BAB II

LANDASAN TEORI

2.1. Pengertian Sistem Informasi

1. Sistem informasi adalah kombinasi antar prosedur kerja, informasi, orang, dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi. [Alter (1992)]
2. Sistem informasi adalah kumpulan perangkat keras dan perangkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna. [Bodnar dan Hopwood (1993)]
3. Sistem informasi adalah suatu sistem buatan manusia yang secara umum terdiri atas sekumpulan komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan, dan mengelola data serta menyediakan informasi keluaran kepada para pemakai. [Gelinas, Oram, dan Wiggins (1990)]
4. Sistem informasi adalah sebuah rangkaian prosedur formal dimana data dikelompokkan, diproses menjadi informasi, dan didistribusikan kepada pemakai. [Hall (2001)]
5. Sebuah sistem informasi mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan yang spesifik. [Turban, McLean, dan Wetherbe (1999)]

2.1.1 Pengertian Data

Data berasal dari bahasa latin yaitu ‘Datum’ yang berisi fakta. Dalam perkembangannya, data tidak selalu mempunyai arti fakta atau kejadiannya yang konkrit atau aktual, tetapi data juga mengasumsikan yang tidak pernah terjadi, misalnya ide. Dalam perancangan *database*, data diasumsikan sebagai penjelasan atau gambaran dari fenomena atau ide yang dipertimbangkan oleh manusia dalam rangka memformulasikan dan mencatatnya.

Berdasarkan beberapa pendapat yang dikemukakan oleh beberapa ahli, maka dapat ditarik kesimpulan sebagai berikut:

“*Data adalah fakta-fakta dan angka-angka yang pada dasarnya belum berarti apabila diolah dalam suatu struktur atau sistem tertentu dalam suatu media tertentu untuk membentuk seperangkat fakta tertentu sehingga dapat membentuk suatu teori, konsep-konsep dan selanjutnya dapat dimengerti* “.

2.1.2 Pengertian Informasi

Informasi merupakan hasil suatu proses. Proses itu sendiri terdiri dari kegiatan-kegiatan mulai dari pengumpulan data, menyusun serta menghubungkannya, meringkas, mengambil inti sarinya, dan menginterpretasikannya sesuai dengan persepsi sistem informasi penerima agar menjadi sebuah informasi yang berguna. Berdasarkan definisi dari beberapa alhi maka dapat disimpulkan bahwa informasi adalah :

1. Merupakan kumpulan data yang telah diproses
2. Diproses dalam suatu format tertentu

3. Dapat memberikan arti bagi penerimanya
4. Bersifat tidak statis dalam proses pengambilan keputusan.

2.1.3 Fungsi Informasi

Beberapa fungsi informasi yang perlu diperhatikan antara lain sebagai berikut:

1. Menambah pengetahuan atau mengurangi ketidakpastian kepada pemakai informasi.

   Informasi kepada pemakai mungkin merupakan hasil data yang dimasukkan ke dalam pengolahan suatu model keputusan, akan tetapi dalam kebanyakan pengambilan keputusan yang kompleks, informasi hanya dapat menambah kemungkinan ketidakpastian atau mengurangi bermacam-macam pilihan.
2. Mengurangi keanekaragaman

   Informasi yang disediakan bagi pengambil keputusan, memberikan suatu kemungkinan faktor resiko dan tingkat pendapatan yang berbeda.
3. Memberikan standar-standar, aturan-aturan ukuran, dan aturan-aturan keputusan untuk penentuan dan penyebaran tanda-tanda kesalahan dan umpan baik guna mencapai tujuan kontrol, misalnya untuk pengambilan keputusan penanam modal dalam suatu proyek, maka informasi yang diperlukan untuk membantu mengontrol pelaksanaan proyek.

2.1.4 Pengertian Sistem

Sistem dijelaskan secara sederhana sebagai perangkat elemen yang digabungkan satu dengan yang lainnya untuk tujuan bersama. Semua sistem adalah merupakan bagian dari sistem yang lebih besar, seperti subsistem divisi, departemen, bagian dan lain sebagainya yang merupakan bagian dari sistem organisasi Dari definisi-definisi beberapa ahli dapat ditarik kesimpulan bahwa *Sistem adalah suatu unit yang terdiri dari komponen-komponen yang bersifat sederhana hingga yang bersifat kompleks dan memiliki sifat saling ketergantungan antara satu denganyang lainnya sehingga komponen-komponen tersebut akan didefinisikan sebagai satu kesatuan, baik secara fungsional maupun secara logik*.

2.1.5 Karakteristik Sistem

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, antara lain sebagai berikut:

1. Komponen-komponen *(Components)*

   Suatu sistem yang terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerjasama membentuk satu-kesatuan. Komponen-komponen sistem dapat berupa sub sistem atau bagian dari sistem.

2. Batas *(Boundary)*

   Merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainya atau dengan lingkungan luarnya.

3. Lingkungan luar *(Environments)*

Segala sesuatu diluar batas dari sistem yang mempengaruhi operasi baik itu menguntungkan operasi sistem yang merupakan energi dari sistem yang harus di pelihara atau di tahan sehingga tidak mengganggu atau merusak sistem.

