BAB II

TINJAUAN PUSTAKA

II.1 Sistem

Definisi sistem berkembang sesuai dengan konteks dimana pengertian sistem itu digunakan. Berikut akan diberikan beberapa definisi sistem secara umum :

1. Kumpulan dari bagian-bagian yang bekerja sama untuk mencapai tujuan yang sama.
2. Sekumpulan objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa dilihat sebagai satu kesatuan yang dirancang untuk mencapai satu tujuan.

Dengan demikian, secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur atau variabel-variabel yang saling terorganisasi, saling berinteraksi dan saling bergantung satu sama lain (Hanif Al Fatta ; 2007 : 3).

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi yang saling berkerja sama membentuk satu kesatuan. Berikut ini adalah komponen – komponen sistem :

1. Batas sistem *(boundary)* merupakan daerah yang membatasi antara suatu sistem dengan sistem lainnya.

2. Lingkungan Luar Sistem *(environment)* dari suatu sistem adalah apapun yang diluar batas sistem yang mempengaruhi operasi sistem.
3. Penghubung sistem *(interface)* merupakan media pendukung antara satu subsistem dalam subsistem yang lainnya.
4. Masukan sistem *(input)* adalah energi yang dimasukkan kedalam sistem.
5. Keluaran sistem *(output)* adalah hasil dari sistem yang diolah dan diklasifikasikan menjadi keluaran yang berguna.
6. Pengolahan Sistem : suatu sistem mempunyai suatu bagian pengolahan yang akan merubah masukan menjadi keluaran.
7. Sasaran sistem : jika suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya.

II.1.1 Informasi

Informasi adalah hasil pemrosesan data yang diperoleh dari setiap elemen sistem tersebut menjadi bentuk yang mudah dipahami yang merupakan pengetahuan yang relevan yang dibutuhkan oleh orang untuk menambah pemahamannya terhadap fakta-fakta yang ada.

Kualitas Informasi :

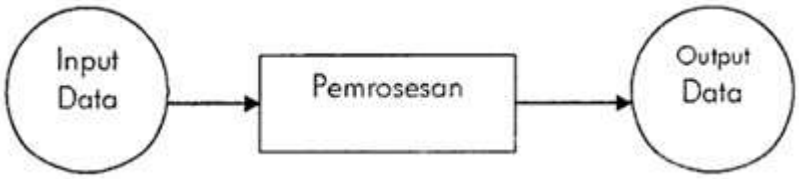
a. Keakuratan dan teruji kebenarannya
b. Kesempurnaan informasi
c. Tepat waktu
d. Relevansi
e. Mudah dan Murah.

Sumber dari informasi adalah data. Data merupakan bentuk jamak dari bentuk tunggal datum atau data item yang artinya ”catatan atau kenyataan”. Data diolah dari bahan mentah menjadi informasi yang dapat digunakan sebagai bukti ataupun bahan pertanggung jawaban. Informasi merupakan alat untuk membenarkan hal-hal yang tadinya kurang dimengerti, sehingga informasi ini sangat penting dalam suatu organisasi.

Informasi adalah hasil data yang telah diolah menjadi sebuah bentuk bagi penerimanya dan bermanfaat dalam mengambil keputusan saat ini atau mendatang (Hanif Al Fatta, 2007:9).

Dari pengertian di atas maka penulis dapat menyimpulkan bahwa informasi adalah data yang telah diolah menjadi bentuk yang lebih berguna, lebih berarti bagi yang menerimanya. Data yang telah diolah dan mempunyai makna tertentu disebut informasi yang dapat dipakai sebagai bahan pertimbangan dalam pengambilan keputusan. Informasi yang dihasilkan harus memenuhi syarat:

1. *Accuracy,* yaitu tepat dan akurat. Yaitu informasi harus benar dan diolah dari data yang benar.
2. *Timeles*, yaitu ada pada saat yang tepat. Yaitu informasi yang akan dipakai sebagai bahan pertimbangan untuk pengambilan keputusan tidak cukup hanya benar adanya, tetapi harus ada pada saat yang tepat.
3. *Completeness,* yaitu lengkap tetapi tidak berlebihan. Yaitu informasi yang dibutuhkan tidak boleh mengandung kekurangan atau tidak lengkap
4. *Conciseness*, yaitu informasi selain harus lengkap, juga ringkas tertera secara teratur, mudah dipahami agar tidak menimbulkan salah penelitian.



