

PERTEMUAN 12

IMPLEMENTASI DIAGRAM UML PADA STATECHART DIAGRAM DAN ACTIVITY DIAGRAM

A. TUJUAN PEMBELAJARAN

Pada pertemuan ini dijelaskan tentang pengertian Statechart Diagram, Activity Diagram, elemen-elemen yang ada pada diagram tersebut, dan tata cara pembuatan yang ada pada Diagram tersebut.

B. URAIAN MATERI

1. Statechart Diagram

Statechart Diagram adalah sebuah diagram yang mendeskripsikan memperlihatkan tentang perilaku sistem atau aplikasi yang dibuat dengan penggambaran yang dibuat kotak yang digambarkan sebuah objek, serta adanya penandaan panah yang berfungsi sebagai penunjuk perpindahan alur suatu state ke state lain, tetapi objek tersebut masih bersifat abstrak. Interaksi diagram menunjukkan pesan yang melewati antara obyek-obyek di dalam sistem pada jangka waktu yang pendek. Statechart diagram menyajikan simbol-simbol dan sejumlah ide untuk sebuah pemodelan. Tipe diagram ini berpotensi menjadi sangat kompleks pada waktu yang singkat. Diperlukannya Statechart Diagram untuk mempermudah analisa, pembuat dan pendevelope untuk membaca tingkah laku obyek yang berada disistem.

Dari Statechart Diagram inilah nantinya akan menghasilkan sebuah Activity Diagram, Activity Diagram itu sendiri merupakan use case khusus yang berada pada Statechart Diagram yang menjelaskan gambaran-gambaran aktivitas yang lebih konkrit, daripada yang digambarkan pada Statechart Diagram, yang masih berbentuk begitu abstrak. Meskipun kedua diagram sama-sama menggambarkan perilaku dan aspek yang berada pada suatu sistem, tetapi keduanya mempunyai perbedaan gambar permodelan ketika diagram dibuat. Pada Activity Diagram dipakai untuk pemodelan yang menggambarkan langsung proses bisnis yang dimana beberapa objek


langsung berpartisipasi pada suatu sistem. Sedangkan pada Statechart diagram cenderung digunakan untuk memodelkan siklus riwayat hidup pada sebuah objek reaktif. Sebuah objek reactive menyediakan sebuah konteks untuk suatu statechart. Konteksnya yaitu :




- Respon terhadap *external events* (even atau kejadian yang terjadi diluar konteks objek).
- Mempunyai model siklus hidup yang jelas sebagai progress suatu statem transisim dan *events*/kejadian.
- Memiliki sifat yang tergantung pada sifat sebelumnya.

Sebuah statechart terdiri dari 1 buah state mesin untuk setiap objek reaktif. Pada dunia nyata yang penuh objek reaktif dapat dimodelkan menggunakan state mesin. Pada pemodelan objek orientasi, dapat menggunakan state mesin untuk memodelkan tingkahlaku yang dinamis terhadap objek reaktif seperti *Classes*, *Use Cases*, *subsystems*, sistem keseluruhan. Namun state mesin umumnya digunakan untuk memodelkan tingkah laku sebuah class yang dinamis. Selanjutnya akan dijelaskan elemen-elemen yang ada pada statechart diagram.

a. Elemen pada Statechart Diagram

Dibagian ini akan dijelaskan elemen-elemen apa saja yang dibutuhkan ketika membuat sebuah Statechart Diagram.

<p>Simbol Transisi</p> <p>Transisi merupakan gambaran sebuah panah yang berfungsi sebagai penunjuk arah jalur dari suatu state ke state selanjutnya. Transisi tidak hanya menunjuk ke arah state lain, tetapi dapat difungsikan juga untuk menunjuk ke arah baliknya atau diri sendiri.</p>	
--	--

<p>Simbol Initial State</p> <p>Initial state biasanya berfungsi sebagai penanda awal berjalannya suatu sistem yang digambarkan pada statechart diagram itu sendiri.</p>	 <p>Initial state</p>
<p>Simbol Final State</p> <p>Simbol ini menunjukkan berakhirnya sebuah aktivitas yang terjadi pada sebuah sistem yang digambarkan pada Statechart Diagram.</p>	 <p>Final state</p>
<p>Simbol State</p> <p>Simbol ini menunjukkan suatu aktivitas yang terjadi pada suatu state yang berada pada Statechart Diagram.</p>	 <p>State</p> <p>A simple state</p>

