BAB II

LANDASAN TEORI

* + 1. **Sistem**

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan kegiatan atau untuk melakukan sasaran yang tertentu. Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urutan-urutan operasi di dalam sistem. Karakteristik sistem terdiri dari :

1. Komponen Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

1. Batasan Sistem

Batasan merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang suatu kesatuan. Batasan suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

1. Lingkungan Luar Sistem

Lingkungan luar sistem (*environment*) adalah diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkung luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

1. Penghubung Sistem

Penghubung sistem merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lain.

1. Masukan Sistem

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintance input*) dan masukan sinyal (*signal input*).

1. Keluaran Sistem

Keluaran sistem adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

1. Pengolah Sistem

Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah akan mengubah masukan menjadi keluaran.

1. Sasaran Sistem

Suatu sistem mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya (Jeperson Hutahaean ; 2014 : 2).

* + 1. **Sistem Pakar**

Sistem pakar menirukan perilaku seorang pakar dalam menangani suatu persoalan. Pada suatu kasus seorang pasien mendatangi dokter untuk memeriksa badannya yang mengalami penyakit kesehatan, maka dokter atau pakar kesehatan akan memeriksa dan melakukan diagnosa. Bila dokter cukup sibuk dan pelaksana diagnosa digantikan oleh sebuah sistem pakar, maka sistem pakar diharapkan dapat membantu memahami dan menganalisa keadaan pasien dan menemukan penyakit yang diderita pasien itu. Sistem pakar diharapkan juga untuk menghasilkan dugaan atau hasil diagnosa yang sama dengan diagnosa yang dilakukan oleh seorang ahli. Tujuan utama sistem pakar bukan untuk menggantikan kedudukan seorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan pengalaman pakar-pakar yang ahli di bidangnya (Andri Saputra ; 2011 : 203).

Salah satu teknik kecerdasan buatan yang menirukan proses penalaran manusia adalah Sistem Pakar. Secara umum, Sistem Pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli Sistem Pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja para ahli. Dengan Sistem Pakar ini, orang awam juga dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, Sistem Pakar ini juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman (Jurnal Rahayu : 3).

* + 1. **Konsep Dasar Sistem Pakar**

Konsep dasar sistem pakar mengandung : keahlian, ahli, pengalihan keahlian, inferensi, aturan dan kemampuan menjelaskan. Keahlian adalah suatu kelebihan penguasaan pengetahuan di bidang tertentu yang diperoleh dari pelatihan, membaca atau pengalaman. Contoh bentuk pengetahuan yang termasuk keahlian adalah :

1. Fakta-fakta pada lingkup permasalahan tertentu.
2. Teori-teori pada lingkup permasalahan tertentu.
3. Prosedur-prosedur dan aturan-aturan berkenaan dengan lingkup permasalahan tertentu.
4. Strategi-strategi global untuk menyelesaikan masalah.
5. Meta-*knowledge* (pengetahuan tentang pengetahuan) (Andri Saputra, 2011 : 203)
   * 1. **Mesin Inferensi (*Inference Engine*) Sistem Pakar**

Inferensi digunakan dalam sistem pakar untuk memperoleh informasi

terbaru dari informasi yang sudah ada. Diataranya :

1. *Forward Chaining*

Adalah strategi inferensi yang dimulai dengan sekumpulan fakta, fakta baru yang diperoleh dengan menggunakan rule, dimana alasan yang digunakan sesuai dengan fakta yang ada, dan melanjutkan proses ini sampai goal diraih atau sampai tidak ada rule selanjutnya yang mempunyai alasan yang sesuai dengan fakta yang ada maupun fakta yang diketahui.

1. *Backwad Chaining*

Adalah strategi inferensi yang diperoleh untuk membuktikan suatu hipotesis dengan dukungan informasi (Rizki Fitriani Maiza ; 2010 : 10).

