BAB II

LANDASAN TEORI

2.1 Konsep Dasar Sistem

A. Pengertian Sistem

Menurut Atmosudirdjo dalam Sutabri (2012:7) Sistem terdiri atas objek-objek atau unsure-unsur atau komponen-komponen yang berkaitan dan berhubungan satu sama lainnya sedemikian rupa sehingga unsur-unsur tersebut merupakan suatu kesatuan pemrosesan atau pengolahan yang tertentu.

Pengertian sistem yang dikemukakan menurut Al Fattah (2009:3) secara sederhana, sistem dapat diartikan sebagai "suatu kumpulan atau himpunan dari unsur atau variable-variabel yang saling terorganisasi.

Sedangkan menurut Mc Leod dalam Darmawan dan Fauzi (2013:4) Sistem adalah sekelompok elemen-elemen yang terintegrasi dengan tujuan yang sama untuk mencapai tujuan.

B. Sistem Berorientasi Objek *(OOP)*

Menurut Rosa dan Shalahuddin (2013:103) "Sistem Berorientasi Objek merupakan sebuah sistem yang dibangun berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok data dan fungsi". Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat dan komponen lainnya serta dapat berinteraksi satu sama lain.

C. Karakteristik Sistem

Menurut Sutabri (2012:13) Model umum sebuah sistem terdiri dari *input*, proses, dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana mengingat sebuah sistem dapat mempunyai beberapa masukan dan keluaran sekaligus. Selain itu sebuah sistem juga memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut :

1. Komponen Sistem (*Components*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat-sifat sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar yang disebut dengan supra sistem.

2. Batasan Sistem (*Boundary*)

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luar Sistem (*Environtment*)

Bentuk apapun yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem. Lingkungan luar sistem dapat menguntungkan dan dapat juga merugikan sistem tersebut, yang dengan demikian lingkungan luar tersebut harus selalu dijaga dan dipelihara. Sedangkan lingkungan luar yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Luar Sistem(*Interface*)

Media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lain. Keluaran suatu subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung. Dengan demikian terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukan Sistem (*Input*)

Energi yang dimasukan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Sebagai contoh, didalam suatu unit sistem computer, "program" adalah *maintenance input* yang digunakan untuk mengoperasikan *computer*. Sementara "data" adalah signal input yang akan diolah menjadi informasi.

6. Keluaran Sistem (*Output*)

Hasil dari energy yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Seperti contoh sistem informasi, keluaran yang dihasilkan adalah informasi, dimana informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal lain yang merupakan input bagi subsistem lainnya.

7. Pengolah Sistem (*Procces*)

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Sebagai contoh, sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan.

8. Sasaran Sistem (*Objective*)

Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasi bila mengenai sasaran atau tujuan yang telah direncanakan.

D. Pengertian Perancangan Sistem

Menurut Kursini dan Koniyo (2007:79) perancangan sistem adalah "proses pengembangan spesifikasi sistem baru berdasarkan hasil rekomendasi analisis sistem."

Dalam tahap perancangan tim kerja desain harus merancang spesifikasi yang dibutuhkan dalam berbagai kertas kerja. Kertas kerja itu harus memuat berbagai uraian mengenai input, proses, dan output dari sistem yang diusulkan.

Desain / Perancangan sistem dapat diartikan sebagai :

1. Tahap setelah analisis dari siklus pengembangan sistem.
2. Pendefinisian atas kebutuhan-kebutuhan fungsional.
3. Persiapan untuk rancang bangun implementasi.
4. Menggambarkan bagaimna suatu sistem dibentuk, berupa penggambaran perencanaan, pembuatan sketsa, pengaturan dari beberapa elemen terpisah dalam satu kesatuan yang utuh dan berfungsi.
5. Konfigurasi komponen software dan hardware sistem.

Tujuan Tahap perancangan sistem :

1. Memenuhi kebutuhan pemakai sistem.
2. Memberikan gambaran yang jelas dan rancang bangun yang lengkap untuk pemorograman dan ahli-ahli teknik yang terlibat.

