BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Penelitian terdahulu yang berhubungan penulisan skripsi dapat dilihat pada sebagai berikut ini :

Tabel II.1. Penelitian Terkait

No	Nama	Judul	Hasil
1	Pratiwi (2015)	Analisis Dan Desain Sistem Informasi Simpan Pinjam Pada Koperasi Sejahtera Bersama Bandung	Sistem informasi yang berjalan disini masih belum efektif karena masih di kombinasikan dengan proses manual untuk proses pinjamannya, hal tersebut memakan waktu lama untuk setiap transakasinya, namun seiring dengan pesatnya perkembangan teknologi, maka sudah seharusnya masalah perkoperasian tersebut diberi sentuhan teknologi sebagai solusi contohnya seperti dibangun sistem informasi untuk memudahkan setiap operasional yang ada. Sistem informasi simpan pinjam ini diharapkan dapat meningkatkan kinerja koperasi itu sendiri, karena sistem ini dirancang dengan tujuan mempersingkat waktu transaksi agar operasional koperasi bisa berjalan lebih efektif.
2	Mohammad Fuad (2015)	Perancangan Sistem Informasi Simpan Pinjam Pada Koperasi“ Kopitama” Depok”	“Penelitian ini bertujuan untuk membuat sistem informasi simpan pinjam pada koperasi “KOPITAMA” Sawangan - Depok. Prosedur simpan pinjam pada koperasi KOPITAMA ditemukan

			beberapa kelemahan diantaranya adalah dokumen pencatatan transaksi simpanan maupun pinjaman, serta laporan yang dihasilkan. Penelitian ini dilakukan untuk membuat perancangan sistem informasi simpan pinjam, dengan tujuan untuk menyempurnakan sistem simpan pinjam dari sistem manual.

II.2. Landasan Teori

II.2.1. Sistem

Sistem dapat terdiri dari sistem-sistem bagian (*subsystem*). Sebagai misal, sistem komputer dapat terdiri dari subsistem perangkat keras dan subsistem perangkat lunak. Masing-masing sub sistem dapat terdiri dari subsistem-subsistem yang lebih kecil atau terdiri dari komponen-komponen. Subsistem perangkat keras (*hardware*) dapat terdiri dari alat masukan, alat pemroses, alat keluaran dan simpanan luar, dan kemudian subsistem-subsistem tersebut akan berinteraksi sedemikian rupa sehingga dapat mencapai satu kesatuan yang terpadu. Dalam buku Analisa dan Design Sistem Informasi Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis. "Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu " (Deppi Linda : 2016 : 62).

II.2.1.1. Karakteristik Sistem

Sistem mempunyai karakteristik atau sifat-sifat tertentu, diantaranya adalah sebagai berikut :

1. Komponen Sistem (*Components*)

   Sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. Batas Sistem (*Boundary*)

   Batas sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan. Batas sistem menunjukan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environments*)

   Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga merugikan suatu sistem.

4. Penghubung Sistem (*Interface*)

   Penghubung merupakan media penghubung antara suatu subsistem dengan subsistem yang lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari suatu subsistem ke subsistem lainnya.

5. Masukan Sistem (*Input*)

   Masukan adalah energi yang dimasukan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*).

6. Keluaran sistem (*Output*)

   Keluaran adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran dapat berupa masukan untuk subsistem yang lain.

7. Pengolah Sistem (*Process*)

   Sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolah. Pengolah yang akan merubah masukan menjadi keluaran.

8. Sasaran Sistem (*Objectives*)

   Sistem mempunyai tujuan atau sasaran yang akan menentukan masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem (Deppi Linda : 2016 : 62).

