ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - CF 1380

PEMANFAATAN SEQUOIA MIDDLEWARE CLUSTERING DATABASE PADA APLIKASI AKUNTANSI BERBASIS WEB

GIANT RACHMAN J
NRP 5203 100 019

Pembimbing
Bambang Setiawan, S.Kom, MT

JURUSAN SISTEM INFORMASI
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2008

ITS
Institut
Teknologi
Sepuluh Nopember

FINAL PROJECT- CF 1380

DESIGNING INFORMATION SYSTEM OF SCHOOL DATA COLLECTION BASE ON ICONIX PROCESS "WITH CASE STUDY DINAS P DAN K JAWA TIMUR

GIANT RACHMAN J
NRP 5203 100 019

Supervisor
Bambang Setiawan, S.Kom, MT

INFORMATION SYSTEM DEPARTMENT
Information Technology Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2008

PERANCANGAN SISTEM INFORMASI PENDATAAN SEKOLAH

"STUDI KASUS DINAS P DAN K JAWA TIMUR"

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Oleh :

GIANT RACHMAN J.
5203 100 019

Surabaya, Januari 2008

KETUA
JURUSAN SISTEM INFORMASI

Ir. A. HOLIL NOOR ALI M.KOM
NIP 131 996 150

PERANCANGAN SISTEM INFORMASI PENDATAAN SEKOLAH

"STUDI KASUS DINAS P DAN K JAWA TIMUR"

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Jurusan Sistem Informasi
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :

Giant Rachman J.
5203 100 019

Disetujui Tim Penguji : Tanggal Ujian : 28 Desember 2007
Periode Wisuda : Maret 2008

Bambang Setiawan, S.Kom., M.T. **(Pembimbing)**

Ir. Khakim Ghozali **(Penguji 1)**

Arif Wibisono **(Penguji 2)**

PERANCANGAN SISTEM INFORMASI PENDATAAN SEKOLAH BERDASAR PROSES ICONIX "STUDI KASUS DINAS P DAN K JAWA TIMUR"

Nama Mahasiswa	: Giant Rachman J.
NRP	: 5203 100 019
Program Studi	: Sistem Informasi FTIf – ITS
Dosen Pembimbing	: Bambang Setiawan, S.Kom, M.T.

Abstrak

Pendataan sekolah sangatlah penting untuk mengetahui kondisi terakhir dari sekolah, yang mana dari hal tersebut dapat dilakukan keputusan-keputusan yang strategis untuk meningkatkan kualitas pendidikan yang ada. Saat ini pendataan tersebut masih dilakukan secara manual yang memakan banyak waktu. Dengan adanya sistem informasi pendataan sekolah, diharapkan mampu mengurangi keterlambatan informasi dari yang disampaikan oleh pihak sekolah. Untuk merancang sistem tersebut digunakan metode yang bernama proses ICONIX, yang mana metode tersebut merupakan salah satu metode perancangan perangkat lunak berdasar dari use case driven dengan menggunakan UML 2.0.

Kata kunci: pendataan sekolah, UML, proses ICONIX

Abstract

School data colection is very important to knows latest condition of school, which is that can be make strategic decision for increasing quality of education. At this time collecting data still use manual method. By existing Information System of School Data Collection, we hope it can decrease late of information from school. For designing this system ICONIX proces is useds, which is that method one of designind method who using use case driven method with UML 2.0.

KATA PENGANTAR

Alhamdulillahirabbilalamiin atas segala karunia dan kasih sayang-NYA, sehingga tugas akhir berjudul PERANCANGAN SISTEM INFORMASI PENDATAAN SEKOLAH BERDASAR PROSES ICONIX

"STUDI KASUS DINAS P DAN K JAWA TIMUR" dapat terselesaikan dan menghantarkan penulis menjadi sarjana komputer dari Program Studi Sistem Informasi, Fakutas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Terima kasih dan penghargaan setinggi-tingginya penulis sampaikan kepada:

1. Bambang Setiawan, S.Kom, M.T. selaku Dosen Pembimbing yang telah memberikan bimbingan dan motivasi kepada penulis;
2. Semua dosen pengajar di Program Studi Sistem Informasi ITS yang telah memberikan ilmu yang berharga kepada penulis;
3. Dr. H. Rasiyo M.Si, selaku Kepala Dinas P dan K Provinsi Jawa Timur;
4. Drs. Zainal Arifin, MA, selaku Kepala Sub-Dinas Penyusunan Program P dan K Provinsi Jawa Timur;
5. Drs. Hudiyono, M.Si, selaku Kepala Seksi PPD P dan K Provinsi Jawa Timur;
6. Para staf Sub-Dinas Sun Gram dan Seksi PPD yang telah memberikan berbagai informasi tentang pendataan sekolah;

7. Seluruh staf karyawan Program Studi Sistem Informasi dan karyawan Fakultas Teknologi Informasi atas dukungannya sehingga tugas akhir ini dapat terselesaikan;
8. Teman-teman senasib seperjuangan saat pengerjaan tugas akhir. Terimakasih atas dukungan moril dan dorongan semangat yang diberikan;
9. Semua teman-teman di Sistem Informasi, terima kasih telah menjadi bagian dari SI;
10. Berbagai pihak yang belum sempat penulis sebutkan jasa-jasanya dalam mendukung penyusunan tugas akhir ini.

Penulis sangat menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu penulis mengharapkan komentar, kritik, dan saran dari berbagai pihak.

Akhirnya, penulis berharap semoga keberadaan tugas akhir ini bermanfaat banyak bagi ilmu pengetahuan dan berbagai pihak.

Surabaya, Januari 2007
Penulis

Daftar Isi

Daftar Gambar

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Mekanisme pendataan sekolah saat ini membutuhkan penyampaian informasi tentang sekolah (misal: data guru dan siswa, aset yang dimiliki) dalam waktu yang singkat, kenyataan untuk menyampaikan informasi dari sekolah ke kabupaten/kota dibutuhkan waktu paling lambat selama tiga hari. Sedangkan dari kabupaten/kota ke provinsi diperlukan waktu paling lambat selama dua minggu bargantung dari jarak kabupaten/kota ke ibukota provinsi. Jadi bisa diperkirakan informasi sampai dari sekolah ke provinsi diperlukan waktu kurang lebih selama satu bulan. Hal ini sangat menyulitkan pengambilan keputusan, dikarenakan informasi di sekolah sudah ada kemungkinan telah berubah.

Namun dengan perkembangan teknologi informasi penyampaian berkas tersebut dapat diselesaikan dalam waktu yang cukup singkat. Sehingga pengambil keputusan dapat memberikan keputusan dengan kondisi yang sesuai dengan kondisi saat di sekolah. Untuk itu dipelukan sistem informasi yang mampu menggantikan sistem pengiriman berkas yang lama.

1.2 Permasalahan

Permasalahan yang ada dalam tugas akhir ini adalah:

1. Menentukan entitas-entitas yang merupakan bagian dari kebutuhan fungsional dan non-fungsional.
2. Membuat rancangan sistem yang mampu merepresentasikan dari sistem pengiriman berkas.

3. Membuat rancangan sistem yang dapat menyajikan laporan-laporan yang dibutuhkan beserta proses pengumpulan datanya.

1.3 Batasan

Dari permasalahan yang telah disebutkan di atas, maka batasan-batasan dalam tugas akhir ini adalah:

1. Tugas akhir ini sebatas perancangan dan tidak sampai implementasi rancangan.
2. Informasi yang diperoleh sistem berasal dari propinsi, kabupaten/kota, sekolah.

1.4 Tujuan

Tujuan dari tugas akhir ini adalah :

1. Membuat spesifikasi kebutuhan perangkat lunak.
2. Membuat rancangan sistem informasi pendataan sekolah.

1.5 Manfaat

Manfaat yang diberikan oleh tugas akhir ini adalah sebagai berikut :

1. Menghemat waktu penyampaian informasi.
2. Menghemat biaya pengiriman.
3. Memberikan informasi yang akurat tentang kondisi sekolah.

BAB 2
TINJAUAN PUSTAKA

2.1 Pedoman Mekanisme Pendataan

Pengumpulan data Sekolah Menengah Atas (SMA) dan Madrasah Aliyah (MA) ini merupakan kegiatan rutin pusat data dan informasi pendidikan, balitbang, depdiknas yang dilakukan sejak tahun 1970an. Kegiatan pendataan ini bertujuan untuk memperoleh data sekolah yang obyektif dalam rangka memenuhi kebutuhan informasi yang digunakan untuk berbagai keperluan penyusunan statistik, perencanaan, pembinaan, penelitian, pengelolaan, dan pengambilan keputusan baik di tingkat propinsi maupun daerah oleh berbagai instansi.

Obyektifitas data sangat diperlukan, karena format ini merupakan satu-satunya instrumen pendataan yang digunakan secara khusus untuk mengukur tingkat keberhasilan program pembinaan pendidikan yang telah dilakukan oleh pemerintah.

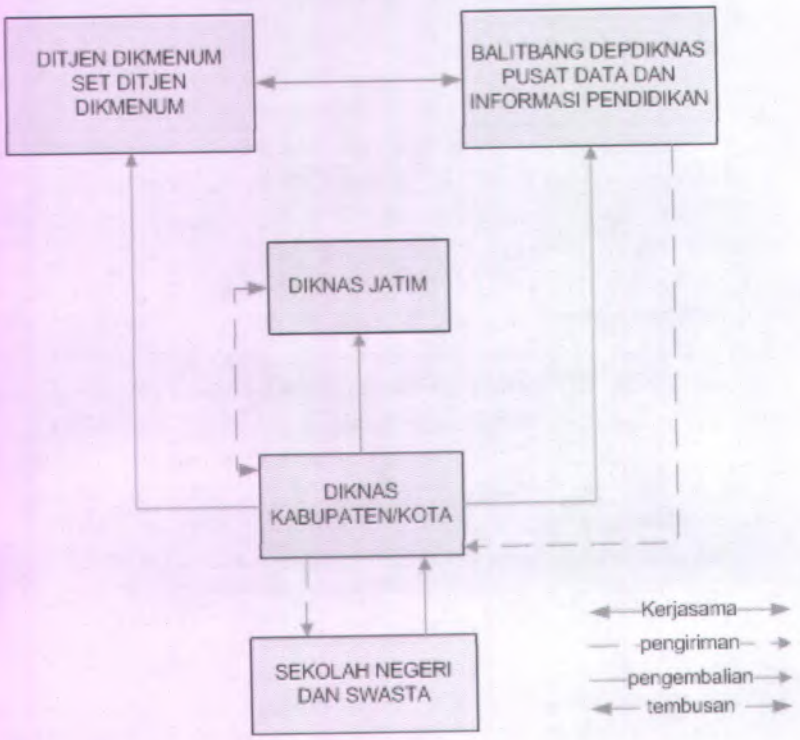
2.1.1 Lingkup Pendataan

Pendataan Sekolah meliputi SMA dan SMK atau sekolah-sekolah lain yang berada di bawah pengelolaan depdiknas. Sejak tahun 1995/1996 dengan disatukannya wajib belajar 9 tahun dengan pendataan rutin yang dikembangkan oleh pusat data dan informasi pendidikan, balitbang, depdiknas. Laporan ini juga disebarkan ke Madarasah Aliyah.

