BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

II.1.1. Pengertian Sistem

Menurut McLeod (2012 : 1) sistem adalah sekelompok elemen-elemen yang terintegrasi dengan tujuan yang sama untuk mencapai tujuan. Organisasi terdiri dari sejumlah sumber daya manusia, material, mesin, uang, dan informasi. Sumber daya tersebut bekerja sama menuju tercapainya suatu tujuan tertentu yang ditentukan oleh pemilik atau manajemen.

II.1.2. Klasifikasi Sistem

Sistem dapat diklasifikasikan dari beberapa sudut pandang. Klasifikasi sistem tersebut diantaranya : sistem abstrak (*abstract system*), sistem fisik (*physical system*), sistem tertentu (*deterministic system*), sistem tak tentu (*probabilistik system*), sistem tertutup (*close system*), dan sistem terbuka (*open system*).

a. Sistem tak tentu (*probabilistik system*), adalah suatu sisten yang kondisi masa depanya tidak dapat diprediksi karena mengandung unsur probabilitas. Sistem arisan marupakan contoh probabilistic system karena sistem arisan tidak dapat diprediksi dengan pasti.
b. Sistem abstak (*abstract system*), adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik.

c. Sistem fisik (*physical system*), adalah sistem yang ada secara fisik. Sistem komputer, sistem akuntansi, sistem produksi, sistem sekolah, dan sistem transportasi merupakan contoh physical system.

d. Sistem tertentu (*deterministic system*), adalah sistem yang beroperasi dengan tingkah laku yang dapat diprediksi, interaksi antara bagian dapat dideteksi dengan pasti sehingga keluarannya dapat diramalkan. Sistem komputer sudah diprogramkan, merupakan contoh deterministic system karena program komputer dapat diprediksi dengan pasti.

e. Sistem tertutup (*close system*), sistem yang tidak bertukar materi, informasi, atau energi dengan lingkungan. Sistem ini tidak berinteraksi dan tidak dipengaruhi oleh lingkungan. Sistem ini tidak berinteraksi dan tidak dipengaruhi oleh lingkungan, misalnya : reaksi kimia dalam tabung yang terisolasi.

f. Sistem terbuka (*open system*), adalah sistem yang berhubungan dengan lingkungan dan dipengaruhi oleh lingkungan. Sistem perdagangan menurupakan contoh open system, karena dapat dipengaruhi oleh lingkungan.

II.2. Pengertian Informasi

Menurut McLeod (2012 : 9) informasi *(information)* adalah data yang diolah menjadi bentuk lebih berguna dan lebih berarti bagi yang menerimanya. Informasi juga disebut data yang diproses atau data yang memiliki arti. Informasi merupakan data yang telah diproses sedemikian rupa sehingga meningkatkan pengetahuan seseorang yang menggunakan. Para pembuat keputusan memahami

bahwa informasi menjadi faktor kritis dalam menentukan kesuksesan atau kegagalan dalam suatu bidang usaha. Sistem apapun tanpa ada informasi tidak akan berguna, karena sistem tersebut akan mengalami kemacetan dan akhirnya berhenti. Informasi dapat berupa data mentah, data tersusun, kapasitas sebuah saluran informasi, dan sebagainya.

II.2.1. Kualitas Informasi

Jogiyanto (2012 : 9) kualitas dari informasi (*quality of information*) tergantung dari tiga hal yaitu : *accurate*, *timeliness*, dan *relevance*.

a. Relevan (*relevance*), berarti informasi tersebut mempunyai manfaat untuk pemakainnya dan relevansi informasi untuk tiap-tiap orang akan berbeda-beda.
b. Tepat waktu (*timelines*), berarti informasi tersebut datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi, karena informasi merupakan landasan di dalam pengambilan keputusan.
c. Akurat (*accuracy*), berarti informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerima informasi kemungkinan banyak terjadi gangguan (*noise*) yang dapat merusak informasi.

II.3. Pengertian Sistem informasi

Menurut O'Brian (2011 : 17) sistem informasi (*information system*) merupakan kombinasi teratur dari orang-orang, perangkat keras (*hardware*), perangkat lunak (*software*), jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah dan menyebarkan informasi dalam sebuah organisasi. Orang tergantung pada sistem informasi untuk berkomunikasi antara satu sama lain dengan menggunakan berbagai jenis alat fisik, perintah dan prosedur pemrosesan informasi, saluran telekomunikasi atau jaringan, dan data yang disimpan atau sumber daya data.