4. Penghubung *(Interface)*

   Merupakan media pengubung antara satu sub sistem dengan sub sistem yang lainnya.

5. Masukan *(Input)*

   Masukan adalah data yang di masukan ke dalam sistem yang diproses untuk mendapat satu-kesatuan atau infomasi.

6. Keluaran *(Output)*

   Output adalah data yang diolah serta di klasifikasikan menjadi keluaran yang berguna atau menjadi informasi yang di butuhkan.

7. Pengolah *(process)*

   Suatu sistem yang mempunyai suatu bagian pengolah yang akan merubah input menjadi output.

8. Sasaran *(Objective)* dan Tujuan *(Goal)*

   Suatu sistem mempunyai maksud tertentu yaitu tujuan atau sasaran, sasaran dari sistem sangat menentukan sekali input yang di butuhkan sistem dan output yang akan di hasilkan sistem.

2.2 Basis Data

2.2.1 Definisi basis data

Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan lainnya, tersimpan dalam perangkat keras komputer dan diperlukan suatu perangkat lunak untuk dapat memanipulasi basis data tersebut.

Basis data didefinisikan sebagai :

"Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah yang disimpan dalam media penyimpanan elektronis" .

2.2.2 Organisasi File Basis Data

Penekanan file pada basis data adalah kemampuan untuk mengakses data dengan cepat dan efisien dalam menggunakan media simpanan luarnya. Faktor yang mempengaruhi hal ini adalah organisasi dari file basis data. Organisasi file basis data ini mencoba meningkatkan struktur dari data agar integrasi data antara satu file dengan file yang lainnya, dengan menunjukkan hubungan antar data yang ada :

a. Struktur data berjenjang (*hierarchial data structure*) atau disebut juga dengan nama data pohon (*tree data structure*) menunjukkan hubungan antara data membentuk suatu jenjang seperti pohon
b. Struktur data jaringan (*network data structure*) disebut juga dengan plex data struktur. Pada struktur data pohon tiap-tiap node tidak dapat mempunyai lebih

dari satu orang tua, maka struktur data jaringan ini, tiap-tiap node punya banyak orang tua

c. Struktur data hubungan adalah meletakkan semua hubungan dalam bentuk tabel dua dimensi. Data dalam model ini dapat diidentifikasi dalam hubungan nyata terhadap data item yang ada.

2.2.3 Operasi Dasar Basis Data

Operasi–operasi dasar yang dapat kita lakukan berkenaan dengan basis data dapat meliputi :

1. Pembuatan basis data baru (*create database*)
2. Penghapusan basis data (*drop database*)
3. Pembuatan file atau table baru ke suatu basis data (*create table*)
4. Penghapusan file atau table dari suatu basis data (*drop table*)
5. Penambahan atau pengisian data baru ke sebuah file atau table di sebuah basis data (*insert*)
6. Pengambilan data dari sebuah file atau table (*retrieve atau search*)
7. Pengubahan data dari sebuah file atau table (*update*)
8. Penghapusan data dari sebuah file atau table (*delete)*

2.2.4 Arsitektur sistem basis data

Terdapat dua bentuk arsitektur sistem basis data, yaitu sistem terpusat dan sistem Client-Server. Sistem basis data terpusat adalah sistem basis data yang

dijalankan pada sistem komputer tunggal dan tidak berinteraksi dengan sistem pada komputer lain. Pengguna terkoneksi kekomputer pusat melalui terminal.

Sistem basis data *Client-Server* adalah sitem basis data yang memisahkan program pengguna dengan program basis data di sistem yang berbeda. Pengguna terkoneksi kepusat data yang disebut *server* sistem melalui suatu program pengguna *(user interface)* yang terdapat pada personal computer. Sistem tempat program pengguna berada disebut client sistem.

2.2.5 File

Sebuah *file* diorganisasikan secara logikal sebagai kumpulan *record. Record* terdiri atas sekumpulan *field*. *Record* adalah informasi satu unit atau individual dalam basis data. *Field* adalah sebuah item data atau elemen data. Satu *field* merepresentasikan sebuah kategori informasi, sedangkan satu *record* merepresentasikan gabungan dari item data, elemen data atau *field.*

2.3 Pengertian Program Aplikasi

Aplikasi merupakan penerapan, penyimpan sesuatu hal, data, permasalahan pekerjaan ke dalam suatu sarana atau media yang di gunakan untuk menerapkan atau mengimplementasikan permasalahan tersebut sehingga berubah menjadi suatu bentuk yang baru tanpa menghilangkan nilai-nilai dasar, data, permasalahan atau pekerjaan. Jadi dalam hal ini hanya bentuk dan tampilan data yang merubah, sedangkan isi yang termuat dalam data tersebut tidak mengalami

perubahan. Jadi Program Aplikasi adalah suatu perangkat lunak yang di bangun untuk membantu pengguna dalam menyelesaikan suatu masalah tertentu.

2.4 Jaringan Komputer

Jaringan komputer merupakan gabungan antara teknologi komputer dan teknologi komunikasi. Gabungan teknologi ini melahirkan pengolahan data yang dapat didistribusikan mencakup pemakaian database, Software aplikasi dan peralatan Hardware secara bersamaan, untuk membantu proses otomatisasi perkantoran dan peningkatan kearah efisiensi kerja.

