IMPLEMENTASI SISTEM INFORMASI IZIN LOKASI (SILOKA) PADA DINAS CIPTA KARYA DAN TATA RUANG KABUPATEN MALANG DENGAN PEMROGRAMAN BERORIENTASI OBJEK

Syeh Mukhamad Iqbal[1], Ismiarta Aknuranda,S.T ,M.Sc, Ph.D[2], Satrio Agung W.,S.Kom,M.Kom[3]
Fakultas Ilmu Komputer
Universitas Brawijaya
Jalan Veteran No.8, Malang, Jawa Timur, Indonesia
Mukhamad.iqbal.davi@gmail.com[1],ismiarta@gmail.com[2],satrio.agung.w@ub.ac.id[3]

Abstrak

Dinas Cipta Karya dan Tata Ruang merupakan salah satu instansi pemerintah di Kabupaten Malang yang mempunyai fungsi salah satunya adalah penyusunan kebijakan dan standarisasi teknis bangunan gedung termasuk pengelolaan gedung dan rumah asset daerah, sehingga terdapat proses izin lokasi ketika masyarakat akan melakukan pembangunan bangunan. Pada Dinas Cipta Karya dan Tata Ruang permohonan izin lokasi masih dilakukan secara manual. Warga harus datang terlebih dahulu ke kantor Dinas Cipta Karya dan Tata Ruang untuk mendapatkan formulir, kemudian menyerahkan formulir dan berkas, dilanjutkan dengan pemeriksaan berkas. Pemeriksaan dilakukan terutama pada lokasi perizinan, apakah wilayah atau lokasi termasuk diizinkan tanpa syarat, diizinkan dengan syarat, dapat dibangun secara terbatas atau tidak diperbolehkan. Pemeriksaan lokasi perizinan berdasarkan peraturan zonasi. Peraturan zonasi, meliputi ketentuan kegiatan penggunaan lahan, ketentuan tata bangunan, ketentuan prasarana dan sarana minimum, ketentuan pelaksanaan, ketentuan perubahan peraturan zonasi, serta ketentuan khusus. Ketentuan-ketentuan peraturan zonasi tersebut masih terdapat pada dokumen buku cetak, dan membutuhkan waktu yang lama serta teliti untuk mencari ketentuan zonasi, sehingga mempunyai resiko kesalahan lebih besar. Berdasarkan permasalahan tersebut, maka dibutuhkan suatu sistem informasi untuk proses izin lokasi. Untuk mengimplementasikan sebuah sistem informasi, dibutuhkan sebuah analisis kebutuhan yang menentukan kebutuhan apa yang dibutuhkan oleh pihak terkait. Selanjutnya, dilakukan perancangan untuk merancang bagaimana sistem akan dibangun. Pada analisis dan perancangan sudah dilakukan pada penelitian sebelumnya. Pada penelitian analisis dan perancangan belum disertai dengan adanya implementasi sistem. Tujuan penelitian ini mencakup perancangan untuk kebutuhan implementasi, implementasi SILOKA, dan pengujian SILOKA dengan menggunakan jenis pengujian *BlackBox*. Perancangan dilakukan untuk menyesuaikan dengan *framework* yang dipakai. Implementasi menggunakan pemrograman berorientasi objek, menggunakan konsep MVC dan menggunakan *framework* CodeIgniter. Hasil dari penelitian ini merupakan produk jadi SILOKA. Pengujian yang dilakukan pada penelitian ini menggunakan pengujian *blackbox* pada semua *use case* dan pengujian *whitebox* pada tiga *use case*. Hasil pengujian *blackbox* pada SILOKA adalah semua berjalan sesuai dengan spesifikasi *use case* yang telah didefinisikan pada penelitian sebelumnya. Sedangkan hasil pengujian *whitebox* merupakan perhitungan kompleksitas dan jalur independen, dimana setiap jalur independen telah dilalui minimal satu kali.

Kata kunci: Kontriuksi, Implementasi, Sistem Informasi, Berorientasi Objek, CodeIgniter, pengujian *Blackbox,* pengujian *Whitebox*

Abstract

Dinas Cipta Karya dan Tata Ruang is one of the government agencies in Malang which has the function of one of them is the development of policies and technical standardization buildings including asset building management area, so there is a location permit process when the public will do the construction of the building. In Dinas Cipta Karya dan Tata Ruang location permit is still done manually. Residents must come first to the office to get a registration form, then submit the form and file, and then the file will be checked. Examination of the file permormed mainly on the location permit, whether the region or location including unconditionally approved, permitted by the terms, can be build in a limited or not allowed. Examining location performed by the zoning regulation permit. Zoning regulations include land use activities, the procedures of buildings, provision of infrastructure and facilities for minimum, implementing provisions, the provision of the zoning regulation changes, as well as special provisions. The provisions of the zoning regulations are still contained in coduments printed books, and takes a long time to looking for zoning provisions, so have a greater risk of error. Based on the problems, Dinas Cipta Karya dan Tata Ruang need a information system to process the location permit. To implement an information system, it takes a needs analysis to determine the requirement. Furthermore, design phase how system will be build. In the analysis and design has been done in previous studies. In previous studies

have not been accompanied by the implementation of the system. The purpose of this research include planning for the needs of implementation, implementation of SILOKA and testing SILOKA using blackbox testing types. The design needs to be done to adjust the framework. Implementation using object-oriented programming, using concept of MVC and using framework CodeIgniter. The result of this research is SILOKA product. Test performed in this research using blackbox testing on all use case and whitebox testing on three use case. The result of blackbox testing in SILOKA is system runs according use case spesification that defined in previous research. Whereas the result of whitebox testing is cyclomatic complexity and independent path, which each independent path has been passed minimal once.