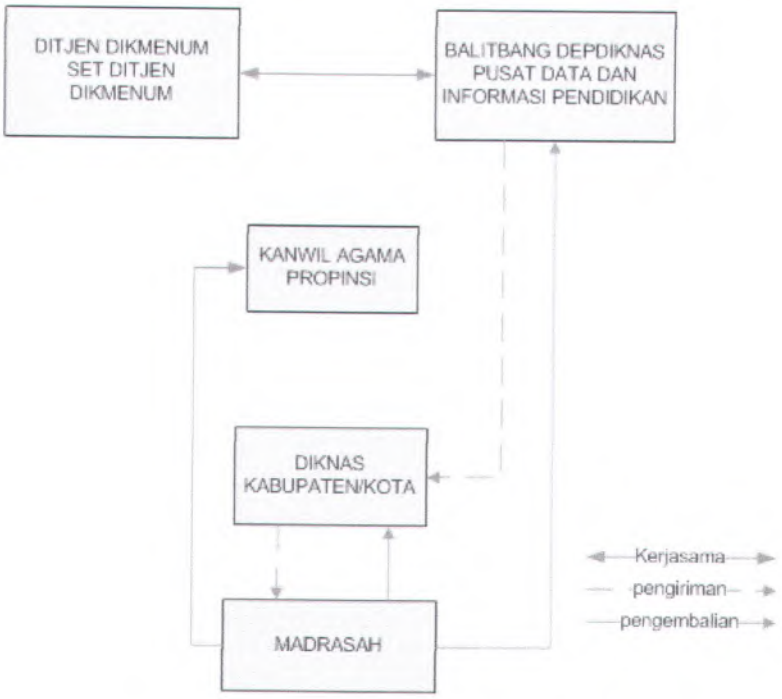
2.1.2 Mekanisme Pendataan Secara Manual

Direktorat Jendral pendidikan menengah umum bersama badan penelitian dan pengembangan depdiknas menyebarkan kuisioner ke pemerintah kabupaten/kota untuk disebarkan ke sekolah-sekolah. Oleh sekolah dilakukan pengisian kuisioner dan kemudian dikembalikan ke

pemerintah kabupaten/kota untuk diteruskan ke diknas Jawa Timur.



Gambar 2.1 Bagan untuk SMA dan SMK



Gambar 2.2 Bagan untuk Madrasah Aliyah

Direktorat jendral pendidikan menengah umum bersama badan penelitian dan pengembangan depdiknas menyebarkan kuisioner ke pemerintah kabupaten/kota untuk disebarkan ke madrasah-madrasah. Oleh madrasah dilakukan pengisian kuisioner dan kemudian dikembalikan kanwil departemen agama propinsi.

2.2 Unified Modelling Language (UML)

UML adalah standar untuk membuat sebuah model yang merepresentasikan Object-Oriented Software dan sistem bisnis. UML merupakan Framework untuk mengukur dan menguji teknik perancangan. UML mengkombinasikan teknik-teknik

penggambaran diagram yang dilakukan oleh para pengembang software selama 40 tahun terakhir.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *Class* dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti C++, Java, C# atau VB.NET.

2.2.1 Definisi UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Kebutuhan adanya standardisasi metode telah menjadi kebutuhan berdasarkan berbagai alasan. Terdapat banyaknya metode dan notasi yang berkompetisi satu sama lain telah membingungkan pengguna dalam memutuskan metode mana yang harus dipakai. Sejak standarisasi metodologi menjadi isu, *Object Management Group* (OMG) telah mengeluarkan *Request for Proposa*l (RFP) pada Juni 1996 untuk mendorong perusahaan-perusahaan sistem informasi, developer software, dan pengguna sistem komputer membuat sebuah RFP bersama. Satu perusahaan software, Rational Software, telah membentuk konsorsium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling System* (UML) sebagai bahasa standar dalam OOAD. Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 2.0. Kontribusi untuk UML telah dihasilkan dari perusahaan-perusahaan besar, diantaranya adalah Hewlett-Packard, IBM, Microsoft, Oracle, Rational Software, IntelliCorp, dan Electronic Data Services Corporation. Kolaborasi ini telah menjadikan UML sebagai bahasa pemodelan yang well-defined, ekspresif, tangguh, dan dapat digunakan secara luas.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan syntax. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering).*

2.2.2 Konsep Dasar UML

Secara garis besar konsep dasar dari UML dapat dijelaskan pada tabel 2.1 (dari buku *The Unified Modeling Language Reference Manual Second Edition*, hal 26-27) :

Tabel 2.1 Konsep Dasar UML

MAJOR AREA	VIEW	DIAGRAMS	MAIN CONCEPTS
Structural	Static view	*Class* diagram	*Class*, association, generalization, dependency, realization, interface
	Use case view	*Use case* diagram	*Use case*, actor, association, extend, include, *use case* generalization
	Implementation view	Component diagram	Component, interface, dependency, location
	Deployment	Deployment	Node,

	view	diagram	component, dependency, location
Dynamic	State machine view	State chart diagram	State, event, transition, action
	Activity view	Activity diagram	State, activity, completion transition, fork, join
	Interaction view	*Sequence* diagram	Interaction, object, message, activation
		Collaboration diagram	Collaboration, interaction, collaboration role, message
Model management	Model management view	*Class* diagram	Package, subsystem, model
Extensibility	all	all	Constraint, stereotype, tagged values

Dari tabel 2.1 diketahui bahwa UML merupakan turunan dari beberapa metode yang mempunyai kumpulan diagram grafis sebagai kombinasi dari konsep pemodelan data (*entity relationship diagram*), pemodelan bisnis (*work flow*), pemodelan objek, dan pemodelan komponen. Diagram grafis tersebut merupakan tampilan dari beberapa level abstraksi yang dapat digunakan secara bersama oleh semua proses pada seluruh

lifecycle pengembangan *software* serta pada implementasi ke beberapa teknologi yang berbeda.

2.3 Proses ICONIX

Proses Iconix merupakan *Agile software development process* yang didahului dengan *Rational Unified Process* (RUP) dan *Extreme Programming* (XP), tapi berbagi kesamaan. Seperti RUP, proses Iconix dikendalikan *use case* tapi lebih sederhana seperti XP. Proses Iconix menekankan pada diagram UML untuk merubah *use case* mencadi kode program.

Proses Iconix memiliki aktivitas utama yaitu analisa robustness, yang meruakan metode untuk menjembatani celah antara analisa dan desain. Analisa robustness mengurangi ambiguitas dari deskripsi *use case*,dengan memastikan bahwa yang ditulis merupakan isi dari domain model yang menyertai. Proses ini membuat *use case* lebih mudah untuk didesain, diuji coba, dan diperkirakan.

2.3.1 *Milestone* Pertama : Tinjau Ulang Kebutuhan

Sebelum memulai proses iconix terdapat beberapa analisa kebutuhan yang harus dipenuhi. Dari analisa *use case* dapat teridentifikasi, terbentuk domain model, dan beberapa tampilan untuk purwarupa antarmuka grafis.

2.3.2 *Milestone* Kedua : Tinjau Ulang Desain Pendahuluan

Setelah *Use case* berhasi teridentifikasi, maka dapat diketahui bagaimana pengguna berinteraksi dengan sistem. Analisa *robustness* digunakan untuk mencari kesalahan potensial dalam sebuah *use case* dan domain model diupdate yang dikarenakan oleh kesalahan tadi. *Use case* penting untuk mengidentifikasi bagaimana pengguna melakukan interaksi dengan sistem yang direncanakan. Hal itu juga melengkapi pengembang sistem dengan sesuatu untuk

ditunjukan kostumer dan memverifikasi hasil dari analisa kebutuhan yang telah benar.

2.3.3 *Milestone* ketiga: Tinjau Ulang Desain Lebih Detil

Selama dalam fase ini domain model dan *use case* dari *milestone* kedua digunakan untuk mendisain sistem yang akan dibangun. *Class* diagram dibuat dari domain model dan *use case* akan digunakan untuk membuat *sequence* diagram.

2.3.4 *Milestone* Keempat : Implementasi

Unit uji coba ditulis untuk melakukan verifikasi terhadap sistem apakah sesuai dengan *use case*, dan *sequence* diagram. Akhirnya penulisan program dengan menggunakan panduan dari *Class* diagram dan *sequence* diagram.

BAB 3
METODOLOGI

Metodologi dalam tugas akhir diperlukan sebagai panduan dalam proses pengerjaan tugas akhir agar tahapan dalam pengerjaan tugas akhir dapat berjalan secara terarah dan sistemtis. Berikut ini merupakan serangkaian metodologi pengerjaan tugas akhir yang dilakukan.

Dalam pengerjaan tugas akhir metode yang digunakan adalah proses Iconix yang memiliki peta jalan pada gambar 3.1.

3.1 Tahapan Pra Analisis

Tahap ini dilakukan sebagai upaya untuk mencari kebutuhan dari sistem. Ada beberapa metode yang digunakan untuk tahapan pra analisis yaitu sebagai berikut:

3.1.1 Pengumpulan Data dan Pencarian Kebutuhan

Untuk mendapatkannya dapat dilakukan dengan cara wawancara, mencari dokumen petunjuk pelaksanaan, dan melakukan *survey* langsung. Dari cara tersebut akan didapat kebutuhan dari sistem yang akan dibuat.

3.1.2 Pembuatan Purwarupa GUI

Tingkah laku kebutuhan secara detil untuk interaksi pengguna dan respon sistem terhadap aksi. Untuk sistem perangkat lunak, interaksi antara pengguna dan sistem ditempatkan via layar, jendela, atau halaman web.

3.1.3 Pembuatan Cerita Dari GUI (GUI *Story Board*)

Setelah mendapatkan kebutuhan tersebut maka dapat dibuatkan purwarupa dari antarmuka pengguna. Dari purwarupa tersebut maka akan didapatkan alur dari perjalanan antarmuka yang ada dalam sistem yang akan dibuat.

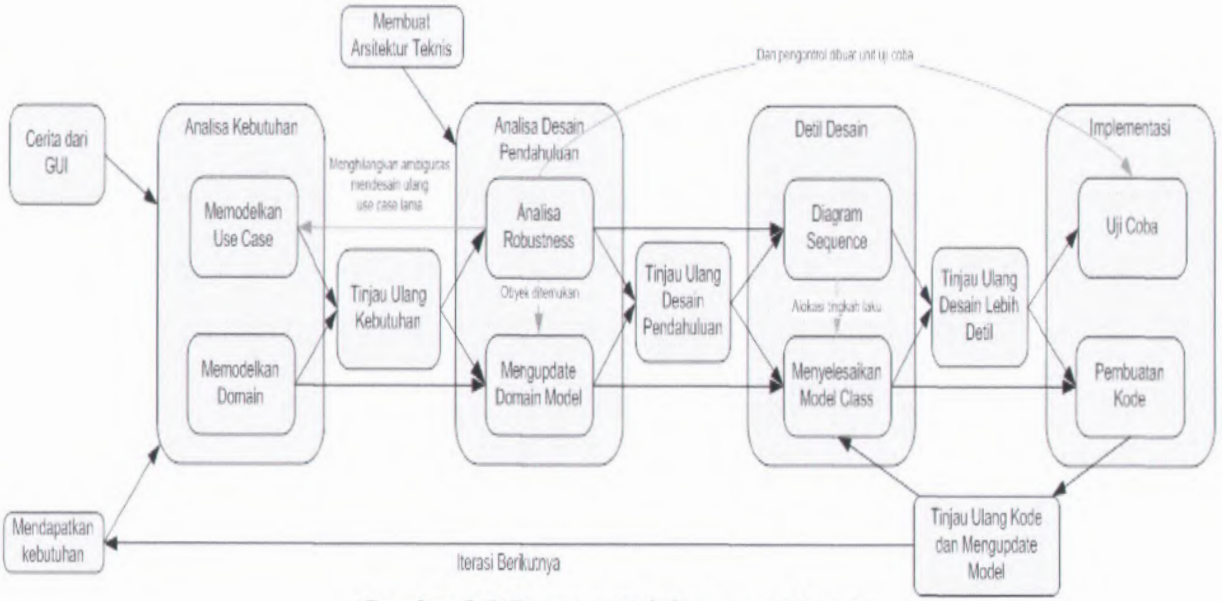
3.1.4 Menentukan Kebutuhan Fungsional

Mendefinisikan apa yang seharusnya sistem mampu lakukan. Kebutuhan fungsional akan menangani semua fungsi bisnis. Untuk mendapatkan kebutuhan fungsional ada 10 panduan yang disarankan.yaitu :

