BAB II
LANDASAN TEORI

1. Konsep Sistem informasi

a. Konsep Dasar Informasi

Informasi adalah hasil dari pengolahan data dalam bentuk yang lebih berguna dan berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian yang nyata yang digunakan untuk pengambilan keputusan.

Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian-kejadian (*event*) adalah suatu yang terjadi pada saat tertentu. Kesatuan nyata (*fact*) adalah berupa suatu obyek nyata seperti tempat, benda atau orang yang benar-benar ada dan terjadi.

Informasi adalah data yang diolah menjadi bentuk yang berguna bagi para pemakainya. Data yang diolah saja tidak cukup dapat dikatakan sebagai suatu informasi. Untuk menjadi suatu informasi, maka data yang diolah tersebut harus berguna bagi pemakainya.

Untuk dapat berguna, maka informasi harus didukung oleh tiga pilar, yaitu sebagai berikut :

1) Akurat (*Accurate*) Akurat berarti informasi harus bebas dari kesalahan-kesalahan dan tidak menyesatkan. Sehingga informasi benar-benar nyata dan tidak berubah fungsinya. Informasi juga

tepat pada saat penyampaiannya. Selain itu informasi harus memiliki unsur edukatif dan berimbang sekali. Akurat juga berarti informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerima informasi kemungkinan banyak terjadi gangguan (*noise*) yang dapat merubah atau merusak informasi tersebut.

2) Tepat pada waktunya (*TimeLiness*) Tepat pada waktunya berarti informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi karena informasi merupakan landasan didalam pengambilan keputusan.
3) Relevan (*Relevance*) Relevan berarti informasi tersebut mempunyai manfaat untuk pemakianya. Relevansi informasi untuk tiap-tiap orang berbeda. Nilai informasi bagi seorang pemakai ditentukan oleh keandalan. Keluaran yang tidak didukung oleh ketiga pilar ini tidak dapat dikatakan sebagai informasi yang berguna, tetapi merupakan sampah (*garbage*).

b. Konsep Sistem Informasi

Data merupakan bentuk yang mentah yang perlu diolah lebih lanjut untuk menghasilkan informasi yang digunakan dalam mengambil keputusan dan melakukan suatu tindakan. Data dapat berbentuk simbol-simbol berupa angka, huruf, gambar dan sebagainya.

Tujuan dari sistem informasi adalah menghasilkan informasi. Dan informasi (information) adalah data yang diolah menjadi bentuk yang berguna bagi para pemakainya.

2. Analisa dan Perancangan Sistem Beroientasi Obyek Dengan UML

a. UML (Unified Modeling Language)

Unified Modelling Language (UML) adalah sebuah “bahasa” yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi obyek. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya : Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modelling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Tujuan utama UML diantaranya adalah untuk :

1) Memberikan model yang siap pakai, bahasa permodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2) Memberikan bahasa permodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3) Menyatukan praktek-praktek terbaik yang terdapat dalam permodelan.

   Cakupan UML diantaranya : Pertama, UML menggabungkan konsep BOOCH, OMT, dan OOSE, sehingga UML merupakan suatu bahasa permodelan tunggal yang umum dan digunakan secara luas oleh para user ketiga metode tersebut dan bahkan para user metode lainnya. Kedua, UML menekankan pada apa yang dapat dikerjakan dengan metode-metode tersebut. Ketiga,

UML berfokus pada suatu bahasa permodelan standar, bukan pada proses standar.

Untuk membuat suatu model, UML mendefinisikan diagram-diagram berikut ini :

a) Use Case Diagram
b) Class Diagram
c) Behaviour Diagram
d) Statechart Diagram
e) Activity Diagram
f) Interaction Diagram
g) Sequence Diagram
h) Collaboration Diagram
i) Component Diagram
j) Deployment Diagram

Dalam menganalisa dan merancang sistem yang ditulis dalam tugas akhir ini, penulis tidak menggunakan semua diagram, hanya beberapa saja yang dibutuhkan seperti *use case diagram*, *activity diagram*, *class diagram* dan beberapa lagi yang digunakan pada masing-masing proses analisa dan perancangan.

b. Analisa Sistem Beroientasi Obyek

Analisa berorientasi obyek adalah suatu pendekatan yang digunakan, yang pertama adalah mempelajari obyek-obyek yang ada agar dapat mengetahui apakah obyek tersebut dapat digunakan berkali-kali atau dapat disesuaikan untuk keperluan yang baru. Dan kegunaan kedua untuk menggambarkan obyek yang baru atau

memodifikasi obyek, yang akan dikombinasikan dengan obyek-obyek yang sudah ada ke dalam sebuah aplikasi bisnis komputer yang dapat bermanfaat.

