BAB II
KAJIAN PUSTAKA

2.1 Teori Dasar

2.1.1 *Augmented Reality*

Menurut (Pamoedji, Maryuni, & Sanjaya, 2017: 2), *augmented reality* atau dalam bahasa Indonesia diterjemahkan menjadi realitas tambahan adalah sebuah teknik yang menggabungkan benda maya dua dimensi maupun tiga dimensi ke dalam sebuah teknik yang menggabungkan benda maya dua dimensi maupun tiga dimensi ke dalam sebuah lingkup nyata tiga dimensi lalu memproyeksikan benda-benda maya tersebut dalam waktu nyata.

Augmented reality (AR) adalah sebuah istilah untuk lingkungan yang menggabungkan dunia nyata dan dunia virtual yang dibuat oleh komputer sehingga batas antara keduanya menjadi sangat tipis dan lebih dekat kepada lingkungan nyata *(real)*. *Augmented reality* (AR) memungkinkan pengguna untuk berinteraksi dengan objek *virtual* dan benda-benda nyata dalam lingkungan nyata secara *real time*. Beberapa aplikasi dikembangkan menggunakan teknologi *augmented reality* seperti aplikasi medis, manufaktur, visualisasi, perencanaan jalur, hiburan, dan militer yang telah dieksplorasi (Husniah, Saputro, & Cahyono, 2016: 33-34).

Augmented Reality AR adalah variasi dari *Virtual Environment* (VE), atau *Virtual Reality* (VR) karena lebih sering disebut. Teknologi *virtual reality* benar-benar membenamkan pengguna di dalam lingkungan sintetis dan saat terbenam, pengguna tidak dapat melihat dunia nyata disekitarnya. Sebaliknya, *augmented reality* mengambil informasi digital atau komputer yang dihasilkan, apakah itu gambar, audio, video, dan sentuhan atau sensasi *haptic* dan *overlaying* mereka di lingkungan *real-time*, *augmented reality* secara teknis dapat digunakan untuk meningkatkan kelima indera, Namun penggunaan saat ini yang paling umum adalah *visual*. Tidak seperti *virtual reality*, *augmented reality* memungkinkan pengguna untuk melihat dunia nyata, dengan benda-benda maya yang *superiniposed* pada atau digabungkan dengan dunia nyata (Kipper & Rampolla, 2013: 1).

2.1.1.1 Sejarah *Augmented Reality*

Menurut (Kipper & Rampolla, 2013: 7- 11) pengembangan teknologi *augmented reality* di mulai pada tahun 1962. Morton Heilig, seorang sinematografer menciptakan sebuah simulator yang disebutnya sensorama dengan visual, getaran, dan bau. Kemudian perkembangan berlanjut tahun 1975, Myron Kruger menemukan *Videoplace* yang memungkinkan pengguna, dapat berinteraksi dengan objek *virtual* untuk pertama kalinya.

Pada tahun 1992 *augmented reality* dikembangkan untuk perbaikan pesawat *Boeing* dan LB Rosenberg mengembangkan salah satu fungsi sistem yang disebut

Virtual Fixtures, yang digunakan di Angkatan Udara Amerika Serikat Armstrong Labs dan menunjukan manfaat bagi manusia.

Pada tahun 2000, untuk pertama kalinya *augmented reality* dikembangkan menjadi media hiburan berupa *Mobile Game Augmented Reality* yang ditujukan di *International Symposium on Wearable Computers*.

Perkembangan *augmented reality* berlanjut dengan diperkenalkannya NyARToolkit dan FLARToolkit. NyARToolkit dikembangkan dengan banyak bahasa seperti Java, C++, C#. FLARToolkit merupakan teknologi *augmented reality* yang dapat dipasang di sebuah *website*.

2.1.1.2 Komponen *Augmented Reality*

Menurut (Kipper & Rampolla, 2013: 5- 7) *Augmented reality* dibangun dengan beberapa kompenen. Komponen-komponen ini yang nantinya akan membawa dunia *Virtual* menuju dunia nyata. *Augmented reality* tidak akan dapat diterapkan tanpa adanya kompenen pembantu ini. Berikut adalah komponen-komponen dasar yang membangun *augmented reality*:

a. *Input Device*

Input device untuk *augmented reality* dapat berupa kamera atau *webcam* yang nantinya akan menerima *input* dari dunia nyata dan biasanya inputan berupa *marker* yang nantinya akan *discan* menggunakan Input Device berupa kamera atau *webcam*.

b. *Output Device*

Output Device bertugas menampilakan hasil pembacaan *input* dan membawa dunia *virtual* ke dunia nyata. *Output Device* biasanya berupa monitor *handpone*, monitor komputer, atau kacamata *Augmented reality* atau yang lebih dikenal dengan *Head mounted Display*.

c. *Tracker*

Tracker pada *augmented reality* biasanya berupa *QR Code* atau *marker* yang berfungsi untuk melacak agar *augmented reality* yang dihasilkan berjalan secara *real time.*

d. Komputer

Tidak diragukan lagi jika komputer adalah kompenen penting augmented reality. Komputer menjadi alat yang natinya akan membuat program augmented reality dan juga yang akan mejalankannya. Namun dalam perkembangan saat ini *augmented reality* tidak hanya dijalankan di komputer saja namun bisa pula dijalakan di *smartphone* yang berkerja sama seperti komputer namun dengan skala yang kecil.

2.1.1.3 Metode *Augmented Reality*

Berdasarkan penelitian (Sembiring & Brahmana, 2016: 23), tedapat dua metode yang dikembangkan pada AR yaitu:

1. Marker *Augmented Reality* (*Marker based Tracking***)**

Metode ini melakukan pendeteksian objek pada *marker* yang biasanya diilustrasikan sebagai hitam dan putih. Komputer akan mengenali posisi dan orientasi *marker* dan menciptakan dunia maya dalam bentuk 3D berupa titik (0,0,0) dan 3 sumbu (X,Y,Z).