1. Gunakan perkakas yang mendukung hubungan antara kebutuhan dengan *use case*.
2. Hubungkan kebutuhan dengan *use case*.
3. Hindari kebutuhan yang disfungsional dengan memisahkan detil-detil fungsional dari spesifikasi tingkah laku.
4. Tulis sekurang-kurangnya satu kasus uji coba untuk setiap kebutuhan.
5. Perlakukan kebutuhan sebagai warga kelas atas dalam model ini.
6. Memperjelas perbedaan antar tiap kebutuhan.
7. Hindari sindrom "dokumen *monolithic* yang besar".
8. Buat perkiraan dari skenarion *use case*, tidak dari kebutuhan fungsional.
9. Jangan takut dengan contoh ketika menulis kebutuhan fungsional.
10. Jangan membuat kebutuhan fungsional dalam bahasa teknis.

3.2 Analisa Kebutuhan Pengguna

Tahap ini dapat dilakukan setelah ada data-data yang diperlukan. Dalam tahap ini terdapat beberapa proses dan memiliki satu pencapaian yaitu tinjauan ulang kebutuhan, yang mana dari tinjauan ulang kebutuhan akan memastikan bahwa *use case* akan sesuai dengan perkiraan kostumer.



Gambar 3.1 Bagan peta jalan proses Iconix

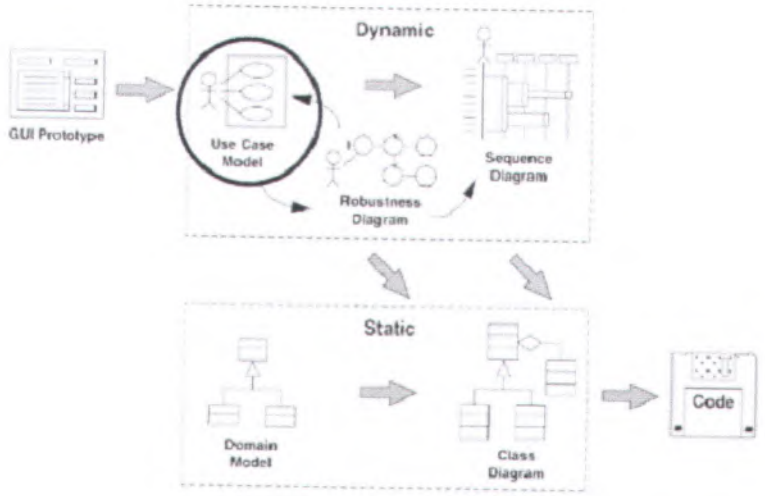
3.2.1 Menganalisa Tingkah Laku Kebutuhan (Bagaimana Pengguna dan Sistem Berinteraksi)

Proses Iconix merupakan pendekatan berdasar dari skenario, yang mana mekanisme untuk dekomposisi dan memodelkan sistem berdasar dari skenario-skenario yang yang ada. Tapi ketika menggunakan proses Iconix, tujuan utamanya untuk membuat sebuah desain berorientasi obyek yang dapat dibuat kode. Karena itu dibutuhkan untuk menghubungkan skenario ke obyek.

3.2.2 Memodelkan *Use case*

Use case mendeskripsikan bahwa user akan berinteraksi dengan sistem dan bagaimana sistem akan merespon. Dalam Iconix process terdapat beberapa tambahan dalam memodelkan use case, teradapat dua tambahan tentang keterkaitan antara use case, selain include dan extends, terdapat precedes dan invokes, precedes merupakan hubungan use case yang dilakukan setelah use case sebelumnya selesai dilakukan, invokes merupakan hubungan use case yang dilakukan pada saat use case pertama dieksekusi. Ada 10 panduan yang disarankan, yaitu :

1. Ikuti dua aturan paragraf (skenario normal dan skenario alternatif).
2. Mengorganisasi *use case* dengan aktor dan diagram *use case*.
3. Menuliskan *use case* dalam kalimat aktif.
4. Menuliskan *use case* mengguna even atau aliran respon, deskripsikan sisi dari dialog sistem/pengguna.
5. Gunakan cerita dari GUI, purwarupa, atau tampilan awal dan lain-lain.
6. Ingat bahwa *use case* yang dibuat berdasar spesifikasi tingkah laku.
7. Tulis *use case* dalam kontek model obyek.



Gambar 3.2 Memodelkan *Use case*

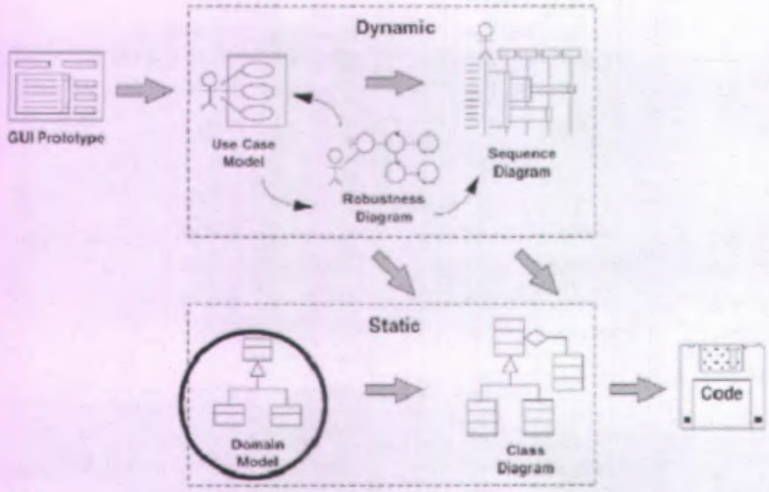
8. Tulis *use case* dengan sturktur S-P-O dalam kalimat.
9. Referensikan *Class* domain dari nama (*class*).
10. Referensikan *Class* boundary dari nama (*boundary*)..

3.2.3 Memodelkan Domain

Memodelkan domain bertujuan untuk menghilangkan ambiguitas dari kebutuhan sistem yang ada. Untuk memodelkan domain ada 10 panduan yang disarankan.yaitu

1. Fokus ke obyek nyata.
2. Gunakan hubungan generalisasi (is-a) dan agregasi (has-a) untuk menunjukkan bagaimana keterkaitan antar obyek.
3. Batasi domain inisial untuk beberapa jam.
4. Atur kelas berdasar dari absraksi dalam pemsalahan domain.
5. Jangan keliru model domain untuk sebuah model data.

6. Jangan bingung obyek dengan tabel basisdata.
7. Gunakan model domain untuk daftar istilah proyek.



Gambar 3.3 Memodelkan domain

8. Buat insial model domain sebelum menulis sebuah *use case* untuk menghindari ambiguitas.
9. Jangan memperkirakan *Class* diagram akhir secara tepat dalam memodelkan domain.
10. Jangan mengambil layar dan GUI yang lebih spesifik dalam domain model.

3.2.4 *Milestone* Pertama : Meninjau Ulang Kebutuhan

Setelah semua kebutuhan didapatkan maka diperlukan untuk memastikan bahwa kebutuhan cukup dimengerti baik oleh developer maupun kustomer. Ada 10 panduan yang disarankan, yaitu :

1. Yakinkan bahwa *domain model* mendeskripsikan sekurang-kurangnya 80% dari abstraksi paling penting dari domain permasalahan, dalam bahasa non-teknis sehingga dapat dimengerti oleh kostumer.
2. Pastikan domain model menunjukkan generalisasi dan agragasi antar obyek domain.
3. Pastikan bahwa *use case* mendeskripsikan basik dan alternatif aksi, dalam kalimat aktif.
4. Jika terdapat daftar dari kebutuhan fungsional. Pastikan tidak terserap kedalam kalimat aktif dari *use case*.
5. Pastikan *use case* terorganisasi dalam paket-paket, yang mana setiap paket terdiir minimal satu *use case*.
6. Pastikan *use case* dituliskan dalam kontek dari model obyek.
7. Ambil *use case* dalam kontek dari antarmuka pengguna.
8. Lengkapi *use case* dengan deskripsi dari beberapa cerita GUI, purwarupa GUI, dan tampilan awalan.
9. Tinjau ulang *use case*, domain model, dan tampilan awal dengan pengguna akhir, stakeholder, dan orang marketing.
10. Tinjau ulang semua secara terstruktur.

3.3 Analisa/Desain Pendahuluan

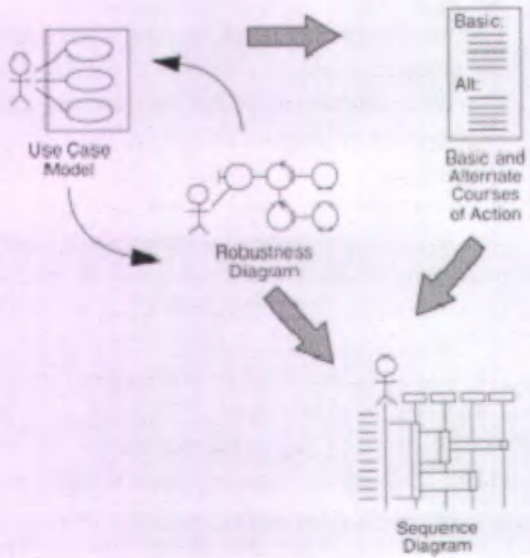
Analisa yang dimaksud tentang bagaimana membangun sistem yang benar. Desain yang dimaksud tentang bagaimana membuat sistem itu benar.

Langkah desain pendahuluan meliputi eksplorasi desain yang diperlukan untuk mengerti kebutuhan, menyempurnakan, dan menghilangkan ambiguitas dari kebutuhan sebagai eksplorasi desain, dan pertautan tingkah laku kebutuhan.

3.3.1 Menganalisa Robustness

Untuk mendapatkan dari use case ke detail desain (dan kemudian kode), diperlukan tautan dari use case ke

obyek. Analisa robustness membantu untuk menjembatani celah antara analisa dengan desain dengan melakukan penepatan.



Gambar 3 4 Analisa robustness

Ada 10 panduan yang disarankan, yaitu :

1. Tempel *use case* secara langsung kedalam robustness diagram.
2. Taruh entitas *Class* dari domain model, dan tambahkan semua yang tidak ada.
3. Perkirakan unutk menulis ulang setiap usecase ketika menggambar di diagram robustness.
4. Buat obyek boundary untuk setiap tampilan GUI, dan berikan nama tidak ambigu.
5. Ingat bahwa *controler* tidak hanya kadang-kadang obyek kontrol nyata; tapi juga fungsi logis perangkat lunak.

