

## **BAB II**

### **LANDASAN TEORI**

#### **2.1. Tinjauan Pustaka**

Merancang sebuah sistem informasi berbasis *web* sebagai media atau sarana informasi penerimaan peserta didik baru khususnya di SMAN 14 Garut guna mempercepat proses pekerjaan, selain itu dengan adanya sistem ini diharapkan dapat memberikan informasi kepada masyarakat syarat-syarat masuk ke SMAN 14 Garut, maka dengan adanya PPDB *online* informasi dapat diterima oleh masyarakat dengan cepat. Mengakomodasi kebutuhan dalam mempermudah dan mempercepat kinerja petugas pendaftaran peserta didik baru dalam mengelola data pendaftar, dengan demikian waktu antri pendaftaran pada sistem ini dapat diminimalkan (Rahayu, dkk, 2012).

Menurut Palilingan, dkk (2014), sistem registrasi calon siswa baru berbasis *mobile android* di SMA N 9 Manado dapat meningkatkan efektivitas dan fleksibilitas sistem sekolah. Aplikasi registrasi calon siswa baru berbasis *mobile android* dapat diakses dimana saja dan kapan saja dapat membantu calon siswa baru untuk melakukan registrasi dan juga membantu guru atau staf administrasi untuk mengelola data calon siswa baru.

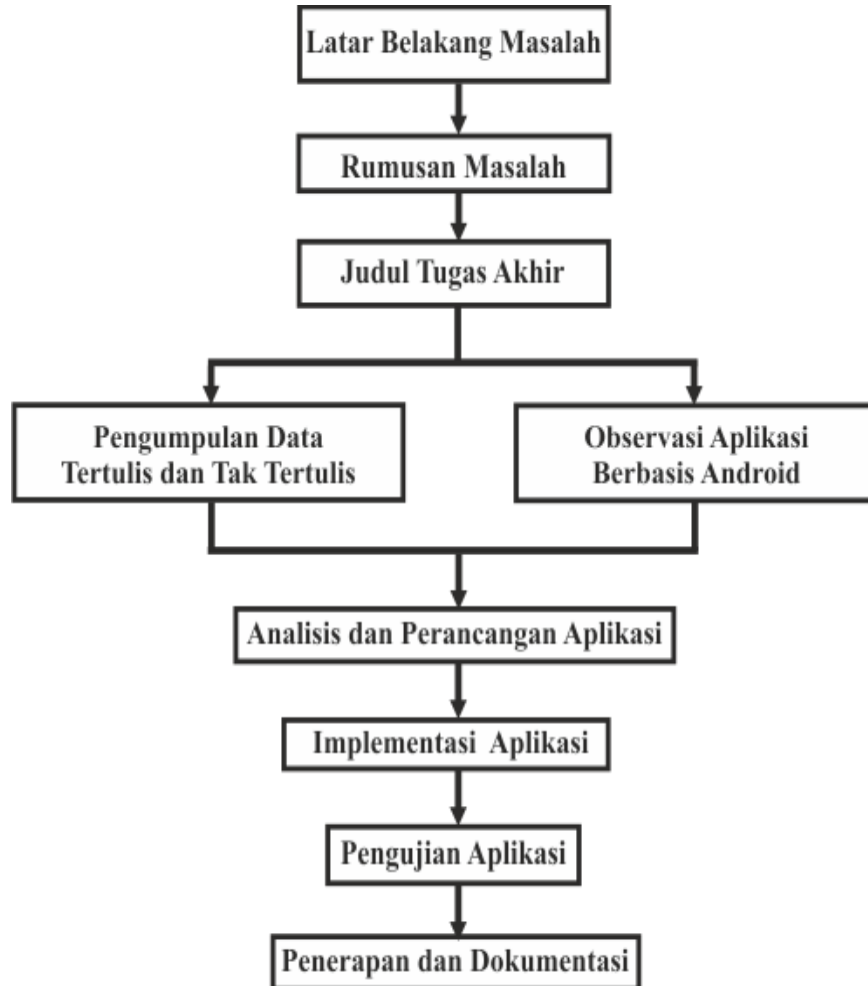
Proses Penerimaan Siswa Baru di SMA Citra Islami yang masih dilakukan secara manual sering menimbulkan terjadinya berbagai masalah dari penginputan data yang lambat, berkas pendaftaran yang tidak tersusun rapih dan antrian pendaftaran yang panjang. Peneliti memberikan solusi terhadap masalah tersebut, dimana peneliti menganalisa sistem yang berjalan dan permasalahan yang ada pada proses penerimaan siswa baru yang digambarkan dalam diagram UML. Kemudian mengembangkan sebuah aplikasi berbasis *mobile web* dengan PHP, *JQuery-mobile*, HTML 5 dan MySQL sebagai solusi dari permasalahan tersebut. Hasil dari penelitian ini adalah tersedianya aplikasi penerimaan siswa baru pada SMA Citra Islami, dengan aplikasi ini calon siswa baru dapat mengisi form pendaftaran dan melihat informasi berkaitan dengan pendaftaran secara *mobile*. Pihak sekolah mendapatkan kemudahan dalam mengelola data baik dari