Keywords: *Contruction, Implementation, Information System , Object Oriented, CodeIgniter, Blackbox Testing, Whitebox Testing*

1. PENDAHULUAN

Dinas Cipta Karya dan Tata Ruang mempunyai tugas pokok, salah satunya adalah melaksanakan urusan pemerintah daerah bidang Cipta Karya dan Tata Ruang. Untuk menyelenggarakan tugas tersebut, Dinas Cipta Karya dan Tata Ruang mempunyai fungsi salah satunya adalah penyusunan kebijakan dan standarisasi teknis bangunan gedung, termasuk pengelolaan gedung dan rumah asset daerah (DCKTR, 2013). Untuk itu Dinas Cipta Karya mempunyai sebuah peraturan zonasi yang disusun sebagai pedoman pengendalian pemanfaatan ruang serta berdasarkan rencana rinci tata ruang untuk setiap zona pemanfaatan ruang. Ketentuan peraturan zonasi, meliputi ketentuan kegiatan dan penggunaan lahan, ketentuan tata bangunan, ketentuan prasarana dan sarana minimum, ketentuan pelaksanaan, ketentuan perubahan peraturan zonasi, serta ketentuan khusus. Klasifikasi zona di kawasan Perkotaan Kepanjen didasarkan pada peraturan pemerintah PU nomor 20 tahun 2011 tentang Penyusunan RDTRK (Rencana Detail Tata Ruang Kota) dan Peraturan Zonasi yang disesuaikan dengan kegiatan yang telah berkembang di wilayah perencanaan. Namun, Peraturan Zonasi pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang masih dalam bentuk dokumen buku cetak. Untuk mencari ketentuan-ketentuan peraturan zonasi harus membuka buku dokumen buku cetak peraturan zonasi dan membutuhkan waktu yang tidak sedikit serta ketelitian yang tinggi dimana mempunyai resiko kesalahan yang lebih besar.

Sebagaimana fungsi Dinas Cipta Karya dan Tata Ruang salah satunya penyusunan kebijakan standarisasi teknis bangunan gedung termasuk pengelolaan gedung dan rumah asset daerah, maka jika terdapat masyarakat yang mewakili sebuah perusahaan atau individu jika ingin membangun sebuah bangunan, harus mengajukan permohonan izin lokasi ke Dinas Cipta Karya dan Tata Ruang apakah layak dibangun di tempat yang diajukan atau tidak. Untuk mengajukan permohonan izin lokasi, masih dilakukan dengan cara manual. Manual yang dimaksud, pemohon harus datang terlebih dahulu ke Kantor Dinas Cipta Karya dan Tata Ruang membawa berkas, lalu meminta formulir, mengisi formulir kemudian menyerahkan formulir dan berkas kepada Dinas. Kemudian pada berkas yang sudah dikumpulkan, dilakukan pemeriksaan berkas. Pemeriksaan dilakukan terutama pada lokasi perizinan, apakah wilayah termasuk diizinkan tanpa syarat, diizinkan dengan syarat, dapat dibangun secara terbatas atau tidak diperbolehkan. Untuk melakukan pemeriksaan tersebut menggunakan Buku Pedoman Peraturan Zonasi. Dengan adanya hal tersebut dan dengan seiringnya perkembangan terknologi, dibutuhkanlah sebuah Sistem Informasi Izin Lokasi (SILOKA) yang juga mencakup Peraturan Zonasi.

Pada penelitian sebelumnya sudah dilakukan sebuah analisis persyaratan dan perancangan Sistem Informasi Izin Lokasi (SILOKA) yang dilakukan oleh Bella Pertiwi selaku mahasiswi Sistem Informasi Universitas Brawijaya dengan judul penelitian Analisis dan Perancangan Sistem Informasi Izin Lokasi (SILOKA) pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang dengan Pendekatan Berorientasi Objek (Pertiwi, 2016). Analisis yang dilakukan, meliputi analisis proses bisnis yang dinotasikan dalam bentuk BPMN, analisis persyaratan, perancangan sistem dalam bentuk UML, dan evaluasi.

Berdasarkan analisis dan perancangan yang sudah dilakukan serta, maka penulis akan melanjutkan tahap pengembangan perangkat lunak ke tahap selanjutnya, yaitu implementasi SILOKA berdasarkan analisis dan perancangan yang sudah dilakukan. Implementasi menggunakan pemrograman berorientasi objek, karena pada analisis dan perancangannya menggunakan pendekatan berorientasi objek. Pada pemrograman beorientasi objek dan menganut konsep MVC (*Model, View, Controller*).

1.1. Rumusan Masalah

Dari latar belakang yang sudah dijabarkan, terdapat rumusan masalah umum dan khusus. Rumusan masalah umum adalah dapatkah SILOKA diimplementasikan. Sedangkan rumusan masalah

khusus adalah bagaimanakah hasil perancangan dan implementasi SILOKA berdasarkan perancangan yang sudah ada, dan bagaimanakah hasil pengujian SILOKA menggunakan metode pengujian *blackbox* dan pengujian *whitebox*.

1.2. Tujuan

Pada penelitian ini terdapat tujuan umum dan tujuan khusus. Tujuan umum adalah untuk mengimplementasikan SILOKA berdasarkan perancangan sebelumnya. Sedangkan tujuan khususnya adalah menghasilkan perancangan detail dan impelementasi rancangan dari SILOKA dan melakuakn pengujian terdapat SILOKA dengan pengujian *blackbox* dan pengujian *whitebox*.

1.3. Batasan Masalah

Batasan masalah dalam penelitian ini diantaranya mengacu pada batasan masalah penelitian sebelumnya. Implementasi dilakukan dengan menggunakan *framework* CodeIgniter

2. LANDASAN KEPUSTAKAAN

2.1. Pemrograman Berorientasi Objek

Pemrograman berorientasi objek merupakan suatu metode pengembangan perangkat lunak untuk membangun sebuah sistem perangkat lunak yang dapat diandalkan, mudah digunakan, mudah dipelihara, terdokumentasi dengan baik dan *reusable* yang dapat memenuhi kebutuhan pengguna dengan sebuah komunitas objek yang bekerjasama dengan satu sama lain (Pillay, 2007). Teknik pemecahan masalah yang digunakan pemrograman beorientasi objek lebih mirip dengan cara manusia dalam memecahkan masalah sehari-hari, kita membuat sebuah objek perangkat lunak seperti objek dunia nyata yang juga memiliki atribut dan perilaku.