6. Jangan kuatir tentang arah dari anak panah pada diagram *robustness*.
7. Diperbolehkan untuk menaruh *use case* ke dalam diagram *robustness* jika hal ini dari *use case parent*.
8. Diagram *robustness* repesentasi dari desain konseptual pendahuluan dari sebuah *use case*.
9. *Boundary* dan *class* entitas dalam diagram robustness akan secara umum menjadi obyek instan dari diagram *sequence*, dimana kontroler menjadi pesan.
10. Ingat diagram *robustness* merupakan gambar obyek dari *use case*, dengan tujuan untuk memperjelas *use case* dan model obyek.

3.3.2 Mengupdate Domain Model

Pada tahap ini dilakukan penyesuaian domain model terhadap diagram *robustness* yang telah dibuat. Dimana *class-class* yang tadinya tidak ada dalam domain model ditambahkan, sehingga domain model mendekati dengan *class* diagram yang akan dibuat.

3.3.3 Membuat Arsitektur Teknis

Tujuan dari tahap ini untuk mendapatkan arsitektur dari sistem secara keseluruhan. Arsitektur teknis bisa didapat melalui framework yang akan dipakai, ataupun bedasar dari proses bisnis yang sudah ada.

3.3.4 *Milestone* Kedua : Meninjau Ulang Desain Pendahuluan

Fase ini membantu untuk memastikan diagram *robustness*, domain model, dan *use case* saling sesuai satu dengan yang lain. Tinjauan ulang ini merupakan gerbang antara desain pendahuluan dan detil desain. Ada 9 panduan yang disarankan, yaitu :

1. Untuk setiap *use case*, pastikan *use case* sesuai dengan diagram robustness.

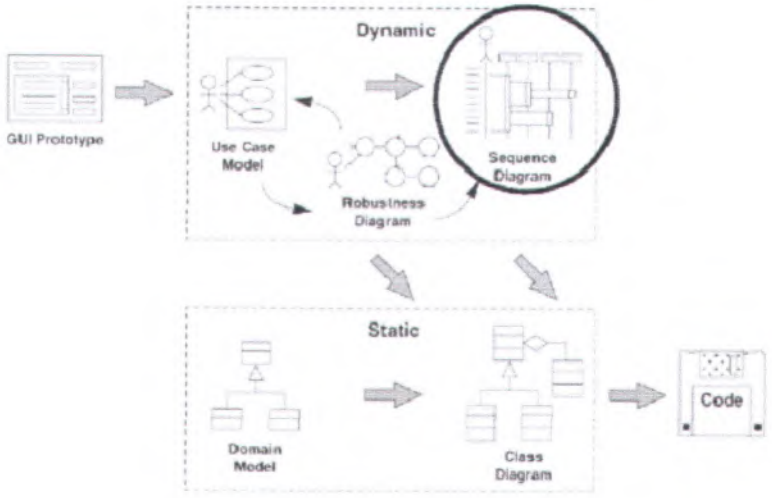
2. Pastikan bahwa semua entitas di diagram robustness tampak dengan domain model yang terupdate.
3. Pastikan bahwa aliran data antara kelas dan layar dapat ditelusuri.
4. Jangan lupa dengan alternatif skenario, dan jangan lupa menuliskan tingkah lakunya untuk setiap item yang ditemui.
5. Pastikan setiap *use case* melindungi sisi dialog antara pengguna dan sistem.
6. Pastikan tidak ada pelangaran aturan dalam analisa *robustness*.
7. Pastikan tinjau ulang termasuk dengan nonteknis (kustomer, tim marketing, dll.) dan tim teknis (programer).
8. Pastikan *use case* dalam kontek obyek model dan dalam kontek GUI.
9. Pastikan diagram robustness tidak menampilkan sampai level detil.

3.4 Desain Lebih Detil

Desain lebih detil merupakan tentang membangun sistem dengan benar. Dalam proses ini diharapkan telah mengerti bagaimana sistem yang benar, karena telah belajar banyak untuk mengerti.

3.4.1 Membuat Diagram *Sequence* (Pengalokasian Tingkah Laku Ke *Class*)

Proses Iconix menggunakan diagram sequence sebagai kendaraan untuk mengeksplorasi detil desain dari sistem ke skenario demi skenario.



Gambar 3 5 Diagram sequnce

Ada 10 panduan yang disarankan, yaitu :

1. Mengerti dengan yang digambarkan dalam diagram *sequence*.
2. Buat diagram *sequence* untuk setiap *use case*, dengan semua skenario yang mungkin
3. Mulai diagram *sequence* dengan *boundary*, entitas, *actor,* dan *use case* dari analisa *robustness*.
4. Gunakan diagram *sequence* untuk menunjukkan tingkah laku dari *use case* dalam menyelesaikan obyek.
5. Pastikan peta *use case* ke pesan dilawati dalam diagram *sequence*.
6. Jangan terlalu banyak menghabiskan waktu dalam fokus di control.
7. Penugasan ke *Class* ketika menggambar pesan.
8. Tinjau ulang diagram *Class* ketika melakukan penugasan ke *Class*.
9. Prefaktor disain ke dalam diagram *sequence* sebelum membuat kode.

10. Bersihkan model statis sebelum melakukan tinjaun ulang terhadap desain.

3.4.2 Membuat Diagram *Class*

Pada tahap ini diagram *class* dibuat berdasar dari semua tahapan yang telah dilakukan. Diagram *class* memiliki semua entitas yang ada pada diagram *robustness,* dan memiliki fungsi-fungsi operasi yang ada pada diagram *sequence*, dan memiliki atribut yang ada pada GUI.

3.4.3 Membersihkan Model Statis

Fase ini setidak sebagai langkah arkhir sebelum melakukan tinajuan ulang terhadap detil desain. Dengan demikian pekerjaan harus sesuai dengan proses bisnis, kebutuhan proyek, aplikasi desain *framework*, *topology Deployment*, dan seterusnya.

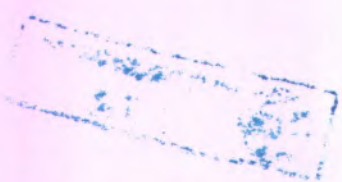
3.4.4 *Milestone* Ketiga : Meninjau Ulang Desain Lebih Detil

Tinjauan ulang membantu untuk meraih tiga tujuan penting, sebelum membuat kode program dari *use case* :

- Memastikan bagaimana desain lebih detil sesuai dengan apa yang dispesifikasikan kebutuhan.
- Meninjau ulang kualitas desain.
- Memeriksa kelanjutan dari pesan dari diagram *sequence*.

Ada 10 panduan yang disarankan, yaitu :

1. Pastikan diagram *sequence* sesuai dengan *use case*.
2. Pastikan sekali lagi setiap diagram *sequence* untuk setiap aksi baik yang normal maupun alternatif.
3. Pastikan operasi telah teralokasikan ke dalam *Class*.
4. Tinjau ulang *Class-Class* dari diagram *Class* untuk memastikan mempunyai cocok dengan atribut dan operasi.



5. Jika desain merefleksikan dari pola atau detil implementasi, periksa apakah detil tersebut sesuai dengan diagram *sequence*.
6. Telusuri kebutuhan fungsional dan non-fungsional dari *use case* dan *Class* untuk memastikan semuanya telah tercakup.
7. Pastikan programer memeriksa desain dan yakinkan mereka dapat membuatnya dan desain dapat bekerja sesusai dengan yang dikehendaki.
8. Pastikan semua atribut dari diketik dengan besar, dan nilai kembali dan daftar dari parameter dari opersi lengkap dan benar.
9. Bangkitkan kode awal dari setiap *Class*, dan periksa lebih dekat.
10. Tinjau ulang rencana uji coba untuk rilis.

3.5 Implementasi

Tahap ini bertujuan menerapkan dari desain yang telah dibuat, dan juga untuk membuat unit uji coba dari diagram *Class*.

3.5.1 Membuat Kode Program

Ada 10 panduan yang disarankan, yaitu :

1. Pastikan membuat kode langsung dari desain.
2. Jika terjadi kesalahan dari desain rubahlah, tapi tinjau ulang prosesnya.
3. Gunakan insapeksi kode reguler.
4. Selalu pertanyakan *framework* yang dipilih.
5. Jangan biarkan *framework* mengambil alih bisnis.
6. Jika kode mulai tidak terkontrol hentikan dan tinjau ulang desain.
7. Jagalah desain dan kode agar tetap tersinkron.
8. Fokus ke unit uji coba ketika mengimplementasikan kode.
9. Jangan memberi komentar berlebihan dalam kode.

10. Tetap ingat untuk mengimplementasikan alternatif skenario seperti skenarion dasar.

3.5.2 Membuat Unit Uji Coba

Ketika kode dibuat, unit uji coba juga harus dibuat dan terkait dengan *use case*. Uji coba ini memperbolehkan untuk menyetujui kode diimplementasikan. Ada 10 panduan yang disarankan, yaitu :

1. Adopsi "*Testing mind-set*" dimana setiap *bug* yang ditemukan merupakan kemenangan dan bukanlah kekalahan. Jika ditemukan (dan diperbaiki) maka pengguna tidak akan memenukannya pada saat perangkat lunak dirilis.
2. Mengerti perbedaan metode dari *testing*, dan kapan dan mengapa digunakan untuk setiapnya.
3. Ketika unit diuji coba buatlah satu atau lebih unit uji coba untuk setiap pengontrol dari setiap diagram *robustness*.
4. Untuk sistem yang nyata gunakan elemen dari diagram state sabagai dasar untuk kasus uji coba.
5. Lakukan verifikasi dari tingkatan kebutuhan.
6. Gunakan matrik penelusuran untuk membantu varifikasi kebutuhan.
7. Lakukan tingkatan skenario untuk menerima hasil testing dari setiap *use case*.
8. Perluas ancaman dalam skenarion uji coba untuk melingkupi semua jalur.
9. Gunakan *framework* uji coba.
10. Jaga unit uji coba agar tetap jelas.

3.5.3 Perluas Ancaman Untuk Integrasi dan Skenario Uji Coba

Fase ini meliputi perluasan ancaman terhadap *use case*. Integrasi uji coba datang dari *use case*, meliputi :

1. Skenario awal.

2. Skenario Alternatif.

3.5.4 Meninjau Ulang Kode dan Mengupdate Model

Tujuan utamanya yaitu untuk mensinkronkan kode dan model sebelum proses SDLC berikutnya. Ada 10 panduan yang disarankan, yaitu :

1. Menyiapkan tinjauan ulang, dan memastikan semua anggota tim yang terlibat telah membaca.
2. Buat daftar item yang akan ditinjau ulang, berdasar dari *use case*.
3. Jabarkan setiap item dalam daftar menjadi checklist.
4. Tinjau ulang pada setiap tingkatan.
5. Mendapatkan data pada saat peninjauan ulang , dan menggunakannya untuk mengakumulasikan checklist untuk peninjauan kedepannya.
6. Tembuskan tinjauan ulang dengan daftar dari titik-titik aksi ke semua orang yang terkait.
7. Coba untuk focus dalam pendeteksian kesalahan, bukan untuk memperbaikinya.
8. Gunakan IDE untuk editor kode program.
9. Tetap "just formal enough" dengan checklist dan tembusan, tapi jangan melakukan birokrasi yang berlebihan.
10. Ingat upadate model juga tidak hanya kode.

