BAB II

LANDASAN TEORI

2.1 Konsep Dasar Sistem

Pengertian Sistem terbagi dua yaitu dilihat dari pendekatan yang menekankan pada prosedur dan pendekatan yang menekankan pada elemen atau komponennya.

Menurut [Jog05] pendekatan sistem yang menekankan pada prosedur mendefinisikan sistem sebagai : ”*jaringan kerja dan prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan sasaran tertentu*”.

Adapun pendekatan sistem yang menekankan pada elemen atau komponennya mendefinisikan sistem sebagai : ”*kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu*”.

Dari kedua pengertian di atas dapat disimpulkan bahwa pengertian sistem adalah kumpulan elemen-elemen atau jaringan kerja dan prosedur-prosedur yang saling berinteraksi satu sama lain untuk mencapai tujuan atau sasaran tertentu.

2.1.1. Karakteristik Sistem

Menurut [Jog05] suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, yaitu :

a. Komponen-komponen (*Components*)

   Suatu sistem terdiri dari kumpulan komponen atau subsistem yang saling berinteraksi atau bekerja sama satu sama lain untuk membentuk satu

kesatuan. Setiap komponen atau subsistem tersebut memiliki sifat-sifat dan fungsi tertentu yang mempengaruhi proses sistem secara keseluruhan.

b. Batas Sistem (*Boundary*)

Batas sistem yaitu daerah yang membatasi antara suatu sistem dengan lingkungan luarnya atau dengan sistem yang lainnya. Batas suatu sistem merupakan ruang lingkup (*scope*) dari sistem tersebut. Dan batas sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan.

c. Lingkungan Luar Sistem (*environments*)

Environments dari suatu sistem adalah apapun yang berada diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem yang harus dijaga dan dipelihara. Sedangkan lingkungan luar sistem yang bersifat merugikan harus ditahan dan dikendalikan, kalau tidak maka akan mengganggu kelangsungan hidup dari sistem.

d. Penghubung Sistem (*Interface*)

Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lainnya. Keluaran (*output*) dari satu subsistem akan menjadi masukan (*input*) untuk subsistem yang lain dengan melalui penghubung. Dengan penghubung satu subsistem dapat berintegrasi dengan subsistem yang lainnya membentuk satu kesatuan.

e. Masukan (*Input*)

Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). *Maintenance input* adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. *Signal input* adalah energi yang diproses untuk didapatkan keluaran. Sebagai contoh di dalam istem komputer, program adalah *maintaenance input* yang digunakan untuk mengoperasikan komputer dan data adalah *signal input* untuk diolah menjadi informasi.

f. Keluaran (*Output*)

Keluaran merupakan hasil dari energi yang diolah. Keluaran ini dapat berupa keluaran yang berguna atau keluaran yang tidak berguna. Sebagai contoh panas yang dikeluarkan dari sistem komputer merupakan keluaran yang tidak berguna dan merupakan sisa pembuangan. sedangkan informasi adalah keluaran yang berguna dan dibutuhkan.

g. Pengolah (*process*)

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukan menjadi keluaran. Sistem akuntansi akan mengolah data-data transaksi menjadi laporan-laporan keuangan dan laporan-laporan yang dibutuhkan oleh manajemen.

h. Sasaran Sistem (*Objectives*)

Suatu sistem pasti mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang

dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya.

2.1.2. Klasifikasi Sistem

Menurut [Jog05] sistem dapat diklasifikasikan dari beberapa sudut pandang diantaranya adalah sebagai berikut:

a. Sistem abstrak (*abstrak system*) dan sistem fisik (*phisical system*)

   Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, misalnya sistem teologi. Adapun sistem fisik merupakan sistem yang ada secara fisik, misalnya sistem komputer.

b. Sistem alamiah (*natural* system) dan sistem buatan manusia (*Human Made System*).

   Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat manusia. Misalnya sistem perputaran bumi. Adapun sistem buatan manusia adalah sistem yang dirancang oleh manusia. Misalnya, sistem buatan manusia yang melibatkan interaksi antara manusia dengan mesin yang disebut *human machine system*.

c. Sistem tertentu (*Deterministik System*) dan sistem tak tertentu (*Probabilistik System*).

   Sistem tertentu beroperasi dengan tingkah laku yang sudah dapat diprediksi. Interaksi diantara bagian-bagiannya dapat dideteksi dengan pasti, sehingga keluaran dari sistem dapat diramalkan. Misalnya sistem komputer. Adapun sistem tak tentu adalah sistem yang kondisi masa depannya tidak dapat di prediksi karena mengandung unsur probabilitas.

d. Sistem tertutup (*Closed System*) dan sistem terbuka (*Open System*).