Tabel 1. Tabel simbol pada Statechart Diagram

Pada table diatas dijelaskan elemen-elemen apa saja yang ada ketika membuat sebuah Statechart Diagram, kedalam sebuah perancangan.

b. Tujuan Perancangan Statechart Diagram

Statechart diagram merupakan sebuah diagram UML yang dipakai untuk memodelkan sifat dinamis yang terjadi didalam sistem. Diagram tersebut mendefinisikan state yang berbeda dari suatu objek selama siklus hidupnya berlangsung dan state ini diubah karena sebuah *events*/kejadian. Baik itu secara eksternal maupun internal.

Statechart diagram, menggambarkan aliran control dari satu state ke state lain. State didefinisikan dengan suatu kondisi dari satu objek dan state tersebut akan berubah ketika terpicu oleh kejadian/*events*. Tujuan terpenting sebuah Statechart Diagram adalah untuk memodelkan keadaan suatu objek dari awal pembuatan hingga dibuang. Berikut ini tujuan utama Statechart Diagram:

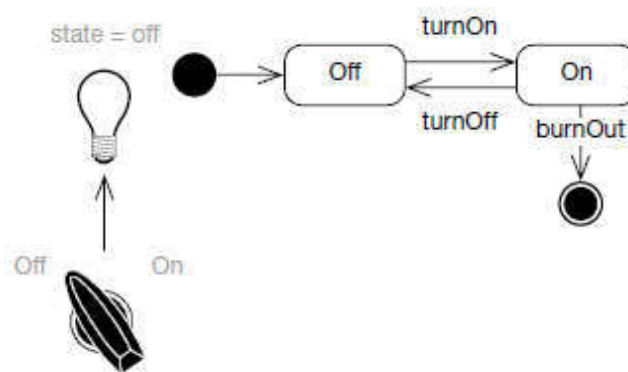
- 1) Untuk memodelkan aspek dinamis yang terjadi pada suatu sistem.
- 2) Untuk memodelkan siklus jangka hidup reaktif yang terjadi pada sistem.
- 3) Untuk mendeskripsikan berbagai status pada suatu state/objek.

Selanjutnya akan dijelaskan studi kasus yang terjadi didunia nyata tentang tata cara penggambaran statechart diagram dengan menggunakan contoh sebuah proses sistem yang terjadi disuatu aktivitas didunia nyata.

c. Contoh state Mesin dan kelasnya

Untuk sebuah class terdiri dari satu state machine semua modelnya ditransisikan antara states terhadap seluruh objek pada suatu class yang merespon sebuah kejadian/*events*. *Events* itu sendiri adalah sebuah pesan yang dikirim oleh objek lain, meskipun dibeberapa kasus objek mungkin terhubung dan menghasilkan suatu kejadian sebagai respon terhadap waktu.

Object yang diberikan pada class dapat berpartisipasi dibanyak use case, dengan fungsi tersebut dapat melihat state mesin untuk sebuah model class dari tingkah laku objek yang berada dikeseluruhan use case. Berikut salah satu contoh pemodelan state mesin simple yang berada disuatu aktifitas kejadian didunia nyata. Contoh ketika menyalakan lampu



Gambar 3. Contoh state mesin menyalakan lampu bohlam

Setiap state mesin diawali dengan simbol initial state (lingkaran tebal) yang mengindikasikan berawalnya sebuah mesin, dan terhenti disimbol lingkaran yang keseluruhan titik tidak tebal, yang mengindikasikan state suatu mesin berakhir. Ketika tombol switch dinyalakan maka akan menjalankan suatu kejadian/*events* yang kalo dibahasakan dengan *pseudo-state* akan menjalankan state event *turnOn*. Pada suatu state mesin, kejadian dipertimbangkan secara seketika. Di lain kata, hanya membutuhkan waktu kosong untuk suatu kejadian yang dikirim dari sakelar ke lampu bola. Kejadian seketika memberikan penyederhanaan yang penting untuk teori state mesin yang membuatnya lebih gampang berinteraksi atau menurut. Tanpa kejadian seketika kita akan melakukannya dengan beberapa kondisi yang besaingan, Yang dimana dua kejadian bersaing dari sumber untuk mencapai objek reaktif yang sama. Dibutuhkan model persaingan kondisi pada sebuah state mesin.

