BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

Secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur-unsur, komponen atau variabel yang terorganisir, dan saling berinteraksi, saling ketergantungan satu sama lain dan terpadu. Suatu sistem pada dasarnya adalah kelompok unsur yang berhubungan erat satu dengan yang lainnya, yang berfungsi bersama-sama untuk mencapai tujuan tertentu.

Pendapat Mustakini (2010:2) "Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu".

Indra dalam Muslihudin dan Oktafianto (2016:2) mengemukakan "sistem adalah sekumpulan elemen atau subsistem yang saling berhubungan satu dengan yang lain membentuk satu kesatuan untuk melaksanakan suatu fungsi guna mencapai suatu tujuan".

Berdasarkan dari dua definisi di atas, maka dapat disimpulkan bahwa sistem adalah sekumpulan komponen-komponen yang berinteraksi satu sama lain untuk mencapai satu tujuan. Berdasarkan beberapa pengertian diatas mengenai sistem, maka suatu sistem seperti sistem informasi akan lebih mudah dipahami dan dirancang jika didekati dengan pendekatan komponen. Oleh karena itu, dalam hal ini akan menggunakan pendekatan komponen untuk menjelaskannya.

2.1.1. Klasifikasi Sistem

Pengklasifikasian sistem menurut Azhar Susanto dalam Novianti (2012:9) adalah seperti yang terlihat dalam tabel berikut:

Tabel II.1.
Klasifikasi Sistem

KRITERIA	KLASIFIKASI	
Lingkungan	Sistem Terbuka	Sistem Tertutup
Asal Pembuatan	Buatan Manusia	Buatan Allah/alamiah
Keberadaannya	Sistem berjalan	Sistem Konsep
Kesulitan	Sulit/Kompleks	Sederhana
Output/Kinerjanya	Dapat dipastikan	Tidak dapat dipastikan
Waktu keberadaannya	Sementara	Selamanya
Wujundya	Abstrak	Ada secara fisik
Tingkatannya	Sub sistm/Sistem	Super sistem
Flesibilitas	Bisa berinteraksi	Tidak dapat beradaptasi

Sumber: Azhar Susanto dalam Novianti (2012:9)

1. Sistem Terbuka dan Sistem Tertutup

   Sebuah sistem dikatakan terbuka bila aktifitas di dalam sistem tersebut dipengaruhi oleh lingkungannya, sedangkan bila tidak terpengaruh oleh perubahan yang terjadi di lingkungannya, maka disebut sistem tertutup

2. Sistem Buatan Manuasia dan Sistem Buatan Allah SWT /Alamiah

   Suatu sistem bila diklasifikasikan berdasarkan asalnya, maka ada sistem buatan manuasia seperti organisasi perusahaan dan lain-lain. Dan sistem buatan Allah SWT (alamiah) seperti manusia, pohon-pohon dan lain-lain.

3. Sistem Berjalan dan Sistem Konsep

   Suatu sistem yang belum diterapkan disebut sistem konseptual/konsep. Sedangkan sistem konseptual yang dapat diterima oleh penggunanya untuk menunjang operasi sehari-hari maka disebut sistem berjalan.

4. Sistem Sederhana dan Kompleks

   Sistem sederhana merupakan sebuah sistem yang terbentuk dari sedikit tingkatan dan komponen atau sub sistem serta hubungan yang sangat sederhana. Sistem kompleks adalah sebuah sistem yang terdiri dari banyak komponen dan tingkatan yang dihubungkan dalam berbagai cara yang berbeda.

5. Sistem yang kinerjanya dapat dan tidak dapat dipastikan

   Sistem yang dapat dipastikan kinerjanya artinya dapat ditentukan pada saat sistem tersebut akan dan sedang dibuat misalnya sistem listrik dimana kita tinggal. Sedangkan sistem yang tidak dapat dipastikan artinya tidak dapat ditentukan dari awal, yakni tergantung kepada situasi yang dihadapi misalnya organisasi perusahaan.

6. Sistem Sementara dan Selamanya

   Sistem sementara artinya sistem yang digunakan hanya dalam periode tertentu misalnya sistem pemilu. Sedangkan sistem selamanya artinya sistem tersebut digunakan untuk waktu yang tidak ditentukan misalnya sistem lalu lintas.

