BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

Dalam proses pengembangan sistem informasi, diperlukan adanya pemahaman mengenai konsep-konsep dari sistem informasi tersebut. Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urutan-urutan operasi didalam sistem. Sistem juga dapat diartikan suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan kegiatan tertentu.

2.1.1. Pengertian Sistem

Sistem dapat didefinisikan sebagai kumpulan dari komponen yang saling berhubungan satu dengan yang lainnya membentuk satu kesatuan untuk mencapai tujuan tertentu, Jogiyanto (2009:34)

Menurut Scott dalam buku Fatta (2007:4) mengemukakan bahwa, ”Sistem terdiri dari unsur-unsur seperti masukan *(input),* pengolahan *(processing),* serta keluaran *(output).*

Sedangkan menurut Mc.Leod dalam Fatta (2007:4) mengemukakan bahwa, ”Sistem sebagai sekelompok elemen-elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan”.

Dari beberapa definisi diatas dapat disimpulkan bahwa sistem merupakan elemen-elemen yang saling berhubungan satu dengan yang lain dengan maksud

yang sama untuk mencapai tujuan tertentu dan memiliki unsur-unsur *input*, *processing* dan *output*.

2.1.2 Karakteristik Sistem

Menurut Ladjamudin (2013:3) memberikan batasan bahwa, ”suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, yaitu mempunyai komponen-komponen, batas sistem, lingkungan luar sistem, pendukung, masukan, keluaran, pengolah dan sasaran atau tujuan”.

Suatu sistem mempunyai karakteristik sistem yang dapat membedakan suatu sistem dengan sistem lainnya, Fatta (2007 : 5) yaitu sebagai berikut:

1. Komponen Sistem (*components*)

   Komponen sistem adalah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. komponen sistem tersebut dapat berupa suatu bentuk subsistem, setiap subsistem memiliki sifat-sifat sistem yang menjalankan suatu fungsi tertentu.

2. Batasan Sistem (*Bondary*)

   Batasan Sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luar sistem tersebut.

3. Lingkungan Luar Sistem (*Environment*)

   Lingkungan luar sistem adalah batasan dari sistem yang mempengaruhi operasi s istem tersebut.

   lingkungan luar yang merugikan harus dikendalikan kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem (*Interface*)

   Media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem atau *interface*.

5. Masukan Sistem (*Input*)

   Energi yang dimasukkan ke dalam sistem masukkan dapat berupa *maintenance input* dan *signal input*.

6. Proses sistem (*Procces*)

   Proses sistem dapat mempunyai suatu bagian pengolah data yang akan merubah masukkan menjadi keluaran.

7. Keluaran Sistem (*Output*)

   Keluaran Sistem adalah hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna.

8. Sasaran dan tujuan Sistem *(objective*)

   Suatu sistem pasti mempunyai tujuan (*goal)* atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran maka operasi tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukkan yang dibutuhkan dan keluaran sistem yang akan dihasilkan oleh sistem itu sendiri.

2.1.2. Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lain karena memiliki sasaran yang berbeda untuk setiap kasus yang terjadi di dalam sistem tersebut (Ladjamudin, 2013:6).

Sedangkan menurut Paryati (2008:5) menyatakan bahwa, ”Pada suatu sistem yang dinamik, sistem memiliki tingah laku yang berbeda dengan sistem lain”.

Pola tingkah laku sistem dibedakan menjadi 4 jenis yaitu:

1. Sistem Abstrak *(Abstrac System)* dan Sistem Fisik *(Phisycal System)*

   Sistem Abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak nampak secara fisik, Sedangkan sistem fisik adalah sistem yang ada secara fisik atau secara nyata.

2. Sistem Alamiah *(Natural system)* dan Sistem Buatan Manusia *(human machine System)*

   Sistem Alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia, sedangkan Sistem buatan manusia merupakan sistem yang melibatkan hubungan manusia dengan mesin, yang disebut dengan *human machine system*.

3. Sistem pasti *(Deterministic System)* dan sistem tidak pasti *(Probalistic System)*

   Sistem pasti adalah Sistem yang beroperasi dengan tingkah laku yang dapat diprediksi, Sedangkan sistem yang bersifat tidak pasti adalah sistem yang kondisi masa depannya tidak dapat diprediksi, karena mengandung unsur *probabilitas*.

