BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Kata sistem berasal dari bahasa Yunani "*systema*" yang artinya "Kesatuan" atau keseluruhan dari bagian – bagian yang saling berhubungan satu sama lainnya dan mempunyai tujuan yang sama. Secara umum pengertian sistem dapat didefenisikan sebagai suatu rangkaian unsur-unsur yang saling berhubungan dalam satu bentuk, untuk menjadi satu kesatuan.

Sistem berarti perangkat unsur yang secara teratur saling berkaitan sehingga membentuk suatu totalitas, susunan yang teratur dari pandangan, teori, asas, dan sebagainya (Dendy Sugono ; 2008 : 1362).

Sistem merupakan kumpulan komponen yang saling berhubungan satu dengan yang lainnya membentuk satu kesatuan untuk mencapai tujuan tertentu (Jogiyanto ; 2005 : 3).

Beberapa karakteristik sistem yang menunjang atau mendukung terlaksananya kegiatan untuk mencapai tujuan yaitu :

a. Komponen Sistem (*components*)

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

b. Batas Sistem (*boundry*)

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

c. Lingkungan Luar Sistem (*environments*)

Lingkungan luar dari suatu sistem adalah apapun di luar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat merugikan sistem tersebut.

d. Penghubung Sistem (*interface*)

Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber – sumber daya mengalir dari satu subsistem ke subsistem yang lainnya.

e. Masukan Sistem (*input*)

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*mainternance input*) dan masukan sinyal (*signal input*). Maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk mendapatkan keluaran.

f. Keluaran Sistem (*output*)

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

g. Pengolah Sistem (*process*)

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

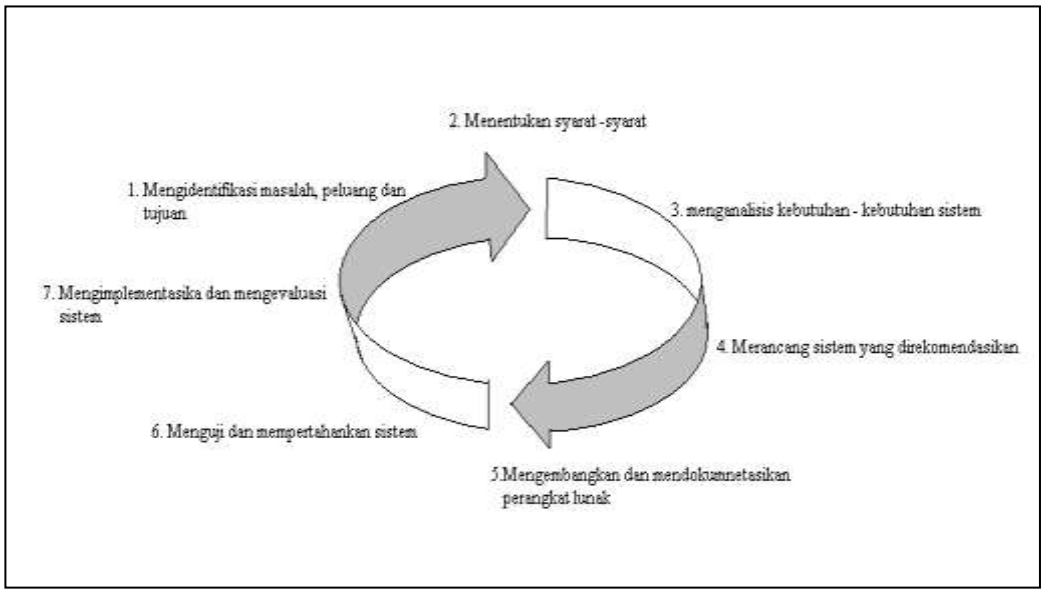
h. Sasaran Sistem (*objectives*)

Suatu sistem mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya. Sistem dapat menentukan masukkan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya (Tata Sutabri ; 2005:11).

Dari beberapa pengertian di atas maka dapat diambil kesimpulan bahwa sistem adalah sekumpulan komponen yang saling terkait dan bekerja sama untuk mencapai tujuan.

Siklus hidup pengembangan sistem adalah pengembangan melalui beberapa tahap untuk menganalisis dan merancang sistem yang dimana sistem tersebut telah dikembangkan dengan sangat baik melalui pengembangan siklus kegiatan penganalisisan dan pemakaian secara spesifik (Kenneth E. Kendall & Julie E. Kendall ; 2003 : 11)

Dari pengertian diatas dapat kita gambarkan tujuh tahap siklus hidup pengembangan sistem, sebagai berikut :



Gambar II.1. Siklus Hidup Pengembangan Sistem

(Kenneth E. Kendall. & Julie E Kendall ; 2003 : 11)

Langkah – langkah dari siklus hidup pengembangan sistem adalah :

1. Mengidentifikasi masalah, peluang dan tujuan

   Pada tahap pertama ini adalah tahapan pengidentifikasian masalah, peluang da tujuan – tujuan yang hendak dicapai. Tahapan ini sangat penting bagi keberhasilan proyek, karena titik awal dari perancangan sistem dimulai pada tahap ini.