pengarsipan, melihat rekapitulasi data pendaftaran hingga menginformasikan jadwal serta hasil dari test calon siswa (Santoso, dkk, 2013). Hasil perbandingan penelitian dari kajian pustaka yang ada disajikan pada Tabel 2.1.

Tabel 2.1 Perbandingan Penelitian

<b>Judul</b>	<b>Tujuan</b>	<b>Metode</b>	<b>Hasil</b>
Registrasi Calon Siswa Baru Berbasis Mobile Android di Sekolah Menengah Atas Negeri 9 Manado	Membuat aplikasi registrasi calon siswa baru berbasis <i>mobile android</i>	<i>Inception, Elaboration, Construction, Transition</i>	Aplikasi <i>smartphone</i> berbasis android dapat digunakan untuk melakukan registrasi calon siswa baru
Perancangan Sistem Informasi Pendaftaran Peserta Didik Baru Berbasis Web Studi Kasus di SMA Negeri 14 Garut	Merancang Sistem Informasi Pendaftaran Peserta Didik Baru Berbasis Web	<i>Preliminary Investigation, System Analysis, System Design, System Implementation</i>	Rancangan basis data dan proses sistem pendaftaran calon peserta didik baru
Aplikasi Penerimaan Siswa Baru Berbasis <i>Mobile Web</i> Studi Kasus : SMA Citra Islami	Membuat aplikasi penerimaan siswa baru berbasis <i>mobile web</i>	Perencanaan, Perancangan Aplikasi, Pembuatan Aplikasi, Uji coba aplikasi.	Sistem informasi penerimaan siswa baru berbasis <i>mobile web</i> yang <i>cross-platform handphone</i>
Pembuatan Aplikasi Penerimaan Siswa Baru di Bimbingan Belajar Gama Jogja Cabang Surakarta Berbasis Android	Membuat aplikasi penerimaan siswa baru berbasis android	Perencanaan, Perancangan Aplikasi, Pembuatan Aplikasi, Uji coba aplikasi.	Aplikasi <i>smartphone</i> berbasis android dapat digunakan untuk melakukan penerimaan siswa baru

## 2.2. Kerangka Pemikiran



Gambar 2.1 Diagram Kerangka Pemikiran

Keterangan diagram Kerangka Pemikiran ( Gambar 2.1 ) :

### 1. Latar Belakang Masalah

Memudahkan siswa yang ingin mendaftar bimbingan belajar tapi tidak mempunyai waktu untuk mendaftar langsung ke kantor dan juga memudahkan staf dalam melakukan pendataan siswa baru yang mendaftar.

### 2. Rumusan Masalah

Membuat aplikasi penerimaan siswa baru yang lebih efektif dan efisien dalam proses pendaftaran siswa baru di Lembaga Bimbingan Belajar Gama Jogja Cabang Surakarta.

### 3. Judul Tugas Akhir

Menentukan judul tugas akhir yang sesuai dengan latar belakang masalah dan rumusan masalah yang ada.

### 4. Pengumpulan Data Tertulis dan Tak Tertulis

Mengumpulkan semua data yang dibutuhkan, baik melalui *interview* dengan petugas, observasi atau studi literatur di Lembaga Bimbingan Belajar Gama Jogja.

