

BAB 5 Pemodelan Spesifikasi Kebutuhan

Perangkat Lunak

Pada peringkat teknis, pekerjaan rekayasa perangkat lunak pada umumnya dimulai dengan sejumlah pekerjaan pemodelan yang menunjukkan spesifikasi kebutuhan perangkat lunak dan representasi perancangan perangkat lunak yang akan dikembangkan. Model – model spesifikasi kebutuhan perangkat lunak merupakan sejumlah model dan merupakan representasi awal perangkat lunak.

5.1. Analisis Kebutuhan

Analisis spesifikasi kebutuhan menghasilkan spesifikasi – spesifikasi dari karakteristik – karakteristik operasional yang akan dimiliki oleh perangkat lunak yang akan dikembangkan, yang dapat mengindikasikan antarmuka perangkat lunak dengan elemen – elemen sistem yang lain dan juga menetapkan batasan – batasan yang harus dihadapi perangkat lunak. Pekerjaan – pekerjaan pemodelan spesifikasi – spesifikasi kebutuhan pada dasarnya akan menghasilkan jenis model berikut (Roger S. Pressman, 2012) :

1. Model berbasis skenario, yang menggambarkan spesifikasi kebutuhan perangkat lunak dari berbagai sudut pandang aktor perangkat lunak.
2. Model data , yang menjelaskan ranah informasi untuk permasalahan yang akan diselesaikan

3. Model berorientasi kelas , yang memperlihatkan kelas – kelas dalam konteks pemograman berorientasi objek dan dengan cara bagaimana kelas – kelas tersebut saling bekerjasama untuk mencapai sasaran – sasaran spesifikasi – spesifikasi kebutuhan perangkat lunak.
4. Model berorientasi aliran, yang menggambarkan elemen – elemen fungsional sistem perangkat lunak dan yang menggambarkan bagaimana cara mereka melakukan transformasi terhadap data, saat data yang bersangkutan melintasi sistem.
5. Model perilaku, yang menggambarkan bagaimana perangkat lunak berperilaku terhadap event – event yang datang dari luar sistem

Model – model analisis diatas memberikan informasi kepada perancang perangkat lunak untuk diterjemahkan menjadi arsitektur sistem, antarmuka – antarmuka sistem dan perancangan berperingkat komponen. Terakhir model – model spesifikasi kebutuhan memungkinkan para pengembang dan para stakeholder melakukan penilaian kualitas ketika perangkat lunak selesai dikembangkan.

5.1.1. Filosofi dan Sasaran – sasaran secara keseluruhan

Di sepanjang pemodelan analisis untuk mendapatkan spesifikasi – spesifikasi kebutuhan perangkat lunak harus berfokus pada apa yang terjadi saat pengguna berinteraksi dalam suatu keadaan tertentu , objek – objek apa yang akan dimanipulasi oleh perangkat lunak, fungsi – fungsi apa yang harus dapat dilakukan oleh sistem perangkat lunak , perilaku apa yang akan diperlihatkan oleh perangkat lunak, antarmuka apa yang

didefinisikan dan batasan – batasan apa yang diterapkan. Model – model analisis untuk mendapatkan spesifikasi – spesifikasi kebutuhan perangkat lunak pada dasarnya harus mencapai 3 sasaran utama yaitu :

1. Untuk mendeskripsikan apa yang pelanggan inginkan
2. Menetapkan dasar bagi perancangan perangkat lunak
3. Untuk mendefinisikan sejumlah kebutuhan yang dapat divalidasi saat perangkat lunak dikembangkan.

Model analisis juga pada dasarnya harus bisa menjembatani kesenjangan – kesenjangan yang terjadi diantara deskripsi – deskripsi berperingkat sistem yang mendeskripsikan fungsionalitas – fungsionalitas sistem atau bisnis secara keseluruhan saat ia dicapai dengan menerapkan solusi – solusi perangkat lunak, perangkat keras, manusia dan elemen sistem yang lainnya, dengan perancangan perangkat lunak yang akan mendeskripsikan arsitektur perangkat lunak antar muka pengguna dan struktur peringkat komponen. Merupakan hal yang penting untuk melihat bahwa semua elemen yang ada dalam model spesifikasi kebutuhan akan dapat dilacak langsung ke bagian – bagian yang ada dalam model perancangan.

5.1.2. Aturan Analisis

Arlow dan Neustadt dalam (Roger S. Pressman, 2012) menyarankan sejumlah aturan yang seharusnya dipatuhi saat analisis sistem membuat model – model analisis .

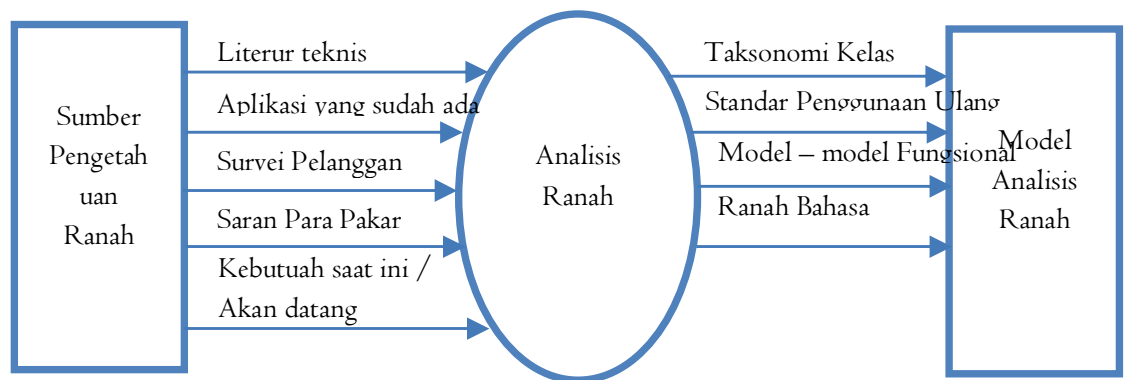
1. Model seharusnya berfokus pada kebutuhan – kebutuhan yang tampak di dalam ranah permasalahan atau ranah bisnis.
2. Masing – masing elemen yang ada di dalam model kebutuhan seharusnya menambahkan suatu pemahaman keseluruhan pada spesifikasi kebutuhan perangkat lunak dan menyediakan pandangan sekilas pada ranah informasi, fungsi dan perilaku sistem.
3. Tunda pertimbangan – pertimbangan yang berkait dengan infrastruktur dan model – model non – fungsional lainnya hingga tahap perancangan dimulai.
4. Minimalkan saling ketergantungan dalam sistem.
5. Upayakan agar analisis sistem merasa pasti bahwa model – model kebutuhan memberikan nilai tertentu pada semua stakeholder.
6. Usahan agar model sesederhana mungkin.

5.1.3. Analisis Ranah

Firesmith dalam (Roger S. Pressman, 2012) mendeskripsikan analisis ranah sebagai proses mengidentifikasi , menganalisis dan menspesifikasi kebutuhan – kebutuhan umum dari suatu ranah aplikasi

yang sifatnya spesifik, agar suatu saat dapat digunakan pada berbagai proyek lain yang ada pada ranah aplikasi yang sama yang merupakan identifikasi , analisis, dan spesifikasi kemampuan – kemampuan yang bersifat umum dan dapat digunakan ulang dalam ranah aplikasi yang sifatnya spesifik, dalam artian objek – objek , kelas – kelas , bagian – bagian dan kerangka – kerangka kerja, yang bersifat umum dari suatu aplikasi yang bersifat spesifik.

Peran seorang analis ranah adalah untuk menyingkap dan mendefinisikan pola – pola analisis , kelas – kelas analisis, dan informasi yang terkait mungkin digunakan oleh orang – orang yang bekerja dengan cara yang serupa tetapi tidak harus berada pada aplikasi yang sama. Berikut gambaran analisis ranah.



Gambar 5.1 Input , Output untuk analisis ranah

5.1.4. Pendekatan – pendekatan untuk Pemodelan Spesifikasi Kebutuhan

Ada dua pendekatan untuk memodelkan spesifikasi kebutuhan yaitu :

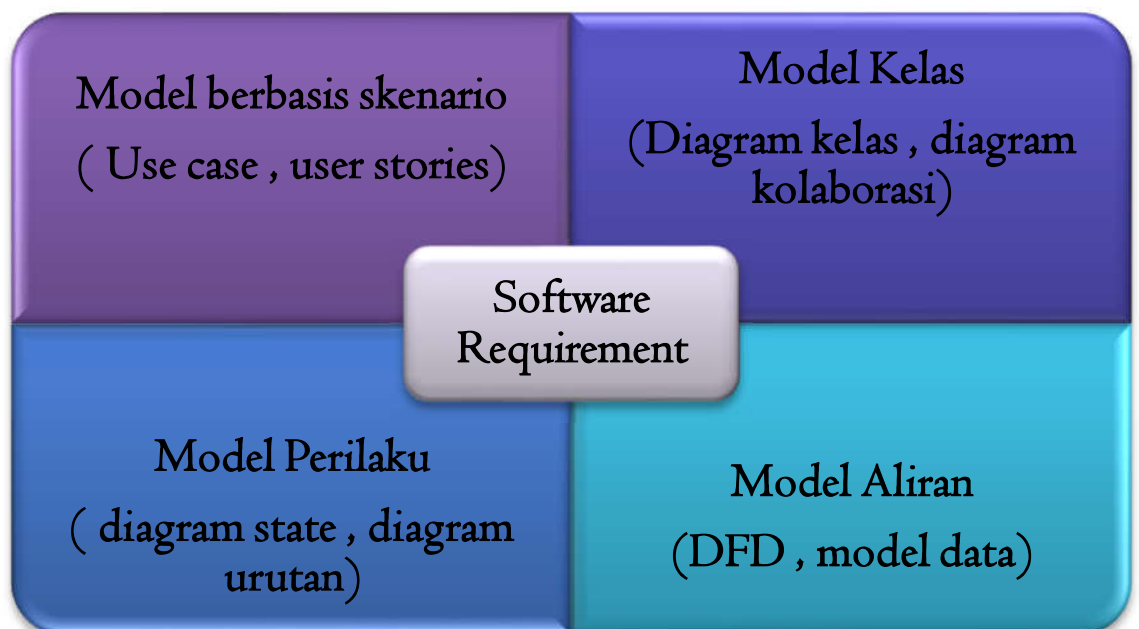
I. Analisis terstruktur

Analisis terstruktur memperlakukan data dan proses yang melakukan transformasi data tersebut sebagai entitas yang terpisah. Objek – objek data dimodelkan dengan cara mendefinisikan atribut – atributnya serta relasi – relasinya. Proses – proses yang dimanipulasi objek – objek data dimodelkan sedemikian rupa sehingga mereka dapat memperlihatkan bagaimana caranya mereka melakukan transformasi data saat objek – objek data mengalir di dalam sistem yang akan dikembangkan.

2. Analisis berorientasi objek

Pemodelan ini berfokus pada pendefinisian kelas – kelas dan cara bagaimana mereka saling bekerjasama satu dengan lainnya untuk memenuhi kebutuhan para pelanggan. UML (*Unified Modeling Process*) dan *Unified Process* merupakan perkakas dan proses yang bernuansa berorientasi objek.

Dua pendekatan tersebut memunculkan pilihan mana yang terbaik, walaupun pengembang perangkat lunak kebanyakan memilih dari salah satu pendekatan tersebut. Pengembang tidak seharusnya bertanya mana yang terbaik tetapi pengembang seharusnya berfikir bagaimana kombinasi dari representasi pendekatan tersebut yang menyediakan bagi para stakeholder model spesifikasi kebutuhan yang terbaik dan yang merupakan jembatan yang paling efektif ke tahapan perancangan lunak.



Gambar 5.2 Elemen – elemen model analisis

Masing – masing unsur model kebutuhan memperlihatkan permasalahan dari sudut pandang yang berbeda. Elemen – elemen berbasis skenario memperlihatkan bagaimana interaksi yang kelak akan terjadi antara pengguna dengan perangkat lunak yang

akan dikembangkan dan juga memperlihatkan sejumlah aktivitas berurutan yang bersifat spesifik yang terjadi saat perangkat lunak digunakan. Elemen model berbasis kelas memodelkan objek – objek yang akan dimanipulasi oleh sistem , memodelkan operasi – operasi yang akan diterapkan pada objek – objek untuk melakukan manipulasi, memodelkan relasi yang terjadi antara objek satu dengan yang lainnya, dan memodelkan kolaborasi yang terjadi di antara kelas – kelas yang telah didefinisikan. Elemen – elemen perilaku memperlihatkan bagaimana event – event eksternal melakukan perubahan pada keadaan sistem atau kelas – kelas yang ada didalamnya. Terakhir elemen – elemen berorientasi aliran memperlihatkan perangkat lunak yang bertindak sebagai perilaku transformasi informasi , memperlihatkan bagaimana objek – objek data di transformasi saat meraka mengalir melintasi fungsi yang dimiliki sistem.

5.2. Pemodelan

5.2.1. Pemodelan Berbasis Skenario

Pemodelan berbasis skenario merupakan suatu cara untuk memahami para pengguna berinteraksi dengan perangkat lunak. Dengan memahami hal tersebut tim pengembang perangkat lunak dapat memahami kebutuhan pengguna dan melakukan analisis untuk melakukan pemodelan perancangan yang baik. Pemodelan spesifikasi kebutuhan pengguna menggeunkan UML diawali dengan membentuk use

case – use case , diagram – diagram aktivitas dan diagram – diagram swimlane.

I. Membuat Use Case Awal





Cara kerja dari use case pada dasarnya adalah menangkap interaksi yang terjadi antara produsen informasi , pengguna informasi dengan perangkat lunak (Roger S. Pressman, 2012). Secara garis besar use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi – fungsi dan fitur – fitur tersebut (Rosa A. S, 2018).


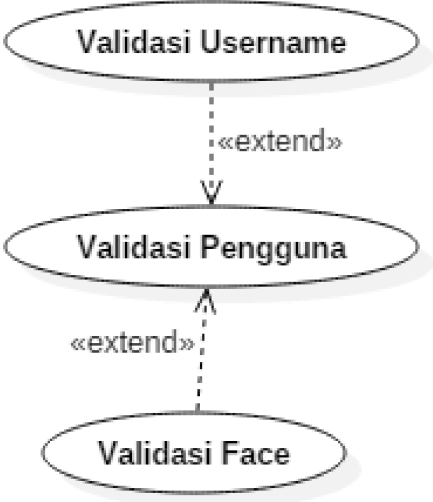
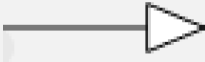
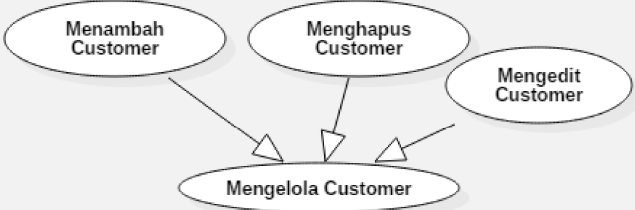
Syarat penamaan pada use case adalah nama didefinisikan sesederhana mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu (Rosa A. S, 2018) :


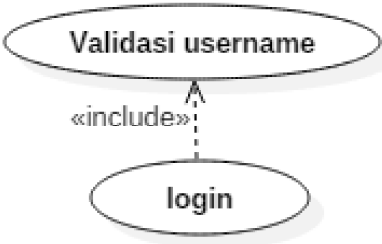

- a. Actor adalah orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sendiri , jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
- b. Use case merupakan fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antar unit atau aktor

Berikut notasi – notasi atau simbol – simbol yang ada pada diagram use case (Rosa A. S, 2018):

Tabel 5.1 Simbol pada use case

Simbol	Deskripsi
<p>Use Case</p> 	<p>Fungsionalitas yang disediakan sistem sebagai unit – unit yang saling bertukar pesan antara unit atau aktor , di nyatakan dengan menggunakan kata kerja.</p>
<p>Aktor / actor</p> 	<p>Orang , proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat sendiri , jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang</p>
<p>Asosiasi / association</p> 	<p>Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor</p> 
<p>Ekstensi /extend</p>	<p>Relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan dapat berdiri sendiri walaupun tanpa use case tambahan itu,</p>

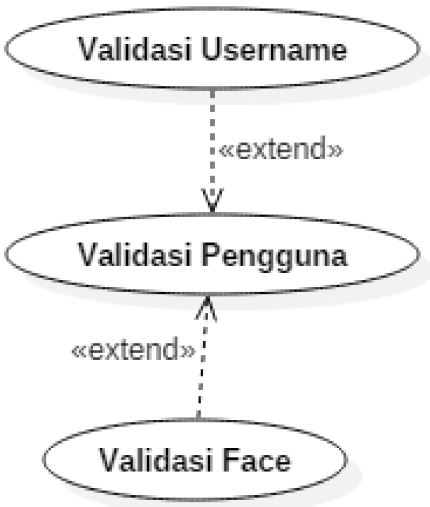
<p>«extend»</p> 	<p>seperti prinsip inheritance pada pemograman berorientasi objek.</p> 
<p>Generalisasi / generalization</p> 	<p>Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya .</p> 

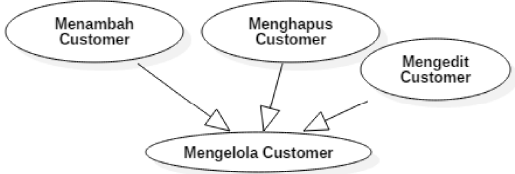
<p>Menggunakan / Include</p> <p>«include»</p> 	<p>Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya sebagai syarat dijalankan use case ini</p> <p>Ada dua perspektif dalam penggunaan include dalam use case :</p> <ol style="list-style-type: none"> Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan .  <ol style="list-style-type: none"> Include berarti use case yang ditambahkan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan. 


