

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Tinjauan Teori Umum**

##### **2.1.1 Pengertian Sistem**

Sistem berasal dari kata Yunani yaitu *systema* berarti kesatuan atau sekumpulan. (Galih Ariadhi Pranata, Haryanto Tanuwijaya, 2014) dalam Jogiyanto (2005: 1), sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan untuk menyelesaikan suatu sasaran tertentu.

Sistem didefinisikan sebagai sekumpulan prosedur yang saling berkaitan dan saling terhubung untuk melakukan suatu tugas bersama-sama (Pratama, I Putu, 2014: 7).

Sistem (*system*) adalah kumpulan dari sub-sub sistem, elemen-elemen, prosedur-prosedur, yang saling berinteraksi untuk mencapai tujuan tertentu, seperti informasi, target atau goal (Hapzi dan Tonny, 2010: 8).

##### **2.1.1.1 Karakteristik Sistem**

Sebuah Sistem mempunyai karakteristik atau sifat-sifat tertentu seperti elemen-elemen, batasan, lingkungan sistem, penghubung, masukan, pengolahan, keluaran, dan tujuan.

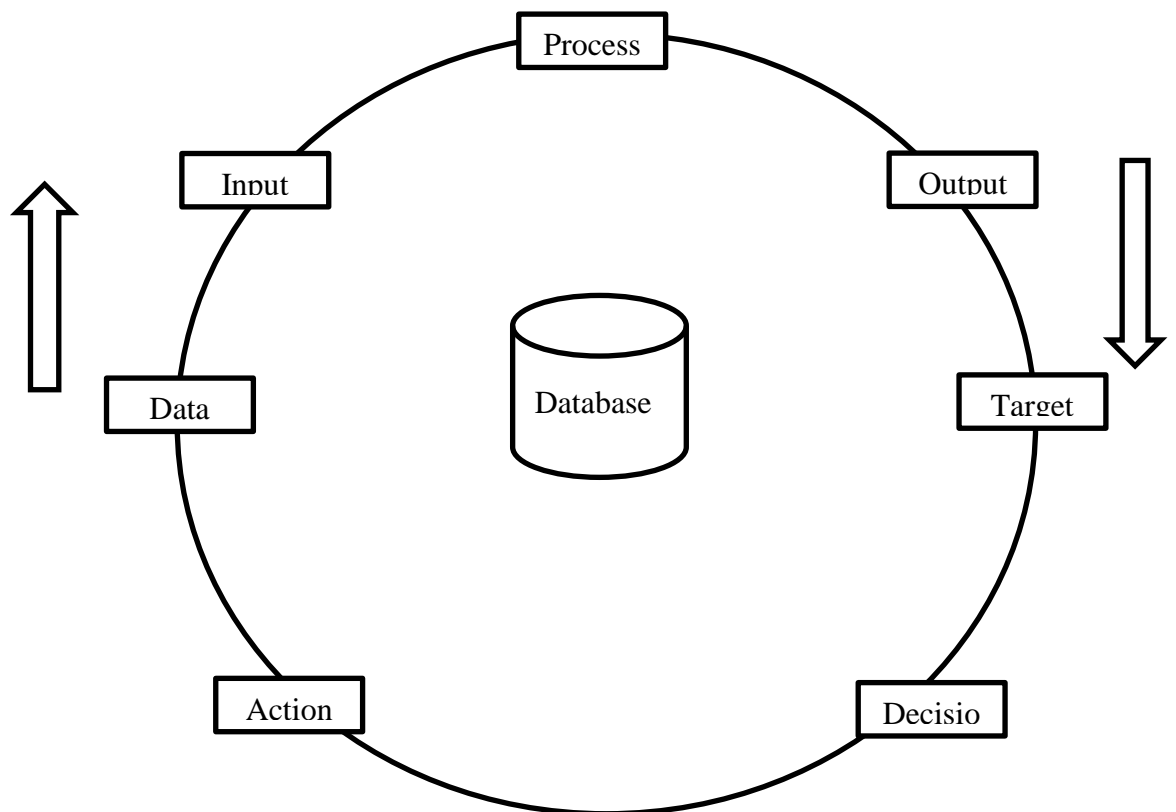
### **2.1.1.2 Daur Hidup Sistem**

Pada awal pengembangan perangkat lunak, para pembuat program (*programmer*) langsung melakukan pengodean perangkat lunak tanpa menggunakan prosedur atau tahapan pengembangan perangkat lunak. Dan ditemuilah kendala-kendala seiring dengan perkembangan skala sistem-sistem perangkat yang semakin besar. SDLC dimulai dari tahun 1960-an, untuk mengembangkan sistem skala usaha besar secara fungsional untuk para konglomerat pada jaman itu. Sistem-sistem yang dibangun mengelola informasi kegiatan dan rutinitas dari perusahaan-perusahaan yang berpotensi memiliki data yang besar dalam perkembangannya (Rossa dan Shalahudin, 2016: 26).

SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembnagkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarka best practice atau cara-cara yang sudah teruji baik) (Rossa dan Shalahudin, 2016:26).