### 5. Observasi Aplikasi Berbasis Android

Mengamati beberapa aplikasi android yang sudah ada, baik dari karya ilmiah, buku atau internet yang dapat dijadikan referensi untuk membangun aplikasi.

### 6. Analisis dan Perancangan Aplikasi

Menganalisa dan merancang aplikasi yang akan dibangun seperti apa, bagaimana desainnya, dan apa saja isinya, sehingga aplikasi ini dapat membantu memecahkan permasalahan yang ada pada Lembaga Bimbingan Belajar Gama Jogja.

### 7. Implementasi Aplikasi

Membuat *database* dari data-data yang telah didapatkan sesuai dengan kebutuhan aplikasi. Membuat aplikasi dengan dasar *database* dan rancangan aplikasi yang telah selesai dibuat sesuai dengan analisis yang ada.

### 8. Pengujian Aplikasi

Pengujian aplikasi ini dilakukan bertujuan untuk mengetahui ada atau tidaknya kesalahan atau kekurangan pada aplikasi yang telah dibuat.

### 9. Penerapan dan Dokumentasi

Pada tahap akhir, dimana aplikasi telah siap digunakan setelah melewati tahap pengujian dan membuat dokumentasi dari keseluruhan kegiatan penyusunan tugas akhir.

## **2.3. Teori-Teori Pendukung**

### **2.3.1. Android**

Menurut Kasman (2015), android merupakan sebuah sistem operasi telepon seluler dan komputer tablet layar sentuh (*touch screen*) yang berbasis Linux. Platform android terdiri dari sistem operasi berbasis Linux, sebuah GUI (*Graphic User Interface*), sebuah *Web Browser* dan aplikasi *End-User* yang dapat di *download* dan juga para pengembang bisa dengan leluasa berkarya serta menciptakan aplikasi yang terbaik dan terbuka untuk digunakan oleh berbagai macam perangkat. Berikut ini adalah versi android dari yang pertama sampai yang terbaru :

- a. Android Beta (dirilis pertama November 2007)
- b. Android 1.0 Astro
- c. Android 1.1 Bender
- d. Android 1.5 Cupcake
- e. Android 1.6 Donut
- f. Android 2.0/2.1 Eclair
- g. Android 2.2 Froyo
- h. Android 2.3 Gingerbread
- i. Android 3.0/3.1 Honeycomb
- j. Android 4.0 ICS
- k. Android 4.1/4.3 Jelly Bean
- l. Android 4.4 Kit Kat
- m. Android 5.0 Lollipop
- n. Android 6.0 Marshmallow
- o. Android 7.0 Nougat (versi terbaru dirilis Juni 2016)

### **2.3.2. Android Software Development Kit (SDK)**

Android SDK (*Software Development Kit*) adalah tools API (*Applications Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java (Kasman, 2015).

### 2.3.3. *Hypertext Preprocessor (PHP)*

PHP merupakan Bahasa pemrograman berbentuk skrip yang ditempatkan di sisi *server*, sehingga php disebut juga sebagai bahasa *Server Side Scripting*, artinya bahwa dalam menjalankan php selalu membutuhkan *web server*, dan untuk melihatnya menggunakan *web browser* (Purbadian, 2015).

PHP dirancang untuk membentuk tampilan *web* yang dinamis, artinya php dapat membentuk suatu tampilan berdasarkan permintaan *user*, misalnya dapat mengakses *database* dan menampilkannya pada halaman *web*. PHP menyatu dengan kode html, namun beda kondisinya. Maksudnya adalah kode yang dibuat menggunakan html dirancang untuk membangun suatu pondasi awal dari kerangka *layout web*, sedangkan PHP digunakan untuk memproses data dari sisi *server*, sehingga terciptalah suatu tampilan *web* yang dinamis.

### 2.3.4. *Basis Data (database)*

*Database* (basis data) merupakan kumpulan data yang saling berhubungan satu dengan lainnya yang tersimpan di perangkat keras komputer dan diperlukan suatu perangkat lunak untuk memanipulasi basis data tersebut (Junindar, 2008).