Pendekatan dalam analisa berorientasi obyek dilengkapi dengan alat–alat dan tehnik yang dibutuhkan dalam pengembangan sistem, sehingga hasil akhir dari sistem yang dikembangkan akan didapatkan sistem yang dapat terdefinisi dengan baik dan jelas. Maka analisa berorientasi obyek akan dilengkapi dengan alat dan tehnik di dalam mengembangkan system.

Alat bantu yang digunakan dalam analisa berorientasi obyek antara lain :

1) *Activity Diagram*

Activity diagram menggambarkan berbagai alur kerja dari satu aktifitas ke aktifitas lainnya dalam sistem yang sedang dirancang, bagaimana masing-masing alur aktifitas berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. 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 aktifitas dari level atas secara umum. Activity diagram dibuat berdasarkan sebuah atau beberapa use case pada use case diagram, atau bahkan tanpa menggunakan use case.

Di dalam diagram aktifitas terdapat :

a) *Start point*, menggambarkan awal dari aktifitas
b) *End point*, menggambarkan akhir dari aktifitas
c) *Activities*, menggambarkan proses bisnis

Jenis—jenis *activities* :

(1) *Black Hole Activities*, ada masukan dan tidak ada keluaran

(2) *Miracle Activities*, tidak ada masukan dan ada keluaran dan dipakai pada waktu *start point*.

(3) *ion points*, tidak ada keterangan (pertanyaan) pada tengah belah ketupat seperti pada flowchart dan harus mempunyai *guards* (kunci).

(4) *Guards* (kunci), sebuah kondisi benar sewaktu melewati sebuah transisi, harus konsisten dan tidak *overlap*.
Contoh : x < 0, x = > 0 konsisten
x < = 0 dan x > = 0 tidak konsisten

(5) *Swimlane*, sebuah cara untuk mengelompokkan *activity*.

2) Analisa Keluaran

Analisa keluaran merupakan analisa mengenai keluaran-keluaran yang dihasilkan melalui proses-proses yang ada dalam sistem berjalan.

3) Analisa Masukan

Analisa masukan adalah untuk mengetahui dokumen-dokumen atau formulir apa saja yang digunakan sebagai masukan dalam pengolahan data pada sistem laporan penjualan yang sedang berjalan.

4) *Use Case Diagram*

Use Case Diagram menggambarkan sebuah fungsionalitas yang diharapkan dari sebuah sistem dan bagaimana sistem berinteraksi dengan dunia luar. Yang ditekankan dalam *Use Case Diagram* adalah “apa” yang diperbuat sistem, dan bukan “bagaimana” sistem itu melakukannya.

Sebuah *Use Case* merepresentasikan sebuah interaksi antara actor dengan sistem. *Use Case Diagram* juga menjelaskan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (*actor*). Use Casemerupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng- *create* sebuah daftar belanja, dan sebagainya.

Secara umum Use Case Diagram terdiri dari :

a) *Actor*

   Actor adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem. Untuk mengidentifikasikan actor harus ditentukan pembagian kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. *Actor* dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti staff penjualan, pelanggan, dll.

b) *Use Case*

   Use case menggambarkan perilaku, termasuk didalamnya interaksi antara *acto*r dengan sistem. *Use case* dibuat berdasarkan keperluan *actor*, merupakan “apa” yang dikerjakan sistem bukan “bagaimana” sistem mengerjakannya. Setiap use case harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Nama *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua use case yang memiliki nama yang sama.

c) *Relationship* (Relasi) / *Association* (Asosiasi)

Asosiasi menggambarkan aliran data / informasi. Asosiasi / relasi juga digunakan untuk menggambarkan bagaimana *actor* terlibat dalam *use case*. Relasi (*relationship*) digambarkan sebagai bentuk garis antara dua simbol dalam use case diagram.

Ada empat jenis relasi / asosiasi yang dapat timbul pada *use case* diagram, yaitu :

(1) Asosiasi antara *Actor* dan *Use Case*

Ujung panah pada association antara *actor* dan *use case* mengindikasikan siapa / apa yang meminta interaksi dan bukannya mengindikasikan aliran data. Sebaiknya gunakan garis tanpa panah untuk association antara *actor* dan *use case*. *Association* antar actor dan use case yang menggunakan panah terbuka untuk mengindikasikan bila actor berinteraksi secara pasif dengan sistem.

