BAB II

TINJAUAN PUSTAKA

2.1 DASAR TEORI

2.1.1 SISTEM INFORMASI AKADEMIK BERBASIS *WEB*

A. Definisi

Sistem Informasi Akademik Berbasis *Web*, merupakan suatu sistem yang terintegrasi antara manusia (pemakai) dengan *hardware* dan *software* untuk menyediakan informasi bagi dukungan operasi, manajemen dan fungsi-fungsi pengambilan keputusan di dalam lembaga pendidikan. Sistem ini mengintegrasikan proses adminstrasi, informasi, dan prosedur akademik ke dalam suatu sistem yang terkomputerisasi yang dapat dapat diakses dari mana dan kapan saja, secara sangat cepat dan real time.

Sistem ini selain mengkomputerasikan proses administrasi pendidikan, tetapi juga digunakan untuk mengintegrasikan data, mempercepat dan mensistemasikan pengolahan data, meningkatkan kualitas informasi, mendorong terciptanya layanan-layanan baru, meningkatkan kontrol, mengotomasikan sebagian pekerjaan administrasi rutin, menyederhanakan alur registrasi atau proses keuangan lainnya dan mendistribusikan informasi tersebut [OET02].

B. Tujuan

Untuk membangun sebuah SIMAK berbasis *web*, para pengelola lembaga pendidikan harus memiliki tujuan yang jelas. Tanpa tujuan yang jelas maka mereka akan kesulitan dalam mengelola, bertahan dan mengembangkan sistem terbut.

Membangun SI bukan hanya mengkomputerisasi prosedur tradisional di dalam lembaga yang kemudian dihubungkan ke jaringan komputer dan Internet. Jika pemahaman ini yang menjadi landasan dalam membangun SI, niscaya sistem itu sulit untuk bertahan. Tujuan pembangunan SI semestinya lebih dari itu, yaitu [OET02] :

1. Adanya keinginan yang kuat dan konsisten untuk membangun aliran data dan informasi yang lancar, sistematis, sederhana dan akurat, yang memberikan respon dengan cepat, mengolah dan menghasilkan keluaran yang dapat digunakan sebagai dasar pengambilan keputusan dan kotrol manajemen pendidikan dengan baik.
2. Membangun SIMAK berbasis *web* berbasis jaringan komputer dan Internet yang mampu mengintegrasikan data dan mendistribusikan informasi dari dan ke berbagai terminal dengan cepat, aman dan akurat.
3. Menciptakan sistem yang memudahkan dan menyederhanakan proses untuk mengup- load dan mendownload materi pendidikan, sehingga setiap pendidik dapat setiap saat memperbaiki modul-modulnya, sementara siswa dapat memperoleh modul itu melalui proses administrasi yang sederhana.

C. Manfaat

Ada banyak manfaat yang dapat dipetik baik oleh lembaga pendidikan, siswa dan masyarakat pada umumnya dengan adanya SIMAK berbasis *web*. Beberapa manfaat tersebut diantaranya [OET02] :

1. Bagi Lembaga Pendidikan, yaitu memperpendek jarak; perluasan pasar; perluasan mitra jaringan kerja; biaya terkendali; hemat; *cash flow* terjamin dan manfaat lainnya.
2. Bagi Siswa, yaitu hemat; biaya terkendali dan fleksibel.
3. Bagi Masyarakat pada umumnya, yaitu dengan adanya SI *e-Education* membuka peluang kerja baru; wahana kompetisi antarlembaga pendidikan untuk memberi materi pendidikan yang kompetitif.
4. Bagi Dunia Akademis, yaitu memberi tantangan bagi dunia pendidikan akademis untuk mempersiapkan SDM yang memahami dan menguasai teknologi; tantangan untuk melakukan analisis terhadap pergeseran pola belajar, proses pendidikan dan pembayaran sistem kredit semester dalam usaha menemukan kesepahaman baru dan pengembangan teori dan konsep baru.

D. Jenis Pelayanan

SIMAK berbasis *web* mengintegrasikan jenis-jenis pelayanan adminstrasi, informasi, dan prosedur akademik ke dalam suatu sistem informasi manajemen akademik yang dapat dapat diakses dari mana dan kapan saja, secara sangat cepat dan real time. Pelayanan yang dapat diperoleh diantaranya :

1. Digitalisasi administrasi, informasi dan prosedur akademik seperti : pendaftaran online, registrasi online, KRS online, dan lain-lain.

2. *Electronic Book (e-book)*
3. *Electronic News (e-news)*
4. *Electronic Library (e-library)*
5. Pembayaran online dan lain-lain

E. Implementasi

Implementasi dari SIMAK berbasis *web* dapat dilihat dari beberapa institusi lembaga pendidikan. Dalam skalanya, lembaga-lembaga tersebut dapat dikelompokkan dalam beberapa kategori sebagai berikut [OET02] : Lembaga Produsen Produk-Produk Teknologi, seperti Cisco, Oracle, Microsoft, Autodesk; Lembaga Pendidikan Menengah dan Tinggi, seperti SLTA dan PT; Lembaga Pendidikan Non-Kurikuler, seperti Inixindo dan lain sebagainya.