Objek perangkat lunak dimodelkan seperti objek dunia nyata bahwa mereka juga memiliki atribut dan perilaku. Sebuah objek perangkat lunak mengelola atribut dalam satu atau lebih variabel. Variabel merupakan item data yang disebut *identifier*. Kelas merupakan *blueprint* dari objek, sebuah kelas digunakan untuk membuat sebuah objek. Kelas menyatakan variabel *instance* yang berisi objek. *Method* adalah fungsi yang berhubungan dengan suatu objek. Objek juga dikenal sebagai *instance* (Pillay, 2007).

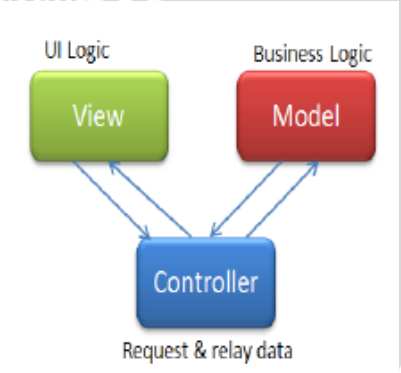
Enkapsulasi merupakan salah satu konsep utama pada perancangan berorientasi objek. Informasi dan *behaviour* yang saling berkaitan harus terdapat pada kelas yang sama, sehingga sistem dapat mencapai prinsip berorientasi objek. Pada konsep enkapsulasi, data dan proses yang dapat memanipulasi data berada pada satu *package* sebagai unit yang kohesif (Pressman, 2010). Pewarisan (*inheritance*) merupakan salah satu karakteristik pembeda antara sistem konvensional dan sistem yang berorientasi pada objek. *Subclass* dapat mewarisi atribut dan operasi yang berkaitan dari *superclass*. Seluruh struktur data dan algoritma yang dirancangan dan diimplementasikan pada *superclass* juga tersedia dan dapat digunakan oleh *subclass*.

2.2. MVC (*Model-View-Controller*)

MVC (*Model-View-Controller*) merupakan konsep arsitektur perangkat lunak yang terdiri dari pola arsitektur dalam rekayasa perangkat lunak yang mempunyai tiga komponen, yaitu *Model, View* dan *Controller* (Sarker dan Apu, 2014).

Berikut merupakan kelebihan MVC:

1. Penggunaan ulang komponen-komponen antarmuka pengguna
2. Kemampuan untuk mengembangkan aplikasi dengan antarmuka pengguna secara terpisah
3. Kemampuan untuk melakukan pewarisan (*inheritane*) dari beberapa bagian yang berbeda pada suatu hierarki kelas



Gambar 1 Komponen MVC

Model adalah bagian yang menangani hal yang berhubungan dengan pengolahan data ke basis data. *View* adalah bagian yang menangani semua hal tentang tampilan *user interface* atau halaman yang diakses oleh pengguna. *Controller* adalah kumpulan dari intruksi aksi yang menghubungkan *model* dan *view*.

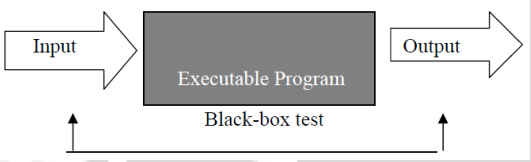
2.3. CodeIgniter

Codeigniter adalah sebuah *framework* untuk pemrograman php yang bersifat *open source* menggunakan model MVC. CodeIgniter dibangun dengan tujuan untuk memudahkan pengembang atau programmer dalam membangun atau mengembangkan sebuah aplikasi berbasis *web*. CodeIgniter mempunyai sintak yang terstruktur, kemudahan dalam menggunakannya, menyediakan

fasilitas *helper* dan *library* yang dapat membantu para pengembang dalam membuat *pagination*, *session*, manipulasi *url* dan lainnya. CodeIgniter mempunyai keunggulan-keunggulan diantaranya adalah ringan, gratis, sampai saat ini CI masih diakui sebagai *framework* yang paling cepat, menggunakan konsep MVC, dukungan teknis yang lengkap di forum CI.

2.4 Pengujian Black Box

Pengujian *blackbox* adalah pengujian yang tidak berfokus pada mekanisme internal sistem atau komponen, tetapi hanya berfokus pada keluaran yang dihasilkan sebagai respon terhadap masukan yang dipilih dan kondisi eksekusi (Williams, 2009). Pada Gambar 2 merupakan ilustrasi pengujian *blackbox.*



Gambar 2 Pengujian *Black Box*

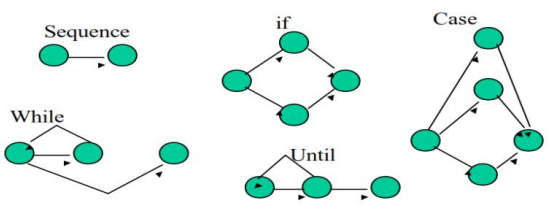
Pengujian *blackbox* juga untuk memastikan bahwa fungsi yang ditetapkan dalam spesifikasi persyaratan bekerja. Pengujian *blackbox* bertujuan untuk menemukan kesalahan dalam perilaku eksternal sistem yang dikategorikan sebagai berikut:

a. Kesalahan atau hilangnya fungsi
b. Kesalahan *interface*
c. Kesalahan data struktur atau akses database eksternal
d. Kesalahan *behaviour* atau performa

2.5. Pengujian *Whitebox*

Pengujian *whitebox* digunakan terutama untuk mendeteksi kesalahan logis dalam kode program (Nindhra dan Jagruthi, 2012). Pengujian *whitebox* digunakan untuk melakukan *debug* pada kode, menemukan kesalahan ketik, dan mengungkap asumsi pemrograman yang salah.

