BAB II

LANDASAN TEORI

2.1. Konsep Dasar Perancangan

Perkembangan sistem harus diawali dengan melakukan perancangan terlebih dahulu dengan tujuan membuat konstruksi yang baik untuk *project* yang ingin dibangun.

Beberapa peniliti mengemukakan mengenai konsep dasar perancangan yang lebih umum digunakan, yaitu sebagai berikut:

Nafisah (2003), "*Perancangan adalah penggambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi Perancangan sistem dapat dirancang dalam bentuk bagan alir sistem (system flowchart), yang merupakan alat bentuk grafik yang dapat digunakan untuk menunjukan urutan-urutan proses dari sistem*"

Verzello / John Reuter III (1982), "*Tahap setelah analisis dari siklus pengembangan sistem: Pendefinisian dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi : "menggambarkan bagaimana suatu sistem dibentuk*".

George M. Scott (1999), "*Desain sistem menentukan bagaimana suatu sistem akan menyelesaikan apa yang mesti diselesaikan tahap ini menyangkut mengkonfigurasikan dari komponen-komponen perangkat lunak dan perangkat keras dari suatu sistem, sehingga setelah instalasi dari sistem akan benar-benar memuaskan rancang bangun yang telah ditetapkan pada akhir tahap analisis sistem".*

2.2. Konsep Dasar Aplikasi Android

Menurut Arifianto (2011), "*Android merupakan perangkat bergerak pada sistem operasi untuk telepon seluler yang berbasis linux*".

Menurut Susanto (2011), "*Android merupakan OS (Operating System) Mobile yang tumbuh ditengah OS lainnya yang berkembang*

dewasa ini. OS lainnya seperti Windows Mobile, I-Phone OS, Symbian, dan masih banyak lagi. Akan tetapi, OS yang ada ini berjalan dengan memprioritaskan aplikasi inti yang dibangun sendiri tanpa melihat potensi yang cukup besar dari aplikasi pihak ketiga. Oleh karena itu, adanya keterbatasan dari aplikasi pihak ketiga untuk mendapatkan data asli ponsel, berkomunikasi antar proses serta keterbatasan distribusi aplikasi pihak ketiga untuk platform mereka".

2.3. Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah penggambaran awal dari perancangan sistem yang akan dibangun. UML memiliki banyak jenis pemodelan yang ada, tetapi hanya beberapa bagian saja yang akan digunakan dan dibahas pada bab ini.

Menurut Booch (2005), "*UML merupakan suatu bahasa. Suatu bahasa terdiri dari kata-kata, dan memiliki aturan untuk menggabungkan kata-kata tersebut, sehingga tercipta komunikasi. Sebuah permodelan bahasa adalah suatu bahasa di mana kata-kata dan aturannya berfokus pada penggambaran sistem secara konseptual dan fisik. Sebuah permodelan bahasa seperti UML telah menjadi bahasa standar untuk merencanakan suatu aplikasi*".

Hasil dari pemodelan tersebut adalah pengertian dari suatu sistem keseluruhan. Satu model yang digunakan tidak cukup untuk menggambarkan keseluruhan sistem, maka dibutuhkan model lain yang akan berhubungan dan melengkapi satu dengan yang lainnya untuk memberikan pengertian dasar dari sistem yang akan dibangun.

Keuntungan UML ini adalah sebagai berikut :

- Sebagai bahasa pemodelan yang *general-purpose*, difokuskan pada pokok pembuatan rancangan yang dapat dipakai dan digunakan bersama serta menggunakan pengetahuan bersama untuk dapat dibagikan kepada bagian – bagian lainnya.

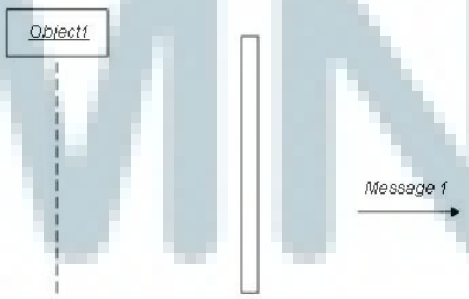
- Sebagai bahasa pemodelan yang mudah diaplikasikan, dapat diaplikasikan untuk bermacam tipe sistem (*software* dan *non-software*), domain dan metode atau proses.
- Sebagai bahasa pemodelan standar industri, bukan merupakan bahasa yang tertutup atau satu-satunya, tapi bersifat terbuka.

