakan aplikasi Visual Paradigm versi 5 dan hanya terdiri dari satu DFD dan ERD, DFD yang digunakan adalah DFD yang sudah melalui proses dekomposisi, serta DFD dan ERD di-*export* menjadi format *.xml*.

II. LANDASAN KEPUSTAKAAN

A. *Function Point*

Function point (FP) adalah ukuran suatu perangkat lunak layaknya meter untuk ukuran panjang, celcius untuk ukuran temperatur, dan sebagainya. FP analisis adalah sebuah metode untuk memecah sistem ke dalam komponen yang lebih kecil, sehingga dapat mudah dipahami dan dianalisis [6].

Prosedur perhitungan FP adalah sebagai berikut:

1. Menentukan tipe perhitungan FP
2. Menentukan lingkup perhitungan dan *application boundary*
3. Mengukur *data function*
   Data function dibagi menjadi dua elemen, yaitu yaitu ILF (*Internal Logical File*) dan EIF (*External Interface File*). ILF adalah data atau kendali informasi yang berrelasi secara logis dan dikelola di dalam *application boundary*. EIF adalah data atau kendali informasi yang berrelasi secara logis dan dirujuk oleh aplikasi tetapi dikelola diluar *application boundary*. Sebuah EIF adalah ILF di aplikasi lain.
4. Mengukur *transactional function*
   Transactional function adalah EI (*External Input*), EQ (*External Inquiries*), dan EO (*External Output*). EI adalah proses dasar yang memproses data atau kendali informasi yang datang dari luar *application boundary*. EQ adalah proses dasar yang mengirim data atau kendali informasi di luar *application boundary*. Sedangkan EO adalah proses dasar yang mengirim data atau kendali informasi di luar *application boundary*. Maksud utama dari EO adalah memberikan informasi pada pengguna melalui pemrosesan logika.
5. Hitung *functional size*/*Unadjusted Function Point* (UFP)
6. Mengukur *Value Adjusment Factor*
7. Hitung *Adjusted Function Point* (AFP)

B. *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) adalah representasi grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan sebagai perpindahan data dari *input* ke *output*. DFD digunakan untuk merepresentasikan sebuah sistem atau perangkat lunak pada level abstraksi. DFD menyediakan mekanisme untuk permodelan fungsional dan juga permodelan aliran informasi [1].

C. *Entity Relationship Diagram* (ERD)

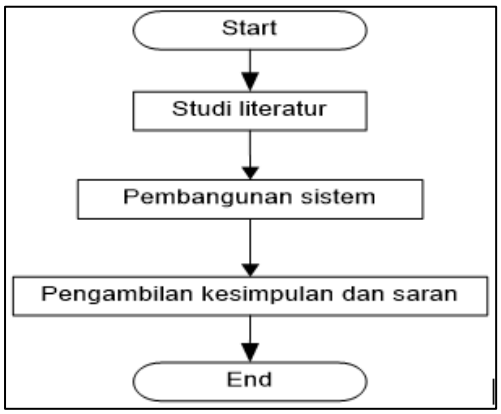
Entity Relationship Diagram (ERD) adalah gambar yang menampilkan informasi yang dibuat, disimpan, dan digunakan dalam sebuah sistem bisnis. Dalam sebuah ERD, jenis informasi yang sama, diletakkan bersama di dalam sebuah *box* bernama entitas. Garis yang digambar di antara entitas menggambarkan hubungan antara data dan tanda khusus yang ditambahkan pada diagram untuk mengkomunikasikan aturan bisnis *high-level* yang butuh didukung oleh sistem. ERD tidak menyatakan sebuah urutan, walaupun entitas yang berrelasi biasanya diletakkan berdekatan [2].

D. *eXtensible Markup Language* (XML)

Kepanjangan dari XML adalah *eXtensible Markup Language*. XML hampir sama dengan HTML, XML menggunakan elemen yang ditandai dengan tag pembuka '<' dan penutupnya '>' dan elemen atribut dinyatakan dalam tag pembuka, misal <form name="isidata">. Tag dalam XML lebih dinamis karena dibuat sesuai dengan kehendak yang membuatnya [7]. *Parsing* XML pada PHP menggunakan *library simple* XML. *Library* ini sudah disediakan oleh PHP 5. Teknik *parsing* XML dapat dilihat pada website php.net.

III. METODOLOGI PENELITIAN

Pada sub ini menjelaskan mengenai langkah-langkah yang ditempuh dalam pembangunan sistem perhitungan ukuran kompleksitas fungsional perangkat lunak dengan metrik FP. Diagram alir metodologi penelitian dapat dilihat pada Gambar 1 dan Gambar 2.