(2) Asosiasi antara *Use Case*

Relasi antara *use case* dengan *use case* :

(a) *Include*, menggambarkan suatu use case termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan << *include*>>.

(b) *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak control form dan mendeklarasikan ekstension pada use case utama atau dengan kata lain adalah perluasan dari use case lain jika syarat atau

kondisi terpenuhi. Digambarkan dengan garis lurus berpanah dengan tulisan <<*extend*>>.

(c) *Generalization* / *Inheritance*, digambarkan dengan garis lurus berpanah tertutup dari base *use case* ke *parent use case.*

c. Perancangan Sistem Berorientasi Obyek

Perancangan sistem adalah sebuah spesifikasi atau konstruksi yang bersifat teknis, pengidentifikasi kebutuhan pemecah masalah bisnis yang berbasis komputer dalam suatu analisa sistem.

Tujuan dari merancang sistem secara umum adalah untuk memberikan gambaran secara umum kepada user tentang sistem yang baru. Merancang sistem secara umum merupakan persiapa dari oerancangan terinci. Komponen-komponen sistem informasi dirancang dengan tujuan untuk dikomunikasikan kepada user bukan untuk pemrogram. Komponen sistem informasi yang dirancang adalah model, output, input, database, teknologi dan control.

Perancangan dengan menggunakan metode berorintasi obyek yaitu :

1) *Class Diagram*

Diagram kelas memperlihatkan aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Diagram ini berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur yang dibuat. Diagram ini merupakan fondasi untuk component diagram dan deployment diagram. Dalam notasi UML *class* digambarkan dengan kotak. Nama *class* menggunakan huruf besar diawal kalimatnya dan diletakkan diatas kotak.

a) *Association*

Association/asosiasi adalah class-class yang terhubungkan satu sama lain secara konseptual. Setiap association mempunyai dua association end. Sebuah *association end* juga memiliki " *multiplicity* ". *Multiplicity* menunjukkan beberapa banyak obyek yang berpartisipasi dalam suatu relasi. Secara umum, multiplicity menunjukkan batasan terendah dan tertinggi untuk obyek-obyek yang berpartisipasi. *Multiplicity* yang paling umum digunakan adalah 1, *, dan 0..1. Langkah-langkah transformasi dari conceptual data model ke tabel relasi adalah sebagai berikut :

(1) Jika hubungan yang terjadi antar class adalah 1 to 1 (*one to one*) maka atribut dari *relationship set* diambil dan dimasukkan ke *entitas* yang lebih membutuhkan.

(2) Jika hubungan yang terjadi antar *class* adalah 1 to 0..1 (*one to zero one*) maka atribut dari relationship set digabung ke entitas yang memiliki *multiplicity* 0..1.

(3) Jika hubungan yang terjadi antar *class* adalah 1 to * (*one to many*) maka atribut dari relationship set digabung dengan set entitas yang memiliki multiplicity banyak (*many*).

b) Atribut (*Attribute*)

Attribute adalah properti dari sebuah class. *Attribute* ini melukiskan batas nilai yang mungkin ada pada obyek dari *class.*

c) Operasi

Operasi adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang ada (atau *class* lain) dapat dilakukan untuk sebuah *class*.

2) LRS

Logical Record Structure (LRS) adalah suatu terstruktur yang terdiri dari sejumlah record type, dimana setiap record type dinyatakan dalam bentuk kotak persegi panjang dan memiliki sebuah nama yang unik ditulis diluar kotak dan nama field yang ditulis didalam kotak yang berisi link diantara *record type*, dimana setiap link diberi label dengan field yang muncul pada kedua buah record yang dihubungkan oleh link tersebut.

3) Tabel

Tabel adalah koleksi objek yang terdiri dari sekumpulan elemen yang diorganisasi secara kontigu, artinya memori yang dialokasi antara satu elmen dengan elmen yang lainnya mempunyai adress yang berurutan.

Pada tabel, pengertian perlu dipahami adalah :

a) Keseluruhan tabel (sebagai koleksi) adalah kontainer yang menampung seluruh elmen
b) Index tabel, yang menujukan adress dari sebuah elemen
c) Elmen tabel, yang dapat dipacu melalui indeknya, bertipe tertentu yang sudah terdefinisi
d) Seluruh elemen tabel bertipe ”sama”. Dengan catatan: beberapa bahasa pemograman memungkinkan pendefinisian tabel dengan elmen generik, tapi pada saat diinstansiasi, harus diinstanisasi dengan tipe sama.