Sistem tertutup merupakan sistem yang tidak berhubungan dan tidak terpengaruh dengan lingkungan luarnya. Sistem ini bekerja secara otomatis tanpa adanya turut campur dari pihak luarnya. Secara teoritis sistem tertutup ini ada, tetapi kenyataannnya tidak ada sistem yang sifatnya tertutup, yang ada hanyalah *relatively closed system* (secara relatif tertutup, tidak benar-benar tertutup). Sistem terbuka adalah sistem yang berhubungan dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan keluaran untuk lingkungan luar atau subsistem yang lainnya. Karena sistemnya terbuka dan terpengaruh lingkungan luarnya, maka suatu sistem harus mempunyai suatu sistem pengendalian yang baik.

2.2. Konsep dasar Informasi

Menurut [Jog05] informasi adalah **”***data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya***”.**

Menurut [Kad03] informasi adalah ”*data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau saat mendatang*”.

Dari pengertian di atas disimpulkan bahwa informasi adalah hasil dari data yang telah diolah yang berarti bagi penerima atau pengguna dan berguna dalam mengambil keputusan saat ini atau keputusan masa yang akan datang.

Adapun informasi yang dihasilkan dari proses pengolahan data tersebut pada kenyataannya memiliki kualitas yang berbeda. Menurut [Jog05] kualitas

informasi tergantung dari tiga hal yaitu informasi harus akurat (*accurate*), tepat waktu (*timeliness*) dan relevan (*relevance*).

a. Akurat

Berarti informasi harus bebas dari kekurangan atau kesalahan-kesalahan. Akurat juga berarti informasi harus jelas mencerminkan keadaan yang sebenarnya.

b. Tepat waktu

Berarti informasi harus selalu ada pada saat dibutuhkan pengguna dan tidak datang terlambat. Karena informasi merupakan landasan di dalam pengambilan keputusan. Jadi bila terlambat, informasi tersebut tidak mempunyai nilai lagi dan dapat berakibat fatal bagi organisasi.

c. Relevan

Informasi untuk tiap orang satu dengan yang lainnya berbeda. Maka informasi yang relevan adalah informasi yang berguna dan sesuai dengan kebutuhan setiap pemakainya.

2.3. Konsep Dasar Sistem Informasi

Berdasarkan kesimpulan mengenai pengertian sistem dan informasi pada subbab sebelumnya, sistem informasi dapat didefinisikan sebagai kumpulan elemen-elemen yang saling berinteraksi satu sama lain untuk mencapai tujuan tertentu yaitu mengolah data menjadi bentuk yang lebih berarti dan berguna bagi penerimanya serta bermanfaat untuk pengambilan keputusan saat ini atau masa yang akan datang.

Menyangkut pemahaman tentang pengertian sistem informasi ini, dalam bukunya [Kad03] mengutip beberapa pendapat para ahli, diantaranya :

Menurut Hall sistem Informasi adalah ”*sebuah rangkaian prosedur formal di mana data dikelompokkan, diproses menjadi informasi dan didistribusikan kepada pemakai*”.

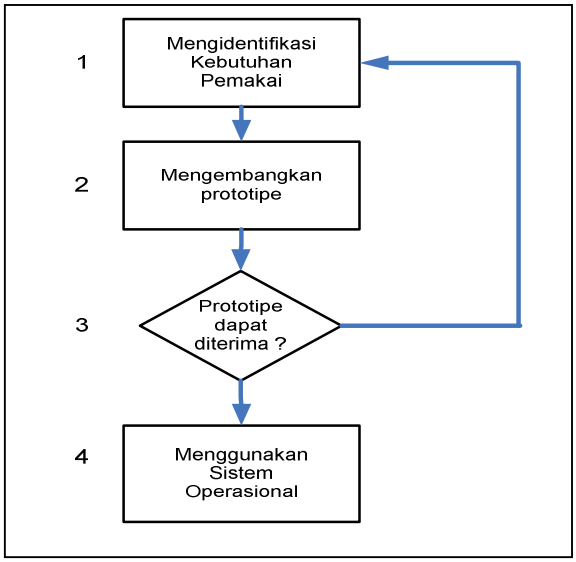
Menurut Bodnar dan Hopwood sistem Informasi adalah ***”****Kumpulan perangkat keras dan perangkat lunak yang dirancang untuk mentransformasikan data kedalam bentuk informasi yang berguna****”.***

Dari ketiga pengertian di atas dapat disimpulkan bahwa yang dimaksud sistem informasi sesungguhnya tidak harus melibatkan komputer ketika proses mengolah data menjadi sebuah informasi. Adapun Sistem informasi yang melibatkan komputer biasa disebut sistem informasi berbasis komputer (*Computer-Based Information Systems* atau CBIS). Akan tetapi, pada umumnya kata sistem informasi diartikan sebagai sistem informasi berbasis komputer, meskipun tidak ditambahkan kata berbasis komputer. Begitupun pada pembahasan skripsi ini, sistem informasi yang dimaksud adalah sistem informasi berbasis komputer.