2. Menentukan syarat – syarat

   Dalam tahap ini, penganalisis memasukkan apa saja yang menjadi syarat – syarat informasi untuk pemakai yang terlibat. Diantara perangkat – perangkat

yang digunakan, untuk menentukan syarat –syarat informasi diantaranya adalah menentukan sample dan memeriksa data mentah, wawancara lanjutan, mengamati prilaku pembuat keputusan dan lingkungan kantor.

3. Menganalisis kebutuhan – kebutuhan sistem

   Tahap berikutnya adalah menganalisa kebutuhan – kebutuhan sistem. Perangkat yang digunakan untuk membantu menganalisa kebutuhan adalah penggunaan diagram alir data untuk menusun input, proses dan output dalam grafik terstruktur. Dari diagram alir data, dikembangkan suatu kamus data berisikan daftar seluruh item data yang digunakan dalam sistem , berikut spesifikasinya.

4. Merancang sistem yang direkomendasikan

   Dala tahap desain dari siklus hidup engembangan sistem, penganalisis menggunakan informasi – informasi yang terkumpul sebelumnya dengan merancang prosedur datar entri yang tepat.

   Bagian perancangan sistem informasi yang tidak kalah pentingnya adalah desain antar muka pemakai. Antar muka pemakai merupakan penghubung antara pemakai dan sistem, dimana pemakai cukup paham dan mengerti apa yang diminta dan yang harus dilakukan dari sistem. Perancangan sistem yang lebih baik, jika sistem tersebut merupakan sistem *user friendly*, agar pemakai benar – benar merasa nyaman pada saat menggunakan sistem.

5. Mengembangkan dan mendokumentasikan perangkat lunak

   Dalam tahap ini penganalisa bekerja sama dengan pemogram untuk mengembangkan suatu perangkat lunak awal yang diperlukan. Beberapa tekhnik terstruktur untuk merancang dan mendokumentasikan perangkat lunak melalui rencana terstruktur, *Nassi-Sneiderman Charts* dan *pseudocode.* Penganalisis juga bekerjasama dengan pemakai untuk mengembangkan dokumentasi perangkat lunak, mencakup prosedur penggunaan secara manual, membuat fitur *Frequential Ask Questions (FAQ)* di file " Read Me " yang nantinya diserahkan bersama dengan perangkat lunak sistem.

6. Mengujikan dan mempertahankan sistem

   Sebelum sistem digunakan secara nyata, maka terlebih dahulu dilakukan uji kelayakan. Ujicoba dilakukan dengan memasukan data real dan aktual keseluruhan sampai batas tertentu. Akan menghemat biaya dan waktu jika ditemukan kesalahan – kesalahan yang terjadi pada saat ujicoba.

7. Mengimplementasikan sistem

Ditahap trakhir dari perancangan sistem, penganalisis membantu untuk mengomlementasikan penggunaan sistem. Tahap ini melibatkan pelatihan bagi pemakai untuk mengendalian sistem yang lama ke sistem yang baru. Pelatihan akan diserahkan kepadda *vendor* , akan tetapi kesalahan pelatihan menjadi tanggung jawab penganalisis. Selain itu penganalisis juga merekomendasikan pemakai untuk mengkonversi pekerjaan dari sistem yang lama ke sistem yang baru (Kenneth E. Kendall & Julie E. Kendall ; 2003 : 11)

II.1.1. Sistem Pendukung Keputusan

Menurut Wibowo S Henry, Dkk (2009: B62) dalam penelitiannya yang berjudul ***Sistem Pendukung Keputusan Untuk Menentukan Penerimaan Beasiswa Bank BRI Menggunakan FMADM*** membahas suatu kasus yaitu mencari altematif terbaik berdasarkan kriteria-kriteria yang telah ditentukan dengan menggunakan metode SAW *{Simple Additive Weighting)* untuk melakukan perhitungan metode FMADM pada kasus tersebut. Metode ini dipilih karena mampu menyeleksi altematif terbaik dari sejumlah altematif, dalam hal ini altematif yang dimaksudkan yaitu yang berhak menerima beasiswa berdasarkan kriteria-kriteria yang ditentukan. Penelitian dilakukan dengan mencari nilai bobot untuk setiap atribut, kemudian dilakukan proses perankingan yang akan menentukan altematif yang optimal, yaitu mahasiswa terbaik

Sistem pendukung keputusan (Inggris: *decision support systems* disingkat DSS) adalah bagian dari sistem informasi berbasis komputer (termasuk sistem berbasis pengetahuan (manajemen pengetahuan)) yang dipakai untuk mendukung pengambilan keputusan dalam suatu organisasi atau perusahaan.

