BAB II
LANDASAN TEORI

2.1 Sistem Informasi

Sesungguhnya yang dimaksud sistem informasi tidak harus melibatkan komputer. Sistem informasi yang menggunakan komputer biasa disebut sistem informasi berbasis komputer (*Computer Based Information System* atau CBIS). Dalam praktik, istilah sistem informasi lebih sering dipakai tanpa embel-embel berbasis komputer, walaupun dalam kenyataannya komputer merupakan bagian yang penting. Di buku ini, yang dimaksudkan dengan sistem iformasi adalah sistem informasi berbasis komputer.

Ada beragam definisi sistem informasi, sebagaimana tercantum pada Tabel 2.1. Berdasarkan berbagai definisi tersebut, dapat disimpulkan bahwa sistem informasi mencangkup sejumlah komponen (manusia, komputer, teknologi informasi dan prosedur kerja), ada sesuatu yang diproses (data menjadi informasi) dan dimaksudkan untuk mencapai suatu sasaran atau tujuan (Kadir, 2014).

Tabel 2.1 Definisi Sistem Informasi

Sumber	**Definisi**
Alter (1992)	Sistem informasi adalah kombinasi antar prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi.
Bodnar dan Hopwood (1993)	Sistem informasi adalah kumpulan perangkat keras dan peragkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.
Gelinas, Oram dan Wiggins	Sistem informasi adalah suatu sistem buatan manusia yang secara umum terdiri atas sekumpulan

(1990)	komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan dan mengelola data serta menyediakan informasi keluaran kepada para pemakai.
Hall (2001)	Sistem informasi adalah sebuah rangkaian prosedur formal, dimana data dikelompokkan, diproses menjadi informasi dan didistribusikan kepada para pemakai.
Turban, McLean dan Wetherbe (1999)	Sebuah sistem informasi mengumpulkan, memproses, mentimpan, menganalisis dan menyebarkan informasi untuk tujuam yang spesifik.
Wilkinson (1992)	Sistem informasi adalah kerangka kerja yang mengoordinasikan sumber daya (manusia dan komputer) untuk mengubah masukan (*input*) menjadi keluaran (informasi) guna mencapai sasaran-sasaran perusahaan.

2.2 Pemograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai komponen objek yang berisi data dan operasi yang diberlakukan terhadapnya (Rosa, 2011). Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunka dibanggun melalui pendekatan objek secara sistematis. Metodologi berorientasi objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas, yang meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemograman berorientasi objek dan pengujian berorientasi objek. Keuntungan menggunakan metodologi pemograman berorientasi objek adalah sebagai berikut :

a. Meningkatkan produktivitas, karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).

b. Kecepatan pengembangan, karena sistem yang dibangun baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.
c. Kemudahan pemeliharaan, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
d. Adanya konsistensi, karena sifat pewarisan dan pengurangan notasi yang sama pada saat analisis, perancangan maupun pengodean.
e. 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.

2.3 Konsep Dasar Berorientasi Objek

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemograman dan pengujian perangkat lunak (Rosa, 2011). Beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek adalah sebagai berikut :

a. Kelas (*class*)

   Kelas adalah kumpulan objek-objek dengan karakteristik yang sama dan memiliki sifat (atribut). Secara teknis, kelas adalah sebuah struktur tertentu dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.
b. Objek (*object*)

   Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur dan hal-hal lainnya yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat berpengaruh pada status objeknya.
c. Metode (*method*)

Operasi atau metode sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi.

d. Atribut (*attribute*)

Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas.

e. Abstraksi (*abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

f. Enkapulasi (*encapulation*)

Pembungkusan atribut dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.

g. Pewarisan (*inheritance*)

Mekasnisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.

h. Antarmuka (*interface*)

Antarmuka sangat mirip dengan kelas, akan tetapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

i. Reusability

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

j. Generalisasi dan Spealisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus.

k. Komunikasi Antar Objek

Komunikasi antar objek dilakukan lewat pesan yang dikirim dari satu objek ke objek lainnya.

l. Poliformisme (*polymorphism*)

Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

m. Package

Merupakan sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang bebeda.

2.4 Analisa dan Desain Berorientasi Objek

2.4.1 Analisa Berorientasi Objek

Analisa berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahapan untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek. OOA biasanya menggunakan kartu CRC *(Component, Responsibility, Collabolator)* untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML *(Unified Modeling Language)* pada bagian diagram *use case,* diagram kelas dan diagram objek (Rosa, 2011).

