BAB II

TINJAUAN PUSTAKA

II.1. Implementasi

Implementasi sistem merupakan tahap penerapan sistem dimana sistem siap untuk dioperasikan. Tahapan yang akan dilakukan untuk mengimplementasikan aplikasi sistem ujian berbasis LAN adalah sebagai berikut :

1. Menyelesaikan Tampilan atau Desain

   Menyelesaikan persiapan pembangunan *database*, menyelesaikan persiapan untuk membangun kode program untuk pengolahan *database*, menyelesaikan desain *input* dan *output* perangkat lunak.
2. Menentukan perangkat keras (*Hardware*) dan perangkat lunak (*Software*) yang dibutuhkan. (Surya Darma Nasution ; 2013 : 99)

II.2. 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*). (Dahlan Abdullah ; 2013 : 152)

II.3. Transfer Data Atau *File*

Transfer data adalah jumlah data dalam *bit* yang melewati suatu medium dalam satu detik dimana simbol verbal dan nonverbal dikirimkan, diterima dan diberi arti. Umumnya dituliskan dalam *bit* per detik (*bit per second*) dan disimbolkan *bit/s* atau *bps* bukan *bits/s*. (Sony Bahagia Sinaga ; 2012 : 45)

Adapun tipe transfer data komunikasi logika menurut Sony Bahagia Sinaga tahun 2012 pada lapisan *transport* dapat berbentuk sebagai berikut :

1. *Reliable* atau *unreliable*

   Reliable adalah data berarti data ditransfer ke tujuannya dalam suatu urutan seperti ketika dikirim. *Unreliable* adalah pengiriman data, *Unreliable* sangat menggantungkan diri pada lapisan jaringan di bawahnya, sehingga tidak dapat meyakinkan apakah *segment* data dapat dikirimkan sampai ditujuannya atau tidak.

2. *Stateful* atau *Stateless*

   Stateful adalah informasi yang dimasukkan pada satu *request*, yang dikirimkan dari pengirim ke penerima, dapat dimodifikasi untuk *request* berikutnya. *Stateless* adalah informasi dalam satu *request* tidak dapat dikaitkan dengan *request* lainnya, sehingga tidak dapat digunakan untuk *request* lainnya. (Sony Bahagia Sinaga ; 2012 : 46)

II.4. Jaringan

Jaringan komputer merupakan sekumpulan komputer serta perangkat-perangkat lain pendukung komputer yang saling terhubung dalam satu kesatuan. Media jaringan komputer dapat melalui kabel-kabel atau tanpa kabel sehingga

memungkinkan pengguna jaringan komputer dapat saling melakukan pertukaran informasi seperti dokumen dan data. Dapat juga melakukan pencetakan pada *printer* yang sama dan bersama-sama memakai perangkat keras dan perangkat lunak yang terhubung dengan jaringan. (Choirul Muallifah ; 2013 : 2)

II.4.1. Jenis-Jenis Jaringan

Berdasarkan jangkauan area atau lokasi jaringan menurut Choirul Muallifah tahun 2013 dibedakan menjadi 3 jenis, yaitu :

1. LAN (*Local Area Network*)

   Merupakan jaringan lokal yang dibuat pada area tertutup. Misalkan dalam satu gedung atau dalam satu ruangan. LAN biasa digunakan untuk jaringan kecil yang menggunakan *resource* bersama. Seperti penggunaan *printer* secara bersama. LAN dapat menggunakan media komunikasi seperti kabel dan *wireless*.

2. MAN (*Metropolitan Area Network*)

   Merupakan jaringan antara LAN satu dengan LAN lain yang dipisahkan daerah lokasi yang cukup jauh. Contoh penggunaan MAN adalah hubungan antara kantor pusat dengan kantor cabang yang ada di daerah-daerah. Dapat dikatakan MAN merupakan pengembangan dari LAN.

