BAB II

LANDASAN TEORI

2.1 Konsep Dasar Sistem

2.1.1 Pengertian Sistem

Sistem adalah Kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Elemen-elemen sistem dapat berupa subsistem yang saling berinteraksi dan saling berhubungan membentuk satu kesatuan, sehingga tujuan atau sasaran sistem tersebut dapat tercapai.

2.1.2 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat yang tertentu, yaitu mempunyai komponen-komponen *(componennts)*, batas sistem *(boundary)*, lingkungan luar sistem *(evironments)*, antarmuka *(interface)*, masukan *(input)*, keluaran *(output)*, pengolahan *(process)* dan sasaran *(objectives)* atau tujuan *(goal). Jogiyanto (2001;3-6)*

2.1.3 Pengertian Sistem Informasi

Pengertian Sistem Infomasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategis dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang dibutuhkan. *Jogiyanto (2001;11)*

Adanya kebutuhan informasi yang makin lama makin memperlihatkan kenaikan dan menjadi faktor yang penting, perlu mendapat perhatian dalam hal penanganannya.

Dengan demikian maka pembangunan suatu Sistem Informasi suatu organisasi unit usaha sangat diperlukan agar penanganan data dan informasi dapat dilaksanakan sesuai dengan prinsip tepat waktu, tepat guna, tepat sasaran dan dapat dipercaya.

Sistem Informasi merupakan suatu sistem dalam suatu organisasi yang merupakan kombinasi dari orang-orang, fasilitas, teknologi, media, prosedur-prosedur dan pengadilan untuk mendapatkan jalur komunikasi penting, memproses tipe transaksi rutin tertentu, memberi sinyal kepada manajemen dan yang lainnya terhadap kejadian-kejadian internal dan eksternal yang penting dan menyediakan suatu dasar informasi untuk pengembalian keputusan.

Sistem Informasi berarti juga Sistem Terotomasi, terdiri dari beberapa komponen, antara lain :

a) *Hardware* : *CPU, Disk, Terminal, Printer*

b) *Software* : Sistem Operasi, Sistem database, program aplikasi.

c) *Personil* :Yang mengoperasikan sistem, menyediakan masukan, mengkonsumsi keluaran dan melakukan aktivitas manual yang mendukung system.

d) Data : Data yang tersimpan dalam jangka waktu tertentu

e) *Prosedur* : intruksi dan kebijakan untuk mengoperasikan sistem

2.2 Peralatan Pendukung (*Tools Sistem*)

2.2.1 UML

Menurut Munawar (2005, hal 17) *Unified Modelling language* (UML) adalah suatu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek, UML menyediakan pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang effektif untuk berbagi dan mengkomunikasikan rancangan mereka dengan yang lain.

Romi Satria Wahono dan Sri Dharyanti *Unified Modelling Language* merupakan sebuah bahasa yang telah menjadi bahan standar dalam dunia industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML, dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, dan dapat digunakan juga untuk mendefinisikan notasi dan *syntax*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram-diagram piranti lunak, setiap bentuk mempunyai makna tertentu dan UML s*yntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan.

Uml menyediakan beberapa notasi dan *artifact* standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. *Artifact* didalam UML didefinisikan sebagai informasi yang dihasilkan dalam proses pengembangan perangkat.

Ada tiga modeling yang terdapat dalam UML yaitu :

a. *Function Modelling* merupakan diagram-diagram yang yang menyatakan bagaimana suatu sistem itu bekerja dan yang termasuk dalam *function modeling* adalah activity diagram, usecase deskripsi dan use case diagram.

b. *Struktural Modelling* terdiri dari beberapa diagram yang memberikan *snapshot* dari elemen-elemen yang berperilaku. *Struktural modeling* mempunyai beberapa model diantaranya *Class Diagram* dan *Class Resposibility Colaboration* (CRC Card).

c. *Behavior Modelling* mendifinisikan bagaimana suatu objek bertindak dan bereaksi, dan berhubungan dengan fungsi diterapkan pada suatu atribut. Mempunyai beberapa model yaitu *sequence digram* dan *state machine diagram*.

Tabel 2.1 Konsep Dasar Unified Modelling Language.

Major Area	*View*	*Diagrams*	*Konsep Utama*
Sturktural	*Static view*	*Class diagram*	*class, association, generalization, depedency realization, interface*
	use case view	*use case diagram*	*usecase actor, association, extend, include, usecase generalization*
	implementation View	*component diagram*	*component, interface, dependency realization*
	deployment View	*deployment diagram*	*node, component, dependency, location*
Dynamic	*state machine View*	*statechart diagram*	*state, event, transition, action*
	activity view	*activity diagram*	*state, activity, complextion transition, fork, join*
	interaction View	*sequence diagram*	*interaction, object, message, activation*
		collaboration diagram	*collaboration, interaction collaboration role message*
model management	*model management View*	*class diagram*	*package, subsistem, model*
extesibility	*All*	*all*	*constraint, stereotype tagged values*

2.2.2 Function Modelling

2.2.2.1 Activity Diagram

Romi Satria Wahono mengatakan bahwa *Activity diagrams* 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.

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

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

2.2.2.2 Use Case Description

Use case Deskripsi merupakan penjabaran tekstual tentang kejadian dan bagaimana pengguna berinteraksi dengan sistem, *Use case* deskripsi juga berfungsi sebagai ilustrasi narasi dalam menjelaskan permasalahan.

2.2.2.3.Use Case Diagram

Munawar (2005, hal 109-100) menjelaskan *Use case diagram* merupakan deskripsi fungsi dari sebuah sistem perspektif pengguna. *Use case* bekerja dengan cara memdeskripsikan tipikal interaksi antar *user* (pegguna) sebuah sistem dengan sistem-nya sendiri bagaimana sistem tersebut digunakan atau "apa" yang diperbuat sistem dan bukan "bagaimana". *Use case* dapat dipresentasikan sebuah interaksi antara *actor* dengan sistem .

