BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu (Jogiyanto; 2008 : 2)

II.2. Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya (Jogiyanto; 2008 : 8).

Informasi adalah data yang telah diklasifikasikan diolah atau diinterpretasi untuk digunakan dalam proses pengambil keputusan. Sistem pengolahan informasi megolah data menjadi informasi atau tepatnya mengolah dari bentuk tak berguna menjadi berguna bagi penerimanya (Tata Sutabri; 2008 :23).

II.3. Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan (Jogiyanto; 2008 : 11)

Sistem informasi adalah berupa suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian yang mendukung operasi yang bersifat manajerial dengan kegiatan strategi suatu

organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Tata Sutabri; 2008 :42).

II.4. Sistem Informasi Akuntansi

Akutansi merupakan bahasa dari bisnis. Setiap perusahaan menerapkanya sebagai alat komunikasi. Secara klasik akutansi merupakan proses pencatatan (*recording*), pengkelompokkan (*classifiying*), perangkuman (*summarizing*) dan pelaporan (*reporting*) dai kegiatan transaksi perusahaan. Tujuan akhir dari kegiatan akutansi adalah penerbitan laporan-laporan keuangan. Laporan-laporan keuangan adalah merupakan suatu informasi. Sistem informasi yang berbasis pada komputer sekarang dikenal dengan istilah sistem informasi akuntansi atau SIA (*accounting information system* atau AIS). Sistem informasi akuntansi SIA didefenisikan oleh Stephen A. Moscove dan Mark G. Simkin sebagai berikut ini. SIA adalah merupakan komponen operasi yang mengumpulkan, mengklasifikasikan, memproses, menganalisis, mengkomunikasikan informasi pengambilan keputusan dengan orentasi financial yang relevan bagi pihak-pihak dalam perusahaan (secara prinsip adalah manajemen). Menurut Robert G. Murdick, Thomas C. Fuller dan Joel E. Ross : SIA adalah kumpulan kegiatan-kegiatan dari organisasi yang bertanggung jawab untuk menyediakan informasi keuangan dan informasi yang didapatkan dari transaksi data untuk tujuan pelaporan internal kepada manajer untuk digunakan dalam pengendalian dan perencanaan sekarang dan operasi masa depan serta pelaporan eksternal kepada pemegang saham, pemerintah dan pihak-pihak luar lainnya (Jogiyanto; 2008 : 17).

II.4.1 Siklus Akuntansi

Siklus akuntansi adalah tahap-tahap kegiatan mulai dari terjadinya transaksi sampai dengan penyusunan laporan keuangan sehingga siap untuk pencatatan.

Transaksi periode berikutnya.

Tahap Pencatatan:

1.Pembuatan atau penerimaan bukti transaksi.

2.Pencatatn dalam jurnal (buku harian).

3.Pemindah-bukuan (*posting*) ke buku besar.

Tahap Pengikhtisaran:

4.Pembuatan neraca saldo (*trial balance*).

5.Pembuatan neraca lajur dan junal penyesuaian (*adjustment*).

6.Penyusunan laporan keuangan.

7.Pembuatan jurnal penutup (*closing entries*).

8.Pembuatan neraca saldo penutup (*post closing trial balance*).

9.Pembuatan jurnal balik (*reversing entries*). (Soemarso, 2008:90)

II.5. Laporan Keuangan

Laporan keuangan (financial statement) adalah laporan yang menyampaikan informasi keuangan yang dipercaya kepada pihak yang berkepentingan.

Berdasarkan Standar Akuntansi Keuangan (SAK) tujuan laporan keuangan adalah :

1. Menyediakan informasi yang menyangkut posisi keuangan, kinerja, serta perubahan posisi keuangan suatu perusahaan yang bermanfaat bagi sejumlah besar pemakai dalam pengambilan keputusan.
2. Untuk memenuhi kebutuhan bersama sebagian besar pemakai informasi termasuk menyediakan informasi yang mungkin dibutuhkan pemakai dalam pengambilan keputusan secara umum yang menggambarkan pengaruh keuangan dari kejadian dimasa lalu (Soemarso; 2009 : 11)

II.6. Kamus Data

Kamus data (KD) atau data *dictionary* (DD) yang disebut juga dengan *system dictionary* data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan demikian KD, analisis sistem dapat mendefenisikan data yang mengalir di sistem dengan lengkap. KD dibuat pada pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis, KD dapat digunakan sebagai alat komunikasi antara analisis sistem dengan pemakai sistem tentang data yang mengalir di sistem. Pada tahap perancangan sistem KD digunakan untuk merancang input, merancang laporan-laporan dan database. KD dibuat berdasarkan arus data yang ada di DAD. Arus data di DAD sifatnya global, hanya ditunjukkan nama arus datanya saja.

II.6.1. Isi Kamus Data

KD harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. Untuk maksud keperluan ini maka KD harus memuat hal-hal berikut ini :

1. Nama Arus Data

   Karena KD dibuat berdasarkan arus data yang mengalir di DAD, maka nama dari arus data juga harus dicatat di KD, sehingga mereka yang membaca DAD dan memerlukan penjelasan lebih lanjut tentang suatu arus data tertentu di DAD dapat langsung mencarinya dengan mudah di KD.