Melalui jaringan komputer inilah kantor, gedung, kota bahkan negara dapat disatukan. Sistem jaringan komputer atau yang lebih dikenal dengan LAN (Local Area Network) adalah solusi yang tepat untuk Sistem Informasi Management (SIM) disebuah perkantoran / perusahaan modern. Dengan LAN dapat menghubungakan sejumlah komputer, komputer mini atau komputer mikro, atau pada umumnya Personal Computer (PC), sehingga dapat mengakses komputer dan periferal seperti printer dan hardisk, bahkan sekarang ini dapat diintegrasikan dengan perangkat mesin atau peralatan telekomunikasi lainnya.

2.4.1 Topologi Jaringan

Topologi jaringan merupakan bagian dari physical layout pada OSI layer, seperti pengkabelan dan interkoneksi antar titik. Ada beberapa topologi dasar yaitu Bus, Star dan Ring. Topologi tersebut harus dipahami terlebih dahulu sebelum interkoneksi dilakukan. Tetapi pada kenyataannya banyak yang

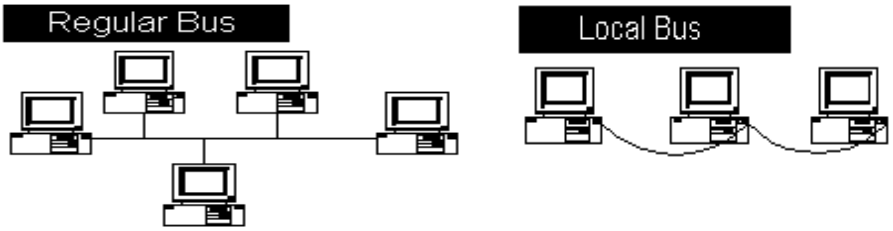
menggunakan gabungan dari topologi dasar tersebut seperti campuran antara Star dan Ring atau Star dan Bus

2.4.1.1 BUS

Topologi BUS merupakan topolgi yang paling sederhana dimana komputer terhubung pada suatu saluran kabel yang menggunakan kabel coaxial yang terhubung satu sama lainnya dan pada ujung-ujungnya menggunakan terminator. Kabel Coaxial pada topologi ini merupakan saluran data primer yang dikenal dengan nama backbone

Pada topologi bus hanya satu komputer pada suatu waktu yang dapat mengirimkan sinyal. Ketika suatu komputer mengirimkan sinyal yang lain mendengarkan. Desain jaringan ini pasif dimana antara komputer yang mengirimkan sinyal dan penerima sinyal sama-sama memonitor data yang ada. Tetapi dalam hal ini komputer tidak memperkuat sinyal yang ada sehingga pada jarak yang jauh, kualitas data akan semakin buruk. Masalah ini disebut dengan redaman yang dapat diatasi dengan repeater untuk memperkuat sinyal.

Topologi ini merupakan pilihan terbaik untuk jaringan dengan skala kecil dengan jumlah komputer yang sedikit karena jaringan ini tidak membutuhkan investasi dana yang besar untuk tambahan hardware yang lain. Tetapi jaringan ini sangat tidak stabil jika ada kerusakan pada pengkabelan maupun terminator yang digunakan. Jika salah satu komputer gagal terhubung ke jaringan maka komputer yang lainpun bias tidak terkoneksi ke jaringan.



Gambar 2.1 Contoh Topologi Bus

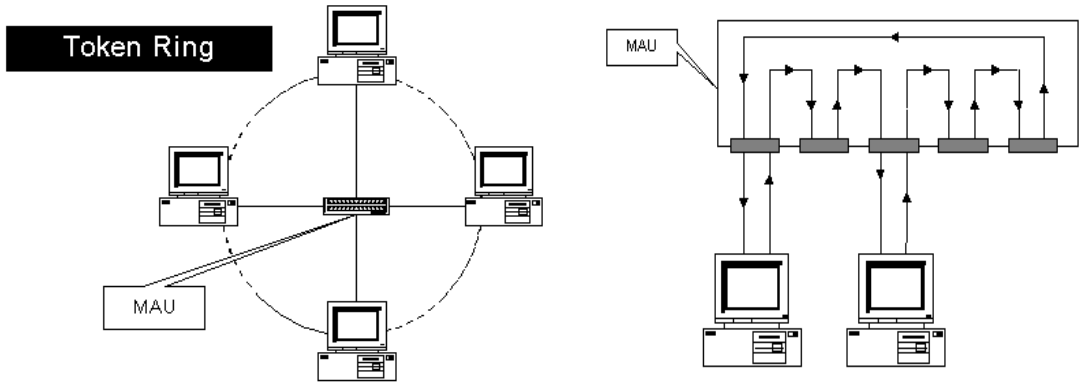
2.4.1.2 Ring

Topologi Ring, sesuai dengan namanya setiap koputer yang terhubung melalui kabel akan membentuk suatu jaringan ring yang tertutup. Karena membentuk jaringan ring yang tertutup maka topologi ini tidak membutuhkan terminator.

