BAB II
LANDASAN TEORI

Landasan teori diperoleh dari studi literatur, studi literatur diperlukan untuk mengeksplorasi teori-teori yang diperlukan dalam menyelesaikan tugas akhir ini.Fungsi dari teori adalah pertama sebagai alat untuk mencapai satuan pengetahuan yang sistematis.Dengan demikian teori sangat penting dalam memperjelas pengetahuan sebagai dasar organisasi pemikiran.Kedua, teori menjadi pembimbing bagi penulis dalam melakukan penelitian.

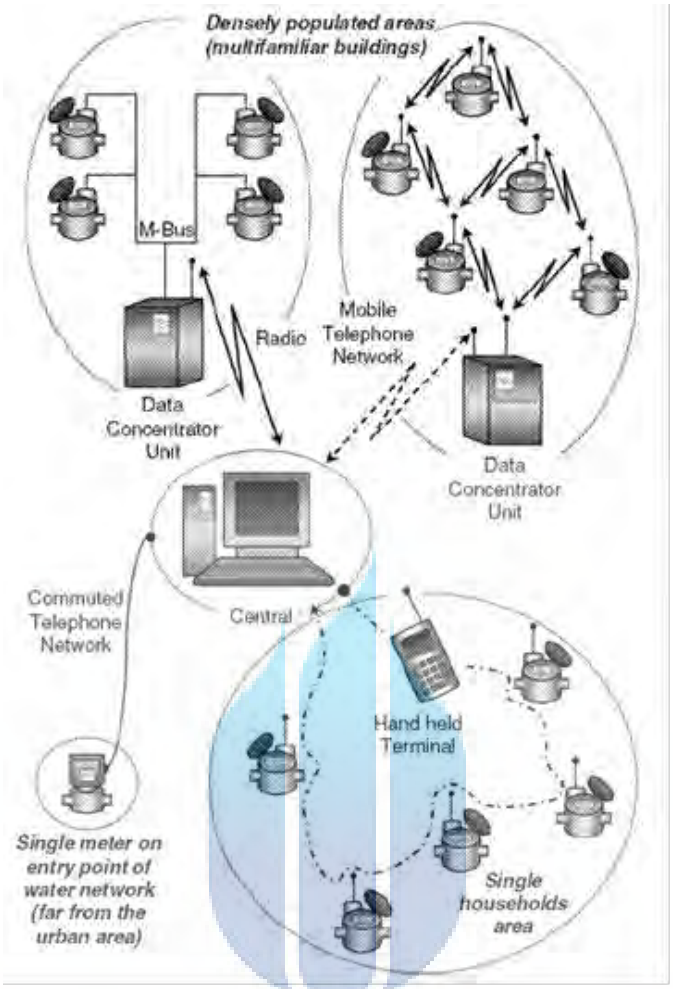
2.1 Konsep Dasar Pembacaan Meter Air

Merupakan kegiatan membaca indeks meter air yang terlihat pada register/totalizer/meteran air, yang dilakukan oleh petugas di lapangan, untuk selanjutnya diolah menjadi data billing/tagihan kepada pelanggan.

2.1.1 Metode Pembacaan Meter Air

Dalam pembacaan meter air dilapangan terdapat 2 metode yang dapat dilakukan, yaitu:

1. **Pembacaan Manual**
   Meter dibaca langsung secara manual dengan melihat langsung di lokasi meter air.
2. **Pembacaan Otomatis *(Automatic Meter Reading)***
   Meter dibaca dengan meggunakan bantuan alat dan meter tidak perlu langsung didatangi.



Sumber: Slide Presentasi Training MRD - PALYJA

Gambar 2. 1: Metode Pembacaan Otomatis *(Automatic Meter Reading)*

Metode *Automatic Meter Reading* (AMR) dapat dilakukan dengan cara berikut ini :

- *Handheld Terminal*

  Data indeks meter air ditarik dengan menggunakan *Portable Data Terminal* (PDT) yang dihubungkan dengan menggunakan sensor

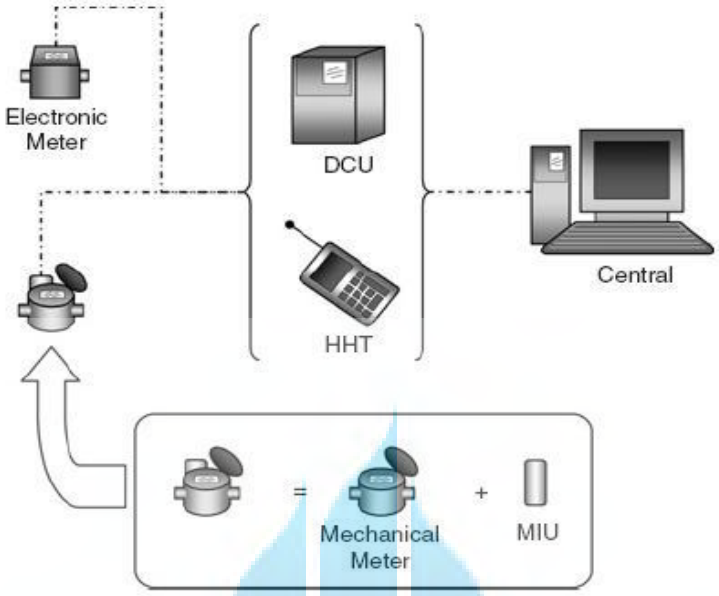
- Sistem *MBus*

  Data indeks meter ditarik oleh sistem dalam perangkat lunak yang diletakkan disuatu lokasi secara permanen.