3. WAN (*Wide Area Network*)

   Merupakan jaringan yang cakupannya lebih luas dari pada MAN. Cakupan WAN meliputi satu kawasan, satu negara, satu pulau, bahkan satu benua. Metode yang digunakan WAN hampir sama dengan LAN dan MAN. (Choirul Muallifah ; 2013 : 2)

II.4.2. Manfaat dan Keuntungan Jaringan

Adapun manfaat dan keuntungan membangun jaringan menurut Choirul Muallifah tahun 2013 adalah sebagai berikut :

1. Manfaat Jaringan Komputer:
    a. *Sharing resources*
    b. Media komunikasi
    c. Integrasi data
    d. Pengembangan dan pemeliharaan
    e. Keamanan data
    f. Sumber daya lebih efisien dan informasi terkini
2. Keuntungan Membangun Jaringan Komputer Yaitu:
    a. Dapat berbagi peralatan (*peripheral*) dan penggunaanya, seperti *printer*, *harddisk*, modem.
    b. Memudahkan bertukar data diantara pengguna komputer tanpa harus menggunakan disket.
    c. Kita dapat menggunakan program-program yang ada di komputer pusat.
    d. Bisa mengirim dan menerima *email* dari *internet* dan mencari informasi lain melalui fasilitas *internet*.

II.5. *Wifi*

Awalnya *wifi* ditujukan untuk penggunaan perangkat nirkabel dan Jaringan Area Lokal (LAN), namun saat ini lebih banyak digunakan untuk mengakses *internet*. Hal ini memungkinan seseorang dengan komputer dengan kartu nirkabel (*wireless card*) atau *personal digital assistant* (PDA) untuk terhubung dengan

internet dengan menggunakan titik akses (*hotspot*) terdekat. *Wifi* dirancang berdasarkan spesifikasi IEEE 802.11. Sekarang ini ada empat variasi dari 802.11, yaitu: 802.11a, 802.11b, 802.11g, dan 802.11n. Spesifikasi b merupakan produk pertama *wifi*. Variasi g dan n merupakan salah satu produk yang memiliki penjualan terbanyak pada 2005. (Sony Bahagia Sinaga ; 2012 : 47)

Spesifikasi *wifi* yaitu :

Tabel II.1. Spesifikasi *Wifi*

Spesifikasi	Kecepatan	Frekuensi Band	Cocok dengan
802.11b	11 Mb/s	2.4 GHz	b
802.11a	54 Mb/s	5 GHz	A
802.11g	54 Mb/s	2.4 GHz	b,g
802.11n	100 Mb/s	2.4 GHz	b,g,n

Versi *wifi* yang paling luas dalam pasaran AS sekarang ini (berdasarkan dalam IEEE 802.11b/g) beroperasi pada 2.400 MHz sampai 2.483,50 MHz. Dengan begitu mengijinkan operasi dalam 11 *channel* (masing-masing 5 MHz), berpusat di frekuensi berikut :

1. *Channel* 1 - 2,412 MHz
2. *Channel* 2 - 2,417 MHz
3. *Channel* 3 - 2,422 MHz
4. *Channel* 4 - 2,427 MHz
5. *Channel* 5 - 2,432 MHz
6. *Channel* 6 - 2,437 MHz
7. *Channel* 7 - 2,442 MHz
8. *Channel* 8 - 2,447 MHz
9. *Channel* 9 - 2,452 MHz

10. *Channel* 10 - 2,457 MHz

11. *Channel* 11 - 2,462 MHz

Secara teknis operasional, *wifi* merupakan salah satu varian teknologi komunikasi dan informasi yang bekerja pada jaringan dan perangkat WLAN (*wireless local area network*). Dengan kata lain, *wifi* adalah sertifikasi *merk* dagang yang diberikan pabrikan kepada perangkat telekomunikasi (*internet*) yang bekerja di jaringan WLAN dan sudah memenuhi kualitas kapasitas interoperasi yang dipersyaratkan. (Sony Bahagia Sinaga ; 2012 : 47)

Teknologi *internet* berbasis *wifi* dibuat dan dikembangkan sekelompok insinyur Amerika Serikat yang bekerja pada *Institute of Electrical and Electronis Engineers* (IEEE) berdasarkan standar teknis perangkat bernomor 802.11b, 802.11a dan 802.16. Perangkat *wifi* sebenarnya tidak hanya mampu bekerja di jaringan WLAN, tetapi juga di jaringan *Wireless Metropolitan Area Network* (WMAN). Karena perangkat dengan standar teknis 802.11b diperuntukkan bagi perangkat WLAN yang digunakan di frekuensi 2,4 GHz atau yang lazim disebut frekuensi ISM (*Industrial, Scientific* dan *Medical*). Sedang untuk perangkat yang berstandar teknis 802.11a dan 802.16 diperuntukkan bagi perangkat WMAN atau juga disebut *Wi-Max*, yang bekerja di sekitar pita frekuensi 5 GHz. (Sony Bahagia Sinaga ; 2012 : 47-48)