BAB 4
ANALISA DAN DESAIN SISTEM

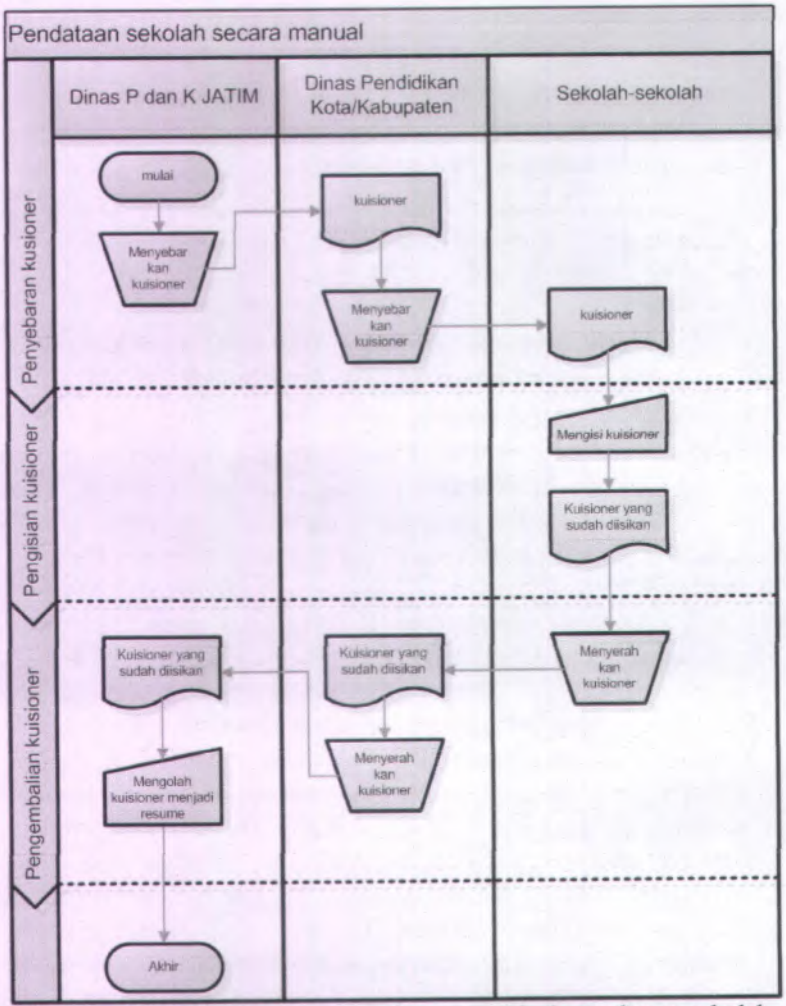
4.1 Hasil Tahapan Pra Analisis

Pada tahapan pengumpulan data didapat hasil dari mempelajari dokumen yang telah ada dan membaca buku pedoman.

4.1.1 Hasil Pengumpulan Data Dan Pencarian Kebutuhan

Pada tahapan ini dilakukan tanya jawab informal terhadap kepala sub-dinas bagian pengolahan data elektorik P dan K Jawa Timur, maka didapatkan buku petunjuk pelaksaan pedoman pelaksaan pendatan sebagai acuan untuk membuat sistem. Berdasar dari buku petunjuk pelaksaan pedoman pelaksaan pendatan yang diperoleh maka didapat analisa proses bisnis pada gambar 4.1.

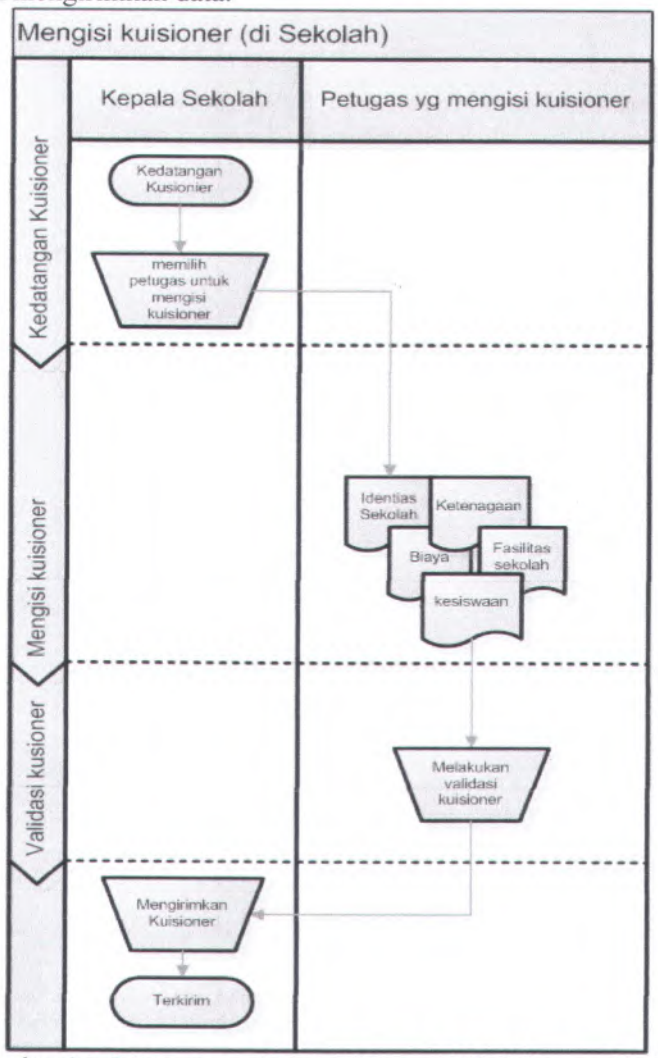
Awalnya Dinas P dan K Jawa timur membutuhkan data dari setiap sekolah untuk itu disebarkan kuisioner yang berisikan form-form identitas sekolah, kesiswaan, fasilitas sekolah, biaya, dan ketenagaan. Kuisioner tersebut nantinya disebarkan ke seluruh kabupaten/kota untuk kemudian disebarkan keseluruh sekolah yang ada di wilayah kabupaten/kota yang bersangkutan. Kusioner tersebut akan diisikan oleh pihak sekolah, yang mana petugas untuk mengisi kuisioner tersebut disetujui dan ditunjuk oleh kepala sekolah sebagai penanggung jawab. Setelah dilakukan pengisian terhadap kuisioner tersebut maka akan dikirimkan kembali ke dinas pendidikan kota/kabupaten setempat. Dari Dinas Pendidikan setempat dikirimkan lagi ke Dinas P dan K Jawa Timur.



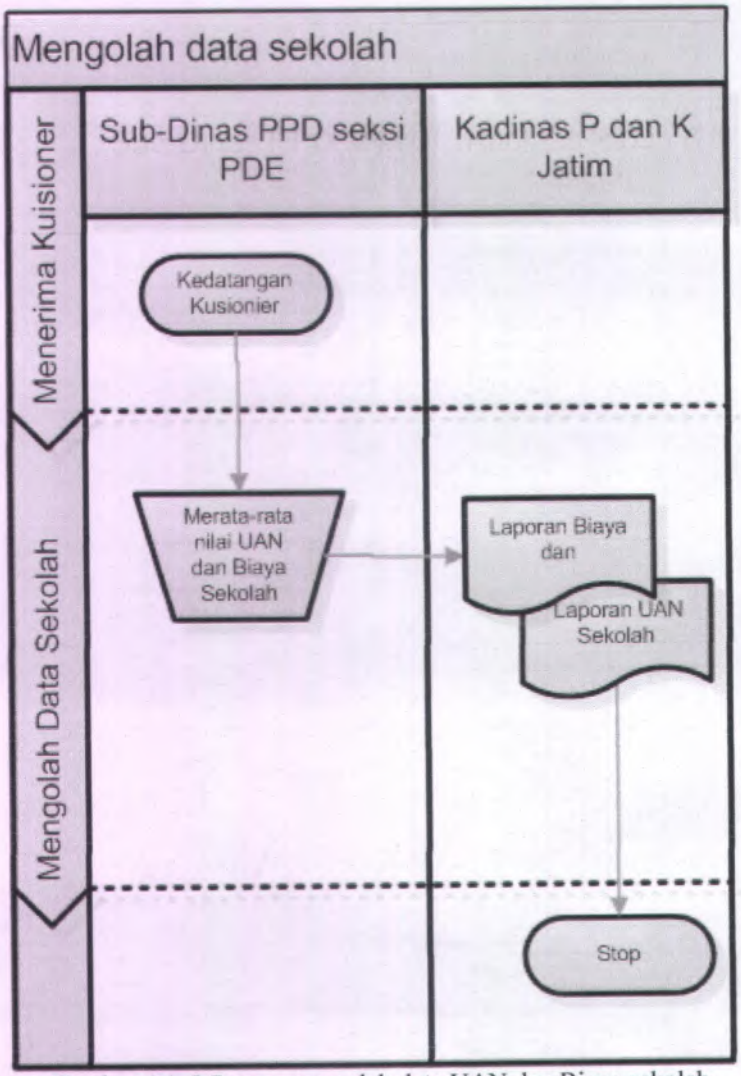
Gambar 4.1. 1 Gambaran umum alur proses bisnis pendataan sekolah

Sedangkan untuk pengoperasian perangkat lunak terbagi atas dua macam perangkat lunak yang mana terdiri atas server pendataan OnLine dan server aplikasi pendataan sekolah. Untuk server pendataan OnLine pengunjung web dapat melihat secara

langsung dari hasil yang dikirimkan oleh sekolah-sekolah yang telah mengirimkan data.



Gambar 4.1. 2 Proses pengisian kuisioner, validasi dan megirimkan kuisioner



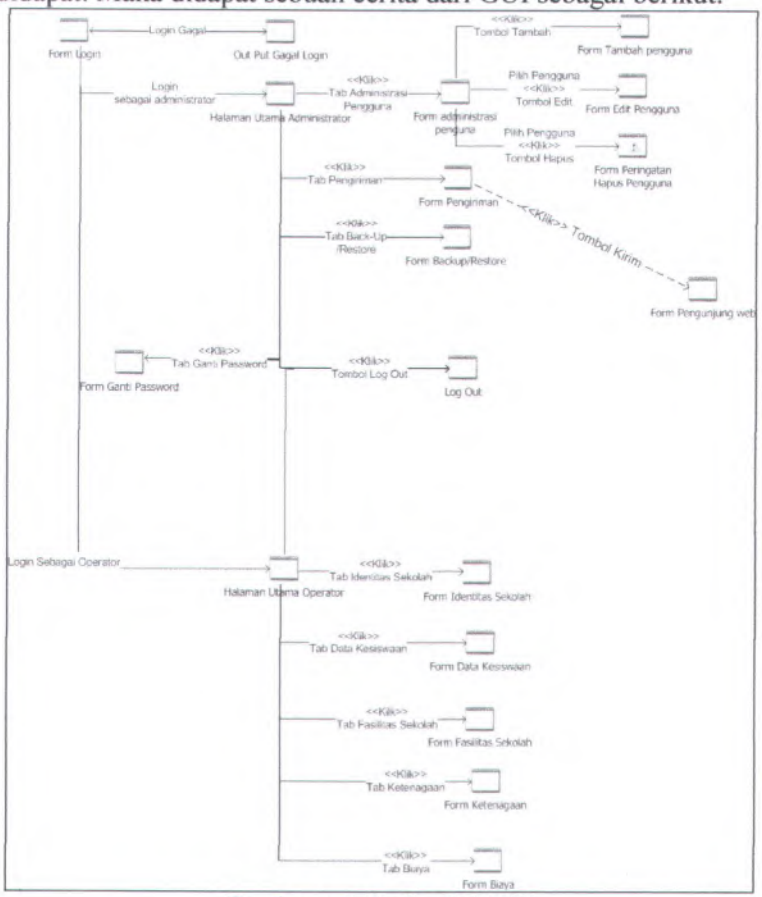
Gambar 4.1. 3 Proses mengolah data UAN dan Biaya sekolah

Untuk mengisikan data dapat dilakukan oleh operator yang telah disetujui oleh kepala sekolah, dan untuk

administrasi dan pengiriman dapat dilakukan oleh pihak sekolah yang telah disetujui oleh kepala sekolah.

