BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pakar.

Sistem pakar merupakan cabang dari *Artificial Intelligence (AI)* yang cukup tua karena sistem ini mulai di kembangkan pada pertengahan 1960. Sistem Pakar yang muncul pertama kali adalah *General-pupose solver (GPS)* yang akan dikembangkan oleh Newel dan Simon.Sampai saat ini sudah banyak sistem pakar yang dibuat, seperti *MYCIN* untuk diagnosis penyakit.

"Sistem pakar adalah sebuah sisten yang menggunakan pengetahuan manusia dimana pengetahuan tersebut di masukkan kedalam sebuah komputer dan kemudian di gunakan untuk menyelesaikan masalah-masalah yang biasanya membutuhkan kepakaran atau keahlian manusia".

" Sistem pakar adalah program komputer yang merepresentasikan dan melakuakan penalaran dengan pengetahuan beberapa pakar untuk memecahkan masalah memberikan saran".

"Sistem pakar adalah program yang berbasikan pengetahuan yang menyediakan solusi 'kualitas pakar' kepada masalah masalah dalam bidang (domain) yang spesifik. (T.Sutojo,dkk;2011:160)

II.2. Kecerdasan Buatan

Kecerdasan Buatan berasal dari bahasa Inggris *" artificial Intelligence "* atau di singkat *AL,* yaitu adalah kata sifat yang berarti cerdas, sedangkan *artificial*

artinya buatan. Kecerdasan buatan yang di maksud di sini merujuk pada mesin yang mampu berfikir, menimbang tindakan yang akan di ambil, dan mampu mengambil keputusan seperti yang di lakukan oleh manusia.

" kecerdasan buatan (*artificial intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemograman komputer untuk melakukan sesuatu hal yang pandang manusia adalah cerdas".

" Kecerdasan buatan *(AI)* merupakan sebuah *study* tentang bagaimana membuat komputer hal-hal yang saat pada saat ini dapat dilakukan lebih baik oleh manusia.

" Kecerdasan buatan *(AI)* merupakan cabang dari ilmu komputer yang dalam merepresentasikan pengetahuan lebih banyak menggunakan bentuk simbol-simbol dari pada bilangan dan memproses informasi berdasarkan metode heuristis atau dengan berdasarkan sejumlah aturan."

Berdasarkan defenisi ini, maka kecerdasan buatan menawarkan media maupun uji teori tentang kecerdasan. Teori-teori ini nantinya dapat di nyatakan dalam bahasa pemograman dan eksekusinya dapat di buktikan pada komputer nyata. (T.Sutojo,dkk;2011: 3)

II.3. *Certainty factor (*Faktor kepastian*)*

Teori *certainty Factor (CF)* diusulkan oleh shortliffe dan Buchanan pada 1975 untuk mengakomadasi ketidakpastian pemikiran *(inexact reasoning)* seorang pakar. Seorang pakar, (misalnya dokter) sering kali menganalisis informasi yang ada dengan ungkapan seperti "mungkin", "kemungkinan besar", "hampir pasti",

untuk mengakomodasi hal ini kita menggunakan *certainty factor (CF)* guna menggambarkan yang sedang dihadapi.

Ada dua cara dalam mendapatkan tingkat keyakinan *(CF)* dari sebuah *rule*, yaitu : Metode *'NetBelief''* yang di usulkan oleh E. H. Shortliffe dan B. G. Buchanan *CF (Rule)* = MB (H,E) – MD (H,E)…….(4-10)

$$MB(H,E) = \begin{cases} 1 & P(H)=1 \\ \dfrac{Max[P(H|E), P(H)]-P(H)}{Max[1,0]-P(H)} & \text{Lainnya} \end{cases} \quad \text{……..(4-10)}$$

$$MD(H,E) = \begin{cases} \dfrac{Max[P(H|E), P(H)]-P(H)}{Max[1,0]-P(H)} & P(H)=0 \end{cases} \quad \text{……..(4-12)}$$

Di mana :

CF (Rule) = Faktor kepastian

MB (H,E) = *Measure of belief* (ukuran kepercayaan) terhadap hipotesis H, jika diberikan *Evidence* E (anatara 0 dan 1)

MD (H,E) = *Measure of disbelief,* (ukuran ketidakpercayaan) terhadap *evidence* H, jika di berikan *evidence* E (antara 0 dan 1)

