BAB II

TINJAUAN PUSTAKA

II.1. Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Contoh utama aplikasi adalah pengolah kata, lembar kerja, memanipulasi foto, merancang rumah dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi suatu paket disebut sebagai suatu paket atau *suite* aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice.org*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja dan beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Sering kali, mereka memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah. (*Dahlan Abdullah ; 2013 : 152*)

Jenis-jenis Software Aplikasi menurut *Dahlan Abdullah 2013* adalah sebagai berikut :

1. Software aplikasi hiburan, contohnya yaitu winamp untuk mendengarkan musik, games dan sebagainya untuk hiburan.

2. Software aplikasi pendidikan yaitu software digunakan untuk mempelajari atau mereferensikan tentang pendidikan atau pengetahuan.
3. Software aplikasi bisnis yaitu software yang digunakan untuk aplikasi bisnis
4. Software aplikasi khusus
5. Software aplikasi untuk produtivitas kerja.

II.2. Pengertian Sistem Informasi

II.2.1. Sistem

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

II.2.2. Informasi

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

II.2.3. Sistem Informasi

Sistem Informasi secara umum adalah merupakan kegiatan atau aktifitas yang melibatkan serangkaian proses, berisi informasi-informasi yang digunakan untuk mencapai tujuan. (*Ika Nur Indah ; 2013 :125*)

II.3. Sistem Informasi Akutansi

Sistem informasi akuntansi merupakan sistem yang berfungsi untuk mengumpulkan dan menyimpan data tentang aktivitas-aktivitas yang dilaksanakan organisasi, mengubah data tersebut menjadi informasi yang berguna bagi pihak manajemen dan membuat perencanaan serta menyediakan pengendalian yang memadai untuk menjaga aset-aset organisasi. Tanpa adanya sistem informasi akuntansi yang mengawasi aktivitas - aktivitas yang berlangsung, perusahaan atau organisasi akan mengalami kesulitan untuk menentukan seberapa baik kinerjanya dan juga akan mengalami kesulitan dalam menelusuri bagaimana pengaruh-pengaruh dari berbagai aktivitas atas sumberdaya - sumberdaya yang ada dibawah pengawasannya. Oleh karena itu, sistem informasi akuntansi yang efektif sangatlah penting bagi keberhasilan jangka panjang organisasi manapun. (*Merystika Kabuhung ; 2013 : 340*).

Sistem Informasi Akuntansi adalah perusahaan yang mempekerjakan orang-orang dan menggunakan catatan – catatan, serta prosedur - prosedur untuk mengubah data ekononomi menjadi informasi keuangan yang diperlukan pihak internal maupun eksternal. Pihak internal yang menggunakan yaitu manajer, maupun eksternal, yaitu pelanggan, pemasok, pemilik saham, kreditor, satuan buruh, pihak bank, pemerintah, dan para stakeholder lainnya (*Wilkinson et al. 2000, p7).*

Sistem informasi akuntansi yang baik dalam pelaksanaannya diharapkan akan memberikan atau menghasilkan informasi - informasi yang berkualitas serta bermanfaat bagi pihak manajemen, serta pemakai pemakai informasi lainnya dalam pengambilan keputusan. Sistem informasi akuntansi yang baik dirancang

dengan sedemikian rupa sehingga dapat memenuhi fungsinya, yaitu mengasilkan informasi akuntansi yang tepat waktu, relevan dan dipercaya. Oleh karena itu, baik buruknya suatu sistem informasi dapat mempengaruhi fungsi manajemen dalam melakukan pengendalian internal karena informasi yang dihasilkan dapat dipergunakan untuk hal pengambilan keputusan.

II.4. *Visual Basic*

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

II.5. *Database*

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

II.5.1. Jenis-Jenis *Database*

1. *Database* Hirarki

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

2. *Database* Relasi

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

II.5.2. Prinsip-Prinsip *Database*

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

1. Ketersediaan

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

2. Pemakaian bersama

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

3. Pengembangan

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

4. Kesatuan

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

II.5.3. Pengertian Database Management System (DBMS)

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

II.6. MySql

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

II.7. Metode *Sliding Rate*

Sliding Rate, merupakan perhitungan suku bunga yang dilakukan dengan mengalikan persentase suku bunga per periode dengan sisa pinjaman, sehingga jumlah suku bunga yang dibayar debitor semakin menurun, akibatnya angsuran yang dibayar pun menurun jumlahnya. (*Jandry R. Merung ; 2013 : 631*)

