BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan / berinteraksi dan saling memengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.

Menurut Jerry FithGerald, Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.

1. Syarat-Syarat Sistem
    a. Sistem harus dibentuk untuk menyelesaikan tujuan.
    b. Elemen sistem harus mempunyai rencana yang ditetapkan.
    c. Adanya hubungan di antara elemen sistem.
    d. Unsur dasar dari proses (arus informasi, energi dan material) lebih penting daripada elemen sistem.
    e. Tujuan organisasi lebih penting dari pada tujuan elemen.
2. Karakteristik Sistem
    a. Komponen (*Component*)

        Suatu sistem terdiri dari jumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan. Komponen-komponen sisten dapat berupa suatu subsistem atau bagian-bagian sistem. Setiap subsistem mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi

tertentu dan memengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai proses sistem yang lebih besar yang disebut *supra sistem*.

b. Batas Sistem (*Boundary*)

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lain atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan, karena dengan batas sistem ini fungsi dan tugas dari subsistem yang satu dengan yang lain berbeda tetapi tetap saling berinteraksi. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

c. Lingkungan Luar Sistem (*Environment*)

Environment merupakan segala sesuatu yang berada di luar batas sistem yang memengaruhi operasi dari suatu sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan atau merugikan. Lingkungan luar yang menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya, sedangkan lingkungan luar yang merugikan harus dimusnahkan atau dikendalikan agar tidak mengganggu operasi sistem.

d. Penghubung Sistem (*Interface*)

Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya untuk membentuk satu kesatuan sehingga sumber-sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya. Dengan kata lain, *output* dari suatu subsistem akan menjadi *input* dari subsistem yang lainnya.

e. Masukan Sistem (*Input*)

Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*Maintenance Input*) adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Masukan sinyal (*Signal Input*) adalah energi yang diproses untuk didapatkan keluaran.

f. Keluaran Sistem (*Output*)

Merupakan hasil dari energi yang diolah oleh sistem, meliputi *output* yang berguna contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembuangan, contohnya panas yang dikeluarkan komputer.

g. Pengolah Sistem (*Process*)

Merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan contoh CPU pada komputer.

h. Tujuan Sistem (*Goal*)

Setiap sistem mempunyai tujuan ataupun sasaran yang memengaruhi *input* yang dibutuhkan dan *output* yang dihasilkan. Dengan kata lain, suatu sistem itu mengenai sasaran atau tujuannya. Jika sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. (Asbon Hendra; 2012 : 157)

II.2. Sistem Informasi

Menurut Kusrini (2007 : 11) Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi

harian, mendukung operasi, bersifat manajerial, dan merupakan kegiatan stategi dari suatu organisasi, serta menyediakan laporan-laporan yang diperlukan pihak luar.

Sistem informasi mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk suatu tujuan khusus. Sistem informasi adalah jantung bagi sebagian besar organisasi. Sebagai contoh, bank dan perusahaan penerbangan tidak akan dapat berfungsi tanpa sitem informasi mereka.

II.3. Sistem Pendukung Keputusan

Sistem pendukung keputusan (*decision support systems* disingkat DSS) adalah bagian dari sistem informasi berbasis komputer termasuk sistem berbasis pengetahuan (manajemen pengetahuan) yang dipakai untuk mendukung pengambilan keputusan dalam suatu organisasi perusahaan atau lembaga pendidikan. Dapat juga dikatakan sebagai sistem komputer yang mengolah data menjadi informasi untuk mengambil keputusan dari masalah yang spesifik.

Menurut Moore and Chang, Sistem Pendukung keputusan dapat digambarkan sebagai sistem yang berkemampuan mendukung analisis data, dan pemodelan keputusan, berorientasi keputusan, orientasi perencanaan masa depan, dan digunakan pada saat-saat yang tidak biasa.

Dari pengertian di atas dapat di jelaskan bahwa sistem pendukung keputusan merupakan sistem yang membantu pengambilan keputusan yang dilengkapi dengan informasi dari data yang telah diolah dengan relevan dan di

perlukan untuk membuat keputusan tentang suatu masalah dengan lebih cepat dan akurat. (Asep Abdul Wahit, Andri Ikhwana, Partono. 2012)

Sedangkan menurut Kusrini (2007 : 15) DSS merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data. Sistem itu digunakan untuk membantu pengambilan keputusan dalam situasi yang semiterstruktur dan situasi tidak terstruktur, dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat.