E. Pengertian Sistem Informasi

Menurut Sutabri (2012:38) Sistem informasi adalah “suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang

mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk menyediakan laporan-laporan yang diperlukan."

Sedangkan menurut Anthony dan Dearden dalam Jogiyanto (2005:8) mendefinisikan bahwa "informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya". Sumber dari informasi adalah data. Data merupakan bentuk jamak dari bentuk tunggal *datum* atau data item. Data adalah fakta atau atau apapun yang dapat digunakan sebagai input dalam menghasilkan informasi. Data bisa berupa bahan atau diskusi, pengambilan keputusan, perhitungan, atau pengukuran.

F. Pengertian Sistem Informasi Manajemen

Menurut Lucas dalam Hartono (2013:20) "Seperangkat prosedur yang tersusun dengan baik yang pada saat dijalankan menghasilkan informasi untuk mendukung pengambilan keputusan dan pengendalian organisasi."

> Menurut Zakiyudin (2012a:19) Menyimpulkan pada dasarnya sistem informasi manajemen adalah suatu sistem informasi yang menggambarkan ketersediaan suatu rangkaian data yang cukup lengkap yang disimpan agar dapat menyediakan informasi untuk mendukung operasi, majemen, dan pembuatan keputusan dalam suatu organisasi.

Menurut Scott dalam Zakiyudin (2012b:19) Sistem informasi manajemen adalah sekumpulan sistem informasi yang saling berinteraksi, yang memberikan informasi baik untuk kepentingan operasi atau kegiatan manajerial.

G. Sistem Penggajian

Menurut Jogiyanto (2005:307) "Gaji merupakan imbalan kepada pegawai yang diberi tugas-tugas administratif dan pimpinan yang jumlahnya, biasanya tetap secara bulanan/tahunan."

Gaji adalah pembayaran atas penyerahan jasa yang dilakukan oleh karyawan yang mempunyai jenjang jabatan. Umumnya gaji dibayar secara tetap per bulan.

H. *Website*

Menurut Ardhana (2012:3) "*World Wide Web* atau lebih sering dikenal Web adalah suatu layanan sajian informasi yang menggunakan konsep *hyperlink* (tautan), yang memudahkan *surfer* (sebutan para pemakai komputer yang melakukan *browsing* atau penelusuran informasi melalui Internet)". Keistimewaan inilah yang telah menjadikan Web sebagai service yang paling cepat pertumbuhanya.

I. *MySQL*

Menurut Rosa dan Shalahuddin (2013:46), " *Structured Query Languange* (SQL) adalah bahasa yang digunakan untuk megelola data pada *Relational-Database Management System* (RDBMS). *Structured Query Languange* (SQL) awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus.

J. *HyperText Markup Languange* (HTML)

Menurut Winarno dkk (2013:3), "*HyperText Markup Languange* (HTML) adalah bahasa *markup* yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah *browser* web di internet".

K. *Personal Home Page* (PHP)

Menurut Winarno dkk (2013:59), "*Personal Home Page* (PHP) adalah Bahasa bahasa pemrograman yang memungkinkan anda menggenerate kode *HyperText Markup Languange* (HTML) secara dinamis, artinya bisa membuat tampilan halaman web yang dinamis, bisa berubah-ubah sesuai dengan keinginan programmernya".

L. *XAMPP*

Menurut Nugroho (2014:1), "XAMPP adalah paket program *web* lengkap yang dapat anda pakai untuk belajar pemograman *web*, khususnya PHP dan MYSQL paket ini dapat di *download* secara gratis dan legal".

M. Basis Data

Menurut Rosa dan Shalahuddin (2013:44) "Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan". Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa *file* teks ataupun DBMS *(Database Management System)*.