2. *Markeless Augmented Reality*

Dengan metode ini, pengguna tidak perlu menggunakan sebuah *marker* untuk menampilkan elemen-elemen digital, tetapi dengan *tool* yang disediakan *qualcomm* untuk pengembangan AR berbasis *mobile device*, mempermudah pengembangan untuk membuat aplikasi yang *markless*.

2.1.2 *Marker*

Maker termasuk dalam salah satu bagian yang tidak dapat dipisahkan dari teknologi *augmented reality*. *Marker* sangat berperan penting untuk *tracking* dan menampilakan hasil *output* teknologi *augmented reality*.

Berdasarkan penelitian (Sembiring & Brahmana, 2016: 23) tedapat dua metode pendeteksian *marker* pada teknologi *augmented reality* yaitu:

1. *Single Marker*

Single Marker dalam mendeteksi gambar yang dijadikan sebagai media marker dan hanya satu objek saja yang keluar. Metode *Single marker* adalah metode kamera melakukan *tracking* objek yang di tangkap hanya satu

maksudnya, waktu kamera *smartphone* melakukan *scan*, satu *marker* akan mengeluarkan objek 3 dimensi.

2. *Multi Marker*

Multi marker yaitu metode yang memungkinkan untuk mendeteksi banyak objek yang dapat keluar dalam satu pendeteksian *marker*. *Multi Marker* adalah sebuah metode perkembangan dari *singlemarker*, dimana kamera men-*Tracking* objek yang ditangkap lebih dari satu. Dalam implementasinya dapat di lakukan seperti penabelan kompenen serta *corner detection* sebagai pengenalan sudut dari bentuk *marker*.

2.1.3 *Android*

2.1.3.1 Pengertian *Android*

Menurut (Masruri, 2015: 1) *Android* merupakan sistem operasi *open source* yang dimana semua orang bisa mengembangkannya, hal itulah yang membuat perkembangan aplikasi *Android* semakin cepat dan bertumbuh kembang.

Android merupakan sistem operasi yang sangat populer sangat banyak digunakan saat ini. Menurut (Masruri, 2015: 2) berdasarkan riset, volume penjualan ponsel *Android* pada saat ini telah menggusur keberadaan *smartphone blackberry*, hal itu

dikarenakan harga ponsel *Android* tidak terlalu tinggi akan tetapi memberikan fitur yang sangat canggih.

android yang juga merupakan sistem operasi *open source* memungkinkan banyak pengembang aplikasi berbasis *android* berlomba-lomba berinovasi menghasilkan aplikasi yang berguna menunjang kebutuhan sehari-hari. Ini menjadi salah satu alasan banyak pengembang aplikasi tertentu memilih agar nantinya aplikasi yang dihasilkan dapat dioperasikan melalui sistem operasi *android*.

2.1.3.2 Sejarah *Android*

Menurut (Masruri, 2015: 3 - 4) awalnya Google Inc membeli *Android Inc* yang merupakan pendatang baru yang membuat peranti lunak untuk *smartphone*. Kemudian untuk mengembangkan *android*, dibentuklah *open handset aliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi. Termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nividia.

Pada perilisan perdana *android*, 5 November 2007, *android* bersama *open handset aliance* menyatakan mendukung pengembalian *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode *android* di bawah lisensi *apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi *android*. Yang pertama mendapat dukungan penuh dari Google atau Google *Mail Service* (GMS) dan yang kedua adalah benar-benar bebas distribusinya tanpa dukungan langsung dari Google atau yang dikenal dengan *Open Handset Distribution* (OHD)

Sekitar September 2007, Google mengenalkan Nexus One, salah satu jenis *smartphone* yang menggunakan *andoird* sebagai sistem operasinya Telepon seluler ini di produksi oleh HTC Corporation dan tersedia di pasar pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja *Android ARM Holdings, Atheros Communications* yang diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc.

Seiring pembentukan *Open Handset Alliance*, OHA mengumumkan produk perdana mereka, *android*, perangkat *mobile* yang merupakan modifikasi kernel Linux 2.6 sejak *android* dirilis telah dilakukan pembaharuan berupa perbaikan *bug* dan penambahan fitur baru.

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini *android* menjadi pesaing utama dari *Apple* pada *Tablet PC*. Pesatnya pertumbuhan *android* selain faktor yang disebutkan di atas adalah karena *android* itu sendiri adalah *platform* yang sangat lengkap baik sistem operasinya, aplikasi dan *tool* pengembangan, market aplikasi *android* serta dukungan yang sangat tinggi dari komunitas *opensource* di dunia. Sehingga *Android* terus berkembang pesat baik dari segi teknologi, maupun dari jumlah *device* yang ada di dunia.

2.1.3.3 Versi *Android*

Android telah dirilis dengan beberapa versi sampai saat ini. *Android* telah banyak mengalami perubahan dan penambahan fitur-fitur lainnya sehingga versi *Android*

terus-terusan mengalami perubahan. Berikut Menurut (Masruri, 2015: 5 - 20) versi *android* yang telah dirilis:

1. *Android* Versi *Beta*
2. *Android* Versi 1.0
3. *Android* Versi 1.1
4. *Android* Versi 1.5
5. *Android* Versi 1.6
6. *Android* Versi 2.0 dan 2.1
7. *Android* Versi 2.2
8. *Android* Versi 2.3
9. *Android* Versi 3.0 dan *Android* Versi 3.1
10. *Android* Versi 4.0
11. *Android* Versi 4.1, 4.2, 4.3
12. *Android* Versi 4.4
13. *Android* Versi 5.0

2.1.4 Bahasa C#

Menurut (Nugroho, 2009: 5) bahasa pemograman C# dirancang oleh Microsoft Corp sebagai bahasa pemograman yang berdaya-guna, aman serta mudah digunakan. Sebagai bagian dari *paltform* .NET, bahasa pemograman C# dirancang untuk bekerja

seangat baik di atas *framework* .NET, yang mampu digunakan untuk menulis perangakat lunak handal demi layanan yang cepat. Bahasa pemograman C# juga dapat digunakan untuk mengembangkan aplikasi-aplikasi sarana bergerak (*mobile application*), aplikasi berbasis web (*web-bassed aplications*), serta aplikasi berskala besar (*enterprise*).