Gambar II.1 Konsep Sistem Informasi

Sumber : Hanif Al Fatta (2007 : 9)

II.1.2 Sistem Informasi

Sistem informasi dapat didefenisikan sebagai sekumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk menintegrasikan data, memproses dan menyimpan serta mendistribusukan informasi. Dengan kata lain, sistem informasi merupakan kesatuan elemen-elemen yang saling berinteraksi secara sistematis dan teratur untuk menciptakan dan membentuk aliran informasi yang akan mendukung pembuatan keputusan dan melakukan kontrol terhadap jalannya perusahan.

II.1.3 Sistem Informasi Akuntansi

Menurut Kieso dan Weygendt, akuntansi adalah suatu sistem informasi yang mengidentifikasi, mencatat, dan mengkomunikasikan kejadian ekonomi dari suatu organisasi kepada pihak yang berkepentingan (Adanan Silaban; 2009:7). Dengan melihat defenisi tersebut maka disimpulkan bahwa SIA (Sistem Informasi Akuntansi) adalah sebuah sistem yang mengumpulkan, mencatat, menyimpan, dan memproses data sehingga menghasilkan informasi yang berguna dalam membuat keputusan.

Menurut James. A. Hall (2007:10), Sistem informasi akuntansi memiliki tiga subsistem yaitu :

1. Sistem pemrosesan transaksi (*transaction processing system*), yaitu mendukung operasi bisnis harian melalui berbagai dokumen serta pesan untuk para pengguna diseluruh perusahaan.
2. Sistem buku besar/pelaporan keuangan (*general ledger/financial reporting system*), yaitu menghasilkan laporan keuangan seperti laporan laba/rugi, neraca, arus kas, pengembalian pajak serta berbagai laporan lainnya yang disyaratkan oleh hukum.
3. Sistem pelaporan manajemen (*management reporting system*), yaitu yang menyediakan pihak manajemen *internal* berbagai laporan keuangan bertujuan khusus serta informasi yang dibutuhkan untuk pengambilan keputusan seperti anggaran, laporan kinerja serta laporan pertanggungjawaban.

Sistem informasi akuntansi tidak lepas dari siklus akuntansi yang meliputi urutan siklus sebagai berikut:

1. Analisis transaksi bisnis, seperti pengumpulan bukti-bukti transaksi yang terjadi.
2. Menjurnal transaksi-transaksi tersebut.
3. Mem-*posting* jurnal tersebut ke buku besar (*general ledger*).
4. Menyiapkan neraca saldo.
5. Menjurnal dan mem-*posting* penyesuaian (jurnal penyesuaian).

6. Menyiapkan neraca penyesuaian.
7. Menyiapkan laporan keuangan, berupa laporan laba rugi (*income statement*), laporan perubahan modal (*statement of equity*) dan neraca saldo (*balance sheet*).
8. Menjurnal dan mem-*posting* penutup (jurnal penutup).
9. Menyiapkan neraca penutup.

II.2 Akun

Akun adalah sarana yang digunakan untuk mencatat transaksi atau peristiwa. Informasi yang terperinci dari tiap-tiap laporan keuangan ada dalam akun. Akun untuk masing-masing perusahaan berbeda, tergantung dari jenis transaksi, besar perusahan, tipe bisnis dan kompleksitas masing-masing perusahaan. Semua akun dapat dikelompokkan menjadi dua jenis yaitu :

1. Kesinambungan/rill/ permanent(Akun Neraca). Akun ini akan dibawa dari suatu period eke periode selanjutnya.
2. Berumur satu periode/nominal(akun laba rugi). Akun sementara yang akan ditutup pada suatu periode. Menunjukkan posisi pendapatan dan biaya. Disebut juga akun modal sementara (*temporary capital account*).

II.2.1 Jurnal

Transaksi dicatat berdasarkan urutan kronologisnya dalam sebuah jurnal sebelum ditransfer pada rekeningnya. Sebuah jurnal dibuat untuk tiap transaksi menunjukkan saldo debit dan kredit yang mempengaruhi rekening tertentu. Berdasarkan frekuensi terjadinya jurnal dibedakan menjadi 2 yaitu :