P(H) = Probabilitas kebenaran hipotesis H

P(H | E) = Probabilitas bahwa H benar karena fakta E

Seandainya seorang pakar penyakit kelamin menyatakan bahwa probabilitas seseorang berpenyakit phimosis adalah 0.02 dari data lapangan menunjukkan bahwa dari 100 orang penderita penyakit phimosis, 40 orang memiliki gejala kulup berminyak. Dengan menganggap H = Phimosis dan E =

kulup berminyak, hitung *factor* kepastian bahwa phimosis di sebabkan oleh adanya kulup berminyak. Jawab :

P (Phimosis) = 0.02

P (Phimosis ⎪ kulup Berminyak) = 40/100 = 0,4

$$MB(H,E) = \frac{Max[p(H \mid E), p(H)]-p(H)}{Max[1,0]-P(H)}$$

$$= \frac{Max[0,4,0,02]- 0,02,}{Max[1,0]-P(H)} = \frac{0,4 – 0,02}{1- \ 0,002} = \underline{0,39}$$

$$MD(H,E) = \frac{Min[p(H \mid E), p(H)]-p(H)}{Min[1,0]-P(H)}$$

$$= \frac{Max[0,4,0,02]- 0,02,}{Max[1,0]-P(H)} = \frac{0,4 – 0,02,}{2- \ 0,002} = 0$$

CF = 0,39 – 0 = 0,39

Rule : if (Gejala = kulup Berminyak) ***THEN*** Penyakit = Phimosis (*CF*=0,39)

Dengan cara mewawancarai seorang pakar nilai *CF (rule)* di dapat dari interprestasi *"term"* dari pakar, yang di ubah menjadi nilai *CF* tertentu sesuai tabel berikut.

II.1 Tabel Faktor Kepastian

Uncertain Term	CF
Definitely not (pasti tidak)	-1.0
Almost certainly not (hampir pasti tidak)	-0.8
Probably not (kemungkinan besar tidak)	-0.6
Maybe not (mungkin tidak)	-0.4
Unknown (tidak tahu)	-0.2 to 0.2
Maybe (mungkin)	-0.4
Probably (kemungkinan besar)	-0.6
Almost Certaintly (hampir pasti)	-0.8
Definitely (pasti)	1.0

Pakar :

jika batuk dan panas, maka 'hampir dipastikan' (*almost certainly*) penyakitnya adalah *influenza*. (T.Sutojo,dkk;2011:195)

II.4. Visual Basic 2010

Visual Basic.NET adalah *Visual Basic 2010* yang direkayasa kembali untuk digunakan pada *platform.NET* sehingga aplikasi yang di buat menggunakan *Visual Basic.NET* dapat berjalan pada sistem komputer apapun, dan dapat mengambil data dari *server* dengan tipe apapun asalkan terinstal. *NET Framework.* (Priyanto Hidayatullah;2014:5)

II.5. Database SQL Server 2008

SQL Server 2008 adalah sebuah terobosan baru dari *microsoft* dalam bidang *database*. *SQL Server 2008* adalah sebuah *DBMS (Database Management System)* yang dibuat oleh *Microsoft* untuk ikut berkecimpungan dalam persaingan dunia, pengolahan data menyusul pendahuluanya seperti *IBM* dan *Orcale* (Andi Offset;2010:2)

