BAB II
LANDASAN TEORI

2.1 Sistem Informasi

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, yaitu :

a. Alter, sistem informasi adalah kombinasi antar prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi.
b. Bodnar dan Hopwood, sistem informasi adalah kumpulan perangkat keras dan peragkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.
c. Gelinas, Oram dan Wiggins, sistem informasi adalah suatu sistem buatan manusia yang secara umum terdiri atas sekumpulan komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan dan mengelola data serta menyediakan informasi keluaran kepada para pemakai.
d. Hall, Sistem informasi adalah sebuah rangkaian prosedur formal, dimana data dikelompokkan, diproses menjadi informasi dan didistribusikan kepada para pemakai.
e. Turban, McLean dan Wetherbe, sebuah sistem informasi mengumpulkan, memproses, mentimpan, menganalisis dan menyebarkan informasi untuk tujuam yang spesifik.
f. Wilkinson, sistem informasi adalah kerangka kerja yang mengoordinasikan sumber daya (manusia dan komputer) untuk mengubah masukan (*input*) menjadi keluaran (informasi) guna mencapai sasaran-sasaran perusahaan.

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).

2.2 Sistem Informasi Geografis

Sistem Informasi Geografis adalah sistem informasi yang berfungsi untuk mengelola data yang berupa informasi keruangan (spasial). Dalam bahasa inggris, SIG disebut juga *Geographics Information System* (GIS). Informasi spasial berupa posisi koordinat suatu objek, luasan wilayah, dan panjang garis yang diproyeksikan dalam sistem koordinat. Selain informasi spasial, data-data tentang keterangan (atribut) suatu objek luasan wilayah, dan panjang garis merupakan bahan-bahan yang diolah dalam SIG. Dari pengertian diatas dapat ditarik kesimpulan bahwa "SIG merupakan informasi berupa data keruangan yang diperoloh dengan basis komputer yang didapatkan dari hasil analisis dan manipulasi data sebelumnya" (Muhammad, 2016).

2.3 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 lunak dibangun 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.

2.4 Konsep Dasar Berorientasi Objek

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemograman dan pengujian perangkat lunak (Rosa, 2014). 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.

2.5 Metode Pengembangan Perangkat Lunak

(Touseff, Anwer, Hussain, Nadeem, 2015) USDP (*Unified Software Development Process*) salah satu metode pengembangan sistem / perangkat lunak yang menggunakan UML (*Unified Modeling Language*) sebagai *tool* utamanya dengan tahapan yaitu :

2.5.1 Perencanaan (*Planning*)

Studi tentang kebutuhan pengguna (*user's specification*), studi-studi kelayakan (*feasibility study*) baik secara teknis maupun secara teknologi serta penjadwalan pengembangan suatu proyek sistem informasi dan atau perangkat lunak.

2.5.2 Analisis (*Analysis*)

Tahap menggali permasalahan yang muncul pada pengguna dengan mendekomposisi dan merealisasikan *use case* diagram lebih lanjut, mengenali komponen-komponen sistem, objek-objek, hubungan antar objek, dan sebagainya.

2.5.3 Perancangan (*Design*)

Mencari solusi permasalahan yang di dapat dari tahap analisis, pada tahap ini dibagi menjadi dua yaitu :

1. Tahap perancangan yang lebih menekankan pada *platform* apa hasil dari tahap analisis yang akan di imlpementasikan.
2. Tahap perancangan yang dimana melakukan penghalusan (*refinement*) kelas-kelas yang di dapat pada tahap analisis serta menambahkan dan memodifikasi kelas-kelas yang akan lebih mengefisienkan serta mengefektifkan sistem/perangkat lunak yang akan dikembangkan.

2.5.4 Implementasi

melakukan penyesuaian setting perangkat lunak agar bisa dipakai di sisi pengguna (misal, install dan setting database di server pengguna, penyesuaian setting IP) dan melakukan perbaikan coding yang ditemukan selama beta testing.

