BAB II

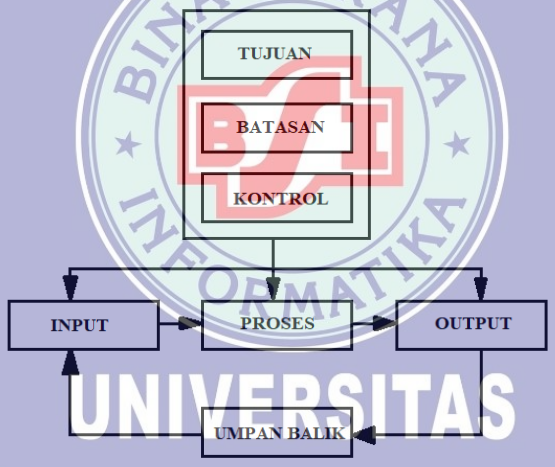
LANDASAN TEORI

2.1. Konsep Dasar Sistem

2.1.1. Konsep Dasar Sistem

Menurut (Kristanto, 2018) mengemukakan bahwa, “Suatu sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan,berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu”.

1. Elemen-elemen Sistem:



Sumber : (Kristanto, 2018)

Gambar II. 1 Elemen Sistem

a. Tujuan Sistem

Tujuan sistem merupakan tujuan dibuatnya sistem, dapat berupa tujuan organisasi, kebutuhan organisasi, permasalahan dalam organisasi maupun prosedur untuk mencapai organisasi.

b. Batasam Sistem

Batasan sistem merupakan sesuatu yang membetasi sistem, dapat berupa peraturan-peraturan, biaya, sarana prasarana maupun angota dalam organisasi tersebut.

c. Kontrol Sistem

Kontrol sistem merupakan pengawasan terhadap pelaksanaan pencapaian tujuan dari sistem.

d. *Input*

Elemen dari sistem yang bertugas untuk menerima masukan data.

e. Proses

Elemen sistem yang bertugas mengolah atau memproses seluruh masukan data menjadi sebuah informasi.

f. *Output*

Output merupakan hasil dari input yang telah diproses.

g. Umpan Balik

Umpan balik merupakan elemen dari sistem yang bertugas mengevaluasi *output*.

2. Klasifikasi Sistem

a. Sistem Abstrak dan Sistem Fisik

Sistem abstrak merupakan sistem yang berupa pemikiran atau ide-ide. Sedangkan sistem fisik merupakan sistem yang bias dilihat secara kasat mata seperti sistem komputer.

b. Sistem Alamiah dan Sistem Buatan

Sistem alamiah merupakan sistem yang diperngaruhi oleh alam. Sedangkan sistem buatan merupakan sistem yang dirancang dan dibuat oleh manusia.

c. Sistem Tertutup dan Sistem Terbuka

Sistem tertutup merupakan sistem yang tidak berhubungan dengan bagian luar sistem dan biasanya tidak terpengaruh oleh kondisi di luar sistem. Sedangkan sistem terbuka merupakan sistem yang berhubungan dengan bagian luar sistem.

3. Analisi Sistem

Merupakan analisa kebutuhan sistem yang digunakan sebagai acuan dalam perencanaan sistem.

2.1.2. NetBeans

Menurut Sugiarti dalam (Sugiarti, 2018) mengemukakan bahwa, “NetBeans merupakan IDE yang ditujukan untuk memudahkan pemrograman Java”.

Dengan kata lain, NetBeans merupakan sebuah *software* yang digunakan untuk memepermudah dalam membuat pemrograman Java. Pemrograman dalam NetBeans dilakukan berbasis visual. Untuk membuat *user-interface* tidak perlu membuat teks program secara manual, cukup klik pada *component-pallete* maka teks program akan dihasilkan secara otomatis.

Langkah-langkah dalam menggunakan NetBeans untuk menjalankan Java (Sugiarti, 2018):

1. Pastikan dulu Anda sudah menginstal Java di komputer.
2. Kemudian unduh juga NetBeans.
3. Install NetBeans di computer Anda.
4. Setelah kedua *software* siap, jalankan NetBeans.

2.1.3. Sistem Basis Data

Menurut (Fathansyah, 2018) menyimpulkan bahwa : "Sistem Basis Data merupakan sistem yang terdiri atas kumpulan tabel data yang saling berhubungan (dalam sebuah basis data di sebuah sistem komputer) dan sekumpulan program (yang biasa disebut DBMS/*Data Base Management System*)".