Diantara beberapa lembaga pendidikan yang telah mengaplikasikan SIMAK berbasis *web* antara lain:

Tabel 2.1 Contoh aplikasi SIMAK berbasis *web*

No.	Alamat Situs	Keterangan
1	www.uii.ac.id	Situs SI *e-Education* UII Jogjakarta
2	www.binus.ac.id	Situs SI *e-Education* Bina Nusantara Jakarta
3	www.ugm.ac.id	Situs SI *e-Education* UGM Jogjakarta
4	www.itb.ac.id	Situs SI *e-Education* ITB Bandung
5	Dan lain-lain	

2.1.2 *UNIFIED MODELLING LANGUAGE* (UML)

Suatu bahasa menyediakan seperangkat kosakata dan aturan-aturan dalam mengombinasikan kata-kata sehingga dapat digunakan dalam berkomunikasi. Sebuah bahasa pemodelan merupakan sebuah bahasa yang kosakata dan aturan-aturannya difokuskan pada representasi konseptual dan fisik dari sebuah sistem. UML, sebagai bahasa pemodelan merupakan sebuah standard untuk *blueprint* perangkat lunak. Booch dkk. menyatakan bahwa UML merupakan bahasa untuk memvisualisasikan, menspesifikasikan, membangun serta mendokumentasikan. UML dapat digunakan dalam berbagai sistem, seperti : Sistem informasi *enterprise*, Layanan perbankan dan keuangan, Telekomunikasi, Transportasi, Kedirgantaraan, *Retail*, Elektronika medis, Layanan berbasis *web* terdistribusi.

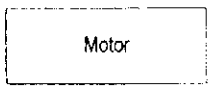
UML tidak terbatas untuk memodelkan perangkat lunak, tetapi juga sistem-sistem yang bukan perangkat lunak. UML terdiri atas seperangkat presentasi grafis dan notasi, yang terdiri dari : Diagram *class*, Diagram *use case*, Diagram interaksi, Diagram rangkaian, Diagram kolaborasi, Diagram keadaan, Diagram aktivitas, Diagram komponen, serta Diagram *deployment*.

A. Diagram *Class*

Class merupakan bangunan utama dari suatu sistem yang berorientasi objek, yang mendeskripsikan sejumlah objek dengan atribut, operasi, relasi, dan semantik.

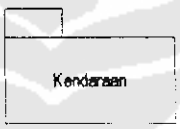
Sebuah *class* diberi notasi berupa persegi panjang. Penamaan *class* diawali dengan huruf besar dan ditulis didalam persegi panjang. Penamaannya dengan huruf kapital di awal, yang diletakkan di atas persegi

tersebut. Jika nama *class* terdiri dari 2 kata, maka kedua kata tersebut digabungkan dan huruf awal setiap kata dikapitalkan. Lihat gambar 2.2 menunjukkan notasi dari *class*.



Gambar 2.1 Notasi *Class*

Konstruksi lainnya adalah paket *(package)*, yang merupakan elemen UML untuk mengelompokan elemen-elemen diagram. Dinotasikan sebagai suatu *tabbed folder*, dengan sebuah nama. Aturan penulisan nama paket, jika ada 2 kata, dipisahkan spasi. Gambar 2.3 menunjukkan notasi dari *package*.



Gambar 2.2 Notasi *Package*

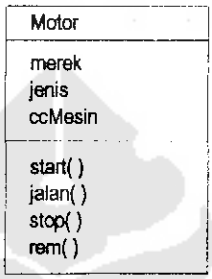
1. **Atribut dan Operasi**

Atribut merupakan properti dari suatu klas. Suatu klas dapat memiliki 0 atau lebih atribut (artinya : dapat tidak memiliki atribut). Penulisan atribut diawali dengan huruf kecil. Jika terdiri dari 2 kata, maka kata pertama diawali huruf kecil, sedangkan kata kedua dengan huruf kapital, dan keduanya digabungkan. Atribut-atribut klas dituliskan di bawah nama klas, dengan dibatasi suatu garis pembatas.

Pada objek, yang merupakan turunan dari klas, memiliki nilai tertentu pada setiap atributnya. Notasi objek serupa dengan notasi klas, hanya saja penulisan nama objek dimulai dengan huruf kecil dengan

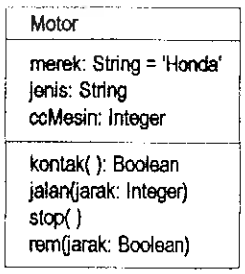
pembatas tanda titik dua antara nama objek dan klasnya. Nama objek juga harus diberi garis bawah.

Operasi merupakan sesuatu yang dilakukan oleh sebuah klas, atau yang dilakukan terhadap sebuah klas. Aturan penamaannya sama dengan atribut. Operasi dinotasikan di bawah atribut, dengan dibatasi garis pemisah. Lihat gambar 2.3.



Gambar 2.3 Penulisan atribut dan operasi

Penulisan atribut dapat juga dilengkapi dengan tipe data. Tanda ':' memisahkan antara atribut dengan tipe data. Nilai bawaan (default) dapat juga ditambahkan pada atribut dengan menggunakan tanda '='. Penulisan operasi pun dapat dilengkapi dengan parameter. Lihat gambar 2.4.