4. Sistem Terbuka *(Open System)* dan Sistem Tertutup *(Close System)*

   Sistem tertutup adalah sistem yang tidak berhubungan dan tidak terpengaruh oleh lingkungan luarnya, sedangkan sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya, yang menerima masukan dan menghasilkan keluaran untuk subsistem lainnya.

2.1.3. Pengertian Informasi

Informasi (*information*) adalah data yang diolah menjadi bentuk yang berguna bagi para pemakainya, (Jogiyanto, 2009:36).

Sedangkan menurut Kusrini dan Koniyo (2007:7) memberikan batasan bahwa, "informasi adalah data yang sudah diolah menjadi sebuah bentuk yang berarti bagi pengguna, yang bermanfaat dalam pengambilan keputusan saat ini atau mendukung sumber informasi".

Dari beberapa definisi di atas dapat disimpulkan bahwa informasi merupakan data yang diolah menjadi bentuk yang berguna yang dapat menambah pengetahuan bagi para pemakainya dan digunakan untuk pengambilan keputusan.

2.1.4. Pengertian Sistem Informasi

Menurut Hutahaean (2015:13) Mengemukakan bahwa "Sistem infomasi adalah suatu sistem didalam suatu organsasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan".

Sedangkan menurut Ladjamudin (2013:13) memberikan batasan bahwa, "sistem infomasi adalah suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan informasi".

Dari kutipan diatas dapat disimpulkan bahwa sistem informasi adalah kegiatan atau aktifitas yang ada dalam suatu organisasi untuk menyajikan

informasi dan meyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan.

2.1.4. Pengertian Sistem Informasi Akuntansi

Sistem informasi akuntansi sebagai sistem informasi yang merubah data transaksi bisnis menjadi informasi keuangan yang berguna bagi pemakainya (Jogiyanto, 2009:227).

Menurut Mulyani (2016:24) memberikan batasan bahwa, "Sistem informasi akuntansi digunakan sebagai alat umtuk melakukan analisis keputusan ataupun sebagai pembuat keputusan yang terkait dengan transaksi-transaksi perusahaan ".

Dari pendapat diatas dapat disimpulkan bahwa sistem informasi akuntansi merupakan informasi yang merubah data transaksi bisnis menjadi informasi keuangan dan sebagai alat untuk melakukan analisis keputusan ataupun sebagai pembuat keputusan yang terkait dengan transaksi perusahaan.

2.1.5. Pengertian Penerimaan Kas

Penerimaan kas merupakan suatu prosedur catatan yang dibuat untuk melaksanakan kegiatan penerimaan uang yang berasal dari berbagai macam sumber yaitu dari penjualan tunai,penjualan aktiva, pinjaman baik, dan setoran modal baru (Sujarweni, 2015:121).

2.1.6. Penegertian Pengeluaran Kas

Menurut Sujarwani (2015:123) memberikan batasan bahwa, "sistem akuntansi pengeluaran kas merupakan sistem yang membahas keluarnya uang yang digunakan untuk pembelian tunai maupun kredit dan untuk pembayaran".

2.2. Peralatan Pendukung *(Tools System)*

Peralatan pendukung (*tools System*) ini berupa alat yang digunakan untuk menggambarkan bentuk logika dari suatu sistem yang menggunakan simbol-simbol, lambang, diagram yang menunjukan secara tepat arti dan fungsinya. Adapun peralatan pendukung yang digunakan seperti *object oriented programming* (OOP), *Unified Modeling Language* (UML), *Entity Relationship Diagram* (ERD), *Logikal Record Structure* (LRS), Basis Data/*Database*, *Xampp, PHPMyAdmin*, *NetBeans* dan *iReport.*

2.2.1. *Object Oriented Programming* (OOP)

Menurut Supardi (2009:128) Pengertian "*Object Oriented Programming* (OOP) Merupakan cara baru berpikir, pandangan, atau paradigma baru untuk membuat program atau merancang sistem dengan memperhatikan objek, ciri objek dan perilakunya".

