BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada penelitian Darmoyo[1], Darmoyo mengembangkan instrumen untuk melakukan evaluasi diri terhadap kelemahan dalam pelaksanaan penjaminan mutu pendidikan. Pada aplikasi tersebut, aspirasi pada evaluasi diri berasal dari mahasiswa, dosen, tenaga kependidikan, orang tua mahasiswa, serta pihak lain yang berhubungan dengan pelayanan di UNNES. Aspirasi dan solusi yang telah dikirimkan akan disampaikan kepada unit kerja terkait melalui pimpinan Universitas Negeri Semarang. Seluruh aspirasi yang disampaikan akan dikelola oleh Sub Bagian Tata Usaha BUHK UNNES.

Berbeda dengan penelitian sebelumnya yang dilakukan oleh Nasution[2], Nasution mengembangkan aplikasi ini untuk menangani solusi masyarakat menyampaikan aspirasi yang menyeluruh dengan mudah, tidak hanya keluhan, melainkan saran, permintaan hingga pujian yang ditujukan untuk pengembangan pelayanan dan mendapatkan respon dari perusahaan. Pada aplikasi tersebut pengguna bisa mengetahui perkembangan aspirasi yang dikirimkan dengan lima status yaitu *pending, rejected, approved, delivered, dan responded*.

Penelitian lainya dilakukan oleh Dhiratara[3], Dhiratara mengembangkan aplikasi yang bertujuan agar masyarakat menjadi mudah berinteraksi dengan pemerintah serta dapat secara aktif berpartisipasi dalam mengawasi pembangunan dan pelayanan publik. Setiap laporan akan diteruskan ke instansi terkait secara digital, cepat, dan tepat serta dapat dipantau tindak-lanjutnya secara interaktif. Pada aplikasi tersebut, masyarakat dapat menyampaikan aspirasi dan pengaduannya melalui berbagai kanal, antara lain SMS 1708, situs www.lapor.ukp.go.id, *mobile appication android* dan blackberry, serta media sosial twitter @LAPOR1708 dan facebook LAPOR!.

Berdasarkan studi pustaka diatas yang membedakan penelitian yang diajukan dengan penelitian sebelumnya adalah penulis akan mengembangkan sebuah aplikasi suara mahasiswa berbasis *android* sebagai media atau wadah saran dan masalah akademik mahasiswa di Perguruan Tinggi Politeknik Negeri Cilacap. Aplikasi yang dikembangkan difokuskan pada kegiatan akademik dan ditujukan untuk dosen, mahasiswa dan BAAK. Mahasiswa dapat mengirimkan saran dan

masalah akademik kepada dosen bersangkutan dengan mudah kapanpun dan dimanapun. Sedangkan pada penelitian sebelumya untuk penelitian Sistem Informasi Aspirasi Publik masih berbasis web yang ditujukan untuk masyarakat dan mahasiswa UNNES dan untuk penelitian Bulp dan LAPOR berbasis android namun aplikasi tersebut untuk kalangan masyarakat luas.

2.2 Landasan Teori

2.2.1 Basis Data

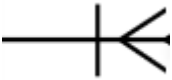
Basis Data atau *Database* adalah himpunan kelompok data yang saling berkaitan. Data tersebut diorganisasikan sedemikian rupa agar tidak terjadi duplikasi data yang tidak perlu, sehingga dapat diolah atau dieksplorasi secara cepat dan mudah untuk menghasilkan informasi[4].

ERD (*Entity Relationship Diagram*)

Pengertian ERD adalah suatu komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut yang mempresentasikan seluruh fakta dari dunia nyata yang ditinjau Entity Relationship Diagram menggambarkan data dan hubungan antar data secara global dengan menggunakan Entity Relationship Diagram[5]. Simbol-simbol yang digunakan dalam ERD dijelaskan pada Tabel 2.1 Simbol ERD (Entity Relationship Diagram) sebagai berikut :

Tabel 2.1 Simbol ERD (Entity Relationship Diagram).