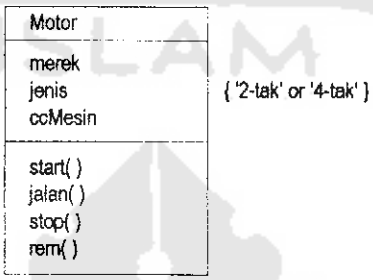


Gambar 2.4 Penulisan atribut dan operasi yang dilengkapi dengan tipe data

Untuk menyederhanakan kompleksitas diagram *class*, bagian atribut dan operasi dapat dikosongkan.

2. **Batasan dan Catatan**

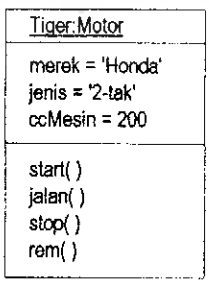
Sebuah *class* dapat memiliki batasan (*constraint*) tertentu. Batasan tersebut ditulis dalam tanda kurung kurawal "{ ... }". Sebagai contoh, atribut jenis pada *class* Motor hanya diperbolehkan '2-tak' atau '4-tak'. Penulisan batasan dapat dilihat pada gambar 2.5.



Gambar 2.5 Penulisan batasan

3. **Objek**

Perlu dibedakan antara diagram *class* dan objek. Objek adalah turunan dari *class*. Atribut pada objek memiliki nilai tertentu. Penamaan objek diawali dengan huruf kecil untuk kata pertama, disambung dengan tanda titik dua yang diikuti dengan nama *class*. Seluruh nama objek digaris bawahi. Lihat gambar 2.6.



Gambar 2.6 Objek Tiger yang diturunkan dari *class* Motor

Ketika mendefinisikan atribut dan operasi, maka konteks pembicaraan adalah *class*. *Class* bertindak sebagai template (kerangka)

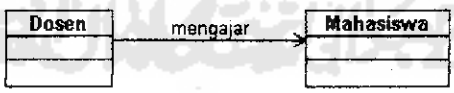
untuk mendefinisikan atribut dan operasi. Banyak objek dapat diturunkan dari *class* yang sama. Objek yang diturunkan dari sebuah *class* disebut *instance*. Objek-lah yang akan berinteraksi dengan objek lainnya.

B. Relasi Antar *Class*

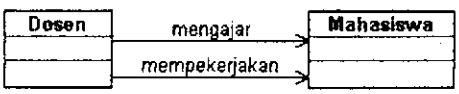
Dalam UML, cara untuk menghubungkan antar suatu komponen secara fisik maupun logik, dimodelkan dengan sebuah relasi *(relationship)*. Dalam pemodelan berorientasi objek ada beberapa macam relasi penting.

1. Asosiasi

Asosiasi antar dua *class* digambarkan dengan garis yang menghubungkan keduanya, disertai dengan nama asosiasi di atas garis tersebut. Gambar 2.7 menunjukkan contoh asosiasi: dosen mengajar mahasiswa. Gambar 2.8 menunjukkan bahwa dua *class* dapat dihubungkan dengan lebih dari satu asosiasi.

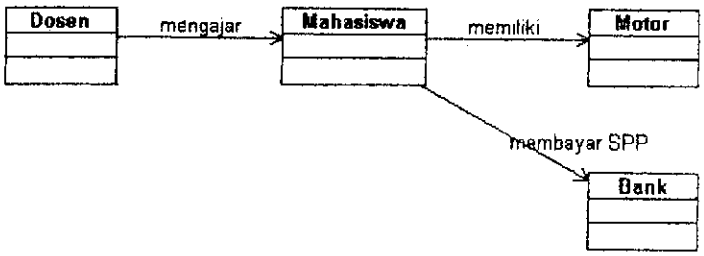


Gambar 2.7 Contoh asosiasi



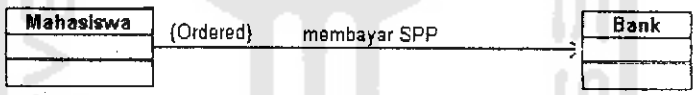
Gambar 2.8 Asosiasi ganda pada dua *class*

Asosiasi dapat melibatkan lebih dari satu *class*, seperti tampak pada gambar 2.9.



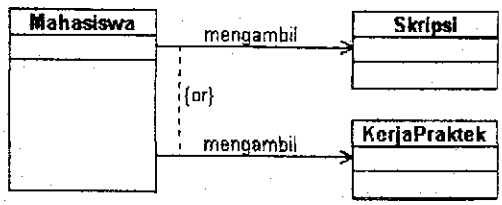
Gambar 2.9 Asosiasi yang melibatkan lebih dari satu *class*

Asosiasi dapat menghubungkan *class* yang memiliki aturan tertentu. Aturan pada *class* diterapkan dalam bentuk batasan (*constraint*). Batasan digambarkan dengan kata yang diletakkan dalam tanda kurung kurawal. Sebagai contoh: mahasiswa antri sesuai urutan membayar SPP di bank. Antri sesuai urutan dapat ditulis dengan tanda {*ordered*}. Lihat gambar 2.10.



