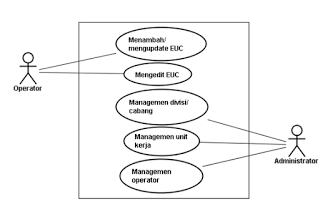
[Mengenal Use Case Diagram](http://nickizoner.blogspot.co.id/2013/06/mengenal-use-case-diagram.html)

Posted by Nicki Hermanto Putro on 23.00 with [8 comments](http://nickizoner.blogspot.co.id/2013/06/mengenal-use-case-diagram.html#comment-form)

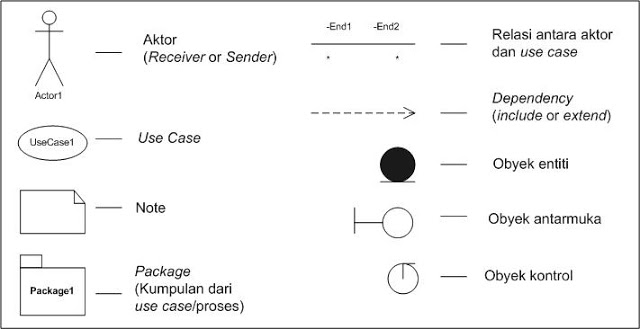
[](http://3.bp.blogspot.com/-B8ONnH7UFZI/UbCt4CMy1VI/AAAAAAAAA4U/I46VbCHhTkU/s1600/usecase_diagram.png)

**Pengertian :**

*Use-case diagram* merupakan model diagram UML yang digunakan untuk menggambarkan *requirement*fungsional yang diharapkan dari sebuah sistem. *Use-case diagram* menekankan pada “siapa” melakukan “apa” dalam lingkungan sistem perangkat lunak akan dibangun. *Use-case diagram* sebenarnya terdiri dari dua bagian besar; yang pertama adalah *use case diagram* (termasuk gambar *use case dependencies*) dan u*se case description*.   
  
*Use-case* diagram adalah gambaran *graphical* dari beberapa atau semua *actor*, *use-case*, dan interaksi diantara komponen-komponen tersebut yang memperkenalkan suatu sistem yang akan dibangun. *Use-case diagram*menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang yang berada di luar sistem. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar.

*Use-case diagram* dapat digunakan selama proses analisis untuk menangkap *requirement system* dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use-case diagram* berperan untuk menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa *use-case diagram*. Kebutuhan atau *requirements system* adalah fungsionalitas apa yang harus disediakan oleh sistem kemudian didokumentasikan pada model *use-case* yang menggambarkan fungsi sistem yang diharapkan (*use-case*), dan yang mengelilinginya (*actor*), serta hubungan antara actor dengan *use-case*(*use-case diagram*) itu sendiri.

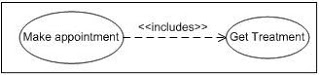
● *Use case class* digunakan untuk memodelkan dan menyatakan unit fungsi/layanan yang disediakan oleh sistem (or bagian sistem: subsistem atau class) ke pemakai.  
  
● *Use case* dapat dilingkupi dengan batasan sistem yang diberi label nama sistem.  
  
● *Use case* adalah sesuatu yang menyediakan hasil yang dapat diukur ke pemakai atau sistem eksternal.   
  
**Karakteristik :**  
– *Use cases* adalah interaksi atau dialog antara sistem dan *actor*, termasuk pertukaran pesan dan tindakan yang dilakukan oleh sistem.  
  
– *Use cases* diprakarsai oleh *actor* dan mungkin melibatkan peran *actor* lain. *Use case*s harus menyediakan nilai minimal kepada satu *actor*.  
  
– *Use cases* bisa memiliki perluasan yang mendefinisikan tindakan khusus dalam interaksi atau *use case* lain mungkin disisipkan.  
  
– *Use case class* memiliki objek *use case* yang disebut skenario. Skenario menyatakan urutan pesan dan tindakan tunggal.