* + 1. **Aktivitas Sistem Pakar**

Bentuk-bentuk ini memungkinkan para ahli untuk dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan ahli. Seorang ahli adalah seseorang yang mampu menjelaskan suatu tanggapan, mempelajari hal-hal baru seputar topik permasalahan (domain), menyusun kembali pengetahuan jika dipandang perlu, memecah aturan-aturan jika dibutuhkan, dan menentukan relevan tidaknya keahlian mereka. Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu :

1. Tambahan pengetahuan (dari para ahli atau sumber-sumber lainnya).
2. Representasi pengetahuan (ke komputer).
3. Inferensi pengetahuan.
4. Pengalihan pengetahuan ke user (Andri Saputra, 2011 : 203)
   * 1. **Bentuk Sistem Pakar**

Pengetahuan yang disimpan di komputer disebut dengan nama basis pengetahuan. Ada 2 tipe pengetahuan, yaitu : fakta dan prosedur (biasanya berupa aturan). Salah satu fitur yang harus dimiliki oleh sistem pakar adalah kemampuan untuk menalar. Jika keahlian-keahlian sudah tersimpan sebagai basis pengetahuan dan sudah tersedia program yang mampu mengakses basisdata, maka komputer harus dapat diprogram untuk membuat inferensi. Proses inferensi ini dikemas dalam bentuk motor inferensi (inference engine). Sebagian besar sistem pakar komersial dibuat dalam bentuk rule-based systems, yang mana pengetahuannya disimpan dalam bentuk aturan-aturan. Aturan tersebut biasanya berbentuk IF-THEN. Fitur lainnya dari sistem pakar adalah kemampuan untuk merekomendasi. Kemampuan inilah yang membedakan sistem pakar dengan sistem konvensional.

Ada 4 bentuk sistem pakar, yaitu :

1. Berdiri sendiri. Sistem pakar jenis ini merupakan software yang berdiri-sendiri tidak tergantung dengan software yang lainnya.
2. Tergabung. Sistem pakar jenis ini merupakan bagian program yang terkandung didalam suatu algoritma (konvensional), atau merupakan program dimana didalamnya memanggil algoritma subrutin lain (konvensional).
3. Menghubungkan ke software lain . Bentuk ini biasanya merupakan sistem pakar yang menghubungkan ke suatu paket program tertentu, misalnya DBMS.

Sistem Mengabdi. Sistem pakar merupakan bagian dari komputer khusus yang dihubungkan dengan suatu fungsi tertentu. Misalnya sistem pakar yang digunakan untuk membantu menganalisis data radar (Andri Saputra, 2011 : 203).

* + 1. **Struktur Sistem Pakar**

Sistem pakar disusun oleh dua bagian utama, yaitu :

1. Lingkungan pengembangan (*development environment*), digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar.
2. Lingkungan konsultasi (*consultation environment*), digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar (Andri Saputra, 2011 : 204).
   * 1. **Antarmuka Pengguna**

Antarmuka pengguna *(user interface)* merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu, antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai. Menurut McLeod (1995), pada bagian ini terjadi dialog antara program dari pemakai,yang memungkinkan sistem pakar menerima instruksi dan informasi *(input)* dari pemakai, juga memberika informasi *(output)* kepada pemakai.

* + 1. **Basis Pengetahuan**

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang obyek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara memperoleh fakta baru dari fakta yang telah diketahui.

Dalam studi kasus pada sistem berbasis pengetahuan, terdapat beberapa karakteristik dibangun yang akan membantu kita dalam membentuk serangkaian prinsip-prinsip arsitekturnya. Prinsip tersebut meliput :

1. Pengetahuan merupakan kunci kekuatan sistem pakar.
2. Pengetahuan sering tidak pasti dan tidak lengkap.
3. Pengetahuan sering miskin spesifikasi
4. Amatir menjadi ahli secara bertahap
5. Sistem pakar harus fleksibel
6. Sistem pakar harus transparan

Sejarah penelitian di bidang AI telah menunjukkan berulang kali bahwa basis pengetahuan adalah kunci untuk setiap sistem cerdas *(intelligence system).*