II.2.2. Sistem Informasi

Sistem informasi adalah sejumlah komponen (manusia, komputer, teknologi informasi, dan prosedur kerja), ada sesuatu yang diproses (data menjadi informasi), dan dimaksudkan untuk mencapai suatu sasaran atau tujuan. Penggunaan sistem informasi telah banyak diterapkan diberbagai bidang termasuk dalam bisnis. Salah satu tujuan penerapan sistem informasi dalam bidang bisnis agar dapat meningkatkan keuntungan bisnis dengan menggunakan

kemampuan yang didapatkan dari sistem informasi. Ada beberapa kemampuan dari sistem informasi yang dapat mendukung dalam bidang bisnis. Kemampuan tersebut seperti pengurangan biaya, mempercepat pekerjaan, dapat meningkatkan kemudahan dalam pengambilan keputusan, dan peningkatan pelayanan terhadap pelanggan. (Alfian Nurlifa : 2017)

Faktor-faktor yang menentukan kehandalan dari suatu sistem informasi atau informasi dapat dikatakan baik jika memenuhi kriteria-kriteria sebagai berikut :

a) Keunggulan (*usefulness*)

   Yaitu suatu sistem yang harus dapat menghasilkan informasi yang tepat dan relevan untuk mengambil keputusan manajemen dan personil operasi dalam organisasi.

b) Ekonomis

   Kemampuan sistem yang mempengaruhi sistem harus bernilai manfaat minimal, sebesar biayanya.

c) Kehandalan (*Reliability*)

   Keluaran dari sistem harus mempunyai tingkat ketelitian tinggi dan sistem tersebut harus beroperasi secara efektif.

d) Pelayanan (*Customer Service*)

   Yakni suatu sistem memberikan pelayanan yang baik dan efisien kepada para pengguna sistem pada saat berhubungan dengan organisasi.

e) Kapasitas (*Capacity*)

Setiap sistem harus mempunyai kapasitas yang memadai untuk menangani setiap periode sesuai yang dibutuhkan.

f) Sederhana dalam kemudahan (*Simplicity*)

Sistem tersebut lebih sederhana (umum) sehingga struktur dan operasinya dapat dengan mudah dimengerti dan *prosedure* mudah diikutin.

g) Fleksibel (*Fleksibility*)

Sistem informasi ini harus dapat digunakan dalam kondisi yang bagaimana yang diinginkan oleh organisasi tersebut atau pengguna tertentu.

h) Komponen Sistem Informasi

Istilah dalam komponen sistem informasi adalah blok bangunan (*building block*) yang dapat di bagi menjadi enam blok yaitu :

a. Blok masukan (*input block*)

Blok input merupakan data–data yang masuk ke dalam sistem informasi, yang dapat berupa *document-document* dasar yang dapat diolah menjadi suatu informasi tertentu.

b. Blok model (*model block*)

Blok ini terdiri dari kombinasi prosedur, logika dan model matematik yang akan mengolah data *input* untuk menghasilkan suatu informasi yang dibutuhkan.

c. Blok keluaran (*output block*)

Merupakan informasi yang menghasilkan sekumpulan data yang nantinya akan disimpan berupa data cetak laporan.

d. Blok teknologi (*technologi block*)

Blok teknologi merupakan penunjang utama dalam berlangsunganya sistem informasi. Yang memiliki beberapa komponen yaitu diantaranya alat memasukan data (*input device*), alat untuk menyimpan dan mengakses data (*storage device*), alat untuk menghasilkan dan mengirimkan keluaran (*output divice*) dan alat untuk membantuk pengendalian sistem secara keseluruan (*control device*). Teknologi informasi terdiri dari 3 (tiga) bagian utama, yaitu teknisi (*humanware* atau *brainare*), perangkat lunak (*software*), dan perangkat keras (*hardware*).

e. Blok basis data (*database block*)