Dapat juga dikatakan sebagai sistem komputer yang mengolah data menjadi informasi untuk mengambil keputusan dari masalah semi-terstruktur yang spesifik.

SPK dapat digambarkan sebagai sistem yang berkemampuan mendukung analisis ad hoc data, dan pemodelan keputusan, berorientasi keputusan, orientasi perencanaan masa depan, dan digunakan pada saat-saat yang tidak biasa.

Tahapan SPK:

1. Definisi masalah
2. Pengumpulan data atau elemen informasi yang relevan
3. pengolahan data menjadi informasi baik dalam bentuk laporan grafik maupun tulisan
4. menentukan alternatif-alternatif solusi (bisa dalam persentase)

Tujuan dari SPK:

1. Membantu menyelesaikan masalah semi-terstruktur.
2. Mendukung manajer dalam mengambil keputusan.

II.2. Pengertian Kredit

Menurut Undang-Undang Perbankan No. 10 tahun 1998 Kredit adalah penyediaan uang atau tagihan yang dapat dipersamakan dengan itu, berdasarkan persetujuan atau kesepakatan pinjam meminjam antara bank dengan pihak lain yang mewajibkan pihak peminjam melunasi utangnya setelah jangka waktu tertentu dengan pemberian bunga (Suryatiningsih, ST, MT, Linda Iresa, Fitri Sukmawati, S.E.M.M ; 2011 : 3).

II.2.1. Tujuan, Fungsi dan Jenis Kredit

Tujuan kredit dapat dilihat dari sudut pandang ekonomi mikro dan makro. Dari sudut pandang ekonomi mikro, tujuan pemberian kredit guna mendapatkan suatu nilai tambah baik bagi nasabah (debitur) maupun bagi bank (kreditur).

Fasilitas kredit berfungsi untuk :

1. Meningkatkan daya guna.
2. Meningkatkan peredaran dan lalu lintas uang.
3. Meningkatkan kegairahan berusaha, dengan adanya kredit nasabah akan bergairah untuk dapat memperbesar atau memperluas usahanya.

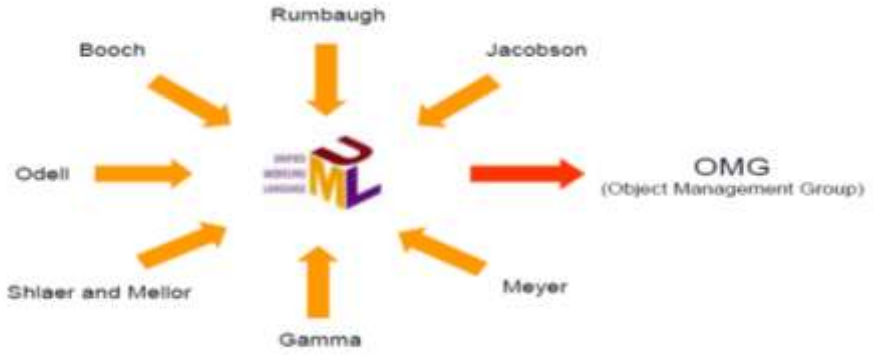
Jenis-jenis kredit dilihat dari berbagai segi terdiri dari :

1. Kredit investasi yaitu kredit yang biasanya digunakan untuk keperluan perluasan usaha atau membangun proyek/pabrik dimana masa pemakaiannya untuk satu periode yang relatif lebih lama.
2. Kredit modal kerja yaitu kredit yang digunakan untuk keperluan meningkatkan produksi operasionalnya.

II.3. UML

Menurut Sri Dharwiyanti, dan Romi Satria Wahono (2003 : 2) 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. 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.



Gambar II.2 *: Metodologi UML*

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikatakan 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 [7] [8] [9]. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

II.3.1. Konsep Dasar UML

Konsep dasar UMl dari berbagai dokumen dan buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar di bawah ini.

Tabel : Konsep Dasar UML

Major Area	View	Diagrams	Main Concepts
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include,use case generalization
	implementation view deployment	component diagram	component, interface, dependency
	view		incation
dynamic	state maching view	statchart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, lork, join
	Interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration,interactio, collaboration, role, message
Model management	model management view	class diagram	package, subsystem, model