Control flow graph (CFG) merupakan grafik terarah yang terdiri dari dua tipe, yaitu *node* dan *control flow* (Nindhra dan Jagruthi, 2012). *Node* dinyatakan oleh lingkaran berlabel, yang mewakili satu atau lebih pernyataan, kondisi, atau pernyataan. *Control flow* dinyatakan dengan panah yang mewakili kontrol program. *Node* yang termasuk kondisi disebut predikat *node*. Area ditetapkan oleh *edge* dan *node* yang disebut sebagai *region*. Gambar 3 merupakan notasi pada *flow graph*.



Gambar 3 Notasi *Flow Graph*

Cyclomatic complexity diperoleh dari jumlah *edges* dari *flow graph* program dikurangi jumlah *node* ditambah dua (Nidhra dan Jagruthi, 2012). *Cyclomatic Complexity* dapat dikalkulasi dengan rumus sebagai berikut:

V (G) = e – n + 2

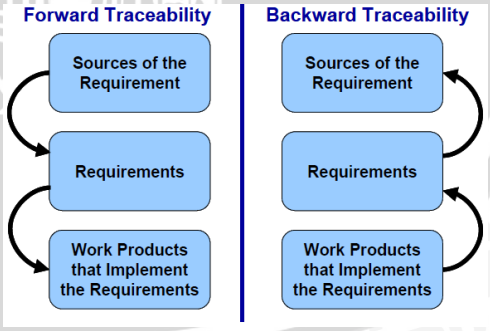
Cara alterntif untuk mengkalkulasi *cyclomatic complexity* program dari pengecekan *flow graph* adalah sebagai berikut (Nidhra dan Jagruthi, 2012):

V (G) = total dari daerah yang dibatasi oleh *edges* dan *node* (*region*)

V (G) = P + 1 dimana P adalah jumlah dari predikat *node*

2.6. Matriks Kerunutan

Matriks kerunutan atau *traceability matrix* merupakan salah satu hal penting dalam manajemen persyaratan yang digunakan untuk memastikan sebuah produk dibangun dengan tepat di setiap fase dari *life cycle* pengembangan perangkat lunak untuk menelusuri progress pengembangan (Westfall, 2006). Matriks kerunutan yang baik memungkinkan untuk melacak dari dua arah, sehingga *traceability* dapat ditelusuri baik dari depan (*forward traceability*) maupun belakang (*Backward Traceability*). Gambar 4 merupaka matriks kerunutan dua arah.



Gambar 4 Matriks Kerunutan Dua Arah

Forward Traceability berfungsi sebagai berikut:

1. Menelusuri sumber persyaratan ke persyaratan produk sehingga digunakan untuk memastikan kelengkapan spesifikasi persyaratan produk
2. Menelusuri persyaratan masing-masing produk yang unik ke dalam desain yang mengimplementasikan persyaratan, kode yang mengimplementasikan desain

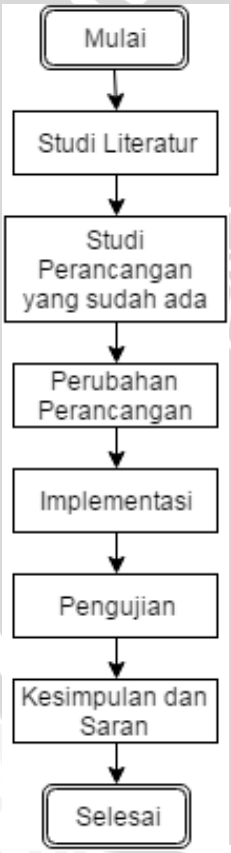
tersebut, dan tes yang memvalidasi persyaratan tersebut. Tujuannya adalah untuk memastikan bahwa setiap persyaratan diimplementasikan dalam bentuk produk dan setiap kebutuhan benar-benar teruji.

3. Menelusuri setiap produk yang dikerjakan kembali ke persyaratan yang terkait.

SDS (*Software Desain Spesification*) termasuk yang *tag* yang mengidentifikasi persyaratan yang dilakukan oleh setiap elemen desain yang diidentifikasi dan UTS (*Unit Test Spesification*) termasuk menelusuri tag elemen desain setiap *test case* yang diverifikasi.

3. METODOLOGI

Langkah-langkah yang dilakukan dalam penelitian ini ditunjukkan pada Gambar 5 dibawah ini:



Gambar 5 Metodologi Penelitian

4. PERANCANGAN

Dalam tahap ini akan dijelaskan mengenai penjelasan produk dan perubahan perancangan ke implementasi.

4.1 Cakupan Produk

Sistem Informasi Izin Lokasi (SILOKA) adalah sebuah sistem informasi yang memungkinkan seseorang atau warga yang mewakili sebuah instansi atau perusahaan dapat mengajukan permohonan izin lokasi dan mengelola berkas secara *online*. SILOKA juga memungkinkan pegawai Dinas Cipta Karya dan Tata Ruang Kabupaten Malang yang terlibat dalam permohonan izin lokasi untuk mengelola data izin lokasi secara *online* melalui sistem. SILOKA adalah sebuah sistem informasi berbasis *website*, jadi SILOKA dapat diakses dimanapun, pengguna hanya membutuhkan koneksi *internet*. SILOKA juga dapat mempercepat dalam pencarian peraturan zonasi yang diperlukan oleh surveyor dinas yang akan melakukan survey.