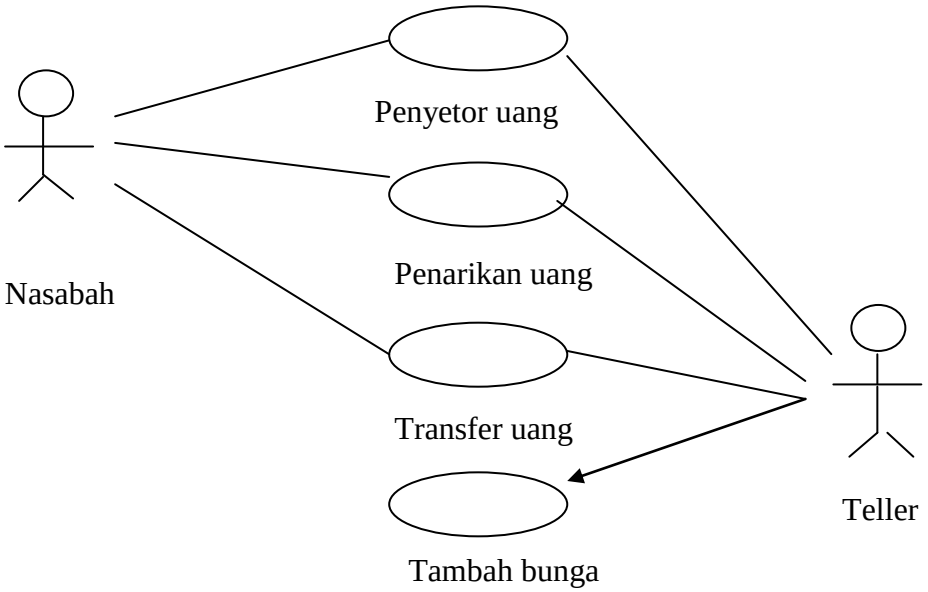
II.6. *Unified Modelling Language*

UML singakatan dari *Unifed Modeling language* yang berarti bahasa pemodelan standar. Mengatakan sebagai bahasa, berarti *UML* memiliki sintaks dan semantik. Ketika kita membuat model menggunakan konsep *UML* ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana sistem mengatasi *error* yang terjadi? Bagaimana kemanan terhadap sistem kita buat? Dan sebagainya dapat di jawab dengan *UML*. (Prabowo Pudjo Widodo,dkk;2011:6)

II.6.1. Komponen-komponen *UML*

a. *Use case* diagram

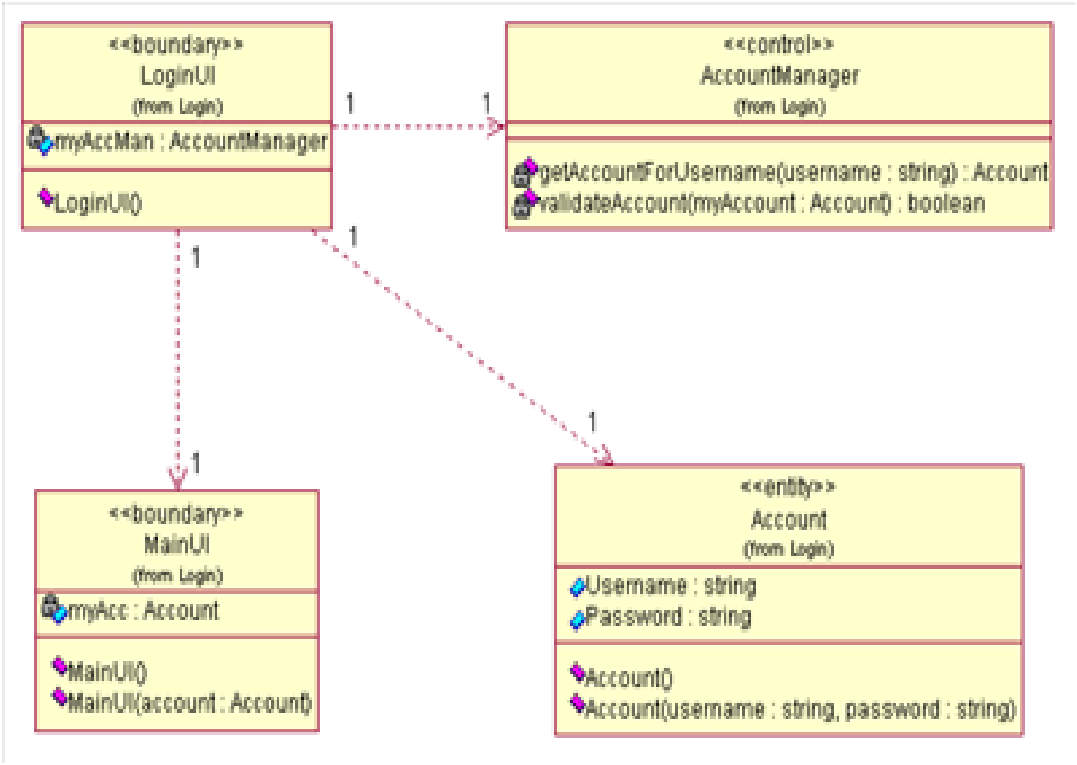
Salah satu kontributor terhadap diagram use case dalam *UML* adalah Ivar Jacobsen. *Use case* menggamabarkan *external view* dari sistem yang akan kita buat modelnya. Gamabar dibawah ini merupakan salah satu contoh bentuk diagram *use case*. (Prabowo Pudjo Widodo,dkk;2011:16)