No	Keterangan
1. Relasi/Hubungan 	Hubungan yang terjadi antara 1 entitas atau lebih yang tidak mempunyai fisik tetapi hanya sebagai konseptual. Dan untuk mengetahui jenis hubungan yang ada antara 2 *file*.
2. Atribut 	Atribut ialah karakteristik dari entitas atau relasi yang menyediakan penjelasan detil tentang entitas atau relasi tersebut. Berfungsi untuk memperjelas atribut yang dimiliki oleh sebuah entitas. Atribut memiliki bentuk lingkaran lebih tepatnya elips.

3. Alur	Alur memiliki fungsi untuk menghubungkan atribut dengan entitas dan entitas dengan relasi. Dan berbentuk garis.
4. *One to One* 	*One to One*, digunakan untuk menghubungkan antar entitas dengan hubungan satu ke satu
5. *One to Many* 	*One to Many* atau *Many to One*, digunakan untuk menghubungkan antar entitas dengan hubungan satu ke banyak atau sebaliknya
6. Entitas (***Entity***) 	Entitas ialah suatu objek yang dapat dibedakan dengan objek lainnya. Entitas berfungsi untuk memberikan identitas pada entitas yang memiliki label dan nama. Entitas memiliki bentuk persegi panjang.

Relasi antar table sebagai berikut:

a) Hubungan One-to-One, masing-masing tabel hanya terdapat satu data yang saling berhubungan.
b) Hubungan One-to-Many, berelasi dengan banyak record pada tabel yang lain.
c) Hubungan Many-to-Many, banyak record pada sebuah table berhubungan dengan banyak record pada tabel yang lain.

2.2.2 Rekayasa Perangkat Lunak

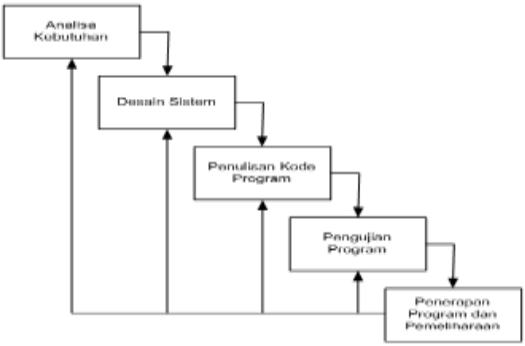
Perangkat lunak adalah seluruh perintah yang digunakan untuk memproses informasi. Perangkat lunak dapat berupa program atau prosedur. Program adalah kumpulan perintah yang dimengerti oleh komputer sedangkan prosedur adalah perintah yang dibutuhkan oleh pengguna dalam memproses informasi [6].

Rekayasa Perangkat Lunak sendiri adalah suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak, mulai dari tahap awal yaitu analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, disain, pengkodean, pengujian sampai pemeliharaan sistem setelah digunakan. Dari pengertian ini jelaslah

bahwa RPL tidak hanya berhubungan dengan cara pembuatan program komputer. Pernyataan "semua aspek produksi" pada pengertian di atas, mempunyai arti semua hal yang berhubungan dengan proses produksi seperti manajemen proyek, penentuan personil, anggaran biaya, metode, jadwal, kualitas sampai dengan pelatihan pengguna merupakan bagian dari RPL [6].

A. Metode Pengembangan Sistem

Metode penelitian yang digunakan adalah metode *waterfall*. *Waterfall* atau air terjun adalah model yang dikembangkan untuk pengembangan perangkat lunak, dan membuat perangkat lunak. Metode ini berkembang secara sistematis dari satu tahap ke tahap lain dalam mode seperti air terjun. Metode ini mengusulkan sebuah pendekatan kepada pengembangan software yang sistematik dan sekuensial yang mulai dari tingkat kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Model waterfall terdiri dari 5 tahapan, yaitu Analisa Kebutuhan, Desain Sistem, Penulisan Kode Program, Pengujian Program, Penerapan Program dan Pemeliharaan. Model waterfall dapat dilihat pada Gambar 2.1.