### **2.1.2 Konsep Dasar Informasi**

Informasi (*information*) adalah data yang telah diolah menjadi suatu bentuk yang penting bagi sipenerima dan mempunyai nilai yang nyata atau dapat dirasakan manfaatnya dalam keputusan-keputusan yang akan datang (Hapzi dan Tonny, 2010: 10).



Sumber : (Hapzi dan Tonny, 2010: 11) dalam Modifikasi Hoffer at all (1996: 25)

**Gambar 2.1** Siklus Informasi

### 2.1.2.1 Kualitas Informasi

Kualitas informasi ditentukan oleh beberapa faktor, yaitu :

1. Keakuratan data teruji kebenarannya

Informasi harus bebas dari kesalahan-kesalahan, tidak biasa, dan tidak menyesatkan.

2. Kesempurnaan informasi

Untuk mendukung faktor pertama tersebut diatas, maka kesempurnaan informasi menjadi faktor penting, dimana informasi disajikan lengkap tanpa pengurangan, penambahan atau perubahan.

3. Tepat waktu

Informasi harus disajikan secara tepat waktu, meningat informasi akan menjadi dasar dalam pengambilan keputusan.

4. Relevansi

Informasi akan memiliki nilai manfaat yang tinggi, jika informasi tersebut diterima oleh mereka yang membutuhkan dan menjadi tidak berguna jika diberikan kepada mereka yang tidak membutuhkan.

5. Mudah dan Murah.

### 2.1.3 Sistem Informasi

Sesungguhnya yang dimaksud dengan sistem informasi tidak harus melibatkan komputer. Sistem informasi yang menggunakan komputer biasa disebut sistem informasi berbasis komputer (*Computer Based Information Systems* atau CBIS). Dalam praktik, istilah sistem informasi lebih sering dipakai tanpa embel-embel berbasis komputer walaupun dalam kenyataannya komputer merupakan bagian yang penting (Kadir, 2014: 8).

Adapun definisi sistem informasi dapat dilihat pada tabel berikut ini (Kadir, 2014: 9) :

**Tabel 2.1** Definisi Sistem Informasi

Sumber	Definisi
Alter (1992)	Sistem informasi adalah kombinasi antar prosedur kerja, informasi, orang, dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi.

