BAB 2
LANDASAN TEORI

2.1 Perangkat Lunak

Perangkat lunak (*software*) adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan (*user manual*). Sebuah program komputer tanpa terasosiasi dengan dokumentasinya maka belum dapat disebut perangkat lunak (*software*). Sebuah perangkat lunak juga sering disebut dengan sistem perangkat lunak. Sistem berarti kumpulan komponen yang saling terkait dan mempunyai satu tujuan yang ingin dicapai.

Sistem perangkat lunak berarti sebuah sistem yang memiliki komponen berupa perangkat lunak yang memiliki hubungan satu sama lain untuk memenuhi kebutuhan pelanggan (*customer*). Pelanggan (*customer*) adalah orang atau organisasi yang memesan atau membeli perangkat lunak (*software*) dari pengembang perangkat lunak atau bisa dianggap bahwa pelanggan (*customer*) adalah orang atau organisasi yang dengan sukarela mengeluarkan untuk memesan atau membeli perangkat lunak. *User* atau pemakai perangkat lunak adalah orang yang memiliki kepentingan untuk memakai atau menggunakan perangkat lunak untuk memudahkan pekerjaannya [15].

Secara umum perangkat lunak ini dapat diklasifikasikan menjadi 2 [7], yaitu:

1. Perangkat Lunak Sistem Operasi

   Merupakan perangkat lunak yang berfungsi untuk mengkonfigurasikan komputer agar dapat menerima berbagai perintah dasar yang diberikan sebagai masukan. Perangkat lunak sistem juga merupakan sekumpulan program yang ditulis untuk melayani program-program yang lain. Contoh: *MS-DOS, LINUX, UNIX, FREE BSD, OS/2, SUN OS (JAVA), WINDOWS, MACINTOSH, NOVELL,* dan lain sebagainya.

2. Perangkat Lunak Aplikasi

   Merupakan program siap pakai yang digunakan untuk aplikasi dibidang tertentu. Perangkat lunak aplikasi yang membantu *user* sehingga dapat bekerja lebih efektif dan efisien. Aplikasi ini dibagi atas beberapa bidang, antara lain:

   1) *Business & Office Application*

      Aplikasi perkantoran dipergunakan untuk menyelesaikan pekerjaan kantor seperti mengetik (*Word processing, document management*) membuat tabel kerja (*Worksheet*), membuat Database sederhana (*MS. Access*) ataupun mengolah *image*/citra sederhana. Contoh aplikasi perkantoran yang terpaket komplit: *Mocrosoft Office, Koffice, Star Office, Abiword, Open Office, MS_Office for MAC,* dan lain sebagainya.

      Aplikasi bisnis lainnya juga meliputi aplikasi pengolahan data keuangan, misal (*Personal Finance Application & Business Acounting Application*). Contoh: Aplikasi perhitungan neraca, rugi laba perusahaan, Aplikasi perhitungan pajak personal & perusahaan, Aplikasi manajemen proyek, Aplikasi persediaan stok barang dan penjualan, dan lain sebagainya.

   2) *Database Application*

      Aplikasi yang digunakan dalam pengolahan data baik yang berukuran kecil maupun besar bisa digunakan secara *stand-alone* (tunggal) maupun dalam sistem berbasis jaringan lokal *Client server* maupun *webbase* (intranet maupun internet). Contoh: *MS SQL, FoxBase, Oracle, MySQL, PostgresSQL, SQL Server*, dan lain sebagainya.

   3) *Graphics Design Application*

      Aplikasi yang digunakan untuk membuat sketsa dua dimensi untuk desain ruangan, perabotan, mesin-mesin model industri, juga model tubuh manusia, hewan, dan lain sebagainya. Contoh: *AutoCad, Pro Design, CorelDraw, Adobe Ilistrator,* dan lain sebagainya.

Ada juga yang memasukan dalam golongan ini aplikasi *image, audio, video manipulator*, yaitu aplikasi yang digunakan untuk mengolah citra, suara hingga gambar 3D untuk pembuatan animasi, pengolahan *image, photo, audio, video editing*, dan lain sebagainya. Contoh: *Adobe (Photoshop, After Effect, Premire, & Audition), Corel (Draw, RAVE, Photo Paint), Ulead Media studio, GIMP (In Linux), Cool Edit Pro, Sound Force, Macromedia (Flash, Director), 3D Studio Max,* dan lain sebagainya.