extensibility	all	all	constraint, stercotype, tagged values

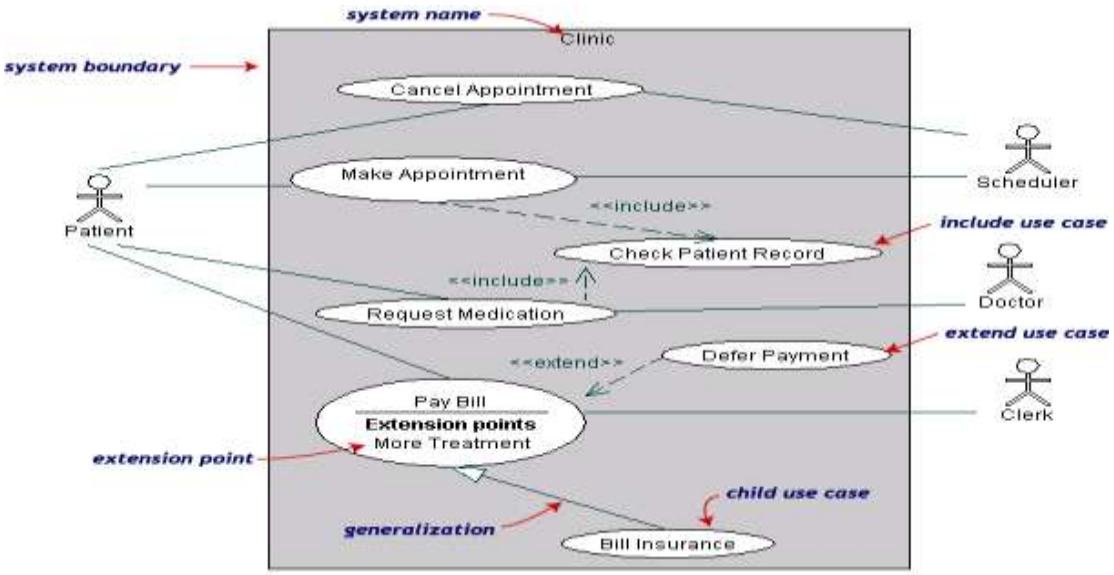
Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II.3.2. *Use case diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri.

Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Contoh diagram Ucase:



Gambar II.3. : *Contoh Diagram Ucase*

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II. 3.3. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan

(atribut/properti) suatu bentuk sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut.

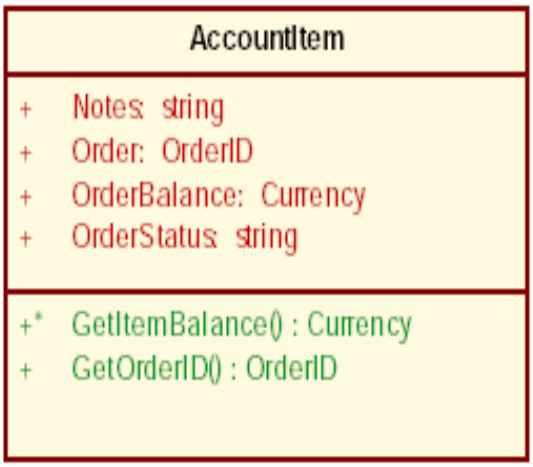
Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja

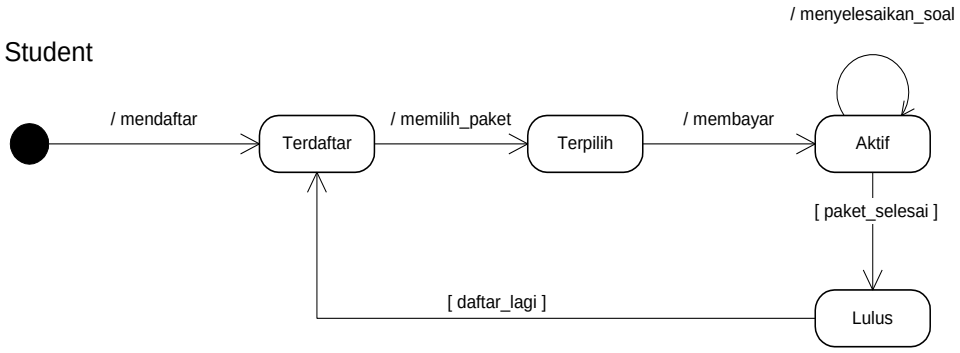


Gambar II.4. *Class Diagram*

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II.3.4. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu statechart diagram).