Hanya satu komputer pada suatu waktu yang mengirimkan sinyal. Sinyal jaringan akan meninggalkan komputer sumber dan akan mengelilingi seluruh kabel pada satu arah sampai menemukan komputer yang dituju.

Topologi ring ini merupakan suatu design jaringan aktif yang artinya antara komputer sumber dan tujuan sama-sama menerima setiap sinyal jaringan.

Jaringan ring ini dapat mengalami kegagalan transmisi jika kabel yang digunakan mengalami kerusakan sehingga jaringan dengan topologi ini jarang digunakan.



Gambar 2.2 Contoh Topologi Ring

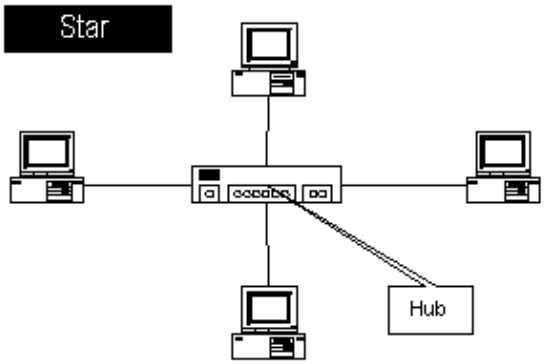
2.4.1.3 Star

Topologi star menggunakan suatu alat tambahan sebagai pusat jaringan yang disebut dengan HUB. Setiap komputer dihubungkan dengan HUB melalui kabel. Hanya satu komputer yang dapat mengirimkan sinyal pada suatu waktu dan semua komputer akan melihat apakah data yang terkirim ditujukan untuk mereka.

Karena jaringan star ini menggunakan satu saluran kabel untuk tiap komputernya maka apabila terdapat kabel yang rusak, hanya komputer tersebut yang tidak dapat terhubung ke jaringan, tetapi bagaiamanpun apabila pusat dari jaringan mengalami kerusakan, maka semua jaringan akan mengalami kerusakan.

Topologi Star ini dapat diterapkan untuk desain jaringan aktif maupun pasif , dimana tergantung oleh HUB yang digunakan. HUB yang pasif aka nmembentuk suatu jaringan pasif dna begitu juga untuk HUB aktif akan membentuk jaringan aktif.

Semua data pada HUB yang pasif dilewatkan pada semua komputer yang terhubung ke HUB. Tetapi untuk HUB yang aktif maka data akan diperkuat sebelum dikirim ke komputer tujuan sehingga data dapat dikirimkan pada jarak yang lebih jauh.



Gambar 2.3 Contoh Topologi Star

2.5 UML (*Unified Modelling Language*)

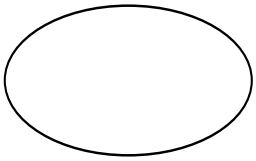
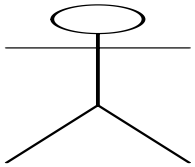
UML adalah sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan *artifacts* dari sistem software untuk memodelkan bisnis, dan sistem software lainnya.

Dengan UML, pemodelan dari suatu sistem dapat lebih mudah di mengerti karena pemodelannya berorientasi terhadap objek *(object oriented)*. Model yang diberikan juga siap dipakai untuk memberikan arah ke pengkodean suatu sistem, dan bukan dimaksudkan untuk menjadi suatu bahasa pemrograman visual. Bahasa pemodelannya yang bebas dari berbagai bahasa pemrograman dan proses rekayasa, sehingga dapat menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.

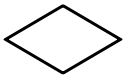
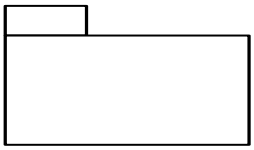
UML adalah language untuk visualizing, specifying, constructing, documenting yang mempunyai kosa kata dan peraturan yg fokus terhadap conceptual dan physical representation dari suatu sistem.

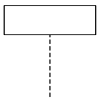
a. *UML sebagai Visualizing*, muncul karena waktu antara berpikir untuk implementasi kemudian langsung meng-coding hampir sama dgn nol, ada istilah yaitu **"You think it, you code it",** sehingga timbul masalah bagi orang lain kalau kode tidak mempunyai standard modelling. Visualisasi dgn model sangat membantu dalam tim development. Visualisasi conceptual model bisa bermasalah jika orang yg terlibat tidak mempunyai presepsi yg sama.
b. *UML sebagai Specifying*, berarti membuat model yg akurat, jelas dan lengkap. UML mampu memberi gambaran spesifikasi dalam analysis, design dan implementasi dari keputusan yang harus diambil dalam developing dan deploying software intensive sistem.
c. *UML sebagai Constructing*, bukan berarti visual programming language, tapi modellingnya dapat langsung berhubungan dengan bahasa pemrograman tertentu atau mapping dari model ke language (contohnya c++, vb, java, dll.). Proses mapping disini ada dua, yaitu:
   1. Forward engineering : model ke language/code generation.
   2. Reverse engineering : language ke model
d. *UML sebagai Documenting*, documentasi software yg baik harus mempunyai semua dokumentasi selain code sendiri.