Istilah-istilah dalam OOP antara lain:

1. Kelas (*class*)

   Kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh "*class of dog*" adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menujuk pada berbagai

macam perilaku atau turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek.

2. Metode (*Method*)

   Merupakan suatu operasi berupa fungsi-fungsi yang dapat dikerjakan oleh suatu objek. *Method* didefinisikan pada *class* akan tetapi dipanggil melalui objek. Metode menentukan perilaku objek, yakni apa yang terjadi ketika objek itu dibuat serta berbagai operasi yang dapat dilakukan objek sepanjang hidupnya. Metode memiliki 4 (empat) bagian dasar:

   a. Nama metode
   b. Tipe objek atau tipe primitive yang dikembaliak metode
   c. Daftar parameter
   d. Badan atau isi metode.

   Tiga bagian pertama megindikasikan informasi penting tentang metode itu sendiri.dengan nama lain, nama metode tersebut = metode lain dalam program.

3. Abtraksi (*Abstraction*)

   Kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari “perilaku” abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomuniksi dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak,dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

2.2.2. *Unified Modeling Language* (UML)

UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek (Rosa dan Shalahudin, 2015:133).

Sedangkan menurut Nugroho (2010:6) menerangkan bahwa "*Unified Modeling Language* (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma atau berorientasi objek".

Dari beberapa definisi diatas penulis menyimpulkan bahwa *Unified Modeling Language* (UML) adalah standar bahasa pemodelan untuk sistem atau perangkat lunak dalam pemrograman berorientasi objek.

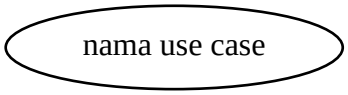
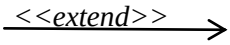
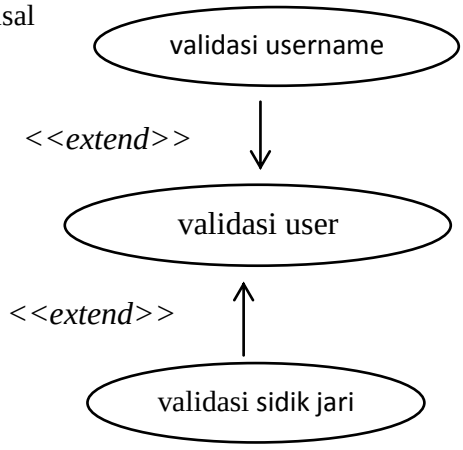
2.2.2.1. *Use Case Diagram*

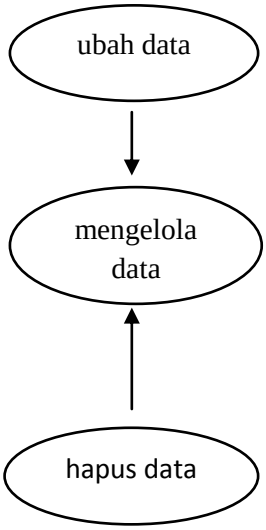
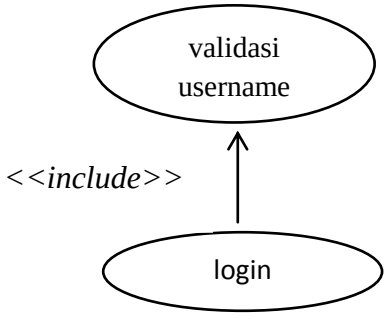
Menurut Rosa dan Shalahudin (2015:155) memberikan batasan bahwa, "*Use case* atau diagram *use case* merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat".

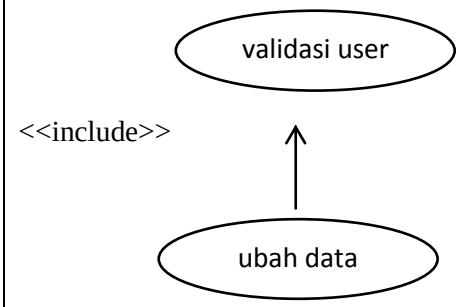
Use case mendefinisikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mugkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor, dan *use case*.