Basis data (*database*) sendiri merupakan kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diubah menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi.

Menurut (Kristanto, 2018), keuntungan penggunaan DBMS (*Data Base Management System*) yaitu:

1. Kebebasan data dan akses yang efisien.
2. Mereduksi waktu pengembangan aplikasi.
3. Integritas dan keamanan data.
4. Administrasi keseragaman data.
5. Akses bersama dan perbaikan dari terjadinya *crashes* (tabrakan dari proses serentak).

2.1.4. *Object Oriented Programming* (OOP)

Menurut Wardana dalam (Hardiyanto dkk., 2019), mengatakan bahwa: "Pemrograman berorientasi *object* atau yang sering disebut *Object Oriented Programming* (OOP) adalah metode pemrograman, di mana *developer* membuat dan mengelompokkan kode-kode yang berkaitan menjadi suatu *object*".

Keuntungan mengguanakan metodologi berorientasi objek merurut (Sukamto dan Shalahuddin, 2018), yaitu:

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnyayang melibatkan objek tersebut.

2. Kecepatan pengambangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisi dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.

3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat denagn dunia nyata dan adanya konsistensi pada saat pengembangan.

2.1.5. Program

Menurut Kadir dalam (Abdussomad, 2018) menyimpulkan bahwa “Program berarti kumpulan perintah yang ditujukan kepada komputer agar komputer dapat melakukan tindakan sesuai dengan yang dikehendaki oleh pembuat perintah”. Program merupakan sebuah *software* komputer yang telah dibuat untuk mempermudah satu atau beberapa pekerjaan *user*.

2.1.6. Java

Menurut (Mulyadi dan Tedyyana, 2019) mengemukakan bahwa, "Java merupakan sebuah bahasa pemrograman yang tergolong *high level language* (mudah bagi manusia untuk memahami), mengingat kata-kata atau *statemen*nya menyerupai bahasa manusia (*english*)".

Java merupakan bahasa pemrograman yang awalnya dibuat oleh James Gosling pada tahun 1995. Java mengadopsi banyak sintaks pada C dan C++ dengan sintaks model objek yang lebih sederhana. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam *p-code* (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM).

Kelebihan dari Java adalah:

1. *Multiplatform*.
2. OOP.
3. Perpustakaan Kelas Yang Lengkap.
4. Bergaya C++.

Sedangkan kelemahan dari Java yaitu:

1. Tidak kompatibel antara *platform* satu dengan *platform* lain.
2. Mudah didekompilasi.
3. Penggunaan Memori Yang Banyak.

2.1.7. *Javascript*

Javascript merupakan salah satu bahasa tertulis dalam pembuatan program atau aplikasi yang berisi perintah atau intruksi untuk mengendalikan jalannya sebuah aplikasi atau program.

Menurut Sibero dalam (Isty dan Afifah, 2018) mengemukakan bahwa, "Javascript adalah bahasa *script* (*scripting Language*) yaitu kumpulan intruksi perintah yang digunakan untuk mengendalikan beberapa bagian dari sistem operasi".

2.1.8. *JQuery*

Menurut Zaki dan Edy dalam (Tabrani, 2017) mengemukakan bahwa, "*JQuery* adalah *library JavaScript OpenSource* yang menekankan pada interaksi antara *JavaScript* dan HTML". *JQuery* merupakan bahasa pemrograman yang bekerja sama dengan *JavaScript* untuk memudahkan para *developer* dalam menggunakan dan menerapkan *JavaScript* di *website*. *JQuery* dibuat oleh John Resig, tepatnya pada tahun 2006. Sintaks yang digunakan lebih sederhana daripada *JavaScript* sehingga program terlihat lebih sederhana tapi mempunyai fitur yang lengkap.

2.1.9. *Website*

Website merupakan sebuah halaman yang memuat satu atau banyak informasi berupa teks, suara, gambar ataupun video yang dapat diakses menggunakan internet. Informasi tersubut dapat diakses oleh siapapun, kapanpun dan dimanapun selama terhubung dengan jaringan internet.