Menurut Jogiyanto (2012 : 17) sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi serta menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. Sistem informasi juga dapat didefenisikan sebagai suatu sistem yang dibuat oleh manusia yang terdiri dari komponen-komponen dalam organisasi untuk menyajikan informasi.

II.4. Pengertian Piutang

Piutang adalah klaim perusahaan atas uang, barang, atau jasa kepada pihak lain akibat di masa lalu.

Piutang Usaha, yaitu piutang yang timbul dari penjualan barang atau jasa yang dihasilkan perusahaan. Dalam kegiatan normal perusahaan, piutang usaha biasanya akan dilunasi dalam tempo kurang dari satu tahun, sehingga piutang usaha dikelompokkan ke dalam kelompok aset lancar.

Istilah piutang mencakup seluruh uang yang diklaim terhadap entitas lain, termasuk orang, perusahaan, dan organisasi lain.

II.4.1. Mengestimasi Jumlah Piutang Tak Tertagih

Seperti yang kita bahas sebelumnya, metode penyisihan mengestimasi jumlah beban piutang tak tertagih pada akhir periode. Bagaimana cara mengestimasi jumlah piutang tak tertagih? Estimasi jumlah piutang tak tertagih pada akhir periode fiskal dibuat berdasarkan pengalaman masa lalu dan prakiraan masa depan. Saat perekonomian secara umum berjalan baik, estimasi jumlah piutang tak tertagih biasanya lebih kecil dibandingkan saat perekonomian lesu.

Terdapat dua metode yang biasa digunakan dalam mengestimasi jumlah piutang tak tertagih pada akhir periode. Estimasi tersebut dapat dibuat berdasarkan (1) persentase penjualan atau (2) analisis piutang (Pengantar Akuntansi – Adaptasi Indonesia 2009 : 444).

Penulis dalam hal ini membahas estimasi jumlah piutang tak tertagih berdasarkan analisis piutang. Estimasi berdasarkan analisis piutang, semakin lama

piutang tidak dilunasi, semakin kecil kemungkinan piutang akan tak tertagih. Oleh karena itu, kita dapat mengestimasi jumlah piutang tak tertagih dengan melihat berapa lama piutang tertentu belum dilunasi. Untuk keperluan ini, kita dapat melakukan proses yang disebut menghitung umur piutang *(again the receivable)* piutang (Pengantar Akuntansi – Adaptasi Indonesia 2009 : 446).

Piutang dihitung umurnya dengan cara menyiapkan daftar yang mengklasifikasikan piutang setiap pelanggan berdasarkan tanggal jatuh temponya. Jumlah hari suatu piutang yang telah lewat jatuh tempo adalah selisih antara tanggal piutang jatuh tempo dengan tanggal saat daftar umur piutang disiapkan piutang (Pengantar Akuntansi – Adaptasi Indonesia 2009 : 446).

Estimasi persentase piutang tak tertagih meningkat seiring semakin lamanya piutang tersebut lewat jatuh tempo. Untuk piutang yang belum lewat jatuh tempo, persentasinya adalah 5%, sementara untuk piutang yang lebih dari 365 hari telah lewat jatuh tempo adalah 80%. Total dari jumlah seluruh kelompok umur ini adalah saldo akhir periode yang diharapkan untuk penyisihan piutang tak tertagih.

II.5. *Microsoft Visual Basic.Net*

Menurut Rahmat Priyanto (2009 : 1) Visual Basic 2008 merupakan satu paket bahasa pemrograman dari *Visual Studio* 2008. Banyak fasilitas yang akan kita dapatkan melalui rilis *Visual Basic* versi ini. *Visual Studio* 2008 sendiri merupakan sebuah *software* untuk membuat aplikasi *Windows*, jadi melalui *software* ini kita bisa membuat sebuah aplikasi seperti aplikasi *database*, aplikasi *inventory*, dan sebagainya. Kebanyakan orang lebih suka menyebut sebuah aplikasi sebagai sebuah program atau *software*, padahal ketiga istilah ini memiliki arti yang sama.