DSS biasanya dibangun untuk mendukung solusi atas suatu masalah atau untuk mengevaluasi suatu peluang. DSS yang seperti itu disebut aplikasi DSS. Aplikasi DSS digunakan untuk pengambil keputusan. Aplikasi DSS menggunakan CBIS (*Computer Based Information Systems*) yang fleksibel, interaktif, dan dapat diadaptasi, yang dikembangkan untuk mendukung solusi atas masalah manajemen spesifik yang tidak terstruktural.

II.4. Visual Basic .Net

Visual basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dikeluarkan oleh Microsoft, yaitu microsoft Visual Studio 2010. Sebagai produk lingkungan pengembangan terintegrasi atau IDE andalan yang di keluarkan oleh microsoft, visual studio 2010 menambahkan perbaikan-perbaikan fitur dan fitur baru yang lebih lengkap visual studio pendahuluanya, yaitu mirosoft visual studio 2008. (Wahana Komputer;2010;2)

Sedangkan menurut Aswan (2012 : 1) Visual basic 2010 adalah salah satu bagian dari microsoft visual studio 2010. Sebuat alat yang digunakan oleh pengembang

windows dari berbagai level untuk mengembangkan dan membangun aplikasi yang bergerak diatas sistem .NET Framework, dengan menggunakan bahasa BASIC. Visual Basic menyediakan cara cepat dan mudah untuk membuat aplikasi.

Setiap generasi baru dari perangkat lunak bahasa pemrograman datang karena adanya keterbatasan dari generasi sebelumnya. Teknologi device, hardware, network dan internet baru yang muncul menyebabkan bahasa pemrograman yang ada tidak lagi menjadi alat yang ideal untuk mengembangkan perangkat lunak yang dapat bekerja dengan teknologi baru tersebut (WAH[12]). Sekarang untuk pertama kalinya, platform pengembang perangkat lunak yang lengkap, Microsoft .NET telah didesain dari dasar dengan internet sebagai fokus utamanya (walaupun tidak secara eksklusif hanya untuk pengembang internet saja). Banyak inovasi baru yang berada dalam platform ini akan mengatasi keterbatasan dari tool-tool dan teknologi lama. Visual Basic .NET adalah pengembangan dari Visual basic sebelumnya. Kelebihan VB .NET 2010 terletak pada tampilannya yang lebih canggih dibandingkan dengan edisi Visual Basic sebelumnya. Selain memiliki kelebihan, VB .NET 2005 memiliki kekurangan. Kekurangan VB .NET 2005 yang terlihat jelas adalah beratnya aplikasi ini apabila dijalankan pada komputer yang memiliki spesifikasi sederhana.

II.5. SQL Server

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang database. SQL Server adalah sebuah DBMS (Database Management System) yang yang dibuat Microsoft untuk 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. (Wahana Komputer:2010;2)

II.6. Metode SAW

Metode SAW (*Simple Additive Weighting*) sering juga dikenal istilah metode penjumlahan berbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternative pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternative yang ada. (Heri Sulistiyo:2010)

Metode SAW dikenal sebagai istilah penjumlahan terbobot. Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada. Formula untuk melakukan normalisasi tersebut adalah sebagai berikut: dengan r adalah rating kinerja ternormalisasi dari alternatif Aij i pada atribut C; i=1,2,...,m dan j=1,2,...,n. (Yohana Dewi : 2010)

$$r_{ij} = \begin{cases} \dfrac{x_{ij}}{\underset{i}{Max}\ x_{ij}} & \text{jika j adalah atribut keuntungan} \\ \dfrac{\underset{i}{Min}\ x_{ij}}{x_{ij}} & \text{jika j adalah atribut biaya (cost)} \end{cases}$$

Nilai preferensi untuk setiap altenatif (Vi) diberikan sebagai :

$$V_i = \sum_{j=1}^{n} w_j r_{ij} \quad .....(1)$$

V = w x r

dengan:

V = Nilai Matriks

w = Matriks rating kepentingan (bobot)

r = rating

Nilai Vi yang lebih besar mengindikasikan bahwa alternatif Ai lebih terpilih. Setelah tujuan dan alternatif keputusan telah didapatkan, langkah selanjutnya adalah mengidentifikasi kumpulan kriteria.