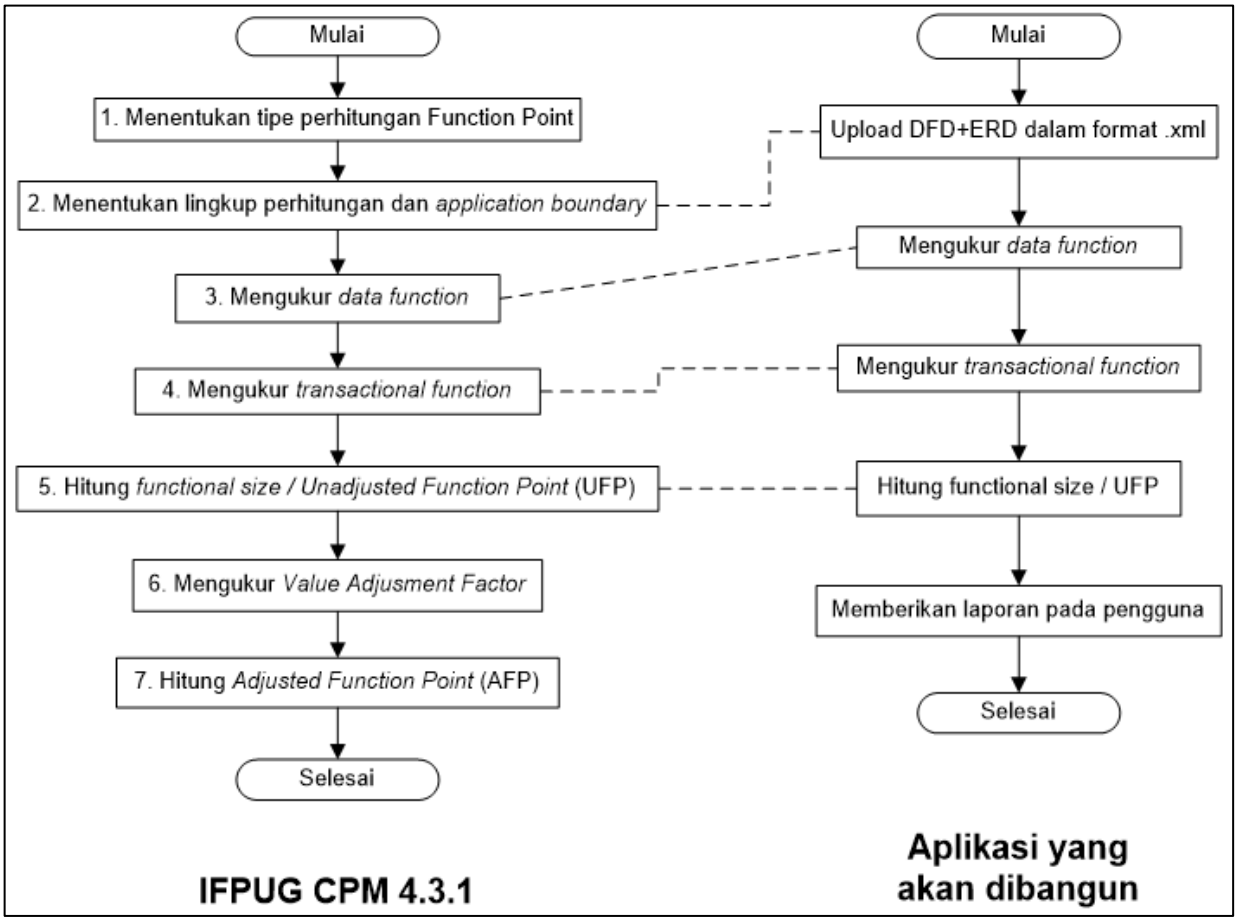
Gambar 1. Alur Penelitian

A. Pembangunan Sistem

Dalam pembangunan sistem ini menggunakan metode siklus hidup *waterfall*. Tahap-tahap pengembangan sistem menggunakan metode *waterfall* adalah analisis kebutuhan, analisis sistem (perancangan), *coding* (implementasi), pengujian, implementasi, dan tahap terakhir yaitu operasi dan perawatan. Dalam penelitian ini siklus hidup pengembangan sistem dilakukan hanya sampai tahap pengujian.

1) Analisis Kebutuhan
   Analisis kebutuhan adalah proses penggalian kebutuhan sistem yang akan dibangun. Proses ini diawali dengan mengidentifikasi permasalahan umum yang diangkat sebagai dasar pembuatan sistem, selanjutnya mendeskripsikan sistem secara umum, fungsi perangkat lunak yang dibuat, lingkungan operasi sistem. Tahap selanjutnya yaitu mengidentifikasi kebutuhan fungsional dan non fungsional sistem. Dari kebutuhan fungsional dan non-fungsional tersebut dibuat permodelan kebutuhan dan mendeskripsikan skenario kebutuhan fungsionalitas sistem.
2) Perancangan Sistem
   Tahap berikutnya adalah perancangan sistem. Perancangan sistem ini adalah proses memodelkan alur kerja sistem untuk dapat memenuhi kebutuhan sistem. Terdapat tiga perancangan, yaitu perancangan arsitektur, perancangan data, dan perancangan antarmuka.
3) Implementasi
   Alur kerja sistem secara umum adalah sebagai berikut:
   1. Membangun sistem perhitungan FP berbasis web.
   2. Mengimplementasikan perhitungan FP untuk mendapatkan ukuran kompleksitas fungsional perangkat lunak.
   3. Membuat DFD dan ERD pada program Visual Paradigm *for* UML 8.0 *Enterprise Edition.*
   4. Sistem melakukan perhitungan secara otomatis pada diagram yang diunggah tersebut.
   5. Sistem menampilkan hasil perhitungan kepada pengguna.