2.5.5 Pengujian (*Testing*)

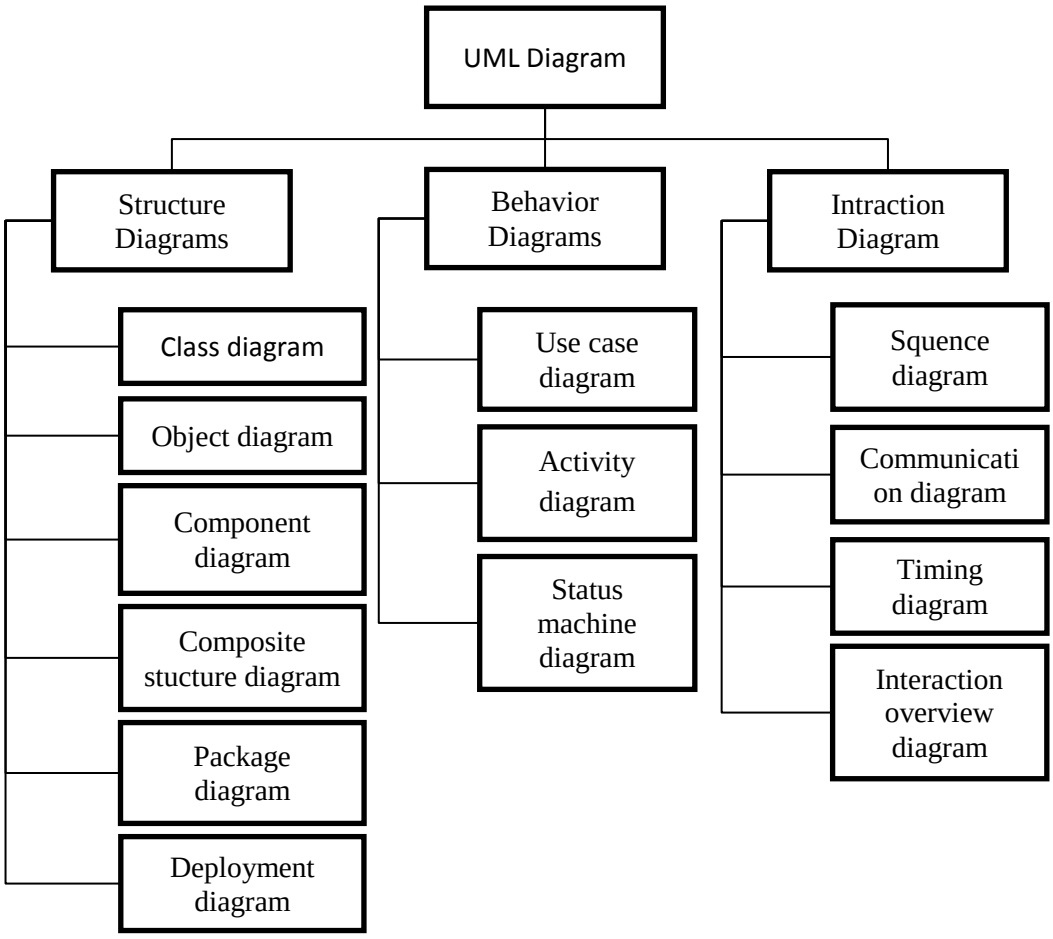
Pada tahap ini digunakan untuk menentukan apakah sistem yang di buat sudah sesuai dengan kebutuhan pengguna atau belum. Jika belum, proses selanjutnya adalah bersifat *interaktif* yaitu kembali ke tahap sebelumnya.

2.6 Alat Bantu Perancangan Sistem

2.6.1 UML (Unified Modeling Language)

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 2.2 (Rosa, 2011).



Gambar 2.1 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

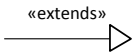
Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam ebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *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.

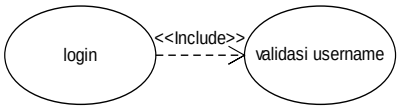
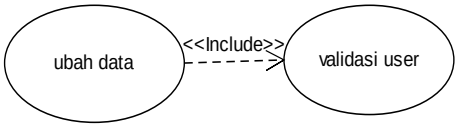
Tabel 2.2 Simbol *Use Case* Diagram

Keterangan	**Simbol**	**Deskripsi**
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*

Tabel 2.2 Simbol *Use Case* Diagram (Lanjutan)

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 merupa kan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Asosiasi		Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi		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.
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).