Gambar 2.10 Batasan {*ordered*} pada *class* Mahasiswa

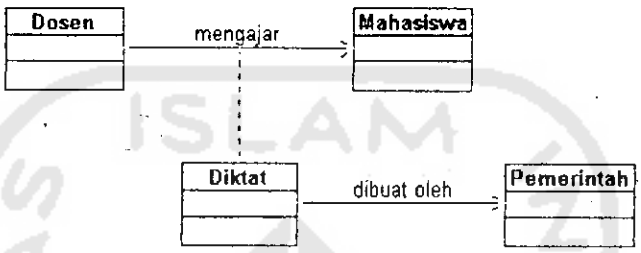
Dua asosiasi dapat memiliki batasan {or} yang dituliskan di atas garis putus-putus yang menghubungkan kedua asosiasi. Sebagai contoh: mahasiswa dapat mengambil kerja praktek atau skripsi. Lihat gambar 2.11.



Gambar 2.11 Batasan {or} pada dua asosiasi

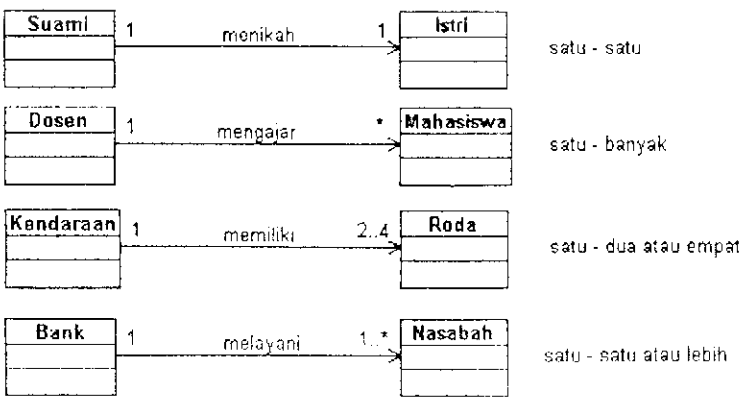
Sebuah asosiasi dapat memiliki atribut dan operasi, disebut *class* asosiasi. *Class* asosiasi terhubung dengan garis asosiasi

menggunakan garis putus-putus. *Class* asosiasi dapat pula terhubung dengan *class* lainnya dengan cara seperti biasa. Sebagai contoh: dosen mengajar mahasiswa dengan diktat yang dibuat oleh pemerintah. Diktat adalah *class* asosiasi. Lihat gambar 2.12.



Gambar 2.12 *Class* asosiasi

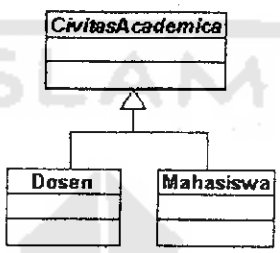
Dalam relasi antar dua *class*, diperlukan keterangan yang menunjukkan seberapa banyak objek yang terhubung dengan objek lainnya. Keterangan itu disebut multiplikasi. Mirip dengan relasi himpunan, multiplikasi dapat berupa relasi satu-satu, satu-banyak, atau kombinasi lainnya. Simbol bintang berarti 'banyak'. Simbol titik ganda (..) dapat berarti 'atau', 'hingga'. Simbol 1..* dibaca 'satu atau banyak'. Simbol 12..18 dibaca '12 hingga 18'. Berbagai kemungkinan bentuk multiplikasi dapat dilihat pada gambar 2.13.



Gambar 2.13 Berbagai bentuk multiplikasi

2. **Generalisasi**

Konsep turunan digambarkan dengan generalisasi. Generalisasi menghubungkan *subclass* dengan *superclass*-nya menggunakan garis panah segitiga terbuka. Lihat gambar 2.14.

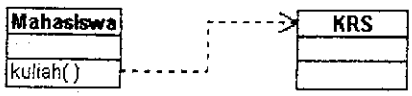


Gambar 2.14 Contoh generalisasi

Pada contoh diatas, objek yang nantinya akan berinteraksi berasal dari *class* Dosen dan Mahasiswa. *Class* CivitasAcademica tidak akan memiliki objek. *Class* yang tidak memiliki objek disebut *class* abstrak. *Class* abstrak ditulis dengan huruf miring.

3. **Dependensi**

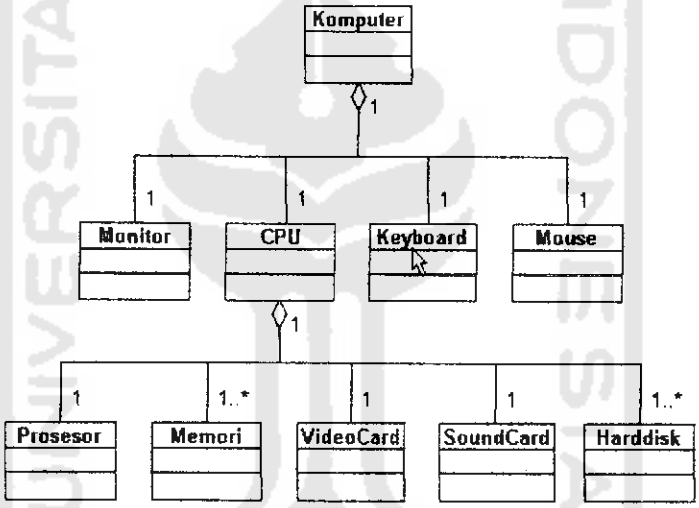
Bentuk lain dari relasi adalah dependensi, yaitu perilaku sebuah *class* tergantung dari *class* lainnya. Dependensi digambarkan dengan panah bergaris putus-putus yang menunjuk pada *class* lain. Sebagai contoh, kuliah yang diambil seorang mahasiswa tergantung dari mata kuliah yang diambil. Lihat gambar 2.15.