7. Sistem yang ada secara Fisik dan Abstrak

   Sistem yang ada secara fisik artinya kita dapat menyentuhnya atau merasakannya. Sedangkan sistem abstrak sebaliknya, yakni tidak dapat disentuh.

8. Sistem, sub sistem dan super sistem

   Berdasarkan tingkatannya/hirarki sebuah sistem bisa merupakan komponen dari sistem yang lebih besar. Sub sistem adalah sistem yang lebih kecil yang ada dalam sebuah sistem. Super sistem adalah sistem yang sangat besar dan sangat kompleks.

9. Sistem yang bisa dan tidak bisa beradaptasi

   Sistem yang bisa beradaptasi adalah sistem yang memiliki kemampuan untuk beradaptasi terhadap setiap pengaruh yang diakibatkan oleh perubahan yang terjadi di lingkungannya. Kebalikannya disebut sistem yang tidak bisa beradaptasi.

2.1.2. Sistem Informasi

Pendapat Sutarman (2012:13) “Sistem informasi adalah sistem yang dapat didefinisikan dengan mengumpulkan, memperoses, menyimpan, menganalisis, menyebarkan informasi untuk tujuan tertentu. Seperti sistem lainnya, sebuah sistem informasi terdiri atas input (data, instruksi) dan output (laporan, kalkulasi)”.

Tantra (2012:2) mengatakan bahwa "sistem informasi adalaha cara yang teroganisisir untuk mengumpulkan, memasukan dan memproses data dan menyimpannya, mengelola, mengontrol, dan melaporkannya sehingga dapat mendukung perusahaan atau organisasi untuk mencapai suatu tujuan".

Berdasarkan beberapa pendapat yang dikemukakan di atas dapat ditarik kesimpulan bahwa, "Sistem informasi adalah sebuah sistem yang terdiri dari pengumpulan, pemasukan, pemrosesan data, penyimpanan, pengolahan, pengendalian dan pelaporan sehingga tercapai sebuah informasi yang mendukung pengambilan keputusan didalam suatu organisasi untuk dapat mencapai sasaran dan tujuannya".

2.1.3. Tujuan Sistem Informasi

Menurut Mustakini (2010:13), tujuan dari sistem informasi adalah menghasilkan informasi (*Information*) dari bentuk data yang diolah menjadi bentuk yang berguna bagi para pemakainya.

Tujuan sistem informasi terdiri dari Kegunaan (*Usefulness*), Ekonomi (*Economic*), Keandalan (*Realibility*), Pelayanan Langganan (*Customer Service*), Kesederhanaan (*Simplicity*), dan Fleksibilitas (*Fleksibility*).

1. Kegunaan (*Usefulness*)

   Sistem harus menghasilkan informasi yang akurat, tepat waktu, dan relevan untuk pengambilan keputusan manajemen dan personil operasi di dalam organisasi.

2. Ekonomi (*Economic*)

   Semua bagian komponen sistem termasuk laporan-laporan, pengendalian-pengendalian, mesin-mesin harus menyumbang suatu nilai manfaat setidak-tidaknya sebesar biaya yang dibutuhkan.

3. Keandalan (*Realibility*)

   Keluaran sistem harus mempunyai tingkatan ketelitian yang tinggi dan sistem itu sendiri harus mampu beroperasi secara efektif bahkan pada

waktu komponen manusia tidak hadir atau saat komponen mesin tidak beroperasi secara temporer.

4. Pelayanan Langganan (*Customer Service*)

   Sistem harus memberikan pelayanan dengan baik atau ramah kepada para pelanggan. Sehingga sistem tersebut dapat diminati oleh para pelanggannya.

5. Kesederhanaan (*Simplicity*)

   Sistem harus cukup sederhana sehingga terstruktur dan operasinya dapat dengan mudah dimengerti dan prosedurnya mudah diikuti.

6. Fleksibilitas (*Fleksibility*)

   Sistem harus cukup fleksibel untuk menangani perubahan-perubahan yang terjadi, kepentingannya cukup beralasan dalam kondisi dimana sistem beroperasi atau dalam kebutuhan yang diwajibkan oleh organisasi.