   Berikut adalah algoritma sistem yang akan dibangun berdasarkan teori perhitungan FP (Gambar 2) yang telah dijelaskan pada landasan kepustakaan. Perhitungan yang dilakukan dalam sistem ini adalah jenis FP pengembangan karena dalam FP jenis pengembangan yang menghitung FP dari fase penggalian kebutuhan sampai implementasi. Penentuan lingkup perhitugan dan *application boundary* dilakukan oleh pengguna, di mana pengguna memutuskan lingkup perhitungan dan *application boundary* dengan mengunggah DFD dan ERD yang akan dihitung untuk mendapatkan nilai FP.



Gambar 2. Pemetaan Teori Menjadi Implementasi Sistem

4) Pengujian
Pengujian sistem dilakukan untuk mengetahui apakah sistem berjalan dengan benar dan sesuai dengan kebutuhan yang didefinisikan di awal proses pembangunan sistem ini. Pengujian yang dilakukan untuk menguji kebutuhan fungsional adalah pengujian unit dan pengujian validasi. Untuk menguji kebutuhan non-fungsional, metode yang digunakan adalah metode pengujian performa. Pengujian performa akan menguji seberapa cepat sistem merespon permintaan *user*.

B. Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua proses studi literatur dan pengembangan sistem telah dilakukan. Kesimpulan yang didapatkan dibuat berdasarkan proses pengujian dan analisis pada sistem yang telah dibangun. Saran dituliskan dengan maksud agar penelitian selanjutnya dapat memperbaiki kelemahan dari penelitian ini.

IV. PERANCANGAN

A. Analisis Kebutuhan

1) Deskripsi Umum Sistem
Setelah melakukan analisis, peneliti mendapatkan fakta bahwa belum adanya sistem perhitungan ukuran kompleksitas fungsional perangkat lunak menggunakan metrik FP yang digunakan secara umum dan dilakukan secara otomatis. Masalah tersebut mendorong peneliti untuk membangun sistem perhitungan ukuran kompleksitas fungsional perangkat lunak secara otomatis dengan mengunggah DFD dan ERD dalam format file XML.
2) Karakteristik Pengguna
Sasaran pengguna sistem ini adalah para pengembang perangkat lunak dengan tidak menutup kemungkinan masyarakat umum juga dapat menggunakannya. Pengguna dapat menggunakan sistem ini untuk mengetahui ukuran kompleksitas fungsional perangkat lunak yang akan dibangun.

pISSN: 2087-8893
eISSN: 2527-3671

3) Daftar Kebutuhan

a) Kebutuhan Fungsional

Daftar kebutuhan fungsional dapat dilihat pada Tabel 1.

TABEL I
KEBUTUHAN FUNGSIONAL

No	Kode Fungsi	Nama Kebutuhan	Nama *Use case*
1	**SRS_001**	Pengguna meng-*upload* file XML yang merupakan hasil *export* dari *project* dari aplikasi Visual Paradigm. *Project* tersebut berisi DFD dan ERD.	*Upload* DFD dan ERD
2	**SRS_002**	Pengguna mendapatkan nilai ukuran kompleksitas fungsional perangkat lunak	Mendapatkan nilai ukuran kompleksitas fungsional

b) Kebutuhan Non Fungsional

Daftar kebutuhan noon fungsional dapat dilihat pada Tabel 2.