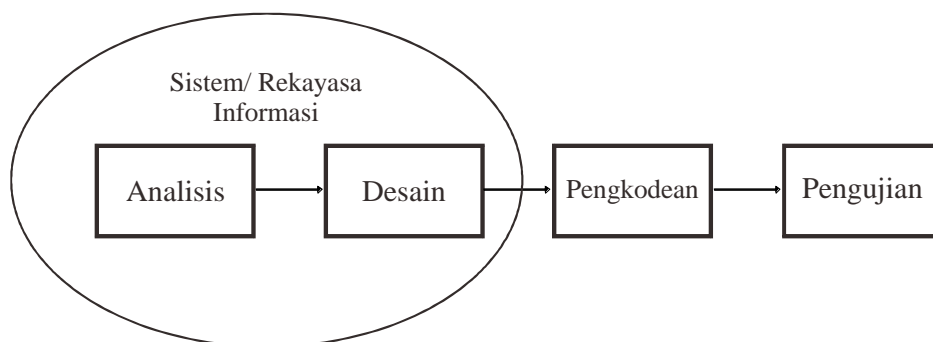
**Tabel 2.1** Lanjutan

Bodnar dan Hopwood (1993)	Sistem informasi adalah kumpulan perangkat keras dan perangkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.
Gellines, Oram, dan Wiggins (1990)	Sistem Informasi adalah suatu sistem buatan manusia yang secara umum terdiri atas sekumpulan komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan, dan mengelola data serta menyediakan informasi keluaran kepada para pemakai.
Hall (2001)	Sistem informasi adalah sebuah rangkaian prosedur formal dimana data dikelompokkan, diproses menjadi informasi, dan didistribusikan kepada para pemakai.
Turban, McLean, dan Wetherbe (1999)	Sebuah sistem informasi mengumpulkan, memproses, menyimpan, menganalisis, dan menyebarkan informasi untuk tujuan yang spesifik.
Willkinson (1992)	Sistem informasi adalah kerangka kerja yang mengkoordinasikan sumber daya (manusia, komputer) untuk mengubah masukan (input) menjadi keluaran (informasi), guna mencapai sasaran-sasaran perusahaan.

## 2.1.4 Pengembangan dan Perancangan Sistem Informasi

### 2.1.4.1 SDLC (*Software Development Life Cycle*)

Proses pengembangan sistem mempunyai beberapa tahapan mulai dari sistem itu direncanakan sampai dengan sistem tersebut diterapkan, dioperasikan, dan dipelihara. Salah satu Contohnya adalah Model Waterfall, Tahapan-tahapan yang ada pada model ini dapat dilihat pada gambar berikut :



**Gambar 2.2** Gambar SDLC Model Waterfall

SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan best practice atau cara-cara yang sudah teruji baik) (Rossa dan Shalahudin, 2016: 26).

Tahapan-tahapan yang ada pada SDLC secara global adalah sebagai berikut (Rossa dan Shalahudin, 2016: 26) :

1. Inisiasi (*Initiation*)

Tahapan ini biasanya ditandai dengan pembuatan proposal proyek perangkat lunak.

2. Pengembangan Konsep Sistem (*system concept development*)

Mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, manajemen rencana, dan pembelajaran kemudahan sistem.

3. Perencanaan (*Planning*)

Mengembangkan rencana manajemen proyek dan dokumen perencanaan lainnya. Menyediakan dasar untuk mendapatkan sumber daya (*resources*) yang dibutuhkan untuk memperoleh solusi.

4. Analisis Kebutuhan (*requirements analysis*)

Menganalisis kebutuhan pemakai sistem perangkat lunak (*user*) dan mengembangkan kebutuhan user. Membuat dokumen kebutuhan fungsional.

5. Desain (*design*)

Mentransformasikan kebutuhan detail menjadi kebutuhan yang sudah lengkap, dokumen desain sistem fokus pada bagaimana dapat memenuhi fungsi-fungsi yang dibutuhkan.

6. Pengembangan (*development*)

Mengonversi desain ke sistem informasi yang lengkap termasuk bagaimana memperoleh dan melakukan instalasi lingkungan sistem yang dibutuhkan; membuat basis data dan mempersiapkan prosedur kasus pengujian; mempersiapkan berkas atau *file* pengujian, pengodean, pengompilasian, memperbaiki dan membersihkan program; peninjauan pengujian.

#### 7. Integrasi dan Pengujian (*Ingration and Test*)

Mendemonstrasikan sistem perangkat lunak bahwa telah memenuhi kebutuhan yang dispesifikasikan pada dokumen kebutuhan fungsional. Dengan diarahkan oleh staff penjamin kualitas (*Quality Assurance*) dan *user*. Menghasilkan.

#### 8. Implementasi (*Implementation*)

Teramsuk pada persiapan implementasi, implementasi perangkat lunak pada lingkungan produksi (lingkungan pada user) dan menjalankan resolusi dari permasalahan yang teridentifikasi dari fase integrasi dan pengujian.