2.4. Metode Pengembangan Sistem

Dewasa ini, metode pengembangan sistem untuk menyelesaikan sebuah proyek sistem informasi banyak sekali ragamnya, dan salah satunya adalah model prototipe. Metode ini bertujuan untuk mendapatkan kebutuhan yang jelas dan disetujui calon pemakai.

Adapun tahapan dari model prototipe dapat dilihat pada gambar di bawah ini :



Gambar 2.1 Model *Prototyping*

[Sumber : Abdul Kadir. “ Pengenalan Sistem Informasi”.(2003)]

Menurut Lucas [Kad03], sasaran prototipe secara garis besar adalah sebagai berikut:

1. Menyediakan umpan balik yang cepat dari pemakai (*user*) kepada pengembang (*sistem developer*).
2. Membantu menggambarkan kebutuhan pemakai (*user*) dengan kesalahan yang lebih sedikit.
3. Meningkatkan pemahaman pengembang (*sistem developer*) dan pemakai (*user*) terhadap sasaran yang seharusnya dicapai oleh sistem.
4. Menjadikan keterlibatan pemakai (*user*) sangat berarti dalam analisis dan desain sistem.

Seluruh metode pengembangan memiliki kelebihan dan kekurangan, berikut adalah kelebihan-kelebihan dan kekurangan-kekurangan dari model prototipe [Kad03].

Berikut ini kelebihan model prototipe yaitu :

1. Pendefinisian kebutuhan pemakai menjadi lebih baik karena keterlibatan pemakai yang lebih intensif.
2. Memperkecil kesalahan disebabkan pada setiap versi prototipe kesalahan segera terdeteksi oleh pemakai.
3. Pemakai mempunyai kesempatan dalam meminta perubahan-perubahan.
4. Mempersingkat waktu dalam mengembangkan sistem secara keseluruhan.
5. Menghemat biaya jika dibandingkan dengan menggunakan metode SDLC tradisional.

Adapun kekurangan metode prototipe yaitu :

1. Sistem akan baik jika pemakai sungguh-sungguh meluangkan waktunya untuk menggarap prototipe.
2. Dokumentasi sering terabaikan karena pengembang lebih berkonsentrasi pada tahap pengujian dan pembuatan prototipe.
3. Waktu yang singkat dapat menghasilkan sistem yang tidak lengkap dan kurang teruji.
4. Jika proses pengulangan terlalu sering dapat mengakibatkan pemakai jenuh dan memberikan respon yang negatif.

5. Apabila prototipe tak dikelola dengan baik dapat mengakibatkan prototipe tak pernah berakhir karena usulan perubahan terlalu sering dipenuhi.

2.5. Metode Pendekatan Sistem

Metode pendekatan sistem yang digunakan adalah *Object Oriented* (OO) yang menekankan pada solusi *logic* berbasis 5 objek. Teknologi *Object Oriented* merupakan paradigma baru dalam rekayasa Perangkat Lunak yang didasarkan pada objek dan kelas.

Object Oriented mencakup bidang aplikasi yang sangat luas. Pengguna sistem komputer dan sistem lain yang didasarkan atas teknologi komputer merasakan efek *Object Oriented* dalam bentuk meningkatnya aplikasi *software* yang mudah digunakan dan *service* yang lebih flexsibel, yang muncul dalam berbagai bidang industri, seperti dalam perbankan, telekomunikasi, dan sebagainya. Sedangkan bagi *software engineer*, *Object Oriented* berpengaruh dalam bahasa pemrograman, metodologi rekayasa, manajemen proyek, *hardware*, dan sebagainya. Sedangkan arti objek dalam *'software analysis & design'* adalah sesuatu berupa konsep (*concept*), benda (*thing*), dan sesuatu yang membedakannya dengan lingkungannya. Secara sederhana objek adalah mobil, manusia, *alarm* dan lainlainnya. Tapi objek dapat pula merupakan sesuatu yang abstrak yang hidup didalam sistem seperti tabel, *database, event, system messages*