II.5.1. Mengenal *Microsoft Visual Basic.Net*

Semenjak *Visual Studio.NET*, *Microsoft* telah banyak melakukan pengembangan dan perubahan pada tampilan *software* ini. Jadi apabila anda sudah terbiasa menggunakan rilis *Visual Basic* sebelumnya, anda harus mulai beradaptasi dengan tampilan baru *Visual Basic*. Pada dasarnya tampilan baru ini memudahkan kita dalam menggunakan *software Visual Basic* (disingkat VB).

II.6. SQL Server 2008

SQL Server 2008 adalah terobosan baru dari microsoft dalam bidang database. *SQL Server* adalah sebuah DBMS (*Database Management System*) yang dibuat oleh microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle. *SQL Server* 2008 dibuat pada saat kemajuan dalam bidang hardware sedemikian pesat. Oleh

karena itu sudah dapat dipastikan bahwa *SQL Server* 2008 membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data.

1. Versi-versi *SQL Server* 2008

Microsoft merilis *SQL Server* 2008 dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju. Versi-versi tersebut adalah sbb:

- Versi 32-bit (x86), yang biasanya digunakan untuk komputer dengan *single processor* 32 bit dan sistem operasi windows XP.
- Versi 64-bit (x64), yang biasanya digunakan untuk komputer dengan lebih dari satu *processor* (Misalnya: Core 2 Dou) dan system operasi 64 bit seperti Windows Xp 64, Vista, dan Windows 7 (Wahana Komputer ; 2010 : 2-3).

II.7. UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah bahasa pemodelan standar. (Chonoles, 2003: bab 1) mengatakan sebagai bahasa, berarti *UML* memiliki sintaks dan semantik. Ketika kita membuat model menggunakan konsep *UML* ada aturan - aturan yang harus diikuti. Bagaimana elemem pada model – model yang dibuat berhubungan satu dengan yang lainnya harus mengikuti standar yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. *UML* diaplikasikan untuk maksud tertentu, biasanya antara lain untuk:

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.

4. Mendokumentasi sistem yang ada, proses – proses dan organisasinya (Prabowo Pudjo Widodo Herlawati ; 2011 : 6).

II.7.1. Diagram – Diagram Pada Metode UML

1. Object Diagram

Diagram objek menggambarkan struktur sistem dari segi penanaman objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas haarus dipakai objeknya, karena jika tidak pendefinisian kelas itu tidak dapat dipertanggung jawabkan. Berikut ini adalah simbol – simbol yang digunakan pada diagram objek:

Tabel II.1. Tabel simbol – simbol objek diagram

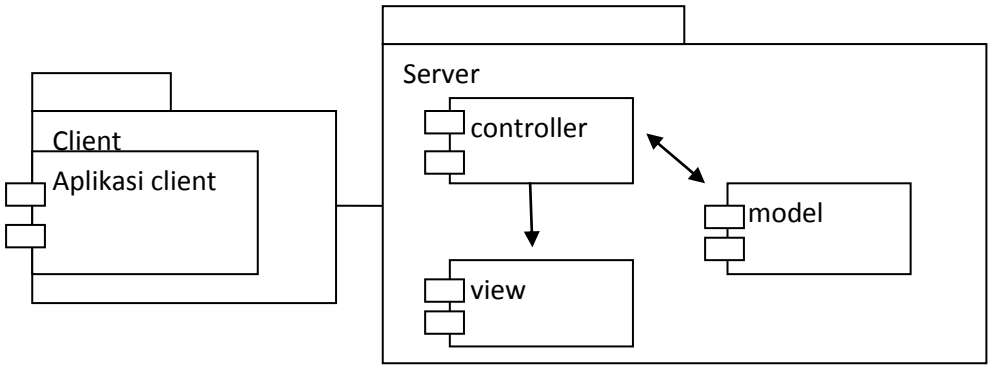
Simbol	**Deskripsi**
Objek <u>nama objek : nama kelas</u> Atribut = nilai	Objek dari kelas yang berjalan saat sistem dijalankan.
Link _____	Relasi antar objek.

Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 124)

2. *Component Diagram*

Diagram komponen atau *Component Diagram* dibuat untuk menunjukkan organisasi dan ketergantungan di antara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut:

a. *Source code* program perangkat lunak.
b. Komponen *executable* yang dilepas ke user.
c. Basis data secara fisik.
d. Sistem yang harus beradaptasi dengan sistem lain.
e. *Framework* sistem, framework pada perangkat lunak merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaan aplikasi, contohnya seperti Struts dari *Apache* yang menggunakn prinsip desain *Model-View-Controller* (MVC) dimana *source code* program dikelompokkan berdasarkan fungsinya seperti pada gambar berikut:



Gambar II.1. Gambar Ilustrasi FrameWork

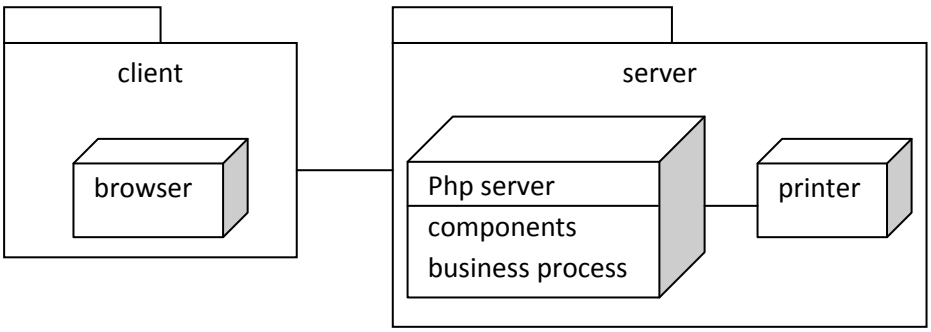
Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 125)

Di mana *Controller* berisi *souce code* yang menangani *request* dan validasi, model berisi *source code* yang menangani manipulasi data dan *business logic*, dan *view* berisi *source code* yang menangani tampilan.

3. *Deployment Diagram*

Diagram *Deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut:

a. Sistem tanbahan (*embedded system*) yang menggambarkan rancangan *device*, *node* dan *hardware.*
b. Sistem *client/server*
c. Sistem terdistribusi murni.
d. Rekayasa ulang apikasi.

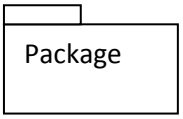
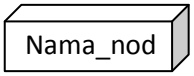


Gambar II.2. Diagram Deployment Sistem Client/Server

Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 129)

Berikut adalah simbol-simbol yang ada pada digram *deployment:*

Tabel II.2. Simbol-simbol pada Diagram Deployment

Simbol	**Deskripsi**
Package 	Package merupakan sebuah bungkusan dari satu atau leih node.
node 	Biasanya mengacu pada perangkat keras *(hardware*), perangkat lunak (*software*), jika didalam *node* disertakan komponen untuk mengkonsistenkan yang diikut sertakan harus sesuai dengan komponen yang telah didefinisikan sebelumnya pada diagram komponen.
Link ———————	Relasi antar *node.*
Kebergantungan /*dependency* ..... >	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai.

Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 130)

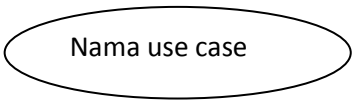
4. *Use Case Diagram*

Use Case atau diagram *use case* merupakan pemodelan untuk kelekuan *(behavior)* sistem informasi yang akan dibuat. *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 sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penanaman pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

a. Aktor merupakan 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, tetapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut adalah simbol-simbol yang ada pada diagram *use case*:

Tabel II.3. Simbol-simbol pada Diagram Use Case

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 nam use case
Aktor  nama aktor	Orang, proses, atau sistem lain yang berintraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat, jadi walaupun simbol aktor adalah orang, tapi aktor belum tenntu merupakan orang.
Asosiasi / *association* ———————	Komunikasi antar aktor dan use case yang berpartisipasi pada use case memiliki interaksi pada aktor.
Asosiasi berarah / *directed association* ——————→	Hubungan generalisasi dan spesialisasi antara dua buah use case dimana fungsi lebih umum dari lainnya.

Ekstensi / extend	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu.
<<extend>> .....>	Case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, misal:
Menggunakan / 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.

Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 131)

5. *Activity Diagram*

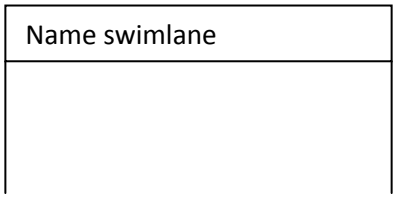
Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefenisikan 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 antar muka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya (Rosa A.S M.Shalahuddin ; 2011 : 134).