Tabel 2.2 Simbol Use Case Diagram (Lanjutan)

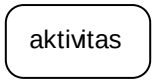
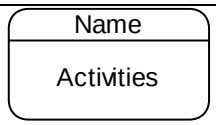
Menggunakan/*include/uses*	<<Include>> «uses»	Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : a. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut :  b. Include berarti use case yang tambahan akan selalu 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

Keterangan	**Simbol**	**Deskripsi**
Status awal		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan		Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan		Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Swimlane		Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
Status akhir		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.6.1.3 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki suatu kelas, sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas (Rosa, 2011).

Kelas-kelas yang ada pada struktur sistem, harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut :

a. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

b. Kelas yang menangani tampilan sistem

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

c. Kelas yang diambil dari pendefinisian *use case*

   Kelas yang menangani fungsi-fungsi yang baru ada diambil dari pendefinisian *use case.*

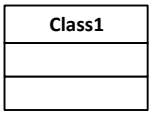
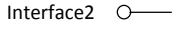
d. Kelas yang diambil dari pendefinisian data

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas tersebut juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks dan lainnya.

Dalam mengidentifikasikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain, sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum, maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah. Simbol-simbol yang ada pada diagram kelas adalah seperti pada Tabel 2.4.

Tabel 2.4 Simbol *Class Diagram*

Simbol	**Deskripsi**
Kelas 	Kelas pada struktur sistem.
Natarmuka/*interface* 	Sama dengan konsep *interface* dalam pemograman berorientasi objek.
Asosiasi 	Relasi antar kelas dalam makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.

Tabel 2.4 Simbol *Class Diagram* (Lanjutan)

Simbol	Deskripsi
Asosiasi berarah	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
Kebergantungan	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi 	Relasi antar kelas dengan maknasemua bagian (*whole-part*).

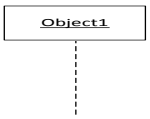
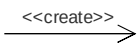
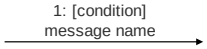
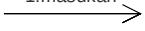
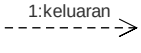
2.6.1.4 Squence Diagram

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikanwaktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interkasi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Rosa, 2011).

Tabel 2.5 Simbol *Squence Diagram*

Simbol	Deskripsi
Aktor 	Orang, proses atau sistem lain yang bernteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang dibuat itu sendiri. Jadi, walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.

Tabel 2.4 Simbol *Squence Diagram* (Lanjutan)

Simbol	Deskripsi
Garis hidup 	Menyatakan kehidupan suatu objek.
Objek 	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan.
Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki operasi atau metode karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode yang menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

2.6.2 Basis Data

Basis data (*database*)adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data di maksudkan 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.

Umumnya DBMSmenyediakan 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 keadaansemula 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. Perankat keras

   Perangkat keras digunakan untuk menjalankan DBMS beserta aplikasi-aplikasinya. Perangkat keras berupa komputer dan periferal pendukungnya. Komputer dapat berupa PC, minikomputer, mainframe, 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 menhentikan 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.

Tedapat beberapa elemen basis data, yaitu :

a. *Database*

Database atau basis data adalah kumpilan tabel yang mempunyai kaitan antara suatu tabel dengan tabel lainya sehingga membentuk suatu bangunan data.

b. Tabel

Tabel adalah kumpulan *record-record* yang mempunyai panjang elemen yang sama dan atribut yang sama namun berbeda data *value*nya.

c. Entitas

Entitas adalah sekumpulan objek yang terdefinisikan yang mempunyai karakteristik sama dan bisa dibedakan satu dengan lainnya. Objek dapat berupa barang, orang, tempat atau suatu kejadian.

d. Atribut

Atribut adalah deskripsi data yang bisa mengidentifikasi entitas yang membedakan entitas tersebut dengan entitas yang lain. Seluruh atribut harus cukup untuk menyatakan identitas objek atau dengan kata lain, kumpulan atribut dari setiap entitas dapat mengidentifikasi keunikan suatu individu.

e. *Data Value* (Nilai Data)

Data value adalah data aktual atau informasi yang disimpan pada tiap data, elemen atau atribut. Atribut nama pegawai menunjukan tempat dimana informasi nama karyawan disimpan, nilai datanya misalnya adalah Anjang, Arif, Suryo dan lain-lain yang merupakan isi data nama pegawai tersebut.

f. *File*

File adalah kumpulan *record* sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda nilai datanya.

g. *Record/Tuple*

Kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entitas secara lengkap. Satu *record* mewakili satu data atau informasi.