2.1.5 Katalog

Menurut (Windyaningrum, Nugroho, & Utomo, 2012: 120) katalog berasal dari bahasa latin "*Catalogus*" yang berarti daftar barang atau benda yang disusun untuk tujuan tertentu. Definisi katalog secara umum adalah kumpulan dari beberapa informasi yang saling berkaitan mengenai sebuah produk baik barang ataupun jasa yang disajikan dalam sebuah media. Biasanya banyak perusahaan yang memakai media katalog untuk melakukan promosi sekaligus pendistribusian informasi bagi perusahaan ataupun untuk para member dan konsumen.

Katalog salah satu media bagi perusahaan atau penjualan yang penting saat akan menawarkan produk mereka. Katalog mencakup informasi mengenai *detail* produk untuk memikat para konsumen. Tidak diragukan lagi jika katalog merupakan media yang sampai saat ini masih sangat penting bagi prusahaan dalam penjualan produk.

2.2 *Software* Pendukung

2.2.1 UML

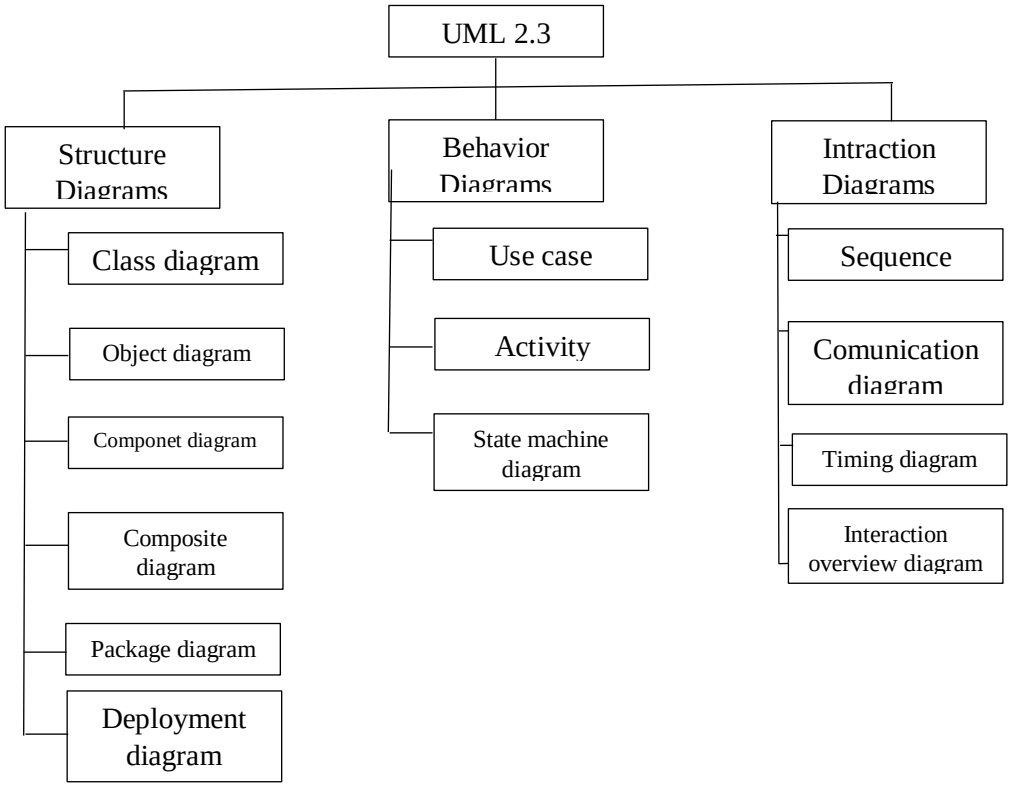
Perancangan sebuah perangkat lunak diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak. Bahasa ini dapat menggambarkan model perancangan perangkat lunak sehingga ide perancangan sebuah perangkat lunak dapat dimengerti oleh banyak orang.

Menurut (A.S & Shalahuddin, 2015: 133 - 137), UML (*Unified Modeling Language*) adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefenisikan *requirement*, membuat analisis analisis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek.

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

2.2.1.1 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar di bawah (A.S & Shalahuddin, 2015: 140).



Gambar 2.1 Diagram UML

2.2.1.2 *Use Case Diagram*

Menurut (A.S & Shalahuddin, 2015: 155 - 161), *Use Case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

Ada dua hal utama pada *use case* dalam pendefinisian apa yang disebut aktor dan *use case*:

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

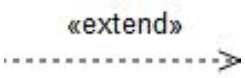
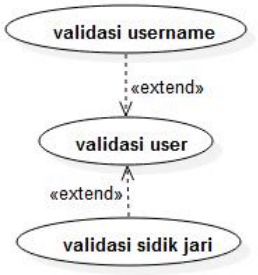
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukas pesan antara unit atau aktor.

Simbol-simbol yang ada pada diagram *use case*:

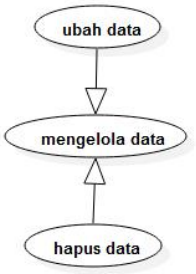
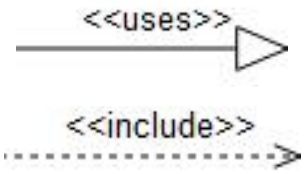
Tabel 2.1 Simbol-simbol diagram *use case*

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antara satu atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal fase nama *use case*.
Aktor / ***actor*** 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; bisanya dinyatakan meggunakan kata benda di awal fase nama aktor.

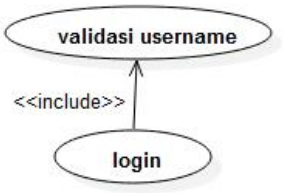
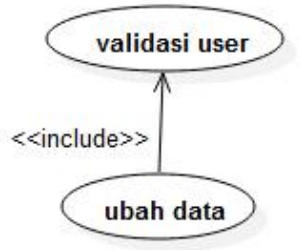
Tabel 2.1 Lanjutan

Asosiasi / *association*	Komunikasi antara aktor dengan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi / *extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemprogram berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi extend-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya.