Tabel II.1.
Simbol-simbol yang ada pada Diagram Use Case

Simbol	Deskripsi
Use case 	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama use case.
Aktor / *actor*  nama actor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat diluar sistem informasi itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang; tapi biasanya dinyatakan menggunakan kata benda diawal frase nama actor
Asosiasi / *association* _____	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan actor
Ekstensi / *extend* 	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 *intheritance* 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;biasanya use case yang menjadi *extend*-nya merupakan jenis yang sama dengan use case yang menjadi induknya

Generalisasi / generalization 	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 pada use case yang menjadi generalisasinya (umum)
Menggunakan / *include* / *uses*  <<*include*>> <<*uses*>> 	Relasi *use case* tambahan ke sebuah *use case* dimana use case yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini Ada dua sudut pandang yang cukup besar mengenai include *use case:* • Include berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan, misal pada kasus berikut:  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:

	 Kedua interprestasi diatas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interprestasi yang dibutuhkan.

Sumber: Rosa A. S dan M. Shalahuddin (2015:156)

2.2.2.2. *Activity Diagram*

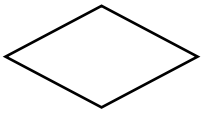
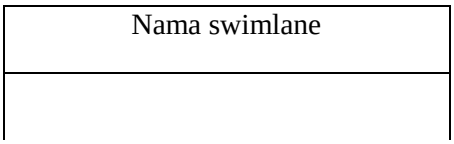
Rosa dan Shalahudin (2015:161) menyimpulkan bahwa:

> Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan yang dilakukan aktor, jadi aktivitas yang dapat 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 / unsur *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
d. Rancangan menu yang ditampilkan pada perangkat lunak

Tabel II.2.
Simbol-simbol yang ada pada Diagram Aktivitas

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

Sumber: Rosa A. S dan M. Shalahuddin (2015:162)

2.2.2.3. *Deployment Diagram*

Diagram deployment atau *deployment diagram* merupakan konfigurasi komponen dalam proses eksekusi aplikasi (Rosa dan Shalahudin, 2015:154).

Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut:

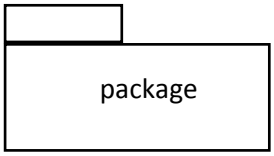
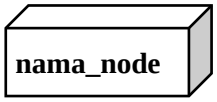
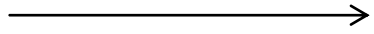
a. Sistem tambahan (*embedded system*) yang menggambarkan rancangan *device, node,* dan *hardware.*

b. Sistem *client*/server

c. Sistem terdistribusi murni

d. Rekayasa ulang aplikasi

Tabel II.3.
Simbol-simbol yang ada pada Diagram *deployment*

Simbol	Deskripsi
Package 	*package* merupakan sebuah bungkusan dari suatu atau lebih *node*
Node 	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika didalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikut sertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen
Kebergantungan / *dependency* 	Kebergantungan antar *node,* arah panah mengarah pada *node* yang dipakai

Link _____	Relasi antar *node*

Sumber: Rosa A. S dan M. Shalahuddin (2015:154)

2.2.2.4. *Sequence Diagram*

Diagram *sequence* menggambarkan kelakuan objek pada *use case* dengan mendefinisikan waktu hidup objek dan *massage* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu, membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case* (Rosa dan Shalahudin, 2015:165).

Banyaknya diagram *sequence* yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

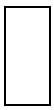
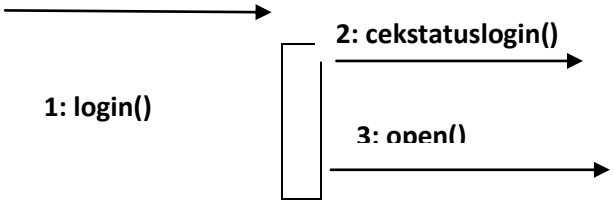
Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

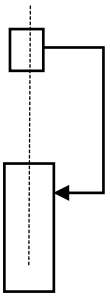
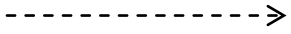
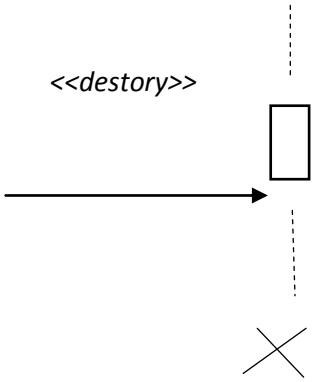
Semua metode didalam kelas harus ada didalam diagram kolaborasi atau sekuen,jika tidak ada berarti perancangan metode didalam kelas itu kurang baik.