2.4.2 Desain Berorientasi Objek

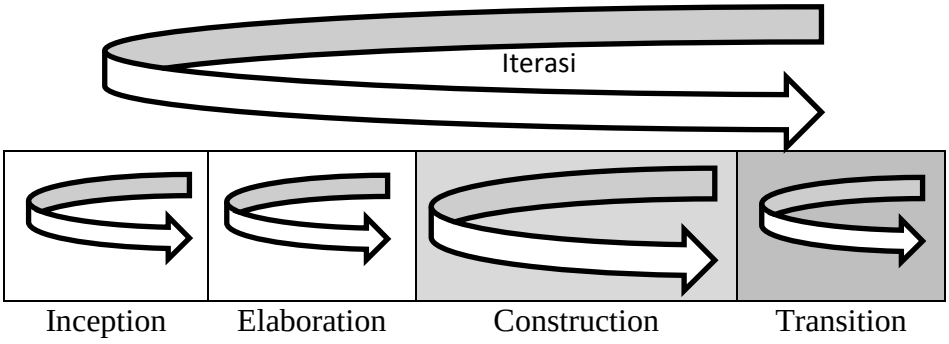
Desain berorientasi objek atau *Objek Oriented Design* (OOD) adalah tahapan perantara untuk memetakan spesifikasi atau kebutuhan sistem yang akan dibangun dengan konsep berorientasi objek ke desain pemodelan agar lebih mudah diimplementasikan dengan pemograman berorientasi objek. Pemodelan beroientasi objek biasanya dituangkan dalam dokumentasi perangkat lunka yang menggunakan perangkat pemodelan berorientasi objek, diantaranya adalah UML (*Unified Modeling Language*).

OOA dan OOD dalam proses yang berulang-ulang seringnya memiliki batasan yang samar, sehingga ke dua tahapan ini sering juga disebut OOAD (*Object Oriented Analysis and Design*) atau dalam bahasa Indonesia berarti Analisis dan desai Berorientasi Objek (Rosa, 2011).

2.5 Metode Pengembangan Perangkat Lunak

Unified Process atau dikenal juga dengan proses iteratif dan inkremental merupakan sebuah proses pengembangan perangkat lunak yang dilakukan secara iteratif (berulang) dan inkremental (bertahap dengan proses menaik). Iteratif bisa dilakukan di dalam setiap tahap atau iteratif tahap pada proses pengembangan perangkat lunak untuk menghasilkan perbaikan fungsi yang inkremental, dimana setiap iterasi akan memperbaiki iterasi berikutnya (Rosa, 2011). Salah satu *Unified Process* yang terkenal adalah RUP (*Rational Unified Process*).

RUP adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang , fokus pada arsitektur, lebih diarahkan berdasarkan penggunaan kasus (*use case driven*). RUP merupakan proses rekayasa perangkat lunak dengan pendefinisian yang baik dan penstrukturan yang baik. RUP memiliki empat buah tahap fase, yaitu seperti pada Gambar 2.2.



Gambar 2.2 Alur Hidup RUP

a. *Inception* (permulaan)

   Tahap ini leih pada memodelkan bisnis yang dibutuhkan dan mendefinisikan kebutuhan akan sistem yang akan dibuat. Tahap yang dibutuhkan pada permulaan ini adalah :

   1. Memahami ruang lingkup dari proyek (termasuk biaya, waktu, kebutuhan, resiko dan lainnya).
   2. Membangun kasus bisnis yang dibutuhkan.

   Hasil yang diharapkan pada tahap ini adalah memenuhi *lifecycle objective milestobe* (batas/tonggak objektif dari siklus) dengan kriteria berikut :

1. Umpan balik dari pendefinisian ruang lingkup, perkiraan biaya dan perkikraan jadwal.
2. Kebutuhan dimengerti dengan pasti dan sejalan dengan kasus primer yang dibutuhkan.
3. Kredibilitas dari perkiraan biaya, perkiraan jadwal, penentuan sekala prioritas, risiko dan proses pengembangan.
4. Ruang lingkup purwarupa (prototype) yang akan dikembangkan.
5. Membangun garis dasar dengan membandingkan perencanaan aktual dengan perencanaan yang direncanakan.