Tabel 2.1 Lanjutan

Generalisasi / *generalization* 	Hubungan genarilasis dan spesialisasi (umum–khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:  arah panah mengarah pada *use case* yang menjadi generalisasinya. (umum)
Menggunakan / *include* / *uses* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. Ada dua sudut pandang yang cukup besar mengenai include di *use case*: a. *Include* berarti *use case* yang ditambahkan akan selau dipanggil saat *use case* tambahan dijalankan, misal pada kasus berikut:

Tabel 2.1 Lanjutan

	 b. *Include* berarti *use case* yang ditambahkan akan selau melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  Kedua interpretasi di atas dapat dianut saat satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

Sumber: (A.S & Shalahuddin, 2015)

2.2.1.3 *Activity Diagram*

Menurut (A.S & Shalahuddin, 2015: 161-163) 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 adalah 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 sebagai berikut:

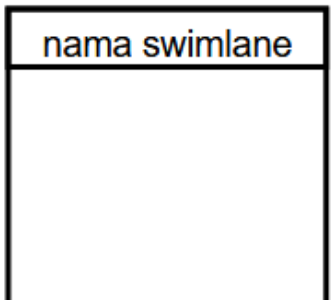
a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sistem atau *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2. 2 Simbol diagram aktivitas

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan / *decision* 	Asosisasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.

Tabel 2. 2 Lanjutan

Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
Swimlane  **Atau** 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Sumber: (A.S & Shalahuddin, 2015)

2.2.1.4 *Class Diagram*

Menurut (A.S & Shalahuddin, 2015: 141-147), diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisan kelas-kelas yang akan dibuat

untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Susunan struktur kelas yang baik pada digram kelas sebaiknya memiliki jenis-jenis kelas, yaitu:

a. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

b. Kelas yang menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

c. Kelas yang diambil dari pendefinisian *use case* (*controller*)

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

d. Kelas yang diambil dari pendefinisian data (*model*)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut *mutivalue* pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap ada di dalam perancangan kelas.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohension* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi atau metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah.

Berikut ini beberapa pertimbangan dalam membuat kelas:

Tabel 2. 3 Pertimbangan dalam membuat kelas

Pertimbangan	Keterangan

Sebuah kelas yang hanya berisi sebuah atribut	Kelas seperti ini kurang efisien sehingga perlu dipikirkan kembali perancangan yang dilakukan, karena secara fisik kelas dengan satu atribut adalah sebagai berikut:
	`Class Pustaka{`
	`  judul : string`
	`}`
	Sebuah berkas atau file hanya berisi seperti di atas tentu saja tidak efisien.
	Cara memperbaiki rancangan adalah dengan menggabungkan satu atribut tersebut ke kelas lain yang memiliki keterkaitan lebih dekat.

Tabel 2. 3 Lanjutan

<table>
<tr>
<td>Sebuah kelas yang hanya berisi atribut</td>
<td>Kelas seperti ini kurang efisien sehingga perlu dipikirkan kembali perancangan yang dilakukan, karena secara fisik kelas dengan hanya berisi atribut saja adalah sebagai berikut:

<pre>Class Pustaka{
    id : string
    judul : string
    penerbit : string
    jumlah : int
    tahun : int
}</pre>
Sebuah berkas atau file hanya berisi seperti diatas tentu saja tidak efisien.
Cara memperbaiki rancangan adalah dengan menggabungkan atribut tersebut ke kelas lain yang memiliki keterkaitan lebih dekat.</td>
</tr>
<tr>
<td>Sebuah kelas yang hanya berisi sebuah metode</td>
<td>Kelas seperti ini kurang efisien sehingga perlu dipikirkan kembali perancangan yang dilakukan, karena secara fisik kelas dengan hanya berisi sebuah metode saja adalah sebagai berikut:

<pre>Class Pustaka{
    id : string
    judul : string
    penerbit : string
    jumlah : int
    tahun : int

    procedure queryMelihatPustaka(){
        query = “SELECT ….”
        execute(query)
    }
}</pre></td>
</tr>
</table>

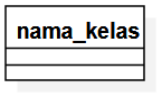
Tabel 2. 3 Lanjutan

	Sebuah berkas atau *file* hanya berisi seperti diatas tentu saja tidak efisien. Cara memperbaiki rancangan adalah dengan menggabungkan atribut dan metode tersebut ke kelas lain yang memiliki keterkaitan lebih dekat.

Sumber: (A.S & Shalahuddin, 2015)

Berikut ini adalah simbol-simbol yang terdapat pada diagram kelas atau *class diagram:*

Tabel 2. 4 Simbol diagram kelas

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem.
Antarmuka / *interface* 	Sama dengan konsep interface dalam pemprograman berorientasi objek.
Asosisasi / *association* 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / *directed association* 	**Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *muliplicity*.**

Tabel 2. 4 Lanjutan

Generalisasi 	**Relasi antarkelas dengan makna generalisasi-generalisasi (umum khusus).**
Kebergantungan / *dependency* 	Relasi antarkelas dengan makna kebergantungan antarkelas.
Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua bagian (*whole-part*).

Sumber: (A.S & Shalahuddin, 2015)

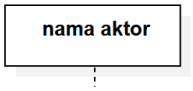
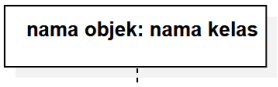
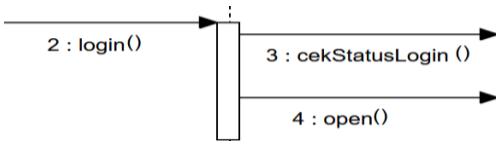
2.2.1.5 *Sequance Diagram*

Menurut (A.S & Shalahuddin, 2015: 165-167), diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh sebab itu, untuk menggambar 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*.

