BAB II
LANDASAN TEORI

Pada bab ini akan diuraikan tentang teori-teori yang melandasi penulisan Laporan Penelitian ini.

2.1. Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. [1]

Pengertian sistem menurut Wikipedia indonesia adalah sistem berasal dari bahasa Latin (*systēma*) dan bahasa Yunani (*sustēma*) adalah suatu kesatuan yang terdiri komponen atau elemen yang dihubungkan bersama untuk memudahkan aliran informasi, materi atau energi. Istilah ini sering dipergunakan untuk menggambarkan suatu set entitas yang berinteraksi, di mana suatu model matematika seringkali bisa dibuat. [2]

"Sistem adalah sekumpulan bagian yang mempunyai kaitan satu sama lain yang bersama-sama beraksi menurut pola tertentu terhadap masukan dengan tujuan untuk menghasilkan pola keikhlasan". [3]

2.2. Pengertian Persediaan (*Inventory*)

"Persediaan adalah barang-barang yang biasanya dapat dijumpai di gudang tertutup, lapangan, gudang terbuka, atau tempat-tempat penyimpanan lain, baik berupa bahan baku, barang setengah jadi, barang jadi, barang-barang untuk keperluan operasi, atau barang-barang unutk keperluan suatu proyek".[4]

"Persediaan adalah suatu aktiva yang meliputi barang-barang milik perusahaan dengan maksud untuk dijual dalam suatu periode usaha yang normal". [5]

Jadi persediaan merupakan sejumlah barang yang disediakan untuk memenuhi permintaan dari pelanggan. Dalam perusahaan perdagangan pada dasarnya hanya ada satu golongan *inventory* (persediaan), yang mempunyai sifat perputaran yang sama yaitu yang disebut "*Merchandise Inventory*" (persediaan barang dagangan). Persediaan ini merupakan persediaan barang yang selalu dalam perputaran, yang selalu dibeli dan dijual, yang tidak mengalami proses lebih lanjut didalam perusahaan tersebut yang mengakibatkan perubahan bentuk dari barang yang bersangkutan. [5]

2.2.1. Jenis-jenis Persediaan

Jenis-jenis persediaan dalam suatu perusahaan menurut fungsinya dapat dibedakan atas :

1. *Bath Stock/Lot Size Inventory* adalah persediaan yang diadakan karena kita membeli atau membuat bahan-bahan atau barang-barang dalam jumlah yang lebih besar daripada jumlah yang dibutuhkan pada saat itu. Keuntungannya adalah:
   a. Potongan harga pada harga pembelian.
   b. Efisiensi produksi.
   c. Penghematan biaya angkutan.
2. *Fluctuation Stock* adalah persediaan yang diadakan untuk menghadapi fluktuasi permintaan konsumen yang tidak dapat diramalkan.
3. *Anticipation Stock* adalah persediaan yang diadakan untuk menghadapi fluktuasi permintaan yang dapat diramalkan, berdasarkan pola musiman yang terdapat dalam satu tahun dan untuk menghadapi penggunaan, penjualan, atau permintaan yang meningkat.

Setiap jenis persediaan memiliki karakteristik tersendiri dan cara pengelolan yang berbeda, sehingga dapat dilihat dari jenis dan posisi barang. Persediaan menurut jenis dan posisi barang dapat dibedakan menjadi beberapa jenis:

1. Persediaan bahan mentah (*raw material*) yaitu persediaan barang-barang berwujud, seperti besi, kayu, serta komponen-komponen lain yang dugunakan dalam proses produksi.
2. Persediaan bagian produk atau komponen-komponen rakitan (*purchased parts/components*), yaitu persediaan barang-barang yang terdiri dari komponen-komponen yang diperoleh dari perusahan lain yang secara langsung dapat dirakit menjadi suatu produk.
3. Persediaan bahan pembantu atau penolong (*supplies*), yaitu persediaan barang-barang yang diperlukan dalam proses produksi, tetapi bukan merupakan bagian atau komponen barang jadi.
4. Persediaan barang dalam proses (*work in process*), yaitu persediaan barang-barang yang merupakan keluaran dari tiap-tiap bagian dalam proses produksi

atau yang telah diolah menjadi suatu bentuk, tetapi masih perlu diproses lebih lanjut menjadi barang jadi.

5. Persediaan barang jadi (*finished goods*), yaitu persediaan barang-barang yang telah selesai diproses atau diolah dalam pabrik dan siap dijual atau dikirim kepada pelanggan. [6]

2.2.2. Pengelolaan Persediaan

"Prinsip pengelolaan persediaan yaitu penentuan jumlah dan jenis barang yang disimpan dalam persediaan haruslah sedemikian rupa sehingga produksi dan operasi perusahaan tidak terganggu, tetapi dilain pihak harus bisa menjaga agar biaya investasi yang ditimbulkan dari penyediaan barang tersebut seminimal mungkin". [4]

2.3. FIFO (*First in First Out*)

FIFO adalah akronim untuk *First In, First Out* (pertama masuk, pertama keluar), sebuah abstraksi yang berhubungan dengan cara mengatur dan memanipulasi data relatif terhadap waktu dan prioritas. Ungkapan ini menggambarkan prinsip teknik pengolahan antrian atau melayani permintaan yang saling bertentangan dengan proses pemesanan berdasarkan perilaku *first-come, first-served* (FCFS): di mana orang-orang meninggalkan antrian dalam urutan mereka tiba, atau menunggu giliran satu di sebuah sinyal kontrol lalu lintas.

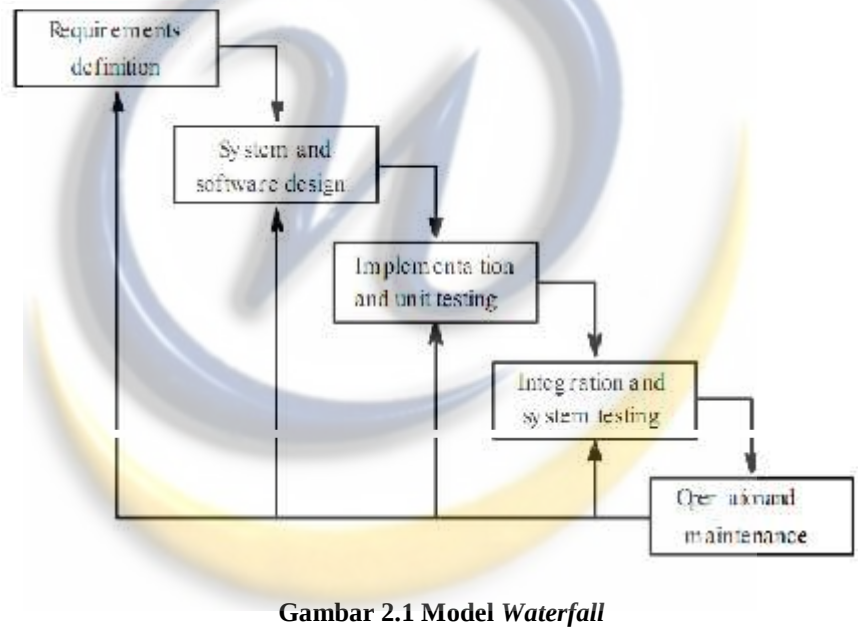
FCFS juga merupakan jargon istilah untuk sistem operasi penjadwalan algoritma FIFO, yang memberikan setiap proses CPU waktu sesuai dengan urutan mereka datang. Dalam arti yang lebih luas, abstraksi LIFO, atau *Last-In-First-Out* adalah kebalikan dari abstraksi organisasi FIFO. Bedanya mungkin adalah yang paling jelas dengan mempertimbangkan sinonim yang kurang umum digunakan dari LIFO, FILO (berarti *First-In-Last-Out*). Pada intinya, keduanya adalah kasus khusus dari daftar yang lebih umum (yang dapat diakses di mana saja). Perbedaannya adalah tidak ada dalam daftar (data), tetapi dalam aturan untuk mengakses konten. Satu sub-tipe menambah satu ujung, dan melepaskan dari yang lain, sebaliknya mengambil dan menempatkan sesuatu hanya pada salah satu ujungnya.

Variasi bahasa populer pada pendekatan *ad-hoc* untuk menghapus item dari antrian telah diciptakan dengan nama OFFO, yang merupakan singkatan *On-*

Fire-First-Out. Antrian prioritas adalah variasi pada antrian yang tidak memenuhi syarat untuk nama FIFO, karena tidak secara akurat menggambarkan perilaku struktur data. Teori antrian mencakup konsep yang lebih umum dari antrian, serta interaksi antara ketat-antrian FIFO. [7]

2.4. Pembangunan Sistem dengan *Waterfall*

Dalam pembangunan sistem ini digunakan model pengembangan *waterfall*. Inti dari metode *waterfall* adalah pengerjaan dari suatu sistem dilakukan secara berurutan atau secara linear. Jadi jika langkah satu belum dikerjakan maka tidak akan bisa melakukan pengerjaan langkah 2, 3 dan seterusnya. Secara otomatis tahapan ke-3 akan bisa dilakukan jika tahap ke-1 dan ke-2 sudah dilakukan. [8]



Gambar 2.1 Model *Waterfall*

Secara garis besar metode *waterfall* mempunyai langkah-langkah sebagai berikut :

1. ***Requirements Definition***

   Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau *study literatur*. Seorang sistem analis akan menggali informasi sebanyak-banyaknya dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh *user* tersebut. Tahapan ini akan menghasilkan dokumen *user requirement* atau bisa dikatakan

sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan sistem. Dokumen ini lah yang akan menjadi acuan sistem analis untuk menterjemahkan ke dalam bahasa pemprogram.

2. *System and Software Design*

Proses desain akan menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat coding. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi *interface*, dan *detail* (algoritma) prosedural. Tahapan ini akan menghasilkan dokumen yang disebut *software requirement*. Dokumen inilah yang akan digunakan *proggrammer* untuk melakukan aktivitas pembuatan sistemnya.

3. *Implementaion and Unit Testing*

Coding merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer. Dilakukan oleh *programmer* yang akan meterjemahkan transaksi yang diminta oleh *user*. Tahapan ini lah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap sistem yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap sistem tersebut dan kemudian bisa diperbaiki.

4. *Integration and System Testing*

Tahapan ini bisa dikatakan *final* dalam pembuatan sebuah sistem. Setelah melakukan analisa, design dan pengkodean maka sistem yang sudah jadi akan digunakan oleh *user*.

5. *Operation and Maintenance*

Perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional. **[8]**

2.5. Seputar Tentang Visual Basic 6.0

Visual Basic adalah salah satu *development tools* untuk membangun aplikasi dalam lingkungan windows. Dengan pendekatan *visual* digunakan untuk merancang *user interface* dalam bentuk *form*, sedangkan untuk kodingnya menggunakan bahasa *basic* yang cenderung mudah dipelajari. Umumnya pembuatan suatu aplikasi dimulai dari perancangan dan pembuatan *user interface*, mengatur *property* dari tiap objek yang digunakan, kemudian melakukan pengkodean.

IDE (*Integrated Development Environtment*) Visual Basic merupakan Lingkungan Pengembangan Terpadu bagi *programmer* dalam mengembangkan aplikasinya, karena dengan menggunakan IDE *programmer* dapat membuat *User Interface*, melakukan koding, melakukan *testing* dan *debugging*, kemudian mengkompilasi program menjadi *executable*.

Visual Basic merupakan bahasa pemrograman tercepat dan termudah untuk membuat suatu aplikasi dalam Micosoft Windows.

2.5.1. Menjalankan Visual Basic 6.0

Dalam pengembangan aplikasi, Visual Basic menggunakan pendekatan *Visual* untuk merancang *user interface* dalam bentuk *form*, sedangkan untuk kodingnya menggunakan dialek bahasa *Basic* yang cenderung mudah dipelajari. Visual Basic telah menjadi tools yang terkenal bagi para pemula maupun para *developer*.

1.Klik menu Start
2.Klik menu Program
3.Klik Microsoft Visual Basic 6.0
4.Klik Microsoft Visual Basic 6.0.
5.Pilih Standard EXE
6.Klik Open

2.5.2. Bagian-bagian dari Jendela

- Garis Menu, digunakan untuk memilih tugas-tugas tertentu seperti menyimpan *project*, membuka *project*, dll
- *Toolbar*, digunakan untuk melakukan tugas-tugas tertentu dengan cepat.

- Jendela *Project*, jendela ini berisi gambaran dari semua modul yang terdapat dalam aplikasi Anda.
- Anda dapat menggunakan *icon Toggle Folders* untuk menampilkan modul-modul dalam jendela tersebut .
- *Form*, jendela ini merupakan tempat Anda untuk merancang *user interface* dari aplikasi Anda. Jadi jendela ini menyerupai kanvas bagi seorang pelukis.
- Jendela *Toolbox*, jendela ini berisi komponen-komponen yang dapat anda gunakan untuk mengembangkan *user interface*.
- Jendela Kode, merupakan tempat bagi anda untuk menulis koding.
- Anda dapat menampilkan jendela ini dengan menggunakan kombinasi Shift-F7.
- *TextBox* adalah kontrol yang mengandung string yang dapat diperbaiki oleh pemakai, dapat berupa satu baris tunggal, atau banyak baris.
- *Frame* adalah kontrol yang digunakan sebagai kontainer bagi kontrol lainnya.
- *CommandButton* merupakan kontrol hampir ditemukan pada setiap *form*, dan digunakan untuk membangkitkan *event* proses tertentu ketika pemakai melakukan klik padanya.

2.5.3. Langkah-langkah Menyimpan

1. Pada menu *File*, klik perintah *Save Project* kemudian akan muncul Kotak dialog *File Project* Melalui kotak dialog tersebut dapat menyimpan program Visual Basic tersebut.
2. Pilih direktori kerja anda misalkan f:\Bab1\Lat01 dengan mengklik pada kontrol *combo box*.
3. Menyimpan Program Visual Basic dengan cara mengetik nama *file* “Lat01” pada kontrol *Text Box File name* dan menekan tombol *Save*.

2.6. Variabel dan Data

2.6.1. Variabel

Variabel adalah tempat untuk menyimpan nilai-nilai atau data-data secara sementara pada aplikasi Visual Basic. Untuk mendeklarasikan sebuah variabel dapat digunakan kata Dim seperti pada sintak dibawah ini :

Dim nama_variable as Tipe_data

Ketentuan untuk membuat nama variabel :

- Harus diawali dengan huruf abjad.
- Tidak boleh menggunakan karakter khusus,misalnya : tanda titik,koma,titik koma,titik dua,tanda seru,dsb.
- Maksimal terdiri dari 255 karakter.
- Tidak boleh menggunakan nama variabel yang sama dalam satu bagian.
- Tidak boleh mengacu pada nama prosedur *form*,dsb.

Variabel dibagi menjadi 2 yaitu

1. Variabel Global

Dalam Visual Basic variabel global merupakan variabel yang dideklarasikan dengan menggunakan kata kunci Public pada sebuah modul BAS. Sifat utama dari variabel global adalah variabel yang dapat dibaca dan dimodifikasi dari segala tempat pada suatu program atau aplikasi. ‘ dalam module BAS Public InvoiceCount as Long ‘Ini adalah variabel global. Variabel level modul.

- Variabel pada *level* modul dideklarasikan dengan menggunakan kata kunci Dim atau Private pada sebuah modul (BAS Module, form Module, Class module, dll).
- Variabel ini dapat dibaca dan dimodifikasi dalam modul tempat deklarasi dan tidak dapat diakses dari luar module.
- Variabel ini biasanya digunakan sebagai pembagian data (data *sharing*) antara prosedur dalam satu modul.

‘ Dideklasikan dalam suatu module Private LoginTime As Date
‘*variable private pada level module*
Dim LoginPassword As String

Selain itu variabel ini dapat dideklarasikan dengan menggunakan kata kunci *public*. Bedanya variabel ini dapat diaskes dari luar *module* seperti contoh berikut :

‘ Dalam *module* Form1
Public CustomerName As String

‘Sebuah Properti yang *public*

‘ Dari luar module form1

FrmTest.CustomerName = “johan jm

2.6.2. Data

Data adalah nilai mentah yang tidak memiliki arti jika berdiri sendiri. Penggunaan tipe data suatu variabel digunakan untuk mengatur jenis data yang dibutuhkan untuk menyimpan nilai-nilai di dalam memori komputer. Visual Basic mempunyai sejumlah tipe data. Tipe data tersebut antara lain tipe data sederhana (*integer, double, string,* dll) dan tipe data *array*.

Penjelasan tipe data

- Tipe data Numerik terdiri dari *integer, long, single*.
- Jika suatu variabel selalu merupakan bilangan bulat maka variabel tersebut dapat dideklarasikan dengan tipe data *Integer* atau *long* (seperti angka 12, 14, dll). Sedangkan jika variabel merupakan bilangan desimal maka dapat digunakan tipe data *Currency*, *Single* atau *Double* (seperti angka 12,23123). *Currency* hanya mampu menyimpan data sampai 4 angka dibelakang koma.
- Tipe data Byte Jika suatu varibel berisi data biner atau heksadesimal, maka variabel tersebut dapat dideklarasikan sebagai tipe data *byte*. Tipe data *Byte* mampu menyimpan nilai antara 0 sam pai 255.
- Tipe data *String* digunakan untuk menyimpan data berupa kalimat, bukan tipe data angka.
- Tipe data *Boolean* digunakan untuk menyimpan informasi *true/false*, ya/tidak, atau benar/salah.
- Tipe data *Date* digunakan untuk menyimpan informasi tanggal dan jam. [9]

2.7. UML

Unified Modeling Language (UML) adalah bahasa pemodelan standar untuk perangkat lunak dan pengembangan system [10]. UML (*Unified Modeling Language)* adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau

dihasilkan dalam suatu proses pembuatan perangkat lunak. *Artifact* dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan system non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

2.7.1. Bagian-Bagian UML

Bagian-bagian utama dari UML adalah *view, diagram, model element,* dan *general mechanism.*

a. *View*

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram.

b. *Use case view*

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa *user* atau sistem lainnya. *Use case view* digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. Use case view digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

c. *Logical view*

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object* dan *relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *Logical view* digambarkan dalam *class diagrams* untuk struktur statis dan dalam *state, sequence, collaboration,* dan *activity diagram* untuk model dinamisnya. *Logical view* digunakan untuk perancang (*designer*) dan pengembang (*developer*).

d. *Component view*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari *code module* diperlihatkan dengan struktur dan

ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. *Component view* digunakan untuk pengembang (*developer*).

e. *Concurrency view*

Membagi sistem ke dalam proses dan prosesor . *Concurrency view* digambarkan dalam diagram dinamis (*state, sequence, collaboration,* dan *activity diagrams*) dan diagram implementasi (*component* dan *deployment diagrams*) serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

f. *Deployment view*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *Deployment view* digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

g. Diagram

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

1. *Use Case Diagram*

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.

2. ***Class Diagram***

Class adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class diagram* dapat memberikan pandangan global atas sebuah system. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan yang lainnya. Sebuah sistem

biasanya mempunyai beberapa *class diagram*. *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem.

3. *Component Diagram*

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi *software* dari sebuah atau lebih *class*. Komponen dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga *component diagram* merepresentasikan dunia riil yaitu *component software* yang mengandung *component, interface dan relationship*.

4. *Deployment Diagram*

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes, executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

5. *State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

6. *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antar *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

7. *Collaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequence diagrams*. Dalam menunjukkan pertukaran pesan, *collaboration diagrams* menggambarkan *object* dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence diagrams*, tapi jika penekanannya pada konteks gunakan *collaboration diagram*.

8. *Activity Diagram*

Menggambarkan rangkaian aliran dari aktifitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

2.7.2. Tujuan Penggunaan UML

Tujuan dari penggunaan UML adalah sebagai berikut :

1. Memberikan bahasa pemodelan yang bebas dari berbagai bahas pemrograman dan proses rekayasa.
2. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan.
3. Memberikan model yang siap pakai, bahasa pemodelan *visual* yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
4. UML bisa juga berfungsi sebagai sebuah (*blue print*) cetak biru karena sangat lengkap dan detail. Dengan cetak biru ini maka akan bisa diketahui informasi secara detail tentang *coding* program atau bahkan membaca program dan menginterpretasikan kembali ke dalam bentuk diagram (*reserve enginering*). **[11]**