[](http://1.bp.blogspot.com/-fR-KuHDqyas/UbLj0ZhyJsI/AAAAAAAAA40/SIx8D-BenuE/s1600/Use+Case+Notasi.jpg)

**Berikut adalah cara menggambar use-case diagram:**

Catatan:  
Sebelum membuat *use-case diagram*, sebaiknya diawali dengan membuat FDD terlebih dahulu. Hal ini sekedar untuk membantu mengidentifikasi proses-proses dalam sistem.  
1) Mulai dengan mendaftarkan aktor yang berhubungan dengan *Use-case*, baik sebagai *sender* maupun*receiver*.  
2) Komponen dalam *use-case diagram* adalah Nama *Use-case*, Deskripsi *Use-case* dan Pelaku yang berpartisipasi dan perannya.  
3) Temukan *dependency* yang mendemonstrasikan hubungan semantik antara dua *Use-case*. Jika *Use-case* “A” berubah dapat mengakibatkan *Use-case* “B” akan berubah pula.   
Ada 2 macam *dependency* yang perlu diperhatikan yaitu*include* dan *extend*.

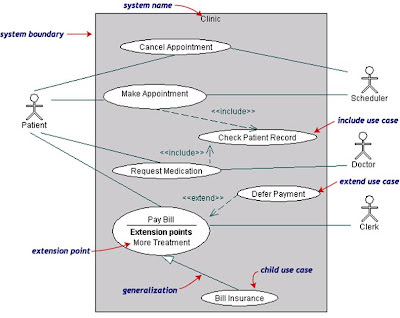
*Dependency include*:  
Sebuah *Use-case* dapat meng-*include* fungsionalitas *Use-case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *Use-case* yang di-*include* akan dipanggil setiap kali *Use-case* yang meng-*include* dieksekusi secara normal. Sebuah *Use-case* dapat di-*include* oleh lebih dari satu *Use-case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.  
Contoh Use-case (*include*)

[](http://4.bp.blogspot.com/-egtp-ipsgUg/UbLpFrnqqvI/AAAAAAAAA5E/maEy_xOSzaA/s1600/Use+Case-Include.jpg)

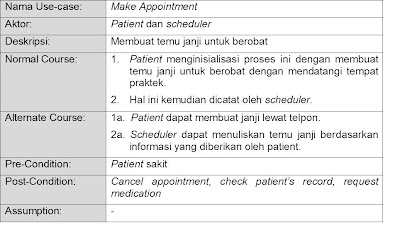
Keterangan:  
Pasien harus membuat temu janji sebelum diberikan perawatan yang diperlukan untuk mengobati penyakit yang dideritanya. *Use-case* “*Make Appointment*” meng-*include* fungsionalitas dari *Use-case* “*Get Treatment*” sebagai bagian dari proses saat dieksekusi.  
  
*Dependency extend*:  
Sebuah *Use-case* juga dapat meng-*extend Use-case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *Use-case* menunjukkan bahwa *Use-case* yang satu merupakan spesialisasi dari yang lain.  
  
                                         
  
Contoh *Use-case (extend)*:

[](http://2.bp.blogspot.com/-Rd1lJ1-yyVw/UbLp8UKXARI/AAAAAAAAA5U/eA42YDjkI2s/s1600/Use+Case-Extend02.jpg)

Keterangan:  
Setelah pasien membuat temu janji dengan dokter, pasien ini tiba-tiba mendapat halangan sehingga tidak dapat memenuhi janjinya. Oleh karena itu, pasien ini membatalkan janji yang sudah dibuat. Ini merupakan contoh dari kasus*extend* dimana “*Make Appointment*” adalah *base Use-case* dan “*Cancel Appointment*” merupakan*extended Use-case*.  
  
