

# Rekayasa Perangkat Lunak

Modul 4

## Requirement Modeling

Disajikan Oleh :  
Irman Hariman, ST., MT.  
Email : [iirmanhariman@gmail.com](mailto:iirmanhariman@gmail.com)

# Requirement Modeling

- SKPL ; menetapkan persetujuan antara pelanggan dan kontraktor atau pemasok tentang apa yang akan dilakukan dan tidak dilakukan oleh produk perangkat lunak
- SKPL ; penentuan kebutuhan secara ketat sebelum memulai desain dan kemudian mengurangi redesain dan realistis untuk memperkirakan biaya produk, risiko, dan jadwal

# Requirement Analysis

- **Mempelajari dan memahami persoalan**

mempelajari masalah yang ada pada perangkat lunak yang dikembangkan, sehingga dapat ditentukan :

- siapa pemakai yang menggunakan perangkat lunak.
- dimana perangkat lunak akan digunakan .
- pekerjaan apa saja dari pemakai yang akan dibantu oleh perangkat lunak.
- apa saja cakupan dari pekerjaan tersebut, dan bagaimana mekanisme pelaksanaannya.
- apa yang menjadi kendala dilihat dari sisi teknologi yang digunakan atau dari sisi hukum dan standar.

# Requirement Analysis

Cara yang digunakan oleh pengembang khususnya analis dalam memahami masalah perangkat lunak biasanya dilakukan :

- wawancara dengan pemakai
- observasi atau pengamatan lapangan
- kuesioner
- mempelajari referensi atau dokumen-dokumen yang digunakan, seperti dokumen hasil analisa dan perancangan perangkat lunak.

# Requirement Analysis

- **Mengidentifikasi kebutuhan pemakai**

Substansi yang ditanyakan ada sedikit perbedaan, yaitu :

- Fungsi apa yang diinginkan pada perangkat lunak.
- Data atau informasi apa saja yang akan diproses.
- Kelakuan sistem apa yang diharapkan.
- Antarmuka apa yang tersedia (software interfaces, hardware interfaces, user interfaces, dan communication interfaces)

## Seorang analis membutuhkan :

- Komunikasi dan brainstorming yang intensif dengan pelanggan.
- Pembuatan prototype perangkat lunak atau screenshoot.
- Data atau dokumen yang lengkap.

# Requirement Analysis

- **Mendefinisikan kebutuhan perangkat lunak**

- 1. Kebutuhan fungsional***

- Entri dan rekam data transaksi penjualan.
    - Retrieve data transaksi penjualan untuk periode tertentu (periode sesuai dengan inputan periode yang diinputkan pada keyboard).
    - Rekam data akumulasi transaksi penjualan periode tertentu ke jurnal umum berikut account pasangannya (kas).

# Requirement Analysis

## **2. *Kebutuhan antarmuka***

- Antarmuka pemakai untuk memasukkan dan merekam data penjualan.
- Antarmuka pemakai untuk menyajikan dan menjurnal informasi transaksi penjualan pada periode tertentu.
- Antarmuka untuk jaringan lokal yang menghubungkan perangkat lunak aplikasi dibagian penjualan dengan perangkat lunak aplikasi dibagian akuntansi.



# Requirement Analysis

## **3. *Kebutuhan unjuk kerja***

Misalkan :

- Proses jurnal hanya bisa dilakukan sekali setelah data transaksi penjualan direkam.
- Adanya otoritas pemakaian perangkat lunak dan akses data sesuai dengan bagian pekerjaan masing-masing.

# Requirement Analysis

- **Membuat dokumen spesifikasi kebutuhan perangkat lunak (SKPL)**

Kebutuhan yang telah didefinisikan selanjutnya dibuat dokumentasinya yaitu Spesifikasi Kebutuhan Perangkat Lunak (SKPL) atau Software Requirement Specification (SRS).

Dibuat untuk menyatakan secara lengkap apa yang dapat dilakukan oleh perangkat lunak, termasuk deskripsi lengkap semua antarmuka yang akan digunakan.