Kebutuhan basis data dalam sistem informasi meliputi:

1. Memasukan, menyimpan dan mengambil data .

2. Membuat Laporan berdasarkan data yang telah tersimpan.

2.2 Teori Pendukung

A. *Entity Relationship Diagram* (ERD)

Menurut Fathansyah (2012:74) "ERD merupakan semesta data yang ada di 'dunia nyata' diterjemahkan dengan memanfaatkan sejumlah perangkat konseptual menjadi sebuah diagram data."

B. Komponen ERD

Dalam pembentukan ERD terdapat 3 komponen yang akan dibentuk yaitu :

1. Entitas

Menurut Fathansyah (2012:75) "Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain."

Sebuah kursi yang kita duduki, seorang yang menjadi pegawai disebuah perusahaan dan sebuah mobil yang melintas didepan kita adalah entitas.Contoh : Mahasiswa, Mobil, Pegawai, Pelanggan.

2. Hubungan (relasi/*relationship*)

Menurut Fathansyah (2012:77) "Relasi menunjukan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda."

Contoh : Seorang mahasiswa dengan nim='100001' dan nama_mhs='Ali Akbar' (yang ada dihimpunan entitas mahasiswa) mempunyai relasi dengan entitas sebuah mata kuliah dengan kode_kul ='IF-110' dan nama_kul='Struktur Data'.

Daftar relasi yang digunakan dalam ERD antara lain ;

a) Relasi satu-ke-satu (*one to one*)

Sumber : Fathansyah (2012:82)

Gambar II.1 Relasi satu ke satu (*one-to-one*)

Dosen hanya memiliki satu jurusan dan jurusan memiliki satu dosen

b) Relasi satu-ke-banyak (*one to many*)

Sumber : Fathansyah (2012:83)

Gambar II.2 Relasi satu-ke-banyak (*one-to-many*)

Dosen hanya mengajar satu mata kuliah dan satu mata kuliah belum tentu hanya memiliki satu dosen.

c) Relasi banyak-ke-banyak (*many-to-many*)

Sumber : Fathansyah (2012:83)

Gambar II.3 Relasi banyak-ke-banyak (*many-to-many*)

Mahasiswa mempelajari banyak mata kuliah dan mata kuliah tidak hanya dipelajari oleh mahasiswa.

3. Atribut

Menurut Fathansyah (2012:76) "setiap entitas pasti memiliki atribut yang mendeskripsikan karakteristik (property) dari entitas tersebut.

Langkah-langkah dalam membuat ERD adalah sebagai berikut :

a) Mengidentifikasikan dan menetapkan seluruh himpunan entitas yang akan terlibat.
b) Menentukan atribut-atribut *key* masing-masing himpunan entitas.
c) Mengidentifikasikan dan menetapkan seluruh himpunan relasi diantara himpunan entitas yang ada beserta *foreign key*-nya.
d) Menentukan derajat dan *cardinality* rasio relasiuntuk setiap himpunan relasi.
e) Melengkapi himpunan relasi dengan atribut-atribut yang bukan kunci (*non-key*).

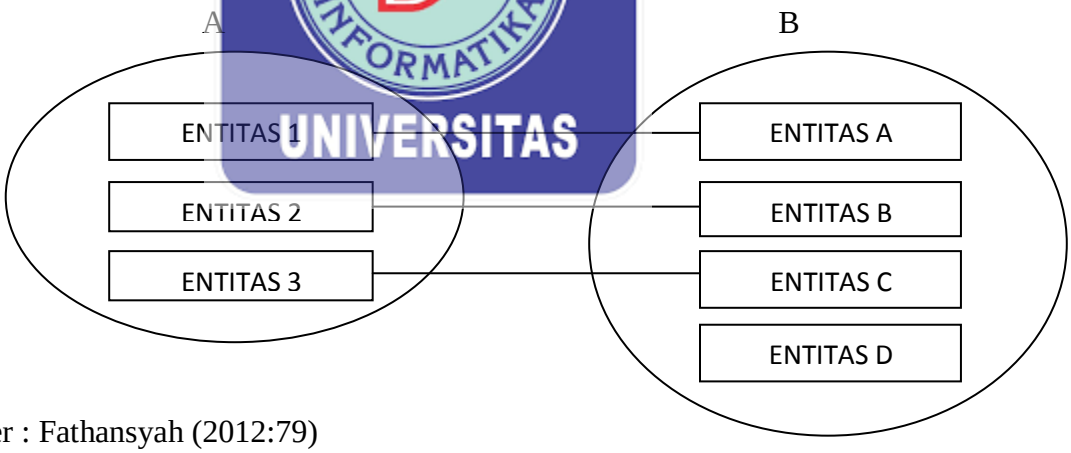
C. *Logical Record Structure* (LRS)