Hal ini dikarenakan ada metode yang tidak dapat di pertanggung jawabkan kegunannya.

Tabel II.4.

Simbol-simbol yang ada pada Diagram Sequence

Simbol	Deskripsi
Aktor / *actor*  nama actor atau nama_aktor tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat diluar sistem informasi itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Garis hidup / *lefeline* 	Menyatakan kehidupan suatu objek.
Objek nama objek : nama_kelas	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhububung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka cekstatuslogin() dan open() dilakukan didalam metode login() Aktor tidak memiliki waktu aktif.

Pesan tipe call 1: nama_metode() 	Menyatakan suatu objek memanggil operasi/ metode yang ada pada objek lain atau dirinya sendiri  1: nama_metode() Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/met ode karena ini memanggil operasi/metode maka yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe return  1: keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghassilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
pesan tipe destroy  	Menyatakan suatu objek mengakhiri hidup objek objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka aa destroy

Sumber: Rosa A. S dan M. Shalahuddin (2015:165)

2.2.3. *Entity Relationship Diagram* (ERD)

Menurut Fatta (2007:121) memberikan batasan bahwa, "*Entity Relationship Diagram* (ERD) adalah gambar atau diagram yang menujukan informasi dibuat, disimpan, dan digunakan dalam sistem bisnis".

Entity Relationship Diagram (ERD) adalah memiliki beberapa aliran notasi atau komponen (Rosa dan Shalahuddin, 2015:50).

Dari kutipan diatas dapat disimpulkan bahwa *Entity Relationship Diagram* (ERD) adalah diagram yang akan menunjukan informasi yang akan dibuat, disimpan dan digunakan dalam sistem bisnis dan memiliki beberapa notasi ataupun komponen.

Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi adalah sebagai berikut:

1. Entitas

   Entitas merupakan data inti yang akan disimpan pada basis data, benda yang memiliki data dan harus disimpan datanya agar dapat diaskes oleh aplikasi komputer, penamaan entitas biasanya lebih kekata benda dan belum merupakan nama table.

2. Atribut

   field atau kolom data yang butuh disimpan dalam suatu entitas.

3. Atribut Kunci Primer

   Field atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci askes *record* yang diinginkan, biasanya berupa id, kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama) .

4. Atribut Multinilai (*Multivalue*)

   Field atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.

5. Relasi

   Relasi yang berhubungan antara entitas; biasanya diawali dengan kata kerja.

6. Asosiasi (*Accociation*)

   Pendukung antara relasi dan entitas dimana kedua ujungnya memiliki *multiplicity* kemugkinan jumlah pemakaian, kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas lain disebut dengan kardinallitas. Misalnya ada kardinalitas 1 ke N atau sering disebut dengan *one to many* menghubungkan entitas A dan entitas B.

2.2.4. *Logikal Record Structure* (LRS)

Hasugian dan Shidiq (2012:608) mendefinisikan:

> *Logikal Record Structure* (LRS) merupakan sebuah model sistem yang digambarkan dalam sebuah *diagram*-ER akan mengikuti pola dan aturan pemodelan tertentu dalam kaitannya dengan konverensi ke LRS, maka perubahan yang terjadi setiap entitas akan diubah kebentuk kotak dan sebuah atribut relasi disatukan dalam sebuah kotak bersam entitas jika dihubungkan yang terjadi dalam *diagram*–ER 1:M (relasi bersatu dengan *cardinality* M) atau tingkat hubungan 1:1 (relasi bersatu denga *cardinality* yang paling membutuhkan referensi), sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubungannya M:M (*many to many*) dan memiliki *foreign key* sebagai *primery key* yang diambil dari kedua entitas yang sebelumnya saling berhubungan.

Menurut buku yang sama *Logikal Record Structure (*LRS) terdiri dari link-link di antara tipe *record, link* ini menunjukan arah dari satu tipe *record* lainnya.