# Requirement Analysis

- **Mengkaji ulang (review) kebutuhan**

Proses ini bisa dilakukan lebih dari satu kali dan sering kali akan muncul kebutuhan-kebutuhan baru dari pemakai.

Diperlukannya negosiasi antara pengembang dengan pelanggan sesuai dengan prinsip “win win solution” sampai kebutuhan tersebut disetujui oleh kedua belah pihak.

Sedangkan menurut Pressman [PRE01], analisis kebutuhan perangkat lunak dapat dibagi menjadi lima area pekerjaan, yaitu :

- Pengenalan masalah
- Evaluasi dan sistesis
- Pemodelan
- Spesifikasi
- Tinjau ulang (review)

# UML Modeling


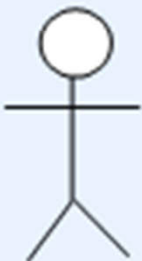

- Tiga aspek definisi berkenaan dengan objek. Sesuatu (something) yang dapat dikategorikan sebagai tipe objek. Tipe objek mungkin termasuk orang (person), tempat, benda (thing) atau peristiwa/kejadian event. Seorang pekerja, pelanggan, pengajar dan pelajar merupakan contoh objek orang (person)
- Attribute adalah data yang mewakili karakteristik interes tentang sebuah objek.
- Object Instance adalah setiap orang khusus ,tempat, sesuatu atau kejadian dan juga nilai untuk atribut dari objek.
- Behavior adalah kumpulan dari sesuatu yang dapat dilakukan oleh objek yang terkait dengan fungsi–fungsi yang bertindak pada data objek (atau atribut).

# Konsep Penting dalam Pemodelan Objek

- Kelas adalah kumpulan/himpunan objek dengan atribut/properti yang mirip
- Inheritance adalah metode dan atau atribut yang ditentukan didalam sebuah objek class dapat diwariskan atau digunakan lagi oleh objek kelas lainnya.
- Generalization /specialization adalah beberapa tipe kelas objek, dikelompokkan (atau di abstraksi ) kedalam kelasnya sendiri (hubungna Supertype dan Subtype)
- Supertype adalah sebuah entitas yang berisi atribut dan behavior yang umum bagi satu atau lebih subtype
- Subtype adalah sebuah kelas objek yang mewariskan atribut dan behavior dari sebuah kelas supertype
- Encapsulation adalah pengemasan beberapa item ke dalam satu unit

# Use Case Diagram

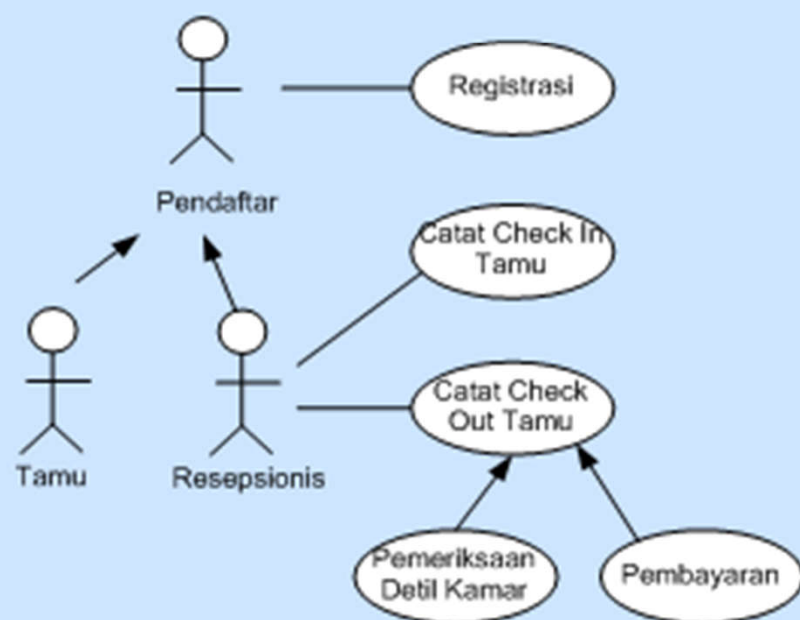
- Tujuan pembuatan model use-case persyaratan adalah untuk mendapatkan dan menganalisis informasi persyaratan yang cukup untuk mempersiapkan model yang mengkomunikasikan apa yang diperlukan dari perspektif pengguna tetapi bebas dari detail spesifik tentang bagaimana sistem akan dibangun dan diimplementasikan.

