BAB 2

LANDASAN TEORI

2.1. Pengertian *Software*

Komputer adalah mesin yang menerima informasi dan menyimpan data kedalam bentuk digital. Komputer merupakan mesin yang manjalankan pemrosesan, kalkukasi dan intruksi yang telah diberikan oleh program perangkat lunak atau perangkat keras.

Komputer terbuat dari beberapa bagian dan komponen yang memfasilitaskan sebuah fungsi. Komponen tersebut terbagi menjadi dua bagian utama yaitu perangkat keras dan perangkat lunak yang saling membutuhkan satu dengan lainnya.

Software atau perangkat lunak mengacu ke program yang mengontrol perangkat keras untuk menghasilkan hasil yang diinginkan. Menurut Hamacher, Vranesic, Zaky dan Manjikian (2012, p. 149) Perangkat lunak menjadi faktor utama yang mengkontribusi fleksibilitas dan kegunaan komputer. Program utilitas memungkinkan pengguna untuk *create, execute* dan *debug* aplikasi software.

Software terdiri dari dua macam, yaitu *system software* dan *application software. System Software* adalah koleksi program yang dibuat untuk layanan program lainnya. Sebagian *system software* memproses kompleks tetapi pasti pada struktur informasi. Aplikasi sistem lainnya sebagian besar memproses data tak tentu (Pressman dan Maxim, 2015, p. 6). Sedangkan *application software* merupakan program yang membantu kebutuhan sehari-hari, seperti e-mail, *word processor,* dan *spreadsheet.* Application software juga merupakan *stand-alone program* yang menyelesaikan keperluan bisnis yang spesifik (Pressman dan Maxim, 2015, p. 7).

Deskripsi standar *software* didalam sebuah buku teks menjelaskan sebagai berikut (Pressman dan Maxim, 2015, p. 4):

1. Instruksi yang ketika di eksekusikan memberikan fitur, fungsi dan performa yang diinginkan.

2. Sruktur data yang memungkinkan program untuk memanipulasi informasi secara memadai.
3. Informasi deskriptif di kedua *hard copy* dan *virtual form* yang menggambarkan operasi dan penggunaan program.

2.1.1. Karakteristik *Software*

Software mempunyai karakteristik yang berbeda dibanding dengan *hardware,* karakteristik tersebut sebagai berikut (Pressman dan Maxim, 2015, p. 5):

1. *Software* dikembangkan atau direkayasa bukan diproduksi seperti halnya hardware. Walaupun ada beberapa kesamaan antara pengembangan software dan produksi hardware, keduanya sangat berbeda.
2. *Software* tidak akan "*wear out*" atau rusak. Dibandingkan dengan *hardware, hardware* lebih mudah mendapatkan kerusakan fisik yang tidak akan didapatkan oleh *software*. Walaupun dari masa prototipe *hardware* sering mengalami perbaikan desain, pada waktu panjang hardware akan mengalami masalah fisik yang membuat produk tersebut mengalami kerusakan. Contoh kerusakan tersebut adalah kerusakan karena getaran, tumpukan debu, temperatur yang ekstrim dan segala macam gannguan dari lingkungan sekitar. *Software* tidak mengalami masalah seperti ini.
3. *Software* dibangun sesuai yang diinginkan atau sesuai dengan pesanan yang telah disepakati.

2.1.2. Kategori *Software*

Software yang mendominasi industri pada zaman sekarang dibagi menjadi empat kategori (Pressman dan Maxim, 2015, p. 9), keempat kategori tersebut ialah:

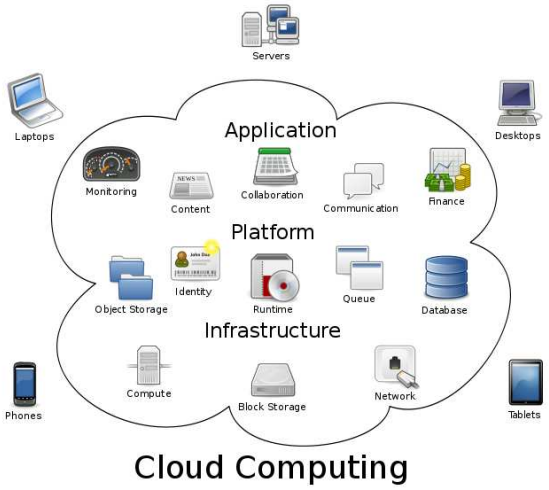
1. *WebApps*

   WebApps atau *Web Application* adalah program yang berjalan di bagian *web server* yang hanya dapat diakses dengan *web browser.* Dengan ini, tidak dibutuhkan pembuatan program untuk menyesuaikan *operating system* (OS) komputer lain.

2. *Mobile Applications*

Pengertian dari *app* telah berubah sejak munculnya ponsel pintar. Pengertian *app* pada zaman sekarang lebih cenderung ke aplikasi yang dibuat khusus untuk ponsel pintar yang berjalan dengan *Operating System Android, iOS,* dan *Windows Mobile. Mobile Application* memberikan *user* layanan yang sama dengan komputer. Perbedaannya *mobile application* memiliki *memory software* yang sangat ringan dibandingkan dengan *software* komputer.

3. *Cloud Computing*



Gambar 2.1 Cloud Computing (Sumber: Pressman dan Maxim, 2015)

Cloud Computing merupakan sebuah infrastruktur dimana *user* dapat mengakses dan membagi data *resources* dari mana saja menggunakan perangkat elektronik komputer. *Cloud Computing* merupakan komputer eksternal yang dapat diakses menggunakan aplikasi *web browser*. Contoh salah satu *cloud computing* adalah Microsoft Office 365.

4. *Product Line Software*

Satu set sistem *software* intensif yang berbagi dengan seperangkat fitur umum, dikelola untuk memenuhi kebutuhan spesifik dari segmen pasar atau misi tertentu dan dikembangkan dari piranti inti aset umum dengan cara yang telah ditentukan.

Keuntungan dari *Product Line Software* dapat membantu organisasi dalam mengatasi masalah yang disebabkan oleh kekurangan sumber daya. Organisasi besar maupun kecil telah mengetahui keuntungan dari strategi ini jika di implementasi dengan cara yang pandai.

2.1.3. Rekayasa *Software/* Piranti Lunak

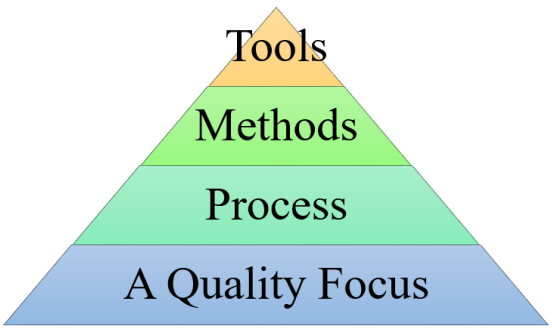
Somerville (2010, p. 7) menjelaskan rekayasa piranti lunak adalah *engineering dicipline* yang berkaitan dengan semua aspek produksi perangkat lunak dari tahap awal spesifikasi sistem menuju ke tahap pemeliharaan sistem.

Tetapi, menurut IEEE [IEE93a] rekayasa piranti lunak adalah (Pressman dan Maxim, 2015):

1. Penerapan pendekatan dari sistematis, disiplin, terukur terhadap pengembangan, operasi, dan pemeliharaan perangkat lunak.
2. Studi tentang pendekatan seperti halnya pada nomor satu.