Tabel 2.1 Daftar simbol

No	Simbol	Dipakai pada	Keterangan
1	 Class	Class diagram, Sequence diagram	Class adalah deskripsi umum dari sekumpulan objek yang mempunyai attribute, operasi relationship dan semantic yang sama
2	 Use case	Use case diagram	Use case adalah deskripsi dari sekumpulan urutan aksi yang yang dilakukan system dan menghasilkan sebuah nilai yang dapat diobservasi oleh actor tertentu
3		Use case diagram	Actor merupakan representasi dari pengguna system yang berada diluar system yang berinteraksi dengan use case serta tidak mempunyai control

	Actor		terhadap use case tersebut
4	message	Sequence diagram	Message adalah suatu perilaku yang terdiri dari message-message yang saling tukar menukar diantara sekumpulan objek dalam ruang lingkup tertentu untuk menyelesaikan maksud tertentu.
5	 Generalization Relationship	Use case diagram, class diagram	Generalization relationship adalah hubungan antara child dengan parent
6		Activity diagram	Decision adalah transisi yang

	 decision		menyatakan suatu tindakan
7	 Package	Use case diagram, Class diagram	Package adalah suatu pengelompokan unsur-unsur model dan diwakili oleh suatu lambang
8	 Activation	Sequence diagram	Activation adaah menunjukkan periode waktu selama suatu aktor atau obyek sedang melakukan/menyelenggarakan suatu tindakan.
9		Sequence diagram	Lifeline menghadirkan

	 Lifeline		keberadaan dari suatu obyek pada situasi tertentu
10	● Initial state	Activity diagram	Initial state adalah suatu status awal menghadirkan status dari suatu obyek (sebelum/di depan)
11	◉ Final state	Activity diagram	Finalstate adalah suatu status akhir menghadirkan penyelesaian aktivitas di dalam memasukkan tindakan atau status menyatakan..

2.5.1 Use Case

Use case adalah sequence dari sebuah transaction dalam sistem yang berfungsi untuk mendapat measurable value (nilai yg terukur) dari actor sistem. Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. Use case mendeskripsikan interaksi tipikal antara para penggun asistem dengan sistem

itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan.

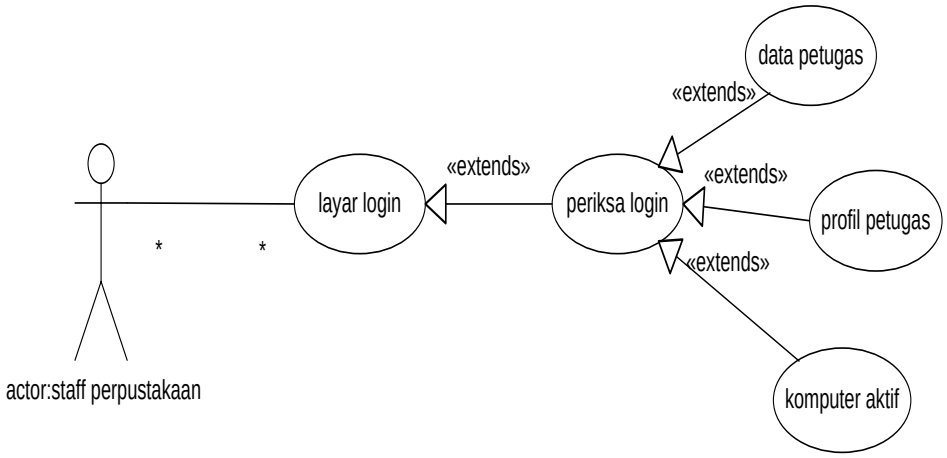
Daripada mendeskripsikan use case secara langsung, akan lebih mudah jika mengintipnya kemudian dan mulai dengan menjabarkan skenario. Skenario adalah rangkaian langkah-langkah yang menjabarkan sebuah interaksi antara seorang pengguna dengan sebuah sistem. Dalam bahasan usecase, pengguna disebut sebagai aktor. Aktor merupakan sebuah peran yang di mainkan seorang pengguna dalam kaitannya dengan sistem. Seorang aktor dapat menggunakan banyak use case, sebaliknya, sebuah use case juga dapat digunakan oleh beberapa aktor. Aktor tidak harus manusia, jika sebuah sistem melakukan sebuah layanan untuk sistem komputer lain, sistem lain tersebut merupakan aktor.

Use case terekenal sebagai sebuah bagian penting dalam UML. Akan tetapi, hal yang mengejutkan adalah jarang terdapat definisi tentang use case dalam UML. Use Case (Class): mendefinisikan gabungan dari use case instances, dimana setiap instance adalah sequence action yg dilakukan sistem yg menghasilkan observable value (nilai yang teramati).

a. Use Case Diagram bukanlah merupakan specification itself (spesifikasi program).
b. Use Case diagram bukanlah software design.
c. Use Case tidak seharusnya memperlihat detail dari UI interaction
d. Abstract Use Case adalah salah satu yg di Include oleh use case lainnya.
e. Abstract Use Case harus digunakan oleh dua atau lebih use case.
f. <<extention>> Use case adalah alternative interaction.

g. <<extention>> (conditional) use case tidak akan pernah menjadi abstract akan tetapi extending use case <<include >> use case lainnya.

h. Use case flow of event akan selalu menggunakan active verb.