Menurut Fathansyah (2012:78) "*Logical record structure* adalah representasi dari struktur *record-record* pada tabel yang terbentuk dari hasil antar himpunan entitas dan kardinalitas relasi menunjukan jumlah maksimum entitas yang dapat berelasi pada himpunan entitas yang lain."

Derajat relasi atau kardinalitas rasio menjelaskan jumlah maksimum hubungan antara satu entitas dengan entitas lainnya, yaitu :

1. *One to one*

   Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.



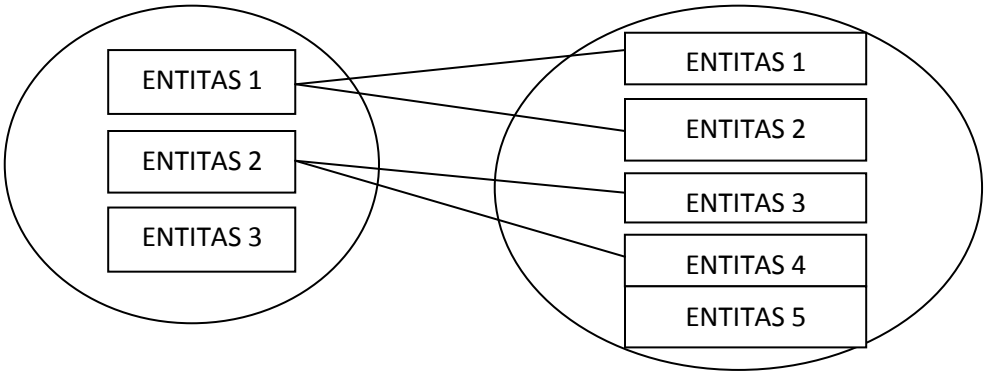
Sumber : Fathansyah (2012:79)

Gambar II.4 kardinalitas *one to one* (1:1)

2. *One to many*

   Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.

A B

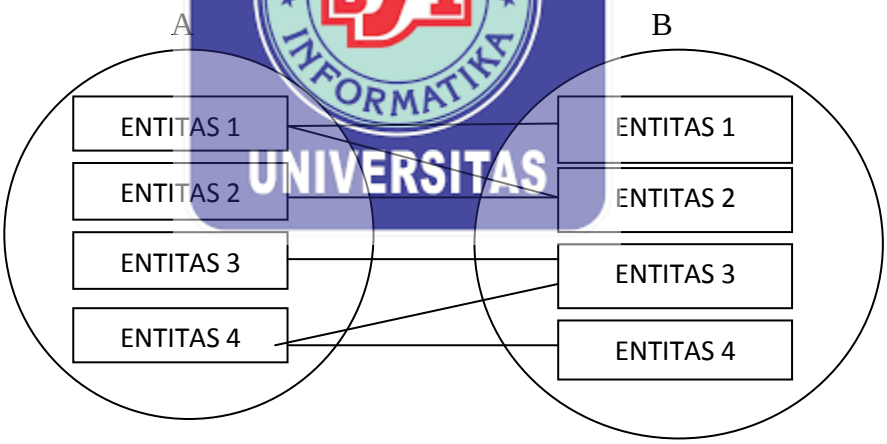


Sumber : Fathansyah (2012:80)

Gambar II.5 kardinalitas *one to many*(1:M)

3. *Many to many*

   Setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya.



