BAB II

LANDASAN TEORI

BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

II.1.1. Sistem

Sistem adalah prosedur logis dan rasional untuk merancang suatu rangkaian komponen yang berhubungan satu dengan yang lainnya dengan maksut untuk berfungsi sebagai suatu kesatuan dalam usaha mencapai suatu tujuan yang telah ditentukan. (*Ika Nur Indah ; 2013 : 125*)

II.1.2. Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. (*Ika Nur Indah ; 2013 : 125*)

II.1.3. Sistem Informasi

Menurut Laudon, K. C dan Laudon, J.P didalam buku yang berjudul "Pengenalan Sistem Informasi" (Kadir, A, 1992). Sistem informasi adalah kombinasi antara prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi. Sistem informasi dapat didefinisikan sebagai sekumpulan komponen yang saling berhubungan, mengumpulkan, memproses, menyimpan dan mendistribusikan informasi untuk menunjang pengambilan keputusan dan pengawasan dalam suatu organisasi. Sistem informasi menerima masukan data dan instruksi, mengolah data tersebut sesuai instruksi, dan mengeluarkan hasilnya. Untuk penerapan pengolahan

informasi dapat dianalisis menjadi masukan, penyimpanan, pengolahan dan keluaran. (*Dessi Tri Santi ; 2014 : 7-8*)

II.2. Sistem Pakar

Sistem pakar adalah sistem berbasis komputer yang menggunakan pengetahuan, fakta, dan teknik penalaran dalam memecahkan masalah yang biasanya hanya dapat dipecahkan oleh seorang pakar dalam bidang tersebut (*Martin* dan *Oxman*, 1998). Pada dasarnya sistem pakar diterapkan untuk mendukung aktivitas pemecahan masalah. Beberapa aktivitas pemecahan yang dimaksud antara lain: pembuatan keputusan (*decision making*), pemaduan pengetahuan (*knowledge fusing*), pembuatan desain (*designing*), perencanaan (*planning*), prakiraan (*forecasting*), pengaturan (*regulating*), pengendalian (*controlling*), diagnosis (*diagnosing*), perumusan (*prescribing*), penjelasan (*explaining*), pemberian nasihat (*advising*) dan pelatihan (*tutoring*). Selain itu sistem pakar juga dapat berfungsi sebagai asisten yang pandai dari seorang pakar (*Martin* dan *Oxman*, 1998). (*Aryati Wuryandari* ; *2013 : 74*)

Untuk membangun sistem yang difungsikan untuk menirukan seorang pakar manusia harus bisa melakukan hal-hal yang dapat dikerjakan oleh para pakar. Untuk pembangun sistem yang seperti itu maka komponen-komponen dasar yang minimal harus dimiliki menurut *Yasidah Nur Istiqomah 2013* adalah sebagai berikut:

1. Antar muka (*user interface*).
2. Basis pengetahuan (*knowledge base*).
3. Mesin inferensi (*Inference Engine*).

II.2.1. Basis Pengetahuan (*knowledge base*)

Basis pengetahuan berisi pengetahuan-pengetahuan dalam penyelesaian masalah, tentu saja didalam domain tertentu. Ada 2 bentuk pendekatan basis pengetahuan yang sangat umum digunakan, yaitu:

1. Penalaran berbasis aturan (*Rule-Based Reasoning*)

   Pada penalaran berbasis aturan, pengetahuan direpresentasikan dengan menggunakan aturan berbentuk: IF-THEN. Bentuk ini digunakan apabila kita memiliki sejumlah pengetahuan pakar pada suatu permasalahan tertentu, dan pakar dapat menyelesaikan masalah tersebut secara berurutan. (*Yasidah Nur Istiqomah* ; *2013 : 34*)

2. Penalaran berbasis kasus (*Case-Based Reasoning*)

   Pada penalaran berbasis kasus, basis pengetahuan akan berisi solusi-solusi yang telah dicapai sebelumnya, kemudian akan diturunkan suatu solusi untuk keadaan yang terjadi sekarang (fakta yang ada). (*Yasidah Nur Istiqomah* ; *2013 : 34*)