- Data Transmission GSM

  Data indeks meter dikirim kepada sistem perangkat lunak dengan menggunakan sinyal GSM.

Dari ketiga hal diatas *Automatic Meter Reading* (AMR) ini diperlukan *Meter Interface Unit* (MIU) sebagai perekam data.

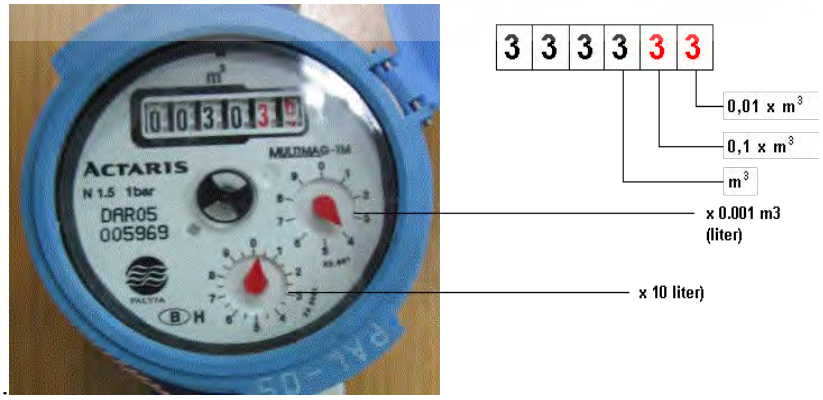


Sumber: Slide Presentasi Pengenalan Pembacaan Meter Air - PALYJA

Gambar 2. 2: Proses Pengiriman Data Pada AMR

2.1.2 Sistem Pembacaan Meter Air

Pembacaan meter air di PALYJA menggunakan sistem manual dengan bantuan alat *Mobile Scanner Barcode / Portable Data Terminal* (PDT) untuk mencatat dan merekam indeks meter air yang didatangi



Sumber: Slide Presentasi Pengenalan Pembacaan Meter Air – PALYJA

Gambar 2. 3: Index Meter Pada Totalizer/Register/Meteran Air

Indeks meter terdiri dari dua warna (sesuai ISO) :

- Warna Hitam

  Menunjukkan satuan m3, digunakan sebagai dasar perhitungan tagihan.

- Warna Merah

  Menunjukkan satuan liter, digunakan untuk pengujian meter air

Dibawah ini adalah contoh disain Norek yang ditempel pada alat meter air/kaca/dinding rumah sebagai identifikasi unik untuk semua pelanggan PALYJA

Sumber: Slide Presentasi Pengenalan Pembacaan Meter Air – PALYJA

Gambar 2. 4: *Label / Sticker* ID Pelanggan

2.2 Konsep Dasar Mobile Application

Mobile Application terdiri dari dua suku kata yakni *mobile* (bergerak) dan *application* (aplikasi). Mobile saat ini sering diorientasikan pada sebuah perangkat nirkabel atau yang biasa kita dengar dengan istilah *handphone* (telepon selular).Sedangkan Aplikasi itu sendiri merupakan sebuah program terdiri dari serangkaian kode-kode program yang diintegrasikan pada sebuah perangkat keras atau komputer. Aplikasi termasuk ke dalam kategori *software* dan dapat melakukan sesuatu sesuai dengan perintah atau instruksi dari seorang pengguna *(user)* pada komputer tersebut. Sehingga bisa diartikan *Mobile Application* adalah suatu aplikasi yang terdapat pada perangkat *mobile* atau nirkabel dan dapat digunakan walaupun penggunanya berpindah-pindah.

2.2.1 Pengertian *Mobile Device*

Mobile Device dikenal dengan istilah *cellphone*, *handheld computing* atau secara sederhana disebut dengan *handheld* adalah suatu perangkat / alat penghitung *(computing device)* yang berukuran kecil, *portable* dan mendukung komunikasi *wireless*, ciri khasnya memiliki layar tampilan *(display screen)* dengan layar sentuh atau keyboard mini.