TABEL II
KEBUTUHAN NON-FUNGSIONAL

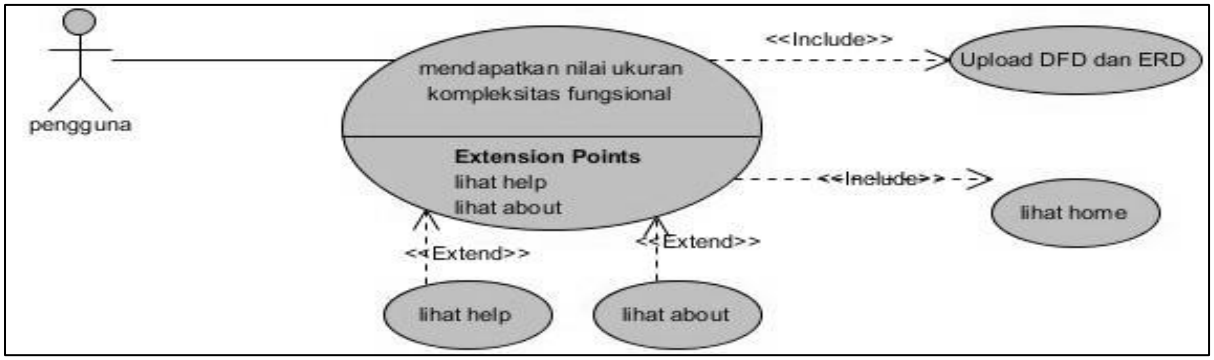
No	Kode Fungsi	Parameter	Deskripsi
1	**SRS_006**	Performa	Aplikasi dapat melakukan perhitungan FP dengan waktu kurang dari 2 menit.

B. Perancangan Sistem

1) Permodelan Kebutuhan

a. *Use case diagram*

Berdasarkan hasil analisis kebutuhan yang tertulis pada Tabel 1, dibuatlah permodelan kebutuhan sistem dengan menggunakan *use case diagram* (Gambar 3). Pengguna diharapkan dapat menggunakan sistem ini dengan fungsi mendapatkan nilai ukuran kompleksitas fungsional perangkat lunak.



Gambar 3. *Usecase Diagram*

b. *Use case skenario*

Use case skenario dapat dilihat pada Tabel 3.

TABEL III
USE CASE SKENARIO MENDAPATKAN NILAI UKURAN KOMPLEKSITAS FUNGSIONAL

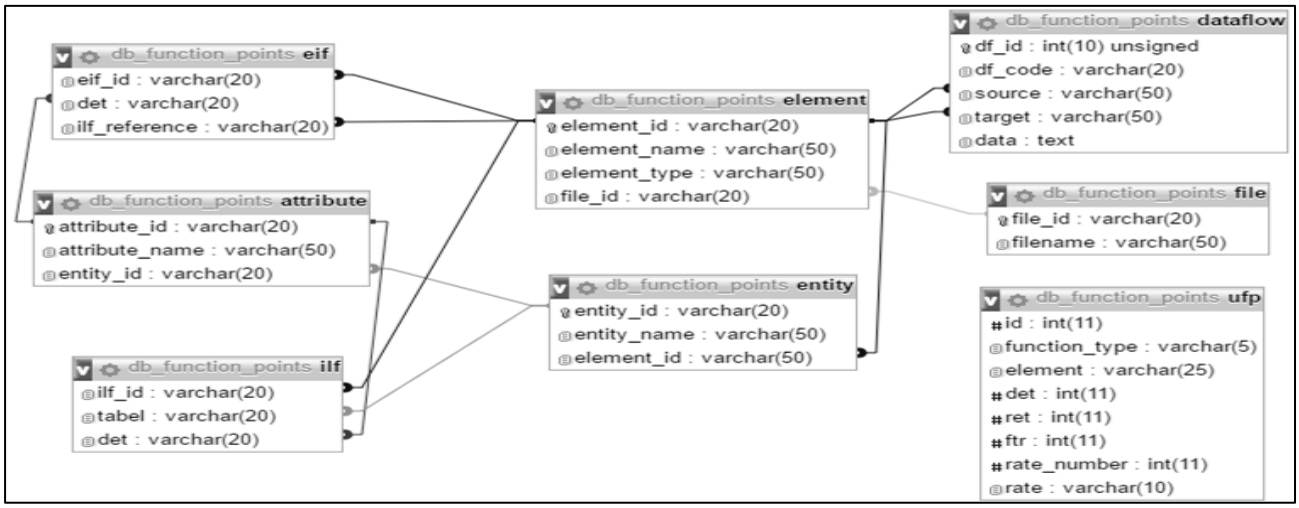
Flow of Events untuk *mendapatkan nilai ukuran kompleksitas fungsional*	
Objective	Untuk menampilkan hasil perhitungan FP kepada pengguna
Actor	*Pengguna*
References	*SRS-002*
Pre-condition	Pengguna berada di halaman *upload* DFD dan ERD
Main Flow	Pengguna menekan tombol 'Hitung' Sistem melakukan perhitungan DFP Sistem menampilkan hasil perhitungan DFP beserta rinciannya
Alternatif Flow	
Post-condition	Pengguna mendapatkan hasil perhitungan ukuran kompleksitas fungsional perangkat lunak.