2.1.3.1. Lapisan Rekayasa *Software/* Piranti Lunak

Rekayasa *Software/*piranti lunak adalah teknologi berlapis, menurut Pressman (2015, p. 15) teknologi tersebut memiliki 4 lapisan. Lapisan dari Rekayasa piranti lunak adalah:



Gambar 2.2 Lapisan rekayasa piranti lunak *(Pressman dan Maxim, 2015)*

1. *A Quality Focus*

   Merupakan landasan yang mendukung rekayasa piranti lunak. Setiap pendekatan teknik terhadap kualitas harus bergantung dari komitmen organisasi. Manajemen kualitas total, *Six Sigma*, dan filosofi serupa mendorong proses budaya perkembangan, dari budaya tersebut pada akhirnya mengarah ke pengembangan pendekatan rekayasa piranti lunak yang efektif.

2. *Process*

   Layer proses sebagai pondasi rekayasa piranti lunak yang mengikat lapisan teknologi dan memungkinkan pengembangan perangkat lunak komputer yang rasional dan tepat waktu.

3. *Method*

   Lapisan metode menyediakan teknik dalam pembuatan perangkat lunak secara terperinci dan efektif. Mencakup beberapa macam tugas yang terdiri dari komunikasi, syarat analisis, model desain, konstruksi program, pengujian program, dan dukungan.

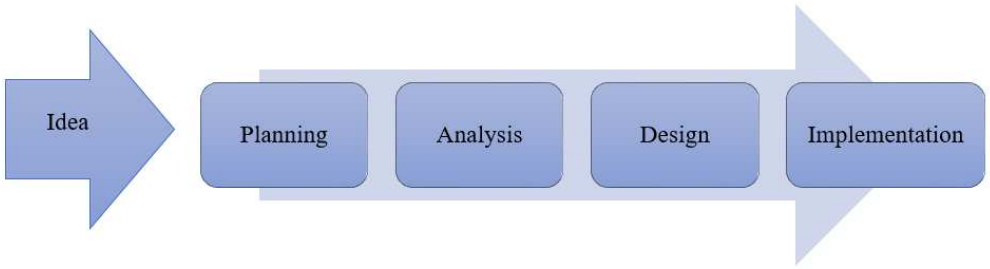
4. *Tools*

   Tools menyediakan dukungan otomatis dan semiotomatis untuk *process* dan *method*.

2.2. **Proses Model**

Menurut Pressman (2015, p. 40) proses model menyediakan arahan spesifik untuk pekerjaan rekayasa piranti lunak. Pada awalnya, proses model diusulkan untuk menertibkan dan memberi arahan perancangan perangkat lunak.

2.2.1. System Development Life Cycle



Gambar 2.3 *System Development Life Cycle* (Sumber: Denis, Wixom, Roth, 2012)

System Development Life Cycle (SDLC) adalah prosedur struktural dalam pengembangan perangkat lunak. SDLC memiliki lima tahap, yaitu:

1. *Planning*

   Tahap *planning* adalah proses dasar untuk memahami mengapa sistem informasi harus dibangun dan menentukan bagaimana tim akan membangunnya (Denis, Wixom, Roth, 2012, p. 13).

2. *Analysis*

   Pada tahap analisis, untuk siapakah program tersebut dibuat, apa yang akan dilakukan program tersebut, dan kapan akan tersedia, haruslah menjawab pertanyaan tersebut terlebih dahulu (Denis, Wixom, Roth, 2012, p. 13). Persyaratan tersebut akan terpenuhi jika tim telah melakukan penelitian.

3. *Design*

   Menurut Alan (2012, p. 14), tahap perancangan memutuskan bagaimana sistem akan beroperasi dalam istilah seperti perangkat keras, perangkat lunak, dan infrastruktur jaringan yang akan ada juga program spesifik, database, dan file yang akan dibutuhkan.

   Menurut Alan (2012, p. 15) fase desain ada empat tahap, yaitu sebagai berikut:

   1. Strategi desain harus ditentukan jika pengembangan program akan dilakukan oleh *programmer* perusahaan tersebut atau membeli program yang sudah jadi dan tinggal mengembangkan dan memakainya.

2. Selanjutnya pengembangan sistem perangkat keras, perangkat lunak, dan infrastruktur lainnya seperti infrastruktur *networking*. Pembuatan desain *User Interface* dilaksanakan pada tahap ini.
3. Database dan spesifikasi data dibangun.
4. Tim analisa mengembangkan desain program, yang mana memberi definisi program yang akan ditulis dan apa yang akan program tersebut lakukan.

4. Implementation

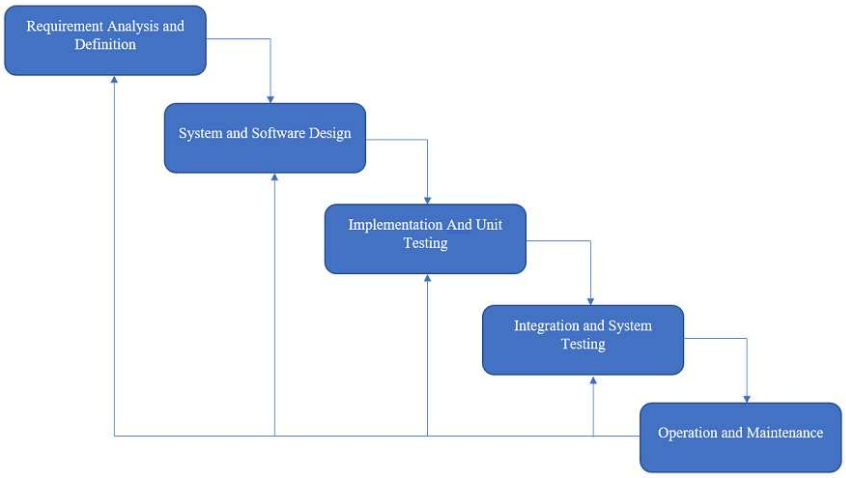
Fase terakhir adalah implementasi pada saat sistem tersebut dikembangkan. Tahap ini yang paling diperhatikan karena biaya dikeluarkan tidaklah kecil. Fase *Implementation* menurut Alan, Barbara, dan Roberta (2012, p. 15) adalah:

1. Tahap awal adalah konstruksi sistem. Sistemnya dirancang, dikembangkan dan di uji agar performanya terpenuhi.
2. Sistem di-*install*.
3. Tim analisis menyusun *support plan* untuk sistem.

Ada beberapa model SDLC yang sering digunakan untuk pengembangan proyek perangkat lunak. Model SDLC yang digunakan adalah:

- Waterfall
- V-Shape
- Incremental Life Cycle
- Spiral

2.2.1.1. Waterfall



Gambar 2.4 Waterfall Model (Sommerville, 2011)

Waterfall Model mengusulkan pendekatan sistematik dan sekuensial terhadap pengembangan software. *Waterfall* mengambil proses aktivitas dasar dari sebuah spesifikasi, pengembangan, validasi, dan evolusi juga mewakili mereka sebagai fase proses yang terpisah seperti spesifikasi persyaratan, perancangan perangkat lunak, implementasi, pengujian, dan sebagainya (Sommerville, 2011, p. 29).

Model *waterfall* merupakan contoh dari proses *plan-driven*. Jadi, sebelum melakukan proses aktivitas harus merencanakan rencana dan jadwal yang akan dikerjakan.