4.1.2 Hasil Membuat Purwarupa dan Cerita dari GUI

Berdasarkan dari dokumen buku pedoman yang telah didapat. Maka didapat sebuah cerita dari GUI sebagai berikut.



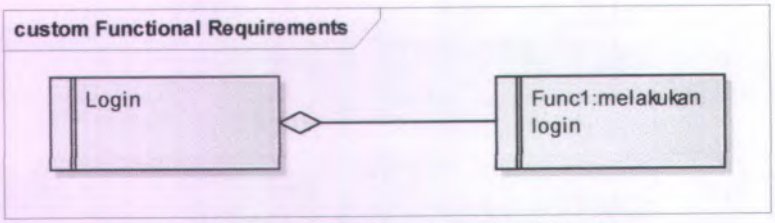
Gambar 4. 1 Cerita dari GUI

Untuk detil tentang purwarupa GUI dapat dilihat di lampiran analisa model.

4.1.3 Hasil Menentukan Kebutuhan Fungsional

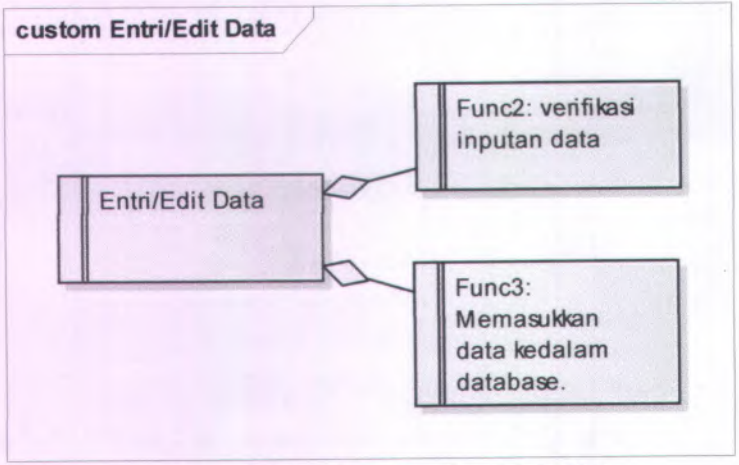
Kebutuhan fungsional merupakan hasil dari tahapan pencarian kebutuhan yang mana nantinya akan diolah menjadi *use case*.

Sesuai dengan permasalahan yang telah diidentifikasi sebelumnya dapat ditentukan kebutuhan fungsional yang dibutuhkan oleh sistem antara lain:



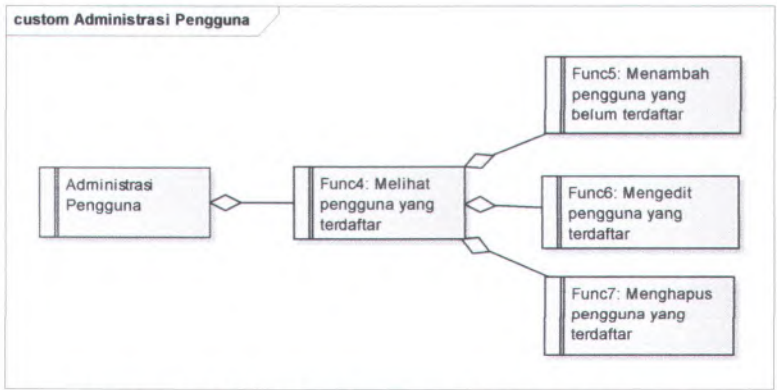
Gambar 4. 2 Kebutuhan Fungsional Login

Kebutuhan fungsional login diperoleh beradasar atas analisa proses bisnis pada tahapan memilih petugas untuk mengisi kuisioner, agar tidak terjadi penyalahgunaa.



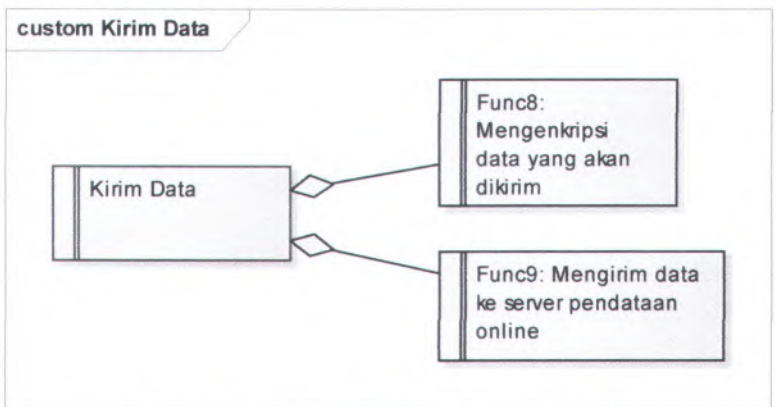
Gambar 4. 3 Kebutuhan Fungsional Entri Data

Kebutuhan fungsional entri data didapat pada tahapan analisa proses bisnis mengisi kuisioner dan melakukan validasi terhadap kuisoner.



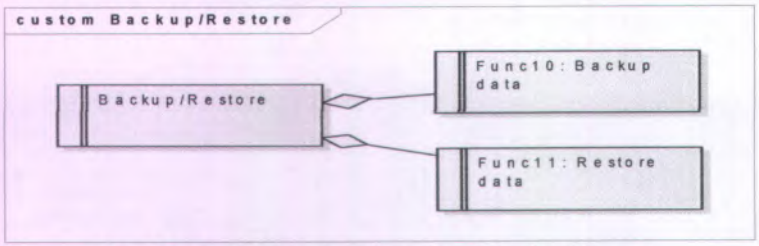
Gambar 4. 4 Kebutuhan fungsional administrasi pengguna

Kebutuhan fungsional administrasi pengguna didapat pada tahapan proses bisnis memilih petugas untuk mengisi kuisioner sehingga petugas yang melakukan pengisian lebih teratur.



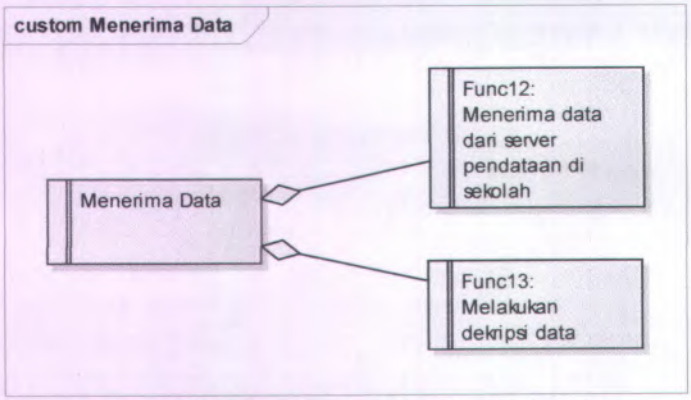
Gambar 4. 5 Kebutuhan fungsional kirim data

Kebutuhan fungsional kirim data didapat pada pada tahapan proses bisnis menyebarkan dan mengirimkan kusioner kembali.



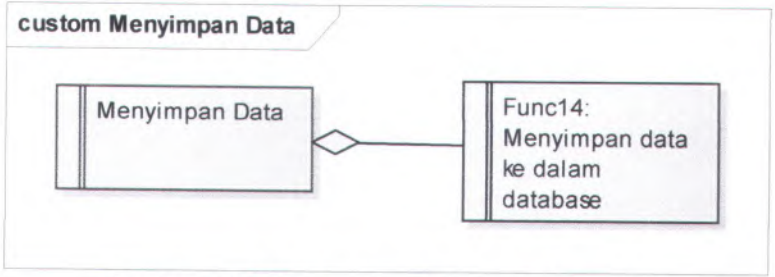
Gambar 4. 6 Kebutuhan fungsional *Backup/Restore*

Kebutuhan fungsional *backup/restore* tidak didapat pada tahapan yang ada pada proses bisnis hanya saja kebutuhan ini untuk mencegah kerusakan data, oleh sebab itu prioritasnya tidak begitu tinggi.



Gambar 4. 7 Kebutuhan fungsional menerima data

Kebutuhan fungsional menerima data didapat pada tahapan proses bisnis kedatangan kuisioner.



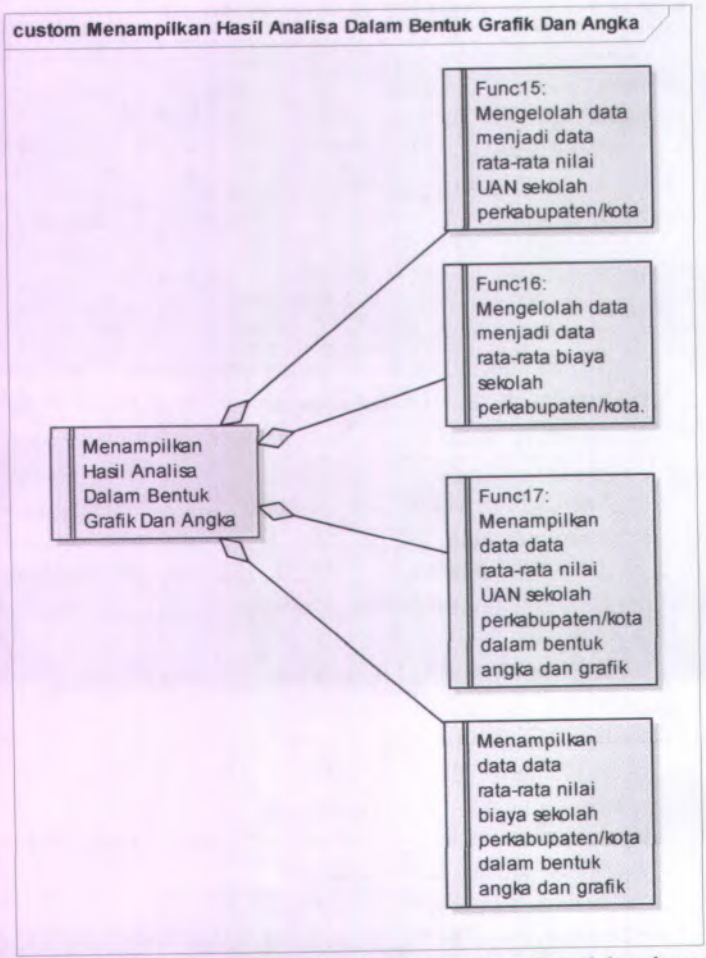
Gambar 4. 8 Kebutuhan fungsional menyimpan data

Kebutuhan fungsional menyimpan data diperoleh pada tahapan proses bisnis kedatangan kuisioner, karena kedatangan kuisioner harus disimpat dulu baru dilakukan pengolahan.

Kebutuhan fungsional (gambar 4.10) menampilkan hasil analisa dalam bentuk grafik dan angka didapat pada tahapan proses bisnis merata-rata UAN dan biaya sekolah dan menampilkannya dalam bentuk laporan.