II.6. *GZip Stream*

GZIP merupakan algoritma standar yang telah dipakai banyak pihak untuk melakukan proses kompresi. *GZIP* atau *GNU Zip* pertama kali diciptakan oleh *Jean-loup Gailly* dan *Mark Adler* untuk *Listing* dekompresi pada tanggal 31

Oktober 1992. Algoritma *GZIP* sendiri telah menjadi metode kompresi default di sistem operasi *Linux* dan *BeOS*, sehingga kehandalannya dapat dipertanggung jawabkan. Secara teoritis, algoritma *GZIP* sendiri merupakan pengembangan dari algoritma *DEFLATE* yang merupakan kombinasi dari algoritma *LZ77* dan *Huffman Coding*, tetapi dengan lisensi *open source* sehingga dapat diterapkan oleh banyak pihak dengan lebih bebas.

Sebuah dokumen hasil kompresi *GZIP* memiliki format sebagai berikut :

1. Sebuah 10-*byte header*, berisi *magic number*, versi dan sebuah *timestamp*.
2. Tambahan *extra headers*, seperti nama asli *file*.
3. Bagian utama berisi hasil kompresi.
4. Sebuah 8-*byte footer*, berisi CRC-32 *checksum* dan panjang dokumen asli.

Class GZIP Stream merupakan *Class* dalam lingkup *.NET framework 2.0* yang mengimplementasikan algoritma kompresi *GZIP* dalam sebuah *stream*. *Stream* merupakan abstraksi dari *byte* sekuensial, seperti *file, input/output device,* atau TCP/IP *socket*. *Class Stream* dan turunannya memandang berbagai tipe *input/output*, dan memisahkan detailnya dari sistem operasi dan perangkat yang berelasi dari sudut pandang programmer. Dari *Class Stream*, selanjutnya diturunkan ke *namespace System.IO.Compression* yang terdiri dari *Class DEFLATE* yang mengimplementasikan algoritma *DEFLATE*, dan *Class GZIP Stream* sendiri.(Soetam Rizky Wicaksono ; 2010 : 88-89)

II.7. *Java*

Java memiliki cara kerja yang unik dibandingkan dengan bahasa pemrograman lainnya yaitu bahasa pemrograman *java* bekerja mengunakan

interpreter dan juga *compiler* dalam proses pembuatan program, *interpreter java* dikenal sebagai pemrograman *bytecode* yaitu dengan cara kerja mengubah paket *class* pada *java* dengan *extensi .java* menjadi *.class*, hal ini dikenal sebagai *class bytecode*, yaitu *class* yang dihasilkan agar program dapat dijalankan pada semua jenis perangkat dan juga *platform*, sehingga program *java* cukup ditulis sekali namun mampu bekerja pada jenis lingkungan yang berbeda. (Defni, Indri Rahmayun ; 2014 : 64)

II.8. *NetBeans*

NetBeans merupakan salah satu *IDE* yang dikembangkan dengan bahasa pemrograman *java*. *NetBeans* mempunyai lingkup pemrograman java terintegrasi dalam suatu perangkat lunak yang di dalamnya menyediakan pembangunan pemrograman *GUI, text editor, compiler*, dan *interpreter*. *NetBeans* adalah sebuah perangkat lunak *open source* sehingga dapat digunakan secara gratis untuk keperluan komersial maupun nonkomersial yang didukung oleh *Sun Microsystem*. (Atik Rusmayanti ; 2013 : 2-3)

II.9. Data *Flow* Diagram (DFD)

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 adalah salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

DFD ini merupakan alat perancangan sistem 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)

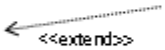
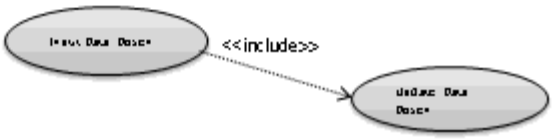
II.10. UML (*Unified Modelling Language*)

UML *(Unified Modeling Language)* adalah sebuah bahasa yang berdasarkan grafik / gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO (*Object-Oriented)*. UML sendiri juga memberikan standar penulisan sebuah sistem *blueprint,* yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database,* dan komponen-komponen yang diperlukan dalam sistem *software*. UML merupakan kombinasi terbaik dari kelebihan-kelebihan data *modeling concept* (model *entitity relationship* diagram), *business modeling (work flow), obejct modeling dan component modeling*. (M. Zulhamsyah ; 2014 : 8)