II.3. Metode Naive Bayesian

Simple naive Bayesian classifier merupakan salah satu metode pengklasifikasi berpeluang sederhana yang berdasarkan pada penerapan Teorema *Bayes* dengan asumsi antar variabel penjelas saling bebas (independen). Algoritma ini memanfaatkan metode probabilitas dan statistik yang dikemukakan oleh ilmuwan Inggris *Thomas Bayes*, yaitu memprediksi probabilitas di masa depan berdasarkan pengalaman di masa sebelumnya. Dua kelompok peneliti, satu

oleh *Pantel* dan *Lin*, dan yang lain *oleh Microsoft Research* memperkenalkan metode statistik *Bayesian* ini pada teknologi anti *spam filter*.Tetapi yang membuat algoritma *Bayesian filtering* ini popular adalah pendekatan yang dilakukan oleh *Paul Graham*. (*Budanis Dwi Meilani ; 2014 : 1-2*)

Menurut Arief Kurniawan untuk menentukan nilai probabilitas untuk setiap kategori, maka teorema bayes dirumuskan pada persamaan (1), sebagai berikut :

$$P(E) = \frac{x}{n} \quad \text{..... (1)}$$

P = Probabilitas
E = *Event* (kejadian)
x = nilai sampel
n = jumlah seluruh sampel

Untuk menentukan nilai likelihood maka digunakan persamaan (2) sebagai berikut:

$$P(X|C_i) = P(x_i|C_i) \text{ x } P(x_2|C_i) \text{x} \ldots . \text{x } P(x_2|C_i) \quad \text{..... (2)}$$

II.4. *Visual Basic*

Visual Basic adalah salah satu bahasa pemrograman berbasis *desktop* yang dikeluarkan oleh perusahaan perangkat lunak komputer terbesar yaitu *Microsoft*. Pada dasarnya *Visual Studio .NET* didesain untuk menampung berbagai macam bahasa pemrograman dan terlingkup dalam *Visual Studio .NET*. Dengan demikian, dapat membangun aplikasi-aplikasi *Windows* di dalam *Visual Studio .NET*. *Visual Basic .NET* merupakan salah satu bahasa pemrograman yang dapat digunakan untuk membuat program aplikasi. (*Rian Aldy Hidayat ; 2013 : 253*)

II.5. *Database*

Database merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan diperangkat keras komputer dan digunakan diperangkat lunak untuk memanipulasinya (Jogiyanto HM : 1999:711). *Database* merupakan salah satu komponen yang sangat penting dalam sistem informasi, karena merupakan basis sistem dalam menyediakan informasi bagi para pemakai.(*Indra Warman ; 2012 : 45*)

II.5.1. Jenis-Jenis *Database*

1. *Database* Hirarki

Yaitu suatu data yang tersusun dengan bentuk hirarki pohon. Susunan yang seperti ini terdiri dari beberapa unsur komponen yang saling mempengaruhi dan tidak dapat dipisahkan, jenis *database* ini merupakan hubungan satu komponen dengan banyak komponen. (*Indra Warman ; 2012 : 46*)

2. *Database* Relasi

Adalah suatu data yang disusun dalam bentuk tabel yang terdiri dari dua definisi dan tersusun secara terstruktur. Bentuk susunan dua dimensi ini terdiri dari beberapa kolom dan *record* yang tersusun berbentuk baris dari kiri kekanan. Data- data yang susunannya berbentuk barus adalah susunan yang menurun kebawah. Dimana pada setiap baris berisikan data- data yang saling berkaitan satu sama lainnya. Artinya setiap pemasukan data yang tersimpan pada *field* merupakan kesatuan dalam bentuk satu baris.(*Indra Warman ; 2012 : 46*)

II.5.2. Prinsip-Prinsip *Database*

Sistem database manajemen dibentuk untuk mengurangi masalah-masalah dalam organisasi. Misalnya data/informasi tidak tersedia atau saling tumpang tindih. Berikut prinsip manajemen database menurut *Anis nurhanafi* (2013 : 3) adalah:

1. Ketersediaan

   Data mudah diakses oleh suatu program dan pemakai (*user*) dimanapun dan kapanpun diperlukan.