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

UML(*Unified Modeling Language)* yang merupakan metodologi kolaborasi antara metoda booch, OMT (*Object Modeling Technique),* serta OOSE (*Oriented Sofware Engineering)* dan beberapa metoda lainnya, merupakan metodologi yang paling sering digunakan saat ini untuk mengadaptasi maraknya

penggunaan bahasa “pemrograman berorientasi objek” (OOP). (Adi Nugroho;2009;4)

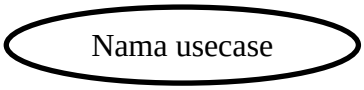
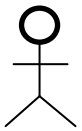
UML (*Unified Modeling Language*) adalah sebuah ”bahasa” yang telah menjadi standar dalam industry untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Seperti bahasa-bahasa lainnya, UML mendefenisikan notasi dan *sintax/semantic*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *sintax* mendefinisikan bagaimana bentuk – bentuk tersebut dapat dikombinasikan. *Unified Modeling Language* biasa digunakan untuk :

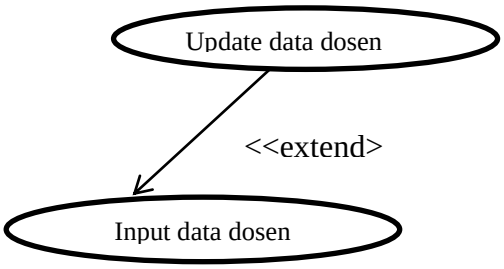
1. Menggambarkan batasan sistem dan fungsi – fungsi sistem secara umum, di buat dengan *use case* dan *actor*.
2. Menggambarkan kegiatan atau proses bisnis yang di laksanakan secara umum, di buat dengan *interaction diagrams*.
3. Menggambarkan representasi struktur *static* sebuah sistem dalam bentuk *class diagram*s.
4. Membuat model behavior “yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan *state transition diagrams*.
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development diagrams*.
6. Menyampaikan atau memperluas *fungsionality* dengan *stereotypes*. (Yuni Sugiarti; 2013 :36)

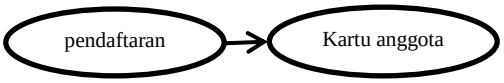
II.7.1. Use Case Diagram

Use case diagrams merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi – fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan diagram *use case*, yaitu *use case*, *actor* dan relasi. Berikut adalah sismbol – simbol yang ada pada diagram *use case*. (Yuni Sugiarti; 2013: 42)

Tabel II.1 Simbol – simbol pada Use Case Diagram

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antar unit atau *actor*; biasanya ditanyakan dengan menggunakan kata kerja di awal frase nama *use case*.
Aktor  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari *acto*r adalah gambar orang, tapi *actor* belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama *actor*.

Asosiasi/ *association* ———————	Komunikasi antara actor daan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan kator.
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 ditambahkan, arah panah menunjuk pada use case yang dituju. Contoh : 
Include <<include>> ——————→	Relasi use case tambahan kesebuah use case dimana use case yang 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 di use case, *include* berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan

	dijalankan, contoh :  <<include>

Sumber: (Yuni Sugiarti; 2013)

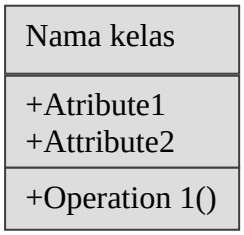
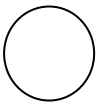
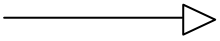
II.7.2. Class Diagram

Diagram kelas atau class diagram menggambarkan struktur sistem dari segi pendefenisian kelas – kelas yang akan di buat untuk membangun sistem. Kelas memiliki apa yang di sebut atribut dan metode atau operasi.

1. Atribut merupakan variabel- variabel yang di miliki oleh suatu kelas.
2. Atribut mendeskripsikan properti dengan sebaris teks di dalam kotak kelas tersebut.
3. Operasi atau metode adalah fungsi – fungsi yang di miliki oleh suatu kelas.