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

1. *Use Case Diagram*

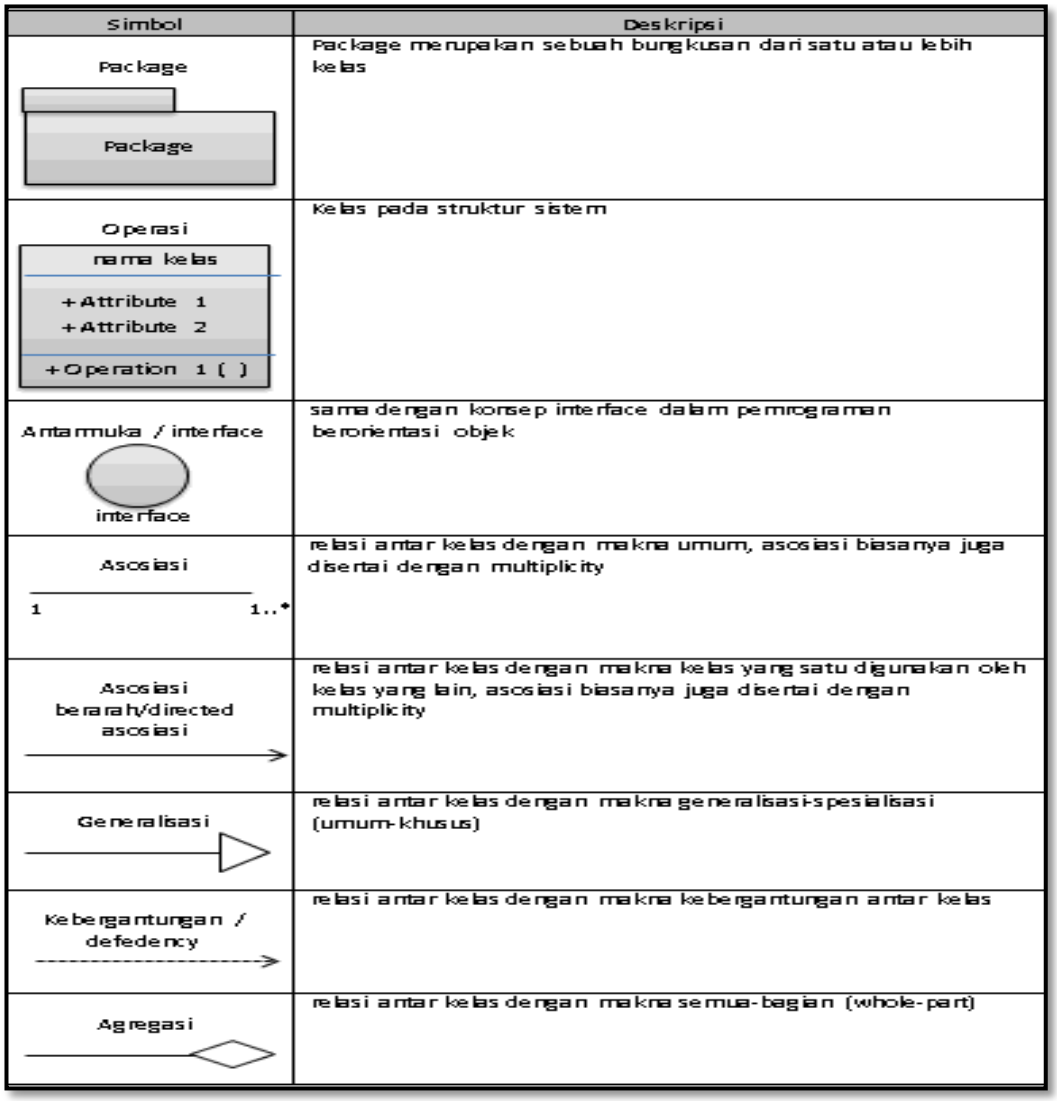
Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang atau 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 *client*, 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. (*Muhammad Yudhi Azriansyah Lubis ; 2015 : 2*)

Simbol	Deskripsi
Use Case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama use case
Aktor  nama aktor	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi / association	komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
Extend 	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, arah panah menunjukan pada use case yang dituju contoh : 
Include 	relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini. Ada dua sudut pandang yang cukup besar mengenai include di use case, include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, contoh : 

Gambar II.1. ***Use Case Diagram***
Sumber : (*Muhammad Yudhi Azriansyah Lubis ; 2015 : 2)*

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 :