Gambar II.1. Contoh *Use case* Diagram
(Sumber : Prabowo Pudjo Widodo,dkk ; 2011: 17)

b. *Class* diagram

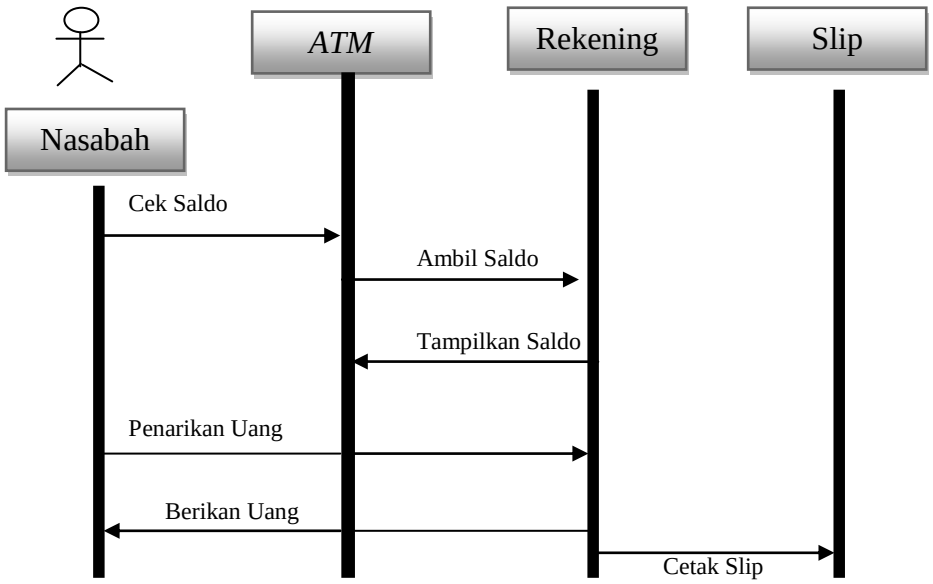
Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat (Jurnal Informatika Mulawarman;Haviluddin;2011:3)



Gambar II.2. Contoh *Class* Diagram
(Sumber : Haviluddin ; 2011: 3)

c. *Sequence Diagram*

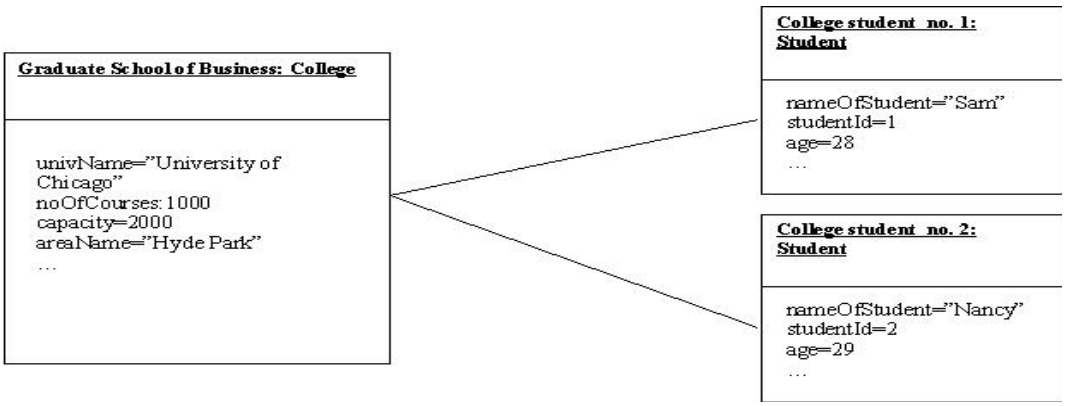
Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence* diagram adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai *use case* diagram. (Jurnal Informatika Mulawarman;Haviluddin;201:5)



Gambar II.3. Contoh *Sequence Diagram*
(Sumber : Adi Nugroho ; 2009 : 102)