Diagram kelas mendeskripsikan jenis – jenis objek dalam sistem dan berbagai hubungan statis yang terdapat di antara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan – batasan yang terdapat dalam hubungan – hubungan objek tersebut. (Yuni Sugiarti; 2013: 57)

Tabel II.2 Simbol – simbol Class Diagram

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur sistem
Antarmuka / interface 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi 1 1..*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity.*
Asosiasi berarah/directed asosiasi	Relasi antar kelas dengan makna kelas yang satu di gunakan oleh kelas yang lain, asosiasi biasanya juga di sertai dengan *multiplicity*.
Generalisasi 	Relasi antar kelas dengan makna generalisasi – spesialisasi (umum khusus).
Kebergantungan / defedency	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	Relasi antar kelas dengan makna semua bagian (*whole-part*)

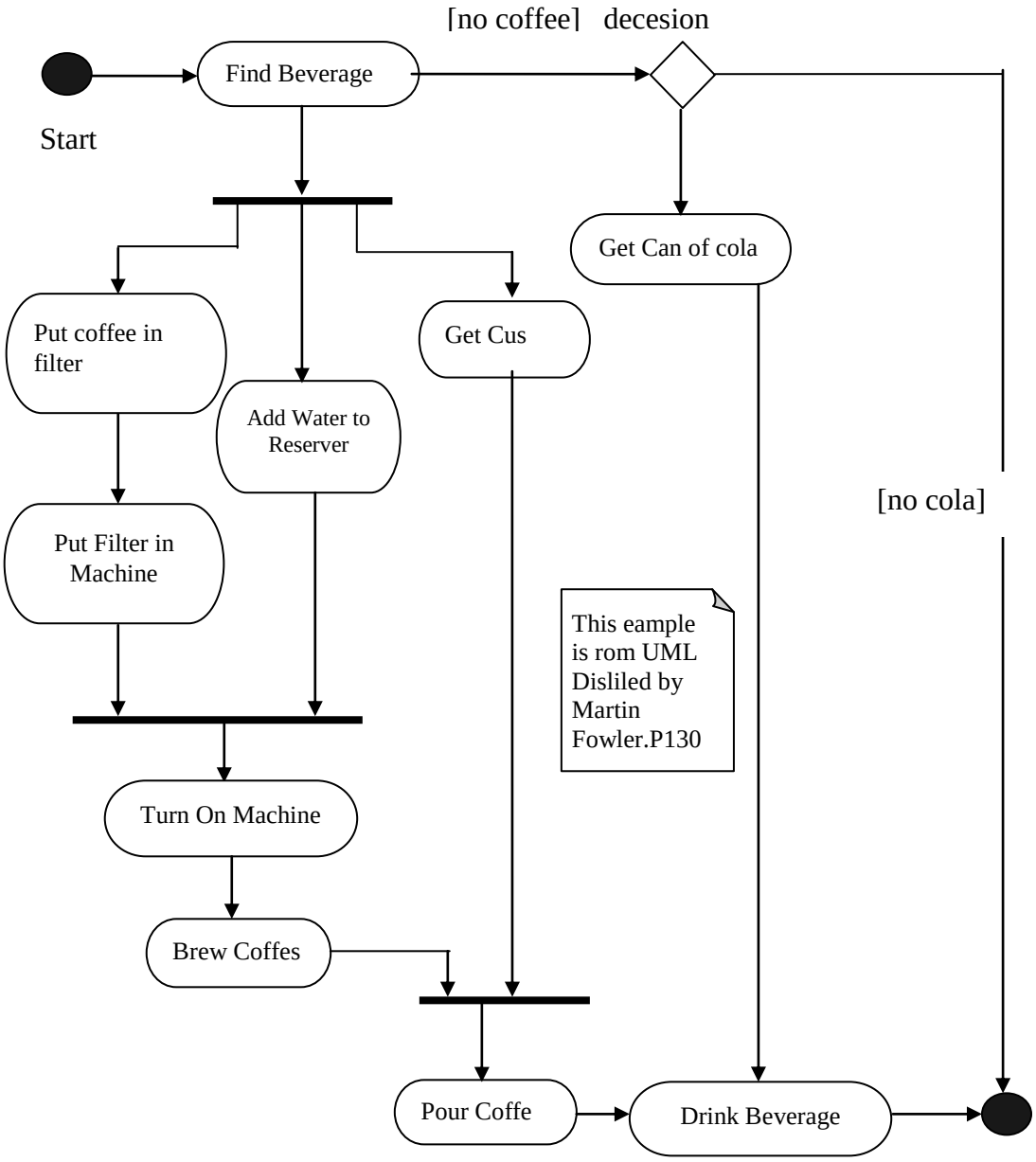
Sumber : (Yuni Sugiarti ; 2013)

II.7.3. Activity Diagram

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis.