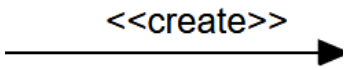
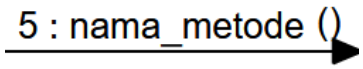
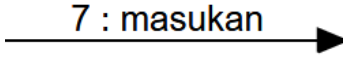
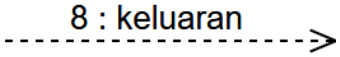
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram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

Di bawah ini adalah beberapa simbol yang ada pada diagram sekuen:

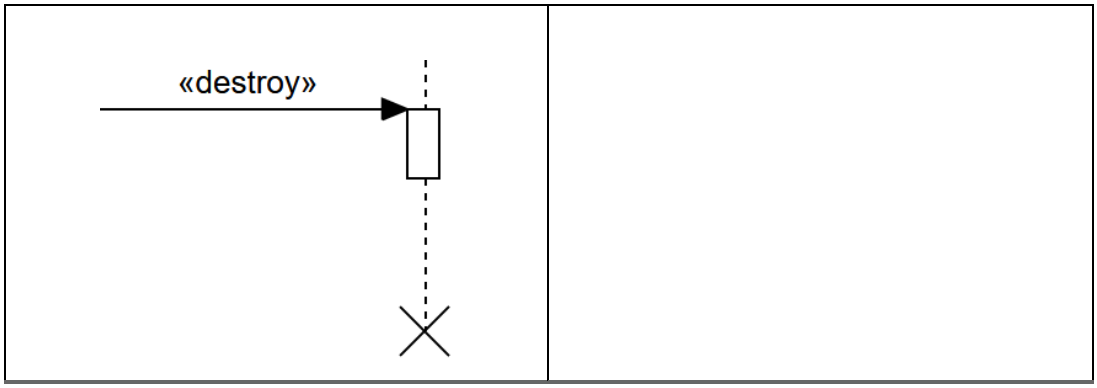
Tabel 2. 5 Simbol-simbol diagram sekuan

Simbol	Deskripsi
Aktor  Atau  Tampa waktu aktif	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek.
Objek  	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadan aktif dan berinteraksi, semua yang terhuung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya,  maka cekStatusLogin() dan open() dilakukan di dalam metode login(). Aktor tidak memiliki waktu aktif.

Tabel 2. 5 Lanjutan

Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri, arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang kirimi.
Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
Pesan tipe destroy	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*.

Tabel 2. 5 Lanjutan



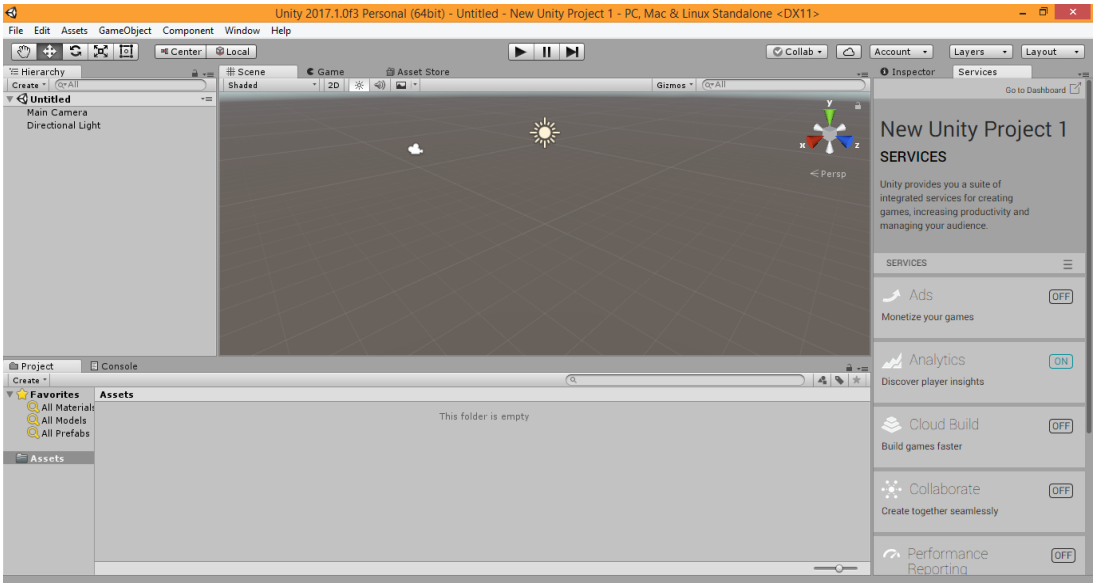
Sumber: (A.S & Shalahuddin, 2015)

Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

Semua metode di dalam kelas harus ada di dalam kolaborasi atau sekuen, jika tidak ada berarti perancangan metode di dalam kelas itu kurang baik. Hal ini dikarenakan ada metode yang tidak dapat dipertanggungjawabkan kegunaannya.

2.2.2 *Unity*

Unity 3D atau yang lebih sering disebut sebagai *Unity* saja, adalah sebuah *software* pemograman yang digunakan untuk membuat berbagai macam aplikasi. Mayoritas penggunaan *Unity* adalah untuk pembuatan aplikasi *Game*. Tetapi dengan menggunakan *Unity*, Anda dapat membuat berbagai macam aplikasi seperti presentasi, *website*, bahkan dapat digunakan untuk membuat *augmented reality* (Pamoedji et al., 2017: 15)



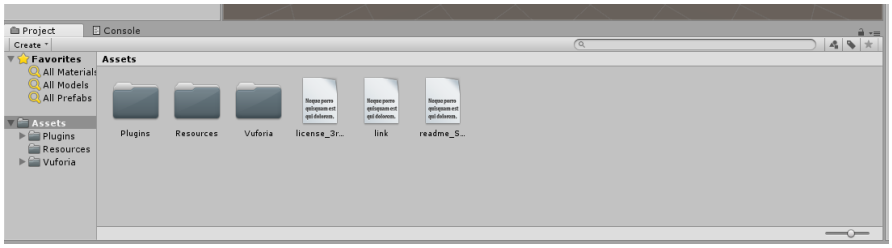
Gambar 2.2 *Unity*

2.2.3 *Vuforia*

Vuforia termasuk salah satu *plugin* dari *software* yang bernama *Unity 3D*. *Vuforia* merupakan *Augmented Reality Software Development Kit* (SDK) yang nantinya akan membantu pembuatan *augmented reality* untuk perangkat *mobile*. SDK *Vuforia* sangat membantu proses pembuatan *augmented reality* untuk perangkat *mobile* karena sangat mendukung dua *platform mobile* besar seperti iOS dan *Android*.