Menurut Sommerville (2011, p. 31) tahap-tahap dari *waterfall model* adalah:

1. *Requirement analysis and definition*

   Sistem berupa servis, kendala dan tujuan yang didirikan dengan konsultasi dengan pengguna sistem. Setelah itu mendefinisikan rincian spesifikasi sistem.

2. *System and software design*

   System design mengalokasi kebutuhan perangkat lunak maupun perangkat keras dengan mendirikan keseluruhan arsitektur sistem.

   Software design meliputi mengidentifikasi dan mendeskripsi desain dasar perangkat lunak abstrak dan relasinya.

3. *Implementation and unit testing*

Pada tahap ini, desain peranti lunak direalisasikan sebagai unit program. *Unit testing* merupakan tahap yang memverivikasi jika unit sistem telah mencapai spesifikasi yang dibutuhkan.

4. *Integration and system testing*

Setiap unit program diintegrasi dan di tes sebagai sistem yang komplit untuk mencapai spesifikasi kebutuhan. Setelah pengujian, sistem perangkat lunak distribusikan ke *user* atau pelanggan.

5. *Operation and maintenance*

Sistem yang telah jadi dipasang di perangkat keras untuk penggunaan praktis. *Maintenance* melibatkan dalam memperbaiki sistem yang terkena *error* yang sebelumnya tidak terlihat ditahap *testing*, meningkatkan implementasi dalam kinerja sistem dan meningkatkam servis sistem jika ada kebutuhan yang baru.

2.3. Multimedia

2.3.1. Pengertian

Multimedia adalah kombinasi dari manipulasi digital teks, grafik, suara, animasi, dan video yang dikembangkan dan dipresentasikan melalui perangkat elektoronik komputer dan sejenisnya (Vaughan, 2011, p. 1). Ketika *end-user* dapat mengontrol, mengkombinasi dan memanipulasi jenis media yang berbeda, hal tersebut termasuk *Interactive Multimedia*. *Interactive Multimedia* menjadi *hypermedia* jika mendesain sistem yang menunjukkan *link* elemen dimana *user* dapat berinteraksi (Vaughan, 2011, p. 53). *Hypermedia* tidak hanya menggunakan teks, tetapi dapat menggunakan media lainnya seperti gambar, video dan animasi.

2.3.2. Tipe-Tipe Komponen Multimedia

Multimedia mempunyai 5 tipe komponen, yaitu:

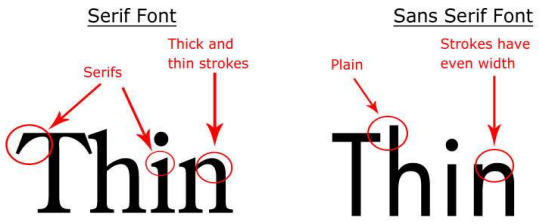
1. Teks

Teks merupakan komponen dasar pada *multimedia application*. Teks berfungsi untuk memberitahukan *user* tentang informasi yang akan ditampilkan. Sejak adanya internet dan *World Wide Web,* teks menjadi sangat penting.

Font merupakan koleksi dari karakter dengan satu ukuran dan ragam dari satu kumpulan *typeface* (Vaughan, 2011, p. 22).

Typeface menurut Vaughan adalah kumpulan dari karakter yang mempunyai banyak tipe ukuran dan ragam (Vaughan, 2011, p. 22).

Kategori dari *typeface* dibagi menjadi dua bagian *serif* dan *sans serif. Serif* merupakan tipe *typeface* yang mempunyai dekorasi yang berbentuk buntut pada ujung sebuah huruf dan memiliki goresan tebal dan tipis. Sedangkan *sans serif* tidak memilikinya dan goresannya memiliki ukuran yang sama.



Gambar 2.5 Perbedaan *Serif dan Sans Serif* (Sumber: visualhierarchy. co)

2. Grafik

Grafik dihasilkan oleh komputer melalui 2 cara, yaitu:

a. Vektor

Vektor merupakan grafik digital yang dibuat menggunakan rumus pernyataan matematika untuk membuat garis, kotak, lingkaran, poligon dan bentuk grafik lainnya di dalam bentuk dua dimensi maupun tiga dimensi. Kelebihan dari vektor yaitu *scalable*

yaitu dapat diperbesar dan diperkecil dengan tidak kehilangan kulalitas gambar (Vaughan, 2011, p. 70).

Contoh *software* yang dapat membuat grafik vektor adalah:

- Adobe Illustrator
- CorelDraw
- Maya 3D
- AutoCad
- Swift 3D

b. Bitmaps

Bit merupakan elemen dasar pada dunia digital, digit elektronik hidup maupun mati, hitam ataupun putih, atau *true (1)* ataupun *false (0). Map* atau peta adalah matriks dua dimensi dari bits tersebut. Jadi *Bitmap* adalah matriks sederhana dari titik kecil yang membentukkan sebuah gambar dan ditunjukkan melalui layar komputer (Vaughan 2011, p. 71).

Contoh *software* yang dapat membuat grafik bitmap adalah:

- Windows Paint
- Adobe Photoshop
- Adobe ImageReady
- CorelDraw
- PaintTool SAI
- Paintbrush
- Paint.NET
- GIMP

3. Suara

Suara adalah fenomena gelombang seperti cahaya, tapi makroskopik dan melibatkan molekul udara terkompresi dan meluas dibawah tidakan dari beberapa perangkat fisik. Contohnya, speaker didalam sistem audio bergetar bolak-

balik dan menghasilkan tekanan gelombang longitudinal yang kita rasakan sebagai suara (Ze-Nian, Mark dan Jiangchuan, 2014, p. 139).

Suara yang digunakan oleh komputer terbagi menjadi dua jenis, yaitu:

- Digital Audio

  Digital Audio diciptakan ketika karakteristik dari suara direpresentasikan menggunakan nomor, proses tersebut disebut sebagai *digitizing* (Vaughan, 2011, p. 106). Proses tersebut mereproduksikan suara kedalam bentuk digital dan disimpan didalam komputer. Jadi, *digital audio* merupakan versi digital analog audio.

- MIDI (*Musical Instrument Digital Interface*)

  MIDI adalah komunikasi standar yang dikembangkan pada awal tahun 1980 untuk alat musik elektronik dan komputer. Ini memungkinkan *synthesizer* musik dan suara dari produsen yang berbeda untuk berkomunikasi satu sama lain dengan mengirim pesan di sepanjang kabel terhubung ke perangkat. MIDI menyediakan sebuah protokol untuk menyampaikan deskripsi rinci tentang skor musikal, seperti catatan, urutan catatan, dan instrumen yang akan memainkan catatan ini (Vaughan, 2011, p. 113).

4. Animasi

Animasi adalah proses yang membuat sesuatu yang statis menjadi hidup dengan menambahkan gerakan ke benda atau grafis statis.

Contoh program yang membuat animasi adalah:

-3DS Max

-Animator Pro

-Flash

5. Video

Video adalah format digital di presentasi multimedia yang menggabungkan teks, grafis, suara dan animasi.

Video dibagi menjadi dua bagian, yaitu:

-Analog Video

-Digital Video

2.3.3. *Storyboard*

Storyboard menggambarkan isi ide awal konsep multimedia dalam rangkaian sketsa untuk sistem yang diharapkan. Menurut Vaughan (2011, p. 295) *Storyboard* adalah sebuah grafik secara garis besar yang mendeskripsikan detail dari projek tersebut. Dengan cara menggunakan kata dan grafis untuk salah satu atau semua layar gambar, suara dan pilihan navigasi dengan secara detail seperti warna yang digunakan, konten teks, atribut dan *font*, bentuk tombol, *style*, respons, dan infleksi suara.