2.1.4. Pengertian Sistem Informasi Akuntansi

Menurut Mujilan (2012:3) Sistem informasi akuntansi adalah kumpulan sumberdaya, seperti manusia dan peralatan, yang diatur untuk mengubah data menjadi informasi. Informasi ini dikomunikasikan kepada beragam pengambil keputusan. SIA mewujudkan perubahan ini secara manual atau terkomputerisasi.

SIA juga merupakan sistem yang paling penting di organisasi dan merubah cara menangkap, memproses, menyimpan, dan mendistribusikan informasi. Saat ini, digital dan informasi online semakin digunakan dalam sistem informasi akuntansi. Organisasi perlu menempatkan sistem di depan, dan mempertimbangkan baik segi sistem ataupun manusia sebagai faktor yang terkait ketika mengatur sistem informasi akuntansi.

SIA pada umumnya meliputi beberapa siklus pemrosesan transaksi :

1. Siklus pendapatan. Berkaitan dengan pendistribusian barang dan jasa ke entitas lain dan pengumpulan pembayaran-pembayaran yang berkaitan.
2. Siklus pengeluaran. Berkaitan dengan perolehan barang jasa dari entitas lain dan pelunasan kewajiban yang berkaitan.
3. Siklus produksi. Berkaitan dengan pengubahan sumberdaya menjadi barang dan jasa.
4. Siklus keuangan. Kejadian-kejadian yang berkaitan dengan perolehan dan manajemen dana-dana modal, termasuk kas.

2.1.5. Penerimaan Kas

Menurut Mulyadi (2008:456) "Sistem akuntansi penerimaan kas adalah satu jaringan prosedur yang dibuat menurut pola yang terpadu untuk melaksanakan kegiatan penerimaan kas dari penjualan rutin dan tidak rutin berdasarkamm ketentuan-ketentuan dari perusahaan yang bersangkutan."

Abdul Halim dalam Novianti (2012:25) menyimpulkan bahwa : Sistem akuntansi penerimaan kas meliputi serangkaian proses baik manual maupun terkomputerisasi, mulai dari pencatatan, penggolongan, peringkasam transaksi dan kejadian keuangan hingga pelaporan keuangan dalam rangka pertanggung jawaban pelaksanaan APBD yang berkaitan dengan penerimaan kas.

Berdasarkan beberapa pengertian di atas yang dimaksud sistem akuntansi kas masuk yaitu suatu jaringan prosedur yang menangani suatu peristiwa atau kejadian yang mengakibatkan terjadinya penambahan uang dalam kas yang berasal dari penjualan tunai, maupun piutang yang melibatkan bagian-bagian yang saling berkaitan satu sama lain.

2.1.6. Pengeluaran Kas

Menurut Mulyadi (2008:509) “Sistem akuntansi pengeluaran kas pada umumnya didefinisikan sebagai organisasi formulir, catatan dan laporan yang dibuat untuk melaksanakan kegiatan pengeluaran baik dengan cek maupun dengan uang tunai untuk mempermudah setiap pembiayaan pengelolaan perusahaan.”

Marshall B Romney dalam Novianti (2012:31-32) mengemukakan bahwa “Sistem akuntansi pengeluaran kas terdapat sistem akuntansi pokok yang bisa digunakan dalam sistem akuntansi pengeluaran kas yaitu sistem akuntansi pengeluaran kas dengan cek dan sistem akuntansi pengeluaran kas dengan uang tunai melalui dana kas kecil”.

Berdasarkan dari beberapa pengertian di atas dapat disimpulkan bahwa sistem akuntansi kas keluar pada umumnya didefinisikan sebagai organisasi formulir, catatan, dan laporan yang dibuat untuk melaksanakan kegiatan pengeluaran baik dengan cek maupun dengan uang tunai untuk mempermudah setiap pembiayaan pengelolaan perusahaan atau institusi.