Bola lampu menerima event *turnOn* dan state berubah menjadi *On* ketika merespon sebuah kejadian/*events*. Ketika menerima kejadian state *turnoff* yang dikirimkan ke bola lampu, maka state berubah menjadi *off* dan lampu menjadi padam.

Pada point ini, kejadian *burnout* mungkin terjadi, dan dan menyelesaikan state mesin.

d. State

Pada sebuah UML state didefinisikan sebagai, sebuah kondisi atau situasi yang terjadi ketika hidupnya sebuah objek yang dimana tertarik dengan sebuah kondisi, menjalankan sebuah aktivitas, atau menunggu suatu kejadian/*event*. Sebuah state pada objek bervariasi dari waktu ke waktu, tetapi titik tertentu ditentukan oleh :

- Nilai atribut pada suatu objek.
- Hubungannya terhadap objek lain.
- Aktifitas yang dijalankannya.

Namun, dari sudut pandang penggunaan terhadap bola lampu, state yang dipakai untuk membedakannya hanya ada state *On* dan *Off*. Ini adalah suatu hal penting kunci kesuksesan untuk membuat pemodelan

Statechart Diagram. Sebagai contoh dibawah

```
class Color
{
    int red;
    int green;
    int blue;
}
```

Gambar 4. Contoh penggambaran nilai atribut.

Seandainya kita asumsikan jika red, green dan blue masing-masing dapat menerima nilai antara 0-225 lalu, yang hanya berdasarkan dari nilai atributnya, berarti tiap objek pada suatu class mempunyai 256 kemungkinan jika dikalikan ke tiganya $256 \times 256 \times 256 = 16777216$ kemungkinan state yang keluar dan itu akan menjadi beberapa statechart. Namun, secara fundamental bermunculan pertanyaan pada diri sendiri secara fundamental. Apa kunci semantic perbedaan antara tiap-tiap state? Tentu saja jawabannya tidak ada. Tiap 16777216 kemungkinan state hanya merepresentasikan tiap warna, hanya itu saja. Faktanya, statechart untuk tiap classnya menjadi membosankan, seperti yang dilihat hanya kemungkinan tiap state saja yang muncul tanpa menjelaskan proses sistem apa yang sedang dilakukan. Singkatnya harus ada perbedaan yang membuat perbedaan semantic antar state untuk memodelkannya pada suatu state mesin.

e. State Syntax

Berikut ini sebuah gambaran state syntax disini akan dijelaskan secara mendetail elemen-elemen apa saja yang ada tiap suatu state ketika merancang suatu state diagram.



Gambar 5. Contoh state ketika memasukan suatu password

Entering password untuk nama state, entry/display password dan

exit/validate perintah untuk menginput dan keluar, alphaKeypress/echo dan help/display merupakan transisi internal yang terjadi didalam sistem, do/get merupakan aktifitas internal yang terjadi pada suatu state. Tiap sebuah state dijelaskan secara abstrak fungsi-fungsi apa saja yang terjadi ketika kita menginput sebuah password.

Tiap state terdiri dari nol tindakan atau lebih dan sebuah aktifitas. Tindakan terbagi menjadi seketika dan tak dapat terganggu, sedangkan aktifitas membutuhkan waktu dan dapat diganggu. Setiap aksi pada state diasosiasikan dengan transisi internal yang memicunya sebuah kejadian atau *event*. Bisa berupa beberapa nomor aksi dan internal transisi didalam suatu state.

f. Transitions

Transisi mempunyai syntax yang simple yang digunakan untuk menggunakan fungsi *external transitions* (yang ditunjukan dengan arah panah) ataupun *internal transitions*. Tiap transisi mempunyai tiga opsi elemen:

- 1) Sebuah kejadian – Sebuah kejadian internal atau eksternal yang memicu transisi.
- 2) Kondisi bertahan – Sebuah ekspresi Boolean yang harus bernilai “true” sebelum transisi terjadi.
- 3) Sebuah tindakan – sebuah lembar kerja yang mengasosiasikan dengan sebuah transisi, dan terjadi ketika transisi menyala.