Pendeteksian pada *marker* yang di simpan di dalam *smartphone android* menggunakan algoritma *natural feature tracking and rating* dari algoritma dasar *fast corner detection* yang telah dikembangkan oleh pihak *Vuforia*, marker akan di deteksi kontras beda antar piksel, lebih kontras *marker* akan lebih baik nilai pendeteksiannya,

dengan memberi tanda pada pojok piksel dan setelah itu akan di ketahui kualitas *marker* dengan memberikan *rating* pada *marker* tersebut (Rifa'i, Listyorini, & Latubessy, 2014: 207).



Gambar 2.3 Plug In Vuforia

2.2.4 *Google Sketchup*

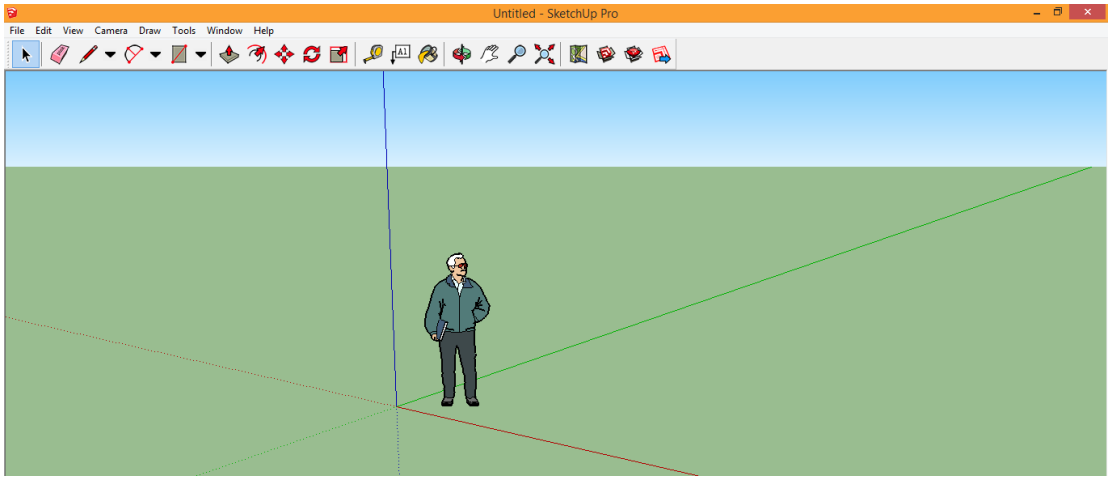
Google Sketchup atau *Sketchup* termasuk ke dalam sebuah aplikasi CAD yang digunakan untuk mendesain objek tiga dimensi dengan tool yang sederhana dan relative mudah untuk dipelajari.

Menurut (Brixius, 2011: xv) tentang *Google Sketchup* yaitu:

> *Sketchup is both simple and intutive; there is no doubt about that. It usually takes a couple hours to get grips on it and to create your firs 3D models. But when you are using it for professional project-with all the attendant time and quality constraints- you have to get a few things straight first to avoid the traps that user inventably fall into, whether are beginners or experienced operators.*

Penjelasan: *SketchUp* bersifat sederhana dan intuitif; Tidak ada keraguan tentang itu. Biasanya butuh beberapa jam untuk mengatasi hal itu dan menciptakan model 3D pertama Anda. Tetapi saat Anda menggunakannya untuk proyek professional dengan semua petugas kendala waktu dan kualitas Anda harus segera menyelesaikan beberapa

hal hindari jebakan yang pasti pengguna terjerumus, apakah mereka bukan pemula atau operator berpengalaman.

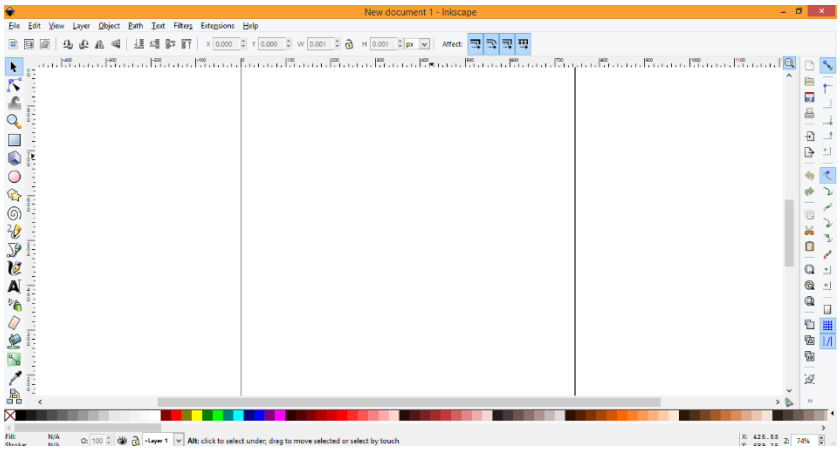


Gambar 2.4 *Google Sketchup*

2.2.5 *Inkscape*

Menurut (Supriyono et al., 2015: 4) Pembuatan desain tampilan menggunakan perangkat lunak grafis yang bersifat *open source* yaitu *Inkscape*. Dalam pembuatan perankat lunak ini dibutuhkan pengolah vektor yang akan dapat menghasilkan desain *marker* yang dibutuhkan. Perangkat lunak ini dibutuhkan untuk mendesain katalog mapun antar muka aplikasi *augmented reality*.

Inkscape nantinya akan berperan sebagai perangkat lunak yang akan medesain tampilan katalog dan elemen paling penting di aplikasi ini yaitu *marker*.