Simbol	Deskripsi
Use case 	Mewakili fungsi pada sistem, menggunakan kata kerja
Aktor 	<ul style="list-style-type: none"> <li>Mewakili orang, sistem atau external entitas / stakeholder yang memberikan atau menerima input atau output ke/dari sistem</li> <li>Actor menggambarkan sebuah tugas/peran dan bukannya posisi sebuah jabatan, menggunakan kata benda</li> <li>Tidak boleh ada komunikasi langsung antar aktor</li> <li>Indikasi &lt;&lt;system&gt;&gt; untuk sebuah aktor yang merupakan sebuah sistem. Aktor berupa sistem dapat dibuat bentuk kotak.</li> <li>Letakkan aktor <b>utama</b> anda pada pojok kiri atas dari diagram</li> </ul>
Association 	<ul style="list-style-type: none"> <li>Bukan menggambarkan aliran data/infomasi melainkan menggambarkan interaksi use case dengan aktor atau use case lain</li> <li>Ada 4 jenis relasi yang bisa timbul pada use case diagram               <ul style="list-style-type: none"> <li>- Association antara actor dan use case</li> <li>- Association antara use case</li> </ul> </li> </ul>



Simbol	Deskripsi
<p>Contoh :</p>	
<p>Generalization / Inheritance</p>	<ul style="list-style-type: none"> <li>• Digambarkan dengan garis berpanah tertutup</li> <li>• Digambarkan secara vertical dengan inheriting sub use case di bawah dari parent use case</li> <li>• Generalization aktor atau/dan use case</li> </ul>
<p>Association &lt;&lt;extend&gt;&gt;</p>	<ul style="list-style-type: none"> <li>• Merupakan perluasan dari use case lain jika kondisi atau syarat terpenuhi</li> <li>• Kurangi penggunaan association Extend ini, terlalu banyak pemakaian association ini membuat diagram sulit dipahami.</li> <li>• Tanda panah terbuka harus terarah ke parent use case</li> <li>• Gambarkan &lt;&lt;extend&gt;&gt; secara vertical</li> </ul>

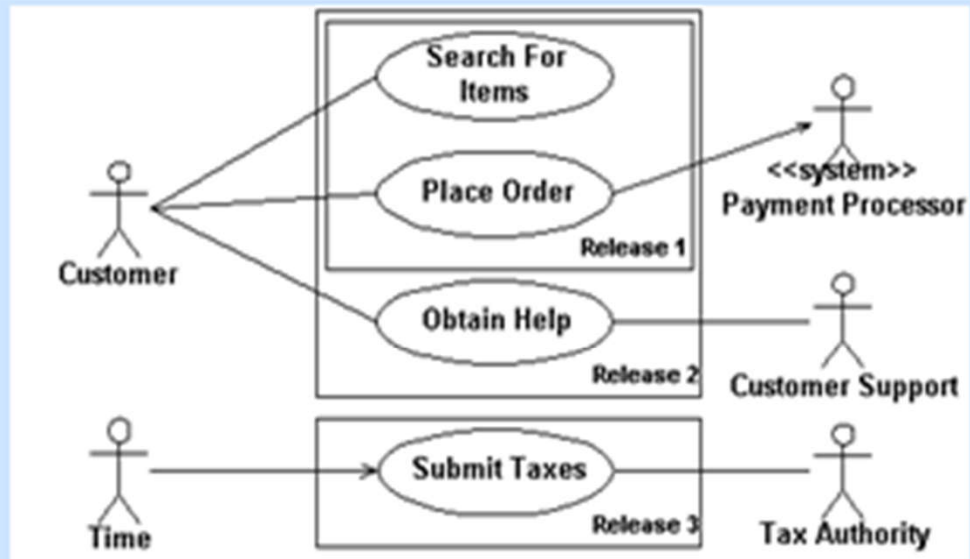
Contoh :