Gambar 2.15 Contoh dependensi

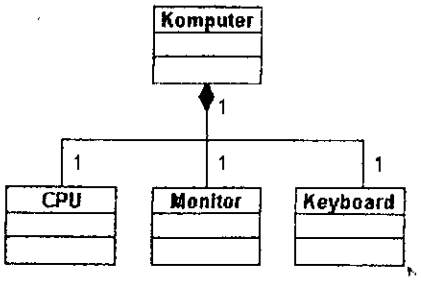
4. **Agregat**

Agregat adalah sebuah *class* yang tersusun atas lebih dari satu *class*. Agregat digambarkan dengan menggunakan garis panah intan. Contoh sederhana dari sebuah agregat adalah komputer. Komputer terdiri dari CPU, monitor, keyboard dan mouse. CPU pun terdiri dari harddisk, drive, CDROM, RAM, videocard dan soundcard. Lihat gambar 2.16.



Gambar 2.16 Contoh agregat

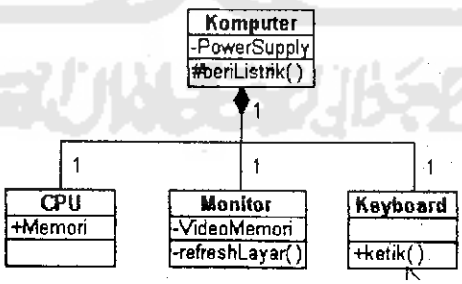
Bentuk lain dari agregat adalah komposit. Salah satu contoh komposit adalah komputer. Sebuah komputer tidak akan dapat berfungsi tanpa adanya CPU, Monitor dan Keyboard. Komposit digambarkan dengan garis panah dengan intan yang berwarna hitam. Lihat gambar 2.17.



Gambar 2.17 Contoh komposit

5. **Visibilitas**

Tiap elemen pada *class* (atribut dan operasi) memiliki visibilitas yang berbeda-beda. Visibilitas menentukan seberapa luas sebuah atribut atau operasi dapat diakses dari *class* yang berbeda. Ada tiga tingkatan visibilitas, yaitu: *public*, *protected*, dan *private*. Visibilitas *public* memungkinkan elemen pada *class* diakses oleh *class* lainnya. Pada Visibilitas *protected,* akses hanya diberikan pada *subclass*-nya. Visibilitas *private* tidak memperbolehkan akses selain *class* itu sendiri. Lihat gambar 2.18.



Gambar 2.18 Contoh visibilitas

Pemodelan entitas sistem yang diwujudkan dalam diagram *class* mendapat porsi terbesar dalam UML. Diagram *class* beserta berbagai macam relasi yang dimilikinya diperlukan untuk membuat model dari permasalahan dan sistem yang akan dikembangkan. Semakin rinci model

yang dibuat, maka pemahaman terhadap sebuah masalah meningkat, dan pengembangan sistem menjadi terarah.

C. Diagram *Use Case*

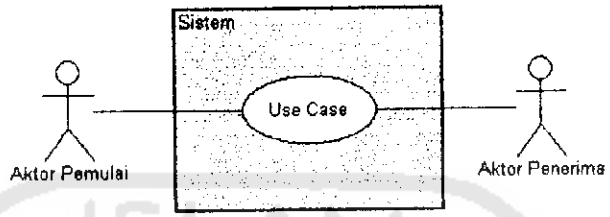
Diagram *use case* menggambarkan bagaimana perilaku sistem menurut sudut pandang pengguna. Sistem akan dibuat dan diuji sesuai dengan diagram *use case*. Dengan demikian, pengguna akan mendapatkan sistem sesuai dengan keinginannya.

Sebuah *use case* dapat dianggap sebagai sebuah skenario penggunaan sistem. Setiap *use case* dideskripsikan dengan menjabarkan urutan langkah-langkah (tahapan). Entitas yang memulai *use case* disebut aktor. Setiap *use case* haruslah menghasilkan sesuatu bagi aktor yang memulai, atau aktor lainnya.

Aktor menggambarkan peran yang dilakukan oleh pengguna, bukan pengguna itu sendiri. Dalam kaitannya dengan penggunaan sistem, seorang pengguna dapat memiliki lebih dari satu peranan. Pengguna yang sama dapat saja berperan sebagai akuntan, operator atau manajer.

Use case digambarkan sebagai sebuah elips yang terhubung dengan gambar orang. Gambar orang di sebelah kiri menunjukkan aktor yang memulai skenario, dan gambar orang di sebelah kanan menunjukkan aktor yang diuntungkan (memperoleh hasil) dari *use case* tersebut. Nama aktor ditulis di bagian bawah gambar orang, dan nama *use case* ditulis didalam

elips. Gambar kotak persegi menunjukkan batasan dari sistem. Lihat gambar 2.19.