d. *Object* diagram

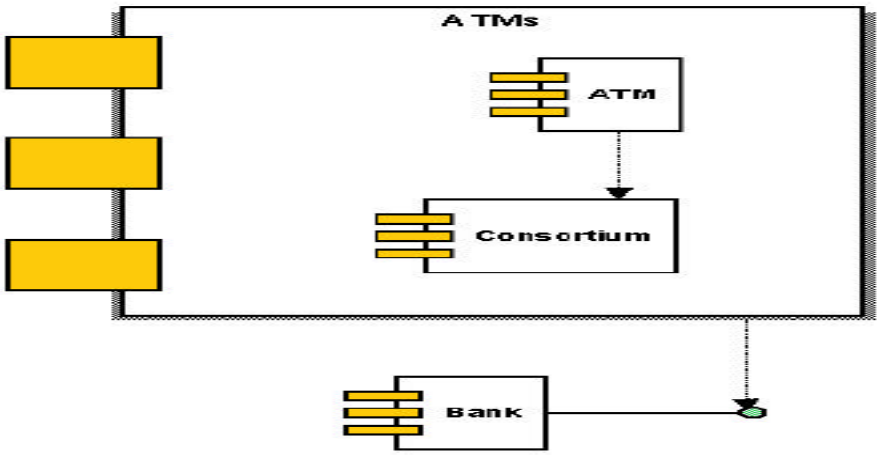
Object diagram menggambarkan kejelasan kelas dan warisan dan kadang-kadang diambil ketika merencanakan kelas, atau untuk membantu pemangku kepentingan non-program yang mungkin menemukan diagram kelas terlalu abstrak. Berikut notasi *object diagram* (Jurnal Informatika Mulawarman;Haviluddin;2011:3)



Gambar II.4. Contoh *Object* Diagram
(Sumber : Haviluddin ; 2011: 3)

e. *Component diagram*

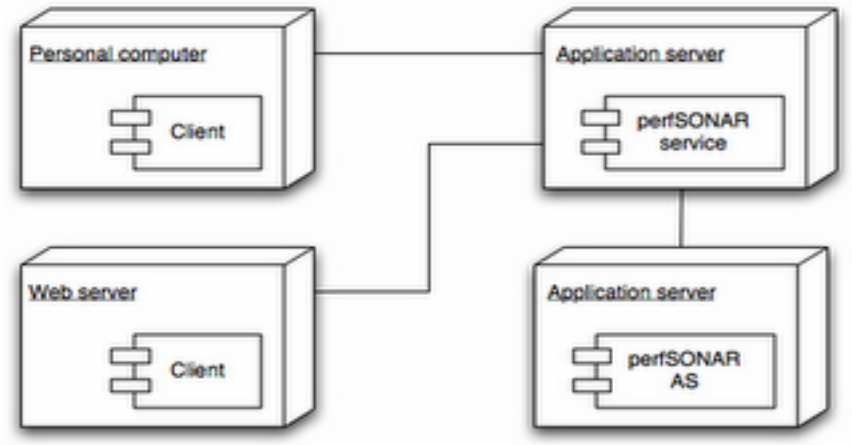
Component diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual di mana logika ini dilaksanakan **(**Jurnal Informatika Mulawarman;Haviluddin;2011:3)



Gambar II.5. Contoh *Component* Diagram
(Sumber : Haviluddin ; 2011 : 3)

f. *Deployment* diagram

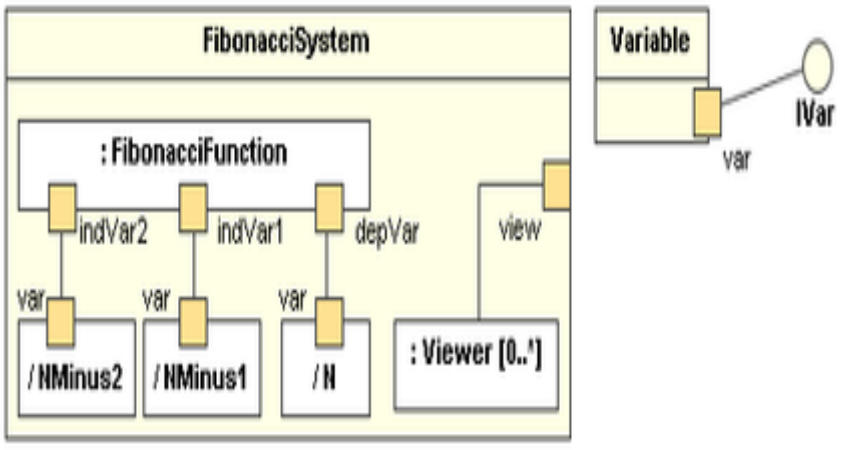
Deployment diagram memberikan gambaran dari arsitektur fisik perangkat lunak, perangkat keras, dan artefak dari sistem. *Deployment* diagram dapat dianggap sebagai ujung spektrum dari kasus penggunaan, menggambarkan bentuk fisik dari sistem yang bertentangan dengan gambar konseptual dari pengguna dan perangkat berinteraksi dengan sistem (Jurnal Informatika Mulawarman;Haviluddin;2011:3)