2.1.10. Internet

Menurut Irawan, dalam (Abdussomad dkk., 2016) menyatakan bahwa, "Internet merupakan kependekan dari kata "*internetwork*", yang berarti rangkaian komputer yang terhubung menjadi beberapa rangkaian jaringan." Internet merupakan sebuah jaringan yang cakupannya paling luas, yaitu mencangkup dunia. Dalam arti lain, dengan internet seluruh orang dari berbagai Negara didunia ini bisa saling terhubung

untuk berkomunikasi, bertukar informasi ataupun lainnya dengan mudah dan dalam waktu yang sangat cepat.

2.2. Teori Pendukung

2.2.1. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram merupakan sebuah diagram yang digunakan untuk mendeskripsikan hubungan Entitas (*Entity*) dan Relasi (*Relation*).

Menurut (rosa salahudin), “ERD adalah bentuk paling awal dalam melakukan perancangan nasis data relasional”.

Entity Relationship Diagram berisi komponen-komponen himpunan Entitas dan himpunan Relasiyang masing-masing dilengkapi dengan atribut-atribut yang mempresentasikan seluruh fakta dari ‘dunia nyata’ yang kita tinjau (Fathansyah, 2018).

1. Notasi-notasi simbolik di dalam ERD yang dapat digunakan adalah:
   a. Persegi panjang, menyatakan himpunan Entitas.
   b. Lingkaran/Elip, menyatakan Atribut (Atribut yang berfungsi sebagai *key* digarisbawahi).
   c. Belah Ketupat, menyatakan himpunan Relasi.
   d. Garis, sebagai penghubung antara himpunan Relasi dengan himpunan Entitas dan himpunan Entitas dengan Atributnya.
   e. Kardinalitas Relasi dapat dinyatakan dengan banyaknya garis cabang atau dengan pemakaian angka(1 dan 1 untuk relasi satu-ke-satu, dan N untuk relasi satu-ke-banyak atau N dan N untuk relasi banyak-ke-banyak).

2. Komponen-komponen yang ada dalam ERD (Fathansyah, 2018), yaitu:
    a. Entitas(*Entity*), merupakan individu yang mewakili suatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain.
    b. Atribut(*Attributes*), mendeskripsikan karakteristik dari Entitas.
    c. Relasi(*Relation*), menunjukan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.
    d. .Kardinalias, menunjukan jumlah maksimum Entitas yang dapat berelasi dengan Entitas pada himpunan Entitas yang lain.
3. Teknik *Entity Relationship* Diagram menurut (Kristanto, 2018):
    a. Analisa kebutuhan

       Pertama kali yang harus dipahami adalah kebutuhan *user*.
    b. Desain konseptual basis data

       Informasi dikumpulkan pada bagian analisis kebutuhan dan digunakan untuk mengembangkan deskripsi.
    c. Desain logika basis data

       Tahap desain logika digunakan untuk mengubah skema ERD kedalam skema data relasional.
    d. Skema perbaikan

       Himpunan relasi dalam skema basis data relasional dianalisa untuk mengidentifikasi persoalan yang akan muncul, kemudian memperbaikinya.
    e. Desain fisik basis data

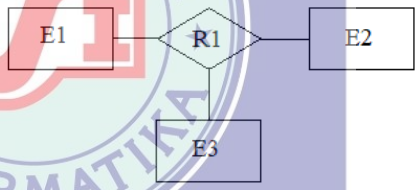
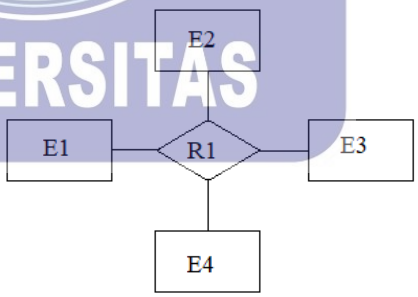
       Ditentukan masukan yang harus didukung, memperbaiki desain basis data untuk memastikan kriteria kinerja yang diinginkan sudah tercapai.
    f. Desain keamanan

Diidentifikasi kumpulan *user* yang berbeda dengan peranannya masing-masing.

ERD biasanya memiliki hubungan *binary* (satu relasi menghubungkan dua buah entitas), *ternary* (satu relasi menghubungkan tiga buah entitas) dan N-*ary* (satu relasi menghubungkan banyak entitas).