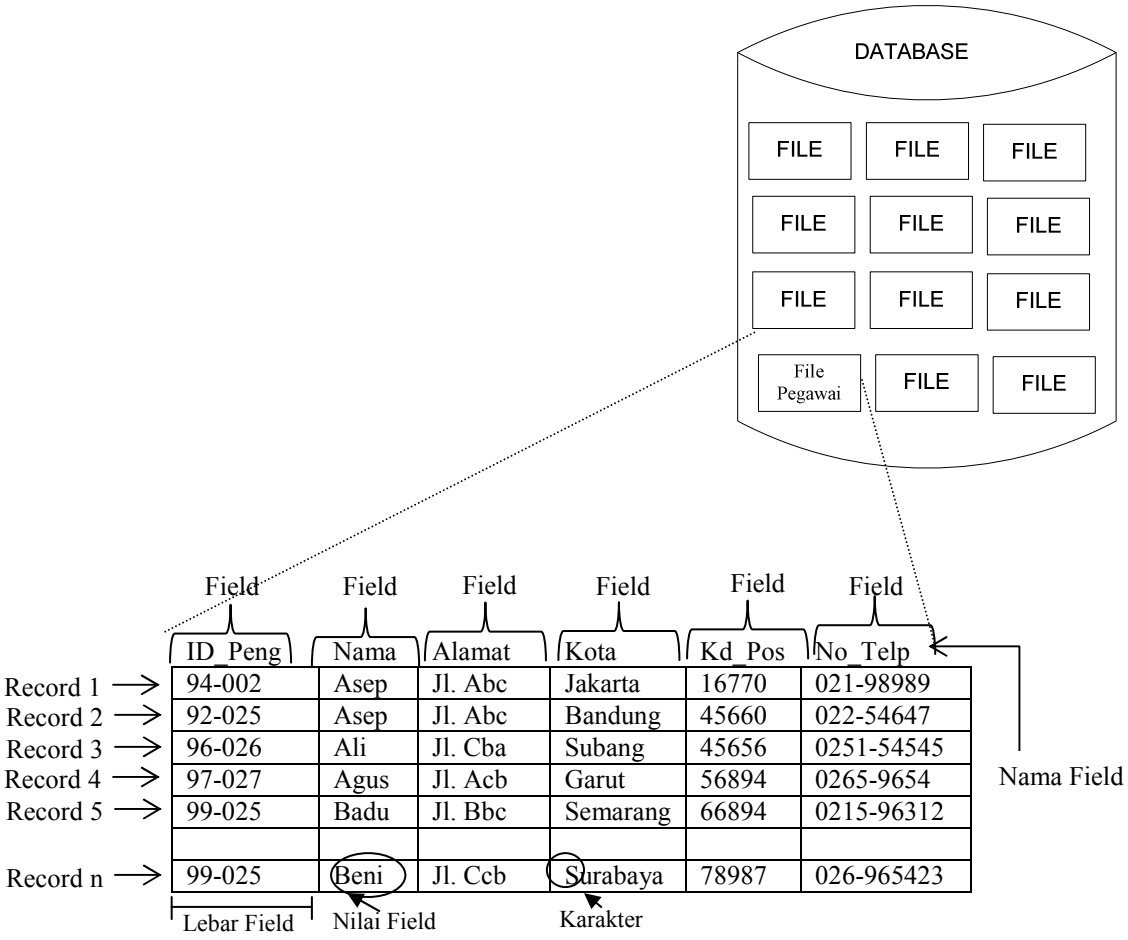
Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkap keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu di simpan dan perlu di organisasi sedemikian rupa, supaya informasi yang dihasilkan berkualitas.

f. Blok kendali (*control block*)

Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal–hal yang dapat merusak sistem dapat di cegah bila terlanjur terjadi. (I Made Budi Adnyana : 2016)

II.2.3. *Database*

Database merupakan kumpulan dari item data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dan dengan *softwar*e untuk melakukan manipulasi untuk kegunaan tertentu *Database* dapat juga diartikan Koleksi data yang terorganisasi untuk melayani beragam aplikasi secara efisien dengan mensentralisasi data dan meminimalisasi data yang berlebih. Sebagian besar aplikasi *database* memiliki bahasa yang khas yang disebut bahasa manipulasi data yang digunakan dalam hubungannya dengan beberapa bahasa pemrograman aplikasi konvensional untuk memanipulasi data dalam *database*. Bahasa ini mengandung perintah-perintah yang memungkinkan pengguna akhir dan para ahli pemrograman untuk mengekstrak data dari *database* untuk memenuhi kebutuhan informasi dan mengembangkan aplikasi. Bahasa manipulasi yang paling banyak digunakan dewasa ini adalah *Structured Query Languange* (SQL).