2. Alias

   Alias atau nama lain dari data dapat dituliskan bila nama lain ini ada. Alias perlu ditulis karena data yang mempunyai nama yang berbeda untuk orang atau departemen satu dengan yang lainnya. Misalnya bagian pembuat faktur dan langganan menyebut bukti penjualan sebagai faktur, sedang bagian gudang menyebutnya sebagai tembusan permintaan persediaan barang. Baik faktur dan tembusan permintaan persediaan ini mempunyai struktur data yang sama tetapi mempunyai struktur yang berbeda.

3. Bentuk data

   Telah diketahui bahwa arus data mengalir.

   a. Dari kesatuan luar ke suatu proses, data yang mengalir ini biasanya tercatat di suatu dokumen atau formulir.

b. Hasil dari suatu proses ke kesatuan luar, data yang mengalir biasanya terdapat di media laporan atau *query* tampilan layar atau dokumen hasil cetakan komputer.
c. Hasil dari suatu proses yang lain, data yang mengalir biasanya dalam bentuk variabel atau parameter yang dibutuhkan oleh proses penerimanya.
d. Hasil dari suatu proses yang direkamkan ke simpanan data, data yang mengalir ini biasanya berbentuk suatu variabel.
e. Dari simpanan data dibaca oleh suatu proses, data yang mengalir ini biasanya berupa suatu *field* (item data).

Dengan demikian bentuk dari data yang mengalir dapat berupa :

1. Dokumen dasar atau formulir
2. Dokumen hasi cetakan komputer
3. Laporan tercetak.
4. Tampilan dilayar monitor.
5. Variabel.
6. Parameter
7. *Field*

Bentuk dari data ini perlu dicatat di KD, karena dapat digunakan untuk mengkelompokan KD ke dalam kegunaanya, sewaktu perancangan sistem KD yang mencatat data yang mengalir dalam bentuk dokumen dasar atau formulir yang digunakan untuk merancang bentuk input sistem. KD yang mencatat data yang mengalir dalam bentuk laporan tercetak dan dokumen hasil cetakan komputer akan digunakan untuk merancang *output* yang akan dihasilkan oleh

sistem. KD yang mencatat data yang mengalir dalam bentuk tampilan layar monitor akan digunakan juga untuk merancang tampilan layar yang akan dihasilkan oleh sistem. KD yang mencatat data yang mengalir ke dalam bentuk parameter dan variabel akan digunakan untuk merancang proses dari program. KD yang mencatat data yang mengalir ke dalam bentuk dokumen, formulir, laporan, dokumen cetakan komputer, tampilan di layar monitor, variabel, dan *field* akan digunakan untuk merancang database.

4. Arus data

   Arus data menunjukkan dari mana data yang mengalir dan kemana data akan menuju. Keterangan arus data ini perlu dicatat di KD supaya memudahkan mecari arus data di DAD (Jogiyanto; 2008 : 725-727)

II.7. Basis Data (*Database*)

Basis data menurut Stephen dan Plew (2000) adalah mekanisme yang digunakan untuk menyimpan informasi atau data.

Ramakrishnan dan Gehkre (2003) menyatakan basis data sebagai kumpulan data, yang umumnya mendeskripsikan aktivitas satu organisasi atau lebih yang berhubungan (Abdul Kadir; 2008 :1).

1. Keuntungan DBMS (*Database Management System*)

   DBMS memungkinkan perusahaan maupun pengguna individu untuk :

   a. Mengurangi perulangan data

      Apabila dibandingkan dengan *file-file* komputer yang disimpan terpisah di setiap lokasi komputer. DBMS mengurangi jumlah total

file dengan menghapus data yang terduplikasi di berbagai *file.* Data terduplikasi selebihnya dapat ditempatkan dalam satu *file.*

b. Mencapai independensi data

Spesifikasi data disimpan dalam skema pada tiap program aplikasi. Perubahan dapat dibuat pada struktur data tanpa mempengaruhi program yang mengakses data.

c. Mengintergrasikan data beberapa *file*

Saat *file* dibentuk sehingga menyediakan kegiatan logis, maka organisasi fisik bukan merupakan kendala. Organisasi logis, pandangan pengguna, dan program aplikasi tidak harus tercemin pada media.

d. Mengambil data dan informasi dengan cepat.

Hubungan-hubungan logis, bahasa manipulasi data, serta bahasa *query* memungkinkan pengguna mengambil data dalam hitungan detik atau menit.

e. Meningkatkan keamanan

DBMS *mainframe* maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi (*password*), direktori pemakai, dan bahasa sandi (*encryption*) sehingga data yang dikelola akan lebih aman.