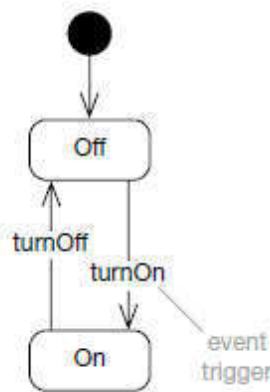
Berikut gambaran sederhana transisi yang terjadi pada suatu state diagram.



Gambar 6. Gambar sebuah transisi pada Statechart

g. Event/Kejadian

Pendefinisian *event/kejadian* pada uml, dispesifikasikan sebagai suatu hal yang patut diperhatikan yang memiliki lokasi didalam ruang dan waktu. *Event/kejadian* memicu transisi yang ada distate mesin. Event ditunjukan diluar pada sebuah transisi. Berikut akan dijelaskan pada gambar.

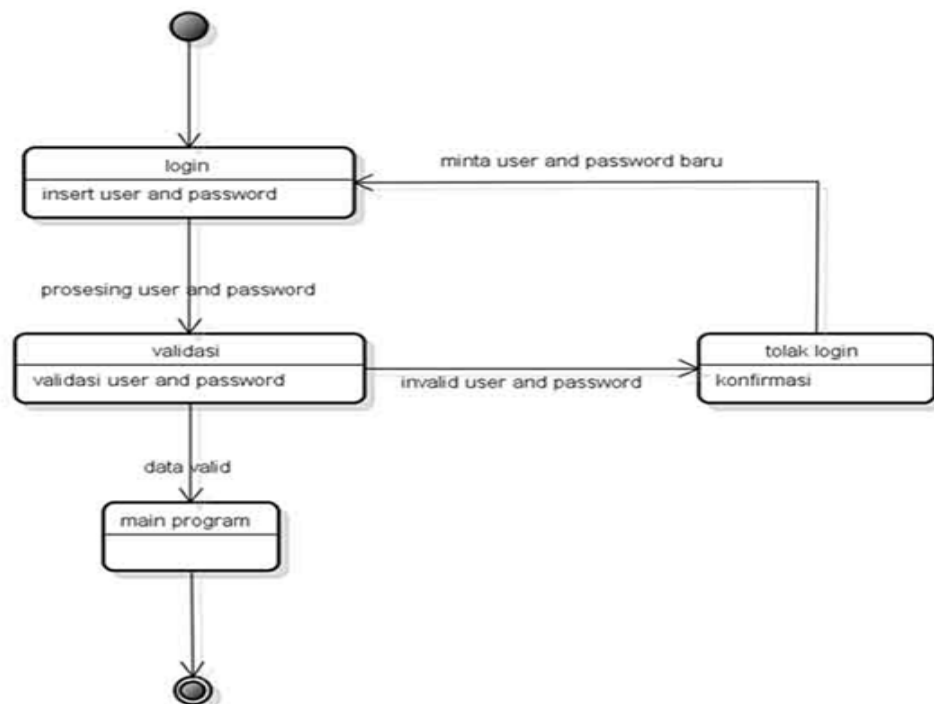


Gambar 7. Gambar event pada suatu state mesin

Dijelaskan pada gambar diatas yang memicu perubahan suatu state dikarenakan adanya suatu transisi yang dapat menimbulkan perubahan status yang terjadi pada suatu state, yang sebelumnya state berstatus off, kini terpicu oleh sebuah transisi “turnOn” menjadi On.

