BAB II

LANDASAN TEORI

2.1. Tinjauan Jurnal

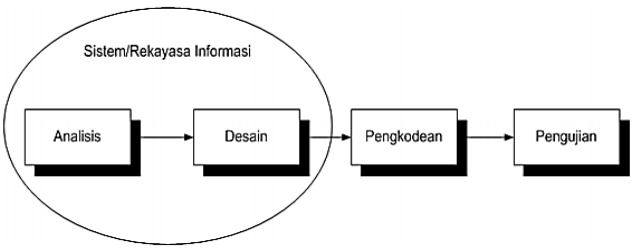
Menurut Hariyanto dalam Sobri (2014:236) “media pembelajaran adalah komponen integral dari sistem pembelajaran serta segala sesuatu yang dapat dipergunakan untuk merangsang pikiran, perhatian, perasaan serta kemampuan sehingga dampaknya akan mendorong semangat belajar”.

Menurut Meltari dan Huzaeni (2016:27) menjelaskan bahwa “Algoritma *fisher-yates shuffle* merupakan suatu algoritma untuk memberikan teknik pengacakan pada soal sehingga soal yang keluar akan berbeda dan dapat dihasilkan tanpa pengulangan atau duplikasi”.

2.2. Konsep Dasar Program

1. *Waterfall*

   Menurut Rosa dan Shalahuddin (2014:28) Model SDLC air terjun (*waterfall*) sering juga disebut sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian dan tahap pendukung (*support*).



Sumber : Rosa dan Shalahuddin (2014:29)

Gambar II.1. Ilustrasi Model *Waterfall*

a. Analisa kebutuhan perangkat lunak

Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh user. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu didokumentasikan.

b. Desain

Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumetasikan.

c. Pembuatan kode program

Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

d. Pengujian

Pengujian fokus pada perangkat lunak secara segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

e. Pendukung (*support*) atau pemeliharaan (*maintenance*)

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke user. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

Dari kenyataan yang terjadi sangat jarang model air terjun dapat dilakukan sesuai alurnya karena sebab berikut :

a. Perubahan spesifikasi perangkat lunak terjadi ditengah alur pengembangan
b. Sangat sulit bagi pelanggan untuk mendefinisikan semua spesifikasi diawal alur pengembangan. Pelanggan sering kali butuh contoh (*prototype*) untuk menjabarkan spesifikasi. Kebutuhan sistem lebih lanjut.
c. Pelanggan tidak mungkin bersabar mengakomodasi perubahan yang diperlukan diakhir alur pengembangan.

Dengan berbagai kelemahan yang dimiliki model air terjun tapi model ini telah menjadi dasar model-model yang lain dalam melakukan perbaikan model pengembangan perangkat lunak.

Model air terjun sangat cocok digunakan kebutuhan pelanggan sudah saat dipahami dan kemungkinan terjadinya perubahan kebutuhan selama

pengembangan perangkat lunak kecil. Hal positif dari model air terjun adalah struktur tahap pengembangan sistem jelas, dokumentasi dihasilkan disetiap tahap pengembangan, dan sebuah tahap dijalankan setelah tahap sebelumnya selesai dijalankan (tidak ada tumpang tindih pelaksanaan tahap).

2. Android

Menurut Safaat (2012:1) "Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka". Android memiliki banyak *versi* mulai dari pertama hingga yang digunakan sampai saat ini, yaitu sebagai berikut :

Tabel II.1.
Tingkatan Versi Sistem Operasi Android

Nama	Versi	Peluncuran
Cupcake	1.5	27 April 2009
Donut	1.6	15 September 2009
Eclair	2.0 – 2.1	26 Oktober 2009
Froyo	2.2 – 2.2.3	20 Mei 2010
Gingerbread	2.3 – 2.3.7	06 Desember 2010
Honeycomb	3.0–3.2.6	22 Pebruari 2011
Ice Cream Sandwich	4.0 – 4.0.4	18 Oktober 2011
Jelly Bean	4.1 – 4.3.1	09 Juli 2012
KitKat	4.4 – 4.4.4	31 Oktobe 2013
Lollipop	5.0 – 5.1.1	12 November 2014
Marshmallow	6.0 – 6.0.1	05 Oktober 2015
Nougat	7.0	Agustus / September 2016

Sumber : *https://haiwiki.info*

3. Java

Menurut Rosa dan Shalahuddin (2014:103) Java dikembangkan oleh perusahaan Sun Microsystem. Java menurut definisi dari Sun Microsystem adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Java 2 adalah generasi kedua dari java *platform.*

Java berdiri di atas sebuah mesin *interpreter* yang diberi nama *Java Virtual Machine* (JVM). JVM inilah yang akan membaca *bytecode* dalam *file* .class dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu bahasa Java disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM.