2.1.7. Karakteristik OOAD

Mathiassen dalam Andria (2016:21) Dalam pendekatan berorientasi objek ada 4 pilar utama yang harus dipahamai dalam pendekatan berorientasi objek yaitu karakteristik. Karakteristik (ciri) suatu program termasuk OOAD/OOP, apabila terdapat abstraksi, pembungkusan (*encapsulation*), *polymorphisme*, dan turunan (*inheritance*).

1. *Abstraction*

Kemampuan untuk menjadikan dalam bentuk yang lebih sederhana. Hal ini juga dikenal dalam metodologi pendekatan struktur yaitu dekomposisi seperti menyerderhanakan suatu sistem dalam bentuk *Context* Diagram.

2. *Encapsulation*

Merupakan suatu karakteristik OOAD dimana program terbungkus (jadi satu) data dan perilaku, artinya lebih memperhatikan aspek internal daripada aspek eksternal. Contoh: dalam program terdapat tombol *button close* didalamnya ada *method system.exit* (0) untuk keluar dari sistem java. Berbeda dengan metodologi terdahulu, metodologi ini menggabungkan atribut dan fungsi / proses kedalam suatu objek yang disebut dengan *encapsulation*. Setiap objek dapat "menyembunyikan" kompleksitasnya dan berhubungan dengan objek lain dengan mengirim "pesan/*message*" yang dapat dikenal dan diproses oleh objek penerima. Contoh: Pada dunia nyata, seorang ibu rumah tangga menanak nasi dengan menggunakan *rice cooker*, ibu tersebut menggunakannya hanya dengan menekan tombol. Tanpa harus tahu bagaimana proses itu sebenarnya terjadi. Disini terdapat penyembunyian informasi milik rice cooker, sehingga tidak perlu diketahui seorang ibu. Dengan demikian menanak nasi oleh si ibu menjadi sesuatu yang menjadi dasar bagi konsep information hiding.

3. *Polymorphisme*

Dengan kata lain suatu mekanisme yang memungkinkan suatu objek memiliki semua atau sebagian definisi dari objek induk. Menurut Bambang *Polymorphism* berasal dari kata Poly yang artinya banyak dan morph yang artinya bentuk. Jadi *polymorphism* adalah kemampuan suatu atribut atau method dapat

berubah dalam berbagai bentuk dalam implementasi. Contoh Pada obyek mobil, walaupun minibus dan truk merupakan jenis obyek mobil yang sama, namun memiliki juga perbedaan. Misalnya suara truk lebih keras dari pada minibus, hal ini juga berlaku pada obyek anak (*child*) melakukan metoda yang sama dengan algoritma berbeda dari obyek induknya. Hal ini yang disebut *polymorphism*, teknik atau konsep dasar lainnya adalah ruang lingkup/pembatasan. Artinya setiap obyek mempunyai ruang lingkup kelas, atribut, dan metoda yang dibatasi.

4. *Inheritance*

Merupakan suatu karakteristik OOAD di mana suatu kelas (*parent/base class*) dapat diturunkan ke kelas lain (*child/derived class*), sehingga kelas anak dapat memiliki data atau perilaku kelas orang tuanya. Contoh dengan beberapa buah mobil yang mempunyai kegunaan yang berbeda-beda. Ada mobil bak terbuka seperti truk, bak tertutup seperti sedan dan minibus. Walaupun demikian obyek-obyek ini memiliki kesamaan yaitu teridentifikasi sebagai obyek mobil, obyek ini dikatakan obyek induk (*parent*). Sedangkan minibus obyek anak (*child*), berarti semua operasi yang berlaku pada mobil berlaku pada minibus.

2.2. Peralatan Pendukung (*Tools System*)

Peralatan pendukung (*Tool System*) merupakan alat yang dapat digunakan untuk menggambarkan bentuk logika model dari suatu sistem.

1.2.1. *Unified Modelling Language* (UML)

Masing-masing diagram UML didesain agar para pengembang dan pengguna dapat melihat sistem dan *software* dari perspektif yang berbeda dan dalam berbagai level abstraksi, UML terdiri dari banyak diagram diantaranya:

2.2.2. *Use Case Diagram*

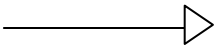
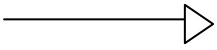
Menurut Sukamto dan Shalahudin (2015:155) *use case* atau diagram *use case* merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat.*use case* mendeskrifsikan sebuah interaksi antara satu atau lebih aktor digunakan untuk mengetahui fungsi apa saja yang ada pada sebuah sistem informasi apa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama di definisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu mendefenisikan apa yang disebut aktor dan *use case*.