Berikut contoh penghitungan menggunakan metode sliding rate menurut *Nanik Susanti ; 2014 : 43* :

Pokok pinjaman : Rp. 12.000.000,-

Bunga : 15 % per bulan

Jangka waktu : 6 bulan

Angsuran pokok : Rp.12.000.000,-/ 6 bulan = Rp. 2.000.000,-

$$\text{Bunga} = \frac{12.000.000,- \times 15\%}{12} = 150.000,-$$

Angsuran I = 2.000.000 + 150.000 = Rp. 2.150.000,-

Cicilan Bunga bulan 2

Karena bulan pertama sudah membanyar 2.000.000,- maka pokok pinjaman jadi sisa 10.000.000,-

Cicilan bulan kedua adalah

$$\text{Bunga} = \frac{10.000.000,- \times 15\%}{12} = 125.000,-$$

Total Cicilan bulan ke 2 adalah

2.000.000 + 125.000 = Rp. 2.125.000,-

Dan seterusnya sampai bulan ke 6.

Dengan metode perhitungan *Sliding Rate* maka besarnya bunga berubah berdasarkan saldo pinjaman tiap periodenya.

Pembebanan bunga setiap bulan akan disesuaikan dengan sisa pinjamannya, sehingga angsuran (cicilan) bunga akan menurun seiring dengan berkurangnya nilai pinjaman, tetapi angsuran pokok akan tetap.

II.8. DFD (Data *Flow* Diagram)

Data *Flow* Diagram (DFD) adalah alat pembuatan model yang memungkinkan professional sistem untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan nama Bubble chart, Bubble diagram, model proses, diagram alur kerja, atau model fungsi.

DFD ini merupakan alat perancangan system yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa

maupun rancangan ,sistem yang mudah dikomunikasikan oleh professional sistem kepada pemakai maupun pembuat program. (*Dahlan Abdullah ; 2013 : 154*)

II.9. UML (*Unified Modelling Language*)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, *Java*, C# atau *VB.NET*. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C. Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: *Grady Booch OOD* (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah:

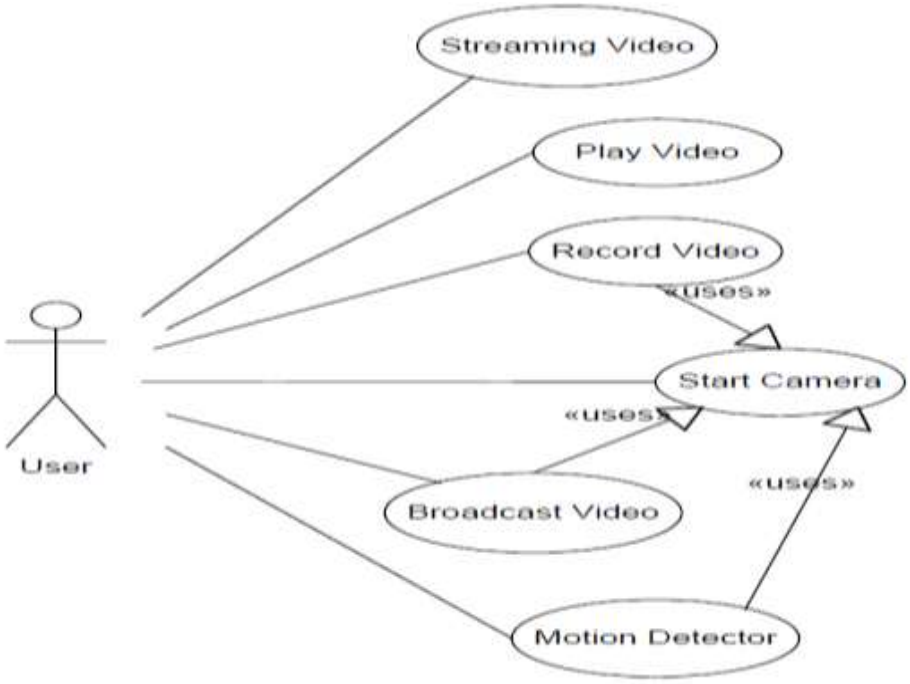
metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul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

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

1. *Use Case Diagram*

Use case diagram adalah diagram yang menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar dan menjelaskan sistem secara fungsional yang terlihat pengguna. Yang