Use case Diagram dapat sangat membantu bila sedang menyusun *requiremet* (Jacobsoonet all, 1992). Diagram ini menunjukan tiga aspek dari sistem yaitu *actor*, *use case*, dan sistem atau subsistem.

Dilihat dari segi pandang Romi Satria Wahono dan Sri Dharyanti sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. 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. Sementara hubungan generalisasi antar *use case*

menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

2.2.3 Struktural Modelling

2.2.3.1 Class Diagram

Class dilihat dari segi pandang Romi Satria Wahono dan Sri Dharyanti merupakan sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan. satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- *Public*, dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian.
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

2.2.4 Behavior Modelling

2.2.4.1 Sequence Diagram

Sequence diagram dilihat dari kacamata Romi Satria Wahono dan Sri Dharyanti menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Munawar mangatakan *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa

yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity*.

2.2.4.2 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.

2.2.4.3 Deployment Diagram

Deployment/physical 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* dan *requirement* dapat juga didefinisikan dalam diagram ini.

2.3 Visual Basic

Visual Basic merupakan salah satu tool programming yang telah menerapkan konsep RAD (Rapid Aplication Development) atau pengembangan aplikasi secara cepat, berbasis visual dan familiar bagi pengguna.yang tinggi untuk dimanfaatkan dalam pengembangan berbagai jenis aplikasi, mulai dari aplikasi sederhana, multimedia, aplikasi database dan interenet. Sejak kemunculannya pada tahun 1960, Bahasa Basic (Beginner's All-purpose Symbol Intruction Code) telah mengalami perkembangan yang sangat pesat sekali.

Di tahun 1970 digunakan oleh Bill Gates dan Paul Allen untuk mengontrol mekrokomputer Alltair dengan menggunakan pita kaset. Kemudian Bahasa Basic diikuti oleh pengembangan-pengembangan software lain dengan nama yang berbeda, namaun aturan dan bahasa yang digunkan adalah sama. Munculnya GW-Basic, Qbasic, Quic Basic dan lain sebagainya semakin mempopulerkan Bahasa Basic ini untuk digunakan pada mikrokomputer sebagai bahasa pemrograman untuk membuat aplikasi.

Visual Basic unutk DOS dan untuk Windows diperkenalkan pada tahun 1991, versi 3.0 dari Visual Basic dikeluarkan pada tahun 1993 dan lebih mengalami kemajuan yang pesat dibandingkan dengan versi sebelumnya. Visual Basic versi 3.0 masih menggunakan kode-kode yang bekerja dalam 16 bit,

kemudian pada akhir tahun 1995 delepas versi baru dari visual basic ini yang mendukung proses 32 bit yang diberi label Visual Basic 4.0 pada akhir tahun1996 diluncurkan Visual Basic versi 5.0 dengan kelebihan yang dapat mendukung contrrol Activex dan mulai menghapus atau menghilangkan dukungan terhadap proses 16 bit. Saat ini Visual Basic saudah sampai pada versi .Net, yang sebelumnya telah tersedia Visual Basic Versi 6.0.

2.4 MySQL

Merupakan software sistem manajemen basis data SQL (bahasa Inggris: database management system) atau DBMS yang multithread dan multi-user. MySQL AB membuat MySQL tersedia sebagai software gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius. (MySql ilmukomputer.com 2003-2007)

2.4.1 Tipe Data di MySQL

SQL meyediakan beberapa jenis tipe data dasar: boolean, bilangan bulat, bilangan pecahan desimal, teks, data biner, tanggal/waktu. Sebenarnya standar SQL:1999 menyediakan beberapa tipe data yang lain, seperti tipe data komposit (array dan row) serta tipe data ref/pointer. Standar SQL:2003 pun menambahkan

tipe data seperti MULTISET (yang mirip array) dan XML. Namun banyak database, termasuk MySQL, yang tidak atau belum mendukungnya. Tipe data yang ada di Mysql sebagai berikut:

a. Boolean
b. Integer
c. Float
d. Character
e. Date
f. Blob

2.5 Crystal Report

Crystal Report adalah program khusus untuk membuat laporan yang terpisah dengan program Microsoft Visual Basic 6.0, tetapi keduanya dapat dihubungkan. Hasil cetakan Crystal Report lebih baik dan lebih mudah, karena pada Crystal Report banyak tersedia objek maupun komponen yang mudah digunakan. (Yuniar Supardi, 2006,Hal:9).

Beberapa kelebihan Crystal Report adalah:

1) Dari segi pembuatan laporan tidak terlalu rumit yang memungkinkan para programmer pemula sekalipun dapat membuat laporan yang sederhana tanpa melibatkan banyak kode pemograman.
2) Integrasi dengan bahasa-bahasa pemograman lain yang memungkinkan dapat digunakan oleh banyak programer dengan masing-masing kaehlian.
3) Fasilitas impor hasil laporan yang mendukung format-format populer seperti Microsoft Word, Excel, Acces, Adobe Acrobat Reader, HTML, Dan sebagainya.

2.6 Administrasi

Administrasi berasal dari bahasa Belanda, "*Administratie*" yang merupakan pengertian Administrasi dalam arti sempit, yaitu sebagai kegiatan tata usaha kantor (catat-mencatat, mengetik, menggandakan, dan sebagainya). Kegiatan ini dalam bahasa Inggris disebut : Clerical works. Administrasi dalam arti luas, berasal dari *Administration*" , yaitu proses kerjasama antara dua orang atau lebih berdasarkan rasionalitas tertentu untuk mencapai tujuan bersama yang telah ditentukan.