4. Android Studio

Android Studio adalah sebuah IDE yang bisa digunakan untuk pengembangan aplikasi Android, dan dikembangkan oleh Google. Android Studio merupakan pengembangkan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu *IntelliJ IDEA*.

5. Android SDK (*Software Development Kit*)

Menurut Safaat (2012:5) menyatakan bahwa "Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java".

6. AVD (*Android Virtual Device*)

Menurut Safaat (2012:19) menyebutkan bahwa "AVD (*Android Virtual Device*) merupakan *emulator* untuk menjalankan program aplikasi android". AVD ini nantinya akan digunakan oleh penulis untuk melakukan test dan menjalankan aplikasi android yang penulis buat".

7. JDK (*Java Development Kit*)

JDK adalah singkatan dari *Java Development Kit*. JDK merupakan perangkat lunak yang digunakan untuk melakukan kompilasi dari kode - kode

Java yang dibuat oleh pengembang aplikasi lalu menerjemahkannya ke dalam *bytecode* untuk dijalankan oleh JRE. JDK wajib diinstall terlebih dahulu sebelum pengembang membuaxt suatu program / aplikasi.

2.3. Metode Algoritma

Sebelum membahas mengenai metode algoritma yang digunakan penulis, berikut ini ada penjelasan dari metode dan algoritma :

1. Metode

Metode (*method*), secara harfiah berarti cara. Selain itu metode atau metodik berasal dari bahasa Greeka, *metha*, (melalui atau melewati), dan *hodos* (jalan atau cara), jadi metode bisa berarti jalan atau cara yang harus di lalui untuk mencapai tujuan tertentu

2. Algoritma

Menurut Suarga (2012:1) "Teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas tetapi tersusun secara logis dan sistematis".

Algoritma yang digunakan penulis adalah algoritma *Fisher-Yates shuffle*

> Menurut Meltari dan Huzaeni (2016:28) *Fisher-Yates shuffle* (diambil dari nama Ronald Fisher dan Frank Yates) atau juga dikenal dengan nama Knuth shuffle (diambil dari nama Donald Knuth), adalah sebuah algoritma untuk menghasilkan suatu permutasi acak dari suatu himpunan terhingga, dengan kata lain untuk mengacak suatu himpunan tersebut.

Jika diimplementasikan dengan benar, maka hasil dari algoritma ini tidak akan berat sebelah, sehingga setiap permutasi memiliki kemungkinan yang sama. Metode dasar yang digunakan untuk menghasilkan suatu permutasi acak untuk angka 1 sampai *N* adalah sebagai berikut:

1. Tuliskan angka dari 1 sampai *N*.
2. Pilih sebuah angka acak *K* diantara 1 sampai dengan jumlah angka yang belum dicoret.
3. Dihitung dari bawah, coret angka *K* yang belum dicoret, dan tuliskan angka tersebut di lain tempat.
4. Ulangi langkah 2 dan langkah 3 sampai semuat angka sudah tercoret.
5. Urutan angka yang dituliskan pada langkah 3 adalah permutasi acak dari angka awal.

Tabel II.2.
Contoh Pengerjaan Algoritma *Fisher-Yates Shuffle*

Range	*Roll*	*Scratch*	*Result*
1 2 3 4 5 6 7 8			
1-8	6	1 2 3 4 5 8 7	**6**
1-7	2	1 7 3 4 5 8	**2 6**
1-6	6	1 7 3 4 5	**8 2 6**
1-5	1	5 7 3 4	**1 8 2 6**
1-4	3	5 7 4	**3 1 8 2 6**
1-3	3	5 7	**4 3 1 8 2 6**
1-2	1	7	**5 4 3 1 8 2 6**

Sumber : Meltari dan Huzaeni (2016:28)

2.4. Pengujian Sistem

Di dalam pengujian sistem terdapat beberapa metode diantaranya adalah :

1. Metode Pengujian *Black Box*

Menurut Rosa dan Shalahuddin (2014:275) menjelaskan bahwa "*Black Box Testing* yaitu menguji perangkat lunak dari segi fungsional tanpa menguji desain dan kode program".

2. Metode Pengujian *White Box*

Menurut Rosa dan Shalahuddin (2014:276) menjelaskan bahwa "*White Box Testing* yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan dan keluaran yang sesuai dengan spesifikasi kebutuhan".

2.5. Peralatan Pendukung

Membuat suatu aplikasi android perlu berbagai macam pendukung yang dapat membantu terbentuknya suatu aplikasi.