1) Jurnal Umum

Mencatat transaksi yang frekuensi terjadinya jarang atau nonrutin seperti membayar pinjaman, penyesuaian di akhir periode, dan jurnal penutup. Jurnal umum ini meliputi :

a) Jurnal penyesuaian
b) Jurnal koreksi
c) Jurnal penutup

2) Jurnal Khusus

Mencatat transaksi yang frekuensi terjadinya sering/tinggi, jurnal khusus menyederhanakan proses pencatatan transaksi yang terjadi berulang dalam jumlah besar. Jurnal khusus meliputi :

a) Jurnal penjualan
b) Jurnal pembelian

II.2.2 Buku Besar (*General Ledger*)

Buku besar adalah catatan akhir yang merupakan kumpulan rekening neraca dan rugi/laba yang merangkum catatan akuntansi. Jumlah rekening yang digunakan oleh perusahaan tergantung pada jenis perusahaan, Besar kecilnya perusahaan dan informasi yang dibutuhkan. Bila ruang lingkup perusahaan sudah cukup luas, akan terdapat jumlah transaksi yang cukup besar. Dalam hal ini catatan dalam rekening buku besar perlu keterangan lebih rinci maka dilakukan dengan cara membuat buku pembantu yang berbentuk rekening-rekening, sehingga dapat dikatakan buku pembantu adalah rincian rekening-rekening dalam buku besar.

Setiap jurnal yang sudah dibuat dipindahkan ke dalam buku besar (*general ledger*) sesuai dengan kelompok rekeningnya. Contoh :

Seperti pada contoh jurnal ,

Kas → Masuk ke buku besar kas disisi debet.

Piutang → Masuk ke buku besar piutang di sisi kredit

II.3 Microsoft Visual Studio 2008

Visual basic merupakan salah satu paket bahasa pemrograman dari visual studio 2008. *Visual studio* 2008 merupakan sebuah *software* untuk membuat aplikasi windows. Semenjak visual studio .NET, Microsoft telah banyak melakukan pengembangan dan perubahan pada tampilan software ini.

II.3.1 Visual Basic 2008

VB (*Visual Basic*) merupakan bahasa pemrograman yang populer dikalangan programer karena kemudahan pemakaian dan juga memiliki fitur-fitur yang sangat handal dalam mengembangkan aplikasi. VB 2008 merupakan kelanjutan dari versi sebelumnya yaitu VB 2005. visual basic merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi *windows*. *Visual basic* 2008 atau *visual basic* 2009 adalah versi terbaru yang telah diluncurkan oleh Microsoft bersama *C#, visual C++,* dan *visual web developer* dalam satu paket *Visual studio 2008.*

Visual basic merupakan aplikasi pemrograman yang menggunakan teknologi *.Net Framework*. Teknologi *.Net Framework* merupakan komponen *windows* yang terintegrasi serta mendukung pembuatan, penggunaan aplikasi dan halaman

web. Teknologi *.Net Framework* mempunyai 2 komponen utama, yaitu CLR (*Common Language Runtime*) dan *Class Library*. CLR digunakan untuk menjalankan aplikasi yang berbasis .Net, sedangkan *library* adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi (Wahana Komputer, 2010:2).

II.4 Microsoft SQL Server 2008

SQL Server merupakan suatu *Relational Database Management Sysetem* (RDBMS) yang digunakan untuk menyimpan data. Data yang disimpan dalam database bisa dalam skala kecil maupun besar. SQL Server merupakan suatu DBMS yang sangat *powerful* sebagai penyimpanan data transaksi untuk keperluan bisnis.

SQL Server 2008 memiliki *resource* yang mumpuni dalam mengolah data. Selain itu, pada perangkat ini dapat mengimplementasikan beberapa fitur baru yang dapat meningkatkan aktivitas dan performa database.

SQL Server 2008 menyimpan data dengan konsep DBMS dan penyajiannya merupakan penyajian level fisik karena langsung menyimpan data pada database dengan kondisi sebenarnya, yaitu disimpan pada tabel, kolom, dan menggunakan data type saat penyimpanan.

II.4.1 Konsep database.

Database merupakan tempat pentimpanan data. Data disimpan pada suatu server yang bisa diolah untuk keperluan tertentu. Pengimplementasian database dapat dilakukan secara terdistribusi dan tersentralisasi. Terdistribusi merupakan suatu konsep yang menerapkan lebih dari satu database. Dan tersentralisasi

menerapkan satu database secara terpusat. Pada database terdapat tiga tingkatan atau level data, yaitu :

1. Level Fisik (*Phisycal Level*).

   Level fisik merupakan level paling rendah karena menggambarkan bagaimana data disimpan pada kondisi yang sebenarnya pada server.

