BAB II
LANDASAN TEORI

2.1. Konsep Dasar Sistem Informasi

Sebuah sistem informasi merupakan kumpulan dari perangkat keras, lunak serta manusia yang akan mengolah dan menggunakan perangkat keras dan lunak tersebut. Informasi merupakan hal yang sangat penting dengan adanya informasi tersebut dapat diketahui kemajuan dan kegagalan proses pelaksanaan. Sistem yang kurang informasi menunjukan bahwa sistem tersebut rapuh. Data merupakan informasi yang diolah supaya berguna bagi yang menerimanya. Definisi sistem secara umum yaitu sekumpulan proses dan seperangkat elemen yang digabung serta dihimpun secara bersama serta saling berintegrasi untuk mencapai suatu tujuan dari organisasi (Warman dan Saputra, 2012).

2.2. Komponen Sistem Informasi

Sistem informasi terdiri dari komponen-kompponen yang disebut dengan istilah blok bangunan (*Building Block*), dimana masing-masing blok ini saling berintegrasi satu sama lainya membentuk satu kesatuan untuk mencapai tujuannya. Adapun blok-blok tersebut adalah sebagai berikut (Anisya, 2013):

1. Blok Masukan

   Meliputi metode-metode dan media untuk menangkap data yang akan dimasukan, dapat berupa dokumen-dokumen dasar.
2. Blok Model

   Terdiri dari kombinasi prosedur, logika dan model matematik yang berfungsi memanipulasi data untuk keluaran tertentu.
3. Blok Keluaran

   Berupa keluaran dokumen dan informasi yang berkualitas.
4. Blok Teknologi

   Untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran serta membantu pengendalian dari sistem secara keseluruhan.

5. Blok Basisdata

   Merupakan kumpulan data yang saling berhubungan satu dengan yang lainnya, tersimpan didalam perangkat keras komputer dan perangkat lunak untuk memanipulasi.

6. Blok Kendali

   Meliputi masalah pengendalian yang berfungsi mencegah dan menangani kesalahan atau kegagalan sistem.

2.3. Metodologi Berorientasi Objek (*Object Oriented Design*)

Menurut Metodologi berorientasi objek dapat didefenisikan sebagai suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya (Rachman dkk, 2012).

Sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkkus (dienkapsulasi) menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat dan komponen lainnya serta dapat berinteraksi satu sama lainnya.

2.4. Teknik pemodelan dalam OOAD

1. Model Objek
   a. Model objek Menggambarkan struktur statis dari suatu objek dalam sistem dan relasinya.
   b. Model objek berisi diagram objek. Diagram objek adalah *graph* dimana *node*-nya adalah kelas yang mempunyai relasi antar kelas.
2. Model Dinamik
   a. Model dinamik menggambarkan aspek dari sistem yang berubah setiap saat.
   b. Model dinamik dipergunakan untuk menyatakan aspek kontrol dari sistem.

c. Model dinamik berisi *state* diagram. *State diagram* adalah *graph* dimana *node*-nya adalah *state* dan *arc* adalah transisi antara *state* yang disebabkan oleh *event*.

3. Model Fungsional
   a. Model fungsional menggambarkan transformasi nilai data di dalam sistem.
   b. Model fungsional berisi *data flow diagram* (DFD). DFD adalah suatu *graph* dimana *node*-nya menyatakan proses dan *arc*-nya adalah aliran data.

2.5. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau DBMS yang *multithread, multi-user,* dengan sekitar 6 juta instalasi diseluruh dunia. *MySQL AB* membuat *MySQL* tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. *Relational Database Management System* (RDBMS) (Syaifudin dkk, 2013).

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan *MySQL,* namun tidak boleh dijadikan produk turunan yang bersifat komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu *Structured Query Language* (SQL).