Sumber : Fathansyah (2012:81)

Gambar II.6 kardinalitas *many to many*(M:M)

2.2.3 *Unified Modelling Language* (UML)

A. Pengertian *Unified Modelling Language* (UML)

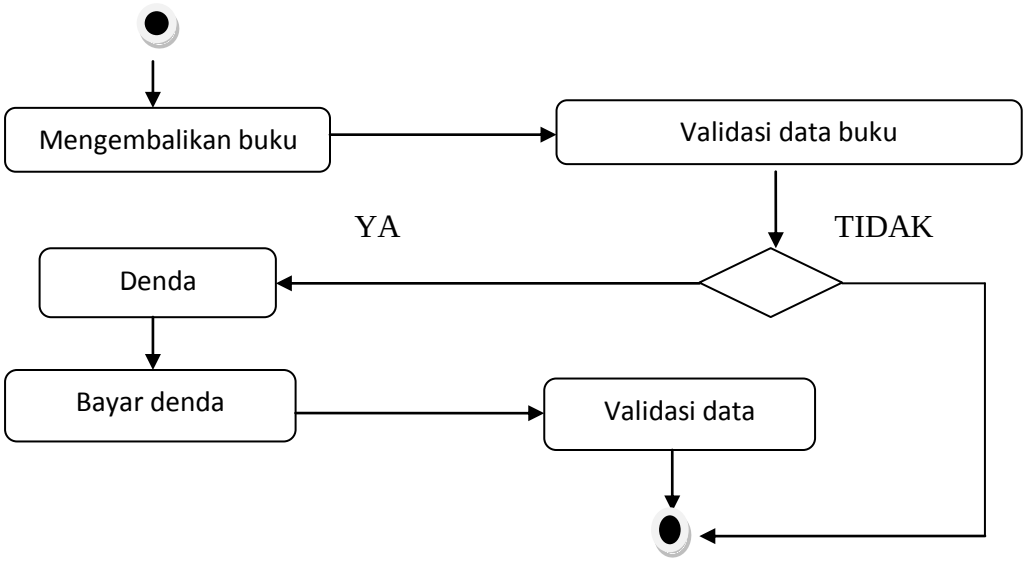
Menurut Rosa dan M. Shalahuddin (2013:138) *Unified Modelling Language* (UML) adalah sekumpulan spesifikasi yang dikeluarkan oleh *Object Management Group* (OMG).

Pada perkembangan teknik pemroggraman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karna adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

B. *Activity Diagram*

Activity diagram menurut Munawar (2005:109) "adalah teknik untuk mendeskripsikan logika *procedural*, proses bisnis dan aliran kerja dalam banyak kasus, *Actifity diagram* mempunyai peran seperti hal nya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku pararel sedangkan *flowchart* tidak bisa."

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.



Sumber : Munawar (2005:111)

Gambar II.7 Contoh *activity diagram* sederhana

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

1. Rancangan proses bisnis dimana setiap urusan 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.

C. *Use Case Diagram*

Menurut Munawar (2005:71) "*Use case* adalah kontruksi untuk mendeskripsikan bagaimana *system* akan terlihat dimata pengguna potensial. *Use case* terdiri dari sekumpulan sekenario yang dilakukan oleh seorang actor (orang-orang, perangkat keras, urutan waktu atau sistem yang lain), sedangkan use case diagram memfasilitasi komunikasi diantara analis dan pengguna serta diantara analis dan klien."

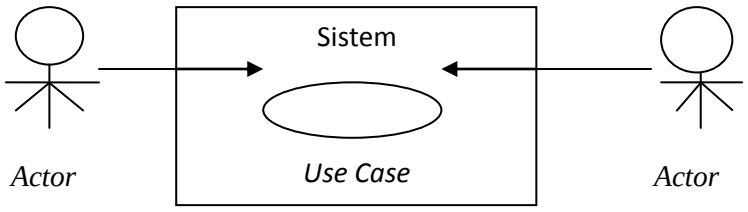
Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

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

1. 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 symbol dari aktor adalah gambar, tapi aktor belum tentu merupakan orang.