h. Studi Kasus Statechart Diagram

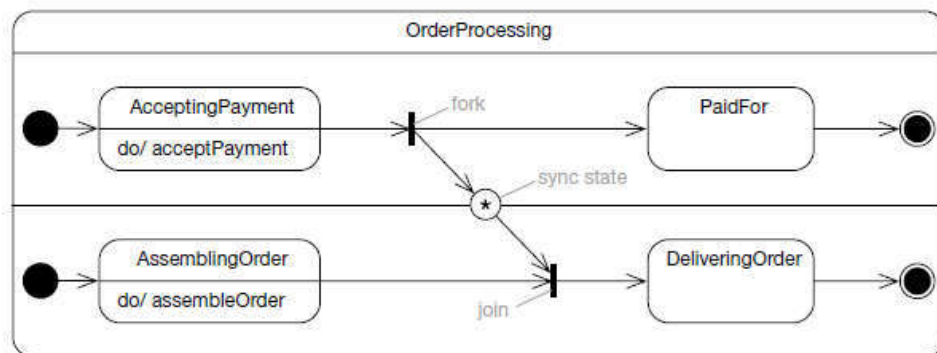
Berikut akan ada contoh studi kasus yang terjadi pada suatu sistem pada fungsi state login, akan dijelaskan elemen-elemen apa saja yang terjadi dibalik adanya fungsi login kedalam suatu sistem.



Gambar 8. Statechart diagram untuk melakukan login pada aplikasi.

Bisa dilihat pada statechart diagram, menggambarkan tahap-tahap suatu aktivitas pada sebuah aplikasi ketika ingin menjalankan fungsi yang ada pada aplikasi. Dari sebuah fungsi login dijelaskan bahwa ketika ingin melakukan login diharuskan mempunyai username dan password yang valid, ketika password kita tidak sesuai maka yang terjadi sistem akan menagih kembali password yang harus dimasukan sampai benar jika tidak akses akan ditolak dan sistem akan kembali kehalaman login utama. Selanjutnya jika berhasil masuk maka sistem akan berjalan dan mempersilahkan user untuk mengakses menu utama.

Selanjutnya akan dijelaskan studi kasus pengkomunikasian penggunaan state yang syncron terhadap dua state yang berbeda tetapi salih berhubungan berikut contoh penggambaran yang terjadi pada sistem pemesanan barang.



Gambar 9. Statechart diagram pada sistem pemesanan

Kedua statechart tersebut berbeda dan mempunyai tugas masing-masing tetapi saling terhubung satu sama lain, ketika sistem menerima pembayaran selanjutnya akan menuju ke state berikutnya yaitu kepembayaran untuk kemudian diakhiri dengan final state, Disitu sudah jelas pembayaran sudah dibayarkan untuk kebutuhan konsumen yang telah dipesan. Lalu state yang dibawah ada state merancang pemesanan, pesanan akan dirancang ketika pembayaran telah berhasil dilakukan, jika tidak proses tidak dapat dilakukan. Digambar diatas terlihat ada hubungan *sync state* yang berfungsi mengsinkronisasikan suatu perintah ke perintah lain distate yang berbeda, selanjutnya transisi tersebut tersinkron dengan state yang ada Digambar bawah karena saling terhubung, dan kebutuhan sudah memenuhi, maka selanjutnya ke state berikut terjadilah pengiriman

barang, dan diakhiri dengan final state .

2. Activity Diagram

Activity Diagram adalah bagian terpenting pada UML, yang menjelaskan aspek yang mudah dipahami dari sistem dan menggambarkan suatu gambaran aktivitas pada sistem yang jelas membuat penggunaanya lebih memahami alur fungsi pada suatu rancangan aplikasi yang ingin dirancang. Elemen yang digunakan pun sama seperti yang ada pada statechart diagram hanya saja gambaran aktivitas pada Activity Diagram lebih diperjelas. Tujuan dibuatnya adalah untuk menangkap perilaku dinamis yang terjadi pada sistem dengan cara menunjukkan aliran deskripsi dari satu aktifitas ke aktifitas lainnya. Secara umum tujuan pembuatan Activity Diagram dapat dijelaskan sebagai berikut:

- a. Menggambarkan alur aktifitas pada sistem
- b. Menggambarkan urutan aktivitas dari awal sampai akhir/dari satu aktifitas ke aktifitas lainnya.
- c. Menggambarkan percabangan aktifitas yang terjadi pada suatu sistem.

Activity Diagram dapat dilampirkan kesemua elemen permodelan apapun tujuan permodelan dan perilaku elemen tersebut. Activity Diagram dapat dilampirkan kedalam:

- a. Use Case;
- b. Class;
- c. Tampilan atau interface;
- d. Komponen;
- e. Node;
- f. Collaborasi atau diagram campuran;
- g. Operasi dan metode.