STORYBOARD	
Project:	*Date*:
Screen:	*ScreenID*:
Drawing Area	
Screen Description:	
Link from ScreenID:	*Link to ScreenID*:
Color Scheme:	
Text Attributes: FontFamily: *FontSize :*	
Still Images:	
Audio:	
Video:	
Animation:	

Gambar 2.6 *Storyboard*

Pada bagian *Project* diisi dengan dengan nama projek tersebut sebagai contohnya nama projek Sushiapp. Bagian *Screen* adalah sebagai nomor dari *screen* sedangkan *screenId* merupakan nomor identifikasi dari *screen* tersebut. *Date* dicantumkan kapan storyboard tersebut dibuat.

Drawing Area merupakan tempat dimana dimasukinnya desain awal dari *user interface* dan diperjelaskan pada bagian *Screen Description.*

Link from ScreenID dan *Link to ScreenID* merupakan hubungan antara halaman web yang akan dibuat. *Color Scheme* memberitahu warna yang digunakan pada desain *User Interface. Text Attribute* menjelaskan *font* dan *font size* yang digunakan. *Still Image* adalah gambar yang dipakai untuk mendesain *user interface. Audio* adalah suara yang dipakai untuk mendesain *user interface. Video* adalah video yang dipakai untuk mendesain *user interface.* Dan *Animation* adalah animasi yang dipakai untuk mendesain *user interface.*

2.4. Interaksi Manusia dan Komputer

Interaksi antara manusia dan komputer adalah penelitian mendesain, menjalankan dan menggunakan sistem komputer interaktif yang terfokus antara interaksi manusia dan komputer. Penelitian tersebut juga meneliti pengaruh komputer terhadap individu, organisasi, dan masyarakat.

2.4.1. *Shneiderman Eight Golden Rules*

Delapan *Golden Rules* dari Schneiderman memberikan ringkasan prinsip desain antar muka yang ringkas dan mudah dimengerti. Prinsip dari Schneiderman dapat dipakai untuk desain dan juga evaluasi sistem. *Golden rules* desain antarmuka sebagai berikut (Shneiderman, 2016):

1. *Strive for consistency.*

   Berusaha untuk konsisten dalam aksi urutan, terminologi yang identik harus digunakan dalam petunjuk, menu, dan layar bantuan, serta warna yang konsisten, tata

letak, kapitalisasi, font, dan sebagainya, harus digunakan di seluruh dunia.

2. *Seek universal usability.*

   Mengenali bahwa kebutuhan *user* sangat berbeda. Mulai dari awam sampai ke ahli, usia yang berbeda, variasi internasional, dan keragaman teknologi masing-masing memperkaya spektrum persyaratan yang memandu desain.

3. *Offer informative feedback for every user action.*

   Feedback dari aksi *user* sangat penting bagi pengembangan sistem.

4. *Design Dialogs to yield closure.*

   Memberikan umpan balik kepada *user* jika *user* telah menyelesaikan tugas. Feedback informasi pada penyelesaian sebuah aksi pada sistem membuat *user* puas. Seperti contoh pada *website* jual-beli online, ketika *user* memilih untuk melakukan transaksi pembayaran mendapatkan *ending* bahwa transaksi telah selesai.

5. *Prevent error.*

   Buatlah antar muka yang dimana *user* tidak akan mendapatkan *error* pada saat memakainya. Tetapi jika *user* mendapatkan *error*, selalu sediakan solusi yang dapat *user* ikuti dengan mudah.

6. *Permit easy reversal of action.*

   Untuk menghilangkan kegelisahan *user* dan mendorong eksplorasi sistem harus memiliki cara untuk membaliki ke keadaan sebelumnya.

7. *Keep user in control.*

   User yang mahir ingin mempunyai keinginan untuk mengontrol antar muka/ *interface* dan antarmuka tersebut menjawab aksi *user*. *User* tidak menginginkan antarmuka yang selalu diubah desainnya karena mereka lebih aman menggunakan antarmuka yang mereka kenali.

8. *Reduces short-term memory load.*

> Kebatasan manusia dalam pemrosesan informasi dalam jangka pendek mengharuskan desainer menghindari antarmuka di mana pengguna harus mengingat informasi dari satu tampilan dan kemudian menggunakan informasi itu di layer lain.

2.5. Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah metode visualisasi dan dokumentasi perancangan sistem perangkat lunak. UML menggunakan simbol untuk merepresentasikan grafis dari komponen dan relasi sistem.

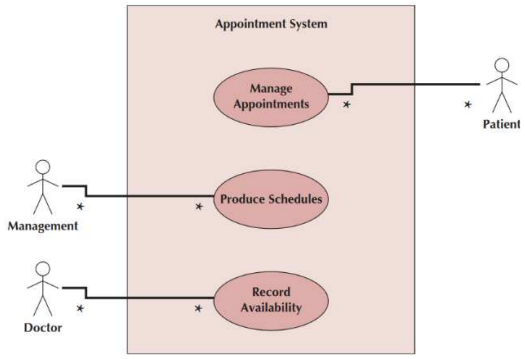
Menurut Denis, Wixom, dan Tegarden (2012, p. 44) UML adalah seperangkat teknik diagram standar yang menyediakan grafis representasi untuk memodelkan setiap proyek pengembangan sistem dari analisis sampai implementasi.

UML didefinisikan sebagai satu set dari empat belas teknik diagram yang digunakan untuk memodelkan sebuah sistem. Diagram tersebut dijadikan menjadi dua bagian utama yaitu *Structure Diagram* dan *Behavior Diagram (*Denis, Wixom dan Tegarden, 2012, p. 39*).*

Structure diagram menyediakan cara untuk merepresentasikan data dan hubungan statis dalam sebuah sistem informasi. Didalam *structure diagram* terdapat *class, object, package, deployment, component,* dan *composite structure design.* Sedangkan *behavior diagram* memberikan analis dengan cara menggambarkan hubungan dinamis antara instansi atau objek yang mewakili sistem informasi bisnis. Dalam *behavior diagram* terdapat *activity, sequence, communication, interaction overview, timing, behavioral state machine, protocol state machine* dan *use-case (*Denis, Wixom dan Tegarden, 2012, p. 39-41*).*

2.5.1. *Use-Case Diagram*

> Use-case modeling merupakan pendekatan yang memfasilitasi pengembangan yang berkonsentrasi kepada penggunaan. Use-case modeling memfasilitasi dan mendorong keterlibatan *user* yang merupakan salah satu faktor sukses untuk kelancaran projek.



Gambar 2.7 *Use-Case* Diagram *(*Denis, Wixom dan Tegarden, 2012*)*

Use-case diagram menyediakan cara sederhana dan mudah untuk berkomunikasi dengan *user* sesuai dengan sistem yang akan lakukan, *Use-case diagram* diambil saat untuk mengumpulkan dan menentukan persyaratan sistem. *Use-case diagram* menggambarkan cara sederhana dari fungsi utama sebuah sistem dan juga *user* yang akan berinteraksi dengan sistem tersebut *(*Denis, Wixom, dan Tegarden, 2012, p. 155*).*

Elemen dari *use-case diagram* sebagai berikut *(*Denis, Wixom, dan Tegarden, 2012, p. 156*)*:

1. *Use Case*

   Use case merupakan proses utama yang dilakukan sistem yang menguntungkan actor. *Use case* juga merupakan representasi dari potongan utama sebuah fungsi sistem. *Use case* digambarkan dalam bentuk elips dengan nama *use case* yang ditampilkan didalam elips.