2.3.1. Sequences Diagram

Menurut Booch (2005), "*suatu sequence diagram adalah suatu diagram interaksi yang menekankan pada pengaturan waktu dari pesan-pesan. Diagram ini menampilkan sekumpulan peran dan pesan-pesan yang dikirim dan diterima oleh instansi yang memegang peranan tersebut*".

Sequence diagram akan menangkap objek – objek yang ada dan terlibat di dalam scenario berdasarkan urutan – urutan pesan yang ditukar antara sesame objek untuk melaksanakan fungsionalitas sistem.

Dalam *Unified Model Language* (UML), objek dalam sequence diagram akan digambar dengan kotak persegi yang berisi nama objek yang diberi garis bawah. Objek dapat diberi nama dengan tiga cara : nama objek, nama class, atau nama objek dan nama *class* dijadikan satu. Berikut notasi *sequence diagram* seperti terlihat pada gambar dibawah ini :



Gambar 2.1 Notasi Sequence Diagram

2.3.2. Activity Diagram

Activity diagram akan menggambarkan semua alur aktivitas yang dilakukan *user* dalam penggunaan sistem yang sedang dirancang, bagaimana masing – masing alur berawal, bagaimana penggunaan *decision* yang mungkin terjadi, dan bagaimana sistem akan berakhir. *Activity diagram* juga dapat menggambarkan berbagai proses yang bersifat pararel.

Activity diagram merupakan state diagram yang sebagian besar dari state tersebut adalah action dan merupakan transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan proses didalam sistem bagaimana spesifik proses sistemasi aplikasi, tetapi lebih menggambarkan alur proses dan jalur aktivitas secara umum.

2.3.3. Class Diagram

Menurut Booch (2005), "*class diagram menunjukan sekumpulan kelas, antarmuka, dan kerjasama serta hubungannya. Class diagram digunakan untuk memodelkan perancangan statik dari gambaran sistem. Biasanya meliputi permodelan vocabulary dari sistem, permodelan kerjasama, atau permodelan skema*".

Class diagram dapat digunakan di dalam membangun sistem yang dieksekusi melalui teknik *forward* dan *reverse*, selain untuk penggambaran, pemodelan, dan pendokumentasian struktur sistem.

Class Diagram terdiri dari:

a) Nama *Class*.
b) Atribut.
c) Operasi/Method.

Tabel 2.1 Class Diagram

(Wahono R.S, 2003)

Nama Class
Atribut
Operasi/method

Atribut dan Operasi/*method* dapat memiliki tiga sifat berikut:

- *Public*, dapat dipanggil oleh *class* apa saja.
- *Protected*, hanya dapat dipanggil atau diakses oleh *class* yang bersangkutan dan *class* turunannya.
- *Private*, hanya dapat dipanggil oleh dirinya sendiri (tidak dapat diakses dari luar *class* yang bersangkutan).

Hubungan antar *class :*

1. Asosiasi, yaitu hubungan yang bersifat statis didalam *class.* Asosiasi menggambarkan *class* yang memiliki atribut berupa *class* lain atau *class* yang harus mengenal adanya *class* lain. Seperti halnya *parent class* dan *child class*.
2. Agregasi, merupakan hubungan antara satu *object* dengan *object* lainnya di mana *object* satu dengan *object* lainnya sebenarnya terpisah namun disatukan, sehingga tidak terjadi kebergantungan (*Object* lain bisa ada walau *object* penampungnya tidak ada).
3. Pewarisan, yaitu hubungan hirarki antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metode *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*messaging*) yang di-*passing* dari satu *class* kepada *class* lain.

2.4. Desain Sistem

Menurut Hartono (2005:196), "*mengemukakan desain sistem adalah tahap setelah analisis dari siklus pengembangan sistem. Pendefinisian dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun menggambarkan bagaimana suatu sistem dibentuk*".

Desain sistem akan menentukan bagaimana suatu sistem akan melakukan penyelesaian dari apa yang harus diselesaikan, yang menyangkut konfigursi dari komponen – komponen perangkat lunak maupun keras dari suatu sistem agar setelah instalisasi dari sistem akan benar – benar memuaskan dan sesuai dengan apa yang diharapkan seperti rancang bangun yang telah ditetapkan pada akhir tahap analisa sistem.

Tahap desain sistem mempunyai dua maksud atau tujuan utama, yaitu:

1. Untuk memenuhi kebutuhan kepada pemakai sistem.
2. Untuk memberikan gambaran yang jelas dan rancang bangun yang lengkap kepada pemograman komputer dan ahli-ahli teknik lainnya yang terlibat.