Karakteristik *Mobile Device:*

1. Ukuran yang kecil
   Perangkat *mobile* memiliki ukuran yang kecil.Konsumen memiliki perangkat yang kecil untuk kenyamanan dan mobilitas mereka.
2. Memory yang terbatas
   Perangkat *mobile* juga memiliki *memory* yang kecil, yaitu *primary* (RAM) dan *secondary* (ROM).
3. Daya proses yang terbatas
   Sistem *mobile* tidaklah setangguh *desktop*
4. Daya yang rendah
   Perangkat mobile menghabiskan sedikit daya dibandingkan dengan perangkat *desktop*
5. Kuat dan dapat diandalkan
   Karena perangkat *mobile* selalu dibawa kemana aja, maka harus cukup kuat terhadap benturan atau getaran, debu dan air.

Macam – macam *mobile device*:

1. Laptop dengan *wireless* LAN
2. *Mobile phone*
3. *Personal Digital Assistance* (PDA) dengan *Bluetooth* atau IRDA

Keuntungan Teknologi *Mobile*:

1. Pengaksesan informasi setiap saat dan dimanapun
   Memungkinkan kita untuk bekerja tanpa batasan waktu dan tempat.
2. Mobilitas tinggi tanpa kerumitan kabel (W-LAN) & instalasi jaringan yang cepat.
3. Kompatibel dengan perangkat lain.

4. Reduksi biaya: seperti pengembangan pemindahan maupun perubahan konfigurasi LAN

Kekurangan Teknologi *Mobile*:

1. *Memory* yang terbatas
2. Daya yang terbatas
3. *Coverage area* terbatas (Wi-Fi)
4. Interferensi terhadap peralatan lain maupun benda disekitarnya
5. Mudah hilang

Media Komunikasi Data

Berfungsi untuk *transfer* data atau sinkronisasi data antara *mobile device* dengan server,

Berikut *transfer* data yang dapat dilakukan oleh *mobile device*:

- PAN (Via Bluetooth),
- Wireless LAN (Via Wi-Fi a/b/g/n),
- Wireless WAN (Via GSM, GPRS, 2G, 3G)
- Wired Ethernet & USB via Docking/Cradle

2.2.2 *Mobile Barcode Scanner* / *Portable Data Terminal* (PDT)

Mobile Barcode Scanner atau *Portable Data Terminal* (PDT) adalah perangkat komputer *mobile* yang digunakan untuk melakukan *entry data* dan umumnya memiliki kemampuan *scanner barcode* dimana data tersebut disimpan didalam PDT untuk kemudian *diupload / download* ke komputer.

Sumber: Brosur M3 Smart

Gambar 2. 5: *Mobile Barcode Scanner*

- *Reduce error possibilities in entry data, requires only one time entry to system.*
- *Ensure the reader comes to customer's location*
- *Accurate prove by having photograph of each meter when reading takes place.*
- *Data processing 7 validation needs less time compare to manual reading.*

2.3 Metodologi Pengembangan Sistem

2.3.1 Pengertian SDLC *(System Development Life Cycle)*

Menurut Rosa (2011:24) SDLC atau *System Development Life Cycle* adalah proses mengembangkan atau mengbah suatu sistem perangkat lunak dengan menggunakan sistem-sistem perangkat lunak sebelumnya.

Menurut Kendall (2010:11) *System Development Life Cycle* (SDLC) adalah pendekatan melalui beberapa tahap untuk menganalisis dan merancang sistem yang dimana sistem tersebut telah dikembangkan dengan sangat baik melalui penggunaan siklus kegiatan penganalisis dan pemakai secara spesifik. Beberapa aktivitas muncul secara simultan dan aktifitas tersebut dilakukan secara berulang-ulang. Lebih berguna lagi memikirkan bahwa SDLC bisa dicapai dalam tahap-tahap (dengan aktifitas yang saling tumpang tindih satu sama lainnya dan menuju tujuan terakhir) dan tidak dalam langkah-langkah terpisah.

SDLC dimulai tahun 1960-an, untuk mengembangkan sistem skala usaha besar secara fungsional untuk para konglomerat pada zaman itu. Sistem-sistem yang dibangun mengelola informasi kegiatan dan rutinitas dari perusahaan-perusahaan yang berpotensi memiliki data yang besar dalam perkembangannya.

System Development Life Cycle atau yang dikenal dengan sebutan SDLC adalah alat bantu proses yang digunakan oleh analisis sitem membangaun dan mengembangkan sebuah sistem informasi (Rudy, 2012:11). Ada beberapa model SDLC, model yang populer dan banyak digunakan adalah *waterfall*. Teori model ini juga dikemukakan dalam buku *Software Engineering* karya Roger S. Pressman, yang menggambarkan proses pengembangan software mengikuti skema air terjun atau disebut *the waterfall model*. Model SDLC air terjun sering juga disebut model sekuensial linier *(Sequential Linier)* atau alur hidup klasik *(classic life cycle)*. Tujuannya adalah untuk memudahkan dalam pengembangan sistem informasi menjadi lebih cermat, terstruktur dan mengikuti metode yang telah ditentukan secara terurut. Sebuah fase tidak bisa dikerjakan sebelum fase sebelumnya telah selesai dikerjakan. Metode ini sering digunakan karena sifatnya general dalam proyek manajemen.