Gambar 2.4 Contoh use case diagram

2.5.2 Class Diagram

Class adalah 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 :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
c. *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*.

Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*.

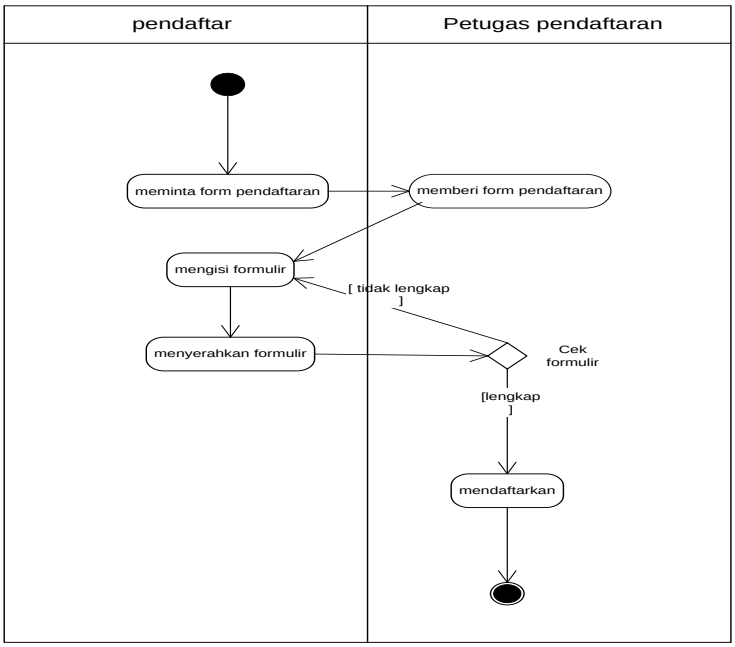
Hubungan Antar *Class*

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 ("terdiri atas..").
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.

2.5.3 Activity Diagram

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 pararel 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 pararel (*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.



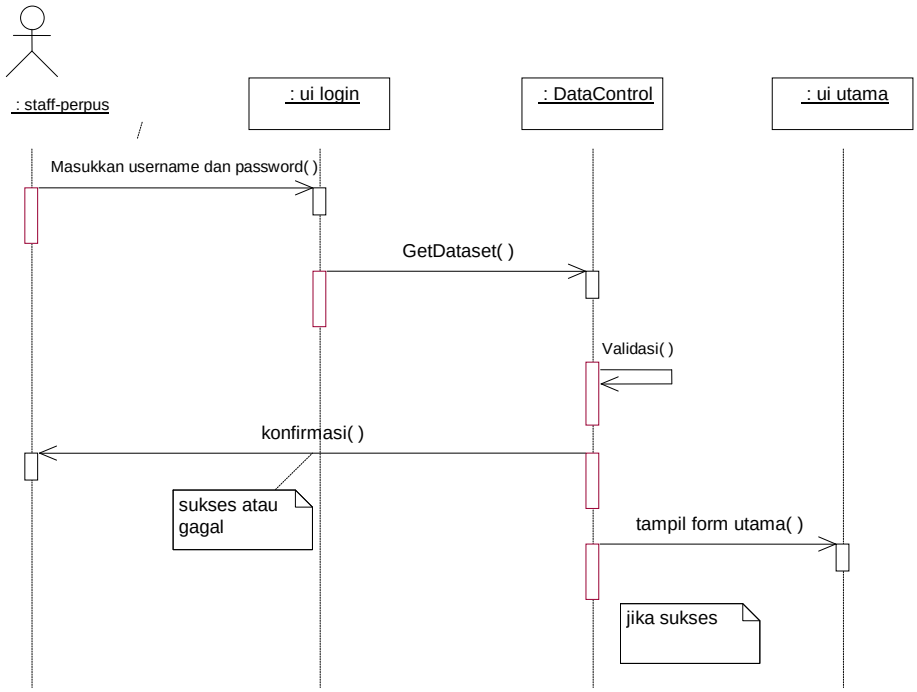
Gambar 2.5 Contoh Activity diagram

2.5.4 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 atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons 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 v*ertikal.

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*.



Gambar 2.6 Contoh Sequence Diagram

2.6 JAVA

2.6.1 Mengenal Bahasa Pemograman Java

Java dikenal sebagai bahasa pemograman tingkat tinggi yang berorientasi objek, atau lazim disebut dengan istilah *Object Oriented Programming* (OOP). Sejarah perkembangan java dimulai pada tahun 1990 ketika James Gosling, seorang ahli pemograman yang bekerja di Sun Microsystem, ditugaskan untuk menciptakan program untuk mengendalikanperangkat elektronk rumah tangga. Disamping itu juga, bahasa yang dihasilkan harus bersifat multiflatform, tidak bergantung pada salah satu vendor. Maka dimulailah proyek tersebut dengan kode *Grend* yang dipimpin oleh Patrick Naughton dan James Gosling.

Mula-mula bahasa pemograman ini dinamakan OAK oleh James Gosling yang mendapa inspirasi dari sebuah pohon yang berada di seberang kantornya. Akan tetapi nama OAK sendiri merupakan nama bahasa pemograman yang telah telah ada sebelumnya, maka kemudian Sun mengganti namanya menjadi Java. Nama java sendiri diinspirasi saat Gosling dan rekan sedang menikmati secangkir kopi disebelah kedai kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebut kata java yang mengandung arti kopi. Akhirnya mereka sepakat untuk memberikan nama bahasa pemograman tersebut dengan Java.