2. *Actor*

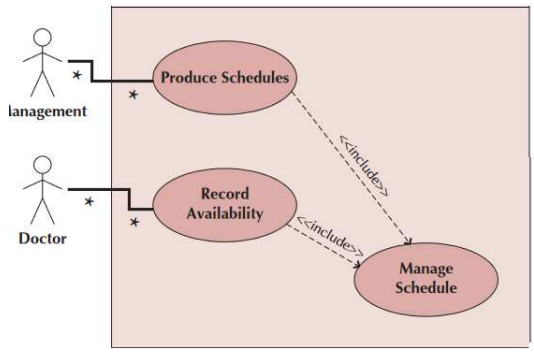
   Use case yang dimulai atau dipicu oleh pihak luar dari sebuah sistem disebut *Actor* atau Aktor. Aktor mewakili elemen utama di lingkungan tempat sistem beroperasi. Aktor dapat menyediakan input ke sistem dan juga dapat menerima output dari sistem dengan cara berinteraksi dengan sistem tersebut. *Actor* digambarkan oleh figur stik jika aktor tersebut adalah manusia dan jika aktor tersebut bukan manusia di representasikan oleh gambar kotak dengan tulisan <<*Actor*>>.

3. *Relasi.*

Digambarkan sebagai garis menghubungkan aktor dengan *use case* atau *use case* dengan *use case* sebagai relasi yang menjelaskan hubungan antar dua simbol. Pengertian relasi berubah sesuai dengan hubungan antara simbol *use case diagram,* diantaranya adalah:

a. *Associations* merupakan relasi yang mendeskripsi hubungan dan interaksi *user* dengan *use case*. Digambarkan sebagai garis *solid.*

b. *Extend* menghubungkan relasi antar *use case* dengan *use case. Extend* merupakan perpanjangan dari use case untuk memasukkan perilaku opsional. Digambarkan dengan sebuah garis putus-putus berpanah dengan label <<*Extend*>>.



Gambar 2.8 Contoh relasi *include* (Denis, Wixom, dan Tegarden, 2012)

c. *Include* adalah relasi antara *abstract use case* dengan *use case*. *Include* merupakan penyertaan fungsionalitas satu *use case* dalam *use case* lainnya. Digambarkan dengan garis putus-putus berpanah yang mengarahkan satu *use case* ke *use case* lain dengan diberikan label <<*include*>>.

d. *Depends On* adalah relasi yang menjelaskan bahwa *use case* tidak dapat dijalankan jika *use case* lainnya tidak dilakukan terlebih dahulu.

2.5.2. *Activity Diagram*

Activity Diagram dapat dilihat sebagai diagram alir data yang canggih digunakan bersamaan dengan analisis terstruktur. *Activity*

Diagram dapat digunakan untuk memodelkan segala sesuatu mulai dari alur kerja bisnis tingkat tinggi yang melibatkan banyak *use case* berbeda, hingga detail *use case* individu, sampai ke rincian spesifik metode individual (Denis, Wixom dan Tegarden, 2012, p. 165-170).

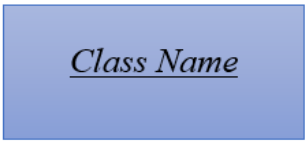
1. *Action* dan *activities.*

   Melakukan kinerja bisnis spesifik, dan meng-gambarkan tindakan manual dan komputerisasi. Simbol didalam diagram aktivitas adalah kotak dengan sudut membulat. Yang membedakan aksi dan aktivitas adalah, aksi dapat diuraikan lebih jauh ke dalam serangkaian aktivitas sedangkan aktivitas merupakan potongan sederhana dari keseluruhan perilaku yang dimodelkan.

Gambar 2.9 *Action* atau *Activities* dari *activity Diagram* (Denis, Wixom dan Tegarden, 2012)

2. *Object Nodes*

   Kegiatan dan tindakan biasanya memodifikasi atau merubah *objects*. *Object nodes* mencontohkan ini kedalam *Activity Diagram.* Simbol pada *Object Nodes* adalah persegi panjang dan nama kelasnya dituliskan didalamnya.



Gambar 2.10 *Object Nodes* dalam *Activity Diagram* (Denis, Wixom dan Tegarden, 2012)

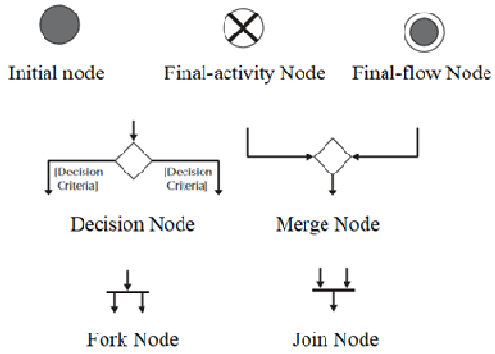
3. *Control Flow* dan *Object Flow*

   Control Flow memodelkan arah pengerjaan melalui proses bisnis. Digambarkan dengan simbol garis *solid* berpanah yang menentukan arah jalurnya. *Control Flow* hanya dapat menunjukkan pada aksi dan aktifitas.

Object Flow memodelkan arah objek melalui proses bisnis. Karena aksi dan aktivitas mengubah *object, object flow* perlu menunjukkan objek sebenarnya yang keluar dan masuk ke *action* dan *activities.* Digambarkan sebagai garis putus-putes berpanah.

Gambar 2.11 *Control Flow* dan *Object Flow* (Denis, Wixom dan Tegarden 2012)

4. Control Nodes



Gambar 2.12 Tujuh Macam *Control Nodes* (Denis, Wixom dan Tegarden, 2012)

Ada tujuh macam *control nodes* didalam *activity diagram*, yaitu:

- *Initial Nodes*

  Menggambarkan awal dari *action* atau *activities.* Digambarkan sebagai lingkaran berisi.

- *Final-activity Nodes*

  Digunakan untuk memberhentikan semua *control flow* dan *object flow* didalam *activity diagram.* Node ini digambarkan menggunakan lingkaran dengan garis silang.

- *Final-flow Nodes*

Digunakan untuk memberhentikan *control flow* dan *object flow* tertentu. Digambarkan dengan lingkaran yang berisi lingkaran berisi.

- *Decision Nodes*

  Digunakan untuk menunjukkan kondisi tes sebenarnya yang menentukan jalan mana yang keluar dari simpul keputusan akan dilalui. Node ini harus ada *guard condition. Guard condition* mewakili nilai uji untuk jalur tertentu yang akan dieksekusi.

- *Merge Nodes*

  Digunakan untuk menyatukan kembali setelah menggunakan *decision nodes.*

- *Fork Nodes*

  Digunakan untuk memisahkan *behavior* ke beberapa bagian *action* atau *activities* yang berjalan secara paralel.

- *Join Nodes*

  Digunakan untuk mengabungkan kembali *action* atau *activities* yang telah dipisah dengan *fork nodes.*

5. *Swimlanes*

   Swimlane dugunakan untuk memisahkan *activity diagram* ke baris dan kolom untuk menentukan aktivitas individual kepada individu atau objek yang bertanggung jawab untuk melaksanakan aktivitas.