2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Diagram *Use case* menunjukan tiga aspek dari *system* yaitu *actor*, *use case*, dan *system / sub system boundary*. *Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Gambar II.7 mengilustrasikan *actor*, *use case*, dan *boundary*.



Sumber : Munawar (2005:64)

Gambar II.8 *Use Case* Model

D. *Class Diagram*

Diagram kelas atau *class diagram* menurut Munawar (2005:66) menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

Diagram kelas dibuat agar pembuatan program atau *Programmer* membuat kelas-kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki kelas sebagai berikut :

1. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2. Kelas yang menangani tampilan (*view*)

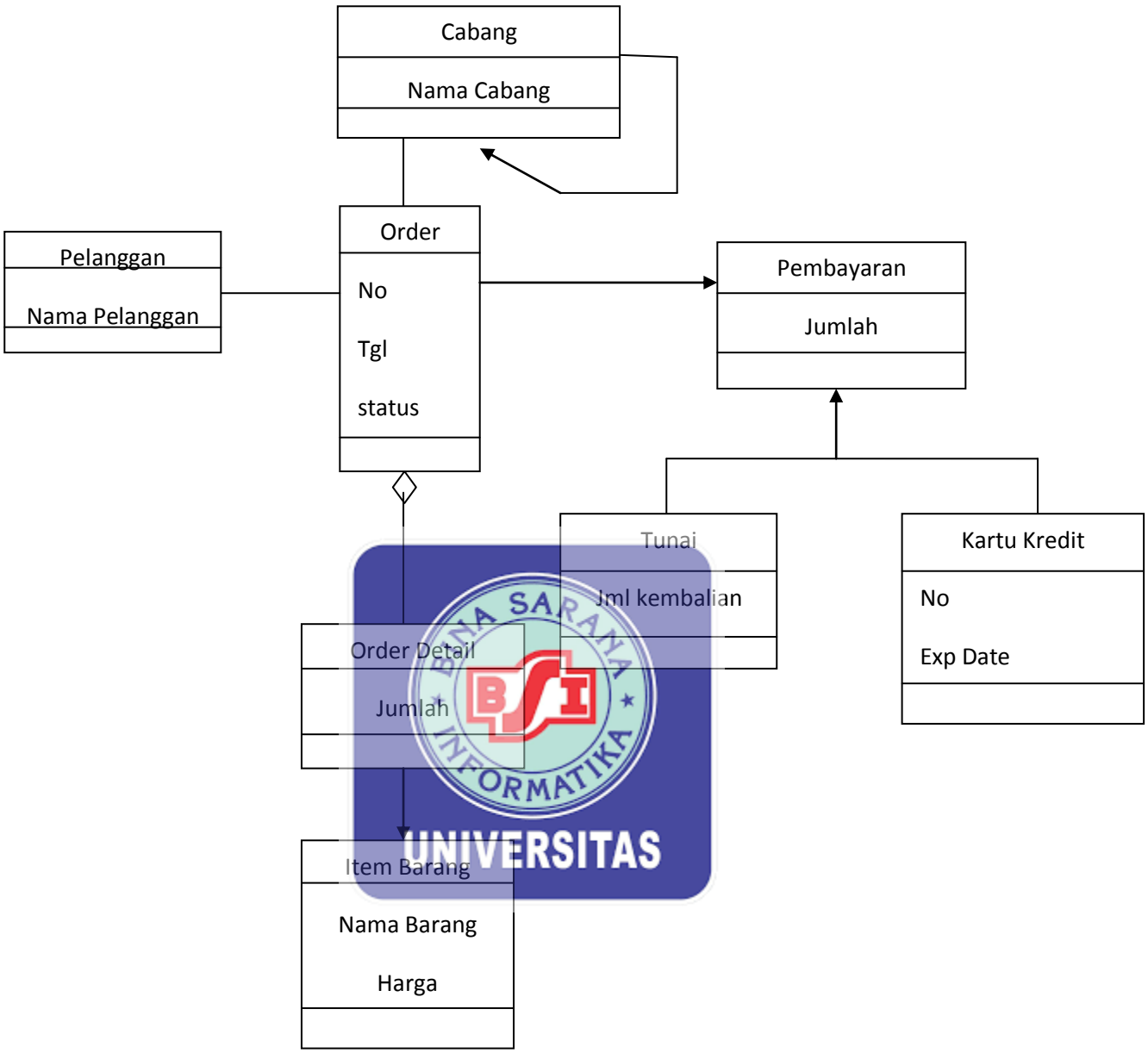
   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3. Kelas yang diambil pendefinisian *use case* (*controller*)