	Kedua perspektif dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.
--	---

Use case ini nantinya akan menjadi kelas proses pada diagram kelas sehingga perlu dipertimbangkan penamaan yang dilakukan. Berikut aturan perubahan use case yang menjadi kelas proses (Rosa A. S, 2018).

Tabel 5.2 Perubahan Use case menjadi kelas proses

Hubungan	Deskripsi
<p>Extensi / extend</p>  <pre> graph BT VP([Validasi Pengguna]) -.-> «extend» VU([Validasi Username]) VF([Validasi Face]) -.-> «extend» VP </pre>	<p>Pada hubungan extend dapat diambil use case induknya untuk</p>

	<p>dihasilkan kelas dengan metode berupa use case extend.</p> <pre> class ValidasiUsername { //atribut procedure validasiUserName() { //proses } procedure validasiFace() { //proses } </pre>
<p>Generalisasi/generalization</p> 	<p>Pada hubungan generalisasi maka dapat hanya diambil use case umumnya dijadikan kelas dengan metode berupa use case khususnya.</p>

	<pre> class MengelolaCustomer { //atribut procedure MenambahCustom er(){ //proses } procedure MenghapusCustom er() { //proses } procedure MengeditCustomer .. </pre>
<p>Use case yang berdiri sendiri</p> 	<p>Metode yang mungkin bisa ada di dalam kelas proses login adalah sebagai berikut :</p>

	<pre> class Login { //atribut procedure Login(){ //proses } procedure Logout() { //proses </pre>
--	--

Setiap use case di lengkapi dengan skenario use case . Skenario use case adalah flow jalannya proses use case dari sisi aktor dan sistem. Berikut adalah contoh format tabel dari skenario use case (Victoria University of Wellington, 2018):

Use case dapat dijelaskan baik pada tingkat abstrak (dikenal sebagai use case bisnis proses) atau pada tingkat implementasi khusus (dikenal sebagai use case sistem). Masing-masing dijelaskan secara lebih rinci di bawah ini (Inflectra, 2018):

- a. Business Use Case - Juga dikenal sebagai "use case tingkat abstrak", kasus penggunaan ini ditulis dengan cara agnostik teknologi, hanya mengacu pada proses bisnis tingkat tinggi yang dijelaskan (misalnya "pengembalian buku") dan berbagai eksternal entitas (juga dikenal sebagai aktor) yang mengambil bagian dalam proses (misalnya "peminjam", "pustakawan", dll.). Kasus penggunaan bisnis akan menentukan urutan tindakan yang perlu dilakukan oleh bisnis untuk memberikan hasil yang berarti dan dapat diobservasi ke entitas eksternal.
- b. System Use Case - Juga dikenal sebagai "use case implementasi", kasus penggunaan ini ditulis pada tingkat detail yang lebih rendah daripada use case bisnis dan mengacu pada proses khusus yang akan dilakukan oleh bagian-bagian yang berbeda dari sistem. Misalnya kasus penggunaan sistem mungkin

"mengembalikan buku ketika terlambat" dan akan menggambarkan interaksi dari berbagai aktor (peminjam, pustakawan) dengan sistem dalam melaksanakan proses end-to-end.

2. Model – model UML yang melengkapi Use Case

a. Diagram Aktivitas

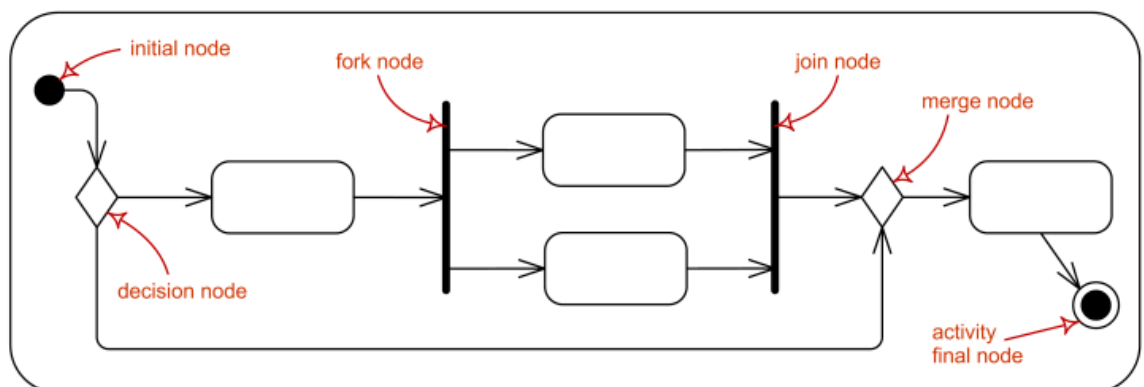
Diagram aktivitas (activity diagram) yang disediakan oleh UML melengkapi use case yang telah dibuat sebelumnya dengan memberikan representasi grafis dan aliran – aliran interaksi di dalam suatu skenario yang sifatnya spesifik (Roger S. Pressman, 2012).

Diagram aktivitas menggambarkan diagram alir (flowchart) dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Pada intinya diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dilakukan sistem. Diagram aktivitas dapat digunakan untuk mendefinisikan hal – hal berikut (Rosa A. S, 2018) :

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

- 2) Urutan atau pengelompokan tampilan dari sistem atau 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 di tampilkan pada perangkat lunak.

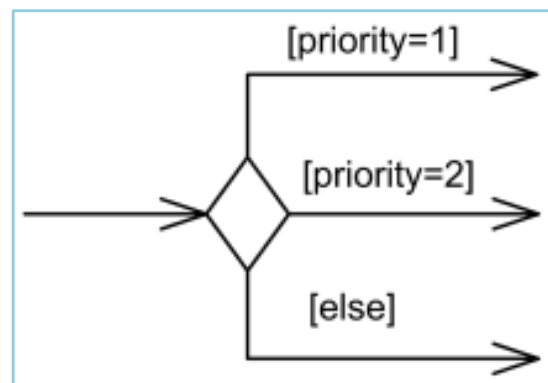
Berikut adalah notasi – notasi atau simbol – simbol yang digunakan pada diagram aktivitas (uml-diagrams.org, UML Activity Diagram Controls, 2018):



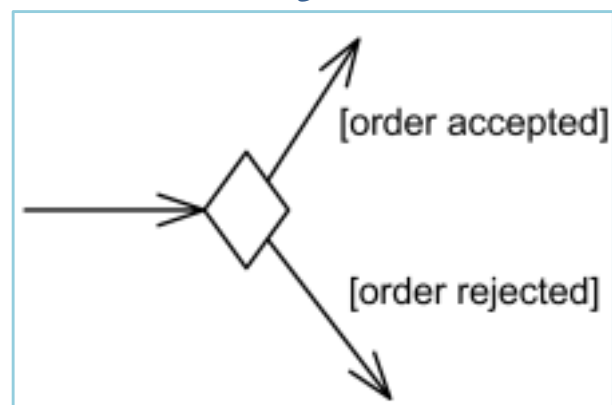
Gambar 5.4 Simbol Activity Diagram

- I) Initial Node merupakan node kontrol dimana aliran aktivitas dimulai

- 2) Decision Node merupakan node kontrol yang menerima token pada satu atau dua sisi yang masuk dan memilih satu ujung keluar dari satu atau lebih aliran keluar. Decision node diperkenalkan di UML untuk mendukung aktivitas kondisional. Contoh decision node :



Gambar 5. 5 Decision tiga cabang



Gambar 5.6 Decison node cabang tiga

- 3) Fork Node, merupakan node kontrol yang memiliki satu tepi masuk dan tepi keluar ganda dan digunakan untuk membagi aliran masuk menjadi beberapa aliran bersamaan. Fork node diperkenalkan untuk mendukung paralelisme dalam aktivitas.
- 4) Join Node, merupakan node yang memiliki beberapa tepi masuk dan satu ujung keluar dan digunakan untuk mensinkronkan aliran bersamaan yang datang. Join node diperkenalkan untuk mendukung paralelisme dalam kegiatan.
- 5) Merge Node, merupakan node kontrol yang menyatukan beberapa aliran alternatif yang masuk untuk menerima aliran keluar tunggal. Tidak ada token yang bergabung. Penggabungan tidak boleh untuk menyingkronkan aliran bersamaan.
- 6) Activity Final Node, merupakan node kontrol akhir yang menghentikan semua aliran dalam suatu aktivitas.

b. Diagram Swimlane

Diagram swimlane merupakan variasi dari diagram aktivitas yang memungkinkan untuk melihat aliran aktivitas – aktivitas yang dideskripsikan oleh use case pada saat yang sama memperlihatkan aktor mana yang melakukan kegiatan. Diagram Swimlane menggambarkan bagaimana berbagai aktor memanggil fungsi – fungsi tertentu yang sifatnya spesifik sehingga sesuai dengan kebutuhan – kebutuhan perangkat lunak. Untuk notasi dari diagram swimlane sama dengan diagram aktivitas hanya ditambahkan swimlane yaitu container yang memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Swimlane diagram, sering disebut juga “Deployment Process Map” atau “Cross Functional Flowchart” adalah sebuah diagram yang merepresentasikan flow proses yang menggambarkan interaksi dari beberapa bagian yang berbeda dan bagaimana perkembangan proses melalui beberapa phase yang berbeda (binus, 2014).

Elemen – elemen yang terdapat dalam swimlane diagram adalah : (a) Process: Aktual proses dan flow, (b) Actors:

Orang, groups, teams, yang melakukan tahapan proses. (c) Phases: Menunjukkan tahapan dari project. (d) Symbols: Simbol fisik yang digunakan menggambarkan apa yang terjadi pada setiap tahapan dari proses (binus, 2014).

Pools and Lanes



Lane
(can be represented vertical)



Multiple Lanes
(can be represented vertical)



Two Lanes in a Pool
(can be represented vertical)



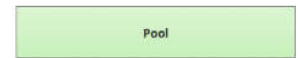
Pool
(can be represented vertical)



Pool (with MI Participant)
(can be represented vertical)



Pool (with Multiple Lanes)
(can be represented vertical)

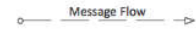


Collapsed Pool
(can be represented vertical)



Collapsed Pool (with MI Participant)
(can be represented vertical)

Flows



Message Flow



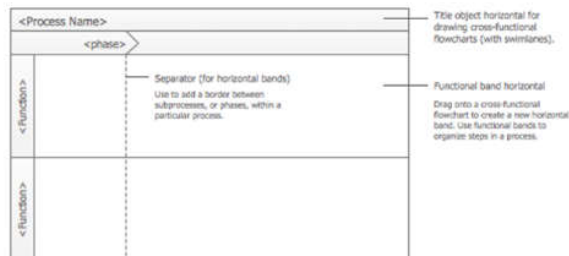
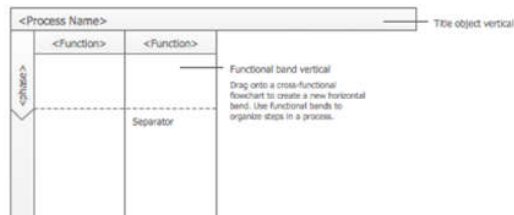
Smart Message Flow



Initiating Message Flow with Decorator



Non-Initiating Message Flow with Decorator



Vertical lane

Multiple vertical lanes

Vertical pool



Horizontal lane

Multiple horizontal lanes

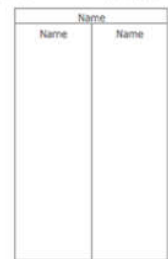
Horizontal pool



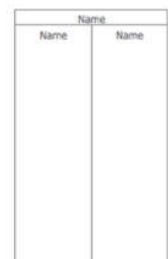
Horizontal pool with two lanes

Horizontal pool with multiple lanes

Horizontal pool with multiple lanes



Vertical pool with two lanes



Vertical pool with multiple lanes

Gambar 5.7 Business Process Elements:
Swimlanes

Sumber : (conceptdraw, 2018)

5.2.2. Pemodelan Berbasis Kelas

Pemodelan berbasis kelas pada dasarnya memperlihatkan objek – objek yang dimanipulasi oleh perangkat lunak, memperlihatkan operasi – operasi yang akan diterapkan pada objek – objek untuk menghasilkan imbas tertentu pada manipulasi objek, memperlihatkan relasi – relasi antar objek serta memperlihatkan kolaborasi – kolaborasi yang terjadi di antara kelas – kelas yang didefinisikan. Elemen – elemen model berbasis kelas mencakup didalamnya elemen – elemen kelas dan objek – objek , atribut – atribut , operasi – operasi , model tanggung jawab kelas (Class Responsibility Collaborator – CRC) , diagram – diagram koaborasi dan paket – paket (Roger S. Pressman, 2012).

Ada beberapa karakteristik objek yaitu state (merupakan suatu kondisi atau keadaan dari objek yang mungkin ada, status dari objek akan berubah setiap waktu dan ditentukan oleh sejumlah properti dan relasi dengan objek lain), Behavior atau sifat (menentukan bagaimana objek merespon permintaan dari objek lain dan melambangkan setiap hal yang dapat dilakukan, sifat ini diimplementasikan dengan sejumlah operasi untuk objek), identity (setiap objek yang ada dalam suatu sistem adalah unik).

I. Identifikasi Kelas – kelas Analisis

Untuk mengidentifikasi kelas – kelas , dapat dimulai dengan melakukan proses identifikasi kelas dengan cara memeriksa skenario penggunaan perangkat lunak yang telah dikembangkan sebelumnya sebagai bagian dari model – model kebutuhan dan kemudian melakukan pemisahan berdasarkan tata bahasa pada usecase yang dikembangkan untuk perangkat lunak. kelas – kelas dapat ditentukan dengan cara menggarisbawahi setiap kata benda dan memasukkannya ke sebuah tabel sederhana. Kelas – kelas analisis pada umumnya memanifestasikan dirinya dalam beberapa cara sebagai berikut (Roger S. Pressman, 2012):

- a. Entitas – entitas eksternal yang menghasilkan atau menggunakan informasi yang akan digunakan perangkat lunak berbasis komputer.
- b. Sesuatu yang merupakan bagian dari ranah informasi untuk permasalahan.
- c. Kehadiran atau event yang terjadi di dalam konteks operasi sistem.
- d. Peran – peran yang dimainkan oleh orang – orang yang berinteraksi dengan perangkat lunak.
- e. Unit – unit organisasional yang relevan untuk perangkat lunak yang akan dikembangkan.

- f. Tempat yang meletakkan konteks tertentu bagi sistem dan fungsi keseluruhan sistem.
- g. Struktur yang mendefinisikan suatu kelas objek – objek

Kelas – kelas analisis dapat didefinisikan ketika awal pemodelan dengan memisahkan kata benda dalam narasi pemrosesan use case yang telah didapat. Dengan mengekstraksi kata benda – kata benda tersebut maka didapat kelas potensial. Dari kelas potensial tersebut maka diidentifikasi karakteristik dari kelas potensial tersebut.

Pada dasarnya langkah pertama dari pemodelan berbasis kelas adalah membuat definisi – definisi kelas dan keputusan – keputusan yang harus dibuat.