* + 1. **Akuisisi Pengetahuan**

Akuisisi pengetahuan *(knowledge acquistion)* adalah akumulasi, transfer dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan ke dalam program komputer. Dalam tahap ini, *knowledge engineer* berusaha menyerap pengetahuan untuk selanjutnya ditransfer ke dalam basis pengetahuan. Pengetahuan diperoleh dari pakar, dilengkapi bukum basis data, laporan penelitian dan pengalaman pemakai. Menurut Turban (2001), terdapat tiga metode utama dalam akuisisi pengetahuan, yaitu :

1. Wawancara

Wawancara adalah metode akuisisi yang paling banyak digunakan. Metode ini melibatkan pembicaraan dengan pakar secara langsung dalam suatu wawancara. Terdapat beberapa bentuk wawancara yang dapat digunakan. Masing-masing bentuk wawancara tersebut mempunyai tujuan yang berbeda.

1. Analisa protokol

Dalam metode ini akuisisi ini, pakar diminta untuk melakukan suatu pekerjaan dan mengungkapkan proses pemikirannya dengan menggunakan kata-kata. Pekerjaan tersebut direkam, dituliskan, dan dianalisis.

1. Observasi pada pekerjaan pakar

Dalam metode ini, pekerjaan dalam bidang tertentu yang dilakukan pakar direkam dan diobservasi.

1. Induksi aturan dari contoh

Metode ini dibatasi untuk sistem berbasis aturan. Induksi adalah suatu proses penalaran dari khusus ke umum. Suatu sistem induksi aturan diberi contoh-contoh daru suatu masalah yang hasilnya telah diketahui. Setelah diberikan beberapa contoh, sistem induksi aturan tersebut dapat membuat aturan yang benar untuk kasus-kasus contoh. Selanjutnya, aturan dapat digunakan untuk menilai kasus lain yang hasilnya tidak diketahui.

* + 1. **Perbaikan Pengetahuan**

Pakar memiliki kemampuan untuk menganalisis dan meningkatkan kinerjanya serta kemampuan untuk belajar dari kinerjanya. Kemampuan tersebut adalah penting dalam pembelajaran terkomputerisasi sehingga program akan mampu menganalisis penyebab kesuksesan dan kegagalan yang dialaminya.

* + 1. **Metode *Case Based Reasoning***

Menurut Toba *et al* Case Based Reasoning (CBR) merupakan salah satu metode pemecahan masalah yang dalam mencari solusi dari suatu kasus yang baru, sistem akan melakukan pencarian terhadap solusi dari kasus lama yang memiliki permasalahan yang sama dan sudah pernah terjadi sebelumnya. Terdapat empat proses yang terjadi pada metode CBR dalam menyelesaikan masalah, yaitu :

1. *Retrieve*

Mendapatkan/memperoleh kembali kasus yang paling menyerupai/relevan (*similar*) dengan kasus yang baru. Tahap *retrieval* ini dimulai

dengan menggambarkan/ menguraikan sebagian masalah, dan diakhiri jika ditemukannya kecocokan terhadap masalah sebelumnya yang tingkat kecocokannya paling tinggi.

1. *Reuse*

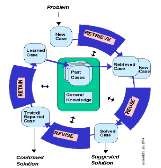
Memodelkan/menggunakan kembali pengetahuan dan informasi kasus lama berdasarkan bobot kemiripan yang paling relevan ke dalam kasus yang baru, sehingga menghasilkan usulan solusi dimana mungkin diperlukan suatu adaptasi dengan masalah yang baru tersebut.

1. *Revise*

Meninjau kembali solusi yang diusulkan kemudian mengetesnya pada kasus nyata (simulasi) dan jika diperlukan memperbaiki solusi tersebut agar cocok dengan kasus yang baru.

1. *Retain*

Mengintegrasikan/menyimpan kasus baru yang telah berhasil mendapatkan solusi agar dapat digunakan oleh kasus-kasus selanjutnya yang mirip dengan kasus tersebut.