2. Level Konsepsual (*Conceptual Level*).

   Merupakan level yang menggambarkan data yang disimpan pada database dan menjelaskan secara keseluruhan hubungan antar data tersebut.

3. Level Pandangan (*View Level*).

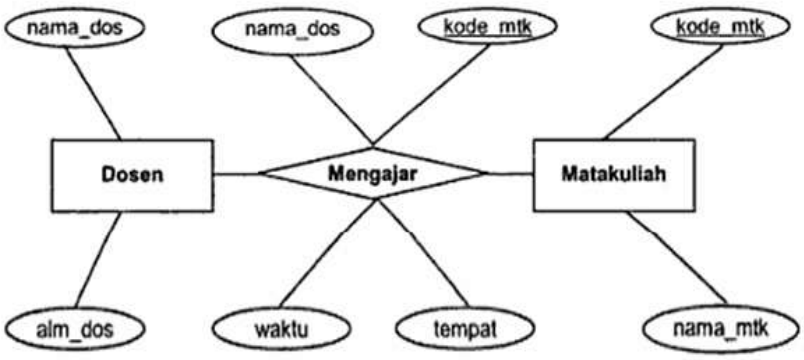
   Merupakan level yang menggambarkan sebagian dari seluruh database sesuai dengan kebutuhan pengguna.

II.4.2 ERD (*Entity Relationship Diagram*)

ERD digunakan untuk menggambarkan secara sistematis hubungan antar *entity-entity* yang ada dalam suatu sistem database menggunakan simbol-simbol sehingga mudah dipahami (Yuhefizard, 2008:17). Simbol yang digunakan adalah:

1. Persegi panjang, berfungsi untuk menyatakan suatu *entity*.
2. *Elips,* berfungsi untuk menyatakan *attribute*. Jika diberi garis bawah menandakan bahwa *attribute* tersebut merupakan *attribute/field* kunci.
3. Belah ketupat, menyatakan jenis relasi.
4. Garis, penghubungan antara relasi dengan *entity* dan antara *entity* dengan *attribute*.

Contoh : hubungan antara *entity* dosen dengan *entity* matakuliah, sebagai berikut ini :



Gambar II.2 Contoh ERD

Sumber : Yuhefizard, 2008:17

II.4.3 Hubungan Antar Tabel.

Pada relasional database dapat didefenisikan hubungan antar table yaitu sebagai berikut :

1. Satu Ke Satu (*One To One*).

   Merupakan hubungan antar tabel dimana suatu tabel hanya memiliki suatu data pada tabel yang direferensikan.

2. Satu Ke Banyak (*One To Many*).

   Merupakan hubungan antar tabel dimana suatu tabel memiliki lebih dari satu data pada tabel yang direferensikan.

3. Banyak Ke Banyak (*Many To Many*).

   Merupakan hubungan antar tabel dimana suatu tabel memiliki lebih dari satu data yang direferensikan pada masing-masing tabel.

II.4.4 Normalisasi

Normalisasi merupakan cara pendekatan dalam membangun desain logika basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standart untuk menghasilkan struktur tabel yang normal (Kusrini, 2007:40). Pada dasarnya desain logika basis data relasional dapat menggunakan prinsip normalisasi maupun transformasi dari model E-R ke bentuk fisik.

Dalam perspektif normalisasi sebuah database dikatakan baik jika semua tabel yang membentuk basis data sudah berada dalam keaadaan normal. Suatu tabel dikatakan normal apabila :

1) Ada dekomposisi/penguraian tabel, maka dekomposisinya dijamin aman (*lossless-join decomposition*).
2) Terpeliharanya ketergantungan functional pada saat perubahan data (*dependency preservasion*).
3) Tidak melanggar *Boyce Code Normal Form* (BCNF), jika tidak bisa minimal tidak melanggar bentuk normal ketiga.

Kondisi-kondisi yang diujikan pada proses normalisasi adalah sebagai berikut :

1) Menambah data.
2) Mengedit.
3) Menghapus.
4) Membaca.

II.4.5 Bentuk-bentuk Normalisasi

Menurut Kusrini (2007: 41), bentuk-bentuk normalisasi adalah sebagai berikut ini :

1) Bentuk tidak normal.

   Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi.

2) Bentuk normal tahap pertama (*1st normal form*).

   Sebuah tabel dapat dikatakan 1NF jika:

   a) Tidak ada baris yang duplikat dalam tabel tersebut.
   b) Masing-masing cell bernilai tunggal.

3) Bentuk normal tahap kedua (2nd normal form).

   Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua *attribute* yang tidak termasuk dalam *primary key* memiliki ketergantungan fungsional pada *primary key* secara utuh. Sebuah tabel tidak memenuhi 2NF, jika ketergantungannya hanya bersifat parsial atau tergantung pada sebagian *primary key*.

4) Bentuk normal tahap ketiga (3rd normal form).

   Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X-> A, dimana A mewakili semua atribut tunggal didalam tabel yang tidak ada didalam X, maka :

   a) X haruslah *superkey* pada tabel tersebut.

b) Atau A merupakan bagian dari primary key pada tabel tersebut.

5) Bentuk normal tahap keempatdan kelima.

   Bentuk normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

6) *Boyce code normal form* (BCNF).

   Suatu tabel dikatakan BNCF jika memenuhi 1NF dan relasi harus bergantung fungsi pada atttribut *superkey*.

II.5 Variabel dan Type data

Dalam dunia pemrograman komputer, variable dan tipe data pasti akan ditemui pada bahasa pemrograman apapun, karena keduanya sangat besar kegunaannya dan sangat penting bagi pembuatan sebuah aplikasi. Variable dan tipe data akan saling berhubungan tidak dapat dipisahkan. Variable berguna untuk menyimpan nilai sementara untuk dapat dipergunakan kembali. Dikatakan sementara waktu karena nilai sebuah variable akan disimpan dalam memori komputer yang bersifat tidak permanent. Untuk menggunakan sebuah variable, harus dilakukan pendeklarasian variable tersebut. Aturan dalam memberikan nama sebuah variable adalah sebagai berikut ini :

1. Harus diawali dengan huruf alphabet, tidak boleh angka.
2. Tidak boleh lebih dari 255 karakter.
3. Tidak diperkenankan untuk mendeklarasikan dua buah variable dengan nama yang sama.

4. Tidak boleh menggunakan *keyword visual basic* yaitu kata-kata yang dipergunakan oleh *visual basic module, integer*.

Tipe data adalah jenis nilai yang tersimpan dalam variable, huruf, angka ataupun tanggal. Tipe data diperlukan agar *visual basic* dapat langsung mengenal jenis data yang tersimpan dalam variable. Berikut ini adalah jenis tipe data yang didukung oleh visual basic :

1. Boolean, tipe data ini hanya boleh diisi oleh dua buah nilai yaitu true (benar) dan false (salah).
2. Byte, 0 sampai dengan 255
3. Char, tipe data ini hanya boleh diisi oleh sebuah karakter (Unicode).
4. Date, tipe data visual basic yang merupakan nilai sebuah tanggal dan waktu, dengan jangkauan tanggal 1 januari 00001 sampai dengan 31 desember 9999.
5. Decimal, 0 sampai dengan +/- 79.228.162.514.264.337.593.543.-950.335 (tanpa bilangan decimal dibelakang koma) atau 0 sampai dengan +/- 7, 9228162514264337593543950335 (dengan bilangan decimal dibelakang koma maksimal 28 angka).
6. Double, -1,79769313486231570E+308 sampai dengan 1, 79769313486231570E+308 (untuk bilangan positif).
7. Integer, -2.147.483.648 sampai dengan 2.147.483.647.
8. Long, -9.223.372.036.854.775.808 sampai dengan 9.223.372.-036.854.775.807.
9. Sbyte, -128 sampai dengan 127.

10. Short, -32,768 sampai dengan 32.767.
11. Single, -3,4028235E+38 sampai dengan -1,401298E-45(untuk bilangan negative) atau 1,401298E-45 sampai dengan 3,4028235E+38 (untuk bilangan positif).
12. String, 0 sampai dengan 2 juta karakter (*Unicode*) bisa huruf, angka atau karakter yang tidak umum lainnya.
13. UInteger, 0 sampai dengan 4.294.967.295.
14. ULong, 0 sampai dengan 18.446.744.073.709.551.615 (1.8…E+19).
15. UShort, 0 sampai dengan 65.535.

II.6 UML

UML (*Unified Modelling Language)* adalah sebuah bahasa standard untuk pengembangan sebuah software yang dapat menyampaikan bagaimana membuat dan membentuk model-model, tetapi tidak menyampaikan apa dan kapan model yang seharusnya dibuat yang merupakan salah satu proses implementasi pengembangan software (Adi Nugroho; 2010:6).