Kelebihan dari *Waterfall Development Methodology* adalah:

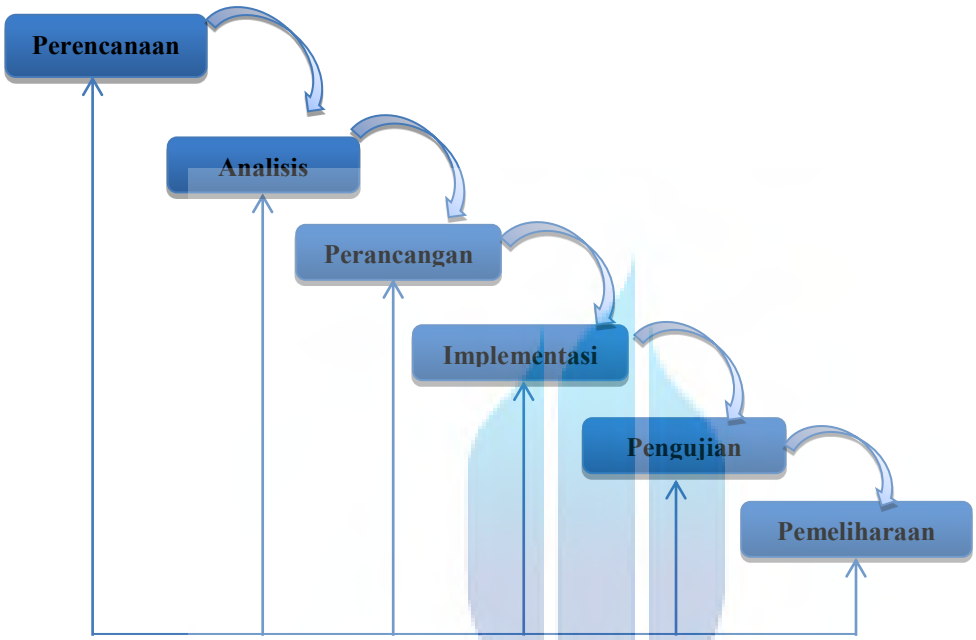
- Merupakan model pengembangan paling handal dan paling lama digunakan.
- Meminimalisasi pengubahan sistem pada saat proses pengembangan perangkat lunak.
- Cocok untuk *system software* berskala besar
- Cocok untuk *system software* yang bersifat umum
- Pengerjaan *project system* akan terjadwal dengan baik dan mudah dikontrol.
- Jadwal menjadi lebih menentu.

Kekurangan dari *Waterfall Development Methodology* adalah :

- Persyaratan sistem dan rincian proses harus digambarkan dengan sangat jelas dan sebaik mungkin tidak berubah-ubah.
- Sulit untuk mengadaptasi jika terjadi perbuahan spesifikasi pada suatu tahapan pengembangan.
- Terjadi selisih waktu yang cukup lama antara pengajuan sistem dan pembaharuan sistem.

2.3.2 Perancangan SDLC (Model Waterfall)

Terdapat enam langkah fase dalam siklus SDLC. Jumlah langkah SDLC pada referensi lain mungkin berbeda, namun secara umum adalah sama. Berikut adalah fase-fase yang digambarkan dalam metode *waterfall* sebagai berikut:



Gambar 2. 6: Metode *Waterfall* (Nugroho:2010)

1. Perencanaan

Dalam tahap ini, menjelaskan dan mengargmentasi untuk melanjutkan proyek yang telah dipilih.Rencana kerja yang matang juga disusun untuk menjalankan tahapan-tahapan lainnya. Pada tahap ini ditentukan secara detail rencana kera yang harus dikerjakan, durasi yang diperlukan masing-masing tahap, sumber daya manusia, perangkat lunak, dokumentasi, perangkat keras, maupun finansial diestimasi. Pembuatan perencanaan ini bukan langkah mudah karena untuk mengestimasi beban kerja dan durasi dari masing-masing tahap dibutuhkan pengalaman yang cukup banyak. Kesalahan pada tahap ini akan mengakibatkan keuntngan yang diperoleh tidak maksimal, bakan bisa rugi. Pada tahapan ini peran manajemen sistem informasi berpengalaman sangat dibuthkan.

2. Analisis

Tahap kedua, adalah tahap analisis, yaitu tahap dimana kita berusaha mengenali segenap permasalahan yang muncul pada pengguna dengan mendekomposisi dan merealisasikan komponen-komponen sistem. Tujuan utama dari tahap analisis adalah untuk memahami dan mendokumentasikan kebutuhan bisnis dan persyaratan proses dari sistem baru. Menganalisa kebutuhan sebagai bahan dalam membuat spesifikasi di tahapan selanjutnya.