Akan sangat diuntungkan ketika menggunakan Activity Diagram untuk proses permodelan bisnis dan workflow. Umumnya Activity Diagram adalah sebuah flowchart operasi, perlu dipertimbangkan bahwa sebenarnya sumber kode digunakan untuk mengoperasikan. Yang nantinya semua aktifitas tersebut akan dipresentasikan secara lengkap dan ringkas melalui Activity Diagram.

a. Action State

Activity Diagram memiliki action state yang dimana state ini mendeskripsikan suatu proses yang sedang dilakukan distate tersebut. Digambarkan persegi Panjang. Berikut contoh sebuah state pada Activity diagram.

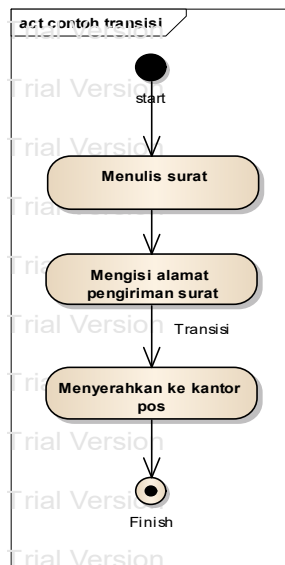


Gambar 29. Gambar sebuah Action state

Pada Activity Diagram dideskripsikan sebuah proses bisnis yang terjadi pada rancangan aplikasi, untuk mempermudah memahami hal-hal apa saja yang terjadi pada suatu state yang berada didalam ruang lingkup sistem. Menurut Spesifikasi UML Action State hanyalah versi penyederhanaan yang lebih umum terhadap elemen yang seringkali disebut sebagai state. Faktanya aksi tersebut dituliskan secara singkat yang mempunyai sebuah aksi dan setidaknya mempunyai transisi kearah luar.

b. Transisi

Sebelumnya dijelaskan seperti apa suatu state yang ada pada elemen Activity Diagram, selanjutnya akan dijelaskan perpindahan-perpindahan aktifitas atau transisi yang terjadi pada suatu Activity Diagram yang terlihat Digambar ini.

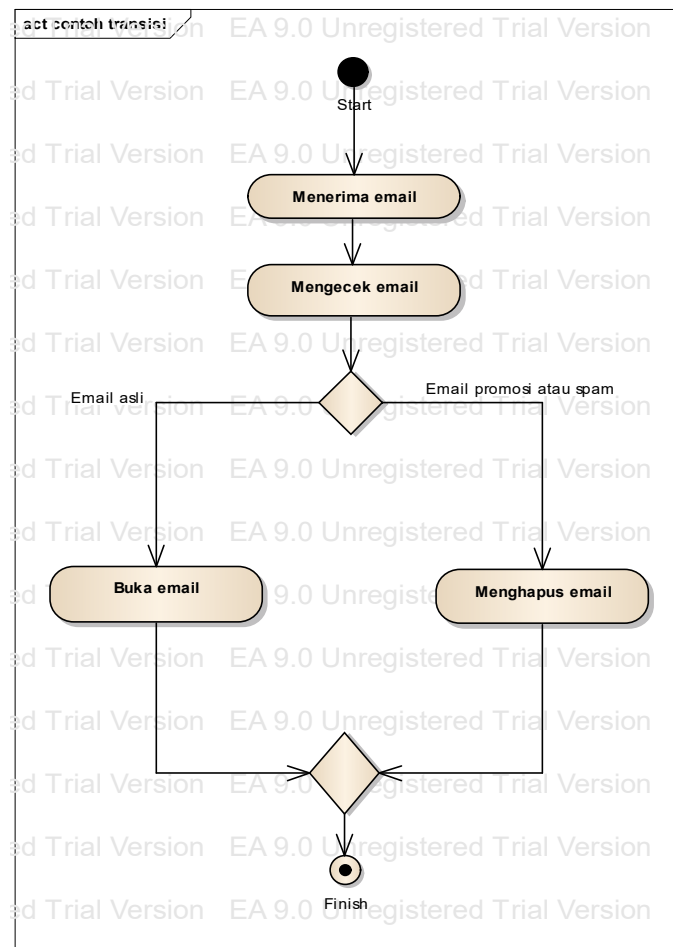


Gambar 30. Contoh perpindahan antara sebuah state ke state yang lain.