Jika pada akhir tahap ini target yang diinginkan tidak dicapai maka dapat dibatalkan atau diulang kembali setelah dirancang ulang agar kriteria yang diinginkan dapat dicapai.

b. *Elaboration* (perluasan atau perencanaan)

Tahap ini lebih difkuskan pada perencanaan arsitektur sistem. tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (*prototype*). Hasil yang diharapkan pada tahap ini adalah memenuhi *lifecycle objective milestobe* (batas/tonggak objektif dari siklus) dengan kriteria berikut :

1. Model kasus yang digunakan (*use case*) dimana kasus dan aktor yang terlihat telah didefinisikan dan sebagian besar kasus harus dikembangkan.
2. Deskripsi dari arsitektur perangkat lunak telah dibuat.
3. Racangan arsitektur yang dapat diimplementasikan dan mengimplementasikan use case.
4. Kasus bisnis atau proses bisnis dan daftar resiko yang sudah mengalami perbaikan.
5. Rencana pengembangan untuk seluruh proyek telah dibuat.
6. Purwarupa (*prototype*) yang dapat didemonstrasikan untuk mengurangi setiap resiko teknis yang diidentifikasi.

Jika pada akhir tahap ini target yang diinginkan tidak dicapai, maka dapat dibatalkan atau diulang kembali.

c. *Construction* (konstruksi)

Tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. tahap ini lebih pada implementasi dan pengujian sistem yang fokus pada implementasi perangkat lunak atau kode program. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tonggak kemampuan operasional awal.

d. *Transition* (transisi)

Tahap ini lebih pada *deployment* atau inisialisasi sistem agar dapat dimengerti *oleh user*. Tahap ini menghasilkan prosuk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tonggak kemampuan operasional awal. Aktivitas pada tahap ini termasuk pada pelatihan *user*, pemeliharaan dan pengujian sistem.