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *usecase*, kelas ini biasanya disebut dengan kalas proses yang menangani proses bisnis dan perangkat lunak.

4. Kelas yang diambil dari pendefinisian data (*model*)

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



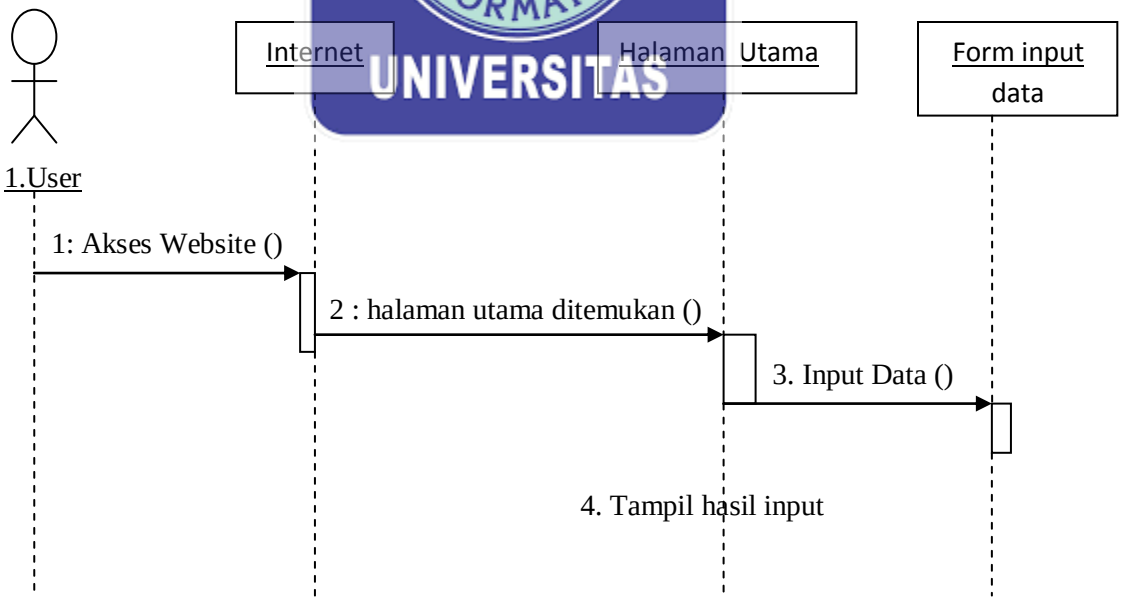
Sumber : Munawar (2005:67)

Gambar II.9 Contoh *Class Diagram*

E. *Sequence Diagram*

Menurut Munawar (2005:87) "*Sequence Diagram* digunakan untuk menggambarkan perilaku pada sebuah *scenario*, diagram ini menunjukan sejumlah contoh obyek dan message (pesan) yang diletakkan diantara obyek-obyek didalam *use case*".

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Untuk menggambar diagram *sequence* 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 sekenario yang ada pada *use case*.



Sumber : Munawar (2005:67)

Gambar II.10 Contoh *Sequence Diagram*