C. Perancangan *Class*

1) *Class diagram*
   Class diagram digunakan untuk memberikan gambaran mengenai struktur obyek *class* dalam sistem perhitungan ukuran kompleksitas perangkat lunak. Struktur *class* ini menyesuaikan dengan *framework* yang digunakan yaitu *framework Model View Controller* (MVC).
2) *Class diagram*
   Langkah awal untuk mendapatkan nilai FP adalah pengguna akan berada pada halaman *upload*. Pengguna mengunggah file DFD dan ERD, lalu menekan tombol 'Hitung'. Dalam tombol 'Hitung' ini sistem akan memanggil fungsi hitung_UFP() yang berada dalam *class* Controller_FP. Fungsi hitung_UFP() akan melakukan perhitungan kompleksitas fungsional dari data DFD dan ERD yang telah diunggah oleh pengguna. Fungsi ini akan menampilkan hasilnya pada *boundary* View_Hasil_FP.

D. Perancangan Data

Perancangan data ini digunakan untuk menyimpan hasil *upload* DFD dan ERD dari pengguna dan hasil membaca file tersebut. Setiap komponen DFD dan ERD yang diperlukan dalam perhitungan FP disimpan dalam *database* untuk memudahkan proses perhitungan. Permodelan perancangan data dapat dilihat pada Gambar 4.



Gambar 4. Perancangan Data

V. IMPLEMENTASI

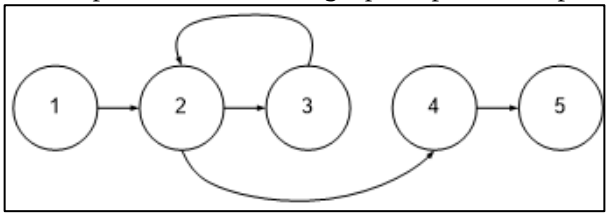
Proses *read* XML adalah proses untuk mengambil data yang diperlukan untuk perhitungan FP dari file XML yang telah diunggah. Proses ini mengidentifikasi data melalui *parsing* tag tertentu pada file XML. Proses *parsing* dilakukan menggunakan *library* yang sudah disediakan oleh PHP, yaitu simplexml, perhatikan Tabel 4.

VI. PENGUJIAN DAN ANALISIS

A. Pengujian *White-box*

1) Pengujian dan Analisis *Basis path testing*

Proses *read* XML adalah proses membaca dan menguraikan (*parse*) data yang tersimpan dalam file XML. Penguraian data tersebut berdasarkan tag-tag tertentu yang berada di dalam file. Proses ini menyeleksi tag yang dibutuhkan untuk keperluan perhitungan FP dan mengambil nilai dalam tag tersebut. Permodelan alur kerja proses read XML dapat dilihat pada Tabel 5. *Flow graph* dapat dilihat pada Gambar 5.



Gambar 5. *Flow graph read* XML

TABEL IV
PROSES READ XML

Nama Class : Controller_Additional	
Nama Method : read()	
1	load file xml;
2	delete history perhitungan sebelumnya dari database
3	jika tag modelType == DFDataStore, DFProcess, DFExternalEntity, DFDataflow, ModelRelationshipContainer
4	masukkan element xml ke dalam tabel element
5	jika tag modelType == DFProcess
6	jika tag FromSimpleRelationships == true
7	masukkan data dalam tag tersebut ke tabel dataflow
8	jika tag ToSimpleRelationships == true
9	masukkan data dalam tag tersebut ke tabel dataflow
10	jika tag modelType == ModelRelationshipContainer
11	jika tag ChildModels == true
12	masukkan data dari dataflow ke dalam tabel dataflow
13	jika tag modelType == DBTable
14	masukkan data dalam tag tersebut ke dalam tabel entity
15	jika tag modelType == DBColumn
16	masukkan data dalam tag tersebut ke dalam tabel attribute
17	Controller_ILF/identifikasi()

TABEL V
PERMODELAN ALUR KERJA PROSES *READ* XML

Nama Class	Controller_Additional
Nama Method	read()
mulai load file xml delete_history	1
mencari letak data DFD dan ERD{	2
simpan data dalam database	3
}	4
selesai	5

- *Cyclomatic Complexity*

  $V(G) = R = 2$
  $V(G) = P + 1 = 1 + 1 = 2$
  $V(G) = E - N + 2 = 5 - 5 + 2 = 2$
- Jalur Independen
  1. $1 - 2 - 3 - 2 - 4 - 5$
  2. $1 - 2 - 4 - 5$

Hasil dari pengujian *white-box* dengan teknik *basis path testing* menghasilkan bahwa kebutuhan fungsional telah berjalan sesuai dengan kebutuhan fungsional yang telah dideskripsikan pada tahap perancangan. Kesesuaian ini dilihat dari hasil yang dikeluarkan oleh sistem telah sesuai dengan hasil yang diharapkan pada kasus uji (Tabel 6).

2) Pengujian dan Analisis Dataset

Pengujian dataset dilakukan dengan uji coba perhitungan FP menggunakan aplikasi dengan 12 data uji. Data uji adalah *project* DFD dan ERD yang dibuat menggunakan aplikasi Visual Paradigm dan diekspor menjadi format file XML. Sample dataset uji dapat dilihat pada Tabel 7.