Gambar 2.1 Pemodelan *Waterfall* menurut Sommerville

Penjelasan tahapan-tahapan dari metode *waterfall* [7] :

1. Analisa Kebutuhan (*Analysis*)
   Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data pada tahap ini bisa melakukan sebuah penelitian, wawancara atau *study* literatur. Seorang sistem analisis akan menggali

informasi sebanyak banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user reqruitment* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen inilah yang akan menjadi acuan sistem analisis untuk menterjemahkan kedalam bahasa pemrograman.

2. Desain Sistem (*Design*)
   Proses desain sistem akan menerjemahkan syarat kebutuhan kesebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi *interface*, dan detail (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *software reqruitment*. Dokumen inilah yang akan digunakan *programmer* untuk melakukan aktifitas pembuatan sistemnya.
3. Penulisan Kode Program (*Coding*)
   Coding merupakan penerjemah *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programme*r yang akan menterjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah kita buat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.
4. Pengujian Program (*Testing*)
   Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Setelah melakukan analisa, *design* dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*.
5. Penerapan Program dan Pemeliharaan (*Maintenance*)
   Perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau dengan sistem operasi yang baru) atau karena pelanggan membutuhkan perkembangan fungsional.

B. Metode Pengujian Sistem

Metode pengujian sistem yang digunakan adalah dengan metode *black box testing.* Metode *black box testing* merupakan pengujian yang berfokus pada spesifikasi fungsional dari perangkat lunak, penguji *(terster)* dapat mendefinisikan kumpulan kondisi dan melakukan pengetesan pada spesifikasi fungsional program [8].

Ciri-ciri *black box testing* [8]:

1. *Black box testing* berfokus pada kebutuhan fungsional pada *software*, berdasarkan pada spesifikasi kebutuhan dari *software.*
2. *Black box testing* bukan teknik alternatif daripada *white box testing.* Lebih dari itu, *black box testing* merupakan pendekatan pelengkap dalam mencakup *error* dengan kelas yang berbeda dari metode *white box testing.*
3. *Black box testing* melakukan pengujian tanpa pengetahuan detil struktur internal dari sistem atau komponen yang dites.

Kategori *error* yang akan diketahui melalui *black box testing* [8]:

1) Fungsi yang hilang atau tak benar.
2) *Error* dari antarmuka.
3) *Error* dari struktur data atau akses *eksternal database*.
4) *Error* dari kinerja atau tingkah laku.
5) *Error* dari inisialisasi dan terminasi.

2.2.3 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah strategi pembangunan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya [9]. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis.

Metodologi berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan tersruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu [9]**.**

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

1) Meningkatkan produktifitas
   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai diulang kembali untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2) Kecepatan pengembangan
   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.
3) Kemudahan pemeliharaan
   Karena dengan model objek, pola-pola yang cenderung tetapdan stabil dapat dipisahkan dan pola-pola mungkin sering berubahubah.
4) Adanya konsistensi
   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
5) Meningkatkan kualitas perangkat lunak
   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek [9]:

1) Kelas (*class*)
   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.
2) Objek (object)
   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, suatu organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.
3) Metode (*method*)

Operasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

4) Atribut (*attribute*)
   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya. Atribut sebaliknya bersifat prifat untuk menjaga konsep enkapsulasi.
5) Abstraksi (*abstraktion*)
   Prinsip untuk mempresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6) Enkapsulasi (*encapsulation*)
   Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.
7) Pewarisan (*inheritance*)
   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.
8) Antarmuka (*interface*)
   Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.
9) Reusability
   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.
10) Generalisasi dan Spesifikasi
    Menunjukan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.
11) Komunikasi Antar objek
    Komunikasi antar objek dilakukan lewat pesan (*message*) yang dikirimkan dan satu objek ke objek lainnya.

12) Polimorfisme (*polymorphism*)
Kemampuan suatu objek yang digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

13) Package
Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokan kelas-kelas sehingga bernama sama disimpan dalam *package* yang berbeda.

Pada pemrograman berorientasi objek, UML digunakan untuk pemodelan sistem. UML adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, *artifact* tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya [9].

a) Use Case Diagram

Use case adalah abstraksi dari interaksi antara sistem dan aktor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara user sebuah system dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan use case diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan client [9].