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel II.4. Diagram Aktivitas

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 pilihan aktivitas lebih dari satu
penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah

	status akhir.
swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

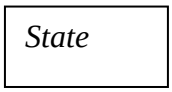
Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 134)

6. *State Machine Diagram*

State Machine Diagram atau dalam bahasa Indonesia disebut diagram mesin status digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem. Perubahan tersebut digambarkan dalam suatu graf berarah. *State machine diaggram* merupakan pengembangan dari diagram *Finite State autmata* dengan penambahan beberapa fitur dan konsep baru. Diagram *Finite State Automata* (FSA) ini biasanya diajarkan dalam mata kuliah Automata (Rosa A.S M.Shalahuddin ; 2011 : 136).

Tabel II.5 Komponen dasar *State Machine Diagram*

Simbol	**Deskripsi**
Start (Initial State) 	*Start* atau *initial state* adalah *state* atau keadaan awal pada saat sistem mulai hidup.

End (Final State) 	*End* atau *final state* adalah *state* keadaan akhir dari daur hidup suatu sistem.
event 	*Event* adalah kegiatan yang menyebabkan berubahnya status mesin.
State 	*State* atau status adalah keadaan sistem pada waktu tertentu. *State* dapat berubah jika ada event tertentuyang memicu perubahan tersebut.

Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 137)

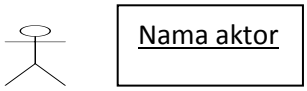
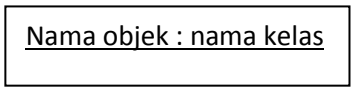
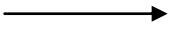
7. *Sequence Diagram*

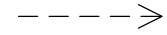
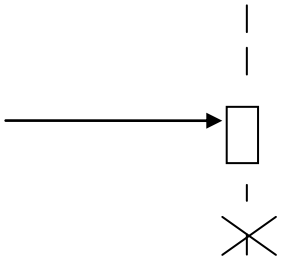
Diagram sekuen menggambarkan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen mamka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambarkan adalah 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 juaga semakin banyak (Rosa A.S M.Shalahuddin ; 2011 : 137).

Tabel II.6. simbol-simbol digram sekuen

Simbol	**Deskripsi**
Aktor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem infromasi
Garis hidup	Menyatakan kehidupan suatu objek.
objek 	Menyatakan objek yang berinteraksi pesan.
State 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan.
Pesan tipe create <<create>> 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe *call* 1:  nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi / metode.
Pesan tipe *send*	Menyatakan bahwa suatu objek

1:masukan 	mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
Pesan tipe *return* 1:keluaran 	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* <<*destroy*>> 	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada destroy.

Sumber : (Rosa A.S M. Shalahuddin ; 2011 : 138)

8. *Communication Diagram*

Communication diagram atau diagram komunikasi pada UML versi 2.x adalah penyederhanaan dari diagram kolaborasi (*collaboration diagram*) pada UML versi 1.x . Jadi *collaboration diagram* sudah tidak muncul lagi pada UML versi 2.x.

Diagram komukasi menggambarkan interaksi antar objek/bagian dalam bentuk urutan pengiriman pesan. Diagram komunikasi merepresentasikan informasi yang diperoleh dari diagram kelas, diagram sekuen, dan Diagram *Use Case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem.

Diagram komunikasi mengelompokkan *message* pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram komunikasi yang dituliskan adalah operasi/metode yang dijalankan antara objek yang satu dan objek lainnya secraa keseluruhan, oleh karena itu dapat diamil dari jalannya interaksi pada semua diagram sekuen. Penomoran metode dapat dilakukan berdasarkan urutan dijalankannya metode/operasi di antara objek yang satu dengan objek lainnya atau objek itu sendiri (Rosa A.S M.Shalahuddin ; 2011 : 140).

II.8. Normalisasi

Normalisasi adalah suatu proses yang digunakan untuk menentukan pengelompokan atribut-atribut dalam sebuah relasi sehingga diperoleh relasi yang berstruktur baik. Dalam hal ini yang dimaksud dengan relasi yang berstruktur baik adalah relasi yang memenuhi dua kondisi berikut:

1. Mengandung redudansi sesedikit mungkin.
2. Memungkinkan baris-baris dalam relasi disisipkan, dimodifikasi, dan dihapus tanpa menimbulkan kesalahan atau ketidak konsistenan (Abdul Kadir ; 2009 : 116).