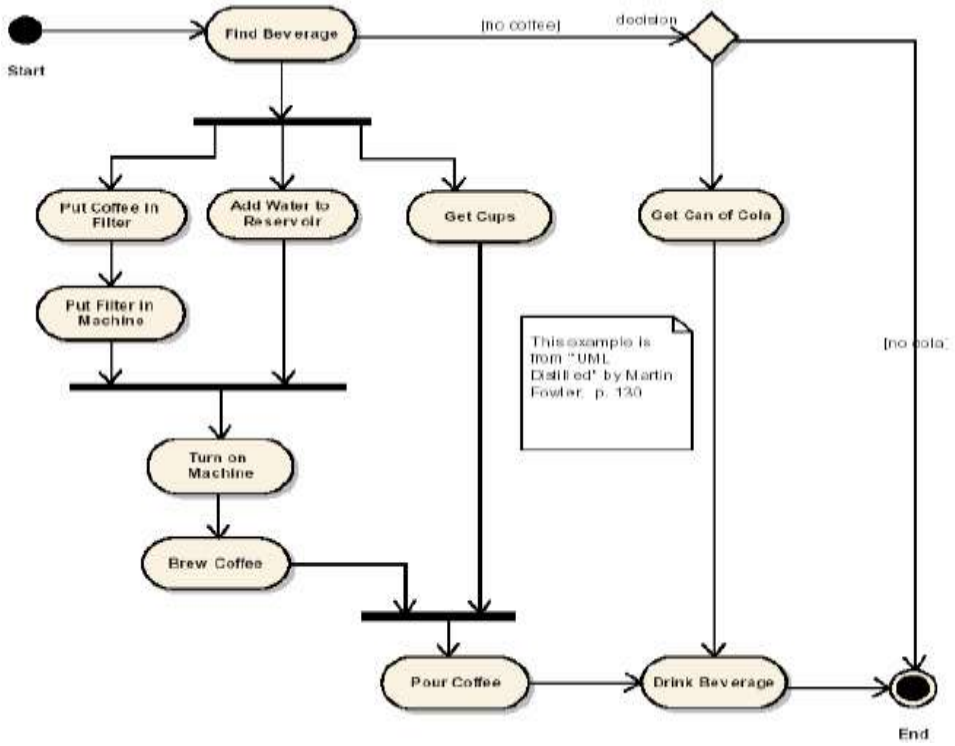


Gambar II.5. ***Contoh Statechart Diagram***

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II.3.5. Activity Diagram

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



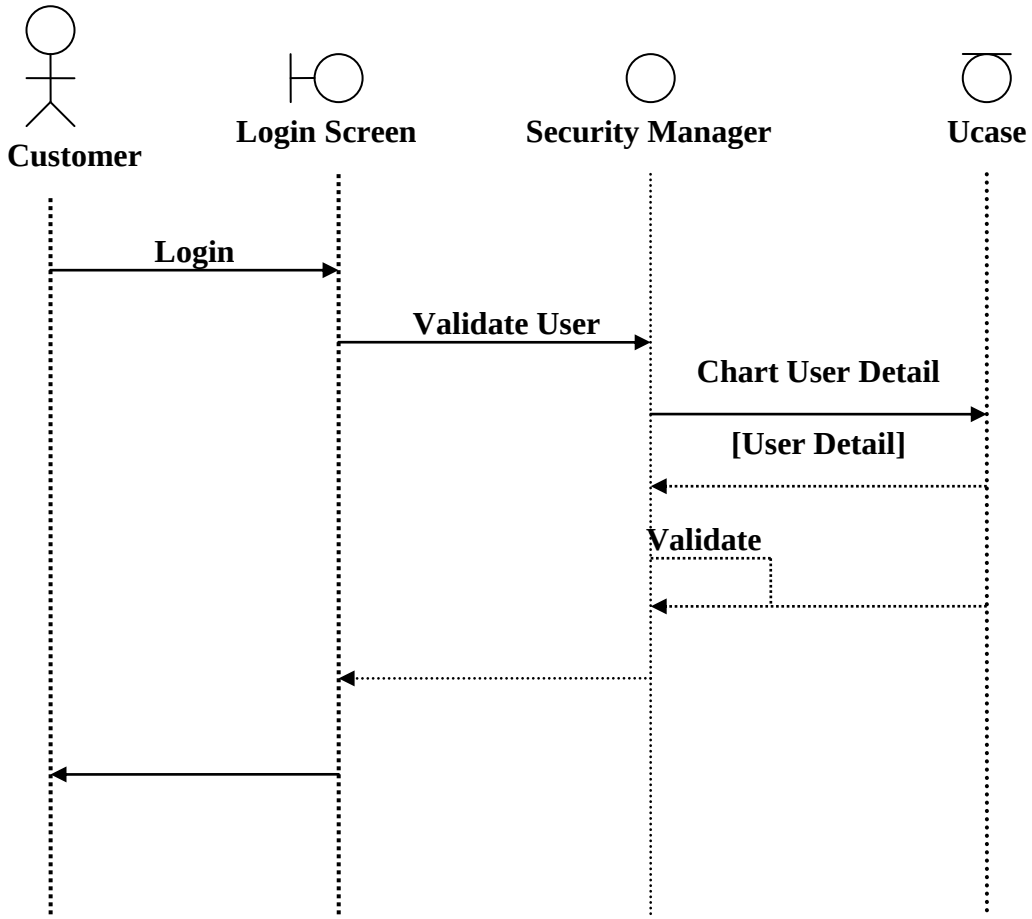
Gambar II.6. Contoh *Activity Diagram* - Tanpa *Swimlane*

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II.3.6. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Contoh Sequence Diagram :