Desain sistem akan terbagi atas 2 jenis, yaitu desain sistem secara umum (*general system design*) dan desain sistem secara terperinci (*detailed system*). Tujuan dari desain sistem secara umum adalah untuk memberikan gambaran secara besar terhadap *user* tentang sistem baru. Sedangkan desain secara terperinci bertujuan untuk mengidentifikasikan komponen – komponen yang ada di dalam sistem dan nantinya akan didesain secara terperinci. Desain yang bersifat umum adalah persiapan untuk menggambarkan desain secara terperinci.

2.5. Java Programming

Java Platform Micro Edition atau Java ME, merupakan salah satu platform java yang dirancang untuk aplikasi *mobile* dan sistem tertanam / *offline software*. Level Java ME digunakan untuk merancang perangkat lunak *handphone* dan PDA. Java ME ini lebih dikenal juga dengan nama

Java 2 *Platform, Micro Edition* atau J2ME. Java ME dirancang oleh Sun Microsystem dan sekarang dimiliki oleh perusahaan Oracle Corporation.

Java Platform, Enterprise Edition atau Java EE merupakan standar untuk mengembangkan aplikasi skala besar / *enterprise*. J2EE banyak digunakan sebagai platform untuk pemrograman di sisi server.

Java Platform Standard Edition atau J2SE banyak digunakan sebagai platform untuk pemrograman dalam bahasa Java. Platform ini digunakan untuk deploying sebuah aplikasi. Java SE terdiri dari virtual machine yang digunakan untuk menjalankan program java bersama-sama dengan library atau paket.

Perbedaan dengan J2EE adalah bahwa J2SE menambahkan *library* yang berfungsi untuk mendeploy program java agar berjalan pada aplikasi server.

2.6. Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu atau biasa disebut dengan *Integrated Development Environment* (IDE) yang bertujuan untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Selain merupakan alat pengembang yang berdaya guna, *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya:

- Sistem pembuatan berbasis *Gradle* yang fleksibel
- Emulator yang cepat dan memiliki banyak fitur
- *Instant Run* yang bertujuan untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
- Terdapat template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
- Alat penguji dan kerangka kerja yang ekstensif

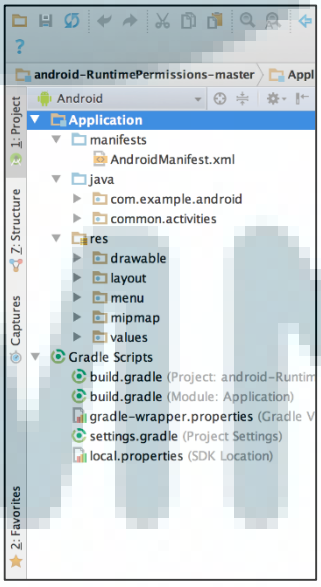
- Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
- Terdapat dukungan C++ dan NDK
- Terdapat dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*

Berikut beberapa dasar fitur dari Android Studio yang akan digunakan untuk pembangunan aplikasi ini, yaitu :

2.6.1. Struktur Proyek

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan *file* kode sumber dan *file* sumber daya. Jenis-jenis modul mencakup:

- Modul aplikasi Android
- Modul library
- Modul *Google App Engine*



Gambar 2.2 File Proyek Di Tampila Android

Secara default, *Android Studio* akan menampilkan *file* proyek Anda dalam tampilan proyek *Android* seperti yang ditunjukkan dalam

gambar 2.2. Tampilan ini diatur menurut modul untuk memberi akses cepat ke *file* sumber kunci proyek Anda.

Semua file versi terlihat di bagian atas di bawah *Gradle Scripts* dan masing-masing modul aplikasi berisi *folder* berikut:

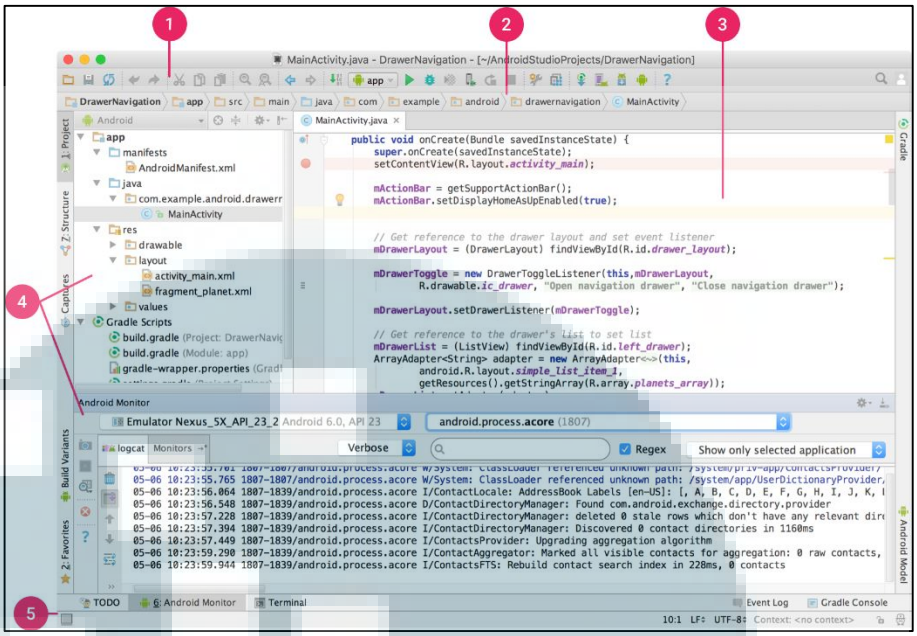
- *manifests*: Berisi *file* AndroidManifest.xml.
- *java*: Berisi *file* kode sumber Java, termasuk kode pengujian JUnit.
- *res*: Berisi semua sumber daya bukan kode, seperti tata letak XML, *string* UI, dan gambar bitmap.

Struktur proyek Android pada *disk* berbeda dari representasi rata ini. Untuk melihat struktur *file* sebenarnya dari proyek ini, pilih *Project* dari menu tarik turun *Project* (dalam gambar 1, struktur ditampilkan sebagai Android).

Anda juga bisa menyesuaikan tampilan *file* proyek untuk berfokus pada aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih tampilan Problems dari tampilan proyek Anda akan menampilkan tautan ke *file* sumber yang berisi kesalahan pengkodean dan sintaks yang dikenal, misalnya tag penutup elemen XML tidak ada dalam *file* tata letak.

2.6.2. Antarmuka Pengguna

Jendela utama Android Studio terdiri dari beberapa bidang logika yang diidentifikasi dalam gambar 2.3.



Gambar 2.3 Jendela Utama Android Studio

1. Bilah alat memungkinkan Anda untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat Android.

2. Bilah navigasi akan membantu Anda menavigasi di antara proyek dan file yang dibuka untuk pengeditan. Di sini tampilan struktur yang terlihat tampak lebih ringkas daripada jendela *Project*.

3. Jendela editor merupakan tempat Anda membuat dan mengubah kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor akan menampilkan Layout Editor.

4. Jendela alat memberi Anda akses ke tugas-tugas spesifik seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi. Anda bisa meluaskan dan juga menciutkannya.

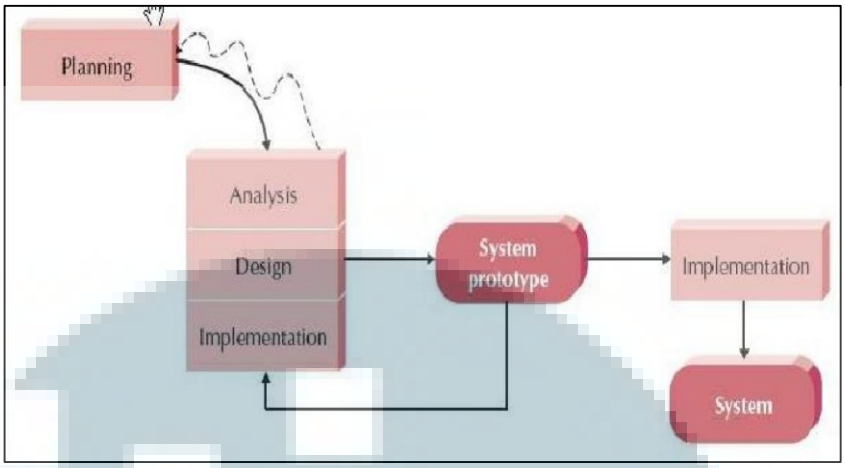
5. Bilah status menampilkan status proyek Anda dan IDE itu sendiri, serta setiap peringatan atau pesan.

Anda bisa menata jendela utama untuk memberi Anda ruang layar yang lebih luas dengan menyembunyikan atau memindahkan bilah alat dan jendela alat. Anda juga bisa menggunakan pintasan keyboard untuk mengakses sebagian besar fitur IDE.