2. Pemakaian bersama

   Struktur data disusun sedemikian hingga dapat digunakan oleh beberapa pemakai bersama-sama untuk mengurangi redudansi data.

3. Pengembangan

   Databases dapat dikembangkan sesuai dengan perkembangan kebutuhan pemakai. *Databases* dapat dimodifikasi untuk pengembangan selanjutnya dan dapat beradaptasi dengan lingkungan.

4. Kesatuan

   Databases dibentuk dalam satu kesatuan untuk memudahkan pengotrolannya (pemeliharaan dan pengawasan) mudah dilakukan.

II.5.3. Pengertian Database Management System (DBMS)

Database Management System (DBMS) adalah satu koleksi data yang saling berelasi dan satu *set* program untuk mengakses data tersebut. Jadi DBMS terdiri dari *database* dan *set* program pengelola untuk menambah, menghapus data, mengambil data dan membaca data. *Database* adalah suatu koleksi data komputer yang terintegrasi, diorganisasikan dan disimpan dalam suatu cara yang memudahkan pengambilan kembali (*Raymond McLeod, Jr*. Jilid 1 Edisi Bahasa Indonesia, 1995). Sedangkan *set* program adalah paket program yang diolah dan dibuat untuk memudahkan dalam pemasukkan atau pembuatan data. Menurut *Date,* basis data dapat dianggap sebagai tempat untuk sekumpulan berkas data terkomputerisasi. (*Anis Nurhanafi ; 2013 : 3*)

II.6. MySql

MySQL Server 2000 adalah suatu Perangkat lunak *Relational Database Mangement system* (RDBMS) yang handal. Didesain untuk mendukung proses transaksi yang besar (seperti *order entri* yang *online*, inventori, akuntansi atau manufaktur). *MySQL Server* akan secara otomatis menginstal enam *database* utama, yaitu *master*, *model*, *tempdb*, *pubs*, *Northwind*, dan *Msdb*. (*Anis nurhanafi ; 2013 : 5*)

II.7. UML (*Unified Modelling 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 bahasa-bahasa 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*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan

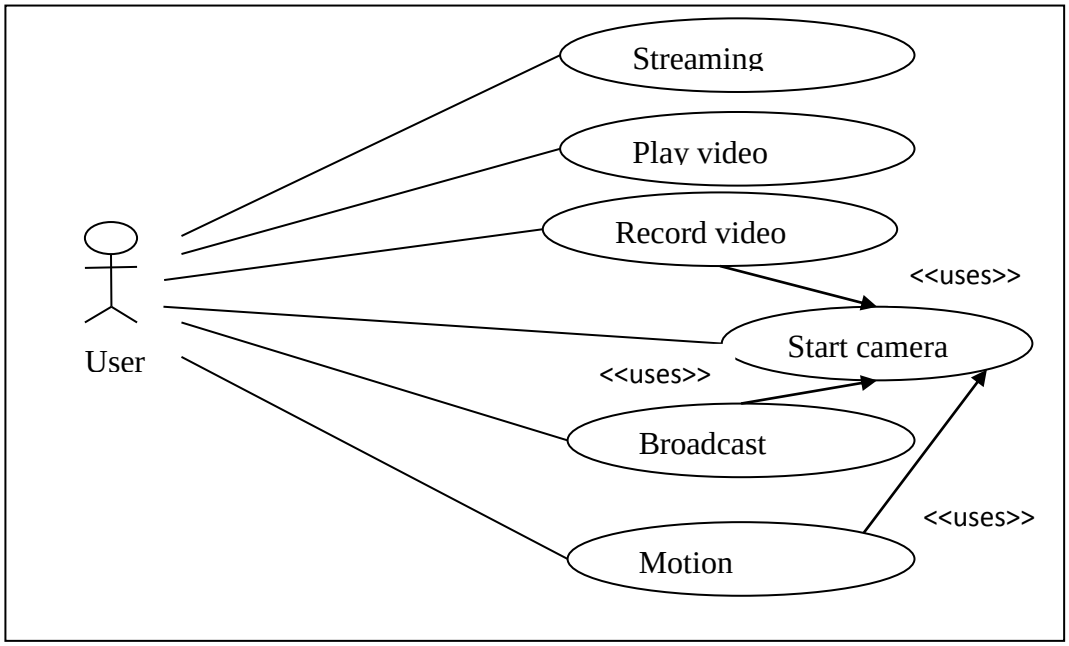
masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 *Booch, Rumbaugh dan Jacobson*, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease *draft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG – http://www.omg.org). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. *Booch, Rumbaugh* dan *Jacobson* menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek. *(Yuni Sugiarti ; 2013 : 33)*