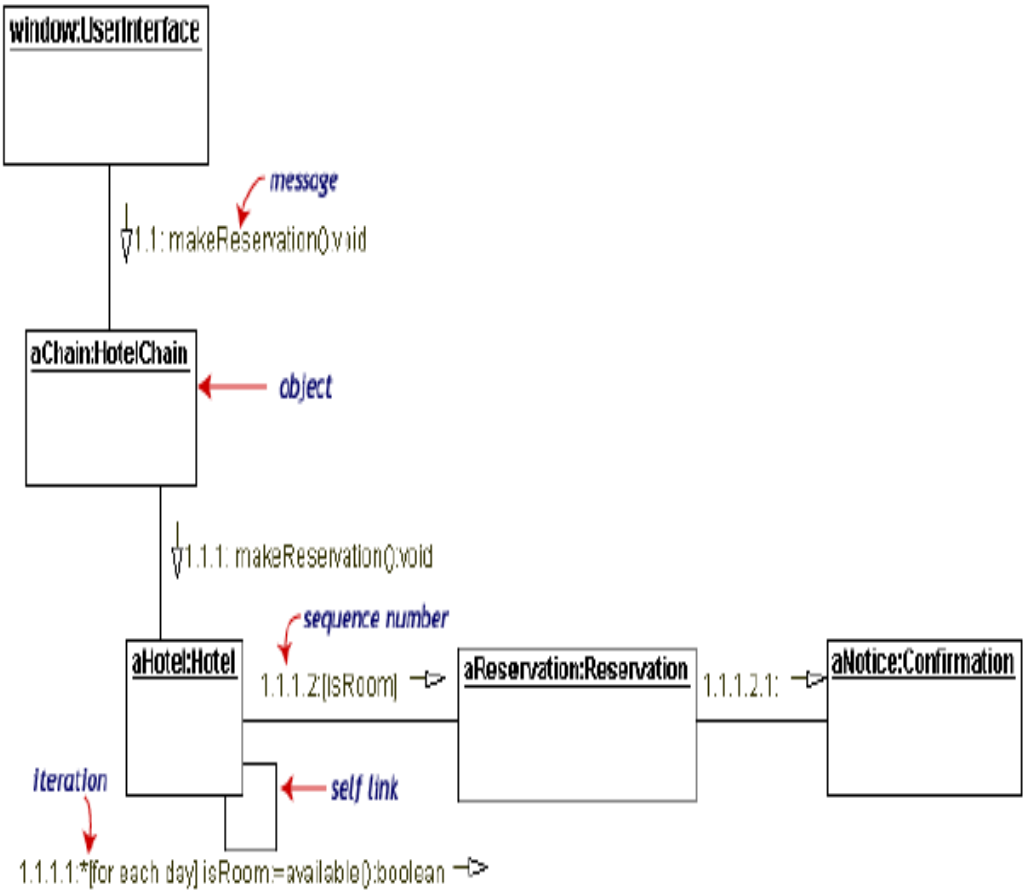
I : Use Case Model

Gambar II.7. ***Contoh Sequence Diagram***

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II.3.7. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti sequence diagram, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message. Setiap message memiliki sequence number, di mana message dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