#### 9. Operasi dan Pemeliharaan (*Operations and Maintenance*)

Mendeskripsikan pekerjaan untuk mengoperasikan dan memelihara sistem informasi pada lingkungan produksi.

#### **2.1.4.2 UML (*Unifed Modelling Languange*)**

Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun menggunakan teknik pemrograman berorientasi objek, yaitu *Unifed Modeling Languange* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspisifikasikan, menggambarkan, membangun, dan dokumentasi dair sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama simula-67 yang dikembangkan pada tahun 1967. Bahasa



pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada developer pengembang bahasa pemrograman berorientasi objek selanjutnya. Karena banyaknya metodologi-metodologi yang berkembang pesat saat itu, maka munculah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi-metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep seperti konsep *Object Modeling Technique* (OMT) dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators* (CRC) dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaugh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernama *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Language* (UML).

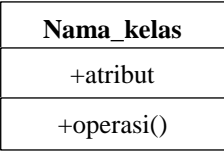


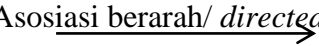
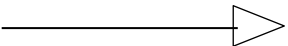
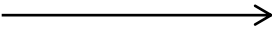

Pada 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan september 1997 UML diakomodasi oleh OMG sehingga sampai saat ini UML telah memberikan kontribusi yang cukup besar didalam metodologi berorientasi objek dan hal-hal yang terkait didalamnya (Rossa dan Salahuddin, 2016: 138-139).

#### 1. *Class Diagram*

Diagram kelas atau *Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Adapun

simbol-simbol yang ada pada diagram kelas dapat dilihat pada tabel 2.3 (Rossa dan Salahuddin, 2016: 141-147):

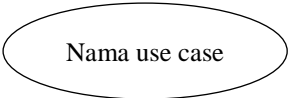
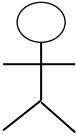

**Tabel 2.2** Simbol-Simbol Class Diagram

Simbol	Deskripsi
kelas 	Kelas pada struktur sistem
Antarmuka/ <i>interface</i> 	Sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek
Asosiasi/ <i>association</i> 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
Asosiasi berarah/ <i>directed association</i> 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i>
Generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus)
Kebergantungan/ <i>dependency</i> 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi/ <i>aggregation</i> 	Relasi antar kelas dengan makna semua-bagian ( <i>whole-part</i> )

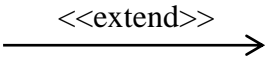
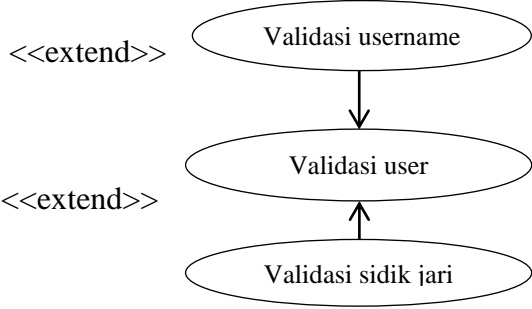
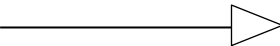
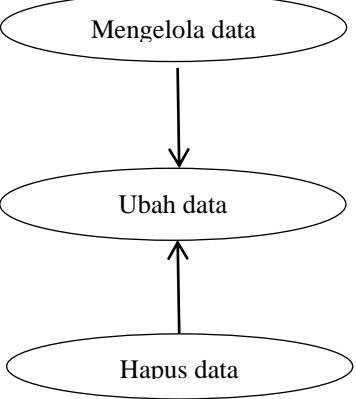
## 2. Use Case

*Use Case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Adapun simbol-simbol pada *use case* dapat dilihat pada tabel 2.4 berikut (Rossa dan Salahuddin, 2016: 155-158):