Gambar diatas menunjukan alur perpindahan sebuah aktivitas yang terjadi ketika ingin mengirimkan sebuah surat, yang diawal dengan state inisial *start* dan diakhiri dengan state final *finish*. Arah panah menunjukan sebuah transisi atau alur perpindahan sebuah proses dari satu state ke state lainnya.

c. Decision/keputusan

Kali ini akan dijelaskan tentang cara penggunaan fungsi decision yang ada pada activity diagram, diperlukan jika suatu sistem mempunyai lebih dari satu state pada kondisi tertentu, untuk mengelompokkan sebuah keputusan. Studi kasus kali ini mengambil contoh dari gambaran aktifitas ketika kita membuka sebuah pesan *email*.



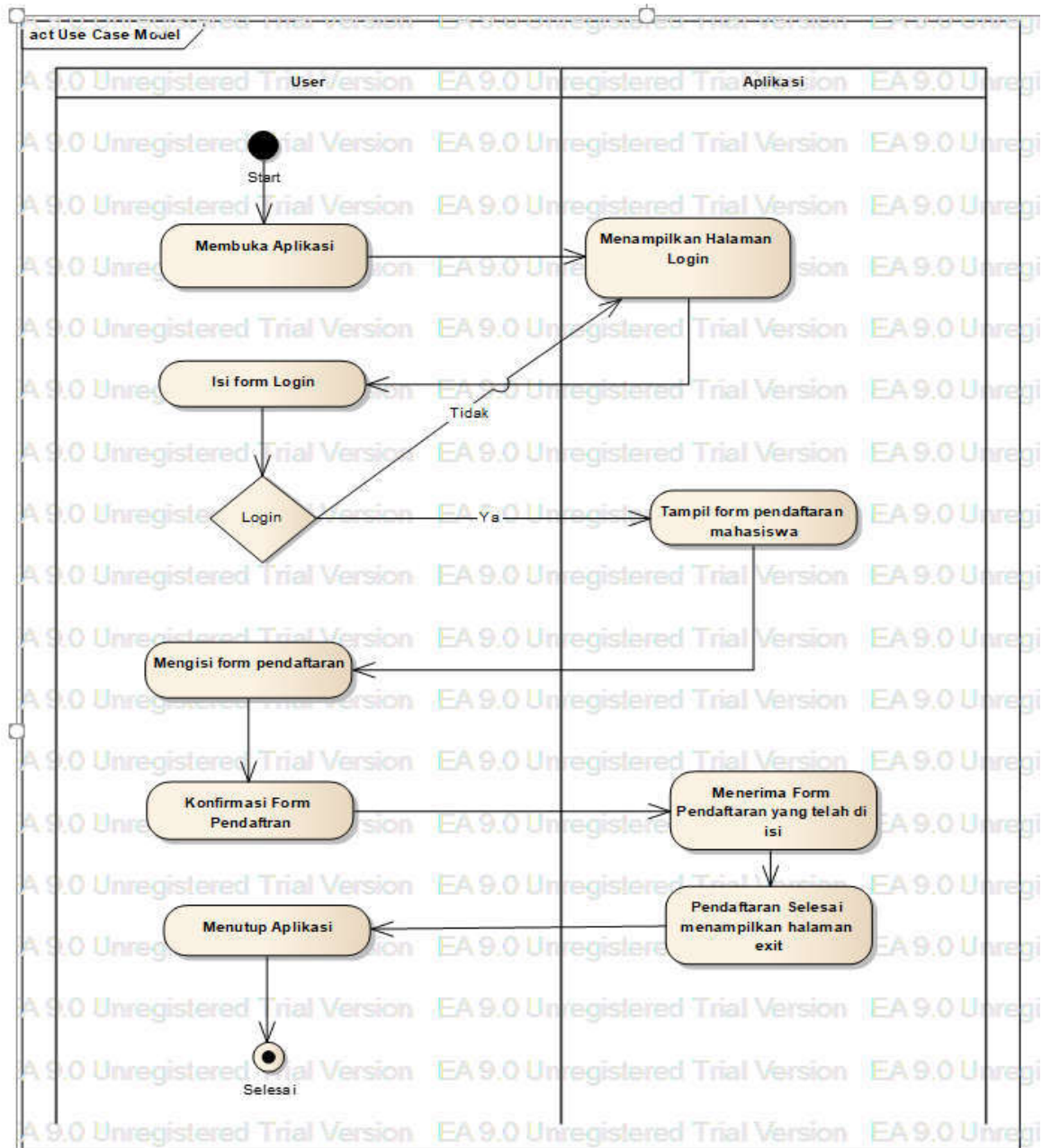
Gambar 31. Contoh penggunaan fungsi decision pada pembuatan Activity Diagram