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. (Yuni Sugiarti; 2013: 75)



Gamabar II.1 Activity Diagram

Sumber : (Yuni Sugiarti ; 2013)

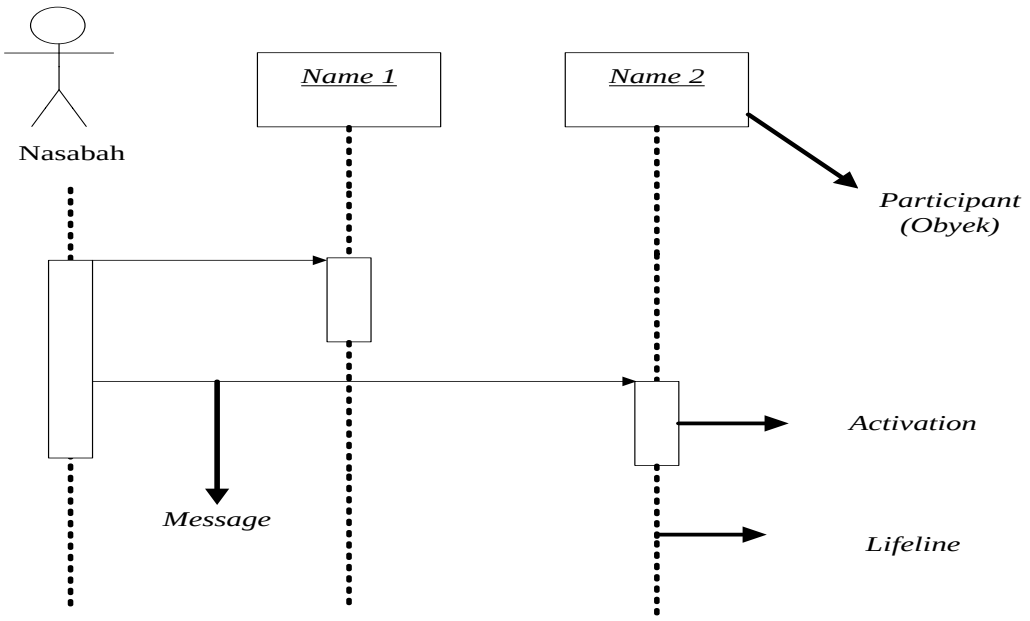
II.7.4. Sequence Diagram

Diagram sekuence menggambarkan kelakuan/ pelaku objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sequence maka harus diketahui objek – objek yang terlibat dalam sebuah use case beserta metode – metode yang dimiliki kelas yang diinstasiasi menjadi objek itu.

Diagram sequence memiliki ciri yang berbeda dengan diagram interaksi pada diagram kolaborasi sebagai berikut :

1. Pada diagram sequence terdapat garis hidup objek. Garis hidup objek adalah garis vertical yang mencerminkan eksistensi sebuah objek sepanjang periode waktu. Sebagian besar objek – objek yang tercakup dalam diagram interaksi akan eksis sepanjang durasi tertentu dari interaksi, sehingga objek – objek itu diletakkan dibagian atas diagram dengan garis hidup tergambar dari atas hingga bagian bawah diagram. Suatu objek lain dapat saja diciptakan, dalam hal ini garis hidup dimulai saat pesan *destroy*, jika kasus ini terjadi, maka garis hidupnya juga berakhir.
2. Terdapat focus kendali (*Focus Of Control*), berupa empat persegi panjang ramping dan tinggi yang menampilkan aksi suatu objek secara langsung atau sepanjang sub ordinat. Puncak dari empat persegi panjang adalah permulaan aksi, bagian dasar adalah akhir dari suatu aksi. Pada diagram ini mungkin juga memperhatikan penyaringan (*nesting*) dan *focus* kendali yang disebabkan oleh proses rekursif dengan menumpuk *focus* kendali yang lain pada induknya. (Yuni Sugiarti; 2013: 70)

Berikut simbol – simbol yang ada pada sequence diagram.