Gambar 2.5 *InkScape*

2.2.6 *Android* SDK

Menurut (Rahadi, 2014: 663) Aplikasi *android* dikembangkan dalam bahasa pemrograman *java* dengan menggunakan *kit* pengembangan perangkat lunak *android* (SDK). SDK ini terdiri dari seperangkat perkakas pengembangan, termasuk *debugger*, perpustakaan perangkat lunak, *emulator handset* yang berbasis QEMU, dokumentasi, kode sampel, dan *tutorial*.

Pegembangan dari *Android Software Development* sendiri sangat tergantung dengan versi *android* yang ada. Ketika google selaku pemilik dan pengembang *android* saat ini ketika mereka merilis versi sistem operasi *android* terbaru, mereka juga akan merilis *Android SDK* terbaru yang akan mensuport kinerja pembuatan perangkat lunak atau aplikasi yang berjalan di sistem operasi *android*.

2.3 Penelitian Terdahulu

Berdasarkan teori yang telah diuraikan diatas maka di dapatkan hasil penelitian terdahulu sebagai berikut:

1. (Husniah et al., 2016), Interaktif *Augmented Reality* untuk Katalog Penjualan Rumah Berbasis *Android*. Pengguna aplikasi melakukan *capture marker* untuk menampilkan model rumah 3D dan mengoperasikan fitur-fitur yang telah disediakan sehingga pengguna dapat mengetahui informasi tentang detail rumah yang ditawarkan. Untuk aplikasi interkatif AR untuk katalog penjualan rumah ada beberapa proses yang dilakukan pada tahap imlementasi yang mengacu pada hasil dari tahap analisis dan desain yang meliputi pembuatan *marker*, pembuatan objek 3D, pembuatan aplikasi interaktir AR untuk katalog penjualan rumah dan pembuatan *interface* meliputi pembuatan *splash screen* aplikasi, pembuatan menu utama aplikasi, dan pembuatan menu aplikasi AR. Pengujian yang dilakukan pada aplikasi interkatif AR untuk katalog penjualan rumah menggunakan teknik pengujian *black-box* untuk mengetahui keberhasilan dari tujuan awal yang ditetapkan. Hasil yang dicapai pada penelitian ini adalah aplikasi interaktif AR untuk katalog penjualan rumah menggunakan dua tipe *marker* yang mewakili masing-masing tipe rumah yaitu *marker* untuk rumah tipe 45 dan marker untuk rumah tipe 36. *Marker* yang digunakan berbentuk katalog rumah, dengan melakukan *capture marker* tersebut menggunakan kamera HP pengguna dapat

menggunakan aplikasi ini untuk melihat spesifikasi rumah, model rumah 3D, denah rumah 3D, mengganti warna cat dinding, pintu dan jendela rumah, memutar objek rumah 3D, dan melihat harga rumah tersebut sesuai dengan tipe rumah. Aplikasi interaktif AR untuk katalog penjualan rumah yang dapat dijalankan pada *smartphone* yang memiliki spesifikasi sistem operasi *Android* minimum 2.3 (*gingerbread*) dengan RAM minimum 512 MB, dan kamera minimum 3.15 MP. Untuk hasil yang dijabarkan pada Tabel 3, aplikasi dijalankan pada *smartphone* Samsung core GT-i8262 dengan spesifikasi RAM 1GB, versi *Android* 4.1.2, memori internal 4GB, dan layar 4.5 inch. Pengujian yang dilakukan menggunakan teknik pengujian *black-box* dengan tujuan untuk mengetahui keberhasilan dari tujuan awal yang ditetapkan dan menguji fungsionalitas dari fitur-fitur yang ada pada aplikasi, khususnya fitur untuk mengganti warna cat dinding, pintu dan jendela.

2. (Sembiring & Brahmana, 2016), Rancang Bangun dan Analisis Aplikasi *Augmented Reality* pada Produk *furniture*. Peneliti terlebih dahulu melakukan studi litratur untuk mendalami materi terkait pengembangan aplikasi AR dari berbagai sumber referensi. Membangun aplikasi AR dengan menerapkan metode Luther-Sutopo dengan 6 kegiatan yaitu *Concept, Design, Matrial Collecting, Assembly, Testing, Distribution*. Aplikasi menggunakan pendekteksian *marker* AR yaitu *sigle marker, multimarker dan 3D Object recognition* dengan parameter pola, bentuk, jarak dan intensitas cahaya. Hasil Implementasi AR adalah produk furniture seperti kursi, jam dinding, sofa, tempat tidur dan lain-lain yang sudah

dikemas dalam bentuk katalog dapat ditampilkan seperti dunia nyata dalam bentuk 3D melalui *smartphone* (*android*) menggunakan *single* dan *multi marker*. Selain menggunakan katalog, media lain yang digunakan adalah berupa miniatur dengan pendekteksinya menggunakan metode *3D Object Recognition*.

3. (Rifa'i et al., 2014), Penerapan Teknologi *Augmented Reality* pada Aplikasi Katalog Rumah Berbasis *Android*. Metode atau tahap-tahap dalam perancangan aplikasi ini menggunakan *Prototype Model* dimana sistem ini nantinya dapat dikembangkan kembali. Tahapan-tahapan *Prototype* yang dimuali dari *Listen to Customer, Build/Revise, Customer Test-Drives Mock-Up.* Aplikasi ini dibangun sebagai alat untuk menampilkan informasi rumah, bentuk rumah dan denah ruangan secara 3 dimensi, dimana bentuk 3D ini akan ditampilkan pada sebuah marker atau gambar rumah yang ada pada katalog rumah yang telah dibuat yang dapat dilihat pada Tabel 1 Dengan dibangunnya aplikasi ini diharapkan dapat meminimalisir pengeluaran sebuah perusahaan pengelolaan perumahan dan menambah media promosi kepada *customer*. Aplikasi ini dibangun dengan menggunakan bahasa pemrograman C# dan dengan menggunakan tools Unity & MonoDevelop, serta *Android* SDK. Untuk pembuatan 3 dimensi rumah digunakan *tools* SketchUp 2013. Selain itu aplikasi ini menggunakan *library Vuforia* sebagai *tools* untuk membuat aplikasi *augmented reality*. Mekanisme proses aplikasi *Augmented Reality* yaitu dimulai dengan disediakannya *marker* gambar rumah pada katalog. Kemudian katalog ditampilkan di depan kamera *smartphone*, dan kamera akan membaca dan aplikasi akan mendeteksi *marker* tersebut dengan