3. Perancangan

Tahap perancangan *(design),* dimana kita mencoba mencari solusi permasalahan yang didapat dari tahap analisis, tahapan mengubah kebutuhan yang masih berupa konsep menjadi spesifikasi sistem yang riil untuk diimplementasikan.Jika pada tahapan analisis *(from requirement to specification),* maka tahapan desain adalah *(from specification to implementation)*.Jadi, bagaimana membuat spesifikasi yang detail untuk bisa diimplementasikan.

4. Implementasi

Tahap implementasi, dimana kita mengimplementasikan perancagnan sistem ke situasi nyata.Di sini kita mulai berurusan dengan pemilihan perangkat keras dan penyusunan perangkat lunak aplikasi (pengkodean / *coding*).

5. Pengujian

Tahap kelima adalah pengujian *(testing)*, yang dapat digunakan untuk menentukan apakah sistem/perangkat lunak yang kita buat sudah selesai dengan kebutuhan pengguna atau belum.Jika belum, proses selanjutnya adalah bersifat *iteratif*, yaitu kembali ke tahap-tahap sebelumnya.

6. Pemeliharaan

Tahap pemeliharaan/perawatan di mana kita mulai melakukan pengoperasian sistem dan jika diperlukan melakukan perbaikan-perbaikan kecil. Kemudian jika waktu penggunaan sistem habis, maka kita akan masuk lagi pada tahap perencanaan *(design)*.

2.4 Unified Modelling Language (UML)

2.4.1 Pengertian UML

UML singkatan dari *Unified Modelling Language*, UML adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek (Adi, 1020:6). Sedangkan

menurut (Widodo, 2011:7) UML merupakan alat komunikasi yang konsisten dan standard dalam mendukung para pengembang sistem saat ini.

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa *Unified Modelling Language* (UML) adalah sebuah alat komunikasi bahasa yang berdasarkan diagram, deskripsi atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis orientasi objek yang standar. Pentingnya UML disinilah ada standarisasi pemodelan sistem. (Widodo:2011) dalam bukunya berjudul Menggunakan UML menjelaskan bahwa sebelum ada UML, para pengembang bahasa sistem sulit untuk berkomunikasi satu sama lain, ada kira-kira 50 jenis notasi dan grafik yang menggambarkan bahasa pemrograman berorientasi objek pada waktu itu. Dengan adanya UML, para pengembang sistem diharapkan melakukan standarisasi pemodelan sistem.Saat ini, versi UML yang digunakan adalah versi 2.0 yang merupakan hasil dari pengembangan metode yang dikreasikan oleh Gardy Booch, Jim Raumbaugh dan Ivar Jacobson.

Adapun tujuan pemodelan UML adalah sebagai sarana analisis, pemahaman, visualisasi dan komunikasi antar anggota tim pengembang agar lebih mudah dipelajari dan dipahami dalam pandangan pikir yang saman, serta sebagai sarana dokumentasi dalam pembangunan sistem.

2.4.2 Diagram – Diagram UML

Beberapa literature menyebutkan bahwa UML menyediakan sembilan jenis diagram. Namun kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Diagram yang sering digunakan adalah Diagram *Use Case*, Diagram Aktivitas *(Activity Diagram)*, Diagram *Sequence*, Diagram *Class*.

2.4.3 Use Case Diagram

Use case diagram merupakan gambaran lengkap tentang interaksi yang terjadi antara aktor dengan sistem/perangkat lunak yang sedang kita kembangkan secara tidak terperinci. Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor. Diagram ini sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibuthkan serta diharapkan pengguna (Widodo, 2011:10). Membuat *use case diagram* yang komprehensif merupakan hal yang sangat penting dilakukan pada tahap analisis. Dengan menggunakan *use case diagram*, kita akan mendapatkan banyak informasi yang berkaitan dengan apa yang terjadi dalam sistem/bisnis.

Komponen pembentuk diagram *use case* adalah:

1. Aktor *(actor),* menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use case*, aktifitas/sarana yang disiapkan oleh bisnis/sistem.
3. Hubungan *(link)*, aktor mana saja yang terlibat dalam *use case*, dan bagaimana hubungan *use case* dengan *use case* lain. Ada hubungan antar *use case*. Digolongkan menjadi 2: yaitu extend digambarkan dengan keterangan <<*extend*>>, dan include digambarkan dengan keterangan <<*include*>>. Berikut perbedaannya:

Tabel 2. 1: Perbedaan *include* dan *extend* pada *use case* (Widodo:2011)