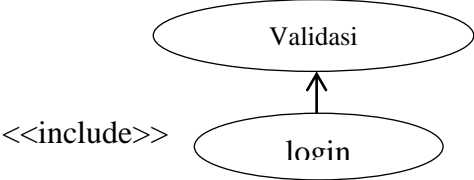
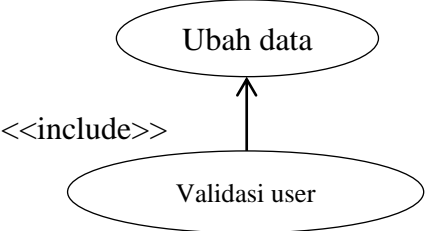
**Tabel 2.3** Simbol-Simbol Use Case

Simbol	Deskripsi
<p><i>Use case</i></p>  <p>Nama use case</p>	<p>Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama <i>use case</i></p>
<p>Aktor</p>  <p>Nama actor</p>	<p>Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan dengan kata benda di awal frase nama aktor</p>
<p>Asosiasi/ <i>association</i></p> 	<p>Komunikasi antar aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor</p>

Tabel 2.3 Lanjutan

<p>Ekstensi/ <i>extend</i></p> <p style="text-align: center;">  </p>	<p>Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu mirip dengan prinsip inheritance pada pemogram berorientasi objek.</p> <p style="text-align: center;">  </p>
<p>Generalisasi/ <i>generalization</i></p> <p style="text-align: center;">  </p>	<p>Hubungan generalisasi dan spesilisasi (umum-khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misal:</p> <p style="text-align: center;">  </p>



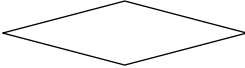


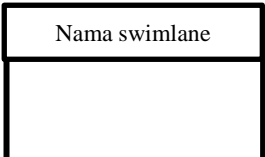
Tabel 2.3 Lanjutan

<p>Menggunakan/ <i>include/</i> <i>user</i></p> <p><code>&lt;&lt;include&gt;&gt;</code> →</p> <p><code>&lt;&lt;User&gt;&gt;</code> →</p>	<p>Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini</p> <p>Ada dua sudut pandang yang cukup besar mengenai include di user use case:</p> <ol style="list-style-type: none"> <li>1. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan misal pada kasus berikut :</li> </ol>  <pre> graph BT     login((login)) -- "&lt;&lt;include&gt;&gt;" --&gt; Validasi((Validasi))   </pre> <ol style="list-style-type: none"> <li>2. Include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misal pada kasus berikut ini:</li> </ol>  <pre> graph BT     ValidasiUser((Validasi user)) -- "&lt;&lt;include&gt;&gt;" --&gt; UbahData((Ubah data))   </pre>
--	---

### 3. Activity Diagram

Diagram aktiviti atau *activity diagram* menggambarkan *worklfow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Adapun simbol-simbol akviti diagram dapat dilihat pada tabel 2.5 (Rossa dan Salahuddin, 2016: 155-158).

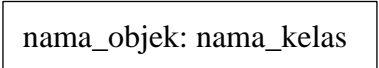

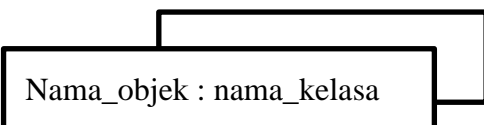

**Tabel 2.4** Simbol-simbol Activity Diagram

Simbol	Deskripsi
Satus awal 	Status awal aktivitas sistem, sebuah diagram aktivitas aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan/ <i>decision</i> 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggambungan/ <i>join</i> 	Asosiasi penggambungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

#### 4. *Collaboration Diagram*

Diagram *Collaboration* menggambarkan interaksi antar objek/bagian dalam bentuk urutan pengiriman pesan. Diagram *Collaboration* mempresentasikan informasi yang diperoleh dari Diagram kelas, Diagram Sekuen, dan Diagram *Use Case* untuk mendeskripsikan gabungan antara struktur statis dan tingkah laku dinamis dari suatu sistem. Adapun simbol-simbol diagram sekuen dapat dilihat pada tabel 2.6 (Rossa dan Salahuddin, 2016: 168-167).