Anda dapat menelusuri seluruh kode sumber, basis data, tindakan, elemen antarmuka pengguna, dan seterusnya setiap saat dengan menekan tombol Shift dua kali, atau mengeklik kaca pembesar di sudut kanan atas dari jendela Android Studio. Ini akan sangat berguna misalnya saat Anda mencoba menemukan tindakan IDE tertentu yang Anda lupakan cara memicunya

2.7. Prototype

Prototype merupakan salah satu metode pengembangan perangat lunak yang sering digunakan. Dengan adanya metode prototyping ini pengembang dan pemakai dapat saling berinteraksi selama proses pembuatan sistem. Masalah yang sering terjadi adalah seorang pemakai hanya mendefinisikan secara umum apa yang dikehendakinya tanpa menyebutkan secara detail output apa saja yang dibutuhkan, pemrosesan, serta data-data apa saja yang dibutuhkan. Sebaliknya disisi pengembang kurang memperhatikan efisiensi algoritma, kemampuan sistem operasi dan interface yang menghubungkan manusia dan komputer. Untuk mengatasi ketidakserasian antara pemakai dan pengembang, maka dibutuhkan kerjasama yang baik antara kedua pihak sehingga pengembang akan mengetahui dengan benar apa yang diinginkan pemakai dan apa yang tidak dibutuhkan oleh pemakai dengan mengesampingkan segi-segi teknis untuk pengembangan sistem. Dengan adanya kerjasama yang baik antara pengembang dan pemakai akan menghasilkan *system* yang sesuai dengan apa yang diharapkan. Kunci agar model *prototype* ini berhasil dengan baik adalah dengan mendefinisikan aturan-aturan main pada saat awal, yaitu pemkai dan pengembang harus setuju bahwa *prototype* dibangun untuk mendefinisikan kebutuhan. *Prototype* dapat dihilangkan sebagian atau seluruhnya apabila sudah termasuk di dalam pengkodean sistem.

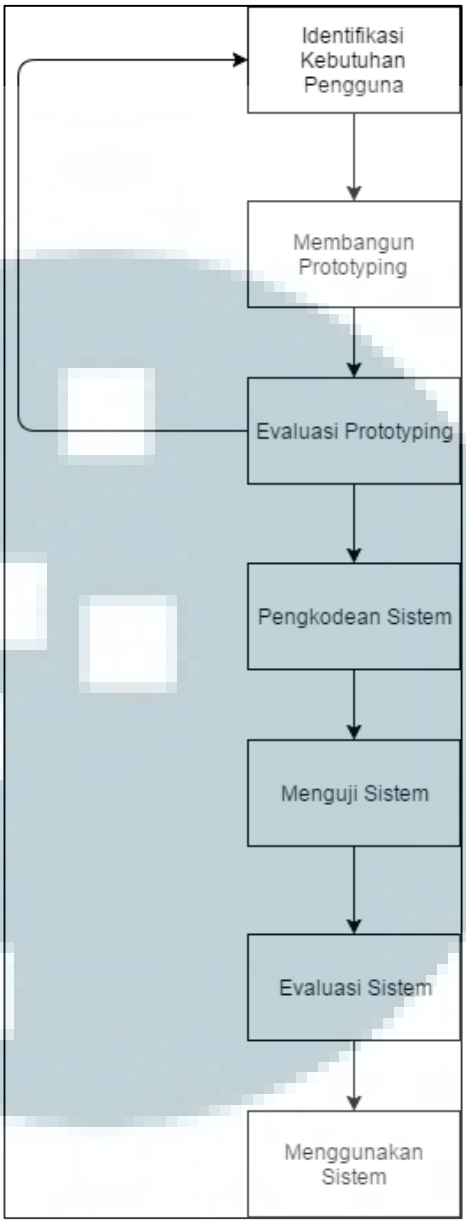


Gambar 2.4 Cara Kerja Protoype

2.7.1. Pemodelan Sistem

Pembangunan model suatu sistem yang baik diperlukan suatu metode atau perangkat pemodelan sistem. Perangkat pemodelan berfungsi sebagai media yang memberikan gambaran penjelasan tentang sistem yang dibuat. Perangkat pemodelan dapat berupa diagram maupun gambar.