Gambar Contoh Interaksi Antara Aktor dan Sistem (*Use-case*):

[](http://3.bp.blogspot.com/-890bTs4NNHo/UbL82WGJdmI/AAAAAAAAA5o/_nHQAGPMuCs/s1600/Use+Case+Interaction.jpg)

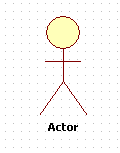
Berikut adalah Contoh *Use-case description*:

[](http://3.bp.blogspot.com/-_Wsb8dvOZaU/UbL9G67710I/AAAAAAAAA5w/lOgTVTUxTz0/s1600/Use+Case+Description.jpg)

Keterangan  
*Normal course*:   
Rangkaian kejadian yang terjadi sesuai harapan.  
*Alternate course*:   
Mendokumentasikan kelakuan *Use-case* jika terjadi *exception.*  
*Pre-condition*:   
Kondisi yang harus dipenuhi sebelum *Use-case* ini dijalankan. Hal ini dapat dilakukan dengan memberikan penjelasan singkat atau dapat pula berupa nama *Use-case*.  
*Post-condition*:   
Batasan pada keadaan sistem setelah *Use-case* ini diesksekusi dengan baik. Dapat berupa nama *Use-case*.

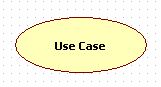
**Komponen Pembentuk *Use Case Diagram* :**  
**1. *Actor***

Pada dasarnya *actor* bukanlah bagian dari *use case diagram*, namun untuk dapat terciptanya suatu *use case diagram*diperlukan beberapa *actor*. *Actor* tersebut mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. Sebuah *actor* mungkin hanya memberikan informasi inputan pada sistem, hanya menerima informasi dari sistem atau keduanya menerima, dan memberi informasi pada sistem. *Actor* hanya berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*. *Actor*digambarkan dengan *stick man* . *Actor* dapat digambarkan secara secara umum atau spesifik, dimana untuk membedakannya kita dapat menggunakan *relationship*.

[](http://3.bp.blogspot.com/-fC2GLCx1if4/UbCuU0tSCXI/AAAAAAAAA4c/W9jzEqVYH6w/s1600/actor.png)

**2. Use Case**

Use case adalah gambaran fungsionalitas dari suatu sistem, sehingga customer atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.   
  
Catatan : *Use case diagram* adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.  
  
Cara menentukan *Use Case* dalam suatu sistem:  
a. Pola perilaku perangkat lunak aplikasi.  
b. Deskripsi tugas dari sebuah *actor*.  
c. Sistem atau “benda” yang memberikan sesuatu yang bernilai kepada *actor*.  
d. Apa yang dikerjakan oleh suatu perangkat lunak (\*bukan bagaimana cara mengerjakannya).

[](http://4.bp.blogspot.com/-RbtOggqgRik/UbCu6-2D57I/AAAAAAAAA4k/Jzjnx2sH1G8/s1600/use-case.png)

**Relasi dalam *Use Case*  
Ada beberapa relasi yang terdapat pada *use case diagram*:**

1. Association, menghubungkan link antar element.
2. Generalization, disebut juga inheritance (pewarisan), sebuah elemen dapat merupakan spesialisasi dari elemen lainnya.
3. Dependency, sebuah element bergantung dalam beberapa cara ke element lainnya.
4. Aggregation, bentuk assosiation dimana sebuah elemen berisi elemen lainnya.

**Tipe relasi/*stereotype* yang mungkin terjadi pada *use case* *diagram*:**

1. **include**, yaitu kelakuan yang harus terpenuhi agar sebuah *event* dapat terjadi, dimana pada kondisi ini sebuah use case adalah bagian dari use case lainnya.
2. **extends**, kelakuan yang hanya berjalan di bawah kondisi tertentu seperti menggerakkan roda.
3. **communicates**, mungkin ditambahkan untuk asosiasi yang menunjukkan asosiasinya adalah *communicates association* . Ini merupakan pilihan selama asosiasi hanya tipe *relationship* yang dibolehkan antara *actor* dan*use case*.

Sumber:

1. http://mitanovia.wordpress.com/belajar-yuk/uml/use-case-diagram/    diakses hari Kamis, 6 Juni 2013, 22:58 WIB.
2. http://kepenakwae.blogspot.com/2013/01/use-case-diagram-uml.html   diakses hari Sabtu, 8 Juni 2013, 15:05 WIB.