2.6. *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah bahasa standar yang digunakan untuk menulis *blueprint* perangkat lunak. UML dapat digunakan untuk memvisualisasi, menspesifikasikan, membangun, dan mendokumentasikan artifak dari sistem perangkat lunak. UML terdiri atas tiga *building block*, yaitu (Alim dkk, 2013):

1. *Things*

   Things adalah building *block* berbasis objek yang utama dari UML. *Things* terdiri atas 4 macam, yaitu:

   a. *Structural Things*

      Structural things adalah bagian model statis yang merepresentasikan elemen konseptual atau fisik. Jenis-jenis *structural things* adalah *class, interface, collaboration,* dan *use case*.

   b. *Behavorial Things*

      Behavorial things adalah bagian dinamis dari model UML yang merepresentasikan *behavior*. Jenis *behavorial things* adalah *interaction, state machine,* dan *activity*.

   c. *Grouping Things*

      Grouping things adalah bagian dari model UML yang berfungsi untuk mengelompokkan elemen. Satu-satunya jenis *grouping things* adalah *package*.

   d. *Annotational Things*

      Annotational things adalah bagian penjelas dari model UML. Jenis *annotational things* adalah *note* yang digunakan untuk memberikan komentar.

2. *Relationship*

   Relationship merupakan *building block* UML yang berfungsi sebagai penghubung antar *things*. Jenis-jenis *relationship* antara lain d*ependency, Association, Generalization, Realization*.

3. *Diagram*

   Diagram adalah presentasi grafis yang merupakan *kombinasi* antara *things* dan *relationship*. *Diagram* dibuat untuk memvisualisasikan sistem dari sejumlah perspektif yang berbeda, sehingga *diagram* merupakan proyeksi terhadap sistem. UML mempunyai tiga belas *diagram*.

2.6.1. *Use Case Diagram*

Use case model adalah teknik pemodelan untuk mendapatkan *functional requirement* dari sebuah sistem, menggambarkan interaksi antara pengguna dan sistem, menjelaskan secara naratif bagaimana sistem akan digunakan, menggunakan skenario untuk menjelaskan setiap aktivitas yang mungkin terjadi. Ada beberapa bagian didalam *use case* model dapat dilihat pada Tabel 2.1 (Edgar dan Johan, 2013).

Tabel 2.1 Simbol *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

Tabel 2.1 Simbol *Use Case Diagram* (lanjutan)

NO	GAMBAR	NAMA	KETERANGAN
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

2.6.2. *Activity Diagram*

Activity Diagram merupakan teknik untuk menjelaskan *business process*, menjelaskan teks *use case* dalam notasi grafis dengan menggunakan notasi yang mirip *flow chart*, meskipun terdapat sedikit perbedaan notasi. Simbol *Activity Diagram* dapat dilihat Tabel 2.2 di bawah ini (Edgar dan Johan, 2013).

Tabel 2.2 Simbol *Activty Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2		*Action*	*State* dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.

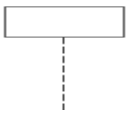
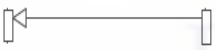
Tabel 2.2 Simbol *Activty Diagram* (lanjutan)

NO	GAMBAR	NAMA	KETERANGAN
4		*Activity Final Node*	Bagaimana objek dibentuk dan dihancurkan.
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.

2.6.3. *Sequence Diagram*

Sequence Diagram menjelaskan interaksi obyek-obyek yang saling berkolaborasi (berhubungan), mirip dengan *activity diagram* yaitu menggambarkan alur kejadian sebuah aktivitas tetapi lebih detil dalam menggambarkan aliran data termasuk data atau *behaviour* yang dikirimkan atau diterima namun kurang mampu menjelaskan detil dari sebuah algoritma. Simbol-simbol *Sequence Diagram* dapat dilihat pada Tabel 2.3 (Edgar dan Johan, 2013).

Tabel 2.3 Simbol *Sequence Diagram*

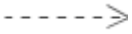
NO	GAMBAR	NAMA	KETERANGAN
1.		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2.		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
3.		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.

2.6.4. *Class Diagram*

Class diagram merupakan diagram paling umum yang dijumpai dalam pemodelan berbasis UML. Dalam *class diagram* terdapat *class* dan *interface* beserta atribut-atribut dan operasinya, relasi yang terjadi antar objek, *constraint* terhadap objek-objek yang saling berhubungan dan *inheritance* untuk organisasi *class* yang lebih baik. *Class diagram* juga terdapat *static view* dari elemen

pembangun sistem. Pada intinya *Class diagram* mampu membantu proses pembuatan sistem dengan memanfaatkan konsep *forward* ataupun *reverse engineering*. Simbol–simbol *class diagram* dapat dilihat pada Tabel 2.4 (Edgar dan Johan, 2013).

Tabel 2.4 Simbol *Class Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.