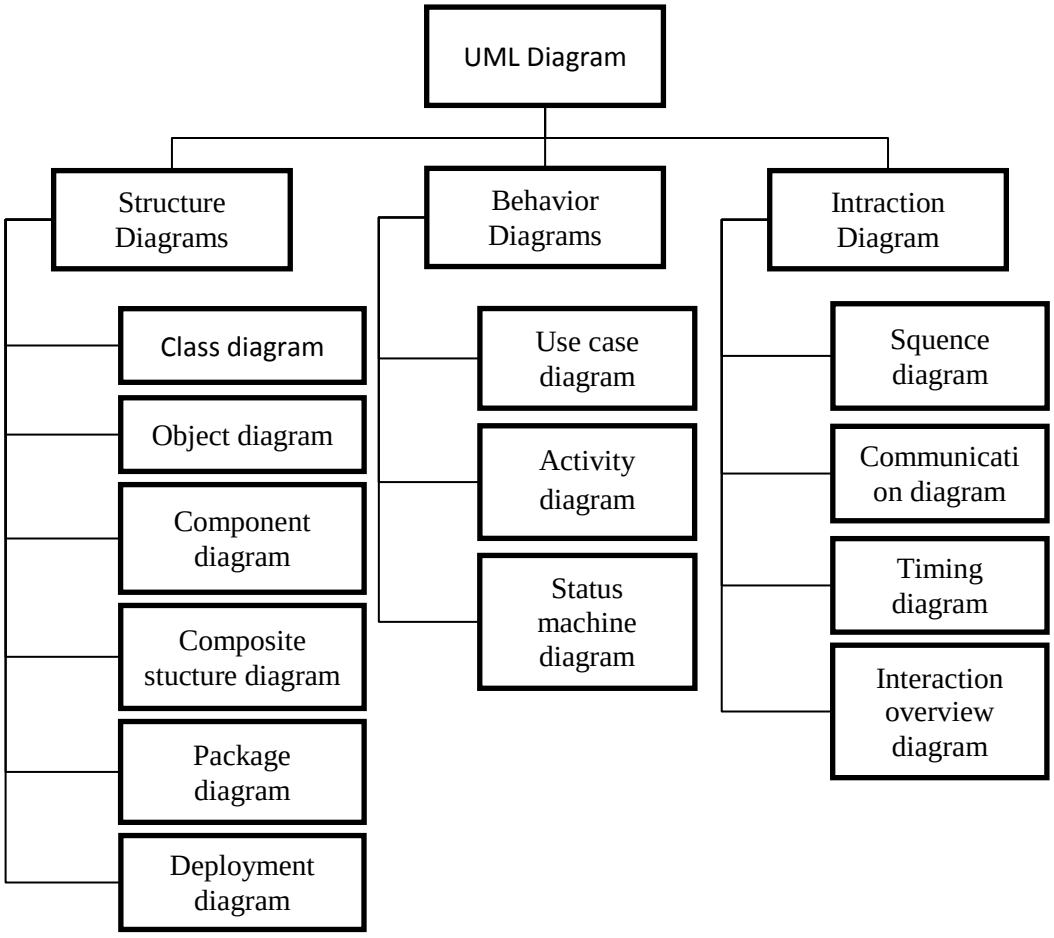
2.6 Alat Bantu Perancangan Sistem

2.6.1 UML

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak yang sesuai dengan teknologi pemograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan diguakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemograman prosedural ataustruktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (wakyu nyata).

Pada perkembangan teknik pemograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan mendokumentasi dari sistem perangkat lunak. UML terdiri dari 13 macam

diagram yang dikelompokkan dalam tiga kategori, yaitu seperti pada Gambar 2.3 (Rosa, 2011).



Gambar 2.3 Diagram UML

Penjelasan dari pembagian kategori tersebut adalah :

a. *Structure diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behavior diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
c. *Interaction diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar sub sistem pada suatu sistem.

2.6.1.1 Use Case

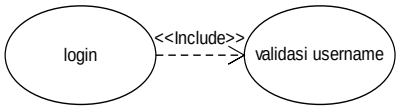
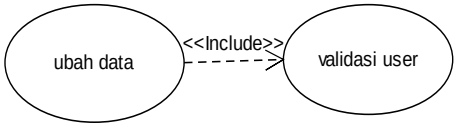
Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam ebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami (Rosa, 2011). Ada dua hal utama pada *use case* yaitu pendefinisian apa yang dibuat aktor dan *use case*.

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi, walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Tabel 2.2 Simbol *Use Case* Diagram

No.	Keterangan	Simbol	Deskripsi
1.	*Use Case*		Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja diawal-awal frase nama *use case*
2.	Aktor		Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar itu sendiri. Jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase

			nama aktor.
3.	Asosiasi		Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4.	Ekstensi	«extends»	Relasi use case tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, misal Arah panah mengarah pada *use case* yang ditambahkan.
5.	Generalisasi		Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya : Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum).
6.	Menggunakan/*include/uses*	<<Include>>	Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : a. Include berarti use case yang

		«uses»	ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut :  b. Include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misal pada kasus berikut :  Ke dua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

2.6.1.2 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapta dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Tabel 2.3 Simbol Diagram Aktivitas

No.	Keterangan	Simbol	Deskripsi
1.	Status awal		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas	aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan		Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan		Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	*Swimlane*	Name Activities	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
6.	Status akhir		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.6.2 Basis Data

Basis data (*database*) adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data dimaksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (DBMS). DBMS adalah perangkat lunak sistem yang

memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses basis data dengan cara yang praktis dan efisien. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda (Kadir, 2014).

Suatu aplikasi dapat berkomunikasi melalui DBMS untuk mengakses basis data dan kemudian membetuk laporan-laporan Selain itu, pemakai juga bisa berinyeraksi secara langsung dengan DBMS untuk mengakses basis data, baik untuk keperluan meminta ataupun untuk melakukan perubahan data. Interaksi secara langsung dengan basis data memungkinkan pemakai untuk memperoleh informasi-informasi yang dibutuhkan sewaktu-waktu dan bersifat sementara, tanpa memerlukan bantuan pemrogram. Umumnya DBMS menyediakan fitur-fitur sebagai berikut :

a. Independensi Data Program

   Karena basia data ditangani oleh DBMS, program dapat ditulis sehingga tidak tergantung pada stuktur data dalam basis data. Dengan perkataan lain, program tidak akan terpengaruh sekiranya bentuk fisik data diubah.

b. Keamanan

   Keamanan dimaksudkan untuk mencegah pengaksesan data oleh orang yang tidak berwewenang.

c. Integritas

   Hal ini ditujukan untuk menjaga agar data selalu dalam keadaan yang valid dan konsisten.