Gambar 2.19 Diagram *use case*

Setiap *use case* dilengkapi dengan deskripsi langkah yang berupa teks. Deskripsi tersebut memuat :

- Aktor yang memulai *use case*
- Keadaan awal sistem
- Langkah yang dilakukan aktor
- Keadaan akhir sistem
- Aktor yang diuntungkan

Diagram *use case* menggambarkan bagaimana sistem digunakan oleh pengguna. Pengguna harus sejak awal dilibatkan dalam proses pengembangan sistem, karena nantinya merekalah pemakai sistem tersebut. Burch dan Grudnitski menyatakan bahwa sebuah sistem haruslah berorientasi pada pengguna. Sistem haruslah memenuhi kebutuhan pengguna, bukan pengguna yang memenuhi kebutuhan sistem.

Diagram *use case* nantinya (pada akhir tahap pengembangan sistem) akan digunakan sebagai *test case*, yaitu skenario pengujian sistem. Sistem

akan diuji untuk kesesuaiannya dengan kebutuhan pengguna. Hasil *test case* yang baik akan meminimalkan efek penolakan sistem baru oleh pengguna yang biasanya terjadi dalam sebuah organisasi.

D. Diagram Interaksi

Diagram interaksi merupakan istilah yang memadukan antara diagram rangkaian *(sequence diagram)* dan diagram kolaborasi *(collaboration diagram)*. Diagram interaksi menampilkan suatu interaksi, yang terdiri dari sejumlah objek dan relasinya, termasuk pesan *(message)* yang dikirim.

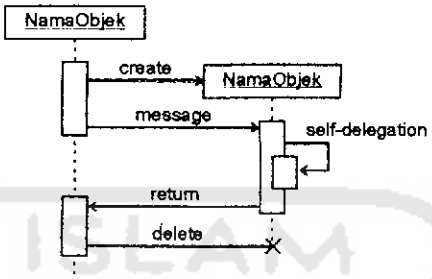
E. Diagram Rangkaian

Diagram rangkaian merupakan diagram interaksi yang menekankan pada urutan waktu penyampaian pesan. Diagram rangkaian merepresentasikan sudut pandang dinamis dari sebuah sistem.

Tiga komponen penyusun diagram ini adalah : objek, pesan, dan waktu. Objek dinotasikan dalam UML dengan sebuah persegi dengan nama bergaris bawah di dalamnya, pesan dinotasikan dengan garis beranak panah, dan waktu direpresentasikan dengan progres secara vertikal.

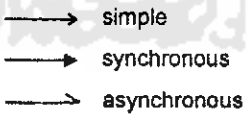
Objek ditempatkan dari kiri ke kanan. Di sebelah bawah objek terdapat garis putus-putus yang disebut *lifeline*. Dan sepanjang garis putus-putus tersebut terdapat persegi yang disebut aktivasi, untuk

merepresentasikan eksekusi operasi oleh suatu objek. Panjangnya bergantung pada durasi suatu aktivasi.



Gambar 2.20 Diagram rangkaian

Pesan yang disampaikan dari suatu objek ke objek lainnya, digambarkan dengan garis yang menghubungkan *lifeline* suatu objek ke *lifeline* lainnya. Ada 3 macam pesan, yaitu sederhana *(simple)*, *synchronous,* dan *asynchronous*. Pesan sederhana adalah suatu transfer kendali dari objek satu ke lainnya. Objek yang mengirimkan pesan synchronous akan menunggu jawaban dari pesan itu sebelum melakukan proses berikutnya. Jika yang dikirim pesan *asynchronous* maka objek tidak perlu menunggu jawaban tersebut. Berikut adalah gambar notasi ketiganya.

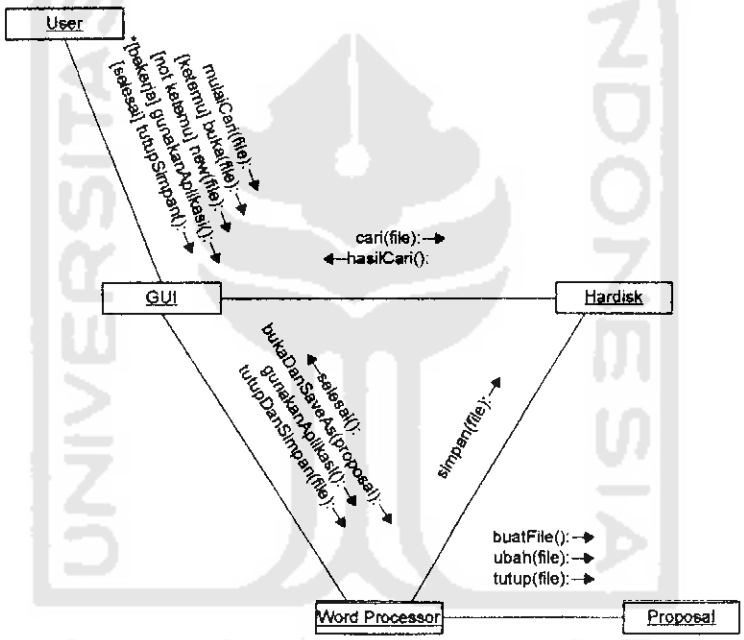


Gambar 2.21 Jenis-jenis pesan

Waktu direpresentasikan dalam diagram dengan arah vertikal, yang dimulai dari atas ke bawah. Pesan yang terjadi di bagian atas akan terjadi lebih dulu dibandingkan yang bagian bawah.