Tabel II. 1

Bentuk Hubungan Relasi

Nama	Gambar
Binary	
Ternary	
N-*ary*	

Sumber : (Sukamto dan Shalahuddin, 2018)

2.2.2. *Logical Record Structure* (LRS)

Menurut (Tabrani, 2014) mengemukakan bahwa, "*Logical Record Structure* dibentuk dengan nomor dari tipe *record.* Beberapa tipe *record* digambarkan oleh kotak empat persegi panjang dan dengan nama yang unik. Perbedaan LRS dengan E-

R diagram adalah nama tipe *record* berada diluar kotak *field* tipe *record* ditempatkan".

2.2.3. *Unified Modeling Language* (UML)

Menurut (Munawar, 2018) mengemukakan bahwa: "UML (*Unified Modeling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang beroientasi objek".

Sumber : (Munawar, 2018)

Gambar II. 2 Unsur-unsur Pembentuk UML

Pada era 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Di antaranya adalah: metodologi *Booch*, metodologi *Coad*, metodologi OOSE, metodologi OMT, metodologi *Shlaer-Mellor*, dan sebgainya.

1. Konsepsi dasar UML:

Tabel II. 2

Konsepsi 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	*Component diagram*	*Component, interface, dependancy, realization*
	Deployment view	*Deployment diagram*	*Node, component, dependency, location*
Dynamic	*State machine view*	*Statechart diagram*	*State, activity, completion transition, fork, join*
	Interaction view	*Sequence driagram*	*Interaction, object, message, activation*
		Collaboration diagram	*Collaboration, interaction, collaboration role, message*
Model management	*Model management view*	*Class diagram*	*Package, subsystem, model*
Extensibility	*All*	*All*	*Constraint, stereotype, tagged values*

Sumber : (Fathansyah, 2018)

2. Hubungan penggunaan UML

Ada 4 macam hubungan dalam penggunaan UML, yaitu:

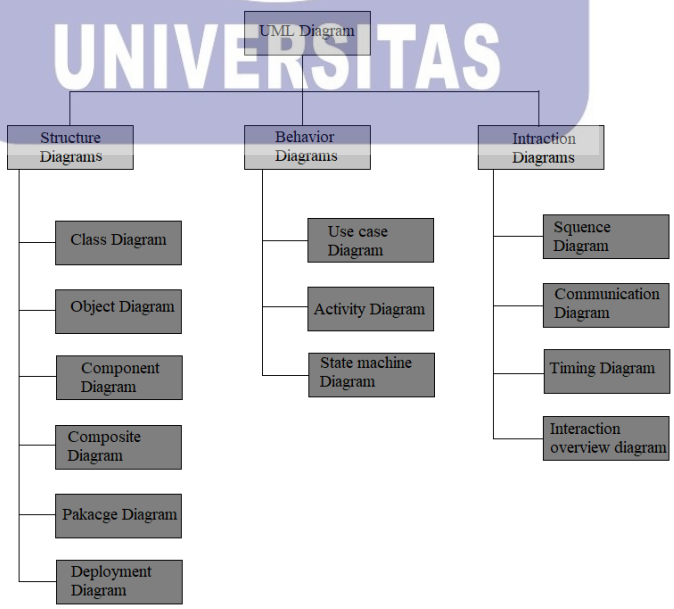
a. *Dependency*, hubungan semantik antara dua objek yang mana sebuah objek berubah mengakibatkan objek satunya akan berubah pula.

b. *Association*, hubungan antar benda secara struktural yang terhubung diantara objek dalam kesatuan objek.

c. *Generalizations*, hubungan khusus dalam objek anak yang menggantikan objek induk dan memberikan pengaruhnya dalam hal struktur dan tingkah lakunya kepada objek induk.

d. *Realizations*, hubungan semantik antarpengelompokkan yang menjamin adanya ikatan diantaranya yang diwujudkan diantara *interface* dan kelas atau *elements*, serta antara *use cases* dan *collaborations*.

3. Pembagian klasifikasi diagram pada UML:



Sumber : (Sukamto dan Shalahuddin, 2018)

Gambar II. 3 Diagram UML

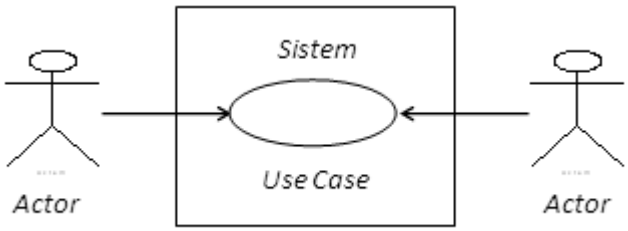
Keterangan:

a. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
b. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
c. *Intraction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

4. Diagram-diagram dalam UML:

a. *Use case* diagram

Menurut (Munawar, 2018) menyatakan bahwa, *"Use case* adalah deskripsi fungsi dari sebuah sistem dari perspektif pengguna". *Use case* adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu sistem dari sudut pandangnya.