2.5.1 OMT (Object Modeling Technique)

Metode pendekatan sistem yang digunakan pada penulisan ini yaitu metode OMT (*Object Modeling Technique*). OMT (*Object Modeling Technique*) sebuah metode *Object Oriented* untuk membangun software lewat koleksi objek-objek yang didalamnya mengandung data dan *behavior*. Karakteristik dari OMT (*Object Modeling Technique*) adalah menemukan hal mendasar dari *Object Oriented* berupa *Identity, Classification (class & Object), polymorphism, Inheritance.* Adapun esensi dari *Object Oriented Development* OMT (*Object Modeling Technique*) adalah mengidentifikasi dan mengorganisasikan *application-domain concept* sampai pada *implementation-domain concept*. Tahapan yang digunakan Metode OMT (*Object Modeling Technique*) terdiri dari 3 tahapan yaitu Analisis, Design dan Implementasi. OMT (*Object Modeling Technique*) membagi analisis dan design menjadi 3 bagian yaitu :

1. *Analysis* : membuat model dari dunia nyata berdasarkan problem domain
2. *System Design* : membangun arsitektur sistem secara keseluruhan
3. *Object Design* : menemukan struktur dari Objek sehingga memudahkan dalam proses implementasi.

2.6. Alat Pengembangan System

Dalam mengerjakan tahapan-tahapan yang berada dalam metode pengembangan sistem, maka dibutuhkan alat agar lebih mudah dalam melaksanakannya. Adapun alat yang digunakan dalam pengembangan sistem adalah sebagai berikut :

2.6.1. UML (*Unified Modeling Language*)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg 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 bahasabahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

2.6.2. Konsep Dasar UML

Abstraksi konsep dasar UML yang terdiri dari *structural classification*, *dynamic behavior*, dan *model management*, bisa kita pahami dengan mudah

apabila kita melihat gambar diatas dari *Diagrams. Main concepts* bisa kita pandang sebagai term yang akan muncul pada saat kita membuat diagram. Dan view adalah kategori dari diagaram tersebut. Lalu darimana kita mulai ? Untuk menguasai UML, sebenarnya cukup dua hal yang harus kita perhatikan:

1. Menguasai pembuatan diagram UML
2. Menguasai langkah-langkah dalam analisa dan pengembangan dengan UML

1. Bagian-Bagian UML

Bagian- bagain utama dari UML adalah *view, diagram, model element,* dan *general mechanism.*

1. View

View digunakan untuk melihat sistem yang dimodelkan dari beberapa yang berbeda. View bukan mlihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis view dalam UML antara lain : *Use Case View, Logical View, Component View, Concurrency View dan Deployment View.*

a. Use Case View

Use Case View mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actor. Actor* yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya.

b. Logical View

Logical View mendeskripsikan bagaimana fungsionalitas sistem, struktur statis (*class,object* dan *relationship*) dan kolaborasi dinamis yang terjadi.

c. Component View

Component View mendeskripsikan implementasi dan ketergantungan modul komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi *administrative* lainnya.

d. Councurrency View

Councurrency View membagi sistem kedalam proses dan prosesor, *view* ini digambarkan dalam diagram dinamis dan diagram implementasi serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*) dan penguji (*tester*).

e. Deployment View

Deployment View mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungan dengan lainnya.

2. Diagram

Diagram merupakan bagian dari suatu *view* tertntu dan ketika digambarkan biasanya dialokasokan untuk *view* tertentu ada beberapa jenis diagram anatara lain :

a. Use Case Diagram

Use Case Diagram adalah deskripsi fungsi yang disediakan oleh sistem dalam bentuk teks sebagai dokumentasi dari *Use Case Symbol* namun dapat juga dilakukan dalam *Activity Diagram*.

b. Class Diagram

Class Diagram menggambarkan struktur statis *class* didalam sistem *class* merepresentasikan sesuatu yang ditangani oleh sistem c*lass* dapat

berhubungan dengan yang lain melalui berbagai cara *associated* (terhubung satu sama lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), atau *package* (grup bersama sebagai satu unit).

c. Statechart Diagram

Statechart Diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

d. Activity Diagram

Activity Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

e. Sequence Diagram

Sequence Diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

f. Collaboration Diagram

Collaboration Diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1.Messages dari level yang sama memiliki prefiks yang sama.

g. Component Diagram

Component Diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code,* baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time,* maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain

h. Deployment Diagram

Deployment Diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar *node* (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