4) Spesifikasi Basis Data

Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lain dan tersimpan diluar komputer serta digunakan perangkat lunak (*software*) tertentu untuk memanipulasinya.

Sedangkan sistem berbasis data adalah suatu sistem penyusunan dan pengelolaan record-record dengan menggunakan komputer dengan tujuan untuk menyimpan atau merekam serta melihat data operasional lengkap pada sebuah organisasi, sehingga mampu menyediakan informasi yang diperlukan untuk kepentingan proses pengambilan keputusan.

5) Rancangan Dokumen Keluaran

Rancangan dokumen keluaran merupakan informasi yang akan dihasilkan dari keluaran sistem yang akan dirancang. Data yang telah diolah menjadi informaasi penjualan tunai ini memiliki berbagai keluaran sesuai dengan penggunaan sistem.

6) Rancangan Dokumen Masukan

Rancangan dokumen masukan merupakan data yang dibutuhkan untuk menjadi masukkan sistem yang akan dirancang.

7) Rancangan Layar Program

Rancangan layar program merupakan bentuk tampilan sistem dilayar komputer sebagai antar muka dengan pemakai yang akan dihasilkan dari sistem yang dirancang.

8) Sequence Diagram

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah sekenario. Sequence diagram menunjukkan objek dan message (pesan) yang diletakkan diantara obyek-obyek ini di dalam use case. Komponen utama sequence diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. Message diwakili oleh garis dengan tanda panah dan aktu yang ditunjukkan dengan progress vertical (Munawar,2005:88).

Sequence diagram adalah adalah visual coding (perancangan form/ layar). Biasanya digunakan untuk tujuan analisa dan desain dan memfokuskan identifikasi metoda di dalam sebuah sistem. Diagram ini secara khusus berasosiasi dengan use case diagram. Memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu di dalam use case.
Sequence diagram biasanya dipakai untuk memodelkan :

a) Deskripsi tentang sistem yang ada pada sebuah atau beberapa *use case* pada *use case diagram*, yang menggambarkan hubungan antar actor dan *use case*.
b) Logika dari method (*procedure*).
c) Logika services (*high level method operation, function*).

Sequence diagram mempunyai :

(1) *Entity*, entitas yang mempunyai atribut yang memiliki data yang bisa direkam.
(2) *Boundary*, menghubungkan user dengan sistem.
(3) *Contro*l, untuk mengontrol aktifitas-aktifitas yang dilakukan oleh sebuah kegiatan.
(4) *Message*, menggambarkan komunikasi yang terjadi antar obyek.
(5) *Return Values*, ditampilkan dengan garis berpanah terputus, yang menggambarkan hasil dari pengiriman *message*.

(6) *Object lifeline*, Garis terputus yang tergantung dari *boxes*, menggambarkan life span (rentang hidup) obyek.
(7) *Actor Object*, Menggambarkan pihak yang melakukan interaksi atau yang memicu sistem untuk berfungsi.
(8) *Lifeline*, Menggambarkan eksekusi obyek selama *sequence*.

3. Konsep Dasar Sistem Penjualan

a. Pengertian Penjualan

Kegiatan penjualan merupakan suatu usaha yang dilakukan untuk mendistribusikan barang kebutuhan yang telah dihasilkan oleh produsen kepada konsumen yang memerlukan dengan memperoleh jasa berupa uang menurut harga.

b. Sistem Penjualan Tunai

Penjualan tunai dilaksanakan dengan mewajibkan pembeli membayar sejumlah harga beli barang lalu barang tersebut diserahkan kepada pembeli. Setelah itu, dilakukan pencatatan transaksi.

Penjualan dapat dilakukan dengan cara antara lain :

1) Penjualan Langsung

Penjualan langsung adalah cara penjualan dimana penjual langsung berhubungan atau berhadapan dengan pembeli. Pembeli dapat langsung mengemukakan keinginannya bahkan sering terjadi tawar-menawar harga untuk mencapai kesesuaian.

2) Penjualan Tidak Langsung

Penjualan tidak langsung dapat terjadi jika terdapat masalah antara lain : jarak antara lokasi penjual dan pembeli cukup jauh, respon masyarakat terhadap sebuah iklan atau katalog, terbatasnya waktu antara penjual dan pembeli. Contoh : Melalui telepon, Email, dan Fax.