F. Diagram Kolaborasi

Diagram objek menggambarkan objek dan relasi diantaranya. Diagram kolaborasi merupakan pengembangan dari diagram objek, dengan melibatkan pesan yang terjadi pada relasi antar objek. Pesan dinotasikan dengan garis antara dua objek. Anak panah mengarah pada objek penerima pesan.



Gambar 2.22 Diagram kolaborasi

G. Diagram Keadaan

Diagram keadaan menampilkan mesin keaadaan *(state machine)*, yaitu terdiri dari keadaan *(state)*, transisi *(transition)*, even dan aktivitas. Diagram keadaan merupakan diagram yang menampilkan pandangan dinamis dari suatu sistem, yaitu perubahan-perubahan yang terjadi pada objek.

Keadaan dinotasikan dengan persegi dengan ujung bulat, sedangkan garis berpanah merepresentasikan transisi. Keadaan awal sering disebut juga titik awal *(starting point)*, dinotasikan dengan lingkaran utuh dan keadaan akhir disebut titik akhir *(end point)* dinotasikan dengan lingkaran kosong dengan titik di dalamnya.

Gambar 2.23 Diagram keadaan

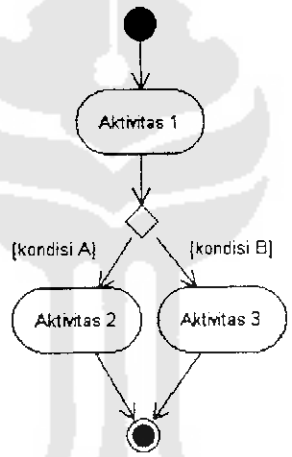
Seperti sebuah klas, notasi keadaan dapat dibagi menjadi 3 bagian, yaitu nama, variabel keadaan, dan aktivitas. Variabel keadaan seperti timer dan counter seringkali berguna. Aktivitas terdiri dari even dan aksi. Tiga aktivitas yang paling sering digunakan adalah *entry* (apa yang terjadi saat sistem memasuki suatu keadaan), *exit* (apa yang terjadi saat sistem meninggalkan suatu keadaan), dan apa yang terjadi saat sistem berada dalam suatu keadaan.

H. Diagram Aktivitas

Diagram aktivitas merupakan bentuk khusus dari diagram keadaan yang menampilkan aliran dari suatu aktivitas ke aktivitas lainnya dalam sebuah sistem. Diagram aktivitas juga merepresentasikan sudut pandang dinamis dari sistem. Jadi pada diagram ini menitikberatkan pada aktivitasnya, sedangkan pada diagram keadaan pada keadaannya. Secara umum diagram ini mirip dengan sebuah diagram alir *(flowchart)*.

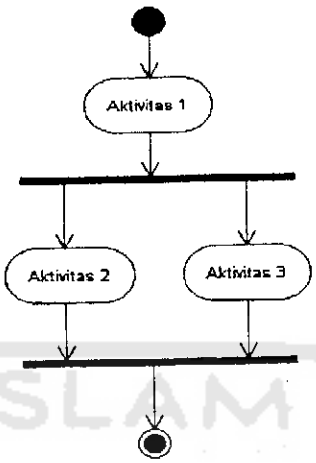
Sebuah aktivitas dinotasikan dengan persegi dengan sudut bulat, dan lebih mengarah ke bentuk oval. Seperti halnya diagram keadaan, diagram aktivitas juga mempunyai notasi transisi, titik awal dan titik akhir.

Kita juga dapat merepresentasikan sebuah percabangan pada diagram ini dengan notasi berupa intan. Namun deskripsi kondisi yang menyertai keputusan diletakkan diluar simbol intan. Deskripsi kondisi ditulis dalam tanda kurung siku. Lihat gambar 2.24.



Gambar 2.24 Keputusan dalam diagram aktivitas

Selain itu, diagram aktivitas dapat menggambarkan konkurensi, yaitu satu atau lebih aktivitas yang berjalan secara bersamaan. Konkurensi diawali dengan sebuah garis tebal horisontal yang menjadi tempat keluarnya garis aktivitas. Konkurensi juga diakhiri dengan garis tebal horisontal.



Gambar 2.25 Konkurensi dalam diagram aktivitas

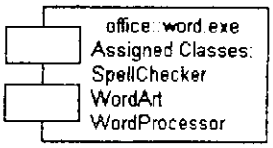
I. Diagram Komponen

Diagram komponen menggambarkan sistem secara fisik. Komponen sistem dapat berupa file *executable*, file help, file data, DLL, dokumen, dan apa pun yang ada dalam sebuah komputer.

Diagram komponen digambarkan dengan persegi panjang yang memiliki dua persegi panjang kecil di sebelah kirinya. Nama komponen diletakkan kotak persegi panjang tersebut. *Interface* digambarkan dengan lingkaran yang terhubung pada komponen. Lihat gambar 2.26.