UML adalah metode pemodelan secara visual sebagai sarana untuk merancang dan membuat software berorientasi objek karena UML merupakan bahasa visual untuk pemodelan bahasa berorientasi objek, maka semua elemen dan diagram berbasiskan pada paradigm objek oriented.

UML tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung duhubungkan keberbagai bahasa pemograman seperti VISUAL BASIC , C++, Visual Basic atau bahkan dihubungkan secara langsung kedalam sebuah object oriented database. Begitu juga mengenai pen-

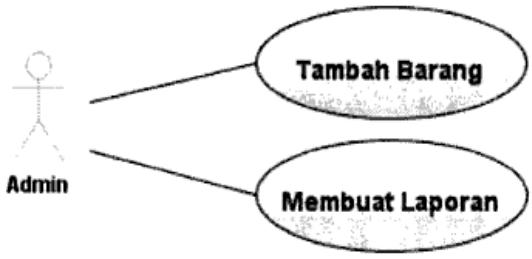
dokumentasian dapat dilakukan seperti requitments, arsitektur, design, source, project plan, tests dan prototypes.

II.6.1 Diagram UML

Diagram berbentuk grafik yang menunjukkan symbol elemen model yang disusun untuk mengilustrasikan bagaian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya dialokasikan untuk view tertentu. UML mendefinisikan diagram-diagram berikut ini :

1. *Use case diagram.*

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara *actor* dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Gambar II. 3Berikut ini adalah contoh *use case* diagram.

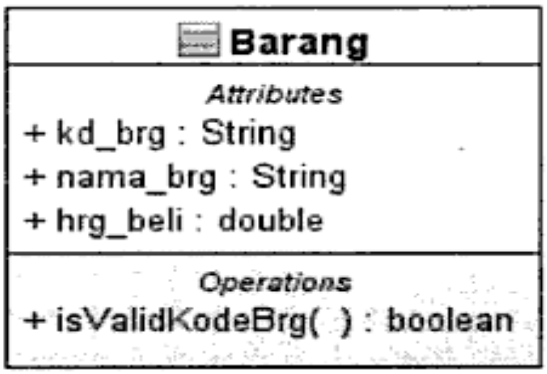


Gambar II.3 Gambar Use Case

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:138

2. *Class diagram.*

   Menggambarkan struktur statis *class* didalam sistem. *Class* merepresentasikan sesuatu yang ditangani oleh sistem. *Class* dapat berhubungan dengan yang lain melalui cara: terhubung satu sama lain (*associated)*, satu *class* tergantung/menggunakan *class* yang lain (*dependent*), satu *class* merupakan spesialisasi dari *class* lainnya (*Specialed)*, grup bersama sebagai satu unit (*Package)*.

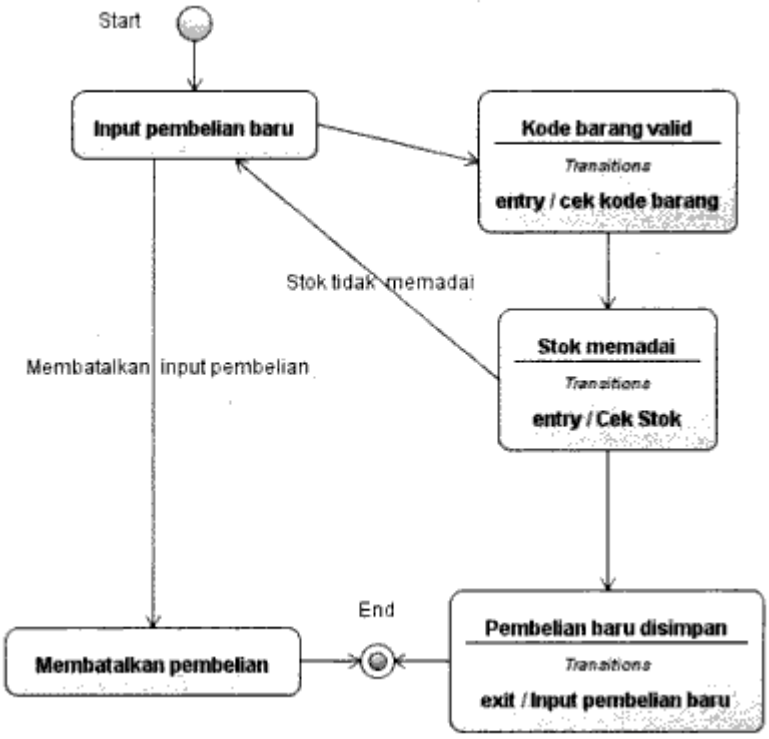


Gambar II.4 Contoh *Class* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:139

3. *Statechart diagram*

Menggambarkan semua state (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan state berubah. Keadaian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class.* Hanya yang mempunyai sejumlah *state* yang terdefenisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda. Gambar II. 5 berikut ini adalah contoh *statechart* diagram.