Pengujian dataset menggunakan 12 data dari beberapa sumber di internet. Data tersebut digolongkan menjadi 3 kriteria kompleksitas, yaitu rumit, sedang, dan kecil. Tiga kriteria tersebut berdasarkan jumlah proses yang terdapat dalam DFD. Kompleksitas kecil adalah DFD yang memiliki 1-4 proses, kompleksitas sedang adalah DFD yang memiliki 5-7 proses, dan kompleksitas rumit adalah DFD yang memiliki lebih dari 8 proses.

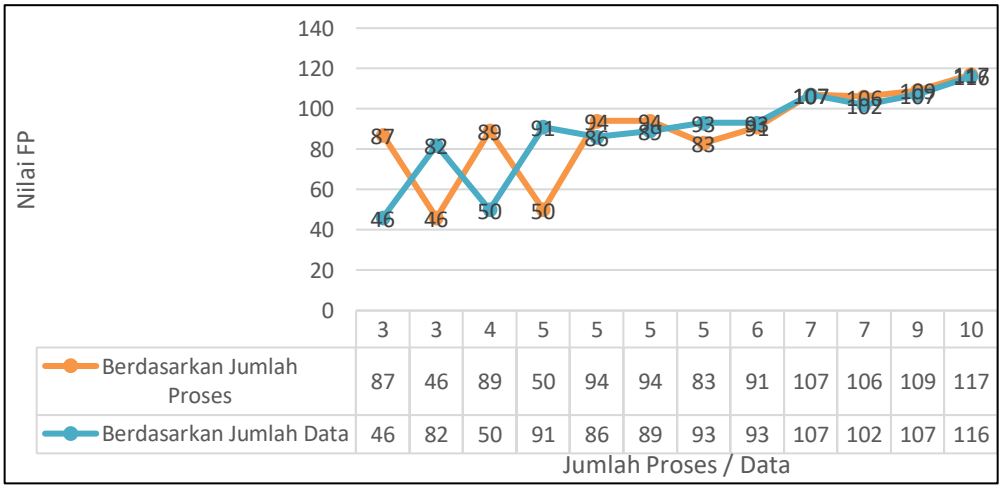
Percobaan pengujian dataset dengan pengelompokan kompleksitas berdasarkan jumlah proses yang terdapat dalam DFD ini menghasilkan grafik seperti pada Gambar 9. Nilai FP tidak berbanding lurus dengan jumlah proses secara konstan karena dalam FP, tidak hanya jumlah proses saja yang menjadi penentu nilai FP, tetapi juga terdapat jumlah data yang menentukan besarnya nilai FP.

TABEL VI
TEST CASE ALUR KERJA PROSES *READ* XML

No Jalur	Data *Input*	*Expected Result*	*Result*	Status
1	File XML mengandung tag modelType dengan nilai: DFDataStore, DFProcess, DFExternalEntity, DFDataFlow, ModelRelationshipContainer	Sistem menyimpan data hasil *parsing* ke dalam *database* dan melanjutkan pencarian data dalam tag	Sistem menyimpan data hasil *parsing* ke dalam *database* dan melanjutkan pencarian data dalam tag	Valid
2	File XML tidak mengandung tag modelType dengan nilai: DFDataStore, DFProcess, DFExternalEntity, DFDataFlow, ModelRelationshipContainer	sistem tidak melakukan proses	Sistem tidak melakukan proses	Valid

TABEL VII
SAMPLE DATASET UJI

No	Nama Data Uji	Nilai FP	Sumber DFD dan ERD	Analisis	Kriteria
1.	DFD HR FUN.xml	50	Lamma, Evelina. Mello, Paola. Riguzzi, Fabrizio. 2003. *A System for Measuring Function Points. Bologna.*	DFD terdiri dari 5 proses dan 7 data store. ERD terdiri dari 8 entitas	Sedang
2.	DFD Keuangan Daerah.xml	91	https://core.ac.uk/download/pdf/11734544.pdf	DFD terdiri dari 6 proses dan 6 data store. ERD terdiri dari 12 entitas	Sedang
3.	DFD Penerimaan Siswa 2.xml	46	http://s3.amazonaws.com/ppt-download/rancangbangunpenerimaan222-151007	DFD terdiri dari 3 proses dan 1 data store. ERD terdiri dari 7 entitas	Sedang
4.	DFD SuperSport 7.xml	87	Riguizzi, Fabrizio. 2003. *Specification of the Application SuperSport with* ER-DFD. Italy. (Proses 44-46)	DFD terdiri dari 3 proses dan 1 data store. ERD terdiri dari 15 entitas	Sederhana



Gambar 6. Hasil pengujian dataset

Hasil pengujian dataset dengan pengelompokan kompleksitas berdasarkan jumlah proses yang terdapat dalam DFD ini menghasilkan grafik seperti pada Gambar 9 dengan garis berwarna hijau. Garis berwarna biru menunjukkan hasil pengujian nilai FP dengan acuan banyaknya jumlah data yang mengalir dalam sistem yang dihitung. Nilai FP tidak berbanding lurus dengan jumlah proses secara konstan karena dalam FP, tidak hanya

jumlah proses saja yang menjadi penentu nilai FP, tetapi juga terdapat jumlah data yang menentukan besarnya nilai FP.