3. Indikator Multiciplicity

Walaupun *multiciplicity* ditentukan untuk *class, multiciplicity* menentukan banyaknya objek yang terhubung sutu dengan yang lainnya. Indikator *multiciplicity* terdapat pada masing-masing akhir garis relasi, baik pada asosiasi dan agregasi, beberapa contoh *multiciplicity* adalah :

e. Tepat Satu

0..* Nol atau Lebih

1..* Satu atau Lebih

0..1 Nol atau Satu

5..8 Range 5 s.d 8

4..6,9 Range 4 s.d 6 dan 9

4. Businees Modeling

Dalam dunia bisnis dan industri, terdapat banyak sistem manual dan otomatis yang muncul secara regular setiap sistem tersebut meiliki satu atau banyak *workflow*. *Business Modeling* menggambarkan semua *workflow* yang terjadi dalam suatu organisasi. Secara formal, kita definisikan *Business Modeling* sebagai segala teknik pemodelan yang digunakan untuk menggambarkan model sebuah bisnis. *Business Modeling* dapat digubakan untuk meninjau, meningkatkan, dan membuat sebuah bisnis. Adapun tujuan *Business Modeling* adalah :

1. Memahami struktur dan dinamika organisasi
2. Memahami masalah-masalah dalam mencapai target organisasi dan menemukan potensi untuk kemajuan organisasi
3. Yakin bahwa para *costumer, end user* dan *developer* mempunyai sebuah pemahaman yang benar mengenai organisasi
4. Dapat menurunkan/mendapatkan *requirement software* aplikasi yang kita buat yang diperlukan dalam mencapai tujuan suatu organisasi.

Untuk mencapai tujuan tersebut, harus membuat dua jenis model bisnis, yaitu model *use case* bisnis (*Business Use Case Model*) dan model objek bisnis (*Business Object Model*) yang berisi proses-proses, peranan-peranan, dan tanggung jawab organisasi untuk mencapai targetnya.

5. Proses Bisnis

Dalam suatu organisasi mungkun terdapat banyak proses bisnis. Proses Bisnis (*Business Process)* adalah sekumpulan aktivitas yang dirancang untuk menghasilkan keluaran (*output*) tertentu bagi *costumer.* Sebuah proses bisnis menekankan pada bagaimana sebuah pekerjaan dikerjakan dalam sebuah organisasi, dengan berfokus pada produk yang dihasilkan proses tersebut. Proses disini adalah urutan aktivitas tertentu terhadap waktu dan tempat, dengan sebuah titik awal, sebuah titik akhir dan sebagai input dan output yang didefinisikan dengan baik.

6. Stereotype Dalam Model Bisnis

Rational Rose memiliki beberapa jenis elemen yang digunakan *Business Modeling*, yaitu :

1. Business Actor

Business Actor (Aktor Bisnis) menggambarkan peran yang dimainkan oleh seseorang atau sesuatu yang dengannya bisnis berinteraksi sebuah *Business Actor* menggambarkan seseorang seorang *costumer* atau *partner* bisnis, tetapi sebuah sistem informasi yang berhubungan dengan bisnis kita dapat berperan sebagai sebuah *bbsuness actor* juga.

2. Business Use Case

Business Use Case merupakan urutan tindakan yang dimainkan suatu bisnis yang menghasilkan sebuah mengasilkan sebuah nilai yang dapat dilihat dan dtujukan untuk suatu *business actor* tertentu. Setiap *business use case* mewakili suatu proses bisnis.

3. Business Use Case Realization

Business Use Case Realization pandang dari dalam (*internal view*) terhadap *business use case*. Yakni menentukan bagaimana pekerjaan (dalam *business us case*) tersebut diatur dan dimainkan untuk mencapai hasil yang diinginkan sebuah *business use case realization* menentukan *business worker* dan *business entity* yang terlihat ketika menjalankan sebuah *business use case*.

4. Business Worker

Business Worker sebuah kelas yang mewakili, seorang manusia yang bertindak dalam bisnis. Sebuah *business worker* berinteraksi dengan para *worker* lainnya dan memainkan entitas bisnis saat berpatisipasi dalam realisasi suatu *business use case*.

5. Business Entity (Entitas Bisnis)

Business Entity (Entitas Bisnis) menggambarkan “benda” yang ditangani atau digunakan oleh *business worker* selama melaksanakan suatu *business use case*.

7. Requirement

Requirement adalah kondisi atau kemampuan yang harus dipenuhi oleh *software* aplikasi yang akan dibuat *requirement* untuk *software* aplikasi dibagi kedalam dua kelompok, yaitu *function requirement* dan *non function requirement.* *Function requirement* menentukan tindakan yang harus dapat dimainkan sebuah software aplikasi *requirement* ini sering dimodelkan dalam bentuk *use-case-use-case* dalam *use-case* model *function requirement* juga menentukan masukan dan keluaran dari sebuah software aplikasi. *Function requirement* mencakup fitur, kemampuan (*capabilities*) dan faktor *securirty* software aplikasi. *Non-Function Requirement* menggambarkan atribut dari software aplikasi dan lingkungannya.