2.7 Kebutuhan Perangkat Lunak

2.7.1 Android

Awalnya, *Android* dikembangkan oleh perusahaan kecil di Silicon Valley yang bernama *Android Inc*. Selanjutnya, *Google* mengambil alih sistem operasi tersebut pada tahun 2005 dan mencanangkannya sebagai sistem operasi yang bersifat "*Open Source*". Sebagai konsekuensinya, siapapun boleh memanfaatkannya dengan gratis, termasuk dalam hal kode sumber yang digunakan untuk menyusun sistem operasi tersebut.

Android yang di gunakan dalam sistem tempat ibadah ini mulai dari versi Ice Cream Sandwich samapai dengan Nougat, berikut penjelasan versi android:

a. Android Ice Cream Sandwich v4.0 adalah android pertama yang mempunyai fitur baru membuka kunci dengan pengenalan wajah. fitur ini belum di miliki oleh android versi di bawah Ice Cream Sandwich tetapi sudah disempurnakan pada versi di atasnya.selain itu Ice Cream Sandwich juga mempunyai penampilan Interface yang bersih dan smooth.
b. Android Jelly bean v4.1.2 Android versi Jelly Bean ini dirilis pada 27 Juni 2014 lewat konferensi I/O Google. Jelly Bean menjadi versi Android yang juga banyak mendapatkan update, tercatat 2 kali sudah update dilakukan di Jelly Bean yakni versi 4.1.2 dimana perbedaan dibanding versi sebelumnya adalah segi User Interface yang lebih elegan seta penambahan fitur Google Search.
c. Android Kitkat 4.4 adalah android yang terdapat fitur SMS yang terintegrasi langusng ke dalam aplikasi Google Hangouts serta terdapat fasilitas Could Printing, dimana pengguna dapat Printing secara nirkabel atau mengirim printah ke Laptop atau Pc yang terhubung ke Printer. desain icon dan tema yang lebih unik dan realistik juga dapat mendengarkan printah suara dari google tanpa menguras daya batrei.

d. Android Lollipop 5.0 adalah Salah satu perubahan yang paling menonjol dalam rilis Lollipop adalah user interface yang didesain ulang dan dibangun dengan yang dalam bahasa desain disebut sebagai "material design". Perubahan lain termasuk perbaikan pemberitahuan, yang dapat diakses dari lockscreen dan ditampilkan pada banner di bagian atas screen. Google juga membuat perubahan internal untuk platform, dengan Android Runtime (ART) secara resmi menggantikan Dalvik untuk meningkatkan kinerja aplikasi, dan dengan perubahan yang ditujukan untuk meningkatkan dan mengoptimalkan penggunaan baterai, yang dikenal secara internal sebagai Project Volta.
e. Android Marshmallow 6.0 memperkenalkan model izin yang didesain ulang: sekarang ada hanya delapan kategori izin, dan aplikasi yang tidak lagi secara otomatis diberikan semua hak akses mereka ditentukan pada waktu instalasi. Sebuah sistem opt-in sekarang digunakan, di mana pengguna akan diminta untuk memberikan atau menolak izin individu (seperti kemampuan untuk mengakses kamera atau mikrofon) untuk aplikasi ketika mereka dibutuhkan. Aplikasi mengingat hibah izin mereka, dan mereka dapat disesuaikan oleh pengguna setiap saat. Model izin baru akan digunakan hanya oleh aplikasi yang dikompilasi untuk Marshmallow menggunakan kit pengembangan perangkat lunak (SDK) tersebut, sementara semua aplikasi lainnya akan terus menggunakan model izin sebelumnya.