Ada pun untuk kebutuhan non-fungsional adalah sebagai berikut :

1. Untuk aplikasi di server pendataan online harus mampu bekerja selama 24 jam sehari, 7 hari seminggu, dan saat tertentu dimatikan untuk perawatan.
2. Untuk aplikasi server pendataan di sekolah harus bisa beroperasi selama hari kerja.
3. Aplikasi server pandataan di sekolah harus mampu menangani minimum 4 klien yang melakukan koneksi.
4. Sistem harus mampu menjamin kerahasian data untuk informasi yang dikirim baik melalui internet maupun jaringan lokal di sekolah.



Gambar 4. 9 Kebutuhan fungsional menampilkan data dalam bentuk grafik dan angka

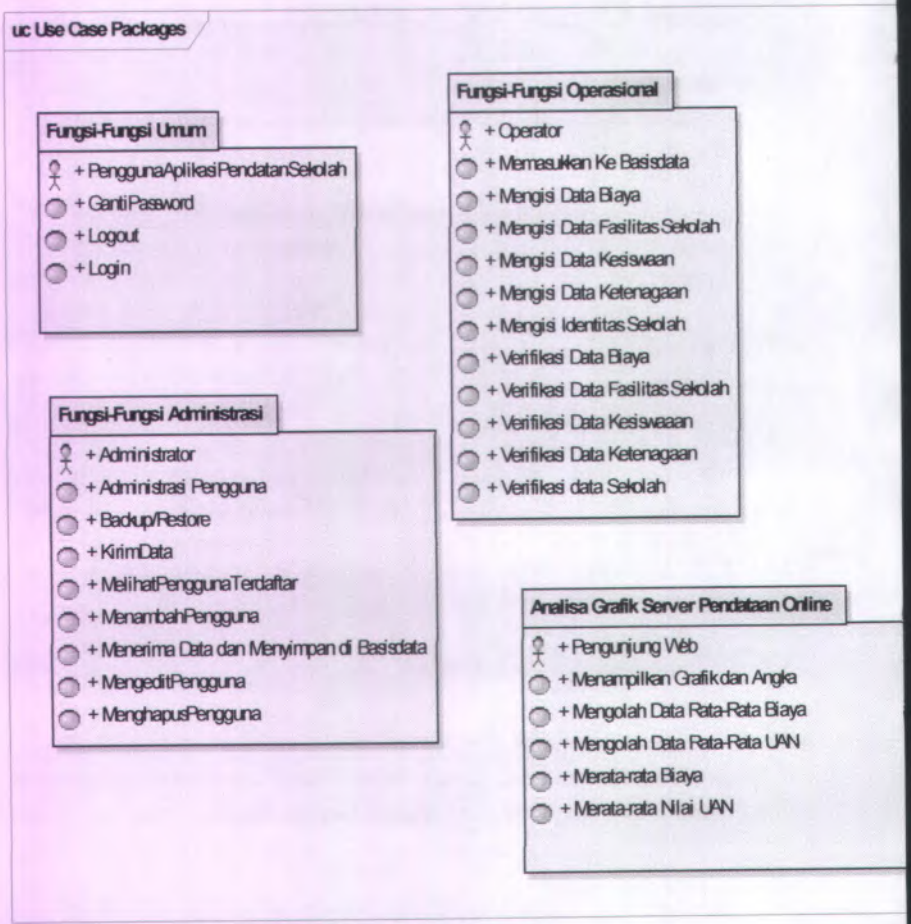
5. Pengaksesan terhadap fungsi yang ada pada aplikasi server pendataan di sekolah harus melalui mekanisme otorisasi.
6. Data yang dikirimkan tidak boleh diubah tanpa ijin dari pihak yang memiliki wewenang.

7. Perubahan data yang dilakukan tanpa ijin dapat mengakibatkan tidak konsistennya informasi yang diberikan oleh sistem.
8. Menggunakan sistem backup dan menyediakan *disaster recovery center* (DRC) yang dilengkapi dengan panduan untuk melakukan pemulihan (*disaster recovery plan*).
9. Sistem memiliki menu login untuk verifikasi pengguna.
10. Setiap pengguna harus memilki username dan password.
11. Aplikasi server pendataan online harus dapat menampilkan data dan grafik dari analisa kurang dari 10 detik.
12. Aplikasi server pendataan online harus dapat menerima inputan dari minimal 30 % dari seluruh aplikasi server pendataan di sekolah yang ada di Jawa Timur dalam waktu yang relative bersamaan.
13. Aplikasi server pendataan di sekolah harus mampu mengirim data dalam waktu kurang dari 30 menit dengan kondisi jaringan internet tidak terputus.
14. Aplikasi server pendataan di sekolah harus mampu menerima inputan dari minimal 4 klien yang akan digunakan.

4.2 Hasil Analisa Kebutuhan

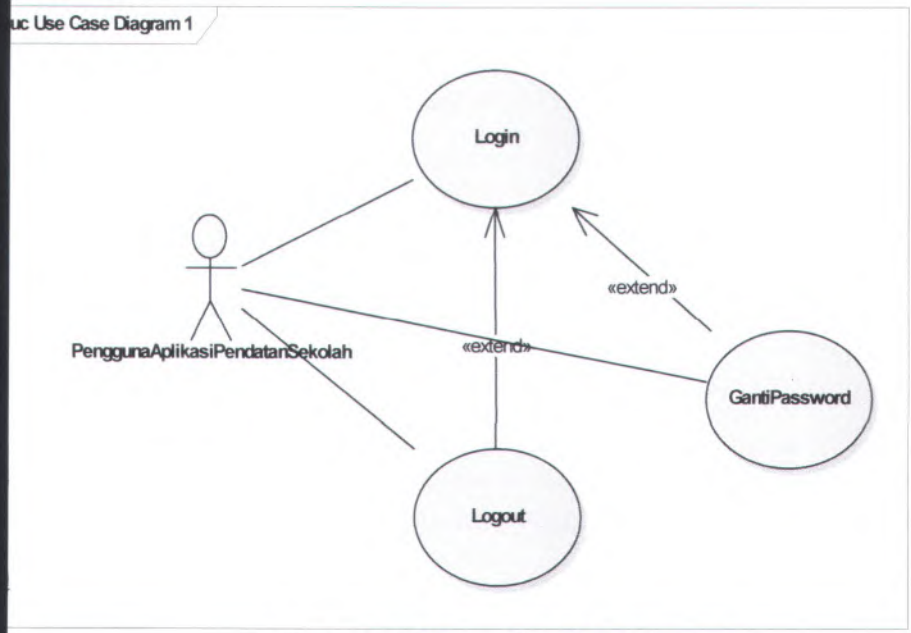
4.2.1 Hasil Memodelkan *Use case*

Dari kebutuhan fungsional dan non-fungsional maka dapat dibuatkan diagram *use case* sebagai berikut.



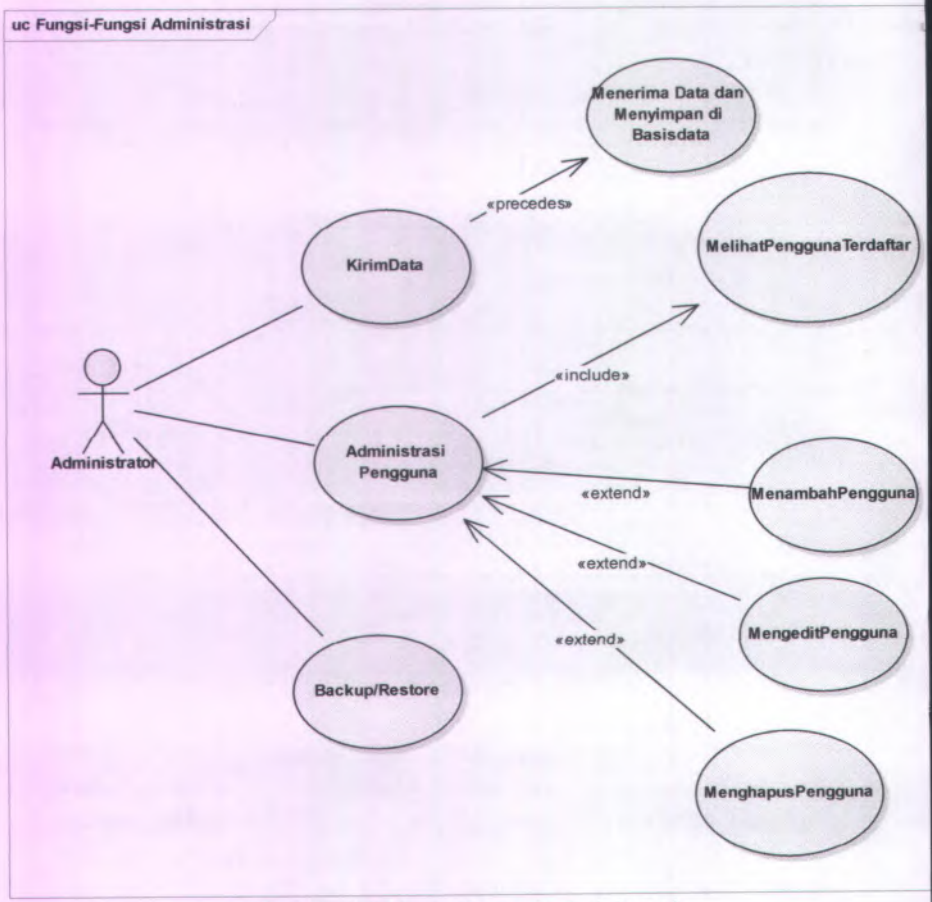
Gambar 2 1 Paket-Paket *Use case*

Untuk memudahkan analisa maka dikelompokanya *use case-use case* tersebut dalam berbagai paket, diantaranya paket fungsi umum, fungsi operasional, fungsi administrasi, dan analisa grafik server pendataan Online. Dari paket-paket *use case* yang telah digambarkan maka dapat dijabarkan untuk setiap paketnya sebagai berikut.



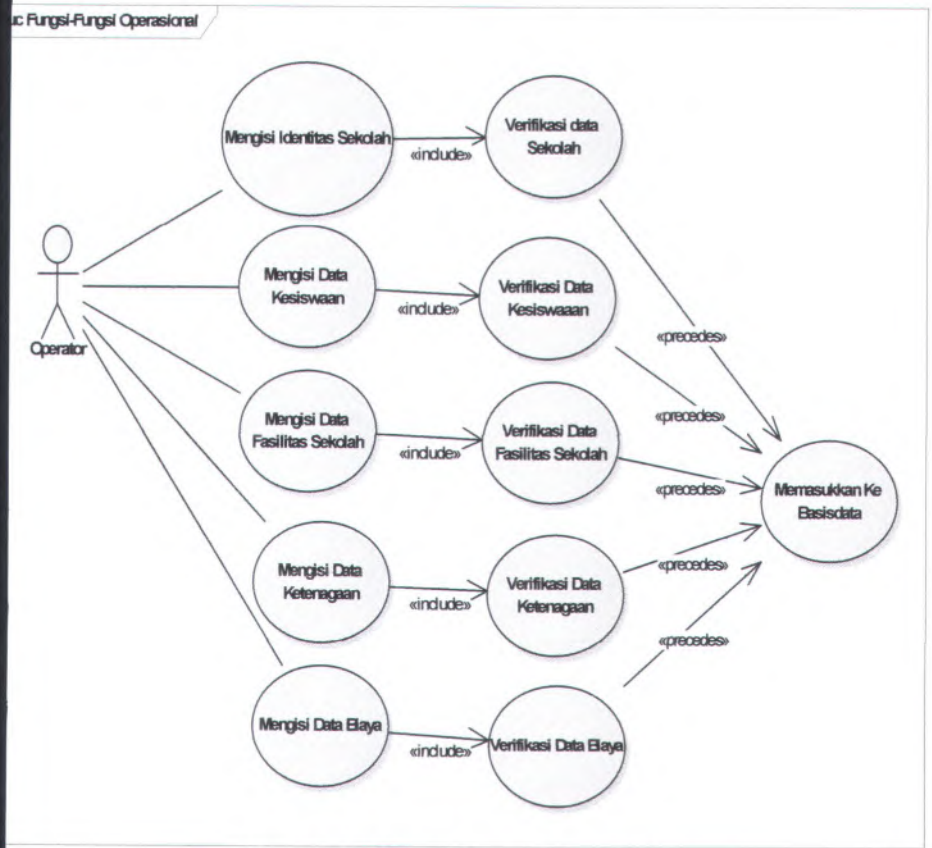
Gambar 2 2 *Use case* fungsi-fungsi umum

Yang dimaksud dari fungsi-fungsi umum adalah kumpulan use case yang dapat dilakukan oleh aktor pengguna aplikasi pendataan sekolah secara umum. Paket-paket use case untuk fungsi-fungsi umum terdiri atas tiga macam use case yaitu login, ganti password, dan logout.