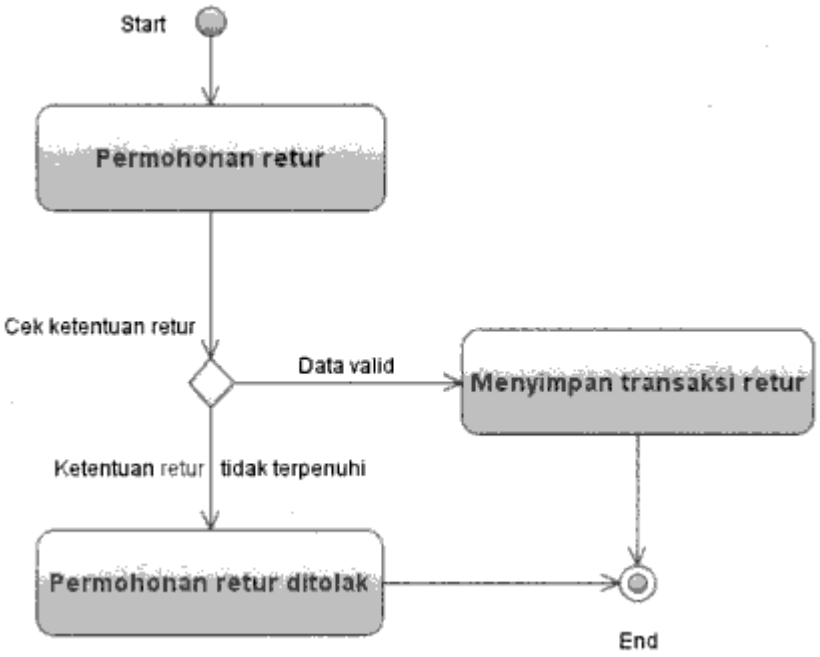
Gambar II.5 Statechart Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:141

4. *Activity diagram*

Menggambarkan rangkaian aliran dari aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga

dapat juga digunakan untuk aktifitas lainnya seperti *use case* dan interaksi. Gambar II.6 berikut ini adalah contoh *activity* diagram.

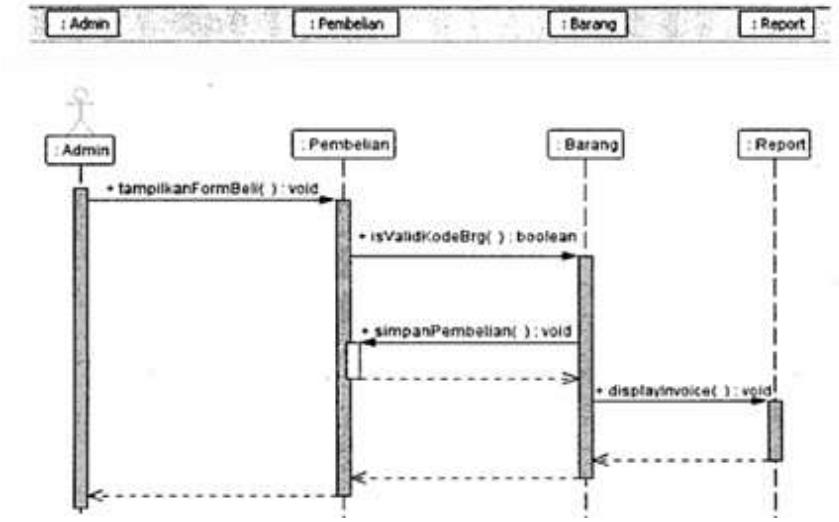


Gambar II.6 Activity Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:142

5. *Sequence diagram*

Menggambarkan kolaborasi dinamis antara sejumlah *object*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