Tabel 2.2 Simbol dan Fungsi *Use Case Diagram*.

No	Simbol	Fungsi
1.	 *Actor*	Segala sesuatu yang berinteraksi dengan sistem aplikasi komputer. Jadi actor ini bisa berupa orang, perangkat keras atau mungkin juga obyek lain dalam sistem yang sama.
2.	 *Include*	Menspesifikasikan bahwa prilaku *use case* merupakan bagian dari *use case* lain.

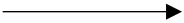
3.	*Association*	Menggambarkan navigasi antar *class*, berupa banyak obyek lain yang berhubungan dengan satu obyek, dan apakah suatu *class* menjadi bagian dari *class* lainnya.
4.	*System Boundary*	*System Boundary* yaitu batasan sebuah sistem.
5.	*Use Case*	*Use case* menjelaskan urutan kegiatan yang dilakukan aktor dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, use case hanya menjelaskan apa yang dilakukan oleh aktor dan sistem, bukan bagaimana aktor dan sistem melakukan kegiatan.

b) Sequence Diagram

S*equence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima oleh objek. Oleh karena itu untuk menggambar diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram *sequence* juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Berikut simbol-simbol yang ada pada diagram *sequence* [9].

Tabel 2.3 Simbol dan Fungsi *Sequence Diagram*.

No	Simbol	Fungsi
1.	*Actor* 	*Actor* yaitu Orang, poses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.

2.	*Lifeline*	*Lifeline* yaitu Menyatakan kehidupan suatu objek.
3.	*Object* 	*Object* yaitu Menyatakan objek yang berinteraksi pesan.
4.	Waktu aktif 	Waktu aktif yaitu Menyatakan objek dalam keadaan aktif dan berinteraksi pesan.
5.	Pesan tipe *create* <<create>> 	Suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
6.	Pesan tipe *call* 1 : nama_method()	Suatu objek memanggil operasi atau metode yang ada pada objek lain atau dirinya sendiri.
7.	Pesan tipe *send* 1 : masukan 	Suatu objek mengirimkan data atau masukan atau informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8.	Pesan tipe *return* 1 : keluaran 	Suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke ojek tertentu.

2.2.4 Flowchart

Flowchart merupakan gambar atau bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung. Flowchart ini merupakan langkah awal pembuatan program. Dengan adanya flowchart urutan poses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah. Setelah flowchart selesai disusun, selanjutnya

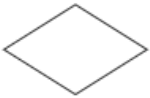
programmer menerjemahkannya ke bentuk program dengan bahasa pemrograman.
Simbol yang di pakai dalam flowchart dibagi menjadi 3 kelompok :

1) *Flow direction symbols*
   Digunakan untuk menghubungkan simbol satu dengan yang lain (*connecting line*).
2) *Processing symbols*
   Menunjukan jenis operasi pengolahan dalam suatu proses atau prosedur.
3) *Input / Output symbols*
   Menunjukkan jenis peralatan yang digunakan sebagai media input atau output.

Simbol-simbol Flowchart

Flowchart disusun dengan simbol-simbol. Simbol ini dipakai sebagai alat bantu menggambarkan proses di dalam program. Simbol-simbol yang dipakai antara lain.

Tabel 2.4 Simbol dan Fungsi Flowchart.

No	Simbol	Nama dan Fungsi
1.		*Flow Direction Symbol* yaitu simbol yang digunakan untuk menghubungkan antar simbol
2.		*Process Symbol* yaitu simbol yang menunjukkan pengolahan yang dilakukan oleh komputer.
3.		*Manual Operation Symbol* yaitu simbol yang menunjukkan pengolahan yang tidak dilakukan oleh komputer.
4.		*Decision Symbol* yaitu simbol yang menunjukan suatu kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban ya tau tidak.

5.		*Terminal Symbol* yaitu symbol yang menyatakan permulaan atau akhir suatu program.
6.		*Input atau Output Symbol* yaitu simbol yang menyatukan proses input atau output tanpa tergantung jenis peralatannya.

(~~*Halaman ini sengaja dikosongkan*~~)