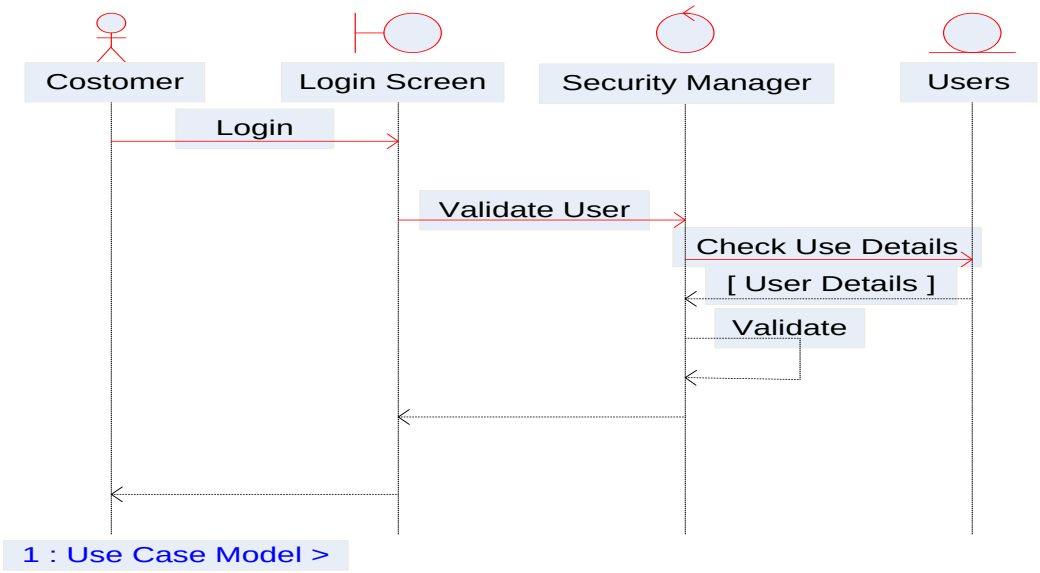
Simbol	Deskripsi
Package Package	Package merupakan sebuah bungkusan dari satu atau lebih kelas
Operasi nama kelas +Attribute 1 +Attribute 2 +Operation 1 ( )	Kelas pada struktur sistem
Antarmuka / interface interface	sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi 1 1..*	relasi 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 : (*Muhammad Yudhi Azriansyah Lubis ; 2015 : 3)*

3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan atau 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 pendefenisian *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 : (*Muhammad Yudhi Azriansyah Lubis ; 2015 : 3)*

4. *Activity Diagram*

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

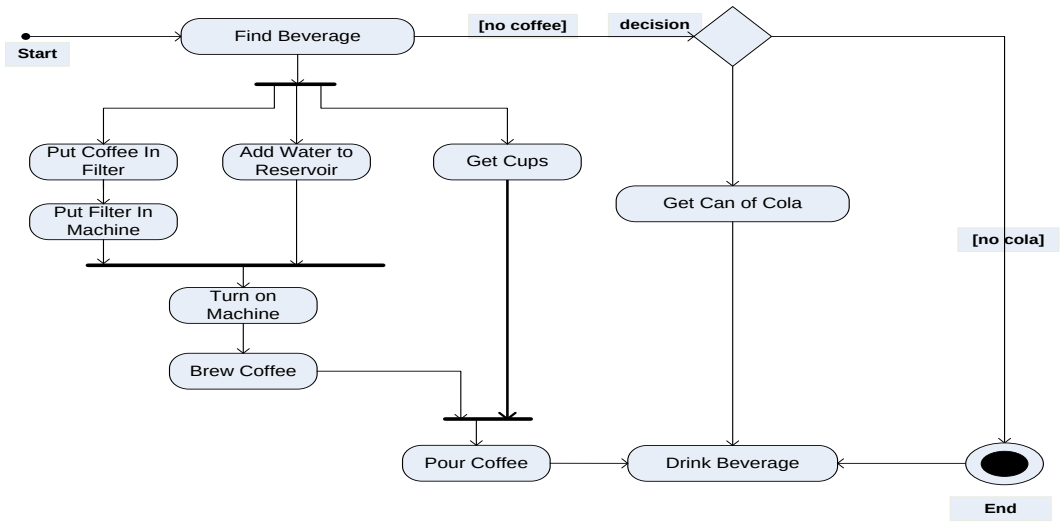
Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak

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 segi empat 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. *Activity Diagram*
Sumber : (*Muhammad Yudhi Azriansyah Lubis ; 2015 : 4)*