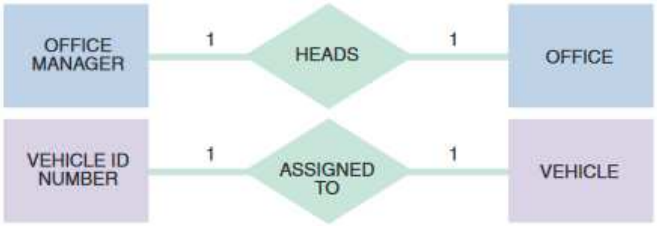
2.6. *Entity-Relationship Diagram*

Entity-Relationship Diagram (ERD) adalah sebuah model yang memperlihatkan relasi logika dan interaksi sesama entitas sistem. ERD menyediakan pandangan keseluruhan sistem dan *blueprint* untuk pembuatan struktur dari data fisik sistem (Shelly B. G. dan Rosenblatt J. H. , 2012, p. 406).

1. Tipe Relasi

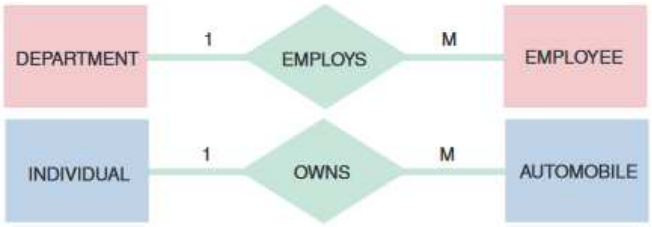
   Ada tiga jenis dari tipe relasi ERD, Diantanya adalah (Shelly B. G. dan Rosenblatt J. H. ,2012, p. 406-407):

a. *One-to-one relationship* terjadi jika tepatnya salah satu dari entitas kedua terjadi untuk setiap permintaan dari entitas pertama.



Gambar 2.13 *One-to-one Relationship* (Shelly B. G. dan Rosenblatt J. H., 2012)

b. *One-to-many Relatonship* ada ketika satu kejadian entitas pertama dapat berhubungan dengan banyak contoh entitas kedua, namun setiap *instance* dari entitas kedua dapat berasosiasi dengan hanya satu instance dari entitas pertama.

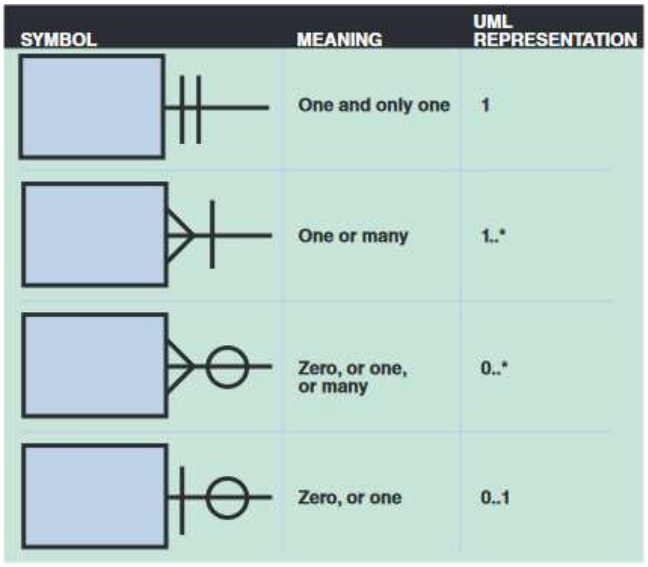


Gambar 2.14 *One-to-many Relationship* (Shelly B. G. dan Rosenblatt J. H., 2012)

c. *Many-to-many Relationship* ada ketika satu *instance* dari entitas pertama dapat berhubungan dengan banyak contoh entitas kedua, dan satu *instance* dari entitas kedua dapat berhubungan dengan banyak contoh entitas pertama.

Gambar 2.15 *Many-to-many Relationship* (Shelly B. G. dan Rosenblatt J. H., 2012)

2. *Cardinality*



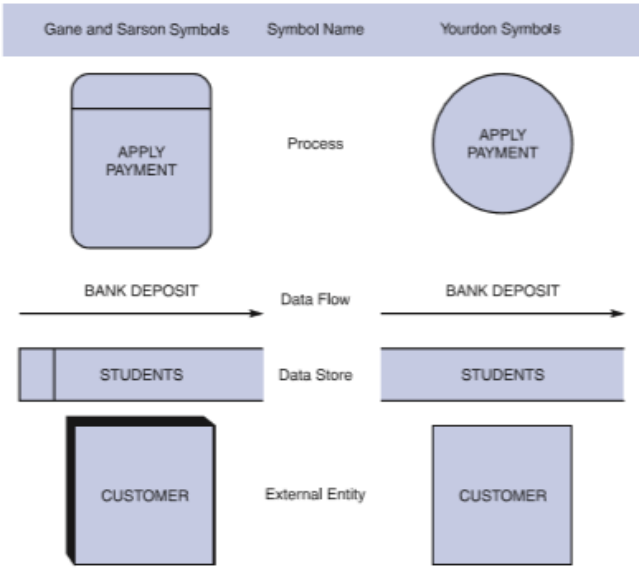
Gambar 2.16 *Crow's Foot Notation* dari Notasi *Cardinality* (Shelly B. G. dan Rosenblatt J. H., 2012)

Kardinalitas menggambarkan hubungan numerik antara dua entitas dan menunjukkan bagaimana instance dari satu entitas berhubungan dengan instance entitas lain (Shelly B. G. dan Rosenblatt J. H. ,2012, p. 408).

Metode yang sering digunakan dari notasi kardinalitas adalah *crow's foot notation* karena menggunakan bentuk seperti lingkaran, bar, dan symbol yang mengindikasi berbagai kemungkinan(Shelly B. G. dan Rosenblatt J. H. ,2012, p. 408). Seperti gambar 2.16, one and only one digambarkan dengan dua gari bar, lingkaran menunjukkan nol, dan *crow foot* menunjukkan *many.*

2.7. ***Data Flow Diagram***

Data Flow Diagram menjelaskan bagaimana data bergerak melalui sistem informasi tetapi tidak memberitahukan logika program dan langkah prosesnya. Satu set dari DFD memberikan model logis yang melihatkan apa yang sistem lakukan dibandingkan bagaimana melakukannya (Shelly B. G. dan Rosenblatt J. H., 2012, p. 200).



Gambar 2.17 Simbol *Data Flow Diagram* (Shelly B. G. dan Rosenblatt J. H., 2012)

Menurut Shelly dan Rosenblatt (2012), DFD menggunakan empat simbol dasar yang merepresentasikan proses, *data flow*, *data stores,* dan *entities.*

1. *Process symbol*

   Sebuah proses menerima *input* data dan menghasilkan *output* yang yang memiliki konten, dan form yang berbeda atau bisa kedua-duanya. Proses berisi logika bisnis yang mengubah data dan prosedur ke hasil yang diperlukan.

   Simbol yang digunakan dalam proses adalah kotak dengan sudut lengkung. Nama dari proses tercantum dalam kotak tersebut. Nama tersebut berindikasi sebagai fungsi.

2. *Data flow symbol*

   Data flow merupakan sebuah jalur diaman data bergerak dari satu sistem informasi ke lainnya. Data flow terdiri dari *item* data tunggal atau terdiri dari kumpulan data.