2. Kerugian DBMS

Keputusan menggunakan DBMS mengikat perusahaan atau pengguna untuk :

a. Memperoleh perangkat lunak

   DBMS *mainframe* masih sangat mahal. Walaupun harga DBMS berbasis komputer mikro lebih murah, tetapi tetap merupakan pengeluaran besar bagi suatu oraganisasikecil.

b. Memperoleh konfigurasi perangkat keras yang besar

   DBMS sering memerlukan kapasitas penyimpanan dan memori lebih besar dari pada program aplikasi lain.

c. Mempekerjakan dan mempertahankan staf DBA

   DBMS memerlukan pengetahuan khusus agar dapat memanfaatkan kemampuannya secara penuh. Pengetahuan khusus ini disediakan paling baik oleh para pengguna basis data (DBA)

Baik basis data terkomputerisasi maupun DBMS bukanlah prasyarat untuk memecahkan masalah. Namun, keduanya memberikan dasar-dasar mengunakan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pengguna (Abdul Kadir; 2008 : 8-9).

II.8. *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modelling Langguage* yang berarti bahasa pemodelan standart. (Chonoles; 2003 : 6) mengatakan sebagai bahasa, berarti *UML* memiliki sintaks dan *semantic*. Ketika kita membuat model menggunakan konsep *UML* ada aturan–aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat harus berhubungan satu dengan lainnya harus mengikuti standart yang ada. *UML* bukan hanya sekedar diagram, tetapi juga

menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana sistem mengatasi error yang terjadi? Bagaimana keamanan terhadap sistem yang ada kita buat? Dan sebagainya dapat dijawab dengan *UML*.

UML diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

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

UML telah diaplikasikan dalam investasi perbankan,lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, retail, sales, dan supplier.

Blok pembangunan utama *UML* adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorentasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat komunikasi yang konsisten dalam mensuport para pengembang sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai *UML*, baik kita sendiri yang membuat sekedar membaca diagram *UML* buatan orang lain (Prabowo Pudji Widodo, Herlawati; 2011 : 6-7).

II.8.1. Diagram-*Diagram UML*

Beberapa literatur menyebutkan bahwa *UML* menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram Kelas. Bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi, serta relasi-relasi diagram. Diagram ini umu dijumpai pada pemodelan sistem berorentasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.
2. Diagram Paket (*Package Diagram*) bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas merupakan bagian dari diagram komponen.
3. Diagram *Use Case* bersifat statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram Interaksi dan *Sequence* (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam waktu tertentu.

5. Diagram Komunikasi (*Communication Diagram*) bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* yang menekankan organisasi *structural* dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram)* bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*State)*, transisi kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi dan terutam penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram Aktivitas (*Activity Diagram*) bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan member tekanan pada aliran kendali antar objek.
8. Diagram Komponen (*Component Diagram*) bersifat statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan diagram kelas dimana komponen dipetakan kedalam satu atau lebih kelas-kelas. Antarmuka-antarmuka serta kolaborasi-kolaborasi.
9. Diagram *Deployment* (*Deployment Diagram)* bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run time)*.

Memuat simpul-simpul berserta komponen-komponen yang ada di dalamnya. Diagram *Deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing).*

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada *UML* dimungkinkan kita menggunakan diagram-diagram lainya misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya (Probowo Pudji Widodo, Herlawati; 2011 : 10-12).

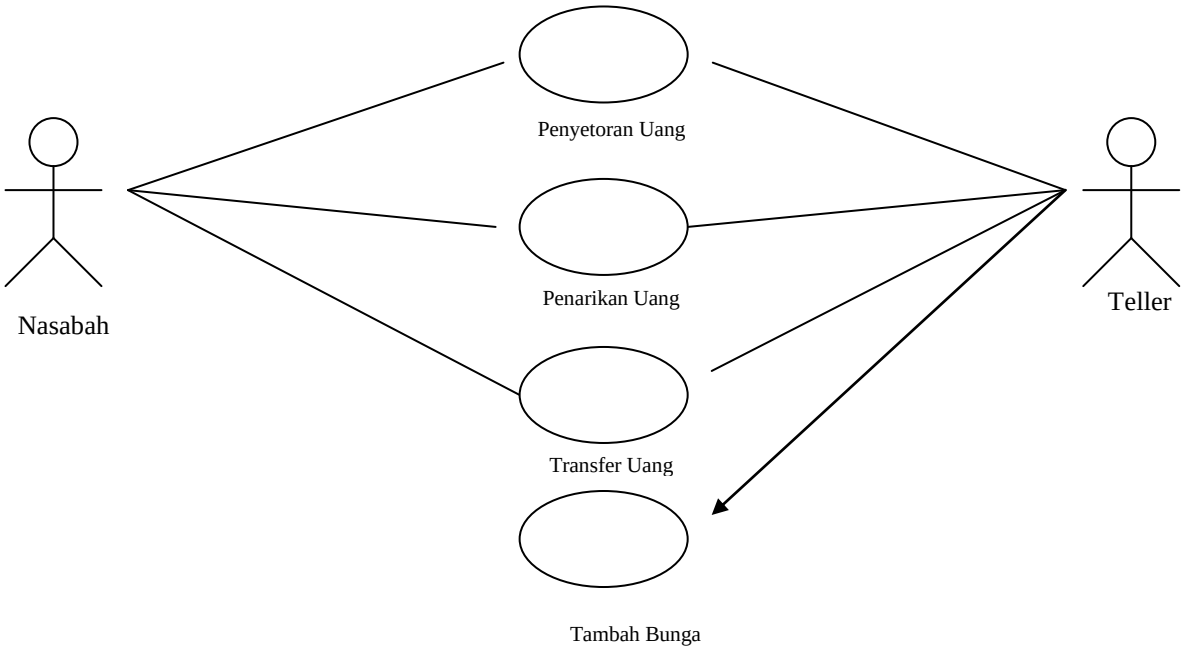
1. *Diagram Use Case (Use Case Diagram)*

   Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut Pooley (2005:15) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak indentik dengan model karena model lebih luas dari diagram.

   komponen pembentuk diagram *use case* adalah :

   a. Aktor (*actor),* menggambarkan pihak-pihak yang berperan dalam sistem.
   b. *Use Case,* aktivitas/ sarana yang disiapkan oleh bisnis/sistem.
   c. Hubungan (*Link),* aktor mana saja yang terlibat dalam *use case* ini.

   Gambar II.2. di bawah ini merupakan salah satu contoh bentuk diagram *use case* (Prabowo Pudji Widodo, Herlawati; 2011 : 15-16).

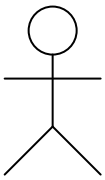


Gambar II.1. Diagram *Use Case*

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:17)

2. Aktor

   Menurut Chonoles (2003 :17) menyarankan sebelum mebuat use case dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder* untuk melihat gambar aktor, lihat pada gambar II.3. sebagai berikut :

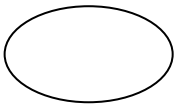


Gambar II.2. Aktor

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:17)

3. *Use Case*

Menurut Pilone (2005 : 21) *use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut Whitten (2004 : 258) mengartikan *use case* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario) baik terotomatisasi maupun secara manual, untuk tujuan melengkapi satu tugas bisnis tunggal. *Use case* digambarkan dalam bentuk *ellips*/oval untuk melihat gambar simbol *use case*, lihat pada gambar II.4. sebagai berikut :



Gambar II.3. Simbol *Use Case*

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:22)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu Chonoles (2003:22-23) menawarkan cara untuk menghasilkan *use case* yang baik yakni :

a. Pilihlah Nama Yang Baik

Use case adalah sebuah *behaviour* (prilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detil tambahkan kata benda mengindikasikan dampak aksinya terhadap suatu kelas objek. Oleh karena itu diagram *use case* seharusnya berhubungan dengan diagram kelas.

b. Ilustrasikan Perilaku Dengan Lengkap.

Use case dimulai dari inisiasi oleh aktor primer dan berakhir pada aktor dan menghasilkan tujuan. Jangan membuat *use case* kecuali anda mengetahui tujuannya. Sebagai contoh memilih tempat tidur (*King Size, Queen Size,* atau dobel) saat tamu memesan tidak dapat dijadikan *use case* karena merupakan bagian dari *use case* pemesanan kamar dan tidak dapat berdiri sendiri (tidak mungkin tamu memesan kamar tidur jenis king tapi tidak memesan kamar hotel).

c. Identifikasi Perilaku Dengan Lengkap.

Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, use case harus lengkap. Ketika memberi nama pada *use case*, pilihlah frasa kata kerja yang implikasinya hingga selesai. Misalnya gunakan frasa *reserve a room* (pemesanan kamar) dan jangan *reserving a room* (memesan kamar) karena memesan menggambarkan perilaku yang belum selesai.