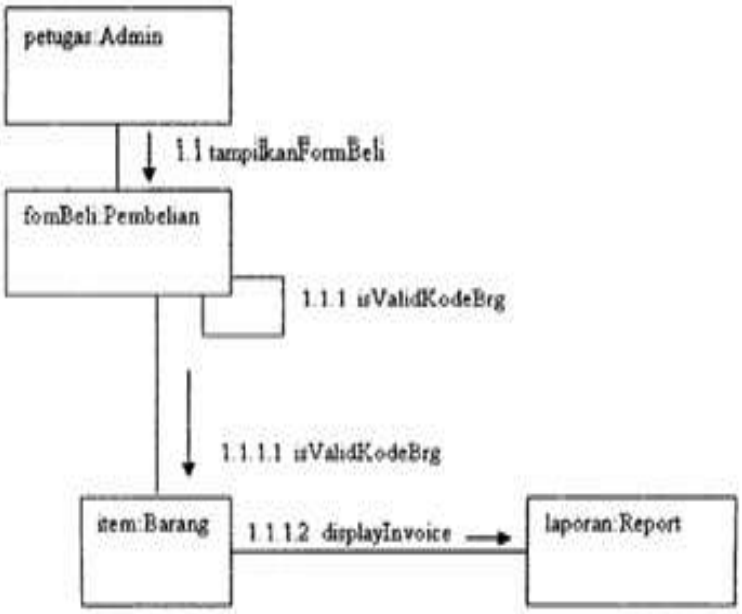


Gambar II.7 Contoh *Sequence* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:143

6. *Collaboration diagram*

Menggambarkan kolaborasi dinamis seperti *sequence diagram.* Dalam menunjukkan pertukaran pesan, *collaboration diagram* menggambarkan *object* dan hubungannya. Jika penekanannya pada waktu atau urutan maka digunakan *sequence diagram*, tapi jika penekanannya pada konteks digunakan *collaboration diagram*.

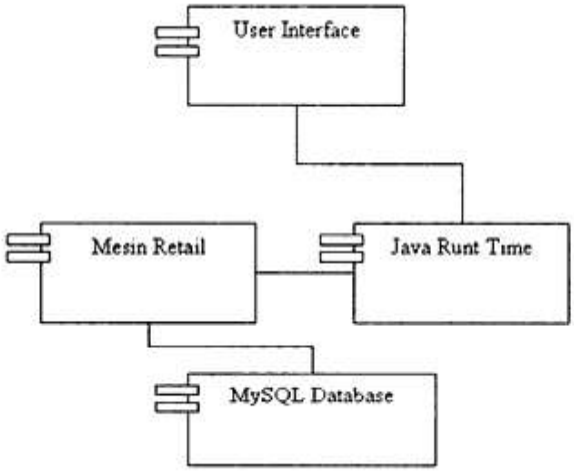


Gambar II.8 Contoh *Collaboration* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:144

7. Component diagram

Menggambarkan struktur fisik kode dari komponen. Dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*.

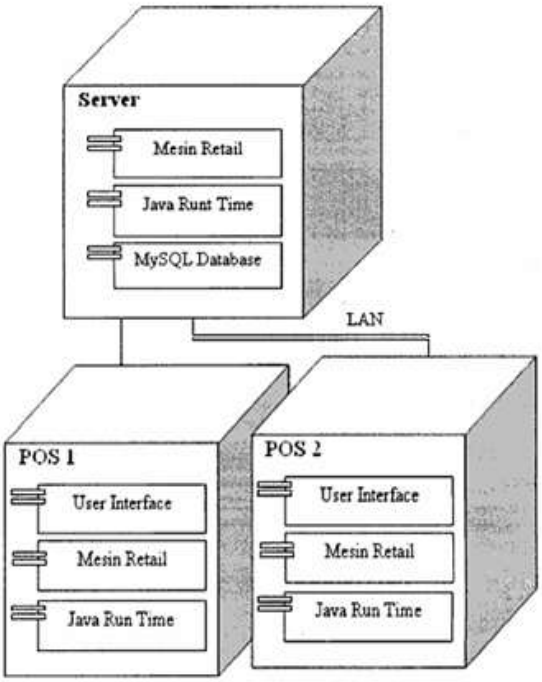


Gambar II.9 Contoh *Component* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:145

8. Deployment diagram

Menggambarkan arsitektur fisik dari perangkat keras dan perangkat lunak sistem, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungan nya. Didalam *nodes, executable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh node tertentu dan ketergantungan komponen.



Gambar II.10 Contoh *Deployment* Diagram

Sumber : Miftakhul Huda & Bunafit Komputer, 2010:146

II.6.2 Tujuan penggunaan UML

Tujuan penggunaan UML adalah sebagai berikut :

1. Memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakn konsep berorientasi *object*.
2. Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.