Gambar II.6. Contoh *Deplomen*t Diagram
(Sumber : Haviluddin ; 2011 : 4)

g. *Composite structure* diagram

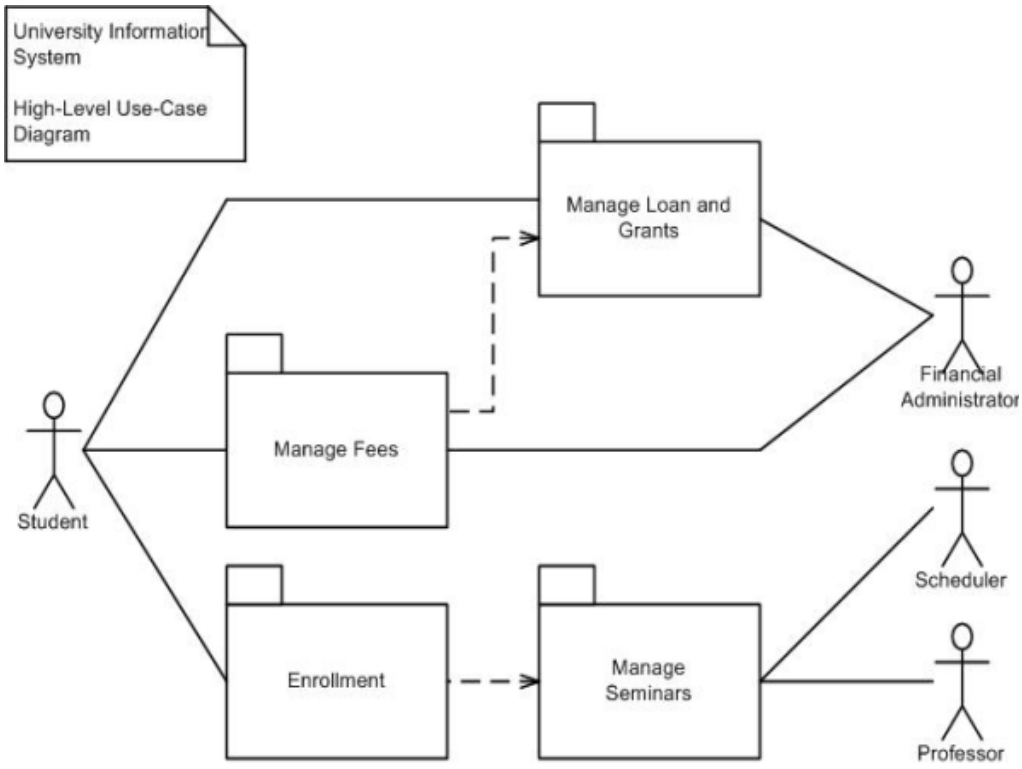
Sebuah diagram struktur komposit mirip dengan diagram kelas, tetapi menggambarkan bagian individu, bukan seluruh kelas. Kita dapat menambahkan konektor untuk menghubungkan dua atau lebih bagian dalam atau ketergantungan hubungan asosiasi. (Jurnal Informatika Mulawarman;Haviluddin;2011:4)



Gambar II.7. Contoh *Composite structure* diagram
(Sumber : Haviluddin ; 2011 : 4)