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case* Diagram diterangkan dibawah ini.

1. *Use Case Diagram*

Use case diagram adalah diagram yang menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar dan menjelaskan sistem secara fungsional yang terlihat pengguna. Yang ditekankan dalam *use case* adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". *Use case* mempresentasikan sebuah interaksi antara aktor dan

sistem. *Use case* dapat digunakan selama proses analisis untuk menangkap persyaratan sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use case* diagram berperan untuk mendapatkan perilaku sistem saat diimplementasikan. Komponen pembentuk diagram use case yang pertama adalah *actor*. *Actor* mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan inputan pada sistem, hanya menerima informasi dari sistem atau keduanya. *Actor* hanya berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. *Actor* digambarkan dengan *stick man.* Selain itu komponen pemebentuk lainnya adalah *use case*. *Use case* adalah gambaran fungsionalitas dari sistem sehingga pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.(*Rian Aldy Hidayat ; 2012 : 253*)

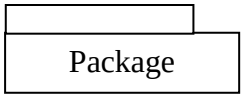
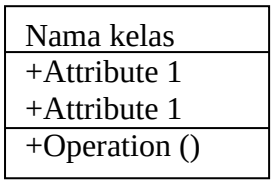
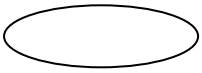
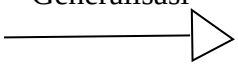
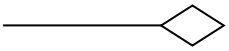


Gambar II.1. *Use Case* Diagram
Sumber : (*Rian Aldy Hidayat ; 2012 : 253*)

2. *Class Diagram*

Diagram kelas atau *class* diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol pada diagram kelas :

Tabel II.1. *Class* Diagram

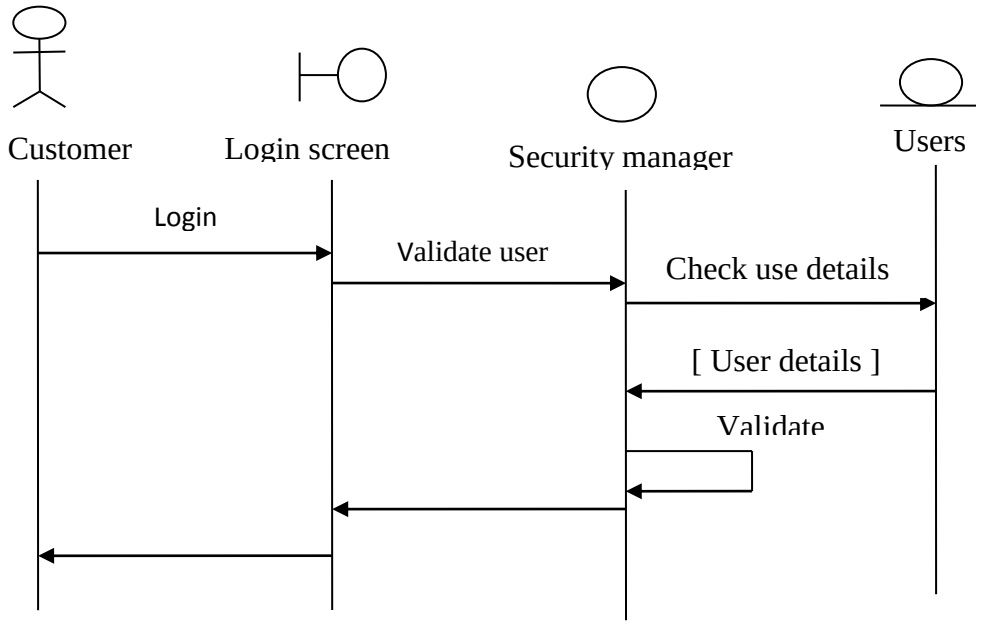
Simbol	Deskripsi
Package 	Package merupakan bungkusan dari satu atau lebih kelas.
Operasi 	Kelas pada struktur sistem.
Antar muka / Interface 	Sama dengan konsep interface dalam pemrograman berorientasi objek.
Asosiasi/Association	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity.
Asosiasi berarah 	Relasi antar kelas dengan makana kelas yang satu digunakan olah kelas yang lain, asosiasi juga disertai dengan multiplicity.
Generalisasi 	Relasi antar kelas dengan makna generalisasi – spesialisasi (umum-khusus).
Kebergunaan / defedency	Relasi Antar kelas dengan makna kebergantungan antar kelas.
Agrasiasi 	Relasi antar kelas dengan makna semua bagan (whole part).