4) *Antivirus & Utility Tools*

   Software ini digunakan untuk operasi minor (*Background*) yang bekerja di belakang sistem, berguna untuk meningkatkan kinerja sistem operasi atau aplikasi dan juga performa *hardware*. Terdapat 2 jenis utilitas di dalam sistem operasi:

   a) *Utilitas Internal*, sudah termasuk dalam paket yang di instalasi dalam sistem operasi seperti *scandisk* di *Windows* atau perintah *fsck* (*file system check*) di OS Linux.
   b) *Utilitas eksternal*, yang harus diinstalasi kemudian pada OS, seperti *System Optimizer, PC Tools, Data/File Recovery, PC Maintenance, Internet Utilities, Norton System Work, Registry Clean & Repair, Hardware Repair*.

   Sementara untuk proteksi virus dan antivirus, contohnya adalah antivirus produk *Mc Affee, Grisoft, Norton, PandaSoft,* dan lain sebagainya.

5) *Development Tools*

   Merupakan sistem bahasa pemrograman komputer yang dilengkapi dengan *compiler* tertentu sebagai sumber media penyusunan program aplikasi. Contoh: *QuikBasic, Clipper, Visual Basic, Turbo Pascal, Visual C++, GCC, Turbo Assembler, Phyton, KDEvelopt, Gdekstop,* dan lain sebagainya.

   Termasuk dalam jenis aplikasi ini adalah *web development tools*, yaitu aplikasi yang digunakan untuk membuat/mendesain tampilan website

(*homepage*) di internet, seperti *FrontPage, Macromedi Dreamweaver, PHPedit, WebPage Editor,* dan lain sebagainya.

6) *Communication Application*

   Aplikasi yang digunakan untuk membangun komunikasi antara satu komputer dengan komputer lain dalam jaringan komputer local maupun global. Contoh: *E-mail & Fax Application, Remote Access, Facebook chat, YM, GTalk, MSN, Terminal Emulation, Instant Messaging, Voice Recognition, Web Browser, Internet Telephony,* dan lain sebagainya.

7) Jenis Aplikasi Lain

   a) *Multimedia Player,* seperti *WinAmp, XMMS, Power DVD, JetAudio, Windows Media Player,* dan lain sebagainya.

   b) *Hacker Tools,* seperti *Find Password, Keygen/Crack Software License, Scanning Port,* dan lain sebagainya.

   c) *Network & Security Application,* seperti *Firewall, Intruder Detections, Network Monitoring,* dan lain sebagainya.

   d) *PC Games Apllication,* dan masih banyak lagi.

2.2 Basis Data

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk penyimpanan data agar dapat diakses dengan mudah dan cepat.

Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa *file* teks ataupun *Database Management System* (DBMS) [15]. Kebutuhan basis data dalam sistem informasi meliputi memasukan, menyimpan, dan mengambil data serta membuat laporan berdasarkan data yang telah disimpan.

2.2.1 *Database Management System* (DBMS)

DBMS (*Database Management System*) atau dalam bahasa indonesia sering disebut sebagai Sistem Manajemen Data adalah suatu sistem aplikasi yang

digunakan untuk menyimpan, mengelola, dan menampilkan data. Suatu sistem aplikasi disebut DBMS jika memenuhi persyaratan minimal dapat menyediakan fasilitas untuk mengelola akses data, mampu menangani integrasi data, dan mampu menangani *backup* data.

DBMS sudah mulai berkembang sejak tahun 1960an. Kemudian sekitar tahun 1970an mulai berkembang teknologi *Relational* DBMS yaitu DMBS berbasis relasional model. Relasional model pertama kali dikembangkan oleh Edgar J. Codd pada tahun 1970. Secara sederhana relasional model dapat dipahami sebagai suatu model yang memandang data sebagai sekumpulan tabel yang saling terkait.

Pada tahun 1980an mulai berkembang *Object Oriented* DBMS (OODBMS). OODBMS berkembang seiring dengan perkembangan teknologi pemrograman berorientasi objek. Secara umum dapat diartikan bahwa OODBMS merupakan DBMS yang memandang data sebagai suatu objek.

Terdapat beberapa macam DBMS versi komersial yang paling sering digunakan di dunia saat ini, diantaranya Oracle, Microsoft SQL Server, IBM DB2, dan Microsoft Access. Sedangkan DBMS versi *open source* yang cukup berkembang dan sering digunakan saat ini adalah diantaranya MySQL, PostgreSQL, Firebird, dan SQLite. Hampir semua DBMS mengadopsi SQL sebagai bahasa untuk mengelola data pada DBMS.