Gamabar II.2 Simbol Squence

Sumber : (Yuni Sugiarti ; 2013)

II.8. ERD *(Entity Relationship Diagram)*

Entity relationship diagram adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas – entitas dan menentukan hubungan antar entitas. Proses memungkinkan analis menghasilkan struktur basis data yang baik sehingga data dapat disimpan dan diambil secara efisien. Elemen – elemen diagram hubungan entitas yaitu :

1. Entitas (*Entity*)

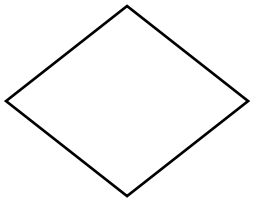
   Entitas dalah sesuatu yang nyata atau abstrak diman kita akan menyimpan data. Ada 4 kelas entitas, yaitu misalnya pegawai, pembayaran, kampus dan buku.

Gambar II.3 Simbol Entitas

Sumber : (Janner Simarmata,dkk; 2013)

2. Relasi (*Relationship*)

   Relasi adalah hubungan alamiah yang terjadi antara satu atau lebih entitas, misalnya proses pembayaran pegawai. Kardinalitas menentukan kejadian suatu entitas untuk satu kejadian pada entitas yang berhubungan. Misalnya mahasiswa bisa mengambil banyak mata kuliah.

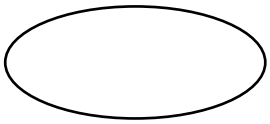


Gambar II.4 Simbol Relasi

Sumber : (Janner Simarmata,dkk; 2013)

3. Atribut (*Attribute*)

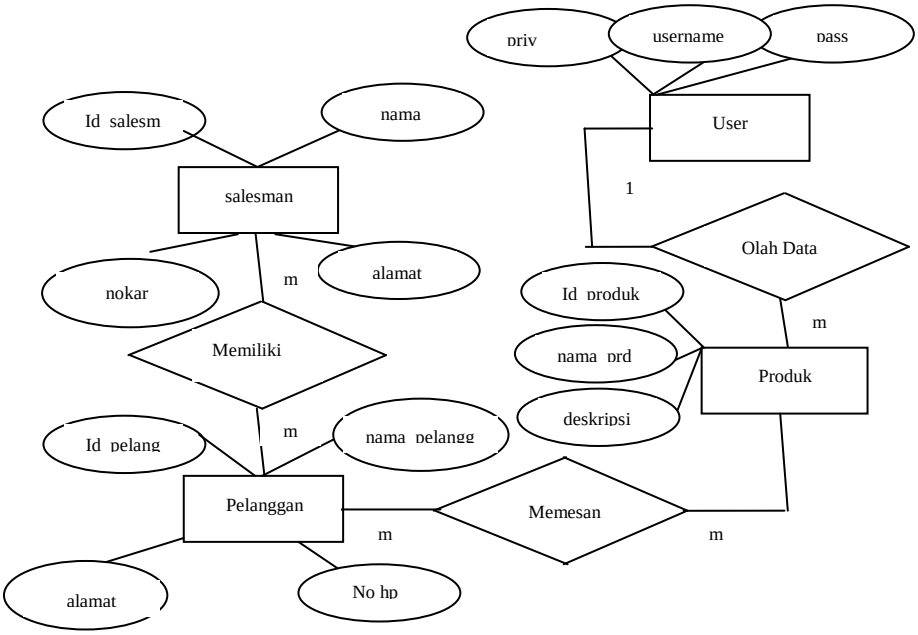
Atribut adalah ciri umum semua semua atau sebagian besar instansi pada entitas tertentu. Sebutan lain atribut adalah *property*, elemen data, dan *field*. Misalnya, nama, alamat, nomor pegawai, dan gaji adalah atribut entitas pegawai. Sebuah atribut atau kombinasi atribut yang mengidentifikasikan satu dan hanya satu instansi suatu entitas disebut kunci utama atau pengenal. Misalnya, nomor pegawai adalah kunci utama untuk pegawai.