Sistem manajemen basis data (*database mangement system/DBMS*) adalah perangkat lunak yang digunakan untuk mengendalikan data, termasuk penyimpanan data, pengambilan data, keamanan data, dan integritas data. Fungsi utama DBMS adalah untuk menyediakan lingkungan yang nyaman dan efisien untuk digunakan dalam pengambilan dan penyimpanan informasi di basis data. (Junindar, 2008)

Terdapat dua bentuk arsitektur sistem basis data, yaitu sistem terpusat dan sistem *client-server*. Sistem basis data terpusat adalah sistem basis data yang dijalankan pada sistem komputer tunggal dan tidak berinteraksi dengan sistem pada komputer lain. Pengguna terkoneksi ke komputer pusat melalui terminal. Sistem basis data *client-server* adalah sistem basis data yang memisahkan program pengguna dengan program basis data di sistem yang berbeda. Pengguna terkoneksi ke pusat data yang disebut *server* sistem melalui suatu program pengguna (*user interface*) yang terdapat pada komputer. Sistem tempat program pengguna berada disebut *client system*.

### 2.3.5. JQuery Mobile

JQuery *Mobile* adalah sebuah *framework* yang dibangun dan dikembangkan dari JQuery, yang menyediakan berbagai elemen antar muka dan fitur untuk digunakan dalam aplikasi berbasis *mobile*. Dengan JQuery *mobile* pengembang dapat membuat aplikasi *web* yang *multi-platform*, tidak bergantung pada piranti *mobile* tertentu (Siregar, dkk, 2010).

### 2.3.6. Apps Geysers

*Apps Geysers* adalah layanan berbasis *online (web)* yang memungkinkan para pengguna membuat aplikasi android mereka dari konten sebuah *website*, melalui *apps geysers* kita dapat dengan mudah mendistribusikan konten dari *website*, *blog* maupun file-file tertentu yang ingin kita bagikan melalui aplikasi android (Kasman, 2015).

### 2.3.7. UML (Unified Modeling Language)

*Unified Modeling Language* adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO) (Fowler, 2004).

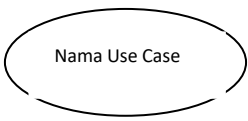
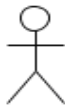




UML memungkinkan *developer* melakukan permodelan secara visual, yaitu penekanan pada penggambaran, bukan didominasi oleh narasi. Permodelan visual membantu untuk menangkap struktur dan kelaikan dari objek, mempermudah penggambaran interaksi antara elemen dalam sistem, dan mempertahankan konsistensi antara desain dan implementasi dalam pemrograman.

#### 2.3.7.1 Use Case Diagram

*Use Case Diagram* adalah model fungsional sebuah sistem yang menggunakan *actor* dan *use case*. *Use Case* adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use Case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan (Fowler, 2004). Cara terbaik menganggap sebuah *use case diagram* adalah sebagai tabel grafis dari isi untuk rangkaian *use case*. Ini mirip dengan *context diagram* yang digunakan dalam

metode terstruktur, karena menampilkan batasan sistem dan interaksi dengan dunia luar. *Use case diagram* menampilkan aktor, *use case* dan hubungan antara mereka. Simbol-simbol yang digunakan pada *use case diagram* disajikan pada Tabel 2.2.

**Tabel 2.2** Simbol – Simbol *Use Case Diagram*

No	Gambar	Nama	Keterangan
1.		<i>Use Case</i>	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
2.		<i>Actor</i>	Orang proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri.
3.		<i>System Boundary</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
4.		<i>Ekstensi / Extend</i>	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> .
5.		<i>Include</i>	Penyisipan perilaku tambahan kedalam basis <i>use case</i> yang secara eksplisit menggambarkan penyisipan.
6.		<i>Association</i>	Hubungan antara dua buah <i>use case</i> .


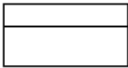

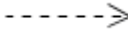

### 2.3.7.2 Class Diagram

*Class Diagram* mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat di antara mereka. *Class diagram* juga menunjukkan properti dan operasi sebuah *class* dan batasan-batasan yang



terdapat dalam hubungan-hubungan objek tersebut. UML menggunakan istilah fitur sebagai istilah umum yang meliputi properti dan operasi sebuah *class* (Fowler, 2004). Simbol-simbol yang digunakan pada *class diagram* disajikan pada Tabel 2.3.