2.7. *One Stop Service*

Semakin majunya era globalisasi membuat masyarakat mengajukan tuntutan untuk memberikan pelayanan publik yang berorientasi kepada masyarakat dan tanggap terhadap kebutuhan masyarakat. Hal ini menyebabkan timbulnya pemikiran baru untuk memberikan pelayanan publik yang didasarkan pada sudut pandang pelanggan baik bagi masyarakat atau kalangan dunia usaha.

Pelayanan menjadi esensi yang sangat penting untuk kita telusuri perkembangannya mengingat dengan berlakunya Undang-Undang Nomor 32 Tahun 2004 tentang Pemerintah Daerah mengakibatkan interaksi antara aparat Daerah dan masyarakat menjadi lebih intens. Hal ini ditambah dengan semakin kuatnya tuntutan demokratisasi dan pengakuan akan hak-hak asasi manusia sehingga melahirkan tuntutan baru terhadap manajemen pelayanan yang berkualitas. Dapat diakui bahwa pelayanan yang diberikan oleh pemerintah kepada masyarakat saat ini terus mengalami pembaruan baik dari sisi paradigma maupun format pelayanan. Berbagai inovasi dan kreativitas dibangun dalam rangka memberikan pelayanan yang terbaik yakni melalui pembentukan pelayanan terpadu satu pintu (*one stop service*) (Khalimah dkk, 2013).

One Stop Government (*One Stop Service*) merupakan konsep baru dalam penelitian dan reformasi pelayanan publik. Adapun yang dimaksud dengan istilah *One Stop Service* atau *One Stop Government* adalah pengintegrasian pelayanan publik dari sudut pandang masyarakat atau pelanggan. Melalui model pelayanan publik ini, semua urusan masyarakat atau pelanggan dapat dipenuhi pada satu tempat, dimana pelanggan dapat melakukan kontak langsung, baik secara tatap muka maupun menggunakan media seperti telepon, *fax*, internet dan media lainnya. Sistem pelayanan publik yang terintegrasi menjanjikan pelayanan yang mulus dari berbagai organisasi pemerintah, menciptakan efisiensi dan pengalaman pelayanan yang lebih baik bagi penyedia layanan serta pengguna layanan itu sendiri. Sedangkan di Indonesia, istilah *One Stop Service* lebih dikenal dengan model Pelayanan Terpadu Satu Pintu (PTSP) (Kusumawandani, 2012).

2.8. *Android*

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *Linux* yang mencakup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, *Google Inc*. membeli *Android Inc*. yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari

34 perusahaan piranti keras, piranti lunak, dan telekomunikasi termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile,* dan *Nvidia* (Murtiwiyati dan Lauren, 2013).

Secara garis besar, arsitektur *Android* dapat dijelaskan sebagai berikut:

1. *Applications* dan *Widgets*

   Application dan *Widgets* ini adalah *layer* dimana berhubungan dengan aplikasi saja, dimana biasanya *download* aplikasi dijalankan kemudian dilakukan instalasi dan jalankan aplikasi tersebut.

2. *Applications Frameworks*

   Applications Frameworks ini adalah *layer* dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi *Android*, karena pada *layer* inilah aplikasi dapat dirancanng dan dibuat seperti *contact-providers* yang berupa sms dan panggilan telepon.

3. *Libraries*

   Libraries ini adalah layer dimana fitur-fitur *Android* berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas *kernel*, *Layer* ini meliputi berbagai *library C/C++* inti seperti *Libc* dan *SSL*.

4. *Android Run Time*

   Layer yang membuat aplikasi *Android* dapat dijalankan dimana dalam prosesnya menggunakan implementasi *Linux*.

5. *Linux Kernel*

   Linux Kernel adalah *layer* dimana inti dari sistem operasi dari *Android* itu berada. Berisi *file-file* sistem yang mengatur sistem *processing, memory, resource, drivers,* dan sistem-sistem operasi *Android* lainnya. *Linux Kernel* yang digunakan *Android* adalah *Linux kernel release* 2.6.

2.9. *Eclipse*

Eclipse adalah sebuah *Integrated Development Environment* (IDE) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua (*platform-independent*) (Sede dkk, 2015).

Berikut ini adalah sifat dari *Eclipse*:

1. *Multi-platform*

   Target sistem operasi *Eclipse* adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX* dan *Mac OS X*.