Normalisasi sendiri dilakukan melalui sejumlah langkah. Setiap langkah berhubungan dengan bentuk normal *(normal form)* tertentu. Dalam hal ini yang

disebut bentuk normal adalah “suatu keadaan relasi yang dihasilkan oleh penerapan aturan-aturan sederhana yang berhubungan dengan dependensi fungsional terhadap relasi tersebut” (Hoffer, dkk ; 2010). Yang dimaksud dengan aturan-aturan tersebut dan juga istilah dependensi fungsional akan dibahas belakangan. Bentuk normal dalam normalisasi dapat berupa:

- Bentuk normal pertama *(*1NF */ First normal form)*
- Bentuk normal kedua (2NF / *Second normal form*)
- Bentuk normal ketiga (3NF / *Third normal form*)
- Bentuk normal Boyce-Codd (BCNF / *Boyce-Codd normal form*)
- Bentuk noral keempat (4NF / *Fourth normal form*)
- Status Bentuk normal kelima (5NF / *Fifth normal form*) (Abdul Kadir ; 2009 : 116-117).

1. Bentuk Normal Pertama (1NF)

Contoh yang digunakan disini adalah sebuah perusahaan yang mendapatkan barang dari pemasok yang berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dam masing-masing kota mempunyai kode status sendiri. Masing-masing pemasok bisa menyediakan banyak barang. Tabel relasionalnya dapat ditulis sebagai berikut:

PEMASOK (p#, status, kota, b#, qty) dimana

P# : kode pemasok (kunci utama)

Status : kode status kota

Kota : nama kota

b# : barang yang dipasok

qty : jumlah barang yang dipasok

Agar bisa digabungkan jumlah barang yang dipasok (qty) secara unik dengan barang (b#) dan pemasok (p#), kita menggunakan kunci utama gabungan yang tersusun dari b# dan p#.

Sebuah tabel relasional secra defenisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolomnya adalah atomik. Ini berarti kolom-kolom tidak mempunya nilai berulang.

Meskipun barada pada 1NF, tabel pemasok mengandung data berulang. Sebagai contoh, informasi tentang lokasi pemasok dan status lokasi harus diulang untuk setiap barang yang dipasok. Perulangan menyebabkan apa yang disebut *update anomalies. Update anomalies* adalah masalah yang timbul ketika informasi ditambahkan , dihapus, atau di-*update*.

2. Bentuk Normal Kedua (2NF)

Defenisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak ada pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama. Ini berarti setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama. Proses mengubah tabel 1NF menjadi 2NF adalah:

a. Tentukan senbarang kolom penentu selain kunci gabungan dan kolom-kolom yang ditentukannya.
b. Buat dan beri tabel baru untuk masing-masing penentu dan kolom-kolom yang ditentukannya.

c. Pindahkan kolom-kolom yang ditentukan dari tabel asal ke tabel baru. Penentu akan menjadi kunci utama pada tabel baru.
d. Hapus kolom yang baru dipindahkan dari tabel asal, kecuali penentu yang berfungsi sebagai kunci tamu.
e. Tabel asal bisa diberi nama baru.

3. Bentuk Normal Ketiga (3NF)

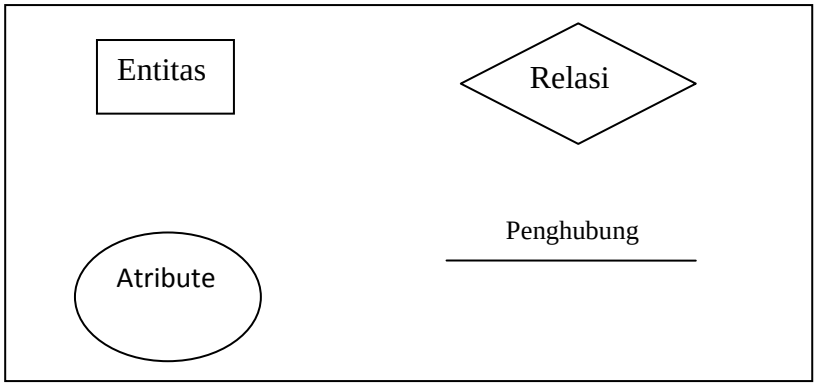
Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secra defenisi, sebuah tabel berada pada normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setip kolom bukan kunci tidak tergantung secara transitif dengan kunci utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama (Janer Simarmata & Imam Paryudi ; 2009 : 79-82).