**Tabel 2.5** Simbol-Simbol *Collaboration Diagram*

Simbol	Deskripsi
Objek Tanpa waktu aktif 	Objek yang melakukan interaksi pesan
Link 	Relasi antar-objek yang menghubungkan objek satu dengan yang lainnya atau dengan dirinya sendiri 
Arah pesan / <i>stimulus</i> 	Arah pesan yang terjadi, jika pada suatu <i>link</i> ada dua arah pesan yang berbeda maka arah juga digambarkan dua arah pada dua sisi link

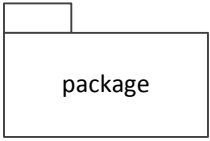
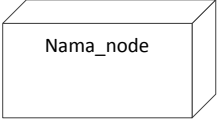
## 5. *Deployment Diagram*

Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut:

1. Sistem tambahan (*embedded system*) yang menggambarkan rancangan *device*, *node*, dan *hardware*.
2. Sistem *client/ server*.
3. Sistem terdistribusi murni.
4. Rekayasa ulang aplikasi.


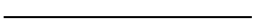
Berikut adalah simbol-simbol yang ada pada diagram *deployment* (Rossa dan Salahuddin, 2016: 154-155):

**Tabel 2.6** Simbol-Simbol Deployment Diagram

Simbol	Deskripsi
<p data-bbox="316 1249 427 1283">Package</p> 	<p data-bbox="790 1249 1370 1357"><i>Package</i> merupakan sebuah bungkusan dari satu atau lebih <i>node</i></p>
<p data-bbox="316 1503 387 1536"><i>Node</i></p> 	<p data-bbox="790 1503 1370 1834">Biasanya mengacu pada perangkat keras (<i>hardware</i>), perangkat lunak yang tidak dibuat sendiri (<i>software</i>), jika di dalam <i>node</i> disertakan komponen untuk mengkonsistensikan rancangan</p>



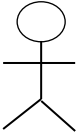
**Tabel 2.6** Tabel Lanjutan



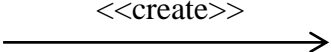
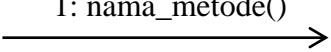
Kebergantungan/ <i>dependency</i> 	Kebergantungan antar <i>node</i> , arah panah mengarah pada <i>node</i> yang dipakai
<i>Link</i> 	Relasi antar <i>node</i>

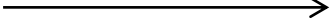

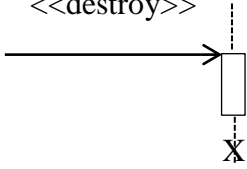
## 6. *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu, untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Adapun simbol-simbol diagram sekuen dapat dilihat pada tabel 2.6 (Rossa dan Salahuddin, 2016: 165-167).

**Tabel 2.6** Simbol-Simbol Sequence Diagram

Simbol	Deskripsi
Aktor  Nama aktor Atau <div style="border: 1px solid black; padding: 2px; display: inline-block;">Nama_aktor</div> Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan dengan kata benda di awal frase nama aktor

<p>Garis hidup/ <i>lifeline</i></p> 	<p>Menyatakan kehidupan suatu objek</p>
<p>Objek</p> <div style="border: 1px solid black; padding: 2px; width: fit-content; margin: 10px auto;"> <p><u>Nama objek:nama kelas</u></p> </div>	<p>Menyatakan objek yang berinteraksi pesan</p>
<p>Waktu aktif</p> 	<p>Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya</p> <p>Maka cek StatusLogin() dan open() dilakukan di dalam metode login()</p> <p>Aktor tidak memiliki waktu aktif</p>
<p>Pesan tipe create</p> 	<p>Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat</p>
<p>Pesan tipe call</p> 	<p>Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,</p> <p>Arah panah mengarah pada objek yang memiliki operasi/metode karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi</p>

<p>Pesan tipe send</p> <p>1: masukan</p> 	<p>Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim</p>
<p>Pesan tipe return</p> <p>1: keluaran</p> 	<p>Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian</p>
<p>Pesan tipe destroy</p> <p>&lt;&lt;destroy&gt;&gt;</p> 	<p>Menyatakan suatu objek Mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destory</p>

## 2.2 Tinjauan Teori Khusus

### 2.2.1 Pengertian Persediaan

Persediaan (*Iventory*), dalam konteks produksi, dapat diartikan sebagai sumber daya menganggur (*ilde resource*). Sumber daya menganggur ini belum digunakan karena menunggu proses lebih lanjut. Yang dimaksud mengunggu proses lebih lanjut disini dapat berupa kegiatan produksi seperti kita jumpai pada sistem manufaktur, kegiatan pemasaran seperti dijumpai pada distribusi ataupun kegiatan konsumsi seperti pada sistem rumah tangga.