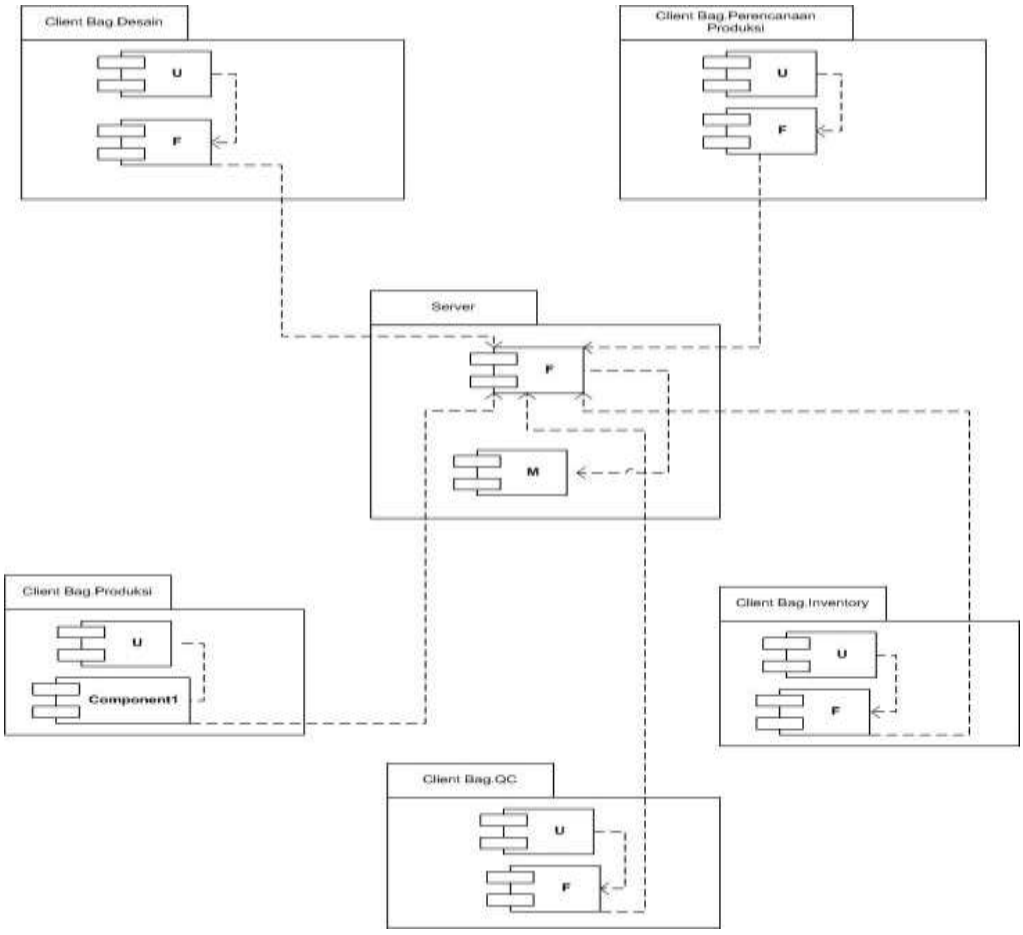


Gambar II.8. ***Contoh Collaboration Diagram***

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II.3.8. Component Diagram

Komponen piranti lunak adalah modul berisi code, baik berisi source code maupun binary code, baik library maupun executable, baik yang muncul pada compile time, link time, maupun run time. Umumnya komponen terbentuk dari beberapa class dan/atau package, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa interface, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.



Gambar II.9. ***Contoh Component Diagram***

Sumber : *Sri Dharwiyanti and Romi Satria Wahono, Pengantar Unified Modeling Language (UML), 2003.*

II.4. Database

Data base berasal dari kata data dan base, data adalah suatu bentuk keterangan baik keterangan numerik, grafik ataupun dalam bentuk lainnya yang digunakan dan diproses dengan baik oleh komputer untuk menghasilkan informasi. Simbol Database dapat dilihat pada tabel II.4.

Tabel II.2. Jenjang Data

Data	Keterangan
Karakter ↓ Field ↓ Record ↓ File ↓ Data Base	Karakter bagian data terkecil yang dapat berupa karakter numerik, huruf, symbol
	Field sekumpulan aksara yang memberi suatu makna bagi mewakili satu unit data seperti nim, nama, tempat tanggal lahir, dan sebagainya
	Record kumpulan dari field yang berhubungan yang menerangkan tentang suatu objek atau menggambarkan dari satu unit dari individu tertentu
	File kumpulan dari record - record yang saling

	berhubungan, menggambarkan satu kesatuan dari data yang sejenis
	Data Base kumpulan dari file – file yang saling berhubungan

(Sumber: Abdul Kadir ; 2003 : 19)

Base adalah dasar atau landasan sebagai pengulangan lanjutan, dengan demikian data base dapat disimpulkan sebagai kumpulan data yang teratur dan saling berhubungan yang dapat digunakan bersama – sama aplikasi tertentu secara terpadu sehingga menghasilkan suatu informasi yang teratur yang disimpan ke dalam komputer.

II.4.1. Implementasi Basis Data

Tahap implementasi basis data merupakan upaya untuk membangun basis data fisik yang ditempatkan dalam memori sekunder dengan bantuan DBMS (Database Management System) yang dipilih. Secara umum sebuah diagram Er akan direpresentasikan menjadi sebuah basis data secara fisik, sedangkan komponen-komponen diagram ER yang berupa himpunan entitas dan himpunan relasi akan ditranformasi menjadi tabel-tabel (file-file data) yang merupakan komponen utama pembentuk basis data. Setiap himpunan entitas akan diimplementasikan menjadi sebuah tabel (file data), sedangkan himpunan relasi tergantung pada derajad relasi. Untuk kardinalitas relasi 1 - 1 , maka relasi tidak diimplentasi menjadi tabel tetapi atribut pada relasi akan disertakan pada tabel

yang mewakili salah satu dari kedua himpunan entitas dengan mempertimbangakn derajad relasi minimunnya. Untuk kardinalitas relasi 1 – N juga akan direpresentasikan dengan penambahan atribut yang ada pada relasi ke tabel yang mewakili himpunan entitas yang berderajad banyak (N).

II.5. SDLC

SDLC *(System Development Life Cycle)* atau Siklus Hidup Pengembangan Sistem adalah proses logis yang digunakan oleh analisis sistem untuk menggambarkan sebuah sistem informasi. SDLC merupakan siklus pengembangan sistem, meliputi langkah-langkah :

1. Plaining / Perencanaan

   Penulis merencanakan langkah-langkah yang akan dibuat dalam pengumpulan data dan dalam mengidentifikasi masalah serta merumuskan masalah-masalah yang dijumpai penulis dalam melakukan penelitian.

2. Analisa

   Setelah mengidentifikasi masalah, penulis menganalisa masalah dan sistem yang ada untuk diambil hipotesisnya guna memperbaiki sistem yang ada.

3. Desain

   Setelah menganalisa masalah dari sistem yang ada, maka penulis menguji hipotesis tersebut apakah didukung oleh data/informasi yang diperoleh.