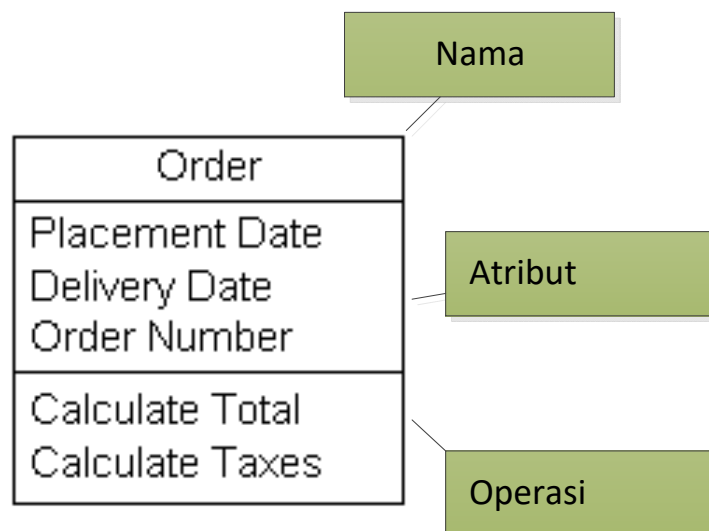
Class adalah sebuah spesifikasi yang jika di instansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, seklaigus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). (Desy, class Diagram, 2018)

Dapat disimpulkan bahwa untuk menemukan class adalah kelompokan kata benda pada dokumentasi use case kemudian kategorikan

kata benda tersebut, setiap objek yang ditemukan pada kata benda yang ditemukan merupakan contoh dari beberapa class. Class memiliki tiga area pokok yaitu nama atau stereotype , atribut dan metoda atau operasi. (Desy, class Diagram, 2018)

Atribut dan operasi dapat memiliki salah satu yaitu (Desy, class Diagram, 2018):

- a. Private , tidak dapat dipanggil dari luar class yang bersangkutan
- b. Protected , hanya dapat dipanggil oleh class yang bersangkutan dan anak – anak yang mewarisinya.
- c. Public, dapat dipanggil oleh siapa saja.



Gambar 5.8 Struktur class

2. Menentukan Atribut – atribut

Atribut mendefinisikan kelas – kelas yang telah dipilih untuk dimasukkan dalam model

spesifikasi kebutuhan perangkat lunak . Atribut – atribut merupakan sesuatu yang mendefinisikan kelas yaitu yang mengklasifikasi makna suatu kelas dalam konteks ruang permasalahan yang telah diketahui sebelumnya. Untuk mengembangkan sejumlah atribut yang bermakna untuk kelas analisis maka perlu diperlajari terlebih dahulu masing – masing use case dan memilih sesuatu yang cukup beralasan untuk dimiliki suatu kelas (Roger S. Pressman, 2012).

Sebuah class mungkin memiliki beberapa atribut, Atribut merepresentasikan beberapa properti dari sesuatu yang dimodelkan. (Desy, class Diagram, 2018)

3. Mengdefinisikan Operasi – operasi

Operasi – operasi pada dasarnya mendefinisikan perilaku suatu objek. Secara umum operasi – operasi dapat dikategorikan menjadi empat yaitu (Roger S. Pressman, 2012):

- a. Operasi yang melakukan manipulasi data dengan cara – cara tertentu misalnya (menambah, menghapus, melakukan performatan ulang, memilih)
- b. Operasi yang melaksanakan komputasi
- c. Operasi yang mencoba menjawab tentang keadaan(state) suatu objek
- d. Operasi yang melakukan pemantauan terhadap suatu objek dengan maksud

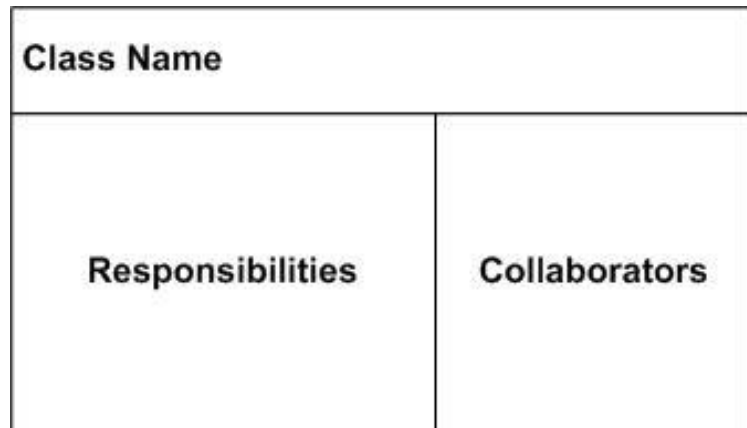
melakukan pengendalian atas objek yang bersangkutan.

Suatu operasi harus memiliki pengetahuan alamaiah tentang atribut – atribut dan asosiasi – asosiasi suatu kelas.

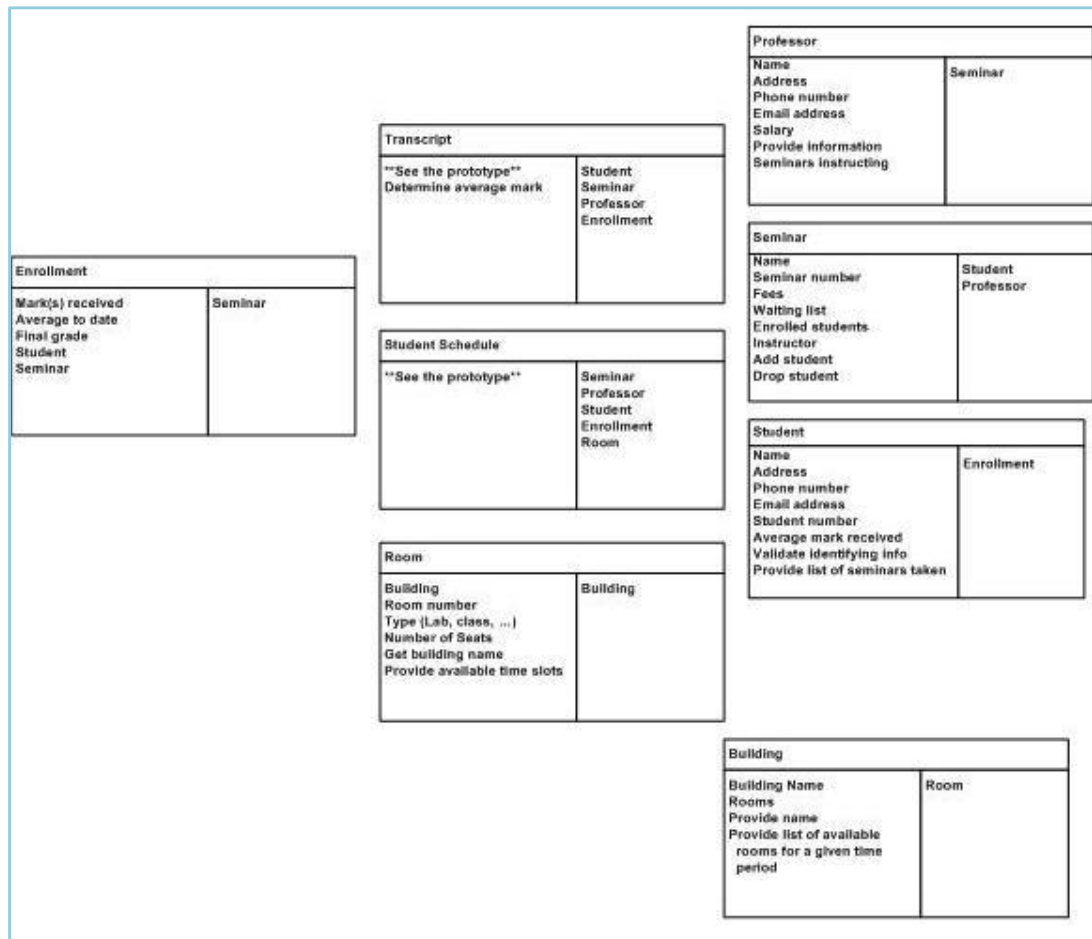
Operasi adalah abstraksi dari segala sesuatu yang dapat dilakukan pada sebuah objek dan berlaku untuk semua objek yang terdapat dalam class tersebut. Class mungkin memiliki beberapa operasi, biasanya pemanggilan operasi pada sebuah objek akan mengubah data atau kondisi dari objek tersebut.

4. Pemodelan CRC (*Class-Responsibility-Collaborator*)

Pemodelan CRC menyediakan cara untuk mengidentifikasi dan mengorganisasi kelas – kelas yang relevan untuk sistem atau relevan untuk menspesifikasi – menspesifikasi kebutuhan produk. Menurut Ambler dalam (Roger S. Pressman, 2012) mengemukakan bahwa suatu model CRC merupakan kumpulan kartu – kartu yang masing – masing merepresentasikan kelas – kelas. Kartu CRC terbagi menjadi tiga bagian yaitu nama kelas, mendaftar tanggung jawab kelas, mendaftar kolaborator-kolaboratornya.



Gambar 5.9 CRC Card



Gambar 5.10Contoh CRC Model

Sumber : (Ambler, 2018)

5. Menggambar Class

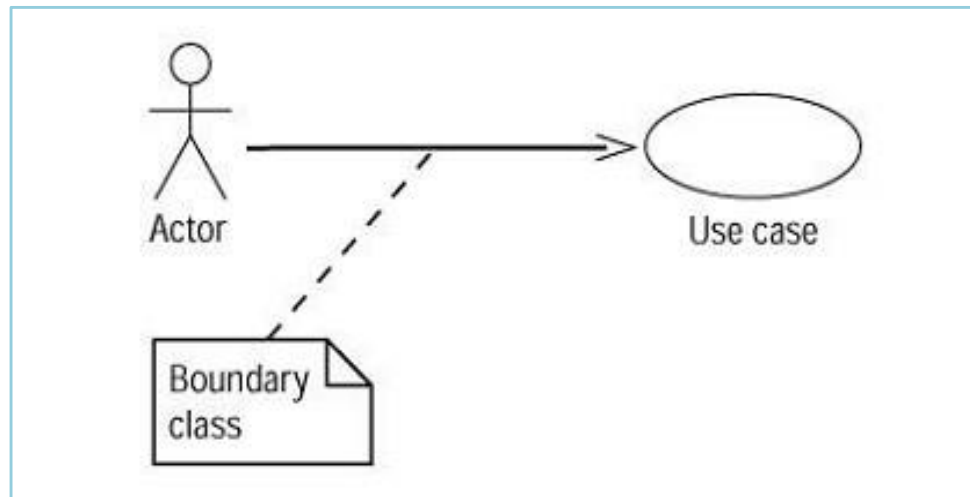
Ketika menggambarkan sebuah class, kita tidak perlu menampilkan seluruh atribut atau operasi, karena sebagian kasus kita tidak dapat menampilkannya dalam sebuah gambar, karena terlalu banyaknya atribut atau operasinya, bahkan terkadang tidak perlu karena kurang relevannya atribut atau operasi yang akan ditampilkan. Sehingga kita dapat menampilkan atribut dan operasinya hanya sebagian. Kosongnya tempat pengisian bukan berarti tidak ada, karena itu dapat menambahkan tanda (...) pada akhir daftar yang menunjukkan bahwa masih ada atribut atau operasi yang lain.

6. Stereotype Class

Stereotype adalah sebuah mekanisme yang digunakan untuk mengkategorikan sebuah class. Sehingga memudahkan kita dalam mengorganisasikan responsibility dari tiap – tiap class. Terdapat tiga stereotype utama dalam UML yaitu boundary, entity dan control.

- a. Boundary class adalah class yang terdapat batasan sistem dan dunia nyata, hal ini mencakup semua form, hardware interface. Boundary class dapat diidentifikasi dari use case diagram. Minimal terdapat satu buah boundary class dalam relasi actor dengan usecase.

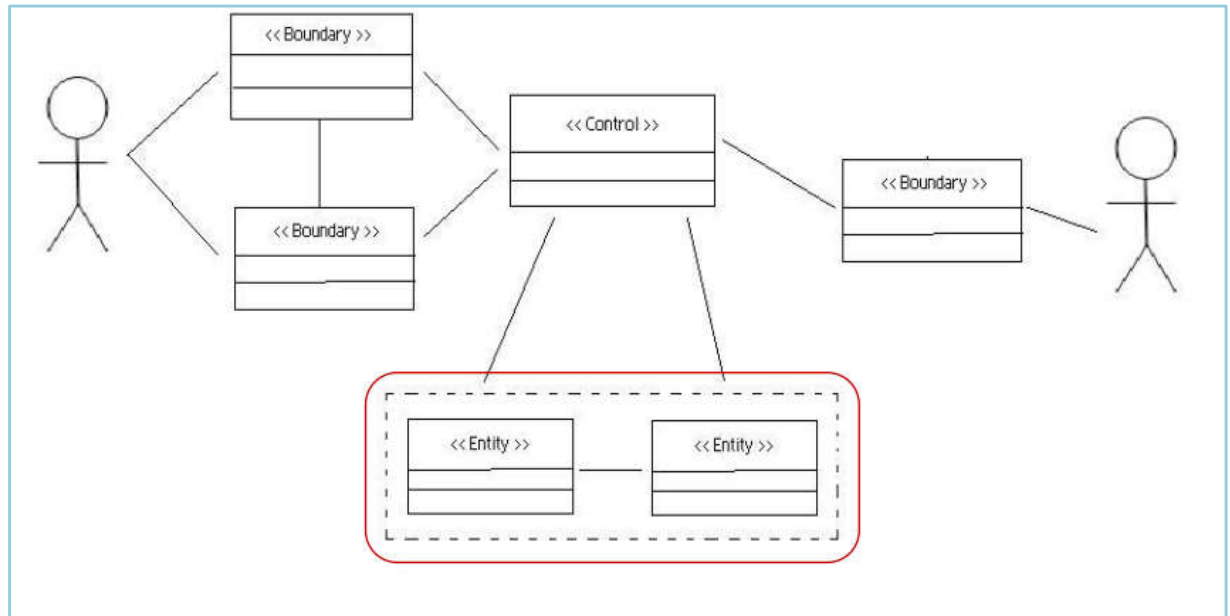
Boundary class adalah yang mengakomodasi interaksi antara actor dengan sistem.



Gambar 5.11 Boundary Class

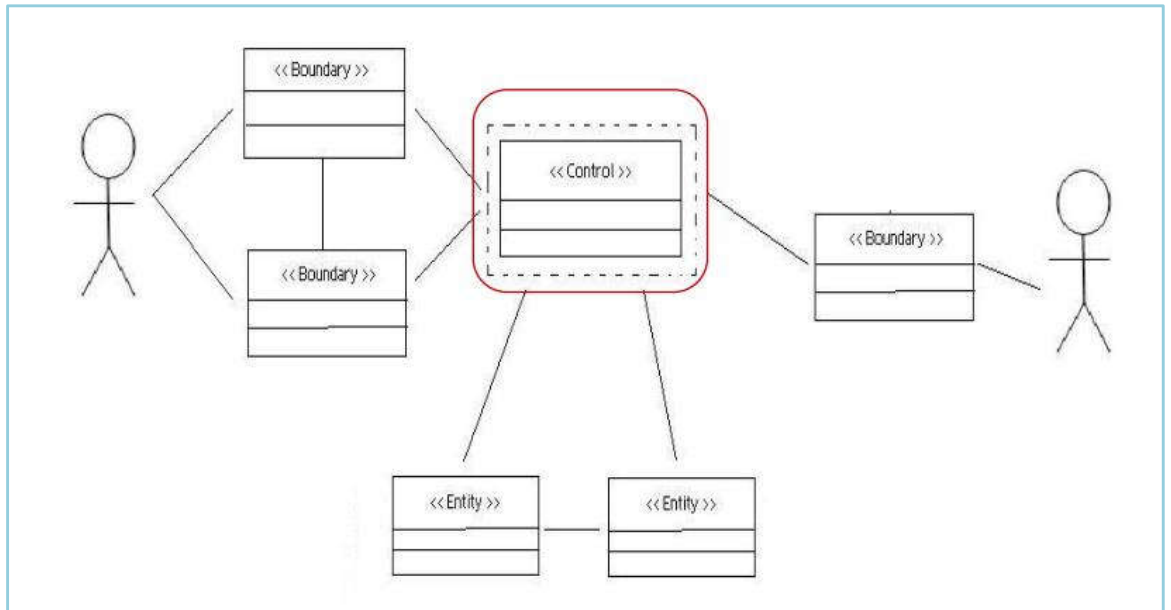
- b. Entity Class
- Entity class menyimpan informasi yang mungkin akan disimpan ke sebuah storage . Class dengan stereotype entity dapat ditemukan di flow of event (scenario dari use case diagram) dan interaction diagram. Entity class dapat diidentifikasi dengan mencari kata benda yang ada pada flow of events . Selain itu, dapat juga diidentifikasi dari struktur database (dilihat dari nama – nama tabelnya). Sebuah entity class mungkin perlu dibuat untuk sebuah tabel. Bila sebuah tabel menyimpan informasi secara permanen, maka entity class akan

menyimpan informasi pada memory ketika sistem sedang running.