d. Menyediakan *Use Case* Lawan (*Inverse)*

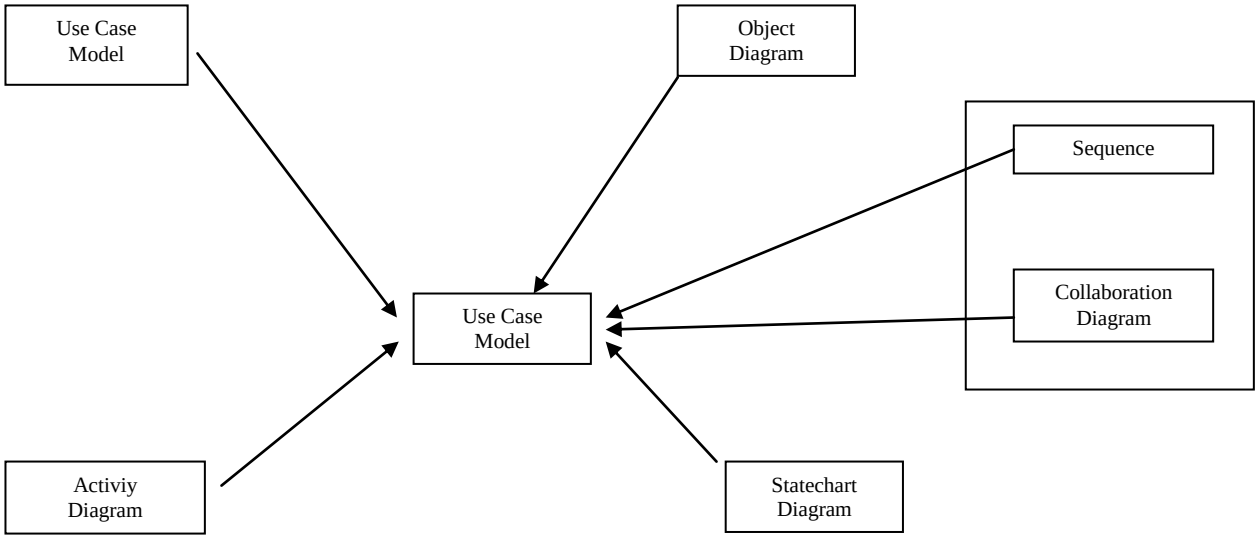
Kita biasanya membutukan *use case* yang membatalkan tujuan, misalnya pada u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

e. Batasi Use Case Hingga Satu Perilaku Saja.

Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah use case kita hanya fokus pada satu hal. Misalnya, penggunaan *use case check in* dan *check out* dalam satu *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda (Prabowo Pudji Widodo, Herlawati; 2011 : 22-23)

4. Diagram Kelas (*Class Diagram*)

   Diagram kelas adalah inti dari proses pemodelan objek. Baik *forward engineering* maupun *reverse engineering* memanfaatkan diagram ini *forward engineering* adalah proses perubahan model menjadi kode program sedangkan *reverse engineering* sebaliknya merubah kode program menjadi model (Probowo Pudji Widodo, Herlawati; 2011 : 37) untuk melihat gambar hubungan diagram kelas dengan diagram *uml* lainya, lihat pada gambar II.5. sebagai berikut :



Gambar II.4. Hubungan Diagram Kelas Dengan Diagram *UML* lainya

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011 : 38)

5. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (Prabowo Pudjo Widodo, Herlawati ;2011 : 143-145)

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langka sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi. Contoh aksinya yaitu :

a. Fungsi Matematika
b. Pemanggilan Perilaku
c. Pemrosesan Data

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classfier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classfier*, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan model proses bisnis, informasi itu biasanya disebut *process-relevant data*. Aktivitas

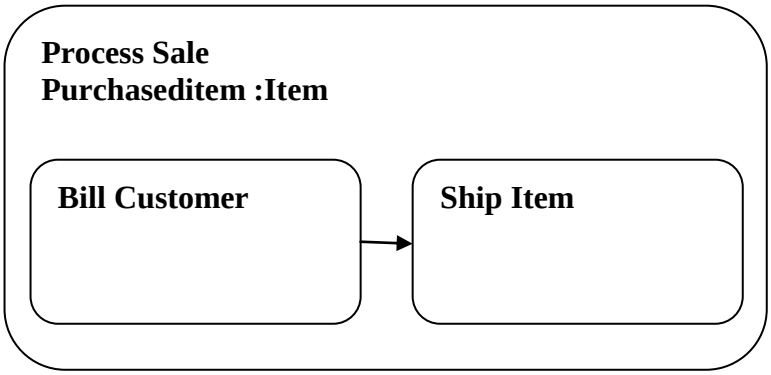
diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya *specific* dan digunakan hanya untuk aktivitas tertentu.

Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya. Untuk melihat gambar aktivitas serderhana tanpa rincian, lihat pada gambar II.6. sebagai berikut :

Gambar II.5. Aktivitas Serderhana Tanpa Rincian

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:145)

Detail aktivitas dapat dimasukan di dalam kotak. Aksi diperlihatkan dengan symbol yang sama dengan aktivitas dan namanya diletakkan didalam persegi panjang. Untuk melihat gambar aktivitas dengan detail rincian, lihat pada gambar II.7. sebagai berikut :