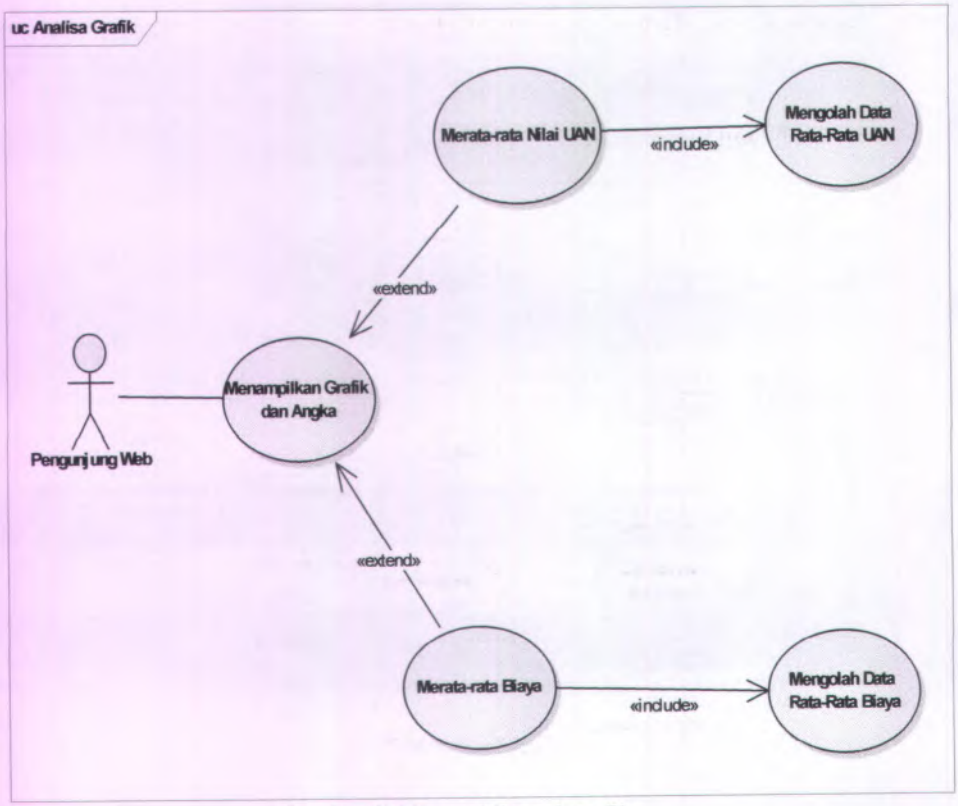
Gambar 2 3 Fungsi-Fungsi administrasi

Fungsi-fungsi administrasi merupakan fungsi-fungsi dimana administrator dapat melakukan administrasi terhadap perangkat lunak, disamping melakukan administrasi pengguna juga melakukan *backup/restore* dan mengirimkan data. Untuk menerima data dan menyimpan di basisdata dilakukan setelah use case kirim data telah selesai dilaksanakan.



Gambar 2 4 Fungsi-Fungsi Opersional

Berdasar dari kebutuhan fungsional maka dibuatlah *use case-use case* yang merepresentasikan kebutuhan fungsional entri data, dan use case-use case tersebut dikelompokan pada paket fungsi-fungsi operasional. Dalam paket fungsi-fungsi operasional verifikasi data dilakukan dalam *use case* pengisian, dan memsukan ke basis data dilakukan setelah use case verifikasi data telah selesai dilaksanakan.



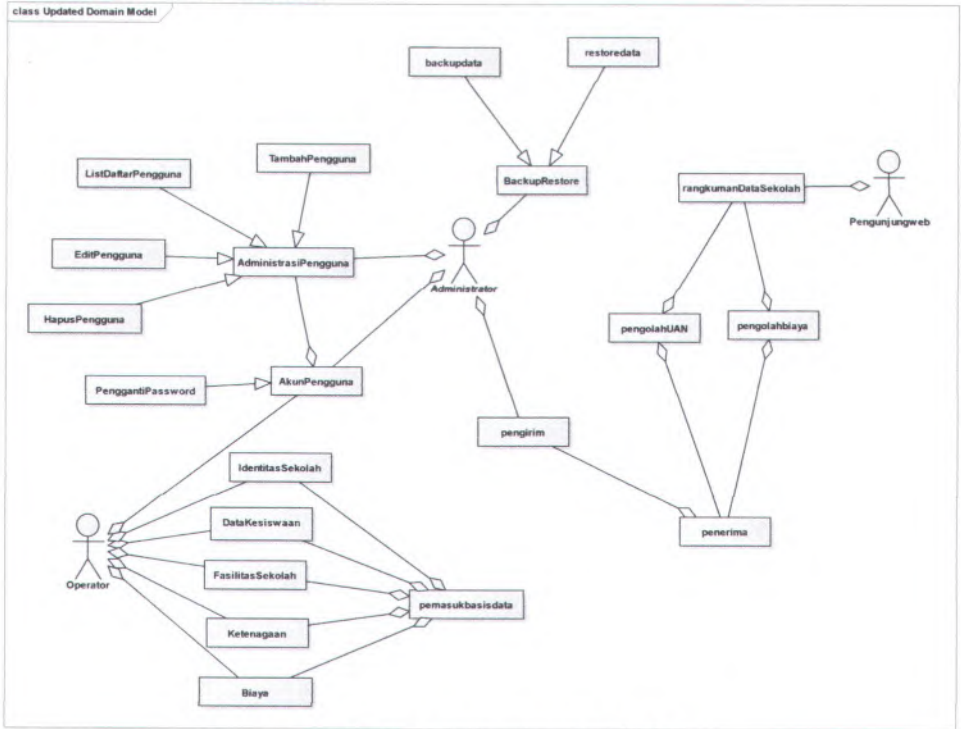
Gambar 2 5 Fungsi-fungsi grafik

Berdasar kebutuhan fungsional maka dibuatlah paket *use case* fungsi-fungsi grafik, yang mana *use case-use case* di dalammnya *use case* yang bertujuan untuk menyajikan laporan ke kepala dinas.

Untuk naratif *use case* dapat dilihat pada lampiran analisa model.

4.2.2 Hasil Memodelkan Domain

Setelah mendapatkan kebutuhan fungsional maka tahap memodelkan domain dapat dilakukan. Berikut diagram hasil dari memodelkan domain.



Gambar 4 1 Domain Model Yang Telah Dibuat

Domain model diatas merupakan purwarupa dari diagram *Class* yang akan dibuat. Dalam domain model yang dibuat atribut dari diagram *Class* tidak perlu ditampilkan karena hal ini dapat membuat desain berikutnya menjadi rancu dan sulit diimplementasikan.

4.3 Hasil Analisa/Desain Pendahuluan

Pada tahapan ini menghasilkan sebuah diagram analisa *robustness*, dan juga melakukan update terhadap domain model untuk lebih jelas, lihat pada lampiran analisa model.

4.4 Hasil Desain Lebih Detil

Pada tahapan ini akan dihasilkan diagram *sequence* dan diagram *Class*. Diagram *sequence* yang merupakan kelanjutan dari analisa *robusteness* telah memiliki beberapa oprasi penting yang sesuai dengan fungsi-fungsinya, untuk diagram *Class* tidak bisa langsung digenerate, memerlukan penyesuaian agar bisa diimplementasikan. Untuk hasilnya dapat diliha lebih detil di lampiran analisa model.

4.5 Hasil Implementasi

Sebagaimana dari batasan dalam pengerjaan tugas akhir, maka bagian ini tidak ada hasilnya, dikarenakan batasan tugas hanya pada tahapan desain.

BAB 5
PENUTUP

5.1 Simpulan

Kesimpulan yang dapat diambil dari pengerjaan tugas akhir ini adalah sebagai berikut:

1. Sistem informasi pendataan sekolah merupakan sistem yang mampu melakukan pendataan terhadap sekolah-sekolah yang ada di Jawa Timur. Sistem ini telah dipetakan kebutuhannya dalam SKPL dan rancangannnya di modelkan dalam format UML.
2. *Robustness* Diagram yang ada pada *Iconix Process* sangat membantu dalam pembuatan Sequence Diagram. Dengan menggunakan Proses Iconix tahapan perancangan sistem perangkat lunak menjadi lebih terarah dan menjadi lebih cepat.

5.2 Saran

1. Sistem informasi pendataan sekolah ini merupakan awalan menuju sistem informasi pendidikan yang terpadu diharapkan di masa mendatang akan ada sistem yang mampu mengolah seluruh data pendidikan di tingkat nasional.
2. Tugas akhir kali ini hanya sampai pada tahapan desain diharapkan nantinya bisa diimplementasikan dan mampu menghasilkan desain dan aplikasi yang lebih baik.

DAFTAR PUSTAKA

Dough Rosenberg, Matt Stephens. 2007. **Use case Driven Obeject Modeling with UML** Apress.

James Rumbaugh, Ivar Jacobson, Grady Booch. 2005. **The Unified Modeling Language Reference Manual Second Edition**. Addison-Wesley.

Pusat Data dan Informasi Badan Penelitian dan Pengembangan Depdiknas. 2004. **Mekanisme Pendataan**. Depdiknas.

http://en.wikipedia.org/wiki/ICONIX akses terakhir 11 Januari 2008, Pukul 18.56 wib.

http://www.sparxsystems.com.au/uml-tutorial.html. akses terakhir 11 Januari 2008, Pukul 18.58.

Penulis dilahirkan di Surabaya, awal Oktober tahun 1985, merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Panca Unggul Surabaya, SDN Ploso V Surabaya, SLTP Negeri 9 Surabaya, dan SMAN 6 Surabaya. Pada tahun 2003 mengikuti SPMB dan diterima di Jurusan Sistem Informasi FTIf-ITS dan terdaftar dengan NRP 5203100019. Di Jurusan Sistem Informasi ini Penulis mengambil bidang minat Pengembangan dan Perencanaan Sistem Informasi. Penulis sempat aktif di beberapa organisasi mahasiswa baik di lingkup jurusan maupun di ITS, menjadi Asisten Praktikum Sistem Operasi.