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

Tabel II.2.
Use Case Diagram

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 di awal frase nama *use case*.

aktor/*actor*  **nama actor**	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; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Asosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau use case memiliki interaksi dengan aktor.
Ekstensi/ *extend* <<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 *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang di tambahkan.
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan/*include*/*uses* <<include>>  <<uses>> 	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

Sumber: Sukamto dan Shalahudin (2015)

2.2.3. *Activity Diagram*

Sukamto dan Shalahudin (2015:161) 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

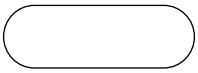
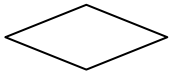
sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

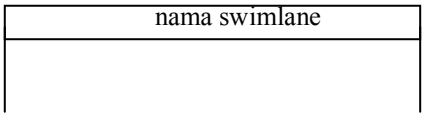
1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
4. Rancangan menu yang ditampilkan pada perangkat lunak

Berikut adalah simbol yang ada pada diagram aktivitas:

Tabel II.3.
Activity Diagram

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/ *decision* 	Asosiasi percabangan dimana jika ada pilihhan lain aktivitas lebih dari satu
Penggabungan/ *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilaakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir

Lanjutan

Swimlane  	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

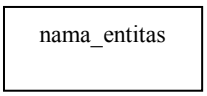
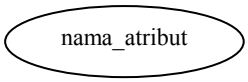
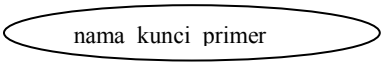
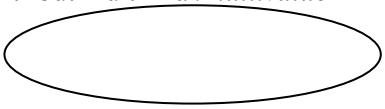
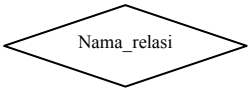
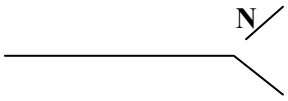
Sumber: Sukamto dan Shalahudin (2015)

2.2.4. *Entity Relationship Diagram* (ERD)

Menurut Sukamto dan Shalahudin (2015:50) Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagram* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data rasional. Sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oelh Peter Chen), Barker (dikembangkan oleh Richard Barker, Ian Palmer, Harry Ellis), *notasi Crow's Fot*, dan beberapa notasi lain. Namun banyak digunakan adalah notasi dari Chen. Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi Chen

Tabel II.4.
***Entity Relationship Diagram* (ERD)**

Simbol	Deskripsi

Entitas/*entity* 	Entitas merupakan data yang akan disimpan; bakal tabel pada basis data; benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih kekata benda dan belum merupakan nama tabel.
Atribut 	*Field* atau kolom data yang butuh disimoan dalam suatu entitas
Atribut kunci primer 	*field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *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).
Atribut multinilai/*multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.
Relasi 	Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja
Asosiasi/*association* 	Penghubung antara relasi dan entitas dimana kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas lain disebut dengan kardinalitas.

Sumber: Sukamto dan Shalahudin (2015)

2.2.5. *Logical Record Structure* (LRS)

Menurut Frieyadie (2007:13) “LRS merupakan hasil dari pemodelan *Entity Relational Ship* (ER) beserta atributnya sehingga bisa terlihat hubungan-hubungan antar entitas”. Dalam pembuatan LRS terdapat 3 hal yang dapat mempengaruhi yaitu:

1. Jika tingkat hubungan *(cardinality*) satu pada satu (*one-to-one*), maka di gabungkan dengan entitas yang lebih kuat (strong entity), atau digabungkan dengan entitas yang memiliki atribut yang lebih sedikit.
2. Jika tingkat hubungan (*cardinality*) satu pada banyak (*one-to-many*), maka hubungan relasi atau digabungkan dengan entitas yang tingkat hubungannya banyak.
3. Jika tingkat hubungan (*cardinality*) banyak pada banyak (*many-to-many*), maka hubungan relasi tidak akan digabungkan dengan entitas manapun,melainkan menjadi sebuah LRS.
4. dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubunganya M:M (*many to many*) dan memiliki *foreign key* sebagai *primary key* yang diambil dari kedua entitas yang sebelumnya saling berhubungan.