**Tabel 2.3** Simbol – Simbol *Class Diagram*

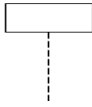
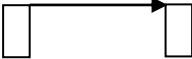

No	Gambar	Nama	Keterangan
1.		<i>Generalization</i>	Hubungan dimana objek anak ( <i>descendent</i> ) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk ( <i>ancestor</i> ).
2.		<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
3.		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
4.		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri ( <i>independent</i> ) akan memengaruhi elemen yang bergantung padanya elemen yang tidak mandiri
5.		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya

### 2.3.7.3 Sequence Diagram

*Sequence Diagram* mendeskripsikan bagaimana entitas dalam sistem berinteraksi, termasuk pesan yang digunakan saat interaksi. Diagram ini juga

menunjukkan serangkaian pesan yang diperlukan oleh objek-objek yang melakukan suatu tugas atau aksi tertentu. Sebuah *sequence diagram*, secara khusus menjabarkan behavior sebuah skenario tunggal. Diagram tersebut menunjukkan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam *use case* (Fowler, 2004). Salah satu hal yang menarik tentang *sequence diagram* adalah hampir tidak perlu menjelaskan notasinya. Simbol-simbol yang digunakan pada *sequence diagram* disajikan pada Tabel 2.4.

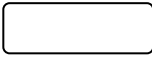



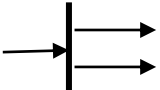
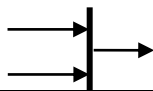
**Tabel 2.4** Simbol – Simbol *Sequence Diagram*

No	Gambar	Nama	Keterangan
1.		<i>Life Line</i>	Objek <i>entity</i> , antarmuka yang saling berinteraksi.
2.		<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
3.		<i>Actor</i>	Pengguna di luar sistem.

#### 2.3.7.4 Activity Diagram

*Activity Diagram* adalah teknik untuk menggambarkan logika prosedural, proses bisnis dan jalur kerja. Dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung *behavior* paralel (Fowler, 2004). *Activity diagram* memungkinkan siapapun yang melakukan proses untuk memilih urutan dalam melakukannya. Dengan kata lain, diagram hanya menyebutkan aturan-aturan rangkaian dasar yang harus kita ikuti. Hal ini penting untuk pemodelan bisnis karena proses-proses sering muncul secara paralel. Ini juga berguna pada algoritma yang bersamaan, di mana urutan-urutan independen dapat melakukan hal-hal secara paralel. Simbol-simbol yang digunakan pada *activity diagram* disajikan pada Tabel 2.5.

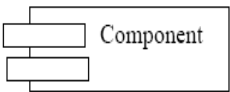
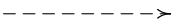
**Tabel 2.5** Simbol-Simbol *Activity Diagram*

No	Gambar	Nama	Keterangan
1.		<i>Activity State</i>	Aktivitas yang mewakili pelaksanaan dalam pernyataan dalam prosedur atau pelaksanaan kegiatan dalam alur kerja.
2.		<i>Branch/Merge</i>	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
3.		<i>Initial State</i>	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
4.		<i>Final State</i>	Status akhir yang dilakukan sistem.
5.		<i>Fork</i>	Percabangan yang menunjukkan aliran pada <i>activity diagram</i> .
6.		<i>Join</i>	Penggabungan yang menjadi arah aliran pada <i>activity diagram</i> .

### 2.3.7.5 Component Diagram

*Component Diagram* menggambarkan struktur fisik kode dari komponen. Komponen dapat berupa *source code*, komponen biner, atau *executable component*. Simbol-simbol yang digunakan pada *component diagram* disajikan pada Tabel 2.6.

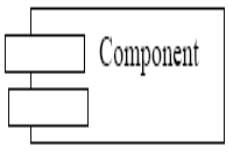
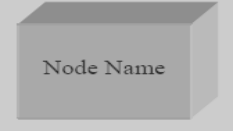

**Tabel 2.6** Simbol – Simbol *Component Diagram*

No	Nama	Gambar	Keterangan
1.	Component		Sebuah <i>component</i> melambangkan sebuah entitas software dalam sebuah sistem.
2.	Dependency		Sebuah <i>dependency</i> digunakan untuk menotasikan relasi antara dua <i>component</i> .