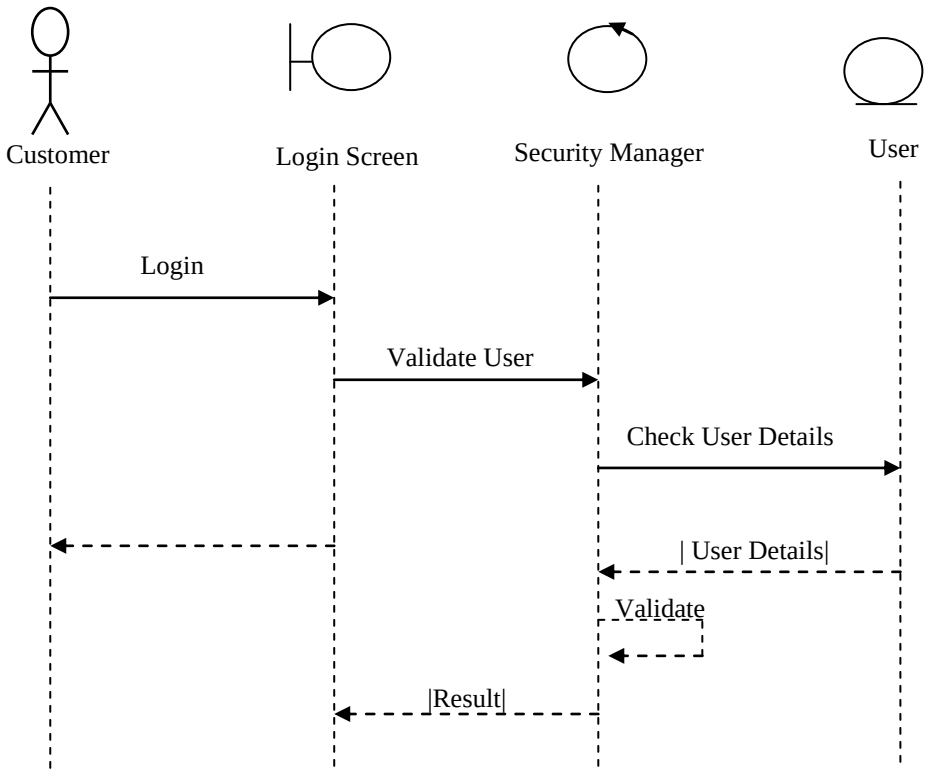
Gambar II.6. Aktivitas Dengan Detail Rincian

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:145)

6. *Sequence Diagram*

   Menurut Douglas (2004 : 174) menyebutkan ada tiga diagram primer UML dalam memodelkan scenario interaksi, yaitu diagram urutan (*sequence diagram)*, diagram waktu (*timing diagram)* dan diagram komunikasi (*communication diagram).*

   Menurut Pilone (2005 : 174) menyatakan bahwa diagram yang paling banyak dipakai adalah diagram urutan. Gambar II.8. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nantinya (Prabowo Pudji Widodo, Herlawati; 2011 : 174-175)



Gambar II.7. Diagram Urutan

(Sumber : Prabowo Pudjo Widodo, Herlawati; 2011:175)

II.9. *MYSQL*

Mysql adalah salah satu software sistem manajemen *database* (DBMS) *structured Query Language* (*SQL*) yang bersifat *open source*. *SQL* adalah bahasa standart untuk mengakses *database* dan didefenisikan dengan standart *ANSI/ISO SQL*. *MYSQL* dikembangkan, disebarluaskan, dan didukung oleh *MYSQL AB. MYSQL AB* adalah perusahaan komersial yang didirikan oleh pengembang *MYSQL*. *MYSQL* merupakan aplikasi *Relational Database Management System* (RDBMS) yang digunakan untuk aplikasi *client* server atau sistem *embedded*.

Mysql mempunyai beberapa sifat yang menjadikannya sebagai salah satu software database yang banyak digunakan oleh pemakai diseluruh dunia. Sifat-sifat yang dimiliki oleh *MYSQL* antara lain :

a. *Mysql* merupakan DBMS (*Database Management System*)

b. *Database* adalah kumpulan data yang terstruktur. Data dapat berupa daftar belanja, kumpulan gambar, atau yang lebih luas yaitu informasi jaringan perusahaan. Agar dapat menambah, mengakses, dan memproses data tersimpan pada sebuah komputer database, kita membutuhkan sistem manajement *database* (DBMS) seperti *MYSQL Server*. Sejak komputer sangat baik menangani sejumlah besar data, *sistem manajement database* (DBMS) memainkan peran utama dalam perhitungan baik sebagai peralatan yang berdiri sendiri maupun bagian aplikasi.

c. *Mysql* merupakan RDBMS (*Relational Database Management System*).

d. *Database relational* menyimpan data pada table-tabel yang terpisah, bukan menyimpan data dalam ruang penyimpanan yang besar. Hal ini menambah kecepatan *fleksibelitas*.

e. *Mysql* merupakan *software open source*.

f. *Open source* berarti setiap orang dapat menggunakan dan mengubah software yang bersangkutan. Setiap orang dapat mendownload *software MYSQL* dari internet dan menggunakan tanpa membayar. Bahkan jika mengkehendakinya anda bisa mempelajari kode sumber dan mengubah sesuai yang anda butuhkan (Wahana Komputer; 2008 : 26-27).