Sumber : (Yuni Sugiarti ; 2013 ; 59)

3. *Sequence Diagram*

Diagram *Sequence* menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.



Gambar II.2. ***Contoh Sequence Diagram***
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

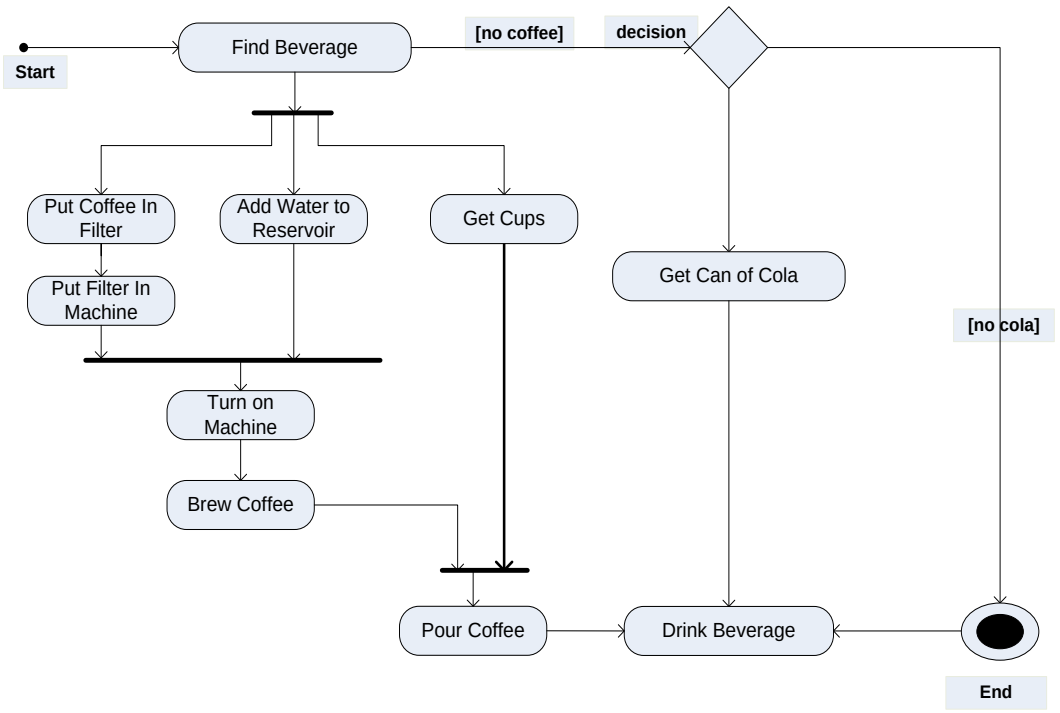
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

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.



Gambar II.3. *Activify* Diagram

Sumber : (Yuni Sugiarti ; 2013 : 76)