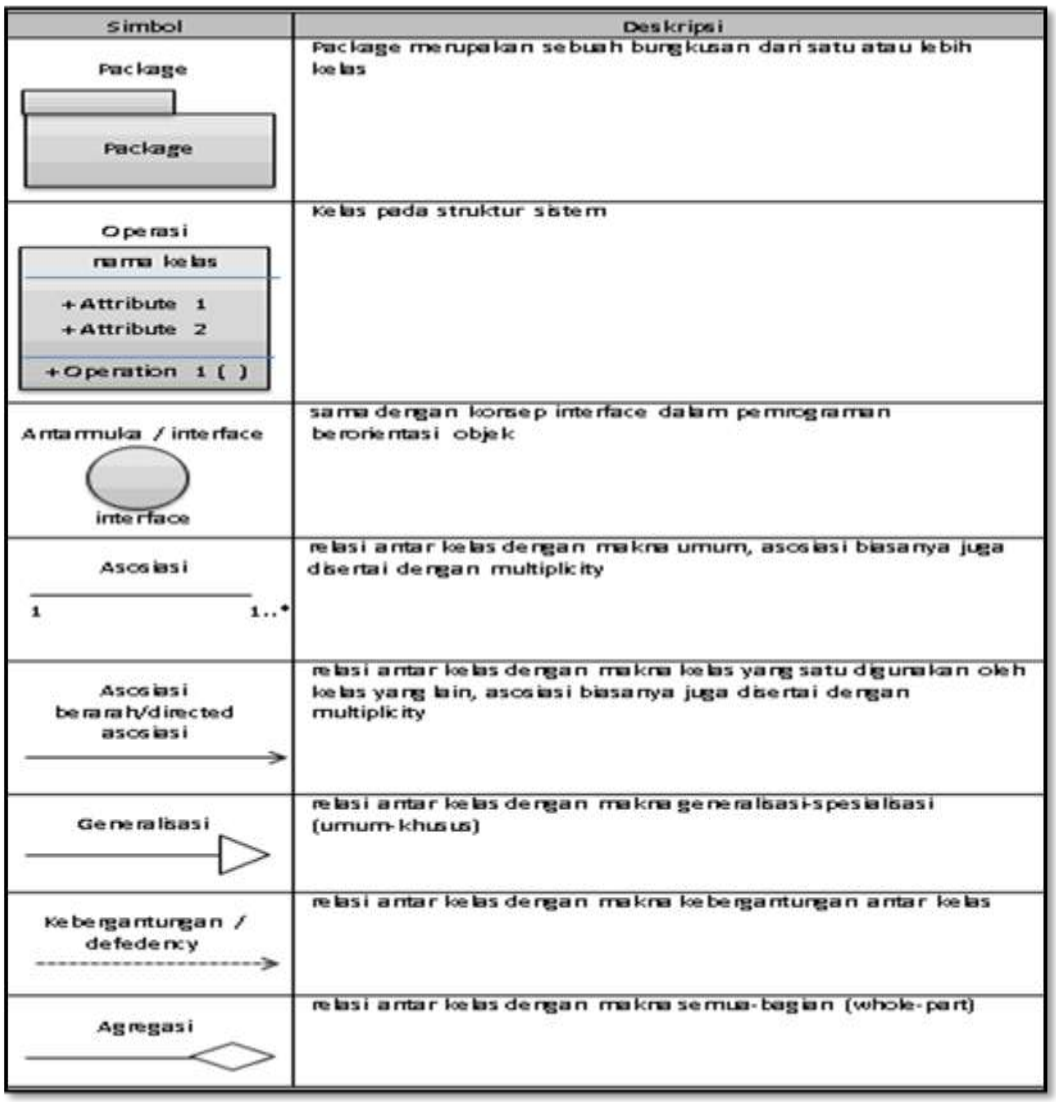
ditekankan dalam *use case* adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. *Use case* mempresentasikan sebuah interaksi antara aktor dan sistem. *Use case* dapat digunakan selama proses analisis untuk menangkap persyaratan sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama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