h. *Package* diagram

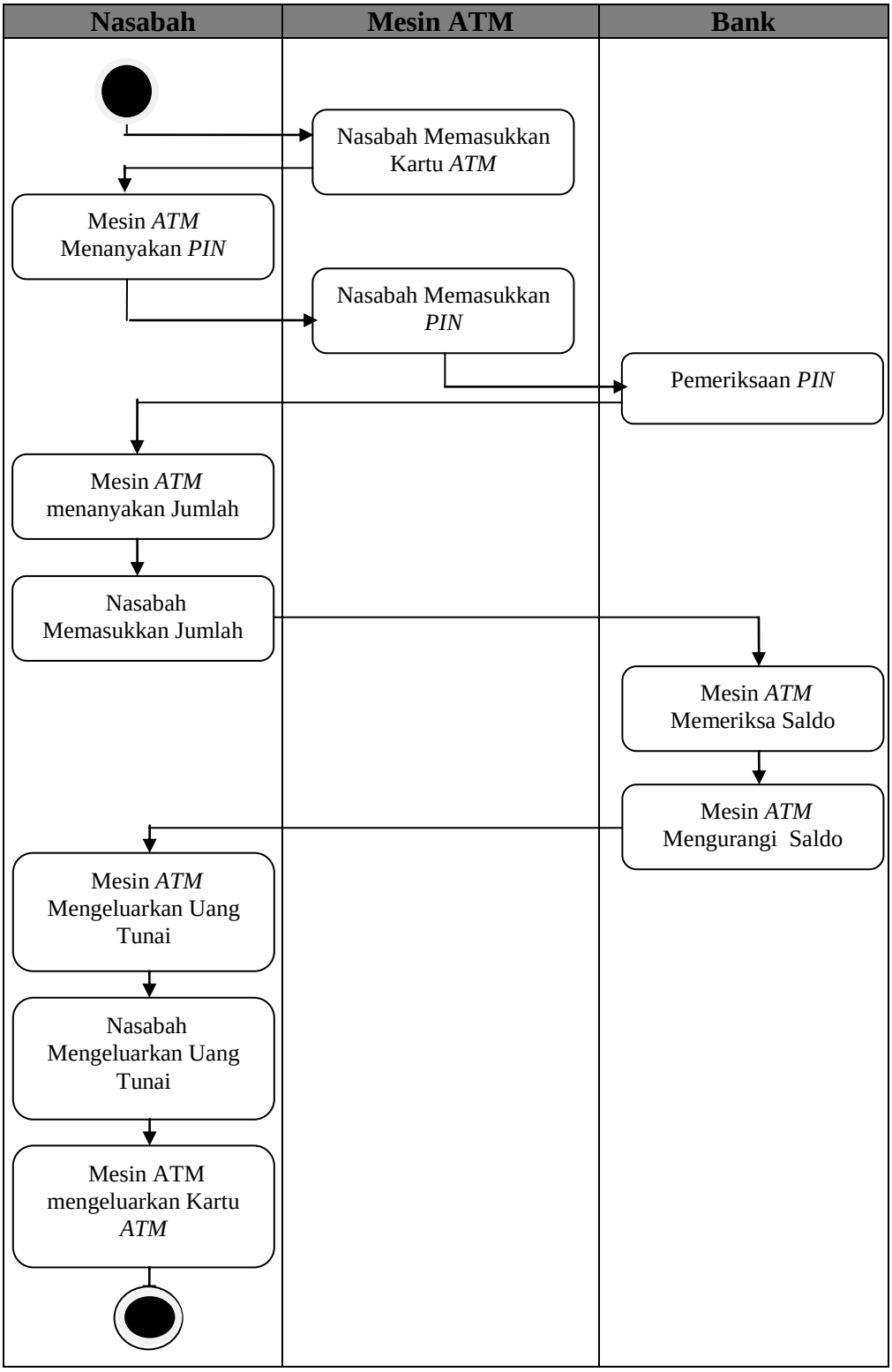
Paket diagram biasanya digunakan untuk menggambarkan tingkat organisasi yang tinggi dari suatu proyek *software*. Atau dengan kata lain untuk menghasilkan diagram ketergantungan paket untuk setiap paket dalam Pohon Model. (Jurnal Informatika Mulawarman;Haviluddin;2011:4)



Gambar II.8. Contoh *package* diagram
(Sumber : Haviluddin ; 2011 : 4)

i. *Activity Diagram*

Menggambarkan aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas (Jurnal Informatika Mulawarman;Haviluddin;201:4)



Gambar II.9. Contoh *Activity* Diagram
(Sumber : Adi Nugroho ; 2009: 11)