2. *Multi-languange*

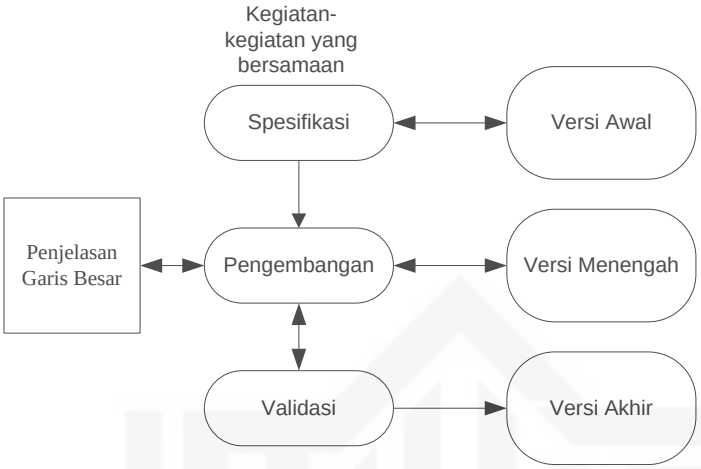
   Eclipse dikembangkan dengan bahasa pemrograman *Java*. Akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti *C/C++, Cobol, Phyton, Perl, PHP,* dan lain sebagainya.

3. *Multi-role*

   Selain sebagai IDE intuk pengembangan aplikasi, *Eclipse* bisa digunakan untuk aktifitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan *web* dan lain sebagainya.

2.10. Metode Pengembangan Evolusioner

Pengembangan evolusioner berdasarkan pada ide untuk mengembangkan implementasi awal, memperlihatkan kepada *user* untuk dikomentari, dan memperbaikinya versi demi versi sampai sistem yang memenuhi persyaratan diperoleh. Tidak ada kegiatan spesifikasi, pengembangan, dan validasi yang terpisah. Akan tetapi kegiatan-kegiatan tersebut dilakukan pada saat yang bersamaan dengan umpan balik yang cepat untuk masing-masing kegiatan dapat dilihat pada Gambar 2.1.



Gambar 2.1 Pengembangan Evolusioner

2.11. Jenis Pengembangan Evolusioner.

Terdapat 2 jenis pengembangan evolusioner yaitu:

1. Pengembangan eksploratori. Tujuan proses ini adalah bekerja dengan pelanggan untuk menyelidiki persyaratan mereka dan mengirimkan sistem akhir. Pengembangan dimulai dengan bagian-bagian sistem yang dipahami. Sistem berubah dengan adanya tambahan fitur-fitur baru sesuai usulan pelanggan.
2. *Prototype* yang dapat dibuang (*throw away*). Tujuan pengembangan evolusioner adalah memahami persyaratan pelanggan dan dengan demikian mengembangkan definisi persyaratan yang lebih baik untuk sistem. *Prototype* berkonsentrasi pada eksperimen, dengan persyaratan pelanggan yang tidak dipahami dengan baik.

Pendekatan evolusioner sering kali lebih efektif dari pendekatan air terjun dalam menghasilkan sistem yang memenuhi kebutuhan langsung pelanggan.

1. Keuntungan

   Spesifikasi dapat dikembangkan secara inkremental. Sementara *user* mendapat pemahaman lebih baik dari masalah mereka, sistem perangkat lunak dapat merefleksikannya.

2. Masalah

Proses tidak bisa dilihat. Manajer memerlukan hasil regular untuk mengukur kemajuan. Jika sistem dikembangkan dengan cepat, tidaklah efektif dari segi biaya jika dihasilkan dokumen yang merefleksikan setiap versi.

Sistem seringkali memiliki struktur yang buruk. Perubahan yang terus-menerus cenderung merusak struktur perangkat lunak. Pernyesuaian perubahan perangkat lunak menjadi kian sulit dan mahal.

Mungkin diperlukan alat bantu dan teknik khusus. Memungkinkan pengembangan yang cepat, tetapi mungkin tidak kompatibel dengan alat bantu atau teknik lain dan relatif hanya sedikit orang yang memiliki keahlian memakainya.

Untuk sistem kecil atau menengah dengan waktu hidup yang pendek, pendekatan evolusioner merupakan pendekatan yang paling baik. Akan tetapi untuk sistem yang besar dan memiliki waktu hidup lama, masalah-masalah dalam pengembangan evolusioner bisa menimbulkan masalah. Untuk sistem-sistem ini, direkomendasikan proses campuran, yang memakai fitur-fitur terbaik dari model air terjun dan pengembangan evolusioner.