   Data flow digambarkan dengan garis dengan satu atau dua anak panah. Nama dari sebuah data flow tertulis di atas garis tersebut.

3. *Data stores symbol*

Dalam DFD, *data store* digunakan sebagai representasi data dimana sistem tersebut simpan karena satu atau lebih banyak proses yang digunakan untuk menggunakannya dilain waktu.

Di DFD, Simbol *Gane* dan *Sarson* untuk *data store* adalah persegi panjang datar yang terbuka di sisi kanan dan tertutup di sisi kiri. Nama dari *data store* tertulis pada symbol tersebut dan mengidentifikasi isi data.

4. *Entity Symbol*

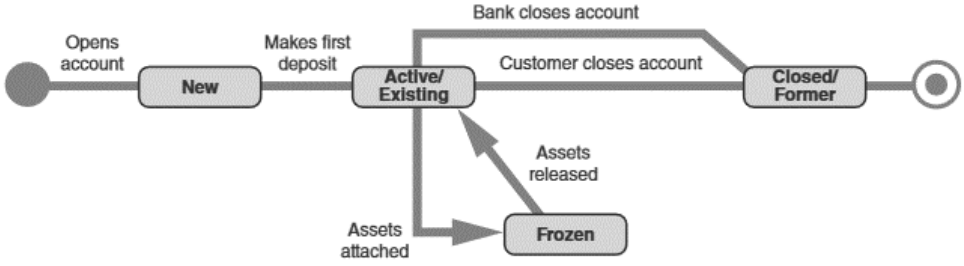
Simbol dari *entity* adalah kotak dengan bayangan dibelakangnya yang memberikan efek tiga dimensi. Nama *entity* tercantum didalam kotak tersebut.

DFD *entity* disebut sebagai *terminator,* karena mereka adalah asal data atau tujuan akhir data tersebut. Sistem analis menyebut *entity* yang memasukkan data ke sistem disebut *source,* sedangkan yang menerimanya disebut *sink.*

2.8. *State Transition Diagram*

State transition diagram menunjukkan bagaimana objek merubah suatu *state* menjadi yang lain, tergantung terhadap *event* yang mempengaruhi objek. Semua keadaan yang mungkin harus didokumentasikan kedalan *state transition diagram* (Shelly B. G. dan Rosenblatt J. H., 2012, p. 265).

Dalam *state transition diagram, state* digambarkan sebagai persegi panjang dengan sudut bulat. Nama dari state tercantum didalamnya. Lingkaran merupakan *initial state* yang dimana awal objek berinteraksi dengan sistem. Garis menunjukkan arah dan menggambarkan tindakan atau peristiwa yang menyebabkan transisi dari satu *state* ke *state* lain. Dan yang terakhir lingkaran didalam lingkaran menunjukkan akhir dari *state* tersebut.



Gambar 2.18 Contoh *State Transition Diagram* (Shelly B. G. dan Rosenblatt J. H., 2012)

2.9. Teori Terkait Penelitian

2.9.1. *Android*

Android adalah sistem operasi didesain untuk perangkat *mobile*. Android dekembangkan oleh Google dan dimiliki oleh Open Hanset Alliance (Gargenta 2011, p. 1). Tujuan Open Handset Alliance adalah untuk memajukan teknologi mobile menjadi teknologi yang meluas, tidak terlalu mahal untuk memilikinya dan mempunyai pengalaman *mobile* yang lebih bagus.

Android merupakan sistem operasi *open source* dan dapat mengakses semua *source code* yang dimiliki android. Mulai dari modul level bawah Linux sampai ke *native library*, dan dari penerapan framework sampai penerapan komplit semuanya dapat diakses (Gargenta, 2011, p. 2).

2.9.2. *Augmented Reality (AR)*

Augmented reality (AR) bertujuan untuk mempersembahkan informasi langsung ke dalam dunia nyata. AR menyatukan dua dunia yaitu dunia virtual dan dunia nyata secara spatial dan kognitif. Jadi, Informasi yang berada di dunia virtual akan tampil di dunia nyata dengan bantuan perangkat AR (Schmalstieg dan Hollerer, 2016, p. 1).

Menurut Azuma pada kertas surveinya menjelaskan *augmented reality* harus mempunyai tiga karakteristik yaitu menggabungkan dunia nyata dan virtual, interaktif secara *real-time*, dan terdaftar dalam bentuk tiga dimensi (Schmalstieg dan Hollerer, 2016, p. 3).

2.9.2.1. Penggunaan *Augmented Reality* (AR)

Ada beberapa kegunaan AR yang telah digunakan saat ini, Contoh diantaranya adalah:

1. Militer

Militer menggunakan AR untuk membantu tentaranya dalam tugas masing-masing. Yang sering digunakan dan diketahui oleh publik dalam kegunaan AR di militer adalah *Heads-Up Display* (HUD). HUD

membantu pilot pesawat tempur untuk mengetahui semua data yang ditampilkan oleh komputer didalam pesawat tersebut. Mulai dari menampilkan ketinggian altitude pesawat, kecepatan pesawat dan garis horizon.

2. Medis

Medis menggunakan AR untuk melatih pelajar medis dalam melakukan operasi didalam ruangan yang terkontrol. AR mengurangi risiko opearsi dengan memberi ahli bedah persepsi yang lebih baik.

3. Game

Telah banyak *game* yang menggunakan teknologi berbasis *augmented reality*. Mulai dari game berbasis mobile, console maupun komputer. Dengan kemajuan teknologi, aplikasi game berbasis *augmented reality* semakin berkembang. Salah satu game terkenal yang menggunakan teknologi AR adalah Pokemon GO.

4. Televisi

AR telah digunakan didalam televisi. Salah satu contoh yang sering dilihat adalah ketika menonton pertandingan sepak bola. Pertandingan tersebut adalah siaran langsung yang dimana ditambahkan dengan informasi pertandingan seperti waktu pertandingan berjalan, skor pertandingan, serta informasi pemain pada layar televisi. Informasi tersebut adalah informasi digital yang disatukan kedunia nyata didalam televisi. Seperti yang didefinisikan sebelumnya bahwa *augmented reality* menyatukan dua dunia yaitu dunia virtual dan dunia nyata.

2.9.2.2. Tipe-tipe *Augmented Reality* (AR)

Ada beberapa tipe dari *augmented reality (AR)* yang telah digunakan, tipe tersebut diantaranya adalah *(Sumber: realitytechnologies.com)*:

1. *Marker based augmented reality*

Gambar 2.19 *Marker Based Augmented Reality (Sumber: realitytechnologies.com)*

Augmented reality tipe ini menggunakan kamera dan marker untuk memunculkan hasil proyeksi AR diatas marker tersebut. Marker dalam tipe ini berupa QR code atau logo yang diatur menjadi marker.

2. *Markerless augmented reality*

Markerless augmented Reality merupakan tipe metode AR yang tidak menggunakan marker untuk tahap pendeteksian. Tipe AR ini menggunakan GPS, digital compas, atau accelero-meter yang berada di ponsel pintar untuk memberikan informasi data lokasi untuk memunculkan proyeksi AR. Tipe *augmented reality* ini di implementasikan dengan lokasi sebagai kunci untuk memastikan agar proyeksi tersebut dapat dimunculkan.