Gambar 2.26 Diagram komponen

Diagram komponen dapat dilengkapi dengan informasi *class* penyusunnya. Jika komponen juga merupakan anggota sebuah *package*, maka nama *package* dapat ditambahkan sebelum nama komponen. Lihat gambar 2.27.



Gambar 2.27 Informasi tambahan pada diagram komponen

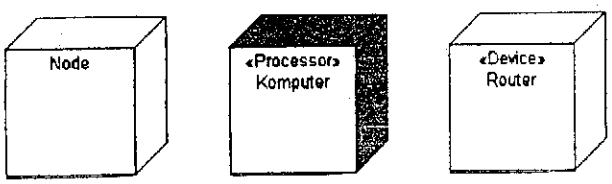
Diagram komponen memberikan wujud nyata dari komponen sebuah sistem. Diagram-diagram konseptual, seperti diagram *class* dan aktivitas, menjadi pembentuk diagram komponen. Pengguna akan lebih mudah melihat sistem yang dikembangkan melalui diagram komponen.

J. Diagram *Deployment*

Diagram *deployment* menggambarkan bagaimana konfigurasi perangkat keras dimana sistem akan diinstalasikan. Diagram ini memberi simbol untuk segala bentuk perangkat keras, seperti: komputer, printer, router, hub, jaringan, dan sebagainya.

Simbol untuk sebuah perangkat keras adalah *node*. *Node* terbagi atas dua jenis. *Processor* adalah *node* yang dapat menjalankan komponen. Contoh dari *node processor* adalah komputer. *Device* adalah *node* yang tidak bisa menjalankan komponen. Contoh *node device* adalah printer, modem, hub, router.

Node digambarkan dengan kubus. Nama *node* diletakkan didalam kubus. Tiap jenis *node* dibedakan hanya dari stereotipnya. *Stereotip* «*Processor*» dan «*Device*» masing-masing digunakan untuk *node Processor* dan *Device*. Lihat gambar 2.28.



Gambar 2.28 Diagram *deployment*

Diagram *deployment* menunjukkan tata-letak perangkat keras secara fisik, dan komponen yang terdapat didalamnya. Diagram ini menunjukkan secara lengkap arsitektur dari sistem dan perangkat kerasnya. Diagram ini akan memperjelas pengguna bagaimana hasil akhir dari sistem yang dikembangkan, dan mempermudah teknisi dalam pemeliharaan sistem.

2.2 STUDI LITERATUR

Penelitian tentang sistem informasi manajemen sudah banyak dilakukan oleh para peneliti. Dari beberapa penelitian tersebut, penulis mengambil beberapa contoh penelitian untuk dijadikan pembanding pada penelitian ini.

Dalam skripsi yang berjudul "Perancangan Sistem Informasi Akademik Berbasis Komputer di Lembaga Pendidikan Bisnis Salemba Yogyakarta", disini dibahas tentang analisa dan perancangan sitem informasi akademik dengan menggunakan metode SDLC (*system development life cycle*). Metode ini terdiri dari beberapa tahap seperti Investigasi sistem, Analisis sistem, Perancangan sistem, Penerapan sistem dan Pemeliharaan sistem. Metode SDLC menggunakan alat bantu beberapa diagram diantranya *Data Flow Diagram, Entity Relationship Diagram* dan Diagram struktur Data. Dengan alat bantu tersebut dilakukan perancangan sistem yang menghasilkan spesifikasi sistem yang berupa manusia, perangkat keras, perangkat lunak, database, jaringan dan *user interface*.

Selain itu, penelitian yang dilakukan oleh Dahlan Abdullah tentang "Perancangan Sistem Informasi Perpustakaan : Studi Kasus pada Perpustakaan FTI UII dengan berbasis *web*", juga menggunakan metode SDLC. Penelitian ini membahas rancangan sistem informasi perpustakaan yang terkoneksi dengan internet.

Terakhir, Adityo Hidayat pada skripsi "Perancangan Perangkat Lunak dengan UML" menjelaskan tentang perancangan perangkat lunak menggunakan UML. UML merupakan metode pemodelan perangkat lunak, yang termasuk pada bahasa pemgrograman yang berbasis objek. UML terdiri atas seperangkat presentasi grafis dan notasi, yang terdiri dari : Diagram *class*, Diagram *use case*, Diagram interaksi, Diagram rangkaian, Diagram kolaborasi, Diagram keadaan, Diagram aktivitas, Diagram komponen, serta Diagram *deployment*. Disini dijelaskan tentang perancangan perangkat lunak menggunakan UML beserta contoh penerapannya pada sistem interkoneksi perpustakaan.

Dari beberapa literatur tersebut, penulis mengambil UML sebagai bahasa pemodelan. Perbedaan dengan penelitian lain yaitu penggunaan diagram UML terbatas pada diagram *class*, diagram *use case*, diagram aktivitas serta bantuan diagram *package* yang digunakan pada perancangan SIMAK. Hal ini dikarenakan rancangan sistem yang diharapkan ditinjau dari sisi pengguna, obyek (atribut dan operasi) dan aktivitas yang terjadi dalam sistem.