Requirement ini biasanya masuk di dalam *use case* dan tercakup dalam sifat (*property*) *use case* tersebut.

2.7. Arsitektur Jaringan

2.7.1. Pengertian Jaringan Komputer

Menurut [Kad03] jaringan komputer adalah hubungan dua buah simpul (umumnya berupa komputer) atau lebih yang tujuan utamanya adalah untuk melakukan pertukaran data.

Dalam prakteknya, jaringan komputer memungkinkan untuk melakukan berbagi (*sharing*) perangkat lunak, perangkat keras, dan bahkan berbagi (*sharing*) kekuatan pemrosesan.

2.7.2. Jenis-jenis Jaringan Komputer

Menurut [Bud05] ada empat kategori utama jaringan komputer, yaitu :

1. LAN (*Local Area Network*)

LAN digunakan untuk menghubungkan komputer yang berada dalam suatu area yang kecil, jarak antara komputer yang dihubungkannya hanya bisa mencapai 5 sampai 10 km. Suatu LAN biasanya bekerja pada kecepatan 10 sampai 100 Mbps. LAN menjadi populer karena memungkinkan banyak pengguna untuk memakai sumber daya secara bersama-sama.

2. MAN (*Metropolitan Area Network*)

MAN merupakan suatu jaringan yang cakupannya meliputi suatu kota. MAN menghubungkan LAN-LAN yang lokasinya berjauhan. Jangkauan MAN bisa mencapai 10 sampai beberapa ratus km. Suatu MAN biasanya bekerja pada kecepatan 1,5 sampai 150 Mbps.

3. WAN (*Wide Area Network*)

WAN dirancang untuk menghubungkan komputer-komputer yang terletak pada suatu cakupan geografis yang luas, seperti hubungan dari satu kota ke kota lain dalam suatu negara. Cakupan WAN dapat meliputi 100 sampai 1.000 km, dan kecepatan antar kota bisa bervariasi antara 1,5 Mbps sampai 2,4 Gbps.

4. GAN (*Global Area Network*)

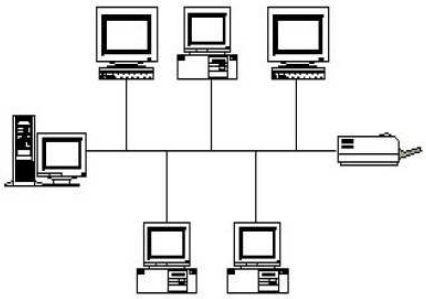
GAN merupakan jaringan yang menghubungkan negara-negara di seluruh dunia. Kecepatan GAN bervariasi mulai dari 1,5 Mbps sampai 100 Gbps dan cakupannya mencapai ribuan kilometer.

2.7.3. Topologi Jaringan Komputer

Menurut [Bud05] topologi fisik jaringan yang digunakan di dalam jaringan diantaranya:

1. *Linear Bus* (Garis Lurus)

Topologi *linear bus* terdiri dari satu jalur kabel utama dimana masing-masing ujungnya diberikan sebuah *terminator*. Semua *nodes* pada jaringan (*file server*, *workstation*, dan perangkat lainnya) terkoneksi sebuah kabel utama (*backbone*).

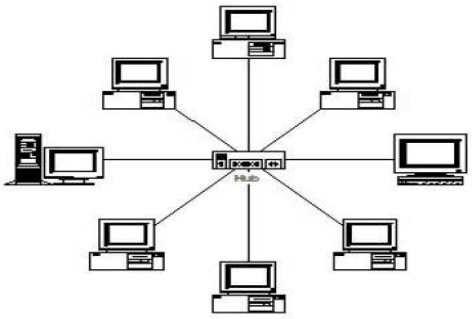


Gambar 2.2 Topologi *Linear Bus*

[Sumber : Dian Ardiyansah.”Teknologi Jaringan Komputer**”.(**2004)]

2. *Star* (Bintang)

Pada topologi *star*, setiap *nodes* (*file server*, *workstation*, dan perangkat lainnya) terkoneksi ke jaringan melalui sebuah *concentrator*.