Gambar 5.12 Entity Class

- c. Control Class
Control class bertanggung jawab dalam mengatur kelas – kelas yang lain. Control class juga bertanggung jawab dalam mengetahui dan menyampaikan business rule dari sebuah organisasi. Class ini menjalankan alternate flow dan mampu mengatasi error . karena alasan ini controll class sering disebut manager class.



Gambar 5.13 Control Class

7. Relationship

Relasi atau relationship menghubungkan beberapa objek sehingga memungkinkan terjadinya interaksi dan kolaborasi diantara objek – objek yang terhubung. Dalam pemodelan class diagram, terdapat tiga buah relasi utama yaitu association, aggregation dan generalization.

a. Asosiasi

Relasi asosiasi merupakan relasi struktural yang menspesifikasikan bahwa satu objek terhubung dengan objek lainnya. Relasi ini tidak menggambarkan aliran data, sebagaimana yang terdapat pada pemodelan desain pada analisa terstruktur. Relasi asosiasi dapat dibagi menjadi dua jenis yaitu :

I) Uni-directional association dan bi-directional association



Objek supir memiliki uni-directional association dengan objek taxi. Relasi uni-directional diatas memungkinkan objek supir untuk memanggil properti dari objek taxi. Namun tidak berlaku sebaliknya . Objek pesawat tidak dapat mengakses properti dari objek supir.

2) Bi-directional



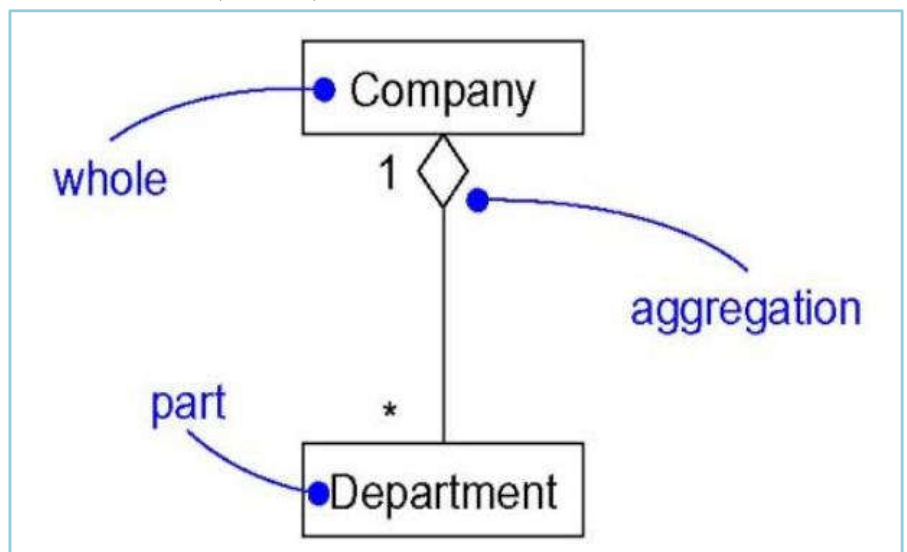
Objek supir dapat memanggil properti yang dimiliki oleh objek taksi. Begitu juga sebaliknya, objek taksi dapat memanggil properti dari objek supir

Hubungan association mempunyai dua titik. Salah satu titik bisa memiliki label untuk menjelaskan association tersebut. Panah association menggambarkan arah

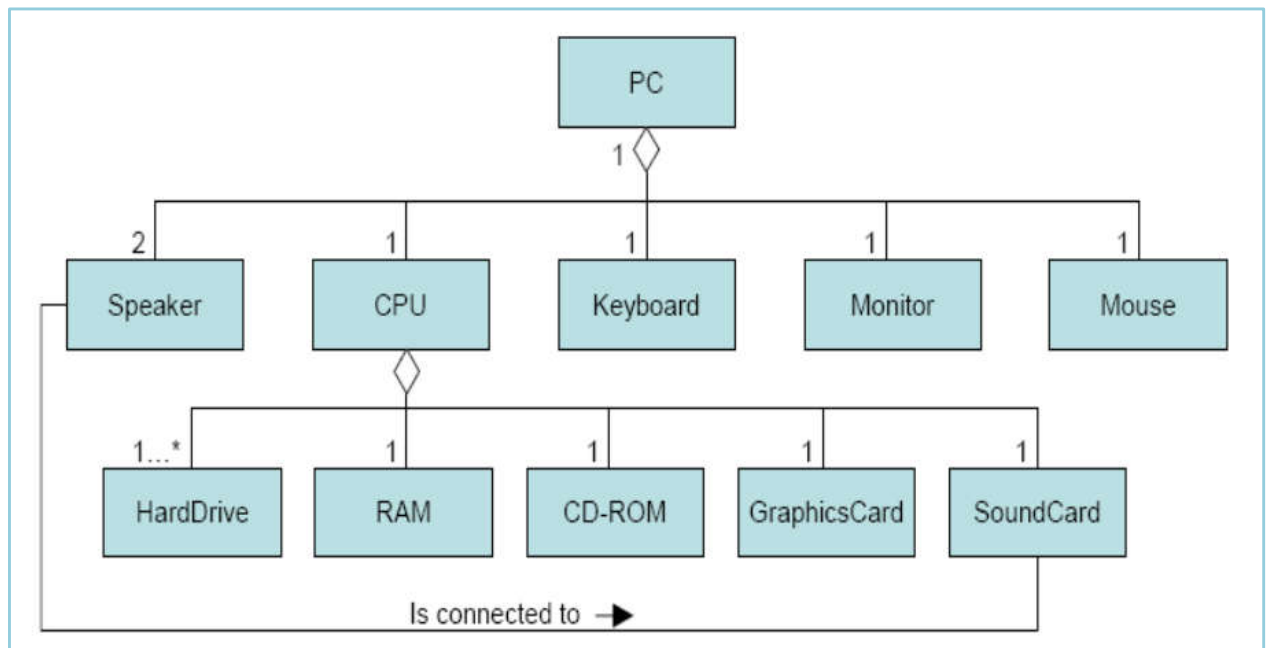
mana association dapat di transfer atau disusun.

b. Aggregation

Aggregation merupakan bentuk khusus dari asosiasi dimana induk terhubung dengan bagian – bagiannya. Aggregation merepresentasikan relasi “has-a”, artinya sebuah class memiliki atau terdiri dari bagian yang lebih kecil. Dalam UML, relasi aggregasi digambarkan dengan open diamond pada sisi yang menyatakan induk (whole).



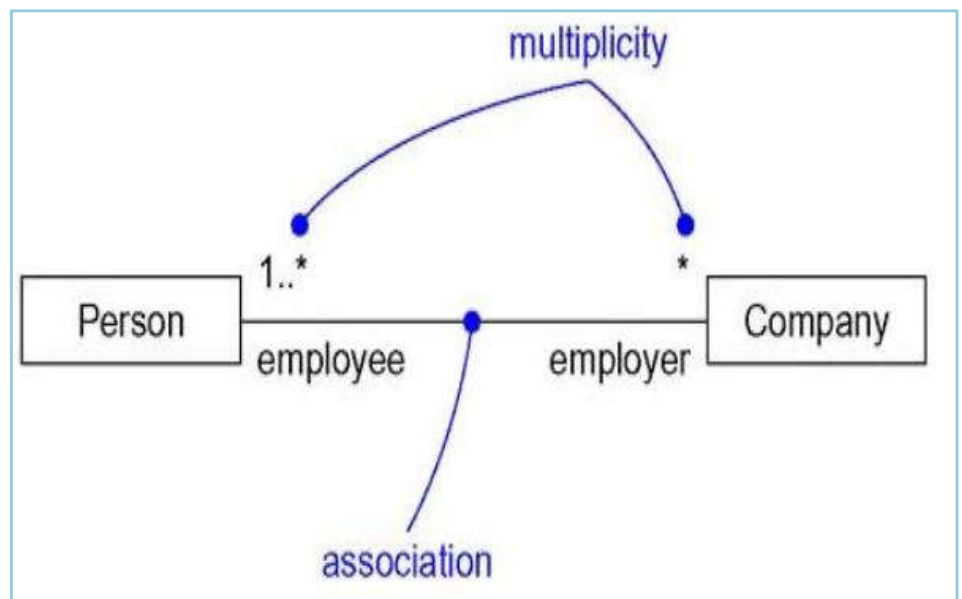
Gambar 5.14 Contoh Aggregation



Gambar 5.15 Contoh Aggregation dari PC

Multiplicity

Multiplicity menentukan atau mendefinisikan banyaknya objek yang terhubung dalam suatu relasi. Indikator multiplicity terdapat pada masing – masing akhir garis relasi baik pada asosiasi.



Gambar 5.16 Contoh Multiplicity

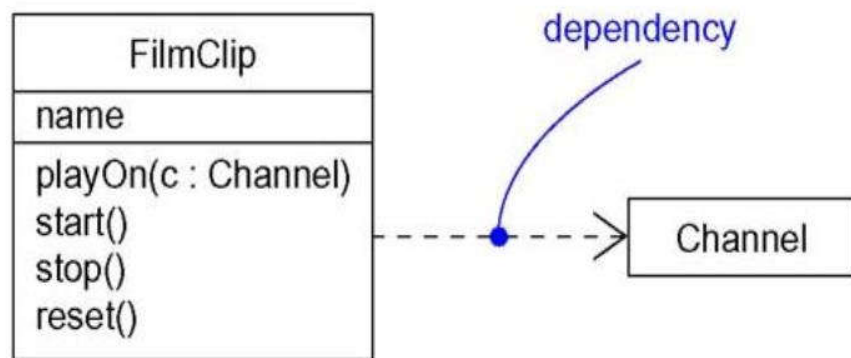
Multiplicity dari suatu titik association adalah angka kemungkinan bagian dari hubungan kelas dengan single instance(bagian) pada titik yang lain.

Tabel 5.1 Multiplicity

Multiplicity	Artinya
0..1	Nol atau satu bagian. Notasi n..m menerangkan n sampai m bagian
0..* atau *	Tak hingga pada jangkauan bagian (termasuk kosong)
1	Tepat satu bagian
1..*	Sedikitnya hanya satu bagian

c. Dependency

Dependency merupakan sebuah relasi yang menyebutkan bahwa perubahan pada satu class, maka akan mempengaruhi class lain yang menggunakannya, tetapi tidak berlaku sebaliknya. Pada umumnya relasi dependency dalam konteks Class Diagram digunakan apabila terdapat satu class yang menggunakan class lain sebagai argumen dari sebuah method.



Gambar 5.17 Contoh dependency

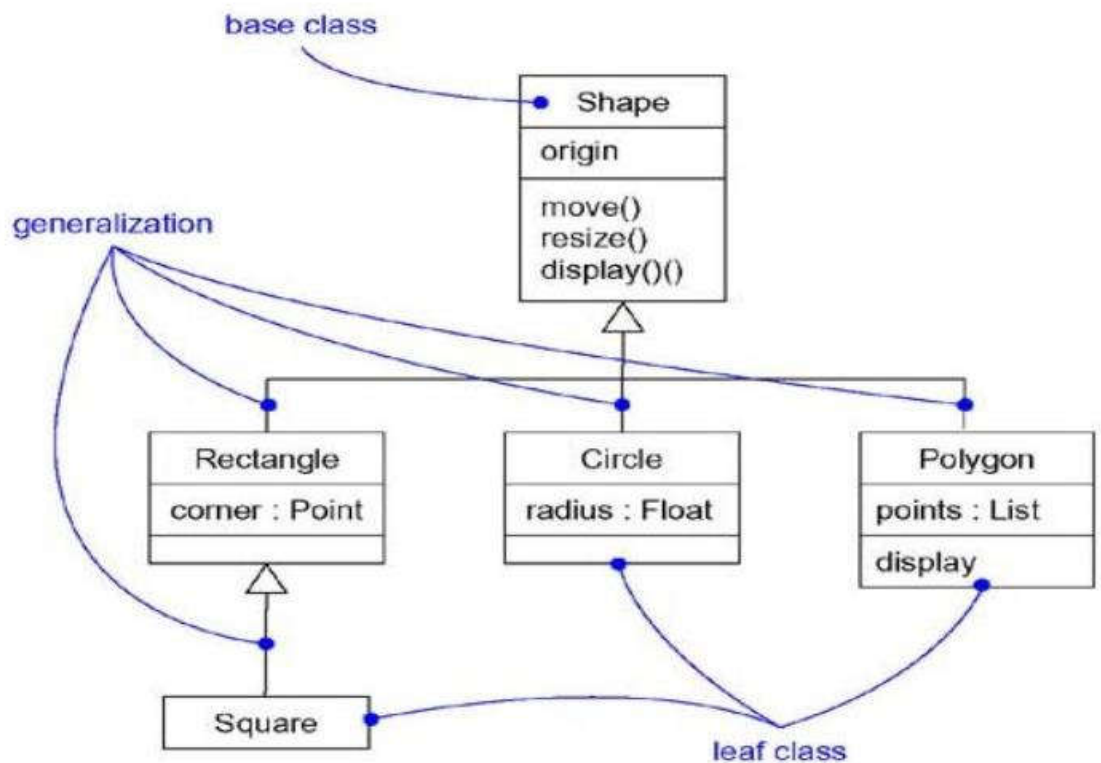
Inheritance

Inheritance merupakan salah satu karakteristik dalam pemrograman berorientasi objek, dimana class mewarisi /inherit sifat – sifat (dalam hal ini atribut dan operasi) dari class lain yang merupakan parent dari class tersebut. Class yang menurunkan sifat – sifatnya disebut superclass sedangkan class yang

mewarisi sifat dari superclass disebut subclass. Inherentance disebut juga hierarki “is-a” (adalah sebuah) atau “kind-of” (sejenis). Subclass dapat memiliki atau menggunakan atribut dan operasi tambahan yang hanya berlaku pada tingkat hierarkinya. Karena inheritance relationship bukan merupakan relationship diantara objek yang berbeda , maka relationship ini tidak diberi nama. Begitu pula dengan penamaan multiplicity.

d. Generalization

Generalisasi merupakan relasi antar class dengan makan generalisasi-spesialisasi (umum – khusus).

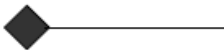




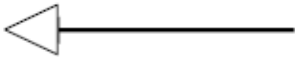
Gambar 5.18 Contoh Generalization

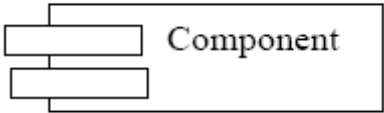
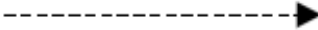
8. Simbol – simbol dalam Class Diagram
Berikut adalah simbol – simbol yang digunakan dalam menggambar Class Diagram.

Tabel 5.2 Simbol – simbol dalam class diagram

SIMBOL	NAMA	KETERANGAN
<div style="border: 1px solid black; padding: 5px; margin: 5px;"> <div style="text-align: center;">Site Config</div> <hr/> +sqlDNS:string +Adminemail:String </div>	Class	<i>Class</i> adalah blok - blok pembangun pada pemrograman berorientasi obyek. Sebuah class digambarkan sebagai sebuah kotak yang

		terbagi atas 3 bagian. Bagian atas adalah bagian nama dari <i>class</i> . Bagian tengah mendefinisikan property/atribut <i>class</i> . Bagian akhir mendefinisikan method dari sebuah <i>class</i> .
<u>1..n Owned by 1</u>	Assosiation	Sebuah asosiasi merupakan sebuah <i>relationship</i> paling umum antara 2 <i>class</i> , dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 <i>class</i> . Garis ini bisa melambangkan tipe-tipe <i>relationship</i> dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah <i>relationship</i> (Contoh: One-to-one, one-to-many, many-to-many).
	Composition	Jika sebuah <i>class</i> tidak bisa berdiri sendiri dan harus merupakan bagian dari <i>class</i> yang lain, maka <i>class</i> tersebut memiliki relasi <i>Composition</i> terhadap <i>class</i> tempat dia bergantung tersebut. Sebuah <i>relationship composition</i> digambarkan sebagai garis

		dengan ujung berbentuk jajaran genjang berisi/solid.
	Dependency	Kadangkala sebuah <i>class</i> menggunakan <i>class</i> yang lain. Hal ini disebut <i>dependency</i> . Umumnya penggunaan <i>dependency</i> digunakan untuk menunjukkan operasi pada suatu <i>class</i> yang menggunakan <i>class</i> yang lain. Sebuah <i>dependency</i> dilambangkan sebagai sebuah panah bertitik-titik.
	Aggregation	<i>Aggregation</i> mengindikasikan keseluruhan bagian <i>relationship</i> dan biasanya disebut sebagai relasi “mempunyai sebuah” atau “bagian dari”. Sebuah <i>aggregation</i> digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid.
	Generalization	Sebuah relasi <i>generalization</i> sepadan dengan sebuah relasi <i>inheritance</i> pada konsep berorientasi obyek. Sebuah <i>generalization</i> dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid

		yang mengarah ke kelas “ <i>parent</i> ”-nya/induknya.
	komponen	Sebuah komponen melambangkan sebuah entitas software dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.
	Depedency	Sebuah Dependency digunakan untuk menotasikan relasi antara dua komponen. Notasinya adalah tanda panah putus-putus yang diarahkan kepada komponen tempatsebuah komponen itu bergantung.

Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. Class diagram memberikan pandangan secara luas dari suatu sistem dengan menunjukkan kelas – kelasnya dan hubungan mereka. Diagram class bersifat statis yaitu menggambarkan hubungan apa yang terjadi bukan apa yang terjadi jika mereka berhubungan. Class diagram dapat membantu memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe

diagram yang paling dalam pemodelan system berbasis object-orientes. Class Diagram memperlihatkan sekumpulan class, interface dan collaborations dan relasi didalamnya. Selama proses analisa, class diagram memperhatikan aturan – aturan dan tanggung jawab entitas yang menentukan perilaku sistem . Selama tahap desain, class diagram berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. Kita memodelkan class diagram untuk memodelkan static design view dari suatu sistem. (Desy, class Diagram, 2018)

5.2.3. Pemodelan Berorientasi Aliran

Walaupun pemodelan berorientasi data saat ini dianggap merupakan teknik yang kadaluarsa oleh kebanyakan rekayaswan perangkat lunak. Meskipun DFD (Data Flow Diagram) serta diagram – diagram dan informasi – informasi yang terkait bukan bagian dari UML, pada dasarnya tetap dapat digunakan untuk melengkapi diagram – diagram UML dan menyediakan wawasan tambahan yang berkaitan dengan kebutuhan – kebutuhan dna aliran – aliran data yang terjadi pada sistem.

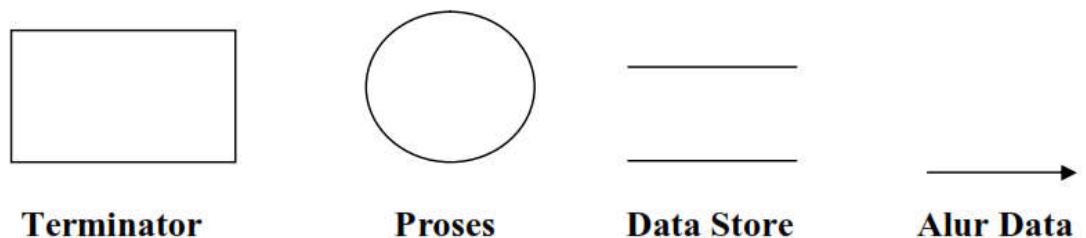
DFD memperlihatkan gambaran tentang masukan proses keluaran dari suatu perangkat lunak yaitu objek – objek data mengalir ke dalam perangkat lunak, kemudian objek – objek data itu akan ditransformasi oleh elemen – elemen pemrosesan, dan objek – objek data hasilnya akan mengalir keluar dari perangkat lunak.

objek – objek data dalam penggambaran DFD biasanya direpresentasikan menggunakan panah berlabel, dan transformasi – transformasi biasanya direpresentasikan menggunakan lingkaran – lingkaran yang biasa disebut dengan gelembung. DFD pada dasarnya digambarkan dalam bentuk hierarki yaitu DFD yang pertama disebut sebagai DFD tingkat 0 atau konteks yang menggambarkan sistem secara keseluruhan. DFD – DFD berikutnya merupakan penghalusan dari diagram kontkes, yang memberikan gambaran yang semakin rinci dari diagram konteks, dalam hal ini akan berlanjut ke peringkat – peringkat selanjutnya.

I. Membuat Model Aliran Data -- Data Flow Diagram (DFD)

a. Komponen Data Flow Diagram

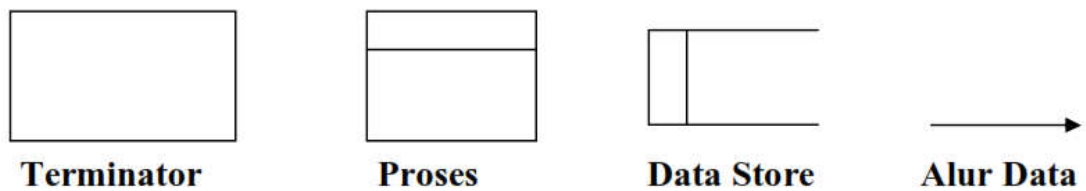
Menurut Yourdan dan Demarco



Gambar 5.19 Komponen DFD menurut Ypurdan dan Demarco

Sumber : (Darmastuti, 2018)

Menurut Gene dan Serson

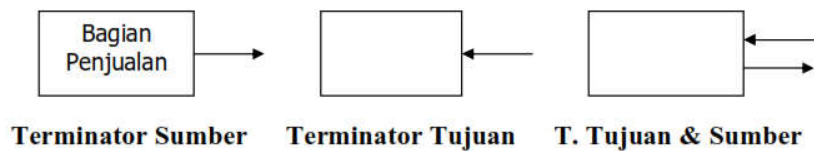


Gambar 5.20 Komponen DFD menurut
Gene dan Serson

Sumber : (Darmastuti, 2018)

I) Terminator

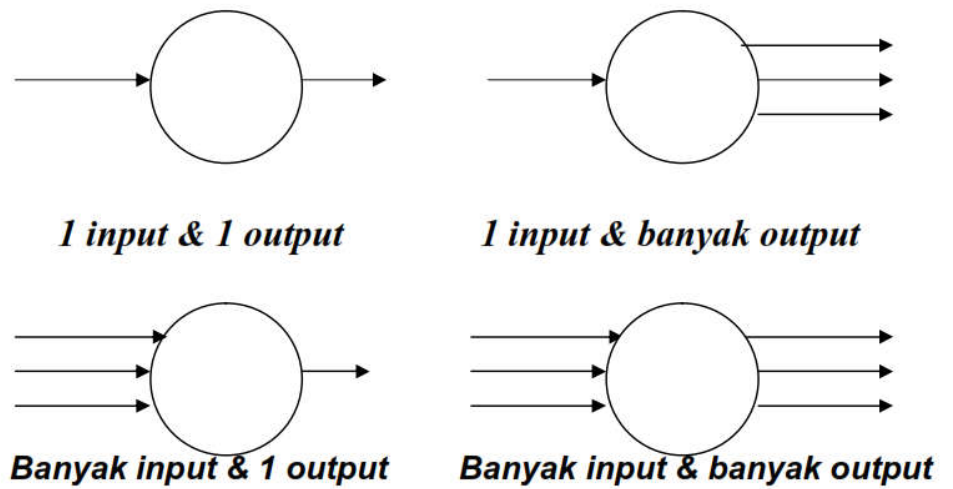
Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Terdapat dua jenis terminator yaitu terminator sumber dan terminator tujuan



Gambar 5.21 Jenis Terminator

2) Proses

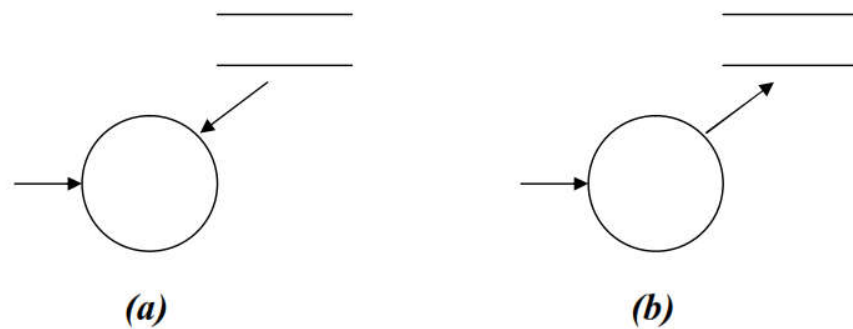
Komponen proses menggambarkan bagian dari sistem yang mentransformasikan input menjadi output. Ada empat kemungkinan yang dapat terjadi dalam proses sehubungan input dan output :



Gambar 5.22 hubungan input output
pada komponen proses

3) Data Store

Komponen data store digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda jamak. Alur data yang menghubungkan data store dengan suatu proses mempunyai pengertian yaitu (a) alur data dari data store yang berarti sebagai pembacaan satu paket tunggal data, (2) alur data ke data store yang berarti sebagai pengupdatetan data.

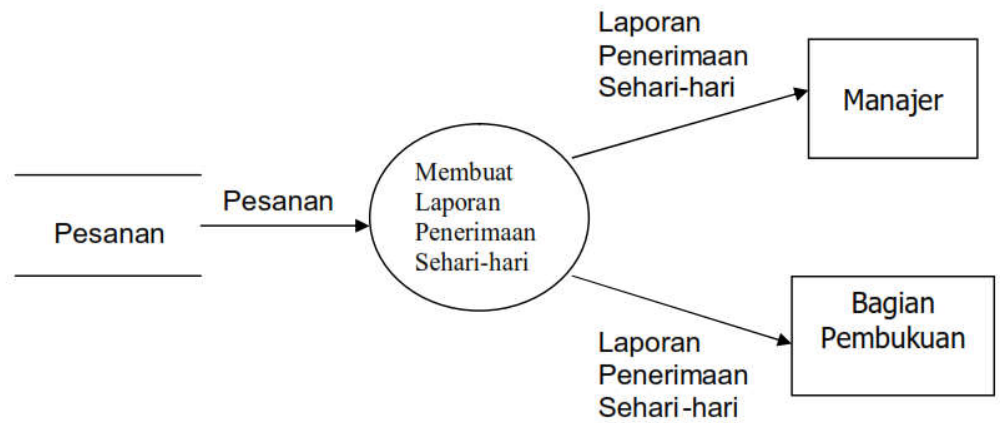


Gambar 5.23 Implementasi Data Store

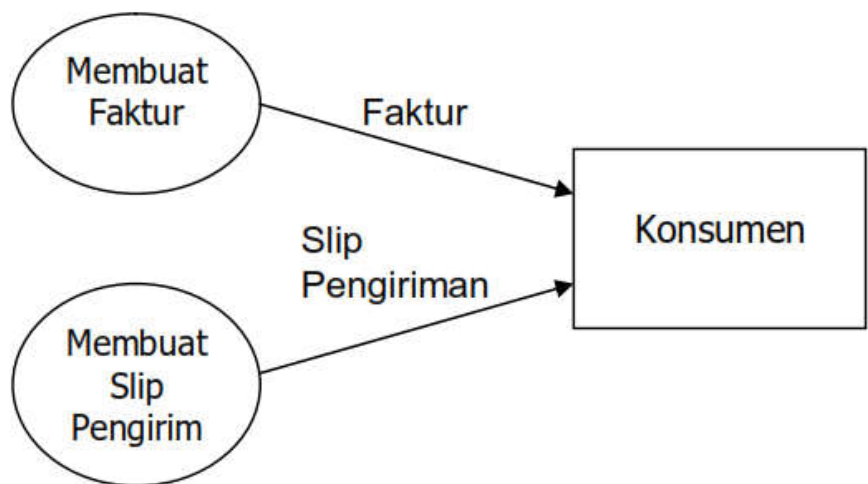
- 4) Data Flow atau Alur data
 Suatu data flow digambarkan dengan anak panah yang menunjukkan arah menuju ke dan keluar dari suatu proses. Alur data digunakan untuk menerangkan perpindahan data atau paket data dari satu bagian sistem ke bagian lainnya. Ada empat konsep dalam penggambaran alur data :



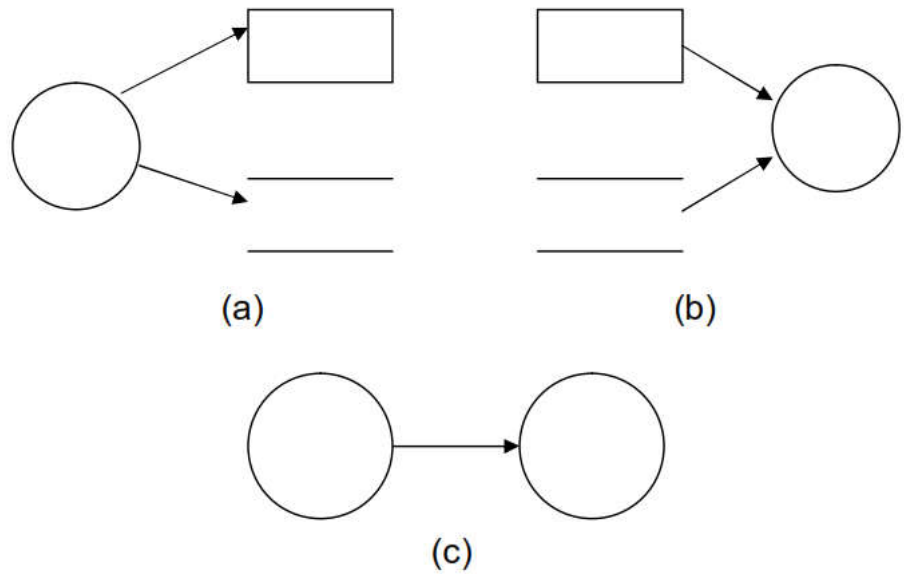
Gambar 5.24 Konsep Paket Data



Gambar 5.25 Konsep Alur Data Menyebar

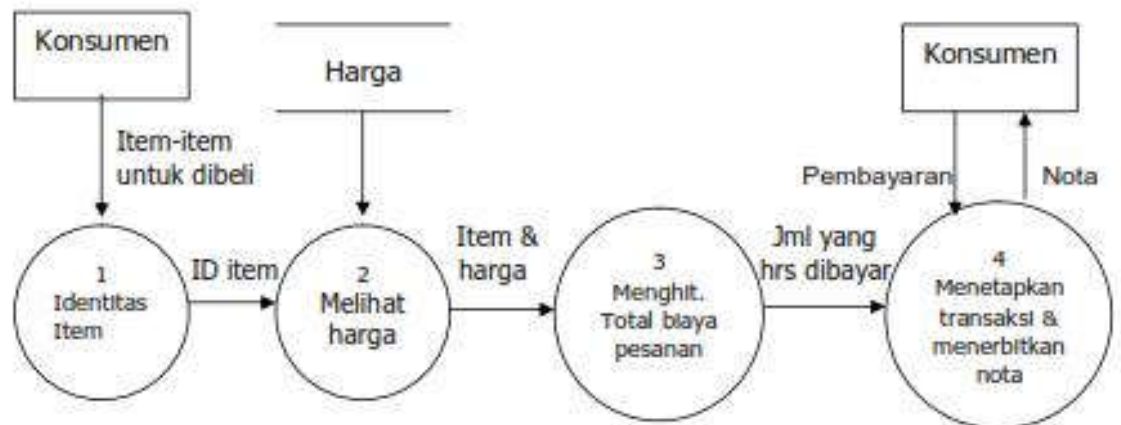


Gambar 5.26 Konsep Alur Data Mengumpul

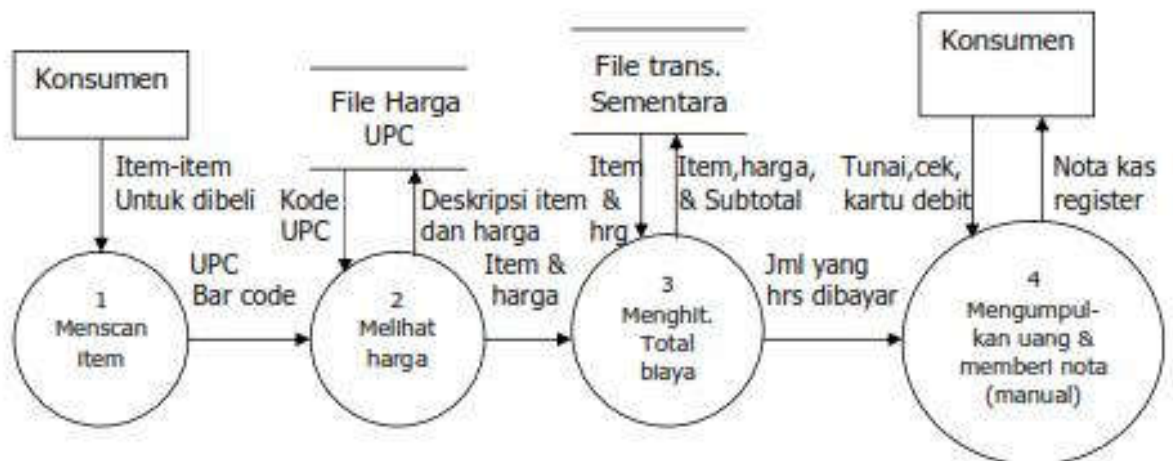


Gambar 5.27 Konsep Sumber atau Tujuan Alur Data

- b. Bentuk Data Flow Diagram
- Terdapat dua bentuk DFD yaitu diagram alur data fisik (DADF) dan diagram alur data logic (DADL) . DADF digunakan untuk menggambarkan sistem yang ada atau sistem yang lama , sedangkan DADL digunakan untuk menggambarkan sistem yang baru atau usulan.