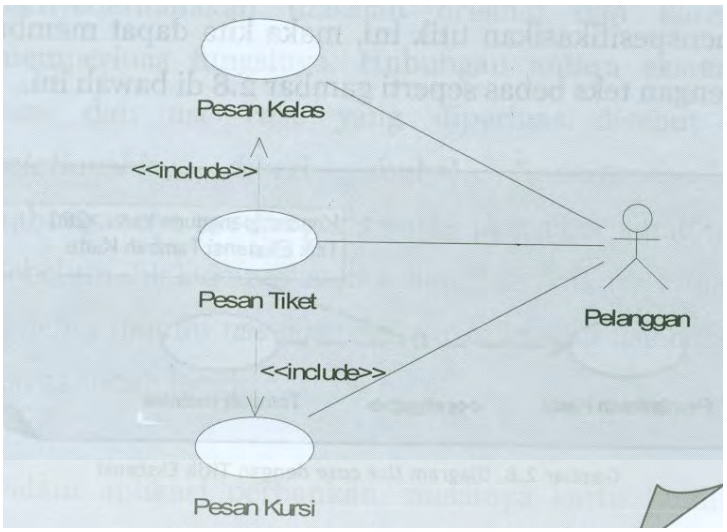
Include	Extend
Use case terpanggil *(included use case)* selalu diperlukan oleh *use case* dasar	*Use case* ekstensi tidak selalu dibutuhkan oleh *use case* dasar
Yang memutuskan kapan dipanggilanya *use case included* adalah *use case* dasar	Yang memutuskan kapan dipanggilnya *use case extend* adalah *use case extend* itu sendiri
Panah hubungan dari use case dasar ke *use case include*	Panah hubungan dari *use case extend* ke *use case* dasar

Terdapat juga relasi yang berhubungan dengan *actor*, antara lain :

Relasi	Fungsi	Notasi
Asosiasi (*association*)	Lintasan komunikasi antara *actor* dengan *use case*.	———
Extend	Penambahan perilaku ke suatu *use case* dasar.	------>
Generalisasi *Use Case*	Menggambarkan hubungan antara *use case* yang bersifat umum dengan *use case-use case* yang bersifat lebih spesifik.	——▷
Include	Penambahan perilaku ke suatu *use case* dasar yang secara eksplisit mendeskripsikan penambahan tersebut.	------>

Gambar 2. 7: Hubungan Relasi *Use Case* (Nugroho:2010)

Use case biasanya disertai dengan penjelasan narasi yang dirangkum dalam tabel yaitu disebut *Use Case Description. Use Case Description* merupakan tabel yang digunakan untuk membuat dan menjelaskan keterangan terperinci mengenai tiap-tiap *use case*. Terdapat istilah *pre condition* dan *post condition*, fungsinya adalah memberikan informasi penting mengenai keadaan sistem sebelum dan sesudah *use case*. Hal ini dapat dilakukan dengan memberikan penjelasan singkat atau dapat pula berupa nama *Use-Case*.



Gambar 2. 8: Contoh *Use Case* (Widodo:2011)

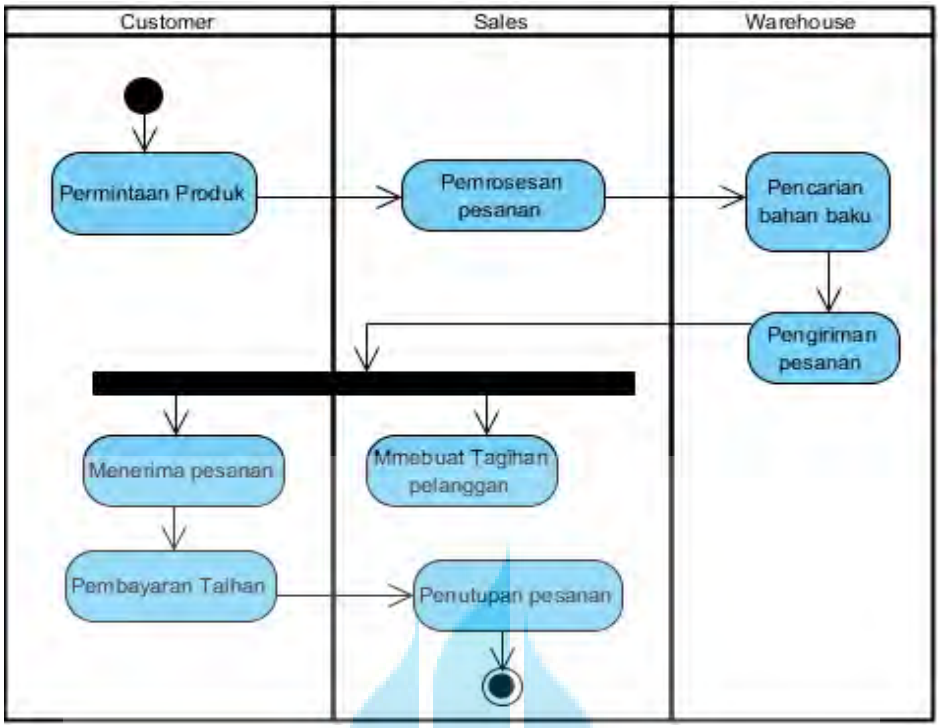
`

Tabel 2. 2: Relasi-relasi dalam *Use Case* (Nugroho:2010)

Nama *Use Case*	Pesan tiket
Aktor	Pelanggan
Deskripsi	Pelanggan harus sudah mempunyai data kapan dia berangkat, dikelas apa dan dikursi nomor berapa
Pre Condition	Pelanggan harus sudah mempunyai data kapan dia berangkat, dikelas apa dan dikursi nomor berapa
Tindakan	Melakukan pemesanan tiket, mengisi data pribadi lalu submit
Post Condition	Mengetahui nomor pemesanan tiket, beserta data tentang jadwal, kelas dan kursi

2.4.4 *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas yang melalui program, dari titik mulai yang sudah didefinisikan sebelumnya, sampai ke titik akhir (Rudy, 2012:150). Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan dengan sistem.