4.2 Fitur Produk

Fitur menjelaskan mengenai kesimpulan kemampuan dari SILOKA yang akan dibangun secara umum. Prioritas pada fitur menggunakan aturan MoSCoW. Tabel 1 merupakan beberapa fitur dari SILOKA :

Tabel 1 Fitur SILOKA

Kode	Nama	Deskripsi	Prioritas
FEAT1	Identifikasi Pengguna	Sistem dapat melakukan identifikasi pengguna	M
FEAT2	Pencarian Data Zonasi	Sistem dapat digunakan untuk melakukan pencarian data peraturan zonasi	M
FEAT3	Mengelola Data Survey	Sistem dapat digunakan untuk melakukan pengelolaan data survey	M

4.3 Persyaratan Fungsional

Dari fitur yang sudah didefinisikan di penelitian ini memiliki fungsi-fungsi yang terdapat di dalamnya. Persyaratan fungsional telah didefinisikan pada penelitian sebelumnya. Tabel 2 merupakan beberapa persyaratan fungsional SILOKA:

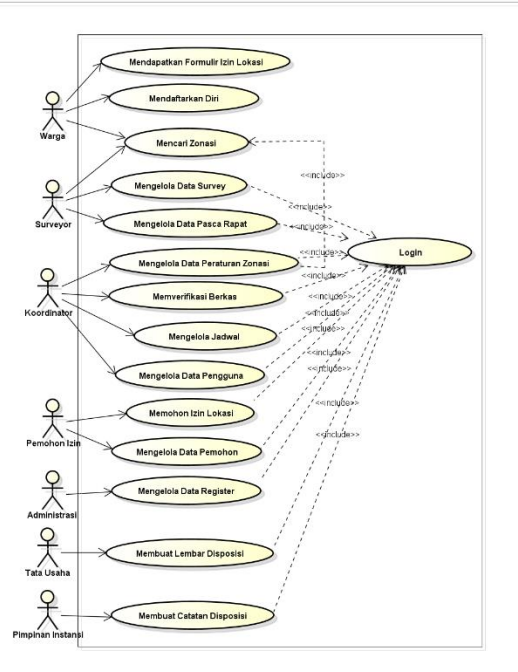
Tabel 2 Spesifikasi Persyaratan

Kode Fitur	Kode Dasar Persyaratan Fungsional	Kode Lengkap Persyaratan Fungsional	Deskripsi
FEAT1	SRS-F-SIL-P01	SRS-F-SIL-P01-1	Sistem dapat melakukan autorisasi apakah pengguna yang menggunakan sistem merupakan pengguna yang teridentifikasi dan terautorisasi untuk menggunakan

			fitur SILOKA
FEAT2	SRS-F-SIL-P02	SRS-F-SIL-P02-1	Sistem dapat digunakan untuk melakukan pencarian data peraturan zonasi dengan kata kunci pencarian berupa kode blok, kode subzona dan peruntukan bangunan

4.4 Pemodelan *Use Case*

Pada penelitian sebelumnya terdapat pemodelan *use case*, pemodelan *use case* bertujuan untuk menggambarkan mengenai tujuan yang ingin dicapai aktor pada sistem Gambar 6 merupakan *use case* keseluruhan dari SILOKA :



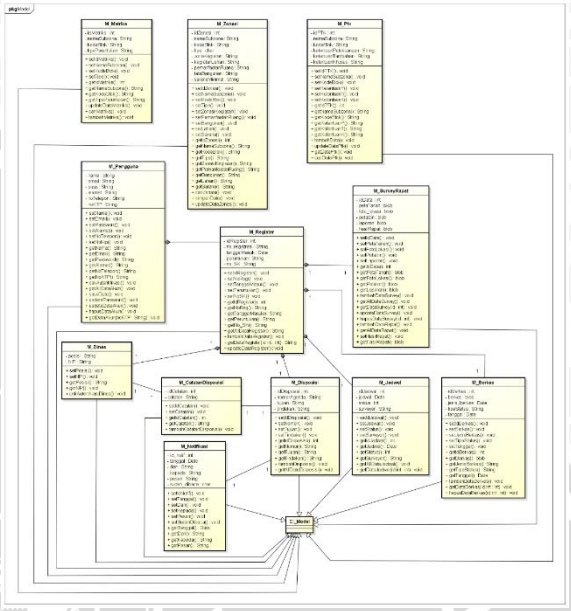
Gambar 6 *Use Case* SILOKA

4.5 Perubahan Perancangan ke Implementasi

Ketika melakukan implementasi, perlu dilakukan penyesuaian pada diagram kelas perancangan dengan *framework* yang dipakai, yaitu codelgniter. Pada codeigniter semua kelas model diturunkan dari CI Model, jadi untuk semua kelas model mempunyai hubungan ke CI Model. Untuk semua kelas controller juga diturunkan dari CI Controller, jadi semua kelas controller mempunyai hubungan ke CI Controller.

4.5.1 Kelas *Model*

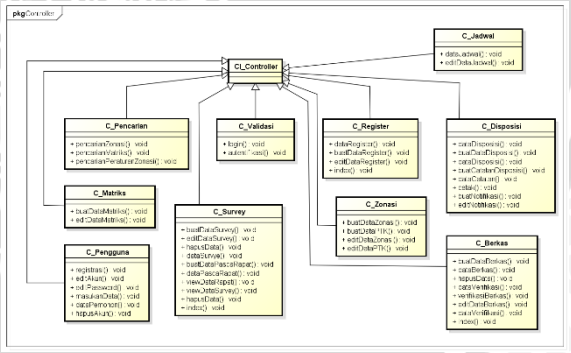
Pada diagram kelas implementasi juga terjadi penambahan *method,* yaitu pada kelas M_Disposisi ditambahkan *method* getDisposisiRegister(). *Method* getDisposisiRegister() digunakan untuk mengambil data permohonan dari pemohon izin. Pada Gambar 7 menunjukkan *model* dari diagram kelas implementasi.