Gambar II.1. Hirarki *Database*
Sumber : Mukhlisulfatih Latief : 2016

Bahasa SQL tersusun atas 3 kelompok pernyataan berdasarkan fungsi dari pernyataan tersebut yaitu:

1. Data *Definition Languange* (DDL) : Mendefinisikan jenis data yang akan dibuat (dapat berupa angka atau huruf), cara relasi data, validasi data dan lainnya. Contoh : *create, drop, alter tale*.
2. Data *Manipulation Languange* (DML) : Data yang telah dibuat dan didefinisikan tersebut akan dilakukan beberapa pengerjaan, seperti menyaring data, melakukan proses *query*. Contoh : *select, update, insert.*

3. Data *Control Languange* (DCL) : Bagian ini berkenaan dengan cara mengendalikan data, seperti siapa saja yang bisa melihat isi data, bagaimana data bisa digunakan oleh banyak *user.*

II.2.4. Normalisasi

Agar model *database* relasional bisa digunakan secara efektif, maka pengelompokkan yang rumit harus disederhanakan sehingga data berlebih dan relasi yang salah bisa dieliminasi. Proses membuat struktur data yang lebih kecil dan stabil dari sekelompok data yang rumit disebut Normalisasi.

Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data / *database,* teknik pengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik tanpa redudansi. Tujuan normalisasi adalah mengorganisasikan data kedalam tabel-tabel untuk memenuhi kebutuhan pemakai, menghilangkan kerangkapan data, mengurangi kompleksitas, mempermudah modifikasi data. (Mukhlisulfatih Latief : 2016)

1. Proses Normalisasi
    a. Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu kebeberapa tingkat.
    b. Apabila tabel yang diuji belum memenuhi persyaratan tertentu maka tabel tersebut pelu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.

II.2.5. Visual Studio 2010

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasi lainnya dalam bentuk aplikasi *console*, aplikasi *Windows,* ataupun aplikasi Web. Kompiler yang dimasukkan ke dalam paket *Visual Studio* antara lain *Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro*, dan *Visual SourceSafe*. *Microsoft Visual Studio* dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas *.NET Framework*).

Selain itu, *Visual Studio* juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas .NET *Compact Framework*).*Visual Studio* kini telah menginjak versi *Visual Studio* 12.0 atau dikenal dengan sebutan *Microsoft Visual Studio* 2013 yang diluncurkan pada 17 Oktober 2013 yang ditujukan untuk *platform Microsoft .NET Framework* 4.5.1. Versi sebelumnya Visual Studio 2012 ditujukan untuk platform 4.5, Visual Studio 2010 ditujukan untuk *.NET Framework* 4.0, *Visual Studio* 2008 ditujukan untuk *platform .NET Framework* 3.5, *Visual Studio* 2005 ditujukan untuk *platform .NET Framework* 2.0 dan 3.0. *Visual Studio* 2003 ditujukan untuk *.NET Framework 1*.1, dan *Visual Studio* 2002 ditujukan untuk *.NET Framework* 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan *Visual Studio.NET*, karena memang membutuhkan *Microsoft .NET Framework*. Sementara itu, sebelum

muncul *Visual Studio .NET*, terdapat *Microsoft Visual Studio* 6.0. *(Ahmad Rasi* Ruli*; 2017 : 10)*

II.2.6. Sql Server

SQL (*Structured Query Language*) adalah bahasa *non procedural* untuk mengakses data pada *database* relasional. SQL adalah bahasa *database* yang dipergunakan dalam menyelesaikan permasalahan dalam *database* serta mempunyai kelebihan dalam mengolah data. Standar SQL mula-mula didefenisikan oleh ISO *(International Standards Organization)* dan ANSI *(the American National Standards Institute)* yang dikenal dengan sebutan SQL86. (Eka Iswandy : 2015 : 73)

Dengan menggunakan SQL, kita dapat melakukan hal-hal berikut:

1. Memodifikasi struktur *database* .
2. Mengubah, mengisi, menghapus isi *database*.
3. Mentransfer data antara *database* yang berbeda. SQL ada yang dikembangkan untuk PC dan ada juga yang dikembangkan untuk dapat mengakomodasi *database* yang sangat besar.