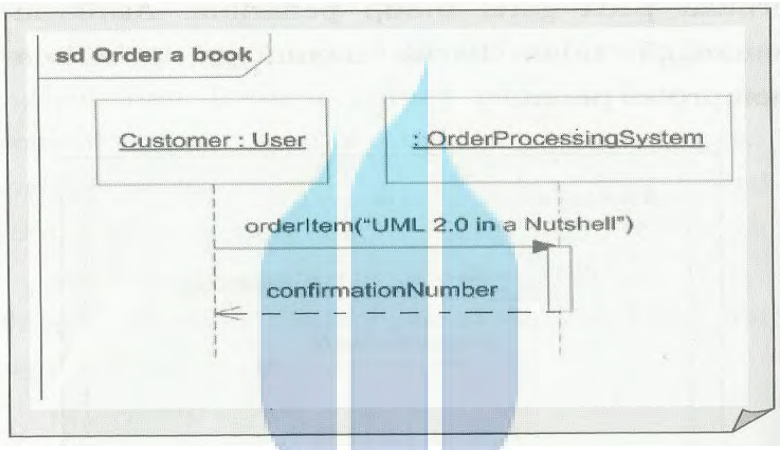
Gambar 2. 9: *Activity Diagram* (Nugroho:2010)

Tabel 2. 3: Komponen *Activity Diagram* (Nugroho:2010)

Notasi	Keterangan	Simbol
Initial Activity	*Initial Activity* sebagai awal dari aktivitas modul sistem aplikasi.	
Activity	*Activity* menunjukkan aktivitas yang dilakukan.	
Final Activity	*Final Activity* menunjukkan akhir dari aktivitas	
Decisions	*Decisions* menunjukkan aktivitas yang harus dipilih apakah pilihan pertama atau kedua	
Signal	*Signal* sebagai pengirim dan penerima pesan dari aktivitas yang terjadi. Sinyal terdiri dari 2 (dua) jenis, yaitu sinyal penerima yang digambarkan dengan poligon terbuka dan sinyal pengirim dengan yang digambarkan dengan convex poligon.	
Concurrent Activities	*Concurrent Activities* menggambarkan atktivitas yang dilakukan bersamaan atau paralel.	

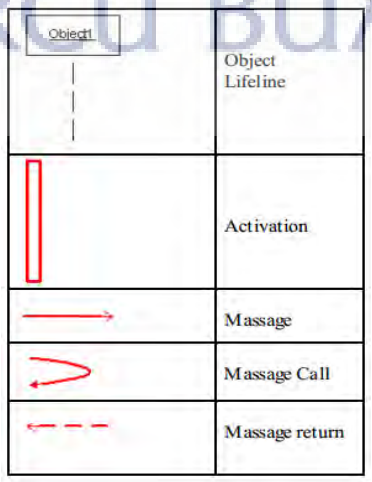
2.4.5 *Sequence Diagram*

Diagram ini memperlihatkan interaksi yang menekankan pada urutan pertukaran pesan/data dalam suatu waktu tertentu yang dilakukan oleh sekumpulan objek atau aktor yang mengerakan pekerjaan (Widodo:2011). *Sequence Diagram* biasanya tersusun dari elemen obyek, *interaction* dan *message*.*Interaction* menghubungkan 2 obyek dengan pesannya.Diagram ini menjelaskan aspek dinamis dari sistem yang sedang dibangun. Untuk satu *use case* bisa dibuat beberapa *sequence diagram*, karena satu *use case* biasanya terdiri dari beberapa aktivitas yang harus dilakukan dan masing-masing aktivitas ini bisa direpresntasikan dalam satu *sequence diagram.*



Gambar 2. 10: Contoh *Sequence Diagram* (Widodo:2011)

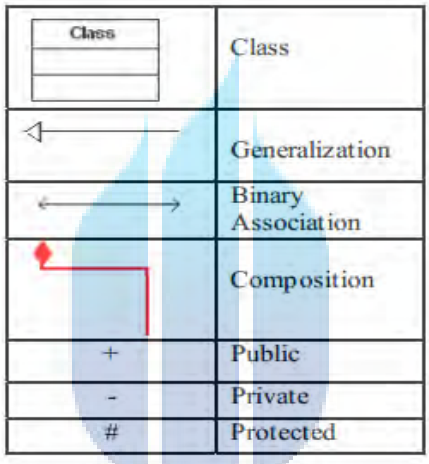
Berikut merupakan komponen utama dalam sequence diagram:



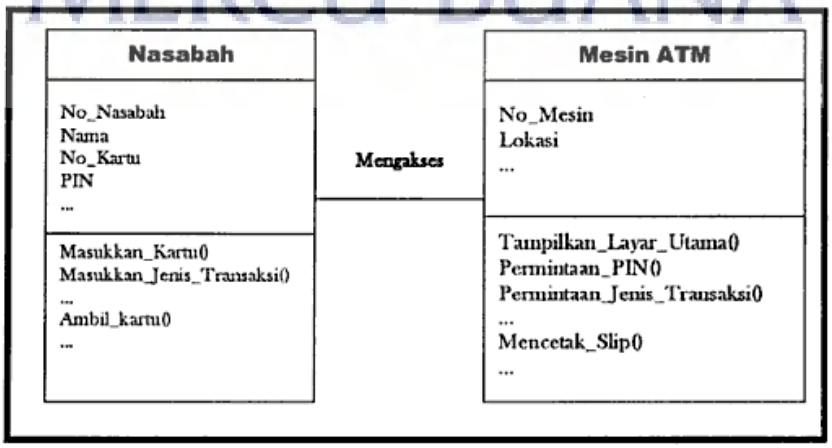
Gambar 2. 11: Komponen *Sequence Diagram* (Widodo:2011)

2.4.6 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi (Rosa, 2011:122). Diagram ini memperlihatkan himpunan kelas-kelas dan objek. Objek adalah entitas yang memiliki atribut (anggota komponen objek), karakter/sifat *(behaviour)* dan kadangkala disertai kondisi. Contoh: siswa, negara, kota sedangkan kelas adalah penggambaran satu set objek yang memiliki atribut dan *behaviour* yang sama (Widodo, 2011:39). Komponen *class diagram*:



Gambar 2. 12: Komponen *Class Diagram* (Nugroho:2011)



Gambar 2. 13: *Class Diagram* (Nugroho:2011)

Untuk atribut, kita bisa menentukan yang secara akal sehat bisa menempel pada kelas tersebut. Sedangkan metode, bisa berasumsi yang didapat dari *sequence diagram.*

2.5 Perangkat Pendukung Sistem

Saat ini telah tersedia berbagai paket perangkat lunak untuk banyak kebutuhan sistem.Dalam perangkat lunak, program yang digunakan telah diuji serta terbukti mampu menghemat waktu dan biaya pengembangan.

2.5.1 *Microsoft Visual Basic .Net*

Microsoft Visual Basic .NET adalah sebuah *tools* untuk mengembangkan dan membangun aplikasi yang bergerak di atas sistem *.NET Framework* dengan menggunakan bahasa *BASIC*. Dengan menggunakan *tools* ini, para *programmer* dapat membangun aplikasi *Windows Forms*, aplikasi web berbasis *ASP .NET*, dan juga aplikasi *command-line*. *Tools* ini dapat diperoleh secara terpisah dari beberapa produk lainnya (seperti *Microsoft Visual C++, Visual C#,* atau *Visual J#*), atau juga dapat diperoleh secara terpadu dalam *Microsoft Visual Studio .NET.*

Bahasa *Visual Basic .NET* sendiri menganut paradigma bahasa pemrograman berorientasi objek yang dapat dilihat sebagai evolusi dari *Microsoft Visual Basic* versi sebelumnya yang diimplementasikan di atas *.NET Framework*. Versi pertama dari *Visual Basic .NET* adalah *Visual Basic .NET 2002* yang dirilis pertama kali pada bulan Februari 2002. Hingga saat ini, sudah berkembang cukup pesat. Dalam penelitian ini, penulis menggunakan *Visual Basic .NET 2005*.

Gambar 2. 14: Contoh Tampilan *Visual Basic .NET*

2.5.2 *Microsoft SQL Server 2008*

SQL Server adalah sebuah RDBMS *(Relational Database Management System)* yang dirancang untuk mendukung aplikasi dengan arsitektur *client/server* ataupun *stand-alone desktop system*. Pada *SQL Server* kita bisa melakukan pengambilan dan modifikasi data dengan cepat, membuat objek-objek yang sering digunakan pada aplikasi bisnis seperti membuat *table, function, stored procedure, trigger* dan *view.*

SQL Server kompatibel dengan beberapa data *access interface* yang digunakan dalam *development tool* seperti *Visual Basic .NET, Power Builder, Delphi* dan sebagainya. *Database SQL Server* dapat diakses dengan menggunakan *Microsoft Jet Engine and Data Access Object* (DAO), *Remote Data Object* (RDO), *ActiveX Data Object* (ADO), OLEDB, ODBC, *SQL Server built-in Library* dan *interface* dari *third party* lainnya.

Gambar 2. 15: Contoh Tampilan *SQL Server*