Gambar 7 Diagram kelas untuk *model*

4.5.2 Kelas *Controller*

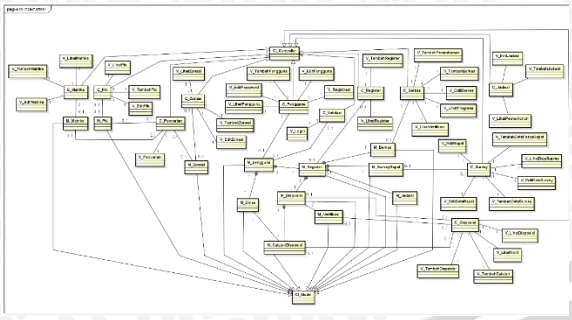
Pada *controller* diagram kelas implementasi juga terjadi perubahan penyesuaian *framework* dimana semua kelas *controller* juga diturunkan dari CI Controller. Gambar 8 menunjukkan *controller* dari diagram kelas implementasi:



Gambar 8 Diagram kelas untuk *controller*

4.5.3 Kelas Diagram Keseluruhan

Gambar 9 menunjukkan gabungan dari *controller* dan *model* diagram kelas implementasi pada SILOKA:



Gambar 9 Diagram Kelas Implementasi

4.6 Matriks Kerunutan Diagram Kelas

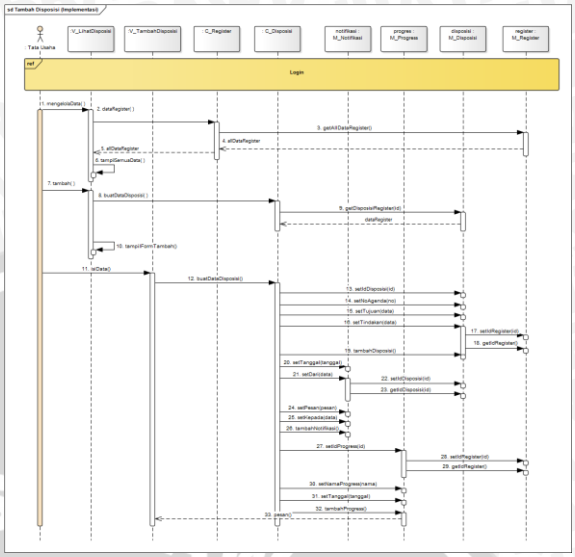
Matriks kerunutan bertujuan untuk merunut antara diagram kelas perancangan ke diagram kelas implementasi. Tabel 3 merupakan matriks kerunutan antara diagram kelas perancangan dan diagram kelas implementasi:

Tabel 3 Kerunutan Diagram Kelas

No	Kelas Perancangan	Kelas Implementasi
1	C_Zonasi	C_Zonasi
2	C_Matriks	C_Matriks
3	C_Pengguna	C_Pengguna
4	C_Pencarian	C_Pencarian
5	C_Register	C_Register
6	C_Survey	C_Survey
7	C_Berkas	C_Berkas
8	C_Jadwal	C_Jadwal
9	C_Disposisi	C_Disposisi
10	C_Validasi	C_Validasi
11	M_Matriks	M_Matriks
12	M_Notifikasi	M_Notifikasi
13	M_Zonasi	M_Zonasi
14	M_Ptk	M_Ptk
15	M_Akun	M_Akun
16	M_Dinas	M_Dinas
17	M_Register	M_Register
18	M_SuveyRapat	M_SuveyRapat
19	M_Berkas	M_Berkas
20	M_Jadwal	M_Jadwal
21	M_Disposisi	M_Disposisi
22	Tidak ada	CI_Model
23	Tidak ada	CI_Controller

4.7 Diagram *Sequence*

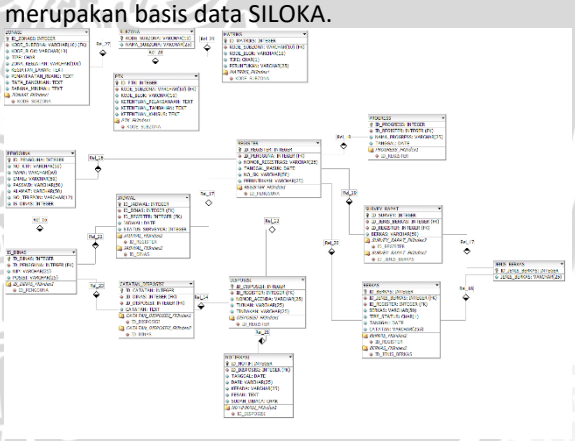
Pada penelitian ini, terjadi beberapa perubahan pada diagram *sequence* dari penelitian sebelumnya berupa penambahan *method*. Gambar 10 merupakan diagram *sequence* membuat lembar disposisi dimana terdapat penambahan *method* getDisposisiRegister(id) ketika Tata Usaha akan membuat lembar disposisi:



Gambar 10 Diagram *Sequence* Membuat Lembar Disposisi

4.8. Pemodelan Basis Data

Basis data bertujuan untuk menyimpan semuan data yang diperlukan pada SILOKA. Basis data untuk implementasi SILOKA merujuk pada penelitan sebelumnya (Pertiwi, 2016). Gambar 11 merupakan basis data SILOKA.



Gambar 11 Basis Data SILOKA

5. IMPLEMENTASI

Pada tahap ini menjelaskan mengenai lingkungan implementasi menjelaskan tentang lingkungan spesifikasi perangkat lunak maupun perangkat keras yang digunakan ketika implementasi. Berikut ini merupakan perangkat lunak yang digunakan untuk pengembangan SILOKA:

1. Sistem Operasi Microsoft Windows 10 Pro Single Language
2. XAMPP Versi 3.2.1 (Apache untuk Web Server dan MySQL untuk basis data)
3. Sublime Text V.3
4. *Web Browser* Google Chrome

5.1 Implementasi Sistem

Pada implementasi dibahas mengenai *screenshot* hasil implementasi dari SILOKA. Gambar 12 merupakan halaman antarmuka pengguna untuk melakukan pengelolaan data berkas:

Gambar 12 Halaman Mengelola Berkas Pemohon

6. PENGUJIAN

Dalam penelitian ini, peneliti akan menguji berdasarkan *use case* perangkat lunak SILOKA dengan menggunakan jenis pengujian *Blackbox* dan pengujian *Whitebox*.