Gambar II.5 Simbol Atribut

Sumber : (Janner Simarmata,dkk; 2013)

Berikut contoh ERD



II.9. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basisdata relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilang data yang terduplikasi dari tabel relasional (www.utexas.edu). (Janner Simarmata & dkk; 2010 : 77)

Normalisasi adalah bagian perancangan basis data.Tanpa normalisasi, sistem basis data menjadi tidak akurat, lambat, tidak efisien, serta tidak memberikan data yang diharapkan. Pada waktu menormalisasikan basis data, ada empat tujuan yang harus dicapai, yaitu:

1. Mengatur data dalam kelompok – kelompok sehingga masing – masing kelompok hanya mengenai bagian kecil sistem.
2. Meminimalkan jumlah data berulang dalam basis data.
3. Membuat basis data yang datanya diakses dan dimanipulasi secara cepat dan efisien tanpa melupakan integritas data.
4. Mengatur data sedemikian rupa sehingga ketika memodifikasi data, anda hanya mengubah pada satu tempat.

II.9.1. Bentuk – Bentuk Normalisasi

1. Bentuk normal pertama (1NF)

   Tahap ini dilakukan penghilangan beberapa group elemen yang berulang agar menjadi satu harga tunggal yang berinteraksi di antara setiap baris pada suatu tabel, dan setiap *atribut* harus mempunyai nilai data yang *atomi*c (bersifat *atomic value*).

2. Bentuk normal kedua (2NF)

   Normal kedua didasari atas konsep *full functioanl dependency* (ketergantungan fungsional sepenuhnya) yang dapat didefinisikan sebagai berikut.

   Jika A dan B adalah *atribut-atribut* dari suatu relasi, B dikatakan *full function dependency* (mimiliki ketergantungan fungsional sepenuhnya) terhadap A, jika B adalah tergantung fungsioal terhadap A, tetapi tidak secara tepat memiliki ketergantungan fungsional dari *subset* (himpunan bagian) dari A.

3. Bentuk normal ketiga (3NF)

   Jika kita hanya meng*update*satu baris saja, sementara baris yang lainnya tidak, maka data di dalam *database* tersebut akan *inkonsisten*/tidak teratur. Anomali *update* ini disebabkan oleh suatu ketergantungan transitif *(transitive dependency)*. Kita harus menghilangkan ketergantungan tersebut dengan melakukan normalisasi ketiga (3-NF).

4. Bentuk normal *boyce-code* (BCNF)

   Suatu relasi dalam basis data harus dirancang sedemikian rupa sehingga mereka memiliki ketergantungan sebagaian *(partial dependency)*, maupun ket-ergantungan transitif.

Berikut Contoh Normalisasi

1. Bentuk Normal Pertama (1NF/ *First Normal Form*)

   a. Tabel Normal Pertama

User name	Priv	Pass	id_ produk	nama produk	Deskripsi	Id_pelanggan	nama_ pelanggan	alamat	No_hp	Id_salesman	Nama_salesman	alamat	No_hp	nokar

   b. Tabel Normal Pertama user

User name	Priv	Pass

2. Bentuk Normal Kedua (2NF/ *Second Normal Form*)

   a. Tabel produk

id_produk	nama_produk	deskripsi

   b. Tabel pelanggan

id_pelanggan	Nama_pelanggan	alamat	No_hp

   c. Tabel salesman

id_salesman	nama_salesman	alamat	No_hp	nokar

3. Bentuk Normal Ketiga (3NF/ *Third Normal Form*)

   a. Tabel produk

id_produk*	nama_produk	deskripsi

b. Tabel pelanggan

id_pelanggan*	Nama_pelanggan	alamat	No_hp

c. Tabel salesman

id_salesman*	nama_salesman	alamat	No_hp	nokar

II.10. Kamus Data

Kamus data (*data dictionary*) dipergunakan untuk memperjelas aliran data yang digambarkan pada DFD. Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukkan (*input)* dan keluaran (*output*) dapat dipahami secara umum (memiliki standar cara penulisan). Kamus data biasanya berisi:

1. Nama - nama dari data
2. Digunakan pada – merupakan proses-proses yang terkait data
3. Deskripsi – merupakan deskripsi data
4. Informasi tambahan – seperti tipe data, nilai data, batas nilai data dan komponen yang membentuk data. (Rosa A.S & M Shalauddin; 2011 : 67)