4. Implementasi

Setelah menguji hipotesis yang diperoleh, maka dilakukan implementasi dari hasil analisa dan desain yang telah dilakukan sehingga pada tahap ini menghasilkan suatu perangkat lunak (*software*) untuk pemecahan masalah.

5. Testing / Pengujian

   Setelah perangkat lunak dibangun/dirancang, maka dilakukan pengujian untuk menguji tingkat kehandalan perangkat lunak yang telah dibangun. Hal ini dilakukan untuk memastikan kehandalan software yang akan digunakan, guna memperbaiki system yang ada.

6. Pemeliharaan / *Maintenance*

   Pemeliharaan sangat penting dalam SDLC. Tahap ini dilakukan untuk memperbaiki sistem yang telah dibangun. Setelah itu tahapan ini juga untuk penambahan dan perubahan sistem secara berkala.

II.6. PHP

PHP merupakan bahasa skrip yang digunakan untuk membuat halaman web yang dinamis. PHP bersifat *open source product*. Pengguna dapat mengubah source code dan mendistribusikannya secara bebas serta diedarkan secara gratis. PHP bersifat server side scripting yang dapat ditambahkan ke dalam HTML, sehingga suatu halaman web tidak lagi bersifat statis, namun bersifat dinamis. Sifat server-side berarti pengerjaan skrip PHP akan dilakukan di sebuah web server, kemudian hasilnya akan dikirimkan ke browser. Salah satu web server yang paling umum digunakan untuk PHP adalah Apache. PHP dapat dijalankan

pada sistem operasi Unix, Windows, dan Mac OS X. (Didik Dwi Prasetyo; 2008 : 2).

PHP 5 dirilis pada bulan juli 2004 dengan inti Zend Engine 2.0. PHP 5 idalah versi PHP terbaru yang mendukung penuh Object-Oriented Programming (OOP), integrasi XML, mendukung semua ekstensi terbaru SQL, pengembangan web services dengan SOAP dan REST, serta rarusan peningkatan lainnya dibandingkan versi sebelumnya, PHP 4. (Arief Ramadhan, S.Kom, Hendra Saputra, S.Kom ; 2005 : 1).

II.7. MySQL

MySQL merupakan software yang tergolong sebagai DBMS (*Database Management System*) yang bersifat *Open Source*. Open Source menyatakan bahwa software ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat MySQL), selain tentu saja bentuk executable-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi, dan bisa diperoleh dengan cara men-download (mengunduh) di Internet secara gratis.

MySQL awalnya dibuat oleh perusahaan konsultan bernama TcX yang berlokasi di Swedia. Saat ini pengembangan MySQL berada di bawah naungan perusahaan MySQL AB. Adapun software dapat diunduh di situs www.mysql.com.Sebagai software DBMS, MySQL memiliki sejumlah fitur seperti yang dijelaskan di bawah ini.

a. Multiplatform

MySQL tersedia pada beberapa platform (Windows, Linux, Unix, dan Iain-lain).

b. Andal, cepat, dan mudah digunakan

MySQL tergolong sebagai database server (server yang melayani permintaan terhadap database) yang andal, dapat menangani database yang besar dengan kecepatan tinggi, mendukung banyak sekali fungsi untuk mengakses database, dan sekaligus mudah untuk digunakan. Berbagai tool pendukung juga tersedia (walaupun dibuat oleh pihak lain). Perlu diketahui, MySQL dapat menangani sebuah tabel yang berukuran dalam terabyte (1 terabyte = 1024 gigabyte). Namun, ukuran yang sesungguhnya sangat bergantung pada batasan sistem operasi. Sebagai contoh, pada sistem Solaris 9/10, batasan ukuran file sebesar 16 terabyte.

c. Jaminan keamanan akses

MySQL mendukung pengamananan database dengan berbagai kriteria pengaksesan. Sebagai gambaran, dimungkinkan untuk mengatur user tertenru agar bisa mengakses data yang bersifat rahasia (misalnya gaji pegawai), sedangkan user lain tidak boleh. MySQL juga mendukung konektivitas ke berbagai software. Sebagai contoh, dengan menggunakan ODBC (*Open Database Connectivity*), database yang ditangani MySQL dapat diakses melalui program yang dibuat dengan Visual Basic. MySQL juga mendukung program klien yang berbasis Java untuk berkomunikasi dengan database MySQL melalui JDBC (*Java Database Connectivity*). MySQL juga

bisa diakses melalui aplikasi berbasis Web; misalnya dengan menggunakan PHP.

Seperti tersirat dalam namanya, MySQL mendukung perintah SQL (*Structured Query Language*). Sebagaimana diketahui, SQL merupakan standar dalam pengaksesan database relasional. Pengetahuan akan SQL akan memudahkan siapa pun untuk menggunakan MySQL (M. Agus J. Alam ; 2005 : 3).