Gambar 2.20 *Markerless Augmented Reality (Sumber: realitytechnologies.com)*

3. *Projection based augmented reality*

Gambar 2.21 *Projection Based Augmented Reality (Sumber: realitytechnologies.com)*

Cara kerja tipe ini adalah menggunakan cahaya untuk memproyeksi cahaya artifisial ke permukaan dunia nyata. Dengan menproyeksikan cahaya tersebut *user* dapat berinteraksi langsung dengan tidak menggunakan ponsel pintar maupun *head-up display*.

2.9.3. Unity

Unity adalah sebuah game engine yang berbasis *cross-platform* dengan IDE yang dikembangkan oleh Unity Technologies. Digunakan untuk membuat video game didalam *platform browser, desktop,* dan *mobile device.* Unity merupakan gabungan dari *game engine,* tempat yang dimana game dapat dibuat dengan *code editor.*

Unity adalah sebuah *authoring tools* yand dirancang untuk PC. Game engine adalah alat yang digunakan untuk video game dan

melakukan keputusan mulai dari rendering. Unity menggunakan bahasa pemograman C++ karena lebih efesien dalam kecepatan komputasi, dan dengan demikian struktur dan perintah *game engine* memerlukan ribuan kode tersebut untuk berfungsi. *Library* yang digunakan adalah library C++ Mono untuk mengirim kode di-*unity* dengan bantuan kompilasi JIT *(Just-in-time*). Dengan bantuan JIT, *engine* seperti Unity mempunyai keunggulan dalam kompilasi cepat. Unity juga mempunyai library lain seperti OenGL, DirectX untuk 3D, Nvidia PhysX dan OpenAL (Goldstone, 2009, p. 1).

Meskipun merupakan sebuah *game engine*, Unity dapat difungsikan untuk membuat aplikasi mobile. Unity merupakan sarana pembuatan aplikasi Augmented Reality karena tersedia SDK Vuforia yang akan digunakan memiliki integrasi yang baik dengan Unity.

2.9.3.1. SDK Vuforia

Vuforia merupakan platform yang digunakan dalam pembuatan aplikasi Augmented Reality dari Qualcomm. Vuforia menyediakan *Software Development Kit* (SDK) yang diperlukan untuk perancangan aplikasi dengan Augmented Reality untuk berbagai perangkat bersistem operasi seperti Android, iOS, dan Windows. (Sumber: library.vuforia.com)

Vuforia dapat terintegrasi dengan program unity. Integrasi ini memudahkan developer dalam alur kerja pengembangan, sinkronisasi fitur baru dan pembenaran *bug* dengan versi Unity. (Sumber: library.vuforia.com)

2.9.4. Adobe Dreamweaver

Dreamweaver adalah program perangkat lunak untuk membuat, menerbitkan, dan mengelola situs web dan konten mobile. Program ini menyediakan *User Interface* bagi *user* untuk membuat dan mengedit halaman web di lingkungan untuk menjadi lebih *user*-friendly. Dreamweaver mendukung banyak bahasa markup, termasuk PHP, HTML, XML, CSS, dan JavaScript.

2.9.4.1. PHP

PHP adalah bahasa *scripting* untuk mengembangkan *dynamic web pages.* PHP dirtafsirkan sebagai *original human-readable script* yang ditulis oleh developer dan diubah menjadi instruksi komputer setiap kali digunakan (Matthews, 2015, p. 212).

Matthews (2015) menjelaskan bahwa PHP menyediakan *programming* dari server atau *server-side programming,* memberikan pengembangan web untuk dapat mengakses server web yang meng-*hosting* situs tersebut, dan semuanya yang disimpan atau dijalankan disana. Yang paling diutamakan bahwa PHP dapat membuat halaman web yang diberikan ke klien agar klien dapat menyesuaikan halaman yang diinginkannya.

Agar PHP bekerja secara efisien, pengembang harus memiliki *tools* atau alat yang dapat mendukung mulai dari penulisan sampai *testing* PHP *scripts. Tools* tersebut adalah:

1. *PHP script writing*
2. *PHP script testing*
3. *Development support*

2.9.5. MySQL

MySQL adalah Sistem Manajemen Database Relasional berbasis *open source* yang tersedia secara bebas dengan menggunakan *Structured Query Language* (SQL). MySQL merupakan salah satu *Relationan Database Management System* terpopuler saat ini.

SQL adalah bahasa yang paling populer untuk menambahkan, mengakses dan mengelola konten dalam database. Hal ini karena pemrosesannya yang cepat, dengan kehandalan, kemudahan dan fleksibilitas penggunaan terbukti.

Istilah yang digunakan dalam MySQL dan database secara general adalah (Peicevic, 2016):

1. *Database* adalah sebuah *container* untuk koleksi data MySQL.

2. *Table* merupakan *subcontainer* didalam database yang menyimpan data aslinya.
3. *Column* adalah nama dari *field.*
4. *Row* merupakan catatan tunggal dalam *table.*
5. *Primary Key* adalah kunci yang secara unik mengidentifikasi setiap record dalam tabel database.
6. *Foreign Key* adalah kunci dalam satu tabel yang menunjukkan ke *primary key* yang berada di tabel lain.
7. *Index* adalah tabel pencarian khusus yang digunakan mesin pencari database untuk mempercepat pengambilan data.

2.10. State of the Art

Dalam Jurnal yang berjudul "Aplikasi Restoran Terintegrasi Menggunakan Java, MySQL, dan Tampilan *Augmented Reality*" (Primadita Banu Anggra, 2015), Aplikasi restoran yang menggunakan *augmented reality (AR)* mempunyai manfaat sebagai aplikasi yang dapat memudahkan dalam pelayanan suatu restoran dalam pemesanan dan proses pembelian juga akan lebih menarik jika aplikasi tersebut menampilkan makanan yang dijual kedalam bentuk 3D *augmented reality.* Aplikasi yang menggunakan AR memudahkan pengimplementasian dalam mendukung kinerja pada tiap layanan restoran pada saat restoran mendapatkan pengunjung yang banyak. Dengan menggunakan sistem yang terhubung pelanggan hanya perlu memesan makanan menggunakan perangkat *mobile* yang sudah disediakan dengan tampilan *augmented reality* untuk melihat menu makanan.

Dalam jurnal berjudul "Pengembangan Aplikasi Buku Menu Rumah Makan Bebek Tepi Sawah Berbasis Augmented Reality" (Putri, Ni Putu Yunita Susandi, I Gede Mahendra Darmawiguna, dan Gede Saindra Santyadiputra, 2015), aplikasi menu makanan berbasis Augmented Reality memiliki manfaat yaitu pengunjung akan mengetahui informasi yang detail mengenai masing – masing menu restoran baik dari bahan, proses pembuatan, hingga penyajiannya, sehingga pelayan restoran tidak perlu menjelaskan informasi detail mengenai masing – masing menu. Selain itu, manfaat lain

yang dirasakan yaitu pihak restoran akan memiliki dokumentasi dari masing – masing menu yang bisa digunakan sebagai arsip oleh restoran.

Dalam jurnal berjudul "Perancangan Kartu Nama Dengan *Augmented Reality* Sebagai Portofolio Digital" (Demi Adidrana, A. Lumenta, Brave A. Sugiarso, dan Virginia Tulenan, 2013) aplikasi berbasis *Augmented Reality* memiliki manfaat yaitu pengaplikasian *augmented reality* memiliki daya tarik tersendiri sehingga banyak yang memakai *augmented reality* baik dari perorangan hingga tingkat perusahaan seperti perusahaan Lego atau IKEA yang mengaplikasikan *augmented reality* untuk mempromosikan produk mereka.