System  
Boundary  
(Boxes)

- Digambarkan dengan kotak disekitar use case, untuk menggambarkan jangkauan (scope) dari sistem
- Digunakan apabila memberikan beberapa alternative sistem yang dapat dijadikan pilihan

Contoh :



## Tahapan yang digunakan untuk membuat use case diagram :

- Mengidentifikasi pelaku bisnis / aktor (seseorang atau sesuatu) yang berinteraksi langsung dengan sistem.
- Mengidentifikasi use case persyaratan bisnis / fungsionalitas sistem dimana setiap satu fungsionalitas menjadi 1 use case
- Membuat diagram use case dengan terlebih dahulu membuat batasannya (boundary)
- Setelah use case dan pelaku teridentifikasi, diagram model use case pun digunakan untuk menggambarkan secara grafis lingkup dan batasan sistem.
- Mendokumentasikan naratif use case persyaratan bisnis

# Activity Diagram

Diagram ini juga dapat digunakan untuk memodelkan action yang akan dilakukan saat sebuah operasi dieksekusi dan memodelkan hasil dari action tersebut.


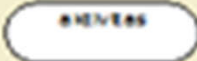


Fungsi Diagram Aktivitas :




- Menjelaskan lebih detail use case.
- Menjelaskan bisnis rule yang kompleks.
- Menjelaskan algoritma yang kompleks.
- Pengganti flow chart dan data flow diagram (DFD)

Yang perlu diperhatikan dalam menggambar diagram aktivitas :

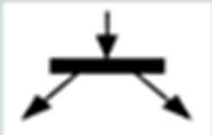



- Menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis.
- Menunjukkan urutan
- Menggambarkan aktivitas sistem bukan apa yang dilakukan aktor
- Mendeskripsikan kegiatan-kegiatan dalam sebuah prosedur / sistem, namun dapat juga digunakan untuk mendeskripsikan alur kegiatan seperti use case

## Simbol-simbol

Simbol	Deskripsi
Status awal 	status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Activity state / aktivitas 	aktivitas (berbentuk elips) yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / decision 	asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu, percabangan dapat diberi label (opsional)
Transisi 	transisi menghubungkan simbol-simbol, transisi dapat diberi label (opsional)

<b>Penggabungan / join</b> 	asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
<b>status akhir</b> 	status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
<b>partisi / swimlane</b> 	<p>memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi,</p> <p>Swimlane dapat berupa kolom (vertikal), atau berupa baris (horisontal)</p>



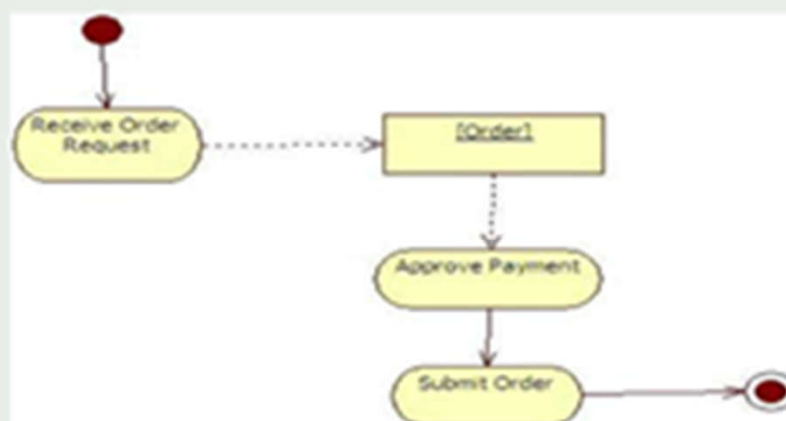
Simbol	Deskripsi
Fork 	Fork, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel
Join 	Join, digunakan untuk menunjukkan kegiatan yang digabung
Contoh Fork dan Join : 	
Objek Data dan alur informasi / data 	dapat menggunakan node objek untuk memperlihatkan aliran data melalui suatu aktifitas (opsional)