Diambil 2 data sample dari 12 dataset di atas, yaitu DFD SuperSport 7 dan DFD Penerimaan Siswa 2. DFD SuperSport 7 memiliki 3 proses, 1 *datastore*, dan 15 entitas. DFD Penerimaan Siswa 2 memiliki 3 proses, 1 *datastore*, dan 7 entitas. Dengan menggunakan sistem perhitungan FP, DFD Penerimaan Siswa 2 memiliki nilai FP 46 dan DFD SuperSport 7 memiliki nilai FP 86. Dengan jumlah proses dan *datastore* yang sama, DFD Penerimaan Siswa 2 memiliki lebih sedikit entitas di dalam ERD dibandingkan dengan DFD SuperSport 7. Hasil ini menunjukkan bahwa kompleksitas fungsional sistem dipengaruhi oleh banyaknya data dalam sistem tersebut.

B. Pengujian dan Analisis Akurasi Hasil Perhitungan

Dari 12 data uji, dilakukan pengujian dataset dengan membandingkan hasil perhitungan manual dengan hasil perhitungan menggunakan aplikasi. Tabel 8 menjelaskan perbandingan hasil dengan dua teknik tersebut. Pengujian hasil perhitungan FP menggunakan aplikasi dibandingkan dengan hasil perhitungan manual menghasilkan 52 data sesuai dan 8 data tidak sesuai. Hal ini dikarenakan aplikasi salah mengidentifikasi jumlah FTR pada EI dan EQ. Aplikasi tidak mampu mengidentifikasi jumlah FTR dengan benar apabila FTR dalam proses tersebut terdapat EIF yang terhubung langsung dengan lebih dari satu ILF. Aplikasi tidak mampu menampung lebih dari satu EIF yang berrelasi dengan ILF. Tabel "calon_siswa" hanya mampu menampung tabel "nilai" sebagai referensi dari tabel "calon_siswa". Oleh karena hal ini, aplikasi salah dalam mengidentifikasi jumlah FTR yang terdapat dalam satu proses EI.

Pengujian dan Analisis Kecepatan Proses Perhitungan

Penelitian ini bertujuan untuk mendapatkan cara yang lebih cepat dalam menghitung nilai FP. Oleh karena itu dilakukan pengujian kecepatan perhitungan FP dengan membandingkan kecepatan menghitung secara manual dan menggunakan aplikasi. Hasil pengujian kecepaatan dapat dilihat pada Tabel 9. Pengujian kecepatan pemrosesan hitung FP menggunakan aplikasi didapatkan bahwa rata-rata kecepatan pemrosesan hitung FP menggunakan aplikasi lebih cepat 50 kali dibandingkan dengan perhitungan manual.

TABEL VIII
PERBANDINGAN HASIL PERHITUNGAN MANUAL DENGAN APLIKASI

Nama Data Uji	Aplikasi						Manual					
	ILF	EIF	EI	EQ	EO	Total	ILF	EIF	EI	EQ	EO	Total
DFD HR FUN	21	10	12	3	4	50	21	10	12	3	4	50
DFD Keuangan Daerah	42	30	19	0	0	91	42	30	19	0	0	91
DFD Penerimaan Siswa	21	25	19	15	0	80	14	25	30	22	0	91
DFD Penerimaan Siswa 2	7	30	9	0	0	46	7	30	9	0	0	46
DFD SuperSport 1	14	60	20	6	7	107	14	60	20	6	7	107
DFD SuperSport 2	14	55	21	3	13	106	14	55	14	3	13	99
DFD SuperSport 3	17	60	24	7	9	117	17	60	21	7	9	114
DFD SuperSport 4	14	65	17	9	4	109	14	65	15	9	4	107
DFD SuperSport 5	7	70	10	3	4	94	7	70	9	3	4	93
DFD SuperSport 6	7	70	10	3	4	94	7	70	9	3	4	93
DFD SuperSport 7	7	70	7	3	0	87	7	70	6	3	0	86
DFD SuperSport 8	7	70	6	6	0	89	7	70	6	6	0	89
jumlah kolom yang berbeda						**8**	**prosentase perbedaan**					**13,3**
jumlah kolom yang sama						**52**	**prosentase kesesuaian**					**86,7**
jumlah semua kolom						**60**						