2.2.2 *Structured Query Language* (SQL)

SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada DBMS. Meskipun SQL diadopsi dan diacu sebagai bahasa standar oleh hampir sebagian besar RDBMS yang beredar saat ini, tetapi tidak semua standar yang tercantum dalam QSL diimplementasikan oleh seluruh DBMS tersebut. Sehingga terkadang terdapat perbedaan prilaku (hasil yang ditampilkan) oleh DBMS yang berbeda padahal *query* yang dimasukan sama.

2.3 Pemrograman Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem

informasi, atau sistem lainnya). Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dari abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dimiliki oleh objek akan dienkapsulasi (dibungkus) menjadi satu kesatuan.

Sistem berorientasi objek merupakan sebuah sistem yang kompunennya dibungkus (dienkapsulasi) menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat, dan kompunen lainnya yang dapat berinteraksi satu sama lain.

Terdapat beberapa konsep utama pada metodologi berorientasi objek, diantaranya [15]:

1. Kelas (*class*), kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dari himpunan objek yang sama yang mungkin lahir atau diciptakan dari kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*), dan arti. Suatu kelas dapat diturunkan dari kelas yang lain, dimana atribut dari kelas semula dapat diwariskan ke kelas yang baru.
2. Objek (*object*), abstraksi dari sesuatu yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.
3. Abstraksi (*abstraction*), prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi suatu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
4. Enkapsulasi (*encapsulation*), pembungkusan atribut data dan layanan (operasi-operasi) yang dimiliki objek untuk menyembunyikan

implementasi dari objek sehingga objek lain tidak mengetahui cara kerjanya.

5. Pewarisan (*inheritance*), mekanisme yang memungkinkan suatu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dari dirinya.
6. Polimorfisme (*polymorphism*), kemampuan seuatu objek untuk digunakan dibanyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.4 *Unified Modeling Language* (UML)

UML (*Unified Modeling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*) [8]. UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software. *Unified Modelling Language* (UML) juga merupakan sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Terdapat beberapa diagram yang umum digunakan dalam pemodelan UML diantaranya [15]:

1. *Use Case Diagram*, merupakan pemodelan untuk kelakuan (*behavior*) sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antar satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu.
2. *Class Diagram*, menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki atribut dan metode (operasi). Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode (operasi) fungsi-fungsi yang dimiliki

oleh suatu kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki kelas *main*, kelas *view*, kelas *controller*, serta kelas *model*.

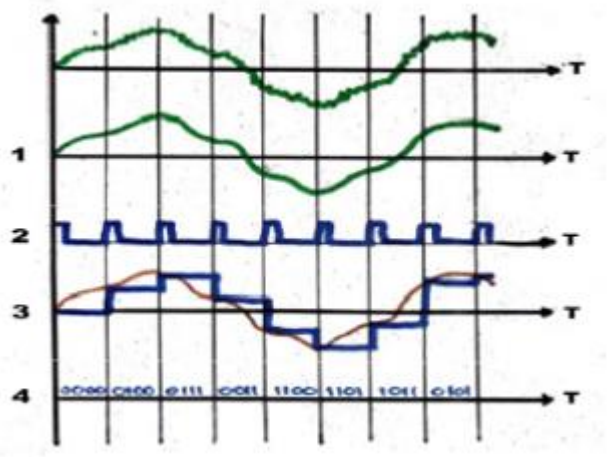
3. *Sequence Diagram,* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui onjek-objek yang terlibat dalam sebuah *use cse* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.
4. *Activity Diagram,* menggambarkan aliran kerja (*workflow*) 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, melainkan aktivitas yang dapat dilakukan oleh sistem.
5. *Component Diagram,* dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram komonen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Komponen dasar yang biasanya ada dalam suatu sistem adalah komponen *user interface* yang menagani tampilan, komponen *bussiness processing* yang menangani fungsi-fungsi proses bisnis, komponen *data* yang menangani manipulasi data, dan komponen *security* yang menangani keamanan sistem. Komponen lebih terfokus pada penggolongan secara umum fungsi-fungsi yang diperlukan.

2.5 *Analog To Digital Converter* (ADC)

Gelombang suara analog tidak dapat langsung direpresentasikan pada komputer / *device*. Gelombang suara dimanipulasi hingga dapat diubah ke dalam bentuk digital. Amplitudo adalah keras lemahnya bunyi atau tinggi rendahnya gelombang yang direpresentasikan pada satuan *decibel* (db). Komputer / *device*

mengukur amplitudo pada satuan waktu tertentu untuk menghasilkan sejumlah angka. Tiap satuan angka ini dinamakan *sample*.