Objek Data dan alur informasi / data

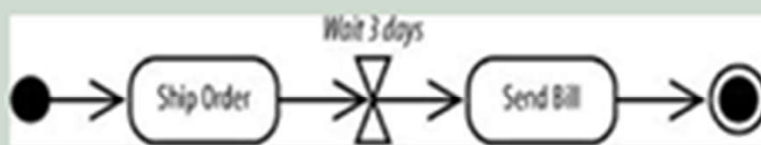


dapat menggunakan node objek untuk memperlihatkan aliran data melalui suatu aktifitas (opsional)

Contoh penggunaan objek data :



Time event



Terkadang waktu adalah salah satu faktor dalam aktifitas suatu sistem. Time event dapat memodelkan :

- waktu tunggu
- interfal waktu

### Signal (sinyal)

Sending a signal




Receiving a signal



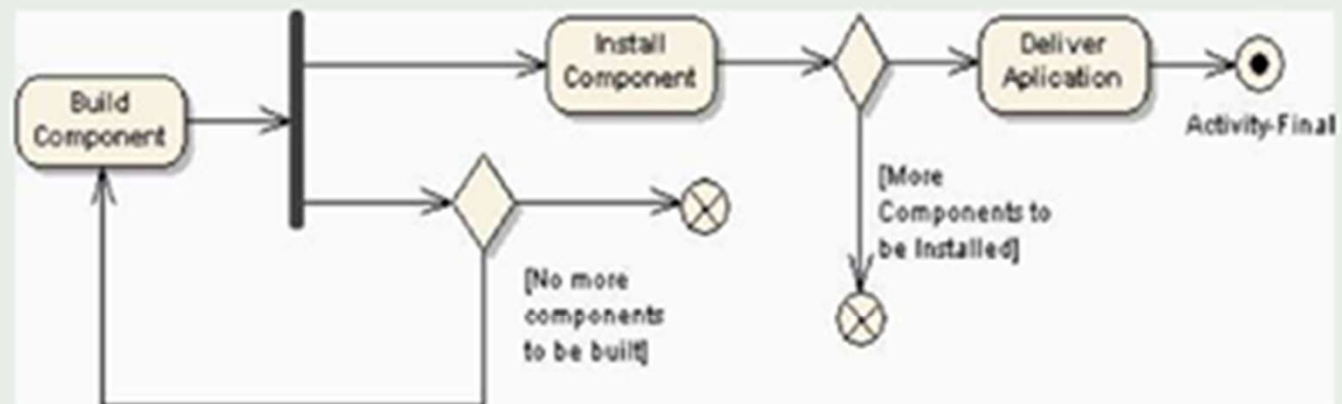
Sinyal menggambarkan adanya interaksi dengan pihak eksternal (pihak ketiga, dst), contohnya bank.

Contoh sinyal : ketika otorisasi pembayaran credit card atau debit card, kasir perlu untuk memverifikasi card tersebut dengan cara berinteraksi dengan layanan approval yang disediakan oleh bank.

Sinyal **tidak harus** digambarkan berpasangan (send dan receive).

Simbol	Deskripsi
<p>Flow Final (Akhir Aliran)</p> 	<p>Sebuah flow final node menghentikan jalurnya sendiri bukan terhadap seluruh aktifitas.</p> <p>Flow final menggambarkan sebuah kondisi keluar dari sistem, sedangkan simbol status akhir menyatakan aktifitas telah selesai.</p>

Contoh :



## Langkah-langkah pembuatan diagram aktivitas :

- Buat simbol status awal ketika mengawali diagram. Diagram aktivitas dibaca dari atas ke bawah (vertikal) atau dari kiri ke kanan (horisontal).
- Gambarkan aksi pertama dan seterusnya sesuai aliran kegiatan sistem. Gunakan simbol-simbol yang dibutuhkan (saja) untuk menggambarkan proses bisnis. Gambarkan aktivitas dari sisi sistem, bukan dari sisi aktor.
- Cabang keputusan digunakan untuk menunjukkan suatu kegiatan yang memenuhi kondisi tertentu. Seluruh pancabangan diakhiri tanda penggabungan (menggunakan tanda decision) sebagai akhir perilaku.
- Akhiri diagram dengan simbol status akhir