### 2.3.7.6 Deployment Diagram

*Deployment Diagram* menunjukkan susunan fisik sebuah sistem, menunjukkan bagian perangkat lunak mana yang berjalan pada perangkat keras mana (Fowler, 2004). Simbol-simbol yang digunakan pada *use deployment diagram* disajikan pada Tabel 2.7.

**Tabel 2.7** Simbol – Simbol *Deployment Diagram*

No	Nama	Gambar	Keterangan
1.		<i>Component</i>	Pada <i>deployment diagram</i> , <i>component - component</i> yang ada diletakkan didalam <i>node</i> untuk memastikan keberadaan posisi mereka.
2.		<i>Node</i>	<i>Node</i> menggambarkan bagian-bagian <i>hardware</i> dalam sebuah sistem. Notasi untuk <i>node</i> digambarkan sebagai sebuah kubus 3 dimensi.
3.		<i>Association</i>	Sebuah <i>association</i> digambarkan sebagai sebuah garis yang menghubungkan dua <i>node</i> yang mengindikasikan jalur komunikasi antara elemen-elemen <i>hardware</i> .

### 2.3.8. Metode Pengujian *Five View*.

Disadur dari *Software Testing and QA Theory and Practice (Chapter 17: Software Quality University of Waterloo)* (Kshirasagar dan Tripathy, 2008) Pengujian *Five View* adalah pengujian yang sifatnya deskriptif dimana *software* yang diuji dinilai melalui lima sudut pandang berbeda, yaitu :

#### 2.3.8.1 *User View* (Pandangan Pengguna)

Pada pandangan ini kualitas perangkat lunak dinilai berdasarkan kepuasan pengguna, apakah sesuai dengan kebutuhan pengguna atau tidak. Perangkat lunak dinilai bagus bila memiliki banyak pengguna.

### **2.3.8.2 Manufacturing View (Pandangan Manufaktur)**

Pandangan ini berkaitan dengan faktor dalam industri, apakah produk memenuhi persyaratan atau tidak setiap penyimpangan dari persyaratan yang dinilai mengurangi kualitas produk. Konsep proses memainkan peran kunci produk yang dibuat harus orisinal sehingga biaya berkurang, misalnya biaya pembangunan dan biaya pemeliharaan. Kualitas dapat secara bertahap ditingkatkan dengan memperbaiki proses.

Proses pembangunan perangkat lunak memainkan peran kunci, produk diproses sejak awal, sehingga biaya pengembangan dan perawatan dapat dikurangi. Keselarasan dengan kebutuhan akan mengarahkan pada keseragaman produk. Apabila prosesnya berkualitas maka produknya juga berkualitas.

### **2.3.8.3 Transcendental View (Pandangan Transendental)**

Kualitas menurut pandangan ini adalah suatu yang dapat dikenali melalui pengalaman tapi tidak dapat selalu digambarkan. Objek atau software yang bagus itu menonjol dan dapat dengan mudah dikenali.

Pandangan ini menilai kualitas berdasarkan pengalaman para ahli. Kualitas adalah sesuatu yang dapat dikenali tetapi tidak dapat didefinisikan, kualitas perangkat lunak bersifat subyektif dan tidak dapat dihitung dengan angka.

### **2.3.8.4 Value-based View (Pandangan Nilai)**

Pandangan ini merupakan gabungan antara keunggulan dan harga. Perangkat lunak diukur kualitasnya berdasarkan keunggulan, dan diukur nilainya berdasarkan harganya. Berapa banyak *customer* yang akan membayar pada level kualitas tertentu. Kualitas tidak dapat diukur jika produk tidak membuat nilai ekonomis. Pandangan *value-based* membuat *trade-off* antara harga dan kualitas.

### **2.3.8.5 Product View (Pandangan Produk)**

Jika sebuah produk diproduksi dengan sifat internal (misalnya bahan dan tindakan) yang baik, maka produk akan memiliki sifat eksternal atau output yang baik dan dapat dieksplorasi hubungan antara sifat internal dan kualitas eksternal.