Empat proses masing-masing melibatkan sejumlah langkah-langkah spesifik, yang akan dijelaskan pada gambar 1 berikut ini:

Gambar 1 Metode *Case Based Reasoning*

**13. *PHP***

PHP singkatan dari PHP: *Hypertext Preprocessor* yang digunakan sebagai bahasa *script serverside* dalam pengembangan web yang disisipkan dalam dokumen HTML. Penggunaan PHP memungkinkan web dapat dinamis sehingga maintenance situs web tersebut menjadi lebih mudah dan efisien. PHP merupakan software Open-source yang disebarkan dan dilisensikan secara gratis serta dapat di download secara bebas dari situs resminya <http://www.php.net>. (Wibowo, 2013 : 63).

PHP adalah suatu bahasa pemrograman *Open Source* yang digunakan secara luas terutama untuk pengembangan web dan dapat disimpan dalam bentuk HTML.Keuntungan utama menggunakn PHP adalah script PHP tidak benarbenar sederhana bagi pemula, tetapi menyediakan banyak fitur tambahan untuk programmer professional. Meskipun PHP lebih difokuskan sebagai script Server Side. Script PHP dapat digunakan dalam 3 hal, yaitu:

1. Penulisan program *Server Side.* Hal ini adalah target utam PHP. Diperlukan tiga hal agar script PHP dapat bekerja antara lain, PHP *Parser* (CGI atau *Server module),* server web dan browser web. menjalankan server web terlebih dahulu, kemudian mengakses keluaran program PHP melalui browser web dan melihat halaman web.
2. Penulisan program *Command Line.* Script PHP dapat berjalan tanpa server atau browser. Hanya diperlukan PHP Parser dalam bentuk *Command Line.*

Penulisan program aplikasi desktop. (Wibowo, 2013 : 63-64)

**14. *Unified Modelling Language* (UML)**

Sukamto dan Shalahuddin (2013:133), “*UML* (*Unified Modeling Language)* adalah salah satu standar bahasa yang banyak digunakan di duniaindustri untuk mendefinisikan requirement, membuat analisis dan desain, sertamenggambarkan arsitektur dalam pemrograman berorientasi objek”.

Widodo dan Herlawati (2011:6), menyatakan bahwa “*UML* singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standar. *UML* juga dapat diartikan sebagai bahasa yang memiliki sintaks dan semantik”.  *UML* menyediakan serangkaian gambar dan diagram yang sangat baik. Beberapa diagram memfokuskan diri pada ketangguhan teori *object-oriented* dan sebagian lagi memfokuskan pada detail rancangan dan konstruksi. Semua dimaksudkan sebagai sarana komunikasi antar *team programmer* maupun dengan pengguna.



**Gambar 1.** Tampilan Logo *UML*

Sumber : Widodo dan Herlawati (2011:6-7)

*UML* diaplikasikan untuk maksud tertentu,

biasanya antara lain untuk :

* + - * 1. Merancang perangkat lunak.
        2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
        3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
        4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Berikut tabel Diagram UML:

**Tabel 1.** Tipe Diagram *UML*

|  |  |  |
| --- | --- | --- |
| No | Diagram | Tujuan |
| 1 | *Class* | Memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi |
| 2 | *Package* | Memperlihatkan kumpulan kelas-kelas, merupakan  dari diagram komponen |
| 3 | *Use case* | Diagram ini memperlihatkan himpunan *use case*  dan aktor-aktor (suatu jenis khusus dari kelas) |
| 4 | *Sequence* | Diagram interaksi yang menekankan pada  pengiriman pesan dalam suatu waktu tertentu |
| 5 | *Communication* | Sebagai pengganti diagram kolaborasi *UML* 1.4  yang menekankan organisasi struktural dari obyek- obyek yang menerima serta mengirim pesan |
| 6 | *Statechart* | Diagram status memperlihatkan keadaan-keadaan  pada sistem, memuat status *(state),* transisi, kejadian serta aktivitas |
| 7 | *Activity* | Tipe khusus dari diagram status yang  memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem |
| 8 | *Component* | Memperlihatkan organisasi serta kebergantungan  sistem/perangkat lunak pada komponen- komponen yang telah ada sebelumnya |
| 9 | *Deployment* | Memperlihatkan konfigurasi saat aplikasi  dijalankan (*run-time*) |

*Sumber : Widodo dan Herlawati (2011:10-12)*

**15. Diagram *Use Case* *(Use Case Diagram)***

Sukamto dan Shalahuddin (2013:155), “*use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat”.

Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Adapun simbol-simbol dan contoh diagram *use case* yang digunakan dalam *use case* adalah sebagai berikut

**Tabel 2.** Simbol-simbol *Use Case* Diagram

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Actor* | Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan use case. |
| 2 | ---------> | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (independent) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (independent). |
| 3 |  | *Generalization* | Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor). |
| 4 | -------🡪 | *Include* | Menspesifikasikan bahwa use case sumber secara eksplisit. |
| 5 |  | *Extend* | Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan |
| 6 |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7 |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8 |  | *Use Case* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 9 |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |
| 10 |  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

*Sumber : Sukamto dan Shalahuddin (2013:156-158)*

Komponen pembentuk diagram *use case* adalah :

1. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use case*, aktivitas / sarana yang disiapkan oleh bisnis / sistem.
3. Hubungan (*link*), aktor mana saja yang terlibat dalam *use cas*

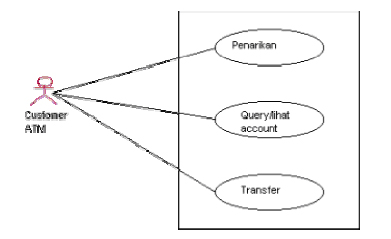
Setiap *use case* dilengkapi dengan skenario. Skenario *use case* adalah alur jalannya proses *use case* dari sisi aktor dan sistem. Berikut adalah format tabel skenario *use case.*

**Tabel 3.** Format Tabel Skenario *Use Case*

|  |  |
| --- | --- |
| **AKSI AKTOR** | **REAKSI SISTEM** |
| Skenario Normal |  |
|  |  |
| Skenario Alternatif |  |
|  |  |

*Sumber : Sukamto dan Shalahuddin (2013:156-158*

Skenario *use case* dibuat per *use case* terkecil, misalkan generalisasi maka skenario yang dibuat adalah *use case* yang lebih khusus. Skenario normal adalah skenario bila sistem berjalan normal tanpa terjadi kesalahan atau *error*. Sedangkan skenario alternatif adalah skenario bila sistem tidak berjalan normal atau mengalami *error*.

Dan berikut contoh diagram *use case :*

http://estrishinta.web.ugm.ac.id/aurora/wordpress/

**16. Diagram Kelas *(Class Diagram)***

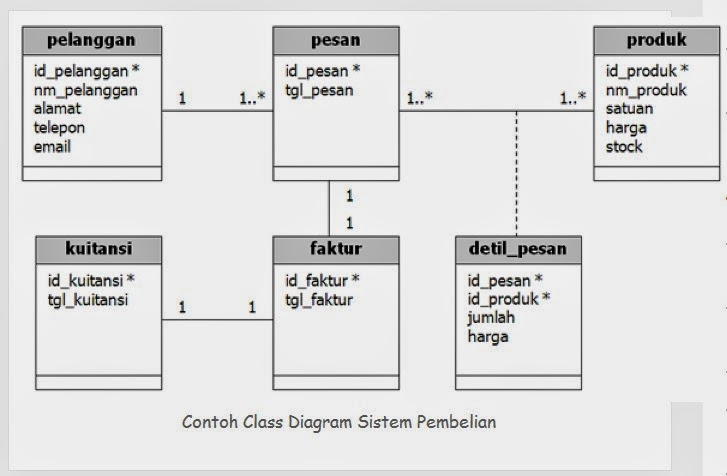
Class Diagram merupakan himpunan dari objek-objek yang sejenis. Sebuah objek memiliki keadaan sesaat (state) dan perilaku (behavior). State sebuah objek adalah kondisi objek tersebut yang dinyatakan dalam atribute/properties. Sedangkan perilaku suatu objek mendefenisikan bagaimana sebuah objek bertindak/berinteraksi dan memberikan reaksi.