(a) Diagram Alur Data Logika



(b) Diagram Alur Data Fisik

Gambar 5.28 DADL dan DADF

Sumber : (Darmastuti, 2018)

- c. Penggambaran DFD

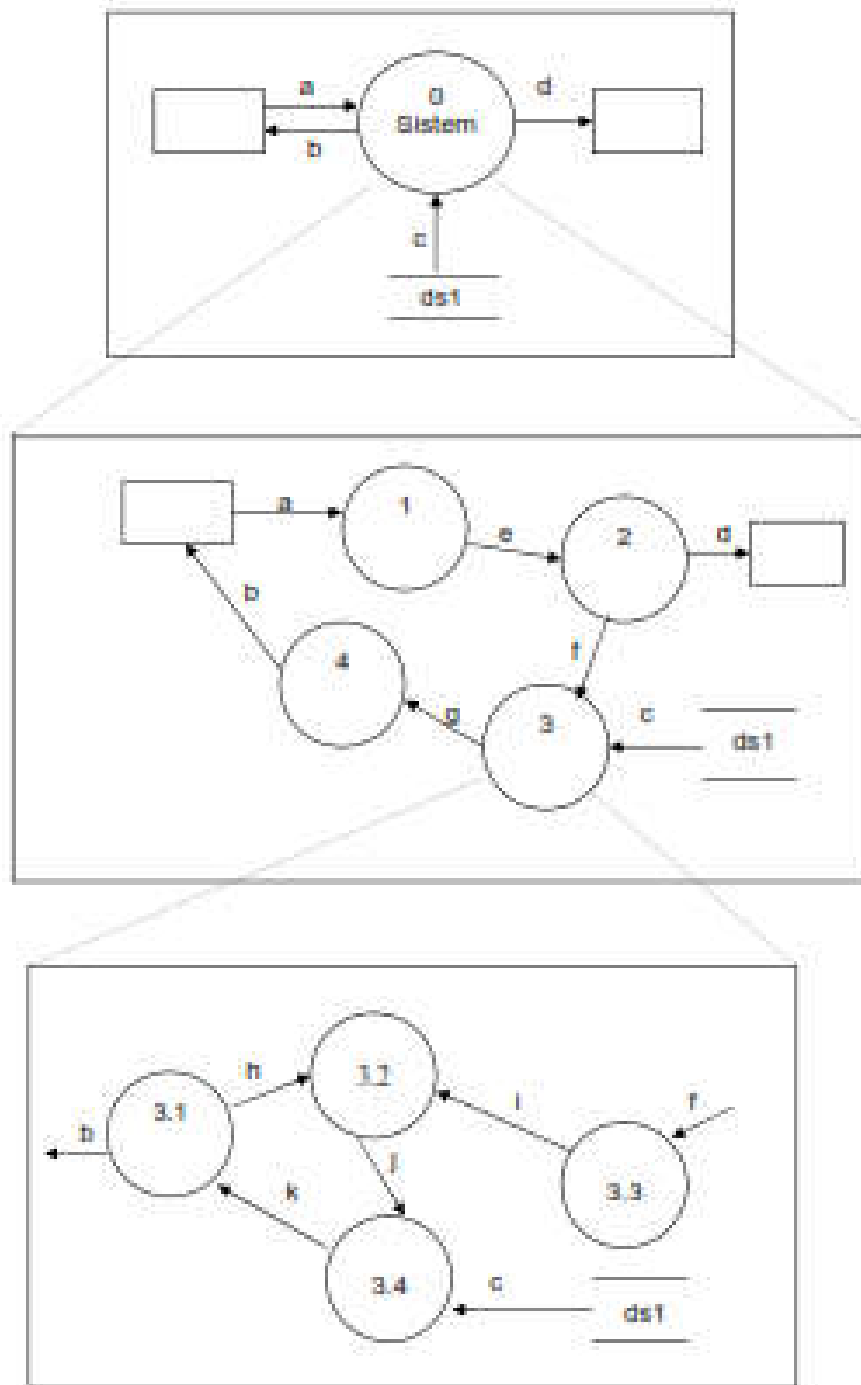
Secara garis besar langkah untuk membuat DFD adalah sebagai berikut :

 - I) Identifikasi terlebih dahulu semua entitas luar yang terlibat di sistem

- 2) Identifikasi semua input dan output yang terlibat dengan entitas
- 3) Buat diagram konteks, diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan hubungan sistem dengan lingkungan luarnya. Cara membuat diagram konteks :
 - a) Tentukan nama sistemnya
 - b) Tentukan batasan sistemnya
 - c) Tentukan terminator apa saja yang ada dalam sistem
 - d) Tentukan apa yang diterima atau diberikan terminator dari atau ke sistem
 - e) Gambarakan diagram konteks
- 4) Buat diagram level zero, diagram ini adalah dekomposisi dari diagram konteks. Cara membuat diagram level zero :
 - a) Tentukan proses utama yang ada pada sistem
 - b) Tentukan apa yang diberikan atau diterima masing – masing proses ke atau dari sistem sambil

- memperhatikan konsep keseimbangan
 - c) Apabila diperlukan munculkan data store (master) sebagai sumber atau tujuan alur data
 - d) Gambarkan diagram level zero.
- 5) Buat diagram level satu, diagram ini merupakan dekomposisi dari diagram level zero. Cara menggambarkan diagram level zero :
- a) Tentukan proses yang lebih kecil dari proses utama yang ada di level zero
 - b) Tentukan apa yang diberikan atau diterima dari proses ke atau dari sistem dan perhatikan konsep keseimbangan
 - c) Apabila diperlukan munculkan data store (transaksi) sebagai sumber maupun tujuan alur data.
 - d) Gambarkan DFD level satu
- 6) DFD level dua, tiga dan seterusnya . Diagram ini

merupakan dekomposisi dari level sebelumnya . Proses dekomposisi dilakukan sampai dengan proses siap dituangkan ke dalam program . Aturan yang digunakan sama dengan level satu.



Gambar 5.29 levelisasi DFD

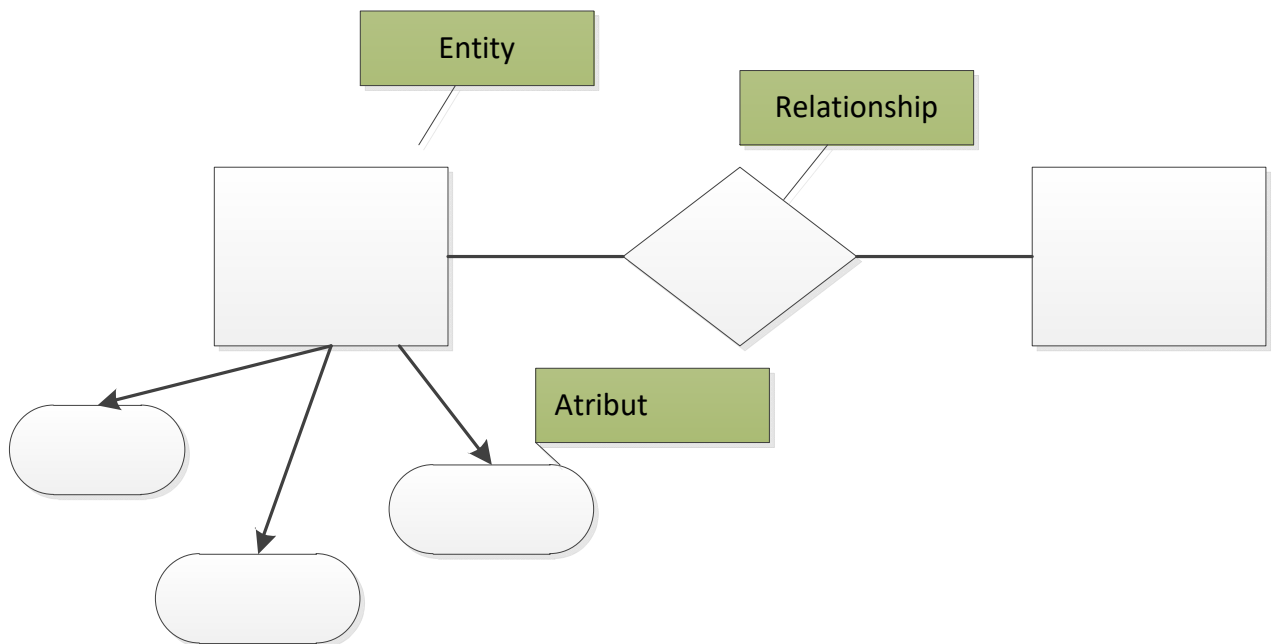
2. Model Data

Jika kebutuhan – kebutuhan perangkat lunak mencakup didalamnya kebutuhan untuk membuat , memperluas atau bersinggungan dengan basis data atau jika struktur data yang kompleks harus dibentuk dan dimanipulasi, tim perangkat lunak dapat menggunakan pemodelan data. Rekayasawan perangkat lunak dapat mendefinisikan semua objek data yang akan diproses didalam perangkat lunak, mendefinisikan relasi antar objek dengan menggunakan pemodelan data yaitu ERD (Entity Relationship Diagram).

Entity Relationship Diagram

Entity relationship diagram adalah suatu model penyajian data dengan menggunakan entity dan relationship. ERD menggambarkan model konseptual untuk menggambarkan struktur logis dari basisdata berbasis grafis yang bertujuan agar database dapat dipahami dan dirancang dengan mudah.

a. Simbol – simbol ERD



Gambar 5.30 Simbol – simbol ERD

- 1) Entity , objek yang dapat dibedakan dalam dunia nyata .
- 2) Relationship , hubungan yang terjadi antara satu atau lebih entity
- 3) Atribut , karakteristik dari setiap entity yang menyediakan penjelasan detail mengenai entity tersebut. Nilai dari atribut adalah data aktual atau informasi yang disimpan pada suatu atribut di dalam entity, dimana tiap atribut memiliki domain tersendiri. Jenis

– jenis atribut yang digunakan dalam ERD :

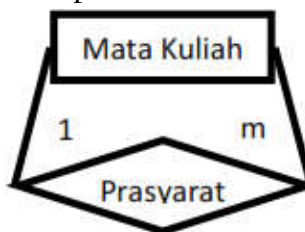
- a) Key , atribut yang digunakan untuk menentukan suatu entity secara unik
- b) Atribut simple , atribut sederhana yang tidak dapat dibagi dalam beberapa bagian
- c) Atribut komposit , atribut yang dapat dibagi dalam beberapa bagian. Contohnya adalah alamat yang dapat dibagi menjadi kota, propinsi , negara.
- d) Atribut single-valued, atribut yang memiliki paling banyak satu nilai untuk setiap baris data.
- e) Multi-valued attributes , atribut yang dapat diisi dengan lebih satu nilai tetapi sejenisnya sama. Contohnya nomor telp. Alamat, gelar
- f) Atribut turunan, atribut yang diperoleh dari pengolahan dari atribut lain yang berhubungan.

- g) Atribut Key , attribut yang dapat dijadikan kunci untuk mencari data dalam relasi .

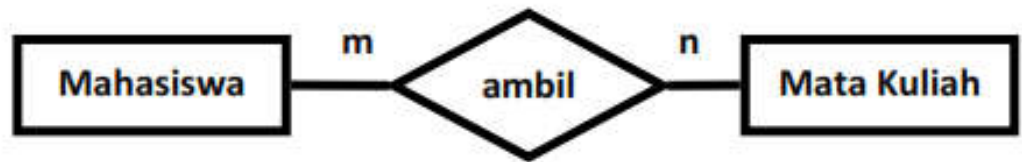
b. Derajat Relasi

Derajat relasi menunjukkan banyaknya himpunan entitas yang saling berelasi . jenis derajat himpunan relasi adalah (Satrio Agung W, 2011):

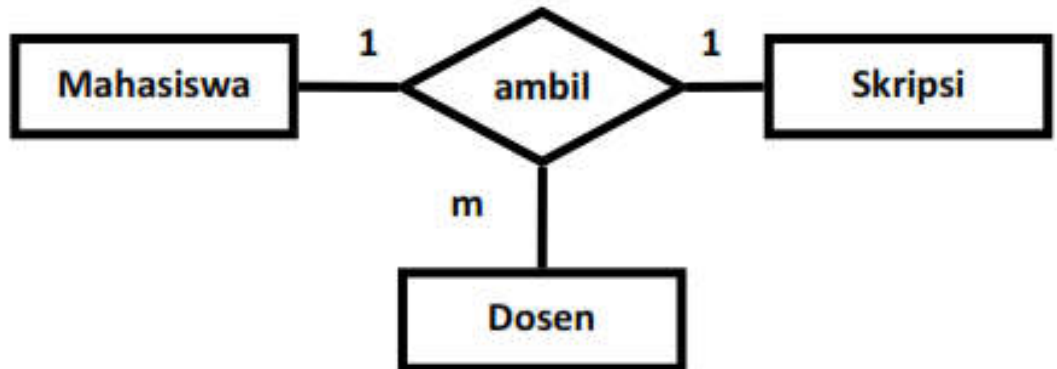
- 1) Unary degree(derajat satu) melibatkan sebuah entitas yang berelasi dengan dirinya sendiri
- 2) Binary degree(derajat dua) himpunan relasi melibatkan dua himpunan entitas. Secara umum himpunan relasi dalam sistem basis data adalah binary.
- 3) Ternary degree(derajat tiga) himpunan relasi memungkinkan untuk melibatkan lebih dari dua himpunan entitas



Gambar 5.3I Unary degree



Gambar 5.32 Binary degree



Gambar 5.33 Ternary degree

- c. Pemetaan Kardinalitas Relasi
- Pemetaan kardinalitas relasi menggambarkan banyaknya jumlah maksimum entitas dapat berelasi dengan entitas pada himpunan entitas yang lain. Untuk himpunan relasi biner pemetaan kardinalitasnya dapat dibagi menjadi (Satrio Agung W, 2011):
- I) One to one, sebuah entity hanya dapat berelasi dengan satu buah objek di entity yang lain.
 - 2) One to many, sebuah entity dapat berelasi dengan banya objek di entity yang lain.

- 3) Many to one, banyak entity akan berelasi dengan satu objek yang sama pada entity yang lain.
- 4) Many to many, banyak entity yang akan berelasi dengan banyak objek di entity yang lain.



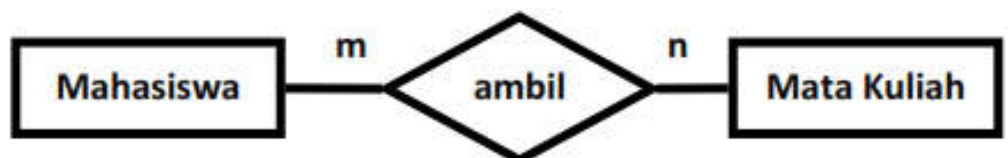
Gambar 5.34 One to one



Gambar 5.35 One to many



Gambar 5.36 Many to one



5.2.4. Pemodelan Berbasis Perilaku

Model perilaku menggambarkan bagaimana perangkat lunak akan berperilaku dalam menghadapi event – event yang datang dari arah luar atau bagaimana perangkat lunak akan berperilaku terhadap rangsangan – rangsangan yang muncul dari arah luar. Berikut langkah – langkah untuk membuat model ini : (a) melakukan evaluasi terhadap semua use case untuk secara penuh memahami urutan – urutan interaksi yang terjadi di dalam sistem perangkat lunak, (b) mengidentifikasi event – event yang mengendalikan urutan interaksi – interaksi dan memahami bagaimana event – event itu berhubungan dengan objek – objek yang bersifat spesifik, (c) membuat diagram – diagram yang memperlihatkan urutan – urutan untuk masing – masing use case, (d) mengembangkan diagram state untuk perangkat lunak yang akan dikembangkan, (e) meninjau model perilaku untuk memverifikasi akurasi dan konsistensinya.