Beberapa contohnya antara lain:

1. *Microsoft Accsess*

Digunakan untuk PC, sangat mudah dipakai dimana perintah SQL dapat langsung dimasukkan atau melalui fasilitas yang telah digunakan.

2. *Microsoft Query*

SQL yang dipaket dengan produk lain dari *Microsoft Windows*, yaitu *Microsoft Visual Studio* seperti *Visual Basic* dan *Visual* C++. Untuk terhubung dengan *database* lain menggunakan ODBC.

3. *Oracle*

Digunakan untuk perusahaan yang mengggunakan *database* besar. (Eka Iswandy : 2015 : 73)

II.2.7. *UML* (*Unified Modelling Language*)

Unified Modelling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak sebuah sistem. UML lebih mengedepankan penggunaan diagram untuk menggambarkan aspek dari sistem, karena tergolong bahasa visual yang lebih mudah dan lebih cepat dipahami dibandingkan dengan bahasa pemrograman. *Unified Modelling Language* (UML) biasa digunakan untuk :

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan *use case* dan *actor*.
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams*.
3. Menggambarkan representasi struktur *static* sebuah sistem dalam bentuk *class diagram*.

4. Membuat model *behavior* yang menggambarkan kebiasaan atau sifat sebuah sistem dengan *state transition diagrams* UML.
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development diagrams*.
6. Menyampaikan atau memperluas *fungsionalty* dengan *stereo types*.

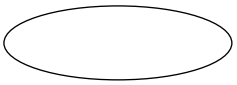
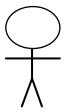
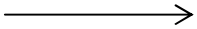
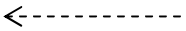
Pemodelan penggunaan UML merupakan metode pemodelan berorientasi objek dan berbasis visual. Karenanya pemodelan objek yang fokus pada pendefinisian struktur statis dan model sistem informasi yang dinamis daripada mendefinisikan data dan model proses yang tujuannya adalah pengembangan tradisional. UML menawarkan diagram yang dikelompokkan manjadi lima perspektif berbeda untuk memodelkan suatu sistem. Seperti satu *set blue print* yang digunakan untuk membangun sebuah rumah (Saipul Anwar, et al., 2016 : 75-76).

II.2.7.1. *Use Case* Diagram

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Ade Hendini, 2016 : 108).

Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1 dibawah ini :

Tabel II.2. Simbol *Use Case*

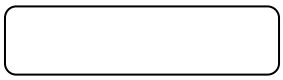
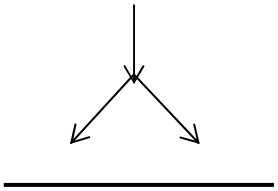
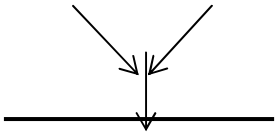
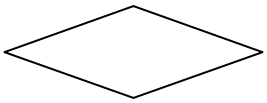
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend,* merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

***(Sumber* : Ade Hendini, 2016)**

II.2.7.2. *Activity* Diagram

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2 dibawah ini:

Tabel II.3. Simbol *Activity Diagram*

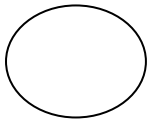
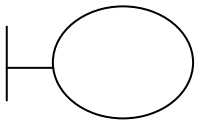
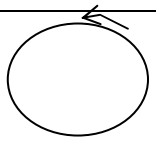
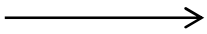
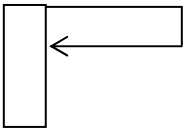
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

***(Sumber :* Ade Hendini, 2016 : 109)**

II.2.7.3. *Sequence* Diagram

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3 dibawah ini :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

***(Sumber :* Ade Hendini, 2016 : 110)**

II.2.7.4. *Class* Diagram

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class* diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class* diagram secara khas

meliputi : Kelas (*Class*), Relasi *Assosiations*, *Generalitation* dan *Aggregation*, attribut (*Attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality* (Ade Hendini, 2016 : 111).

Tabel II.5. *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

***(Sumber :* Ade Hendini, 2016 : 111)**