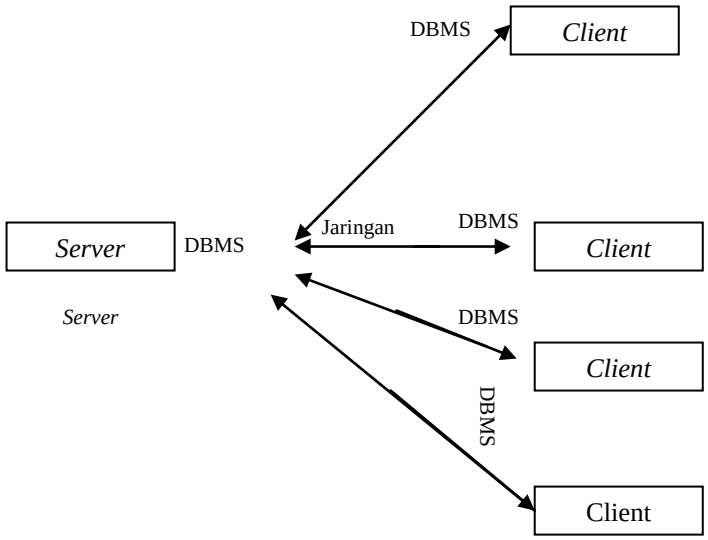
II.9.1. *Client Server*

Client server adalah satu model komunikasi 2 komputer atau lebih yang berfungsi melakukan pembagian tugas. *Client* bertugas untuk melakukan input, update, dan menampilkan data sebuah *database*. Sementara *server* bertugas untuk menyediakan pelayanan untuk melakukan manajemen yaitu : menyimpan dan mengolah *database* (Wahana Komputer; 2008 : 5).

II.6.2. Arsitekur *Client* Server

1. Arsitektur StandAlone (1-*Tier*)

   Model pertama aplikasi pemograman *database client server* adalah standalone atau 1 *tier* (1 -tingkat) adalah sebuah komputer yang mengakses sebuah *database* dari komponen sendiri. Dengan kata lain, aplikasi antarmuka *user* dan aplikasi DBMS terdapat pada komputer yang sama.



Gambar II.9. Arsitektur StandAlone (1-*Tier*)

Sumber : (Wahana Komputer; 2008 : 6)

Adapun karakteristik arsitektur 1 tier sebagai berikut :

a. Beban jaringan menjadi tinggi karena yang diminta adalah file database secara keseluruhan pada komputer *server client* melalui jaringan.

b. Setiap komputer pada jaringan harus mempunyai DBMS tersendiri untuk menyimpan hasil salinan dari *server* sehingga mengurangi sumber daya yang dimiliki oleh komputer *client* terutama *memory*.

c. Komputer *client* harus mempunyai kemampuan proses yang tinggi untuk mendapatkan waktu respon yang baik saat komputer *server* mengirimkan *file* yang diminta.

d. Arsitektur 1-*tier* cocok untuk bisnis kecil yang hanya membutuhkan data sebuah komputer untuk memproses dan menyimpan data sekaligus, tetapi

kurang tepat diterapkan pada model jaringan (Wahana Komputer; 2008 : 7).

2. Arsitekur 2-*Tier*

Model kedua sebuah pemograman *database* adalah model 2-tier. Arsitektur pada model demikian membagi tugas antara komputer *client* server. Komputer *client* bertugas menyediakan antar muka untuk *user*, permintaan (*request data*) ke DBMS Server, serta pemrosesan data (mencakup logika penyajian data, logika pemrosesan data, dan logika atau bisnis) komputer *client* hanya mengirimkan sebuah *statement* untuk menambah (*insert*) data, mengubah (*update*), menghapus *(delete*), dan yang terakhir meminta *(select*) data untuk ditampilkan melalui antarmuka yang dibuat oleh *programmer*. Sedangkan *server* bertanggung jawab terhadap penyimpanan, pengelolaan, melayani permintaan akses data, dan pemrosesan *client*.

Karakteristik arsitektur 2-*tier* adalah :

a. 2-*tier* terjadi pada jaringan dan melakukan pemodelan programan *database* dalam 2 tingkat. Tingkat pertama adalah *client* dan tingkat kedua adalah *server*.

b. Tingkat pertama komputer *client* sebagai penyedia aplikasi antarmuka untuk mengolah *database*, baik menampilkan data kedalam *user interface*, menambah, menghapus data, maupun logika bisnis (*bussines logic*)

c. Tingkat kedua adalah *server* yang menyediakan aplikasi untuk mengelola *database* serta menyediakan pula *query stored procedure*, dan *triggers*, yang dapat dipanggil *client* untuk mengolah data.

d. Komputer *client* hanya mengirimkan sebuah *statement sql* untuk meminta data ke *server*.