I. Mengidentifikasi Event – event Menggunakan Use Case

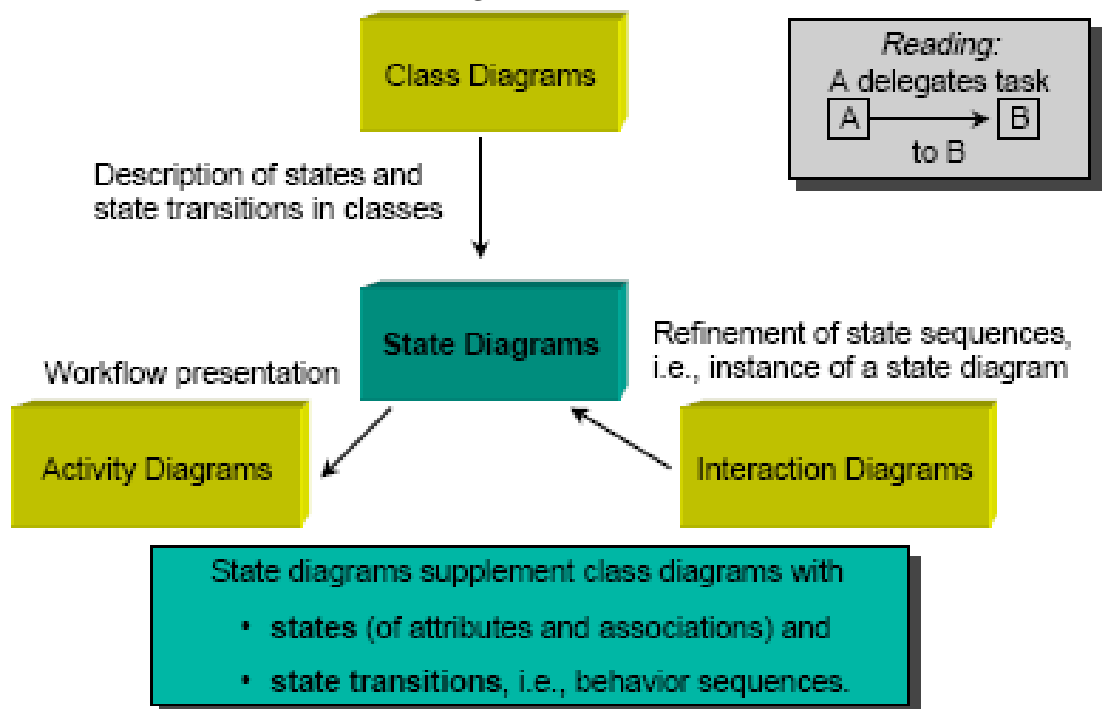
Use case pada dasarnya merepresentasikan suatu urutan aktivitas – aktivitas yang terjadi di antara aktor – aktor dan perangkat lunak yang sedang dirancang atau kembangkan, dan event terjadi saat sistem dan aktor – aktornya bertukar informasi. Jadi kita dapat mengidentifikasi event – event dengan melihat aktivitas – aktivitas use

case serta melihat event pertukaran informasi atau data yang ada pada aktivitas use case tersebut.

2. Representasi Keadaan (*State*)

Pada pemodelan berbasis perilaku ada dua karakteristik keadaan (*state*) yang perlu dipertimbangkan yaitu (a) keadaan dari masing – masing kelas , saat kelas – kelas itu melaksanakan fungsinya masing – masing, (b) keadaan sistem saat sistem itu diobservasi dari luar saat sistem itu melaksanakan fungsinya. Komponen dari pemodelan perilaku dalam UML adalah diagram state dan sequence diagram / diagram urutan aksi – aksi.

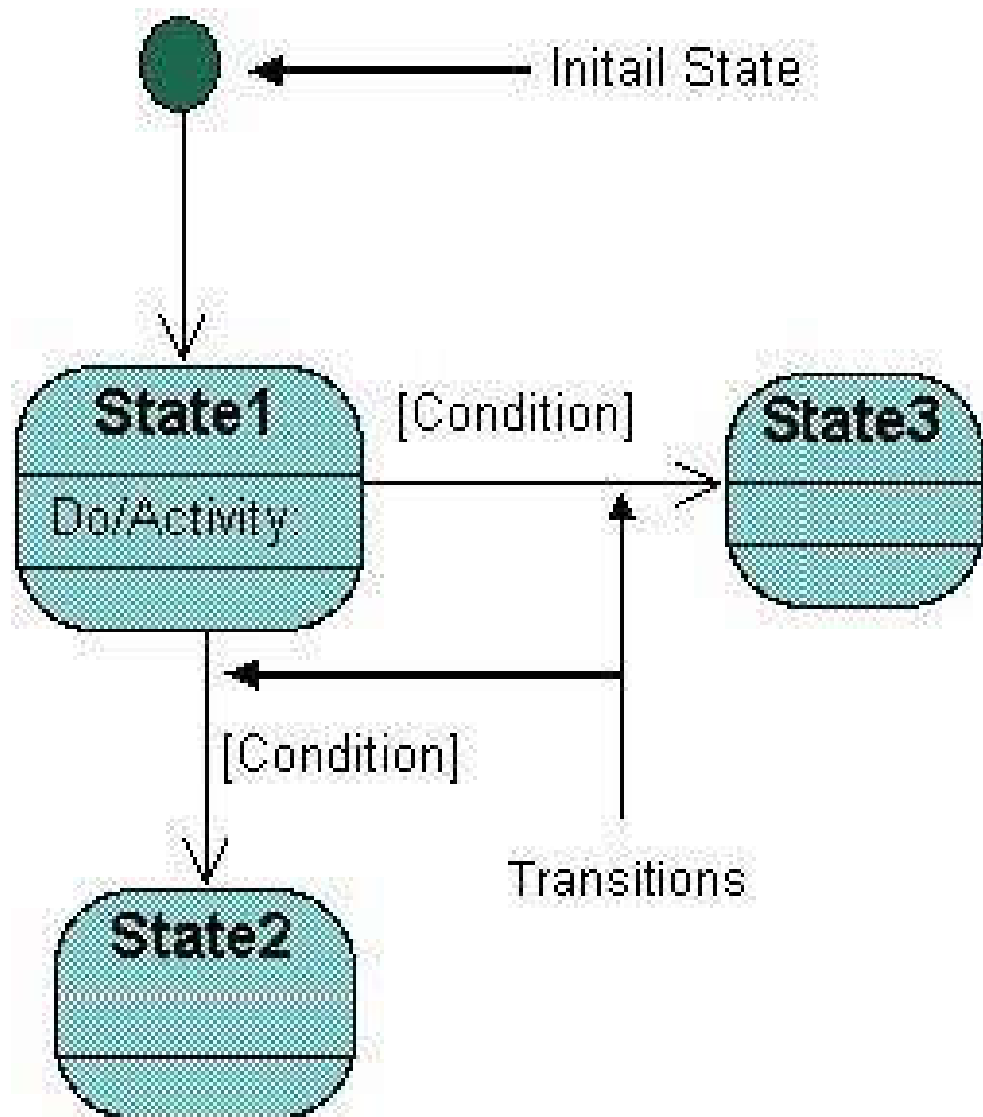
a. State Diagram



Gambar 5.38 Peran State Diagram dalam UML

Sumber : (Gunadarma.ac.id, 2018)

Diagram state merupakan diagram untuk menggambarkan behavior yaitu perubahan keadaan di suatu class berdasarkan event dan message yang dikirim dan diterima oleh class tersebut. Setiap diagram state hanya boleh memiliki satu start state (initial state) dan boleh memiliki satu atau lebih dari satu stop states.

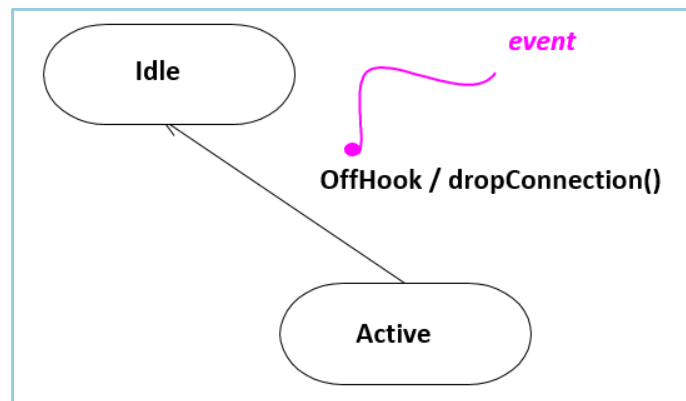


Gambar 5.38 Contoh diagram state

Sumber : (Gunadarma.ac.id, 2018)

State merupakan abstraksi dari nilai – nilai atribut dan asosiasi dari sebuah objek, state merepresentasikan kondisi dari sebuah objek pada periode waktu tertentu dan berhubungan dengan suatu interval waktu antara dua event. Respons

terhadap event dapat tergantung kepada state suatu objek. Event merupakan spesifikasi dari suatu kejadian tertentu, dan kita dapat memodelkan sesuatu kejadian sebagai event.



Gambar 5.39 Contoh event

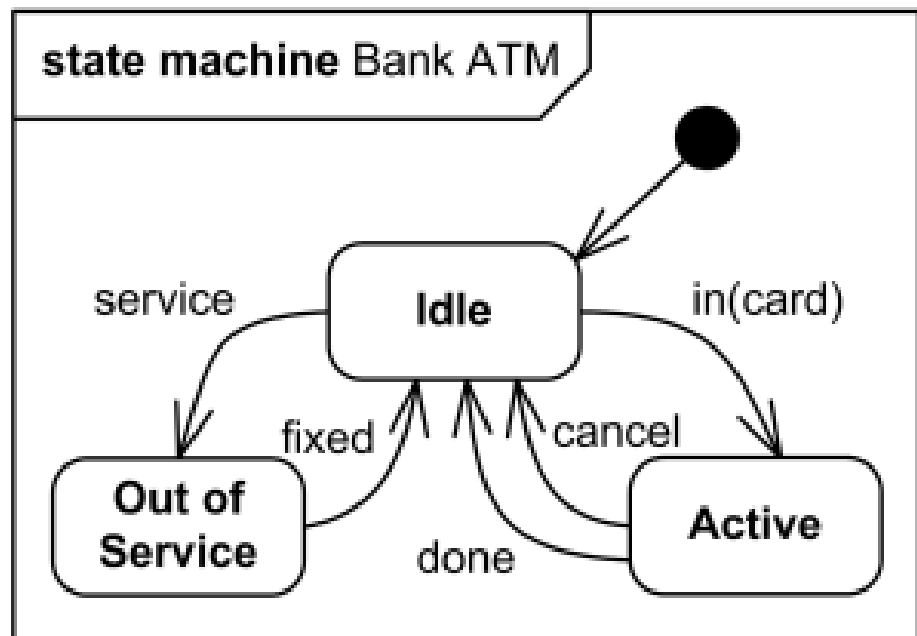
Sumber : (Gunadarma.ac.id, 2018)

Event dapat dikategorikan menjadi dua yaitu internal event (event yang berasal dari dan menuju ke objek pada sistem), eksternal event (event yang berasal dari aktor ke sistem atau sebaliknya).

State Machine Diagram

State machine adalah behavior yang menggambarkan urutan state dari objek sepanjang waktu hidupnya. State Machine Diagram digunakan untuk menggambarkan perubahan status atau

transisi status dari sebuah mesin atau sistem atau objek (Rosa A. S, 2018).







Gambar 5.40 Contoh State Diagram

Sumber : (uml-diagrams.org, State Machine Diagrams, 2018)

Berikut ini komponen – komponen dasar yang ada dalam state machine diagram :

Tabel 5.3 Komponen – komponen dasar pada state machine

NO	GAMBAR	NAMA	KETERANGAN
I		<i>State</i>	State atau status adalah keadaan sistem pada waktu tertentu. State dapat berubah

			jika ada event tertentu yang memicu perubahan tersebut
2		<i>Initial State</i>	Start atau initial state adalah state atau keadaan awal pada saat sistem mulai hidup.
3		<i>Final State</i>	Final State adalah state keadaan akhir dari daur hidup suatu sistem
4	Event 	<i>Event</i>	Event adalah kegiatan yang menyebabkan berubahnya status mesin.

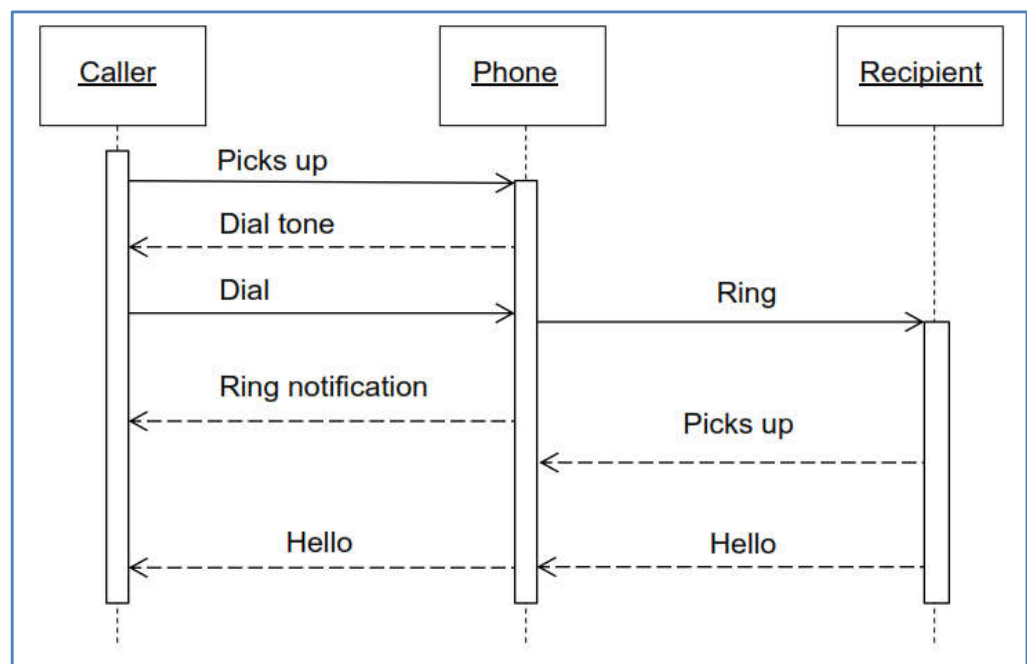
b. Sequence Diagram

Sequence diagram menggambarkan perilaku objek pada use case dengan mendiskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Untuk membuat sequence diagram perlu diketahui terlebih dahulu objek – objek yang terlibat dalam use case dan skenario perannya.