6.1 Pengujian *Blaxkbox*

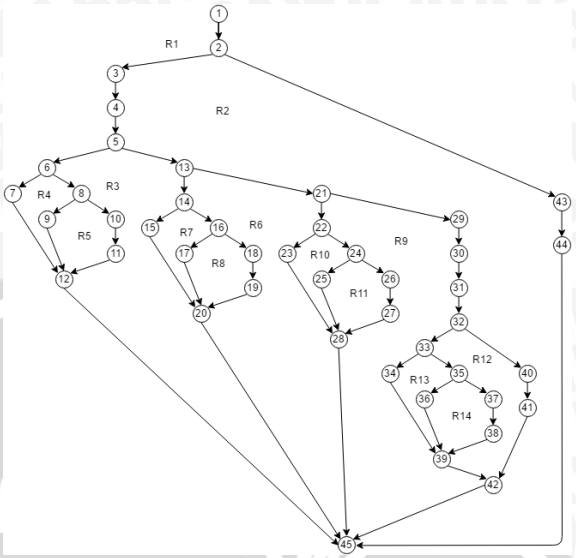
Pada penelitian ini, *test case* dibuat berdasarkan skenario yang sudah didefinisikan pada *use case* spesifikasi di penelitian. Pada *test case* ini akan didefinisikan skenario yang akan dijalankan, kemudian mendefinisikan masukan, lalu keluaran yang dihasilkan. Pada Tabel 4 merupakan pengujian pada use case mencari zonasi

Tabel 4 Pengujian Mencari Zonasi

Mencari Zonasi		
Test case 7	*Use case* mencari zonasi Skenario 1	Berhasil melakukan pencarian
	Masukan	Kode Blok: A1 Kode Subzona: PS Peruntukan: Rumah Tunggal
	Keluaran yang diharapkan	Sistem menampilkan hasil pencarian berupa ketentuan peruntukan bangunan
	Keluaran	Sistem menampilkan hasil pencarian zonasi berupa ketentuan peruntukan bangunan

6.2. Pengujian *Whitebox*

Pada gambar 11 merupakan *flow graph* dari pencarian zonasi.



Gambar 11 *Flowgraph* pencarian zonasi

Berikut merupakan perhitungan *cyclomatic complexity* dari pencarian zonasi:

V(G) = 14
V(G) = E – N + 2
= 57 – 45 + 2
= 14
V(G) = PN + 1
= 13 + 1 = 14

Berikut merupakan jalur independen dari pencarian zonasi:

1) 1-2-44-45-45
2) 1-2-3-4-5-6-7-12-43-45
3) 1-2-3-4-5-6-8-9-12-43-45
4) 1-2-3-4-5-6-8-10-11-12-43-45
5) 1-2-3-4-5-13-14-15-20-43-45
6) 1-2-3-4-5-13-14-16-17-20-43-45
7) 1-2-3-4-5-13-14-16-18-19-20-43-45
8) 1-2-3-4-5-13-21-22-23-28-43-45
9) 1-2-3-4-5-13-21-22-24-25-28-43-45
10) 1-2-3-4-5-13-21-22-24-26-27-28-43-45
11) 1-2-3-4-5-13-21-29-30-31-32-33-34-39-42-43-45
12) 1-2-3-4-5-13-21-29-30-31-32-33-35-36-39-42-43-45
13) 1-2-3-4-5-13-21-29-30-31-32-33-35-37-38-39-42-43-45
14) 1-2-3-4-5-13-21-29-30-31-32-40-41-42-43-45

Pada tabel 5 merupakan *test case* dari pencarian zonasi.

Tabel 5 *Test case* pencarian zonasi

Jalur	Data masuka	Hasil yang diharapka	Hasil yang	Statu s

	n	n	diperoleh	
1	Data pencarian zonasi dan *session* kosong	Menuju ke halaman registrasi	Menuju ke halaman registrasi	Valid
2	Submit: cariZonasi ID Zonasi: *NULL* *Session*: Koordinator Kode blok: A1 Kode subzona: PS Peruntukan: Rumah tunggal	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan penggunaan lahan, pemanfaatan ruang, tata bangunan serta sarana dan prasana pada Koordinator	Menampilkan halaman hasil pencarian berupa ketentuan zonasi, yaitu ketentuan pengguna an lahan, pemanfaatan ruang, tata bangunan serta sarana dan prasana pada Koordinator	Valid

6.3. Matriks Kerunutan Pengujian

Matriks kerunutan sangat penting dalam aktivitas pengembangan perangkat lunak, karena dapat digunakan untuk melacak kesesuaian dari fitur hingga *test case*. Pada Tabel 5 merupakan beberapa matriks *traceability* dari pengujian.

Tabel 5 Matriks *Traceability* Pengujian

Kode Fitur	*Use Case*	*Test case*	Validasi
FEAT2	Mencari Zonasi	*Test case* 7	Valid
		Test case 8	Valid
		Test case 9	Valid
		Test case 10	Valid
		Test case 11	Valid
		Test case 12	Valid
		Test case 13	Valid
		Test case 6	Valid

7. KESIMPULAN DAN SARAN

7.1 Kesimpulan

1. SILOKA dapat diimplementasikan berdasarkan perancangan yang telah dilakukan. Implementasi dilakukan dengan penyesuaian *framework* CI dan terdapat beberapa *method*.
2. Hasil dari perancangan detail adalah perubahan yang menghasilkan diagram kelas implementasi, dimana diagram kelas implementasi sudah menyesuaikan dengan *framework* CodeIgniter. Penyesuaian yang dimaksud adalah semua kelas *controller* diturunkan dari kelas CI Controller, sedangkan semua kelas *model* diturunkan dari kelas CI Model. Jadi terdapat penambahan kelas beupa CI_Model dan CI_Controller yang merupakan kelas dari *framework*. Selain itu juga terdapat penambahan *method* pada model diagram kelas impelementasi, yaitu pada *model* M_Disposisi terdapat penambahan *method* getDisposisiRegister (id) dimana *method* tersebut berfungsi untuk mengambil data pemohon dan permohonan. Perubahan penambahan *method* tersebut juga mempengaruhi diagram *sequence* membuat lembar disposisi, karena terdapat penambahan alur dari *controller* C_Disposisi method buatDataDisposisi() ke model M_Disposisi *method* getDisposisiRegister(id).
3. Hasil implementasi berupa Program SILOKA dengan fitur yang dapat dijalankan, yaitu identifikasi pengguna, pencarian data zonasi, mengelola data zonasi, mengelola data pengguna, mengelola data register, mengelola disposisi, mengelola data pasca rapat, registrasi.
4. Hasil pengujian SILOKA merupakan 13 *use case* diagram atau 90% dari 15 *use case* diagram adalah berjalan sesuai dengan *use case* spesifikasi yang telah dispesifikasikan pada penelitian sebelumnya. Dari 13 *use case* diagram yang telah diimplementasikan dapat bahwa dari *validasi testing* dari 13 *use case* yang telah diimplementasikan hasilnya adalah valid. Sedangkan hasil pengujian *whitebox* pada SILOKA dengan menguji fitur pencarian zonasi, memohon izin lokasi, mengelola data pengguna, telah menghasilkan *flow graph* dan perhitungan kompleksitas, jalur independen dan *test case* berdasarkah jumlah jalur indepnden bahwa setiap *test case* yang diujikan mendapatkan hasil valid.