Keberadaan persediaan atau sumber daya menganggur ini dalam suatu sistem mempunyai suatu tujuan tertentu. Alasan utamanya adalah sumber daya tertentu tidak bisa didatangkan ketika sumber daya tersebut dibutuhkan. Sehingga, untuk menjamin tersediannya sumber daya tersebut perlu adanya persediaan yang siap digunakan ketika dibutuhkan, (Nur Heri Cahyana, Bambang Yuwono, Anjar Yudo Asmoro, 2012) dalam Rosnani (2007: 253).

Persediaan merupakan salah satu elemen utama dari modal kerja yang terus menerus mengalami perubahan. Tanpa persediaan, perusahaan akan menghadapi resiko, yaitu tidak dapat memenuhi keinginan pelanggan atas barang produksi. Oleh karena itu, dalam suatu persediaan, harus menghadapi investasi yang tidak terlalu rendah namun juga jangan terlalu tinggi. Ada beberapa ahli yang mengemukakan pengertian persediaan. Persediaan merupakan salah satu unsur yang paling aktif dalam operasi perusahaan yang secara kontinu diperoleh, diubah, kemudian dijual kembali, (Wahyudi, 2015: 166) dalam Martono(2002: 67).

### **2.2.1 Fungsi Persediaan**

Pengendalian persediaan merupakan fungsi manajerial yang sangat penting karena persediaan fisik banyak melibatkan investasi terbesar. Bila perusahaan menanamkan terlalu banyak dananya dalam persediaan, menyebabkan biaya penyimpanan yang berlebihan, dan mungkin mempunyai "*Opportunity Cost*" (dana dapat ditanamkan dalam investasi yang lebih menguntungkan). Sebaliknya, bila perusahaan tidak mempunyai persediaan yang cukup, dapat mengakibatkan meningkatkan biaya-biaya karena kekurangan bahan.

Istilah persediaan adalah suatu istilah umum yang menunjukkan segala sesuatu atau segala sumber daya perusahaan yang disimpan dalam antisipasi pemenuhan permintaan. Permintaan sumber daya internal ataupun eksternal meliputi persediaan bahan mentah, barang dalam proses, barang jadi atau produk akhir, bahan-bahan pembantu atau pelengkap dan komponen-komponen lain yang menjadi bagian keluaran produk perusahaan, (Wahyudi, 2016), dalam Martono (2002: 67).

### **2.2.2 Jenis Dan Tipe Persediaan**

Persediaan ada berbagai jenis. Setiap jenisnya mempunyai karakteristik khusus dan cara pengelolaannya juga berbeda. Persediaan jenisnya dapat dibedakan, (Rudy Wahyudi, 2015: 166) dalam Assauri (2004:171) :

#### a. Persediaan bahan baku (*Raw Material Stock*)

Persediaan dari barang-barang berwujud yang digunakan dalam proses produksi, barang mana dapat diperoleh dari sumber-sumber alam ataupun dibeli dari *supplier* atau perusahaan yang menghasilkan bahan baku bagi perusahaan pabrik yang menggunakannya.

#### b. Persediaan bagian produk (*Purchased part*)

Persediaan barang-barang yang terdiri dari part atau bagian yang di-terima dari perusahaan lain, yang dapat secara langsung di-*assembling* dengan *part* lain, tanpa melalui proses produksi sebelumnya.