Pada awal tahun 1996, Sun secara resmi merilis versi awal Java yang kemudian terus berkembang hingga muncul JDK 1.1 (*Java Development Kit versi 1.1*). Perkembangan terus dilakukan hingga saat ini, sehingga muncul versi terbaru yang disebut dengan java 2. perubahan utama antara versi sebelumnya dengan

java 2 adalah adanya *swing* yang merupakan teknologi *Grphical User Interface (GUI)* yang mampu menghasilkan aplikasi desktop yang benar-benar portable.

Pada tahun 1998-1999 diluncurkan teknologi java yang berbasis Enterprise yang disebut J2EE *(java 2 Enterprise Edition).* Teknolgi J2EE ini diawali dengan servlet dan EJB kemudian diikuti JSP. Kelebihan java di lingkungan jaringan dan terdistribusi serta memiliki kemampuan *multithreading*, menjadikan java cepat popular dilingkungan server.

2.6.2 Karakteristik dan Kelebihan Java

Java memiliki beberapa karakteristik yang menjadikan Java memiliki kelebihan dibandingkan bahasa pemograman yang lain. Adapun karakteristik tersebut adalah:

1. Sederhana

   Java adalah suatu bahasa sederhana. Para perancang Java berusaha untuk menciptakan suatu bahasa yang mudah dan cepat untuk dipelajari, sehingga bahasa Java dibuat dengan ukuran relative kecil.

2. Berorientasi Objek

   Bahasa pemrograman ini mengarah ke bahasa pemrograman berorientasi objek fitur pemrograman berorientasi objek pada Java pada dasarnya sama dengan C++. Perbedaannya terletak pada kemampuan penurunan berganda *(multiple inheritance)* pada Java yang lebih baik dari C++

3. Dapat didistribusikan dengan mudah

Java memiliki library rutin yang lengkap untuk dirangkai pada protocol TCP/IP, seperti HTTP dan FTP dengan mudah kemampuan jaringan java lebih kuat dan mudah di pakai.

4. Kuat

   Program yang dibuat dengan java dapat dipercaya dalam berbagai hal karena java banyak menekankan pada pengecekan awal untuk menghindari kemungkinan terjadi masalah, pengecekan pada saat runtime dan mengurangi kemungkinan timbulnya masalah *(error)*

5. Aman

   Java dimaksudkan untuk digunakan pada pemrograman jaringan dan system terdistribusi. Dengan demikian, penekanan terutama ditujukan pada masalah keamanan.

6. Portabel

   Tidak seperti pada C dan C++, spesifikasi Java tidak terdapat aspek yang bergantung pada lingkungan implementasi.Ukuran tipe data primitive telah ditentukan sejak awal. Missal, *"int"* selalu berarti sebuah integer 32 bit dua komplemen bertanda (*signed two's complement 32 bit integer*), dan *"float"* selalu berarti sebuah bilangan floating point 32bit (mengikuti spesikasi IEEE 754). Penentuan ukuran tipe data ini akan membuat keseragaman program pada setiap platform, tidak ada tipe data yang tidak kompatibel untuk semua arsitektur peranti keras dan peranti lunak.

7. Multiplatform

   Java dirancang untuk mendukung aplikasi yang dapat beroperasi dilingkungan jaringan yang berbeda. Untuk mengkomodasi hal tersebut, komplilator (*compiler)* java membangkitkan code byte, format yang tidak tergantung pada arsitektur tertentu yang didisain untuk mengirimkan code ke banyak platform peranti keras dan peranti lunak secara efisien. Dengan selogan "*Write Once, Run Anywhere*" (WORA), java dapat dijalan kan pada berbagai platform seperti Linux, Unix,Windows, Soralis maupun Mac.

8. Multithread

   Multitriding adalah kemampuan suatu program komputer untuk melakukan beberapa pekerjaan sekaligus. Thread dalam java juga memiliki kemampuan untuk memanfaatkan kelebihan dari multiproccesor jika sistem informasi yang digunakan mendukung multiproccesor.

9. Interpreter

   Interpreter java dapat mengesekusi *code byte* java secara langsung pada setiap mesin yang terdapat interpreser dan setiap *run-time* java. Pada sistem java, tahap *link* dari program adalah sederhana, bertahap dan ringan.

10. Kinerja yang tinggi

    Java dapat mencapai kinerkerja yang tinggi dengan cara mengadopsi sebuah skema yang memungkinkan interpreter dapat berjalan dengan kecepatan yang penuh tampa perlu memeriksa lingkungan *ran-time*.

11. Dinamis

Dalam beberapa hal, java merupakan bahasa pemograman yang lebih dinamis dibandingan dengan C atau C++. Java dirancang untuk beradaptasi dengan lingkungan yang sedang berkembang.