7.2 Saran

1. SILOKA diharapkan dapat dilanjutkan ke penelitian selanjutnya, yaitu dengan penambahan fitur GIS (Geographic Information System) pada fitur pencarian

zonasi yang dapat digunakan untuk keperluan mencari data kode blok, kode subzona, dan peruntukan berdasarkan wilayah peta.

8. DAFTAR PUSTAKA

A.S, R dan M Shalahudin. 2013. Rekayasa Perangkat Lunak Terstuktur dan Berorientasi Objek. Bandung: Informatika.

Basuki, Awan Pribadi. 2010. Membangun *Web* Berbasis PHP dengan *Framework* Codeigniter. Yogyakarta: Lokomedia.

CodeIgniter. 2014. *Application Flowchart*. [Online]. Tersedia di : <http://www.codeigniter.com/user_guide/overview/appflow.html> [Diakses 26 Juni 2016]

DCKTR (Dinas Cipta Karya dan Tata Ruang), 2013. Tugas Pokok dan Fungsi Dinas Cipat Karya dan Tata Ruang Kabupaten Malang. [Online]. Tersedia di : <http://ciptakarya.malangkab.go.id/konten-22.html> [Diakses 5 April 2016].

Fowler, Martin. 2003. UML *Distilled Third Editian : A Brief Guide to The Standard Object Modelling Language*. [Pdf]. Tersedia di : <http://www.agentgroup.unimore.it/~nicola/courses/IngegneriaDelSoftware/uml/books/UMLDistilled.pdf> [Diakses oada 3 Mei 2016].

George H, William S, Hopwood., 2005. *Sistem Informasi Akuntansi, Buku Satu.*. Jakarta: Salemba Empat.

Gordon, B, Davis., 2010. *Kerangka Dasar Sistem Informasi Manajemen Bagian 1*. Jakarta : PT. Pustaka Binamas Pressindo.

Herlawati dan Widodo., 2011. *Menggunakan UML*. Bandung: Informatika.

IBM. 2004. *An Introduction to Structure Diagrams in UML 2 : The Class Diagram*. [Online]. Tersedia di : <https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/> [Diakses 25 Juni 2016]

IBM. 2004a. *IBM Knowledge Center: Relationship types.* [Online]. Tersedia di : <http://www.ibm.com/support/knowledgecenter/SS8PJ7_9.1.1/com.ibm.xtools.modeler.doc/topics/rreltyp.html> [Diakses 25 Desember 2016].

IBM. 2004b. *IBM Knowledge Center: Relationship in class diagrams*. [Online}. Tersedia di: <http://www.ibm.com/support/knowledgecenter/SS8PJ7_9.1.2/com.ibm.xtools.modeler.doc/topics/crelsme_clssd.html> [Diakses 30 Desember 2016]

Leffingwell, D dan Don Widrig. 2002. *The Rational Edge: The Role of Requirements Traceability in System Development*. [Pdf}. Tersedia di: <http://www.therationaledge.com/content/sep-02/m_requirementsTraceability_di.jsp> [Diakses 5 Agustus 2016]

Nidhra, S dan Jagruthi, Dondeti. 2012. *Black Box and White Box Testing Techniques – A Literature Review*. Vol. 2, No. 2. [e-Journal]. Tersedia melalui : <http://airccse.org/journal/ijesa/papers/2212ijesa04.pdf> [Diakses 1 Oktober 2016].

Pertiwi, Bella., 2016. Analisis dan Perancangan Sistem Informasi Izin Lokasi (SILOKA) pada Dinas Cipta Karya dan Tata Ruang Kabupaten Malang dengan Pendekatan Berorientasi Objek. S1. Fakultas Ilmu Komputer. Universitas Brawijaya. Malang.

Pillay, Anban. 2007. *Object Oriented Programming using Java*. [pdf]. Tersedia di : <http://math.hws.edu/eck/cs124/downloads/OOP2_from_Univ_KwaZulu-Natal.pdf> [Diakses 4 Juni 2016].

Pressman, Roger S. 2010. *Software Engingeering: A Practitioner's Approach 7th*. United States: Raghothaman Srinivasan.

Sarker, H dan K. Apu., 2014. "*MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application*". Volume 7 No. 6. [e-journal]. Tersedia melalui : < http://www.sersc.org/journals/IJHIT/vol7_no5_2014/29.pdf> [Diakses 16 Juni 2016].

Sommerville, 2011. *Software Engineering 9th*. U.S.A : Addison-Wesley.

Westfall, Linda., 2006. "*Bidirectional Requirements Traceability*". Tersedia melalui : <http://www.compaid.com/caiinternet/ezine/westfall-bidirectional.pdf> [Diakses 1 Juni 2016]

Williams, Laurie., 2006. "*Testing Overview and Black-Box Testing Techniques*". [e-journal]. Tersedia melalui : < http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf> [Diakses 27 Maret 2016].