Bagian kunci dari diagram sequence adalah sebagai berikut (Fowler, 2018) :

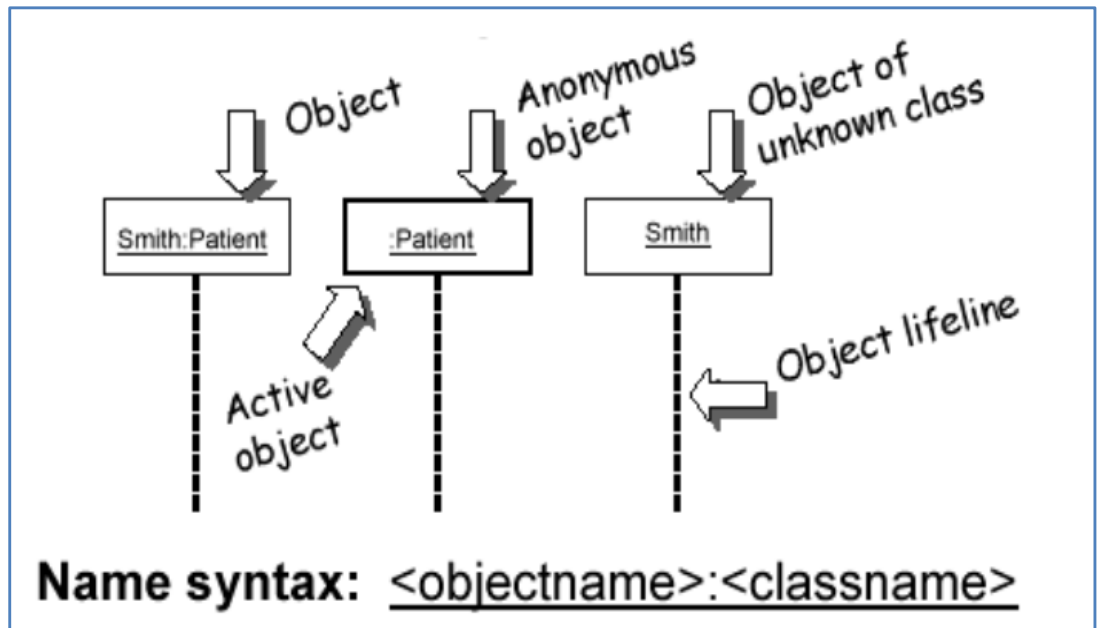
- 1) Participant yaitu objek atau entitas yang melakukan kegiatan dalam diagram.
- 2) Message yaitu komunikasi antara participant dengan objek
- 3) The axes dalam diagram sequence :
 - a) Horizontal yaitu objek atau participant yang melakukan kegiatan
 - b) Vertical yaitu waktu

Berikut adalah contoh diagram sequence dari panggilan telepon :



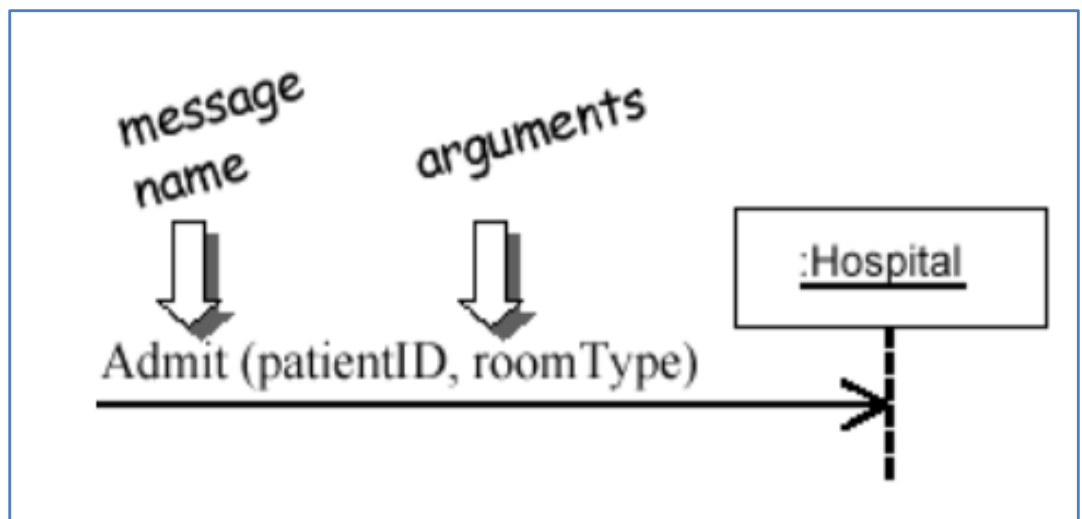
Gambar 5.4I Contoh diagram sequence dari perilaku panggilan telepon

Sumber : (Fowler, 2018)



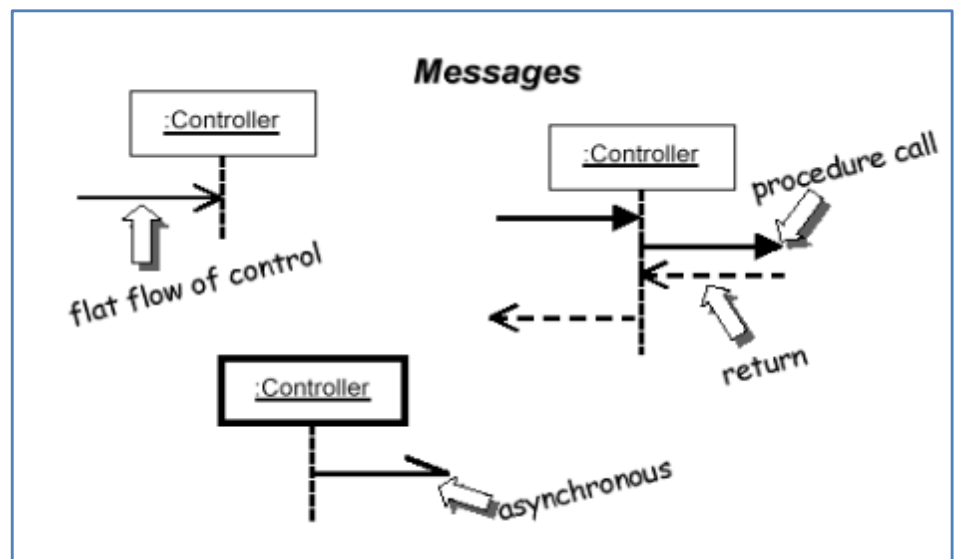
Gambar 5.42 Implementasi objek dalam
syntax

Sumber : (Fowler, 2018)



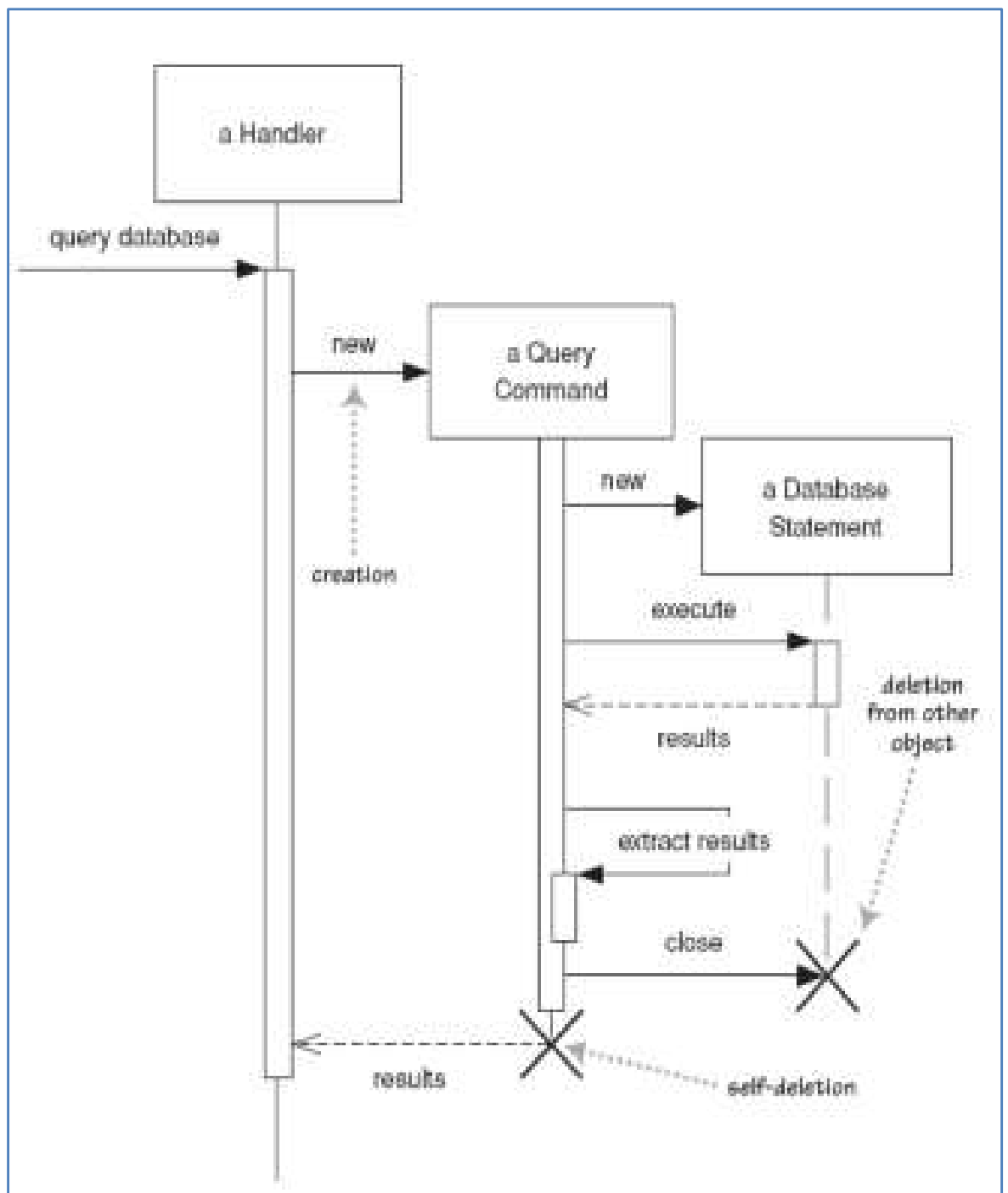
Gambar 5.43 Pesan antar objek

Sumber : (Fowler, 2018)



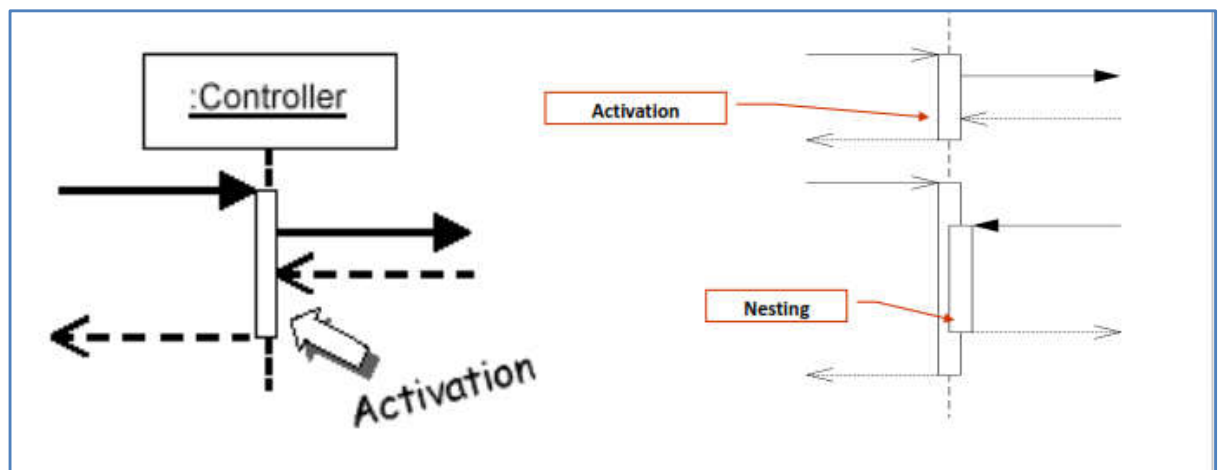
Gambar 5.44 Pesan yang dilanjutkan

Sumber : (Fowler, 2018)



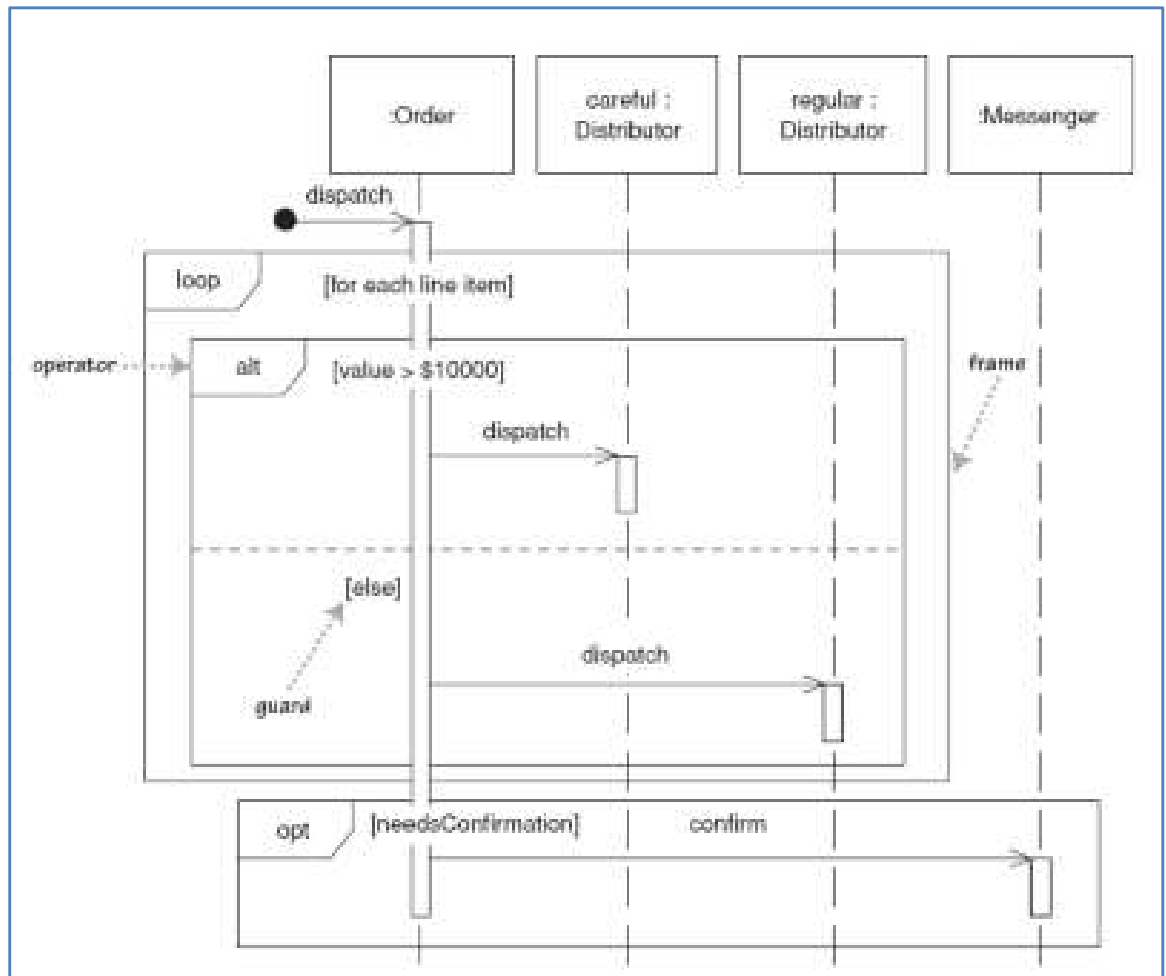
Gambar 5.45 Life time objek

Sumber : (Fowler, 2018)



Gambar 5.46 Activation

Sumber : (Fowler, 2018)



Gambar 5.47 Contoh loop dan selection

Sumber : (Fowler, 2018)

5.2.5. Pemodelan Kebutuhan – kebutuhan untuk Aplikasi – aplikasi Web

- I. Masukan Pemodelan Kebutuhan pada aplikasi WEB

Proses perangkat lunak generik versi cepat dapat digunakan pada rekayasa aplikasi – aplikasi web. Masukan – masukan untuk model – model kebutuhan pada aplikasi web adalah informasi

yang terkumpul melalui aktivitas – aktivitas komunikasi.

2. Keluaran dari Pemodelan Kebutuhan

Analisi kebutuhan pada aplikasi web menyediakan suatu mekanisme yang representasikan dan mengevaluasi isi – isi dan fungsi – fungsi yang dimiliki suatu web, merepresentasikan interaksi – interaksi yang dilakukan pengguna dan merepresentasikan lingkungan dan infrastruktur dimana aplikasi web itu berada. Masing – masing dari karakteristik dapat di representasikan sebagai sebuah model. Ada lima jenis kelas model yang utama untuk aplikasi web yaitu :

a. Model isi

Mengidentifikasi isi yang ada dalam aplikasi web. Model isi memuat di dalamnya elemen – elemen struktural yang menyediakan suatu bentuk tampilan untuk kebutuhan isi dari aplikasi web.

b. Model interaksi

Mendefinisikan cara para pengguna akan berinteraksi dengan aplikasi web. Model interaksi dapat disusun ke dalam elemen – elemen *use case*, *sequence diagram*, *state diagram* dan prototipe – prototipe antar muka pengguna.

c. Model fungsional

Mendefinisikan operasi – operasi yang akan diterapkan pada isi aplikasi web dan

mendeskripsikan fungsi – fungsi pemrosesan isi yang diperlukan oleh pengguna. Model fungsional menyelesaikan dua elemen proses yaitu fungsionalitas yang dapat diobservasi oleh pengguna dan operasi – operasi yang dimuat di dalam kelas – kelas analisis yang mengimplementasikan perilaku – perilaku yang berhubungan dengan kelas – kelas analisis yang mengimplementasikan perilaku – perilaku yang berhubungan dengan kelas – kelas analisis. Model fungsional dapat di susun kedalam diagram aktivitas.

- d. Model navigasi, mendefinisikan strategi navigasi keseluruhan untuk aplikasi web
- e. Model konfigurasi, mendeskripsikan lingkungan serta infrastruktur dimana aplikasi web akan berada.