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

2. *Class Diagram*

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



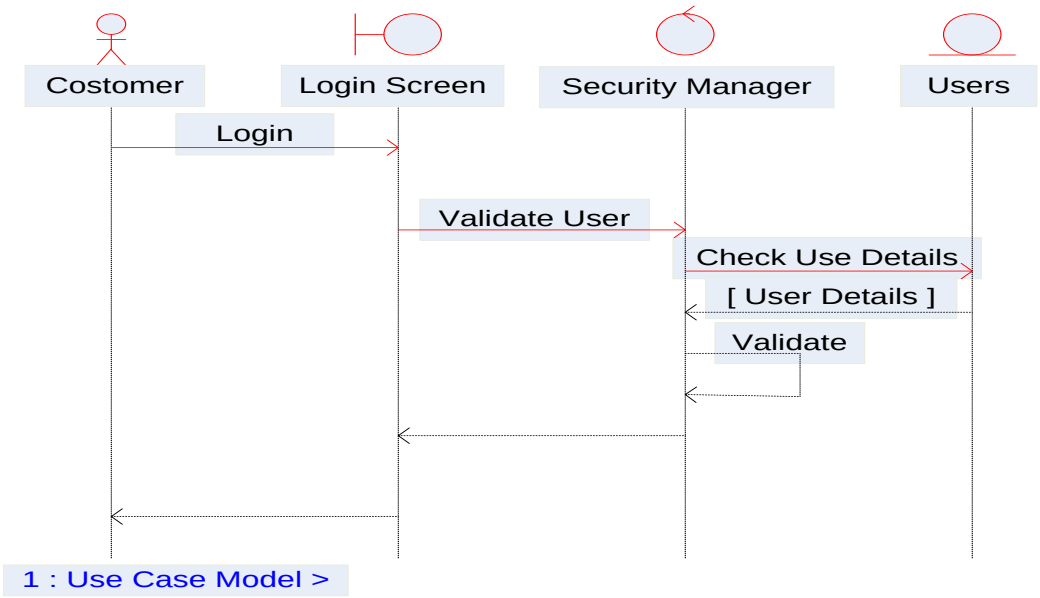
Simbol	Deskripsi
Package Package	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi nama kelas +Attribute 1 +Attribute 2 +Operation 1 ( )	Kelas pada struktur sistem
Antarmuka / interface interface	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 1 1..*	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity
Asosiasi berarah/directed asosiasi	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan multiplicity
Generalisasi	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan / defedency	relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi	relasi antar kelas dengan makna semua-bagian (whole-part)

Gambar II.2. *Class* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 59)

3. *Sequence Diagram*

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

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



Gambar II.3. *Contoh Sequence Diagram*
Sumber : (Yuni Sugiarti ; 2013 : 63)

4. *Activity Diagram*

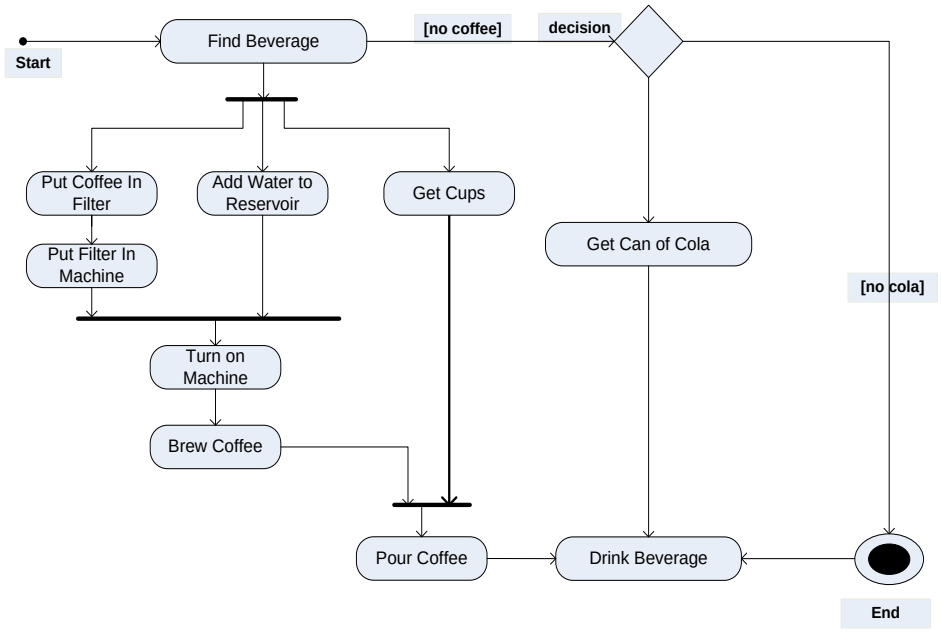
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak

menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar II.4. *Activify* Diagram
Sumber : (Yuni Sugiarti ; 2013 : 76)