TABEL IX
PENGUJIAN KECEPATAN APLIKASI

No	Nama Data Uji	Kecepatan Perhitungan Manual (detik)	Kecepatan Perhitungan Aplikasi (detik)	Perbandingan
1.	DFD HR FUN.xml	600	15.3204	39:1
2.	DFD Keuangan Daerah.xml	600	7,8605	76:1
3.	DFD Penerimaan Siswa 2.xml	960	16.753	57:1
4.	DFD SuperSport 1.xml	1020	22,1943	46:1
5.	DFD SuperSport 8.xml	780	25,9418	30:1
			Rata-rata	50:1

TABEL X
KASUS UJI PERFORMA

Nama Kasus Uji	Performa
Objek Uji	Kebutuhan non fungsional (SRS_006)
Tujuan Pengujian	Untuk menguji kecepatan pengaksesan sistem
Data Masukan	
Prosedur Uji	Pengguna mengakses setiap halaman dalam sistem
Hasil yang diharapkan	Pengaksesan setiap halaman maksimal 2 menit
Hasil yang didapatkan	Pengaksesan halaman terlama adalah 25,9418 detik
Status Validasi	Valid

C. Pengujian black-box

Kasus uji performa

Hasil pengujian kasus uji performa dapat dilihat pada Tabel 10. Berdasarkan pengujian *black-box* dengan teknik validasi yang telah dilakukan, dapat disimpulkan bahwa kebutuhan fungsional dan non fungsional sistem yang dibangun telah sesuai dengan kebutuhan yang dirancang pada analisis kebutuhan. Kecepatan akses aplikasi telah memenuhi kebutuhan non-fungsional kode SRS_006 dengan kecepatan akses selama kurang dari 2 menit, yaitu 25,9418 detik.

VII. PENUTUP

A. Kesimpulan

Berdasarkan hasil penelitian mengenai perhitungan ukuran kompleksitas fungsional perangkat lunak menggunakan metrik *Function Point* dapat disimpulkan bahwa:

1. Sistem otomatisasi perhitungan *Function Points* (FP) mampu mendapatkan nilai FP dari DFD dan ERD secara otomatis dengan menemukan dan menghitung nilai elemen FP (ILF, EIF, EI, EQ, EO) dari data hasil parsing file XML yang dibuat menggunakan *tool* Visual Paradigm.
2. Setelah dilakukan pengujian pada 12 data uji, didapatkan kesimpulan bahwa aplikasi perhitungan FP mampu menghitung 50 kali lebih cepat dibandingkan dengan perhitungan manual. Dari 60 elemen FP dari 12 data uji didapatkan kesimpulan bahwa aplikasi mampu mengeluarkan hasil 52 elemen sesuai dengan perhitungan manual yaitu sebesar 86,7% dan 8 elemen tidak sesuai dengan perhitungan manual..

B. Saran

Saran untuk penelitian dalam pengembangan sistem perhitungan FP lebih baik kedepannya adalah sebagai berikut:

1. Aplikasi ini masih memiliki kekurangan dalam mengidentifikasi nilai FTR pada EI dan EQ, oleh karena itu disarankan untuk memperbaiki identifikasi FTR pada EI dan EQ untuk pengembangan aplikasi yang lebih baik ke depannya.
2. Aplikasi ini masih belum mampu menghitung nilai *Adjusted Function Point* (AFP) yaitu nilai FP yang memperhitungkan nilai non fungsional. Oleh karena itu, disarankan untuk menambahkan fitur perhitungan *Value Adjusment Factor* (VAF).

VIII. DAFTAR PUSTAKA

[1] R. S. Pressman, Software Engineering: A Practitioner's Approach, New York: McGraw-Hill, 2001.

[2] A. Dennis, B. H. Wixom and R. M. Roth, System Analysis and Design: Fifth Edition, USA: John Wiley & Sons, Inc, 2011.

[3] V. A. Batista, C. C. D. Peixoto and E. P. Borges, "RemoFP: A Tool for Counting Function Points from UML Requirement Models," 2011.

[4] IFPUG team, "Function Point Counting Practices Manual," 2000. [Online]. Available: http://perun.pmf.uns.ac.rs/old/repository/research/se/functionpoints.pdf. [Accessed 20 2 2016].

[5] J. A. Pow-Sang, D. Villanueva, L. Flores and C. Rusu, "A Conversion Model and a Tool to Identify Function Point Logic Files using UML Analysis Class Diagrams," in *Joint Conference of the 23nd International Workshop on Software Measurement (IWSM) and the Eighth International Conference on Software Process and Product Measurement (Mensura)*, Peru, 2013.

[6] D. Longstreet, "Function Point Training Course," [Online]. Available: softwaremetrics.com. [Accessed 26 09 2015].

[7] M. Junaedi, "Pengantar XML," 2003. [Online]. Available: ilmukomputer.com.