d. Konkurensi

   Konkurensi memungkinkandata dapat diakses oleh banyak pemakai tanpa menimbulkan masalah.

e. Pemulihan (*Recovery*)

   DBMS menyediakan mekanisme untuk mengembalikan basis data ke keadaan semula yang konsisten sekiranya terjadi gangguan perangkat keras atau kegagalan perangkat lunak.

f. Katalog Sistem

Katalog sistem adalah deskripsi tentang data yang terkandung dalam basis data yang dapat diakses oleh pemakai.

g. Perangkat Produktivitas

Untuk menyediakan kemudahan bagl pemakai dan meningkatkan produktivitas, DBMS menyediakan sejumlah perangkakat produktivitas seperti pembangkit *query* dan pembangkit laporan.

Komponen-komponen yang menyusun lingkungan DBMS terdiri atas:

a. Perangkat Keras

Perangkat keras digunakan untuk menjalankan DBMS beserta aplikasi-aplikasinya. Perangkat keras berupa komputer dan periferal pendukungnya. Komputer dapat berupa *PC, mini komputer, main frame* dan lain-lain.

b. Perangkat Lunak

Komponen perangkat lunak mencakup DBMS itu sendiri, program aplikasi serta perangkat lunak pendukung untuk komputer dan jaringan. Program aplikasi dapat dibangun dengan menggunakan bahasa pemrograman seperti *C++, Pascal, Delphi atau Visual BASIC*.

c. Data

Bagi sisi pemakai, komponen terpenting dalam DBMS adalah data karena dari data inilah pemakai dapat memperoleh informasi yang sesuai dengan kebutuhan masing-masing.

d. Prosedur

Prosedur adalah petujuk tertulis yang berisi cara merancang hingga menggunakan basis data. Beberapa hal yang dimasukkan dalam prosedur :

1. Cara masuk ke DBMS (*login*).
2. Cara memekai fasilitas-fasilitas tertentu dalam DBMS maupun cara menggunakan aplikasi.
3. Cara mengaktifkan dan menghentikan DBMS.
4. Cara membuat cadangan basis data dan cara mengembalikan cadangan ke DBMS.

e. Orang

Komponen orang dapat dibagi menjadi tiga kelompok, yaitu :

1. Pemakai akhir (*end-user*).

2. Pemogram aplikasi.
3. Administrator basis data.

2.7 Kebutuhan Perangkat Lunak

2.7.1 Android

Android merupaja sistem operasi berbasis *Linux* yang bersifat terbuka (*open source*) dan dirancang untuk perangkat seluler layar sentuh seperti *smartphone* dan komputer tablet. Android dikembangkan oleh *Android Inc* (Sherief, 2014). Dengan dukungan *financial* dari Google yang kemudian dibeli pada tahun 2005 dan dirilis resmi pada tahun 2007 bersamaan dengan didirikannya *Open Handset Alliance* (OHA).

Sejak tahun 2008, Android terus melakukan sejumlah pembaharuan untuk meningkatkan kinerja sistem operasi. Setiap versi utama yang dirilis dinamakan secara alfabetis berdasarkan nama-nama makanan pencuci mulut atau cemilan bergula, misalnya versi 1.5 bernama *Cupcake*, yang kemudian diikuti oleh versi 1.6 *Donut* sampai ersi terbaru adalah 5.0 *Lollipop*.

Android memiliki empat karakteristik sebagai berikut :

a. Terbuka

   Android dibangun untuk benar-benar terbuka sehingga sebuah aplikasi dapat memanggil salah satu fungsi inti ponsel seperti membuat panggilan, mengirim pesan teks, menggunakan kamera dan lainnya. Android menggunakan sebuah mesin virtual yang dirancang khusus untuk mengoptimalkan sumber daya memori dan perangkat keras yang terdapat di dalam perangkat. *Android* merupakan *open source*, dapat secara bebas diperluas untuk memasukkan teknologi baru yang lebih maju pada saat teknologi tersebut muncul. *Platform* ini akan terus berkembang untuk membangun aplikasi *mobile* yang inovatif.

b. Semua aplikasi dibuat sama

   Android tidak memberikan perbedaan terhadap aplikasi utama dari telepon dan aplikasi pihak ketiga (*third-party application*). Semua aplikasi dapat

dibangun untuk memiliki akses yang sama terhadap kemampuan sebuah telepon dalam menyediakan layanan dan aplikasi yang luas terhadap para pengguna.