marker yang telah di deteksi sebelumnya yang di simpan di *smartphone*. Pendeteksian pada *marker* yang di simpan di dalam *smartphone Android* menggunakan algoritma *Natural Feature Tracking and Rating* dari algoritma dasar *Fast Corner Detection* yang telah dikembangkan oleh pihak *Vuforia*, *marker* akan di deteksi kontras beda antar piksel, lebih kontras *marker* akan lebih baik nilai pendeteksiannya, dengan memberi tanda pada pojok piksel dan setelah itu akan di ketahui kualitas *marker* dengan memberikan *rating* pada *marker* tersebut. Jika marker tidak cocok dengan *marker* yang di simpan pada *smartphone Android* maka proses akan di ulang terus menerus, dan jika *marker* cocok aplikasi akan merendering objek 3D dan kemudian menapilkannya.

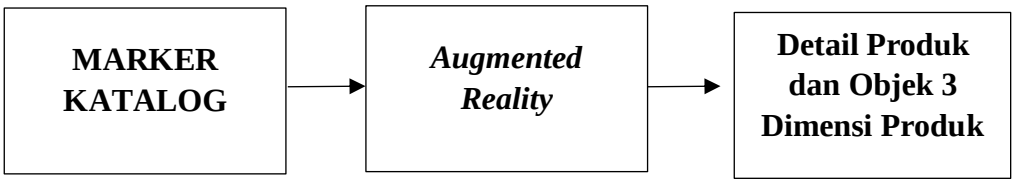
4. (Subagyo, Listyorini, & Susanto, 2015), Pengenalan Rumus Bangun Ruang Matematika Berbasis *Augmented Reality*. Metodologi yang digunakan dalam penulisan tugas akhir ini adalah dengan metode *prototype*. Adapun tahapan-tahapan yang sesuai dengan diagram *prototype* dalam. Membuat *marker* dalam bentuk bangun ruang kemudian memproses mengunakan *Unity* dan *Vuforia* untuk membuat *Interface* dan *marker* yang kemudian ketika *discan* akan menghasilkan bangun ruang 3D tada *marker* yang bergambar bangun ruang.
5. (Prasetya & Nurruzzaman, 2013), Menerapkan Aplikasi *Augmented Reality* pada Obyek-obyek Museum Radya Pustaka. Penelitian ini menggunakan teknologi AR yang digunakan untuk menampilkan obyek-obyek koleksi Museum Radya Pustaka Surakarta. Obyek yang sengaja dikerjakan pada aplikasi ini belum mencakup semua koleksi museum. Hanya sekitar 10 obyek saja yang dibuat visual tiga

dimensi (3D) agar aplikasi dapat berjalan dengan baik. Pembuatan obyek museum didahului dengan studi literatur baik berupa tulisan-tulisan yang membahas tentang obyek dan secara langsung mendatangi museum untuk diambil gambarnya dengan kamera digital. Hasil foto tersebut digunakan untuk memandu dalam proses pembuatan model 3D. Obyek yang dibuat tidak terlalu detail (*low-poly*) agar tidak memberatkan aplikasi AR. Obyek *low-poly* tersebut ditutup dengan tekstur yang didapatkan dari obyek menggunakan kamera digital. Setelah didapatkan berbagai obyek 3D koleksi museum dengan ekstesi *file* .wrml, *file* tersebut selanjutnya dipasangkan ke dalam sistem ARToolKit. Proses aplikasi *Augmented Reality* yaitu dengan mengenalkan *marker* pada tahap awalnya. *Marker* adalah sebuah gambar berpola khusus yang dicetak di atas kertas yang sudah dikenali oleh t*emplates memory ARToolKit*. *Marker* ini berfungsi sebagai pola penanda yang dibaca dan dikenali menggunakan kamera lalu dicocokkan dengan *template ARToolKit*. Setelah itu aplikasi akan menambahkan obyek 3D di atas *marker* yang terlihat hasilnya pada *display* komputer. Obyek akan terlihat sedikit kasar karena menggunakan *lowpoly*. Tampilan obyek 3D pada *marker* AR dengan kondisi seperti jarak, kemiringan *marker* dan pencahayaan yang sesuai, sehingga obyek bisa langsung muncul di atas *marker*. *Marker* yang digunakan untuk menampilkan obyek 3D tersebut berdimensi 4x4 cm.

2.4 Kerangka Pemikiran

Menurut (Sugiyono, 2014: 60) kerangka berfikir merupakan model konseptual tentang bagaimana teori berhubungan dengan berbagai faktor yang telah diidentifikasi sebagai masalah penting. Katalog yang tidak dapat memuat informasi detail produk dari berbagai sisi membutuhkan penerapan teknologi *augmented reality* yang nantinya akan memberikan informasi dengan perwujudan produk secara dimensi-dimensi sehingga dapat memberikan informasi lebih kepada konsumen produk alat-alat rumah tangga.

Berdasarkan tinjuaan pustaka, maka disusunlah kerangka pemikiran yang diilustrasikan seperti gambar berikut:



Gambar 2.6 Kerangka Pemikiran

Gambar atau citra produk yang *diinput* menggunakan *vuforia* akan menghasilkan *marker*. *Marker* ini nantinya yang akan berperan sebagai *input* informasi kepada teknologi *augmented reality*. Ketika kita melakukan *scan* terhadap *marker* katalog maka, perangkat lunak *augmented reality* yang dirancang akan membaca *marker*

kemudian menampilkan objek tiga dimensi di atas *marker* dan memberikan informasi detail produk sebagai *output* dari proses.