II.7. Kamus Data

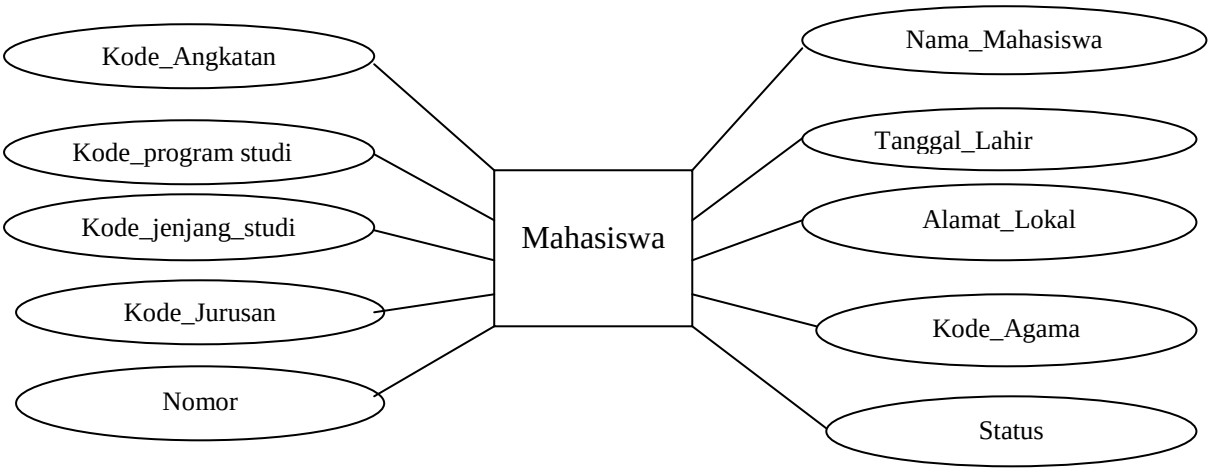
Kamus data tidak menggunakan notasi sebagaimana DFD, kamus data mempunyai fungsi yang sama dengan pemodelan sistem, selain itu kamus data berfungsi membantu pelaku sistem untuk mengerti aplikasi secara detil dan mereorganisasi semua elemen data yang digunakan dalam sistem secara tepat,sehingga pemakai dan penganalisa sistem mempunyai dasar pengertian yang sama tentang masukan, keluaran , penyimpanan dan proses.(Jurnal Sistem Komputer Hermawan,dkk;2011:11)

II.8. *Entity Relationship* Diagram (ERD)

Sebuah diagram *ER/ER_D* tersusun atas tiga komponen, yaitu entitas, atribut dan kerelasian. Secara garis besar, entitas merupakan objek besar dasar yang terlibat dalam sistem. Atribut berperan sebagai penjelas entitas, sedangkan kerelasian menunjukan hubungan yang terjadi di antara dua entitas (Edhy Sutanta ;2011:92)

Objek Dasar	Simbol Entitas
Mahasiswa	Mahasiswa
Dosen	Mahasiswa
Wali Mahasiswa/Orang tua	Wali_Mahasiswa

Gambar II.10. Contoh Entitas Berupa Orang
(Sumber : Edhy Sutanta ; 2011 : 94)

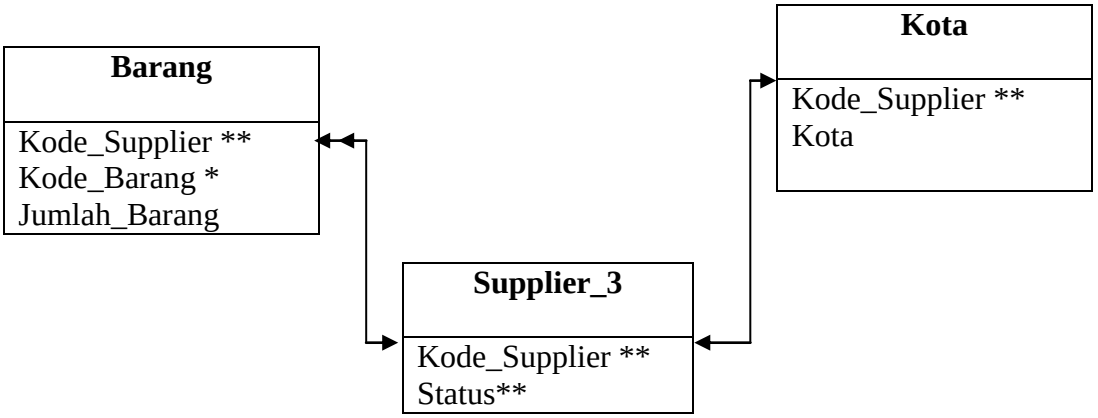


Gambar II.11. Contoh Atrribut pada Entitas Mahasiswa
(Sumber : Edhy Sutanta ; 2011 : 100)

Gambar II.12. Contoh kerelasian antara entitas Mahasiswa dan Mata_Kuliah
(Sumber : Edhy Sutanta ; 2011 : 110)

II.9. Normalisasi

Normalisasi diartikan sebagai suatu teknik yang ,menstrukturkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan (*anomalies)* yang terjadi akibat adanya kerangkapan data dalam relasi dan in-efisiensi pengolahan (Edhy Sutanta;2011:174)



Gamabar II.13. Contoh Kerelasian Antara Relasi hasil Normalisasi
(Sumber : Edhy Sutanta ; 2011: 183)