1. OOP (*Object-Oriented Programming*)

Menurut Raharjo, dkk (2012 : 35) ciri dari *Object Oriented Programming* adalah abstraksi (*abstraction*), pembungkusan (*encapsulation*), pewarisan (*inheritance*), dan polimorfisme atau kebanyakan rupaan (*polymorphism*).

Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian berorientasi objek, metode berorientasi objek banyak dipilih karena metode lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasikan hasil dari satu tahapan pengembangan ketahap berikutnya, keuntungan menggunakan metode berorientasi objek adalah sebagai berikut.

a. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut.

b. Kecepatan pengembangan

Karena sistem yang dibangun dengan baik dan benar pada saat analisa dan perancangan akan menyebabkan berkurangannya kesalahan pada saat pengkodean.

c. Kemudahan pemeliharaan

Karena dengan model-model, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-rubah.

d. Adanya konsistensi

Karena sifat pewaris dan penggunaan notasi yang sama pada saat analisa, perancangan maupun pengkodean.

e. Meningkatkan kualitas perangkat lunak

Karena pendekatanpengembangan lebih dekat dengan dunia nyata dan adanya kosisten pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

Pada saat ini banyak bahasa pemrograman berorientasi objek. Banyak yang berfikir bahwa pemrograman berorientasi objek identik dengan bahasa java. Memang bahasa java merupakan bahasa yang paling kosisten dalam mengimplementasikan paradigma pemrograman yang mendukung pemrograman berorientasi objek tidak hanya bahasa java, berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metologi berorientasi objek .

a. Kelas (*class*)

Menurut Rosa dan Shalahuddin (2014:86) menyatakan bahwa "Kelas adalah kumpulan objek-objek dengan karakteristik yang sama".

Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakannya dan kelas tersebut, sebuah kelas akan mempunyai sifat (*atribute*), kelakuan (*method*), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lainnya, dimana *atribute* dan kelas semula dapat diwariskan kedalam yang baru.

b. Objek

Menurut Rosa dan Shalahuddin (2014:87) menyatakan bahwa "Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, atau hal lain-lainnya yang bersifat abstrak".

Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai oprasi (kelakuan) yang dapat diterapkan data dapat berpengaruh pada status objek

c. Metode (*method*)

Menurut Rosa dan Shalahuddin (2014:88) menyatakan bahwa "Operasi atau metode atau method pada sebuah kelas hampir sama dengan fungsi atau prosedur".

Pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau oprasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan

fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

d. Atribut (*attribute*)

Menurut Rosa dan Shalahuddin (2014:88) menyatakan bahwa "*Attribute* dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas".

Attribute dapat berupa nilai atau element-element data yang dimiliki oleh dalam kelas objek. *Attribute* dipunyai secara individu oleh sebuah objek, misalnya berat, jenis, nama , dan sebagainya.

e. Abstraksi (*abstraction*)

Menurut Rosa dan Shalahuddin (2014:89) menyatakan bahwa "Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan".

f. Enkapsulasi

Menurut Rosa dan Shalahuddin (2014:89) menyatakan bahwa "Pembungkus atribute data dan layanan (oprasi-oprasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek hingga objek lain tidak mengetahui cara kerjanya".

g. Pewaris (*inheritance*)

Menurut Rosa dan Shalahuddin (2014:89) menyatakan bahwa "Mekasime yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lainnya sebagai bagian dari dirinya".

h. Polimorfism

Menurut Rosa dan Shalahuddin (2014:90) menyatakan bahwa "Kemampuan suatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program".

2. UML (*Unified Modeling Language*)

Menurut Rosa dan Shalahuddin (2014:133) "UML (*Unified Modeling Language*) adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek". Macam-macam diagram yaitu :

a. *Use Case Diagram*

Use case atau diagram *use case* merupakan permodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

b. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1) Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
2) Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
3) Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
4) Rancangan menu yang ditampilkan pada perangkat lunak

c. *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.

1) Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2) Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada

gunanya lagi sebuah perancangan karena apa yang dirancang dan hasil jadinya tidak sesuai.

d. *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. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case.*

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

e. *Deployment Diagram*

Diagram *deployment* atau *deployment diagram* menunjukan 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.

3. Adobe Photoshop

Adobe Photoshop atau yang lebih dikenal dengan Photoshop merupakan perangkat lunak yang diproduksi oleh Adobe Systems serta berguna untuk mengolah/mengedit sebuah gambar ataupun foto. Perangkat lunak ini dapat dijalankan untuk operasi *system* Windows, MacOS X dan MacOS.

4. Audacity

Menurut Supraja dan Luthfi (2012:6) "Audacity adalah sebuah *software Editor* untuk *Audio* dan dapat digunakan sebagai *recorder*. Audacity adalah salah satu Program yang bersifat *freeware* dan mempunyai *fitur* yang bagus sebagai kategori *Audio Editor"*.