2.6.3 Cara kerja java

Java merupakan bahasa pemograman kompilator dan juga interpleter yang menjadikan java dapat dijalankan pada platform yang berbeda. Kompilator dapat melakukan kompilasi pada code sumber menjadi bytecode java. Bytecode java merupakan intruksi mesin yang tidak spesifik terhadap processor kamputer dan akan dijalankan pada platform dengan mengunakan Java Virtual Machine (JVM) yang biasa disebut dengan interpleter bytecode atau java run-time interpleter.

2.7 MySQL

MySQL merupakan sistem manajemen basis data SQL yang bersifat open source. MySQL termasuk jenis RDBMS *(Relational Database Management System)*. MySQL dibangun, didistribusikan, dan di dukung oleh MySQL AB.MySQL AB merupakan perusahaan komersial yang dibiayai oleh pengembang MySQL. Pada MySQL, sebuah database terdiri dari tabel-tabel. Sebuah tabel terdiri dari baris dan kolom.

Sebenarnya software MySQL mempunyai dua macam lisensi. Lisensi pertama bersifat open source dengan menggunakan GNU *Generar Public License* dan lisensi kedua berupa *Standard Commercial License* yang dapat dibelidari MySQL AB.

1. MySQL merupakan sistem manajemen database. Database merupakan struktur dari penyimpanan data. Untuk menambah, mengakses, dan memproses data yang disimpan dalam sebuah database komputer maka di perlukan sistem manajemen database seperti MySQL Server.
2. MySQL merupakan sistem manajemen database terhubung *(relational database manajemen system)*. Database terhubung menyimpan data table-tabel terpisah. Hal tersebut akan menambah kecepatan dan fleksibelitasnya. Kata SQL pada MySQL merupakan singkatan dari *Structured Query Language*. SQL merupakan bahasa standar yang digunakan untuk mengakses database dan ditetapkan aleh ANSI/ISO SQL Standard.
3. MySQL merupakan software open source. Open sourceberarti semua orang diijjinkan untuk menggunakan danmemodifikasinya. Semua orang dapat mendownload software MySQL dari internet dan menggunakannya tanpa harus membayar. Dan juga dapat mempelajari source code dan menggunakannya sesuai kebutuhan.
4. MySQL database server mempunyai kecepatan akses yang tinggi, mudah digunakan,serta handal. MySQL dikembangkan untuk menangani database yang besar secara cepat dan lebih sukses digunakan selama bertahun-tahun. Karena konektivitas, kecepatan, dan kemampuannya, MySQL server cocok utuk mengakses database di internet.
5. MySQL Server bekerja di Client/Server atau system embedded. Software database MySQL merupakan sistem client/server yang terdiri dari

multithread SQL server yang mendukung software klien dan library yang berbeda . Administrative tools, dan sejumlah *Application Programming Interface* (APIs).

6. MySQL tersedia dalam beberapa bahasa.

2.7.1 Fitur MySQL

Terdapat fitur-fitur utama dari MySQL adalah sebagai berikut:

1. Ditulis dalam bahasa C dan C++.
2. Bekerja dalam berbagai flatform, misalnya Mac Os X, Solaris, Sun OS, Unix, Windows, dan lain-lain.
3. Menyediakan mesin penyimpanan *(engine storage)* transaksi dan non-transaksi.
4. Server tersedia sebagai program yang terpisah untuk digunakan pada lingkungan jaringan *clien server.*
5. MySQL mempunyai library yang dapat ditempelkan pada aplikasi yang berdiri sendiri *(standalone application)* sehingga aplikasi tersebut dapat digunakan pada komputer yang tidak berada dalam jaringan.
6. Mempunyai sistem password yang fleksibel dan aman.
7. Dapat menangani basis data dalam skala besar. Basis data dalam MySQL Server dapat berisi 50 juta *record.*
8. Klien dapat terkoneksi ke MySQL Server menggunakan socket TCP/IP pada flatform manapun.
9. Server dapat mengirim pesan *error* ke klien dalam berbagai bahasa.

2.7.2 Perintah-Perintah SQL

1. Menambah Data (*Insert*)

Perintah SQL yang digunakan untuk menambah data ke *database* adalah perintah *INSERT*. Perintah ini mempunyai bentuk umum sebagai berikut:

INSERT [INTO] nama_tabel [(daftar *field*/kolom)] VALUES (daftar_data)

2. Memilih/Mengakses Data (*Select*)

Perintah SQL yang digunakan untuk memilih data adalah perintah *SELECT*. Bentuk umum perintah ini adalah sebagai berikut:

SELECT daftar_select FROM daftar_tabel

[WHERE kondisi_pencarian]

[GROUP BY daftar_group_by]

[HAVING kondisi_pencarian]

[ORDER BY daftar_order[ASC │DESC]]

3. Mengedit/Mengubah Data (*Update*)

Perintah SQL yang digunakan untuk mengubah data adalah perintah *UPDATE SET*. Perintah ini mempunyai bentuk umum sebagai berikut:

UPDATE nama_table SET *field*1=databaru1 [,data2=databaru2]

[WHERE kondisi_update]

4. Menghapus Data (*Delete*)

Perintah SQL yang digunakan untuk menghapus data adalah perintah *DELETE FROM*. Perintah ini mempunyai bentuk umum sebagai berikut:

DELETE FROM tabel_hapus WHERE kondisi_hapus