#### c. Persediaan bahan-bahan pembantu atau barang-barang perlengkapan (*Supplies stock*)

Persediaan barang-barang atau bahan-bahan yang diperlihatkan dalam proses produksi untuk membantu berhasilnya produksi atau yang dipergunakan dalam bekerjanya suatu perusahaan, tetapi tidak merupakan bagian atau komponen dari barang jadi.

d. Persediaan barang setengah jadi atau barang dalam proses (*Work in process / progress stock*)

Persediaan barang-barang yang keluar dari tiap-tiap bagian dalam satu pabrik atau bahan-bahan yang telah diolah menjadi suatu bentuk, tetapi lebih perlu diproses kembali untuk kemudian menjadi barang jadi.

e. Persediaan barang jadi (*Finished goods stock*)

Barang-barang yang telah selesai diproses atau diolah dalam pabrik dan siap untuk dijual kepada pelanggan atau perusahaan lain.

### **2.2.3 Sejarah PHP**

PHP Hypertext Processor atau sering disebut PHP merupakan bahasa pemrograman berbasis server-side yang dapat melakukan parsing script php menjadi script web sehingga dari sisi client menghasilkan suatu tampilan yang menarik. PHP merupakan pengembangan dari FI atau Form Interface yang dibuat oleh Rasmus Lerdoff pada tahun 1995.

Berbeda dengan HTML, kode PHP tidak diberikan secara langsung oleh server ketika ada permintaan atau request dari sisi client namun dengan cara pemrosesan dari sisi server. Kode PHP seringkali digabungkan dengan Kode HTML. Untuk membedakannya dengan HTML, setiap kode PHP ditulis selalu

diberi tag pembuka yaitu `<?php` dan pada akhir kode PHP diberi tag penutup yaitu `?>`

#### **2.2.4 Fungsi dan Keunggulan PHP**

PHP memiliki manfaat yang sangat besar bagi para web programmer dan web developer pada saat membuat website keren yang dinamasi seperti membaca file, menulis file, menampilkan gambar, animasi atau movie, dan yang paling penting adalah dapat melakukan koneksi terhadap database seperti MySQL.

Keunggulan dari bahasa pemrograman PHP yaitu :

1. PHP memiliki native API untuk koneksi ke berbagai database, sehingga secara otomatis dalam melakukan koneksi lebih cepat dibandingkan dengan open database Connectivity (ODBC).
2. Eksekusi scripting dilakukan lebih cepat sehingga meningkatkan throughput pada server.
3. Penulisan program yang simpel dan sederhana yang membuat programmer newbie (pemula) mudah dalam memahami PHP.
4. Dukungan koneksi yang hampir bisa dilakukan ke semua database seperti MySQL, PostgreSQL, Sybase, Infomix, Interbase, ORACLE, SQL serve, dan lain-lain
5. PHP dapat dijalankan di beberapa web server seperti PWS, IIS, Apache, Xitami, Netscape Enterprise, AOL server dan Orelly webbsite pro, CGI dan ISAPI.
6. PHP juga dapat berjalan di berbagai platform seperti Unix dan Windows.

7. PHP dapat didistribusikan kembali dibawah lisensi Gnu Public License (GPL) karena bersifat open source.

### 2.2.5 Pengenalan MySQL

MySQL atau dibaca "My Sekuel" adalah salah satu RDBMS (*Relational Database Management System*) yaitu aplikasi yang menjalankan fungsi pengolahan data. MySQL pertama dikembangkan oleh MySQL AB yang kemudian diakuisisi Sun Microsystem dan terakhir dikelola oleh Oracle Corporation. MySQL terdiri dari Data Definition Language (DDL) dan Data Manipulation Language.

1. Data Defination Language (DDL)

Menurut Sibero (2011: 104) Data Definition Language (DDL) adalah suatu tata bahasa definisi data pada MySQL, DDL digunakan untuk mendefinisikan suatu Database, Table, Table-space, logfile group, server, index. DDL umumnya digunakan untuk mendefinisikan suatu wadah data atau record. DDL terdiri dari Create, Alter, Drop, Rename.

2. Data Manipulation Language (DML)

Menurut Sibero (2011: 108) Data Manipulation Language (DML) adalah suatu tatabahasa manipulasi data pada MySQL. DML digunakan untuk memanipulasi data pada table Database. Perintah DML terdiri dari SELECT, UPDATE, DELETE, INSERT.