**Tabel 4 Simbol – simbol *Class* Diagram**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Gambar** | **Nama** | **Keterangan** |
| 1 |  | *Class* | Class adalah blok-blok pembangun pada pemrograman berorientasi obyek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari class. Bagian tengah mendefinisikan property/atribut class. Bagian akhir mendefinisikan method-method dari sebuah class. |
| 2 |  | *Association* | Sebuah asosiasi merupakan sebuah relationship paling umum antara 2 class dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 class. Garis ini bisa melambangkan tipe-tipe relationship dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah relationship. (Contoh: *One-to-one, one-to-many, many-to-many*). |
| 3 |  | *omposition* | Jika sebuah class tidak bisa berdiri sendiri dan harus merupakan bagian dari class yang lain, maka class tersebut memiliki relasi Composition terhadap class tempat dia bergantung tersebut. Sebuah relationship composition digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid. |
| 4 |  | *Dependency* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 5 |  | *Aggregation* | Aggregation yaitu mengindikasikan keseluruhan bagian relationship dan biasanya disebut sebagai relasi. |
| 6 |  | *State* | Nilai atribut dan nilai link pada  suatu waktu tertentu,  yang dimiliki oleh suatu objek. |

*Sumber : Sukamto dan Shalahuddin (2013:146-147)*

Dan berikut contoh diagram class



\

**17. Diagram Aktivitas *(Activity Diagram)***

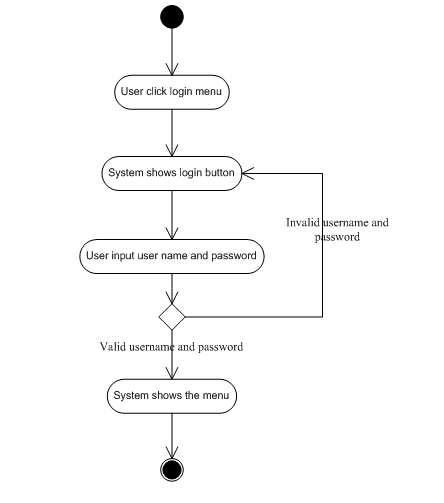
Sukamto dan Shalahuddin (2013:161) “*activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak”.

Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Adapun simbol-simbol yang digunakan dalam *activity* diagram adalah sebagai berikut:

**Tabel 5.** Simbol-simbol *Activity Diagram*

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Simbol** | **Nama** | **Deskripsi** |
| 1 |  | Status awal | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal |
| 2 |  | Aktivitas | Aktivitas yang dilakukan sistem,biasanya diawali dengan kata kerja |
| 3 |  | Decision | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |
| 4 |  | *Join* | Asosiasi penggabungan dimana lebihdari satu aktivitas digabungkan menjadi satu |
| 5 |  | Status akhir | Status akhir yang dilakukan sebuahsistem, sebuah diagram aktivitas memiliki sebuah status akhir |
| 6 |  | *Swimlane* | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi |

*Sumber : Sukamto dan Shalahuddin (2013:162-163)*

**Contoh diagram Activitas**

**18. Sequence Diagram**

Sukamto dan Shalahuddin (2013:165), “diagram sekuen menggambarkan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek”.

*Sequence* diagram menunjukkan urutan *event* kejadian dalam suatu waktu. Komponen *sequence* diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama *message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progress vertikal. Simbol-simbol yang digunakan dalam *sequence* diagram adalah

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **GAMBAR** | **NAMA** | **KETERANGAN** |
| 1 |  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |
| 2 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |
| 3 |  | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi |

**Tabel 6. Simbol *Sequence Diagram***

*Sumber : Sukamto dan Shalahuddin (2013:162-163*

Berikut merupakan contoh sederhana dari Sequence Diagram :