Proses pembuatan prototipe merupakan proses yang interaktif dan berulang dan menggabungkan langkah - langkah siklus pengembangan tradisional. *Prototype* dievaluasi beberapa kali sebelum pemakai menggunakan sistem tersebut dan menyatakan protipe tersebut diterima. Gambar 2.5 mengilustrasikan proses pembuatan *prototype*:



Gambar 2.5 Model Prototype

1. Pengumpulan kebutuhan

   Pelanggan dan pengembang bersama-sama mendefinisikan *form*at seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

2. Membangun prototyping

   Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada pelanggan (misalnya dengan membuat input dan *form*at output).

3. Evaluasi protoptyping

   Evaluasi ini dilakukan oleh pelanggan apakah prototyping yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Jika sudah sesuai maka langkah keempat akan diambil. Jika tidak, maka prototyping direvisi dengan mengulang langkah 1, 2 dan 3.

4. Mengkodekan sistem

   Dalam tahap ini prototyping yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

5. Menguji sistem

   Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan.

6. Evaluasi sistem

   Pelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan. Jika sudah, maka langkah ketujuh dilakukan, jika belum maka mengulangi langkah 4 dan 5.

7. Menggunakan sistem

   Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.

2.8. Penelitian Terdahulu

Penelitian – penelitian sejenis ini telah dilakukan sebelumnya, sebab penelitian – penelitian terdahulu dirasa sangat penting dalam sebuah penelitian yang akan dilakukan. Berikut beberapa penelitian terdahulu yang berhubungan dengan penelitian ini yaitu:

Penggunaan masukan suara ucapan manusia untuk mengendalikan sistem penyaklaran pada beberapa alat elektronik menggunakan *smartphone android* (2014). Penelitian ini memiliki tujuan untuk merancang bangun sebuah aplikasi android berdasarkan pengenalan ucapan untuk mengatur beberapa alat elektronik (Boy, 2014).

Adapun manfaat dari penelitian ini diharapkan mampu membantu manusia dalam melakukan kegiatan sehari-hari seperti menyalakan lampu, kipas angin dan membuka gorden.

Adapun beberapa aplikasi yang terdapat pada *Google Play* yang menjadi acuan penulis untuk membangun aplikasi ini. Berikut beberapa aplikasi yang dipilih :

Mic To Speaker adalah aplikasi yang hampir sama kegunaannya untuk mengeluarkan suara, tetapi pengeluaran suara yang dilakukanya hanya dapat dikeluarkan oleh *device* yang terhubung dengan *handphone* tetapi menggunaan kabel.

Microphone Loudspeaker adalah aplikasi yang fleksibel. Aplikasi ini akan mengeluarkan suara yang diterima kepada *device* yang terhubung dengan tipe *out device*, dengan begitu aplikasi inipun dapat melakukan pengeluaran suara kepada *speaker Bluetooth* yang terhubung.

2.9. Sekilas Mengenai Bluetooth

Bluetooth adalah jaringan kawasan pribadi (*personal area networks* atau PAN) tanpa kabel. *Bluetooth* menghubungkan dan dapat dipakai untuk melakukan transaksi penukaran informasi di antara satu *device* ke *device* lainnya. *Bluetooth* beroperasi dalam pita frekuensi 2,4 GHz dengan menggunakan sebuah *frequency hopping traceiver* yang mampu menyediakan layanan komunikasi data dan suara secara real time antara *host - host Bluetooth* dengan jarak yang terbatas. Kelemahan teknologi ini adalah jangkauannya yang pendek dan kemampuan transfer data yang rendah.

Berikut beberapa *link* yang ada pada *Bluetooth*, yaitur terdapat dua tipe dari *link* fisik ditemukan pada versi 1.1 dari Spesifikasi *Bluetooth*, dua tipe itu adalah *Synchronous Connection Oriented* (SCO) dan *Asynchronous Connection Less* (ACL). *Link* SCO dan ACL adalah bagian dasar dari spesifikasi *Bluetooth*.

Link SCO digunakan untuk pengiriman paket berupa suara elektrik. Semua *link* SCO beroperasi pada 64 kbps. Sebuah master dapat mengatur

tiga *link* SCO secara bersama-sama pada waktu yang sama pula, untuk *slave* yang sama atau *slave* yang berbeda. *Link* ACL digunakan untuk pengiriman data. Sebuah *link* ACL memiliki "*error free transmission*" yang berarti jika ada paket yang hilang akan dikirim ulang. *Link* ACL ini beroperasi pada 650 kbps.

Logical Link Control and Adaptation Layer (L2CAP). Layer ini mengatur segmentasi paket dan perakitan ulang paket (SAR), *protokol multiplexing*, dan menyediakan informasi *quality of service* (Valvano, 2016).