2.2.5. Basis Data/*Database*

Basis data adalah media untuk menyimpan data agar dapat diaskes dengan mudah dan cepat, Rosa dan Shalahuddin (2015:43).

Sedangkan menurut Zaki (2008:94) memberikan batasan bahwa, "*database* adalah kemudahan dalam menyimpan dan menampilkan data karena dalam bentuk tabel".

Dari kutipan diatas dapat disimpulkan bahwa *database* adalah suatu tempat untuk penyimpanan dan menampilkan data dalam bentuk tabel agar dapat diakses dengan mudah dan cepat.

2.2.6. *XAMPP*

Menurut Aryanto (2016:4) memberikan batasan bahwa, "*Xampp* merupakan sebuah aplikasi perangkat lunak pemrograman dan *database* yang didalamnya terdapat berbagai macam aplikasi pemrograman seperti *Apache HTTP server, MySQL database*, bahasa pemrograman *PHP* dan *Perl*".

Sedangkan menurut Wahana Komputer (2015:55) memberikan batasan bahwa "*XAMPP* merupakan singkatan dari X (empat sistem operasi apapun), *Apache, MySQL, PHP, Perl* dan merupakan *tool* yang menyediakan paket perangkat lunak dalam satu buah paket".

Dari kutipan diatas dapat disimpulkan bahwa *XAMPP* merupakan aplikasi perangkat lunak yang dibutuhkan dalam sebuah pemograman database.

2.2.7. *PHPMyAdmin*

Menurut Sibero (2013:376) memberikan batasan bahwa, "*PHPMyadmin* adalah aplikasi *web* yang dibuat oleh admin digunakan untuk administrasi database".

Sedangkan menurut Putri dalam buku Agung, Deddy dan Tuti (2015:2) Mengemukakan bahwa " *PHPMyAdmin* merupakan aplikasi yang dapat digunakan untuk membuat database, pengguna (user), memodifikasi tabel, maupun mengirim database secara cepat dan mudah tanpa harus menggunakan perintah (*command*) SQL".

Dari pengertian diatas dapat disimpulkan bahwa *PHPMyAdmin* merupakan salah satu aplikasi yang digunakan untuk membuat database.

2.2.8. *NetBeans*

NetBeans adalah salah satu aplikasi *Integrated Development environment* (IDE) yang digunakan oleh *developer software* komputer untuk menulis, meng-*compile*, mencari kesalahan, dan untuk menyadarkan program, Wahana Komputer (2015:20).

Menurut Nofriadi (2015:4) "Netbeans merupakan sebuah aplikasi *Integrated Development Environment* (IDE) yang berbasiskan Java dari *Sun Microsystems* yang berjalan diatas *swing* dan banyak digunakan sekarang sebagai editor untuk berbagai bahasa pemrograman".

Dari beberapa definisi diatas dapat disimpulkan bahwa NetBeans adalah sebuah aplikasi pemrograman.

2.2.9. iReport

iReport adalah laporan yang diperlukan dalam suatu aplikasi sistem informasi. *Tools* yang cukup dikenal untuk membuat laporan keuangan yaitu *Crystal Report*. *iReport* termasuk kedalam *report desiger visual* yang dibangun pada *JasperReports* yang mengisi kekurangan itu. *iReport* bersifat mudah digunakan pembuatan laporan visual atau desainer untuk *JasperReports*, dan penulisan dalam kitab java. Sebagai alternatif, terdapat *tools iReports* (dengan *library JasperReport*) yang dapat pula membantu dalam pembuatan laporan. *Library JasperReport* sendiri merupakan Java *Library* (JAR) yang bersifat *open* dan dirancang untuk menambah kemampuan pelaporan (*reporting capabilities*) pada aplikasi java, Wahana Komputer (2015:38).

Menurut Supriyanto (2010:182) Menyatakan bahwa " *iReport* merupakan software yang digunakan untuk membuat dan mendesain template *report* laporan dengan tampilan GUI".

Dari beberapa definisi diatas dapat disimpulkan bahwa iReport merupakan software yang digunakan pada aplikasi java untuk mendesain dan mencetak laporan.