Terdapat beberapa tahapan dalam mengkonversi suara berupa sinyal analog kedalam bentuk digital (*Analog Digital Converter*). Proses ADC dapat dilihat pada gambar 2.1 [1].



Gambar 2.1 Ilustrasi *Analog Digital Converter*

Keterangan:

1. Membuang frekuensi tinggi dari *source signal*
2. Mengambil sample pada interval waktu tertentu (sampling)
3. Menyimpan *amplitudo sample* dan mengubahnya kedalam bentuk diskrit (kuantisasi)
4. Merubah bentuk diskrit menjadi *biner*

2.6 Internet

Internet merupakan jaringan global komputer dunia, besar dan sangat luas dimana setiap komputer saling terhubung satu sama lainnya dari negara ke negara lainnya di seluruh dunia dan berisi berbagai macam informasi, mulai dari teks, gambar, audio, video, dan lain sebagainya yang terhubung melalui protokol TCP/IP. Dalam mengatur integrasi dan komunikasi jaringan komputer ini menggunakan protokol yaitu TCP/IP. TCP (*Transmission Control Protocol*) bertugas untuk memastikan bahwa semua hubungan bekerja dengan benar, sedangkan IP (*Internet Protocol*) yang mentransmisikan data dari satu komputer

ke komputer lain. TPC/IP secara umum berfungsi memilih rute terbaik transmisi data, memilih rute alternatif jika suatu rute tidak dapat di gunakan, mengatur dan mengirimkan paket-paket pengiriman data [9].

2.7 Aplikasi *Mobile*

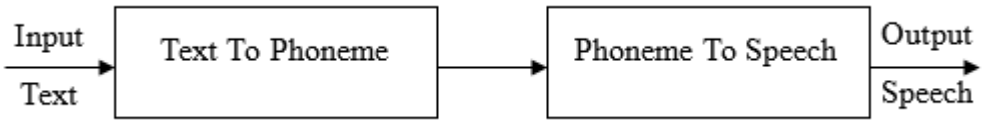
Aplikasi *mobile* adalah sebuah aplikasi yang memungkinkan untuk melakukan mobilitas dengan menggunakan perlengkapan seperti telepon genggam (*hanphone*), PDA, atau *smartphone*. Aplikasi *mobile* dapat mengakses dan menggunakan sebuah aplikasi secara *nirkabel* dengan menggunakan perangkat *mobile*, dimana data yang diperoleh hanya berupa teks sehingga tidak perlu membutuhkan *bandwidth* yang terlalu besar. Data dalam bentuk teks tidak semenarik data dalam bentuk teks disertai dengan gambar. Namun penggunaan data teks adalah yang paling banyak digunakan saat ini. Penggunakan aplikasi *mobile* hanya memerlukan *handphone* yang sudah dilengkapi dengan fasilitas GPRS dan koneksinya [3].

Terdapat beberapa aspek yang harus diperhatikan khususnya perangkat keras dalam membangun sebuah aplikasi *mobile*. Selain itu pertimbangan terhadap keterbatasan pada piranti *mobile* pun harus di perhatikan, antara lain:

1. Keterbatasan dari kecepatan *processor* dalam mengeksekusi program
2. Keterbatasan memori penyimpanan dan RAM
3. Perbedaan ukuran layar secara fisik dan resolusi pada masing-masing piranti
4. Keterbatasan input pada masing-masing piranti mobile
5. Ketahanan baterai yang berbeda pada setiap piranti mobile
6. Selain itu dari segi perangkat lunak (*software*), kompatibilitas dan *mobile* pendukung cukup berpengaruh dalam menjalankan sebuah aplikasi *mobile*.

2.8 *Text To Speech* (TTS)

Sistem *Text to Speech* pada prinsipnya terdiri dari dua sub sistem [5], yaitu:



Gambar 2.2 Skema *Text To Speech*

1. Bagian Konverter Teks ke Fonem (*Text to Phoneme*)

   Bagian Konverter Teks ke Fonem berfungsi untuk mengubah kalimat masukan dalam suatu bahasa tertentu yang berbentuk teks menjadi rangkaian kode-kode bunyi yang biasanya direpresentasikan dengan kode fonem, durasi serta *pitch*-nya. Bagian ini bersifat sangat