Jika kita lihat terdapat simbol belah ketupat yang menandakan adanya keputusan yang hanya dapat diambil salah satu ketika sebuah sistem berjalan, seperti gambar diatas ketika membuka email, jika pesan baru yang didapat adalah surat asli, maka email akan dibuka, ketika email atau pesan baru yang didapat ternyata hanya iklan promosi atau spam, maka keputusan yang dibuat email akan diabaikan dan dihapus. Inilah sebuah gambaran fungsi decision yang terjadi didalam suatu Activity

Diagram.

d. Contoh Studi Activity Diagram

Berikut ini sebuah contoh studi kasus Activity Diagram pada sistem pendaftaran mahasiswa baru.



Gambar 32. Activity Diagram penerimaan mahasiswa baru

Pada Activity diatas, state diklasifikasikan antara pengguna atau user dengan aplikasi tersebut. Ini berfungsi untuk mengidentifikasi perbedaan ruanglingkup aktivitas-aktivitas yang terjadi pada tiap state. Adanya pengulangan state dikarenakan kebutuhan yang perlu diinput belum memenuhi persyaratan, sehingga data yang akan masuk nantinya sudah terfilter oleh sistem dan otomatis semua data yang masuk pasti sudah memenuhi kualifikasi kebutuhan pada suatu sistem.

C. SOAL LATIHAN/TUGAS

1. Carilah Definisi pengertian Statechart Diagram dan Activity Diagram selain yang disebutkan diatas!
2. Jelaskan Perbedaan yang ada pada Statechart Diagram dan Activity Diagram menurut kalian!
3. Buatlah Studi kasus contoh untuk pembuatan Statechart Diagram!
4. Buatlah Studi kasus contoh untuk pembuatan Activity Diagram!

D. REFERENSI

Alan Dennis, Barbara Haley Wixom, David Tegarden (2015). *Systems Analysis & Design: an object-oriented approach with UML*. 5th edition. New Jersey: John Wiley & Sons, Inc

Jim Arlow, Ila Neustadt (2002). *UML And The Unified Process Practical Object-Oriented Analysis & Design*. Great Britain: Pantek Arts,Ltd, Maidstone, Kent

GLOSARIUM

Events merupakan sebuah kejadian atau peristiwa yang terjadi didalam kelas pada suatu sistem yang berjalan.

Statechart Diagram merupakan diagram *UML* yang dipakai pada salah satu fungsi perancangan pada sistem.

External Events merupakan sebuah kejadian atau peristiwa yang terjadi diruang lingkup suatu sistem tetapi memiliki keterkaitan pada sistem.

Class merupakan sebuah kelas dan salah satu elemen yang ada pada diagram *UML*.

State merupakan sebuah pernyataan pada suatu elemen pada diagram *UML*.

Pseudo-state merupakan bahasa Pseudocode yang menggambarkan algoritma pernyataan.

Pseudocode merupakan metode penulisan yang memakai bahasa sederhana, untuk menjelaskan penggambaran sebuah algoritma.

TurnOn merupakan sebuah contoh pseudo-state yang ada pada suatu sistem.

TurnOff merupakan sebuah contoh pseudo-state yang ada pada suatu sistem.

Activity Diagram merupakan diagram *UML* yang digunakan sebagai perancangan sistem.

Initial State adalah sebuah state yang menggambarkan awal bermulanya berjalannya pada suatu sistem.

Finish State adalah sebuah state yang menggambarkan berakhirnya alur perjalanan pada suatu sistem.

Decision State adalah sebuah state yang menggambarkan suatu sistem harus mengambil sebuah keputusan yang harus diambil ketika memiliki kondisi tertentu.