2.2.6. *Sequence Diagram*

Sukamto dan Shalahuddin (2015:165) diagran *sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibar dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang

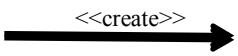
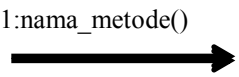
diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case.*

Banyaknya diagram sekuen 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.

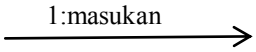
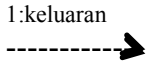
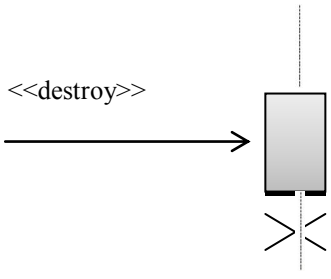
Berikut adalah simbol-simbol yang ada pada diagram sekuen:

Tabel II.5.
Simbol *Sequence Diagram*

Simbol	Deskripsi
Aktor  **nama aktor** 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 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.
Garis hidup/ lifeline	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 terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya.

Pesan tipe create  	Menyatakan suatu objek membuat objek 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 mengarahkan pada objek yang memiliki operasi /metode maka operasi/metode yang dipanggil sesuai dengan kelas objek yang berinteraksi.

Lanjutan

Pesan tipe send  Pesan tipe return  	Menyatakan bahwa suatu objek yang mengirimkan data masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim. Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe destroy  	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panak mngarah pada objek yang di akhiri,sebaiknya jika ada create maka ada destroy

Sumber: Sukamto dan Shalahudin (2015)

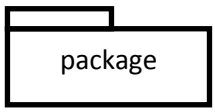
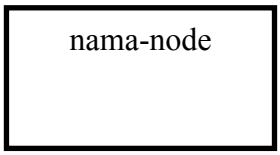
2.2.7. *Deployment Diagram*

Menurut Sukamto dan Shalahuddin (2015:154) Diagram *deployment* atau *deployment* diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut:

- Sistem tambahan (*embedded system)* yang menggambarkan rancangan *device*, *node,* dan *hardware.*
- Sistem *client/server*

Berikut adalah simbol-simbol yang ada pada diagram *deployment* dapat dilihat pada Tabel II.6:

Tabel II.6.
Simbol *Deployment* Diagram

Simbol	Deskripsi
Package 	*Package* merupakan sebuah bungkusan dari satu atau lebih *node.*
Node 	Biasanya mengacu pada perangkat keras(*hardware),* perangkat lunak yang dibuat sendiri(*software*), jika didalam node di sertakan 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: Sukamto dan Shalahudin (2015)

2.2.8. Netbeans IDE 8.1

Nofriadi (2015:4) *Netbeans* merupakan sebuah aplikasi *Integrated Development Environment* (IDE) yang berbasiskan *Java* dari *Sun Microsystems* yang berjalan di atas *swing* dan banyak digunakan sekarang sebagai editor untuk berbagai bahasa pemrograman. Sampai sekarang, *Netbeans* sudah sampai ke versi 8.0. Pada Netbeans, kita bisa membuat bahasa pemrograman *Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure*. *Swing* merupakan teknologi *Java* untuk pengembangan aplikasi desktop yang bisa dijalankan di berbagai sistem operasi, seperti *windows*, *linux*, *Mac OS X*, dan *Solaris.*

2.2.9. MySQL

Menurut Wahana Komputer (2010:5) MySQL adalah program database yang mampu mengirim dan menerima data dengan sangat cepat dan multi *user*. MySQL memiliki dua bentuk lisensi, yaitu *free sofware* dan *shareware*.

2.2.10. XAMPP

Wahana Komputer dan Andi (2014:55) mengemukakan bahwa "XAMPP adalah tool yang menyediakan paket perangkat lunak dalam salah satu buah paket".