c. Memecah hambatan pada aplikasi

Android memecah hambatan untuk membangun aplikasi yang baru dan inovatif. Misalnya, pengembang dapat menggabungkan informasi yang diperoleh dari *web* dengan data pada ponsel seseorang seperti kontak pengguna, kalender atau lokasi geografis.

d. Pengembangan aplikasi yang cepat dan mudah

Android menyediakan akses yang sangat luas kepada pengguna untuk menggunakan library yang diperlukan dan *tools* yang dapat digunakan untuk membangun aplikasi yang semakin baik. *Android* memiliki sekumpulan *tools* yang dapat digunakan sehingga membantu para pengembang dalam meningkatkan produktivitas pada saat membangun aplikasi yang dibuat.

Google Inc. sepenuhnya membangun *Android* dan menjadikannya bersifat terbuka (*open source*) sehingga para pengembang dapat menggunakan *Android* tanpa mengeluarkan biaya untuk lisensi dari *Google* dan dapat membangun *Android* tanpa adanya batasan-batasan. *Android Software Development Kit* (SDK) menyediakan alat dan *Application Programming Interface* (API) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java*.

2.7.2 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Aplikasi-aplikasi berbasis *java* umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai *Mesin Virtual Java* (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin.

Karena fungsionalitasnya yang memungkinkan aplikasi *java* mampu berjalan di beberapa platform sistem operasi yang berbeda, *java* dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web (Wayan, 2012).

2.7.3 Android Studio

Android Studio merupakan sebuah *Integrated Development Environment* (IDE) untuk *platform Android*. *Android Studio* ini diumumkan pada tanggal 16 Mei 2013 pada Konferensi Google I/O oleh Produk Manajer Google, Ellie Powers. Android Studio bersifat *free* dibawah *Apache License* 2.0. *Android studio* awalnya dimulai dengan versi 0.1 pada bulan mei 2013, Kemudian dibuat versi *beta* 0.8 yang dirilis pada bulan juni 2014. Yang paling stabil dirilis pada bulan Desember 2014, dimulai dari versi 1.0. Berbasiskan *JetBrainns' IntelliJ IDEA*, Studio didesain khusus untuk Android Development yang kini sudah bisa di *download* untuk *Windows, Mac OS X,* dan *Linux* (Eric, 2016).

2.7.4 SQLite

SQLite merupakan sebuah sistem manajemen basisdata relasional yang bersifat *ACID-compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. *SQLite* merupakan proyek yang bersifat public domain yang dikerjakan oleh D. Richard Hipp. Tidak seperti pada paradigma *client-server* umumnya, Inti *SQLite* bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead, latency time*, dan secara keseluruhan lebih sederhana.

Seluruh elemen basisdata (definisi data, tabel, indeks dan data) disimpan sebagai sebuah *file*. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan *file* basis data pada saat sebuah transaksi dimulai (Okki, 2012)

2.8 Penelitian Terkait

Penelitian yang terkait dengan penelitian yang sudah dilakukan sebelumnya adalah sebagai berikut :

a. Menurut Nurdiansyah dalam penelitiannya menyimpilkan bahwa Berdasarkan hasil penelitian pada Apotek Tulakan lebih mudah menggunakan sistem informasi berbasis komputer daripada menggunakan sistem konvensional. Dengan adanya sistem informasi Apotek Tulakan dapat memudahkan pekerjaan karyawan sehingga keryawan merasa terbantu dan dengan sistem tersebut pemberian laporan akan lebih cepat.
b. Menurut Tri dalam penelitiannya menyimpulkan bahwa Sistem Informasi Apotek yang dibangun, telah mampu memberikan informasi bagi apotek, yaitu informasi tentang pasien, resep, dan obat. Sistem peresepan yang terintegrasi ke dalam sistem informasi apotek membantu proses penyiapan obat sesuai resep dan juga menghemat waktu tunggu bagi penebus resep karena data resep obat dapat langsung dipantau oleh apoteker. Sistem juga memberikan informasi bagi penebus resep mengenai tanggal resep, harga resep yang harus ditebus dan status resep.
c. Menurut Anisatun dalam penelitiannya menyimpulkan bahwa dengan adanya aplikasi interaksi obat berbasis Android ini, maka pencarian mengenai interaksi obat akan lebih cepat dan mudah.