Hal yang menarik, Android tidak hanya ditujukan untuk ponsel, tetapi juga perangkat elektronik bergerak lainnya. Pada tahun 2012, Android telah digunakan pada peranti-peranti berikut :

a. *Smartphone*.
b. Tablet.
c. Peranti pembaca buku elektronik.
d. *Netbook*.
e. MP4 *player*.
f. TV internet.

Level API menyatakan suatubilangan unik yang digunakan untuk mengidentifikasi API (Application Progrmming Interface) yang digunakan pada suatu versi Android. Dengan kata lain, setiap versi Android ditandai dengan sebuah level API (Abdul Kadir, 2013).

2.7.2 Java

Java adalah Bahasa Java adalah bahasa modern, bahasa ini memiliki kumpulan konsep-konsep terbaik bahasa-bahasa pemrograman sebelumnya(hariyanto,2014).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 .

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 Database SQLite

Menurut Hafizh Herdi(12 jan 2013) Sistem operasi Android mempunyai library database sendiri yang bernama SQLite. SQLite sendiri adalah sebuah library database cross platform yang ditulis menggunakan bahasa C, dan dengan ukuran database yang sangat kecil. Sehingga sangat cocok untuk diimplementasikan pada perangkat mobile yang notabene mempunyai kapasitas ruang penyimpanan yang terbatas. android memiliki fasilitas untuk membuat database yang di kenal dengan SQLite yaitu salah satu software yang embedded yang sangat popular, kombinasi SQL interface dan penggunaan memory yang sangat sedikit dengan kecepatan yang sangat cepat. SQLite di android termasuk android runtime, sehingga setiap versi dari android dapat membuat database dengan SQLite.

dalam sistem android terdapat teknik untuk melakukan penyimpanan data. teknik umum di gunakan adalah sebagai berikut:

a. Shared prefences yaitu meyimpan data beberapa nilai (velue) dalam bentuk groups key yang di kenal dengan frefences.

b. Files yaitu menyimpan data dalam file, dapat berupa menulis ke file atau membaca dari file.

c. SQLite Database yaitu menyimpan data dengan database.

d. Content Providers yaitu menyimpan data dalam bentuk providers service.

2.7.5 Google Maps

Google Maps adalah sebuah jasa peta *globe virtual* gratis dan *online* disediakan oleh Google dapat ditemukan di http://maps.google.com. Peta digital dari Google berbasis web dapat ditempatkan pada *website* tertentu dengan menggunakan *Google Maps API*. *Google Maps* sendiri mempunyai fitur-fitur antara lain navigasi peta dengan *dragging mouse*, *zoom-in* dan *zoom-out* untuk menunjukkan informasi peta secara detil, memberi penanada pada peta dan informasi tambahan. Google telah membuat *Google Maps API* untuk memfasilitasi para *developer* untuk mengintegrasikan Google Maps pada *website*nya. Ini merupakan layanan gratis yang sementara tidak mengandung iklan. Dengan menggunakan *Google Maps API* kita dapat menampilkan seluruh

fasilitas *Google Maps* dengan membuat *API key* (*API Key* ini berfungsi sebagai kunci akses untuk *website*) dan kita sudah dapat menggunakan fungsi-fungsinya yang ada pada *Google Maps API* untuk aplikasi (Agus, 2015).

2.8 Penelitian Terkait

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

a. Menurut Suci Rahma Nursuci dalam penelitiannya menyimpulkan bahwa Sistem Informasi Tempat Ibadah di Kota Bogor dapat secara mudah memperoleh informasi letak dan daya tampung jamaah tempat ibadah.
b. Menurut Novrianti dalam penelitiannya menyimpulkan bahwa Rancang Bangun Sistem Informasi Geografis Untuk Pemetaan Letak Tempat ibadah Kecamatan Toboali Berbasis Web ini memberikan inventarisasi keberadaan tempat ibadah oleh pemerintah daerah toboali sangat baik 28%, baik 50%, kurang baik 13% dan 1% menilai buruk.