Sumber : (Munawar, 2018)

Gambar II. 4 Use case model

Tujuan *use case* diagram secara umum adalah:

1.) Digunakan untuk mengumpulkan kebutuhan dari sebuah sistem.
2.) Untuk mendapatkan pandangan dari luar sistem.
3.) Untuk mengidentifikasi factor yang mempengaruhi sistem baik internal maupun eksternal.
4.) Untuk menunjukan interaksi dari para aktor dari sistem.

Menurut (Sukamto dan Shalahuddin, 2018), ada dua hal utama dalam *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1.) Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri.
2.) *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Ada 4 jenis relasi yang bisa timbul pada *use case* diagram:

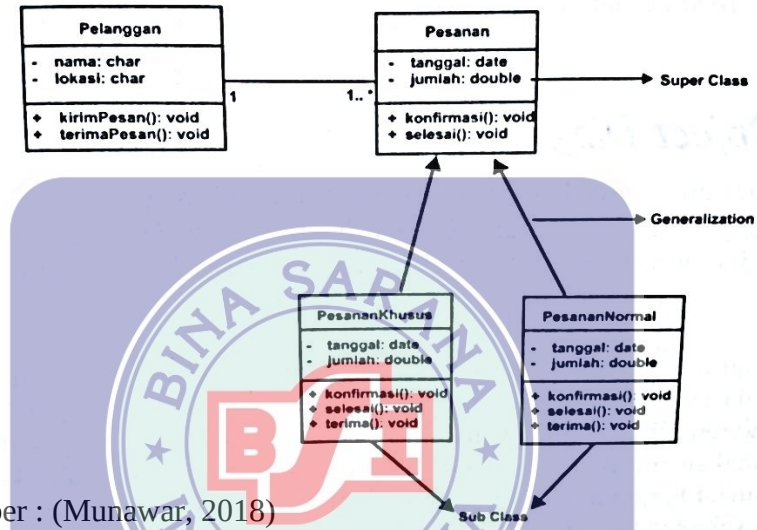
1.) *Association* antara aktor dan *use case*
2.) *Association* antara *use case*
3.) *Generalization/Inheritance* antara *use case*

    Generalization/inheritance dipakai ketika ada sebuah keadaan yang lain sendiri/perlakuan khusus (*single condition*).
4.) *Generalization/Inheritance* antara aktor

b. *Class* diagram

Class diagram menggambarkan struktur sistem dari pendefinisian *clas-clas* yang akan dibuat untuk membangun sistem.

Menurut (Munawar, 2018), "*Class* diagram adalah diagram statis. Menggambarkan atribut, *operation* dan juga *constraint* yang terjadi pada sistem".



Sumber : (Munawar, 2018)

Gambar II. 5 Contoh Class Diagram Sistem Pemesanan

Tujuan dari *class* diagram adalah memodelkan pandangan statis suatu aplikasi. Secara lebih rinci,tujuan *class* diagram yaitu:

1.) Analisi dan desain pandangan statis aplikasi.

2.) Menjelaskan tanggung jawab suatu sistem.

3.) Basis untuk diagram komponen dan penyebaran (*deployment*).

4.) *Forward and reserve engineering.*

Menurut (Sukamto dan Shalahuddin, 2018) susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1) Kelas *main*

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2) Kelas yang menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai (*user*).