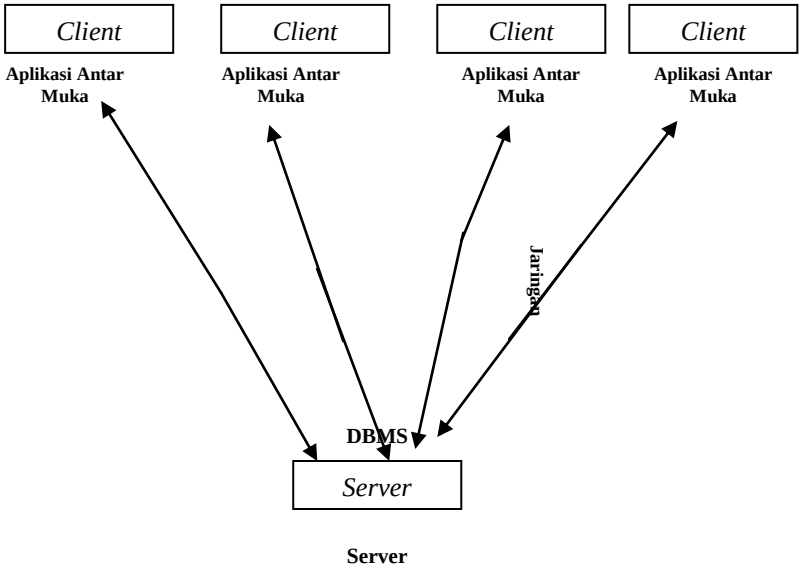
e. *Server* hanya memberikan data yang diminta melalui *statement* bersangkutan.

f. Komputer *server* dituntut untuk memiliki kemampuan pemrosesan yang tinggi karena harus melayani permintaan banyak komputer *client* yang mengakses satu atau lebih DBMS.

g. Beban jaringan menjadi ringan karena data yang berjalan pada jaringan hanya data yang diminta oleh *client*.

h. Otentifikasi pemakai, pemeriksaan integritas, dan pemeliharaan kamus data dilakukan pada sisi *server.*

i. Sederhana dan mudah untuk diterapkan, khususnya pada bisnis kecil yang hanya terdapat pada satu gedung (Wahana Komputer; 2008 : 7-8).



Gambar II.10. Arsitekur 2-Tier

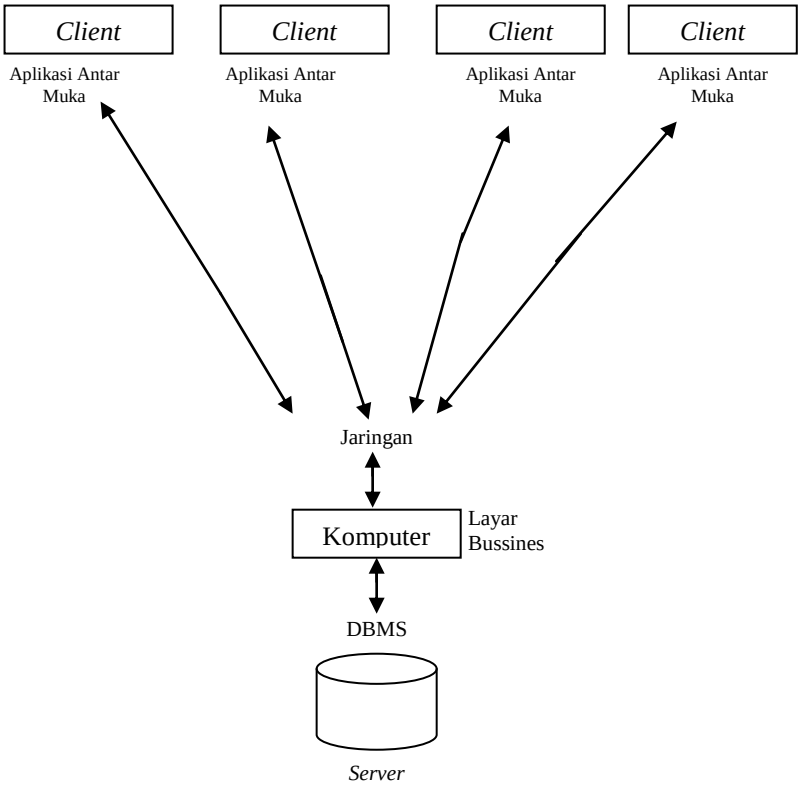
Sumber : (Wahana Komputer; 2008 : 8)

3. Arsitekur N-*Tier*

Arsitektur n-*tier* berarti membagi komponen menjadi n entitas yaitu 1 tier *client* dan n-1 *tier server*. Seperti pada model sebelumnya *client* bertugas menyediakan antarmuka aplikasi, sedangkan bertugas menyediakan data. Pada model n-tier (sebagai contoh adalah 3-tier), server dibagi 2 menjadi, yaitu satu *server* yang dipakai sebagai *bussines object* (*middle tier*) dan satu *server* yang hanya menyimpan *database* (*server tier).*

Secara nyata model 3-*tier* adalah pada jaringan internetyang hanya memanfaatkan *database*. Internet lapisan pertama adalah komputer *client* yang menampilkan halaman *web*, tempat kontent atau data halaman *web* berasal dari *database*. Lapisan kedua adalah *web* atau *HTTP server* yang menterjemahkan *script server side* (*PHP, JSP, ASP*, dan lainnya) dari

komputer *client* untuk meminta data pada *database*. Kemudian lapisan ketiga adalah komputer *database server* yang menyediakan *database* yang diminta oleh *web* atau HTTP *server* (Wahana Komputer; 2008 : 6-9)



Gambar II.11. Arsitekur N-Tier

Sumber : (Wahana Komputer; 2008 : 9)