II.9. *Entity Relationship Diagram* (ERD)

Entity relationship Diagram (ERD) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek). Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain. Entitas digambarkan dalam basis data dengan kumpulan atribut. Misalnya: nim, nama, alamat, dan

kota. Relasi adalah hubungan antara beberapa entitas. (Jurnal Teknologi Informatika, 2011: 150).

Struktur logis (skema database) dapat ditunjukkan secara grafis dengan diagram ERD yang dibentuk dari komponen-komponen berikut : (Jurnal Teknologi Informatika, 2011 : 150).



Gambar II.3. Komponen – Komponen ERD

Sumber: Jurnal Teknologi Informatika (2011 : 151)

II.9.1 Pemetaan Kardinalitas

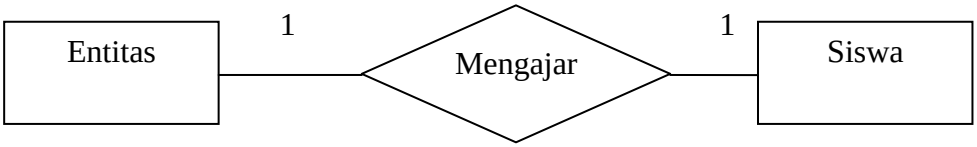
Pemetaan kardinalitas menyatakan jumlah entitas di mana entitas lain dapat dihubungkan ke entitas tersebut melalui sebuah himpunan relasi. (Jurnal Teknologi Informatika, 2011:150).

1. *One to One*

Sebuah entitas pada A berhubungan dengan paling banyak satu entitas pada B dan sebuah entitas pada B berhubungan dengan paling banyak satu entitas pada A.

Contoh:

Pada pengajaran privat, satu guru satu siswa. Seorang guru mengajar seorang siswa, seorang siswa diajar oleh seorang guru.



Gambar II.4. Hubungan *One To One.*

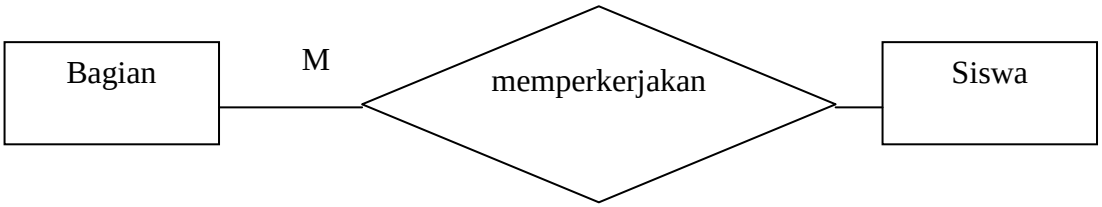
Sumber: Jurnal Teknologi Informatika (2011 : 152)

2. *One to Many/ Many to One*

Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dansebuah entitas pada B berhubungan dengan paling banyak satu entitas pada A, atau sebaliknya (Many to One).

Contoh:

Dalam satu perusahaan, satu bagian mempekerjakan banyak pegawai. Satu bagian mempekerjakan banyak pegawai, satu pegawai kerja dalam satu bagian.



Gambar II.5. *One To Many*

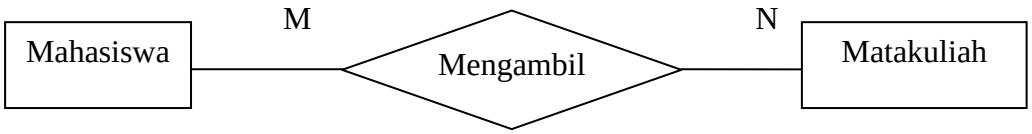
Sumber: Jurnal Teknologi Informatika (2011 : 152)

3. *Many To Many*

Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dan sebuah entitas pada B berhubungan dengan lebih dari satu entitas pada A.

Contoh:

Dalam universitas, seorang mahasiswa dapat mengambil banyak mata kuliah.Satu mahasiswa mengambil banyak mata kuliah dan satu mata kuliah diambil banyak mahasiswa.



Gambar II.6. Hubungan *Many To Many*

Sumber: Jurnal Teknologi Informatika (2011 : 152)