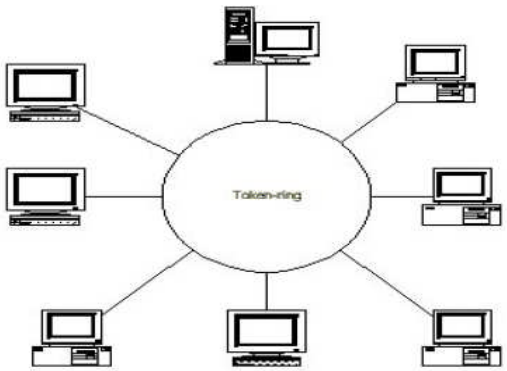


Gambar 2.3 Topologi *Star*

[Sumber : Dian Ardiyansah.”Teknologi Jaringan Komputer**”.(**2004)]

3. *Ring* (Cincin)

Topologi *ring* menggunakan teknik konfigurasi yang sama dengan topologi star tetapi pada topologi ini terlihat bahwa jalur media transmisi menyerupai suatu lingkaran tertutup.



Gambar 2.4 Topologi *Ring*

[Sumber : Dian Ardiyansah.”Teknologi Jaringan Komputer**”.(**2004)]

***4. Tree* (Pohon)**

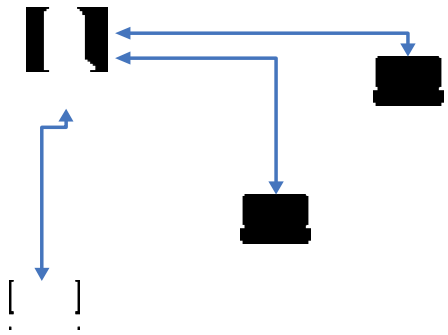
Topologi model ini merupakan perpaduan antara topologi *linear bus* dan *star*, yang terdiri dari kelompok-kelompok dari *workstation* dengan konfigurasi *star* yang terkoneksi ke kabel utama yang menggunakan topologi *linear bus*.

Topologi ini memungkinkan untuk perkembangan jaringan yang telah ada dan memungkinkan untuk mengkonfigurasi jaringan sesuai dengan kebutuhan.

2.8. Pengertian Client Server

Client-server merupakan model hubungan sebuah jaringan yang memungkinkan untuk mensentralisasi fungsi dari aplikasi kepada satu atau dua *dedicated file server*. Model hubungan ini menyediakan mekanisme untuk mengintegrasikan seluruh komponen yang ada di jaringan dan memungkinkan banyak pengguna secara bersamaan memakai sumber daya pada *file server.*

Sebuah *file server* menjadi jantung dari keseluruhan sistem, dan *Workstation* yang berdiri sendiri dapat mengambil sumber daya yang ada pada *file server*.



Gambar 2.5 Model hubungan *Client-Server*

[Sumber : Dian Ardiyansah.”Teknologi Jaringan Komputer**”.(**2004)]

2.9. Perangkat Lunak Pendukung

Sistem Informasi Penjualan Barang Di CV DEROSE dikembangkan dan diimplementasikan dengan menggunakan program *Borland Delphi 7.0* dengan

Sistem Operasi *Microsoft Windows Xp*, Dengan Menggunakan Database *Microsoft SQL Server 2000*.

2.9.1. Sekilas Tentang Borland Delphi 7.0

Menurut Teddy Zakaria Marcus dalam bukunya yang berjudul "*Pemograman Delphi Untuk Pemula*" mengatakan, Delphi adalah

"*Suatu program berbasis bahasa Pascal yang berjalan dalam lingkungan Windows"*.

Delphi telah memanfaatkan suatu teknik pemrograman yang disebut RAD yang telah membuat pemrograman menjadi lebih mudah. *Delphi* adalah suatu bahasa pemrograman yang telah memanfaatkan metode pemrograman *Object Oriented Programming* (OOP). *Delphi* juga merupakan sebuah adalah kompiler atau penterjemah bahasa *Delphi* (awalnya dari *Pascal*) yang merupakan bahasa tingkat tinggi sekelas dengan *Basic*, C.

Bahasa Pemrograman di *Delphi* disebut bahasa *prosedural* artinya bahasa atau sintaknya mengikuti urutan tertentu atau *prosedur.* Ada jenis pemrograman non-prosedural seperti pemrograman untuk kecerdasan buatan seperti bahasa Prolog. *Delphi* termasuk keluarga *Visual* sekelas *Visual Basic*, *Visual* C, artinya perintah-perintah untuk membuat objek dapat dilakukan secara visual. Pemrogram tinggal memilih objek apa yang ingin dimasukkan kedalam *Form* atau *Window*, lalu tingkah laku objek tersebut saat menerima *event* atau aksi tinggal dibuat programnya.

Delphi merupakan bahasa berorentasi objek, artinya nama objek, properti dan *methode*/ *procedure* dikemas menjadi satu kemasan (*encapsulate*). Semenjak Delphi 6, Delphi telah di lengkapi dengan sejumlah komponen yang tergolong sebagai *dbExpress*, yang memungkinkan koneksi ke *Microsoft* SQL Server 2000 ataupun *Oracle* dilakukan dengan mudah, sehingga *Delphi* dapat digunakan sebagai aplikasi *front-end* yang berhubungan dengan *database server*. Pada versi 7.0, komponen yang tergolong *dbExpress* sedikit berubah. Ada beberapa hal yang menjadi daya tarik dari *Delphi* sendiri, *Delphi* langsung membuat *Application* sendiri (.EXE) dan *listing* bisa dibuka dalam *notepad* (.*pas).

2.9.2. Sekilas Tentang Microsoft SQL Server 2000

Microsoft SQL Server diperkenalkan pada tahun 1990 untuk platform Microsoft OS/2 dalam kerjasamanya dengan sistem basis data. SQL Server 2000 dikomersilkan pada tahun 2000 dan mempunyai desain modern. SQL Server 2000 adalah mesin *database client/server* yang berbeda dengan *database* komputer tunggal tradisional yang memakai sistem pemakaian file secara bersama-sama (misalnya dBase, Microsoft Jet, Microsoft Visual Foxpro). *Database* sistem memakai file secara bersama-sama bergantung pada sebuah proses *client* tunggal per *user* untuk memanipulasi data pada file yang dipakai bersama pada *server* jaringan.

SQL Server 2000 memberikan bahasa dan antarmuka yang baik untuk pemrograman dan komunikasi pada *server*. Transact SQL adalah bahasa pemrograman *server* yang merupakan superset dari ANSI-SQL. ANSI-SQL mendifinisikan empat perintah dasar untuk memanipulasi data yaitu SELECT,

INSERT, UPDATE, DELETE, dan sejumlah perintah untuk mendefisikan struktur *database.*

SQL Server 2000 memakai konsep transaksi dan *logging* transaksi untuk menjamin konsistensi dan kemampuan *recover* informasi yang dismpan didalam *database* SQL Server mendukung *backup* secara online. File-file data yang dipakai pada aplikasi *database* dibuka secara ekslusif oleh proses *server.* Oleh sebab itu melakukan *backup* pada file secara individu tidak memungkinkan, kecuali *server* diberhentikan terlebih dahulu. *Backup database* dapat dilakukan meskipun ada transaksi yang aktif.

2.10. Pengertian penjualan

Penjualan adalah merupakan salah satu ujung tombak dari perusahaan, sebab tujuan utama bagi perusahaan setelah perancangan menejemen dibuat secara baik sesui dengan system-sistem yang akurat penjulanlah yang menjadi utama untuk memperoleh laba dari hasil penjualan sebagai kelangsungan hidup perusahaan itu dapat dipertahankan dan direalisasikan.

Pengertian penjualan menurut SWA [5] adalah :

“ Interaksi antar individu, saling bertatap muka yang ditujukan untuk menciptakan, memperbaiki, menguasai, atau mempertahankan hubungan pertukaran yang saling menguntungkan dengan pihak lain”

Kegiatan menjual terbagi menjadi dua cara yaitu :

1. Penjualan kredit yaitu penjualan barang yang dilakukan dengan cara mengirim barang sesuai dengan pesanan dari pembeli dan tidak ada

pembayaran langsung yang terjadi dalam jangka waktu tertentu dan perusahaan mempunyai tagihan pada pembeli barang

2. Penjualan tunai adalah apabila perusahaan tersebut menjadi produknya secara langsung pada pembeli dan dibayar pada saat itu juga oleh pembeli.

2.11. Usability

Menurut Sweeney et al., 1993 usability adalah " *emergent quality of an optimum design which is reacted it the efective and satisfying use of the IT* ". [http://www.aptik.or.id/buletin/buletin-I-3-okt-04.pdf].

Menurut [Lun01] '' *Usability appears to consist of usefulness and ease of use, and usefulness and ease of use are correlated. Each factor in turn drives user satisfaction and frequency of use* ".

Dalam artikelnya [Lun01] mengemukakan bahwa mereka telah mengembangkan alat untuk mengukur usability software, yakni kuesioner USE. USE terdiri dari *Usefulness* (Kegunaan), *Satisfaction* (Kepuasan), *and Ease of use* (Kemudahan Penggunaan) dan indikator *Ease of Use* dikembangkan menjadi dua yaitu *Ease of Use* dan *Ease of Learning*