3) Kelas yang diambil dari pendefinisian *use case* (*controller*)

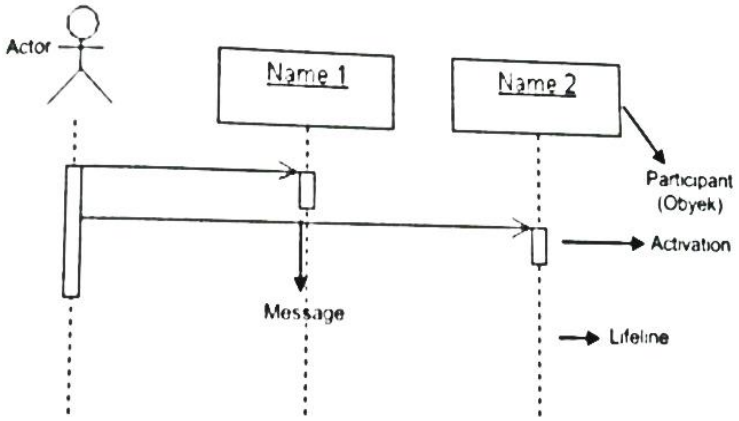
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4) Kelas yang diambil dari pendefinisian data (model)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

c. *Sequence* diagram

Sequence diagram menggambarkan *behavior* objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. *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.



Sumber :(Munawar, 2018)

Gambar II. 6 Contoh *Squence* Diagram

Menurut (Munawar, 2018) menyatakan bahwa: "tujuan *sequence* diagram secara umum ialah menunjukan urutan waktu aliran pesan dari satu objek ke objek lainnya".

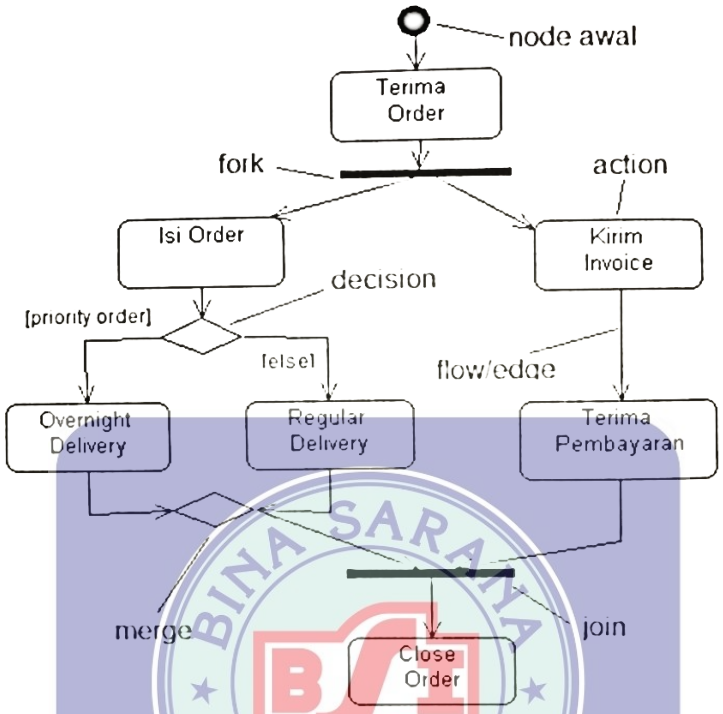
Tujuan *sequence* diagram secara lebih spesifik adalah:

1.) Model interaktiftingkat tinggi antara objek aktif dalam suatu sistem.
2.) Model interaksi antara *instance* (contoh) objek dalam kolaborasi yang merealisasikan *use case*.
3.) Model interaksi antar objek dalam kolaborasi yang mewuudkan operasi.
4.) Menunjukan model interaksi generik.

d. *Activity* diagram

Menurut (Munawar, 2018), "*Activity* diagram adalah bagian terpenting dari UML yang menggambarkan aspek dinamis dari sistem". *Activity* diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan

flowchart adalah *activity* diagram bias mendukung perilaku parallel sedangkan *flowchart* tidak bisa.



Sumber : (Munawar, 2018)

Gambar II. 7 Contoh Activity Diagram

Tujuan dari *activity* diagram adalah untuk menangkap tingkah laku dinamis dari sistem dengan cara menunjukan aliran pesan dari suatu aktivitas ke aktivitas lainnya. Secara umum tujuan *activity* diagram adalah:

1.) Menggambarkan aliran aktivitas dari sistem.

2.) Menggambarkan urutan aktivitas dari satu aktivitas ke aktivitas lainnya.

3.) Menggambarkan paralelisme, percabangan dan aliran konkuren dari sistem.

Menurut (Sukamto dan Shalahuddin, 2018), *activity* diagram digunakan untuk mendefinisikan hal-hal berikut:

1.)Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

2.)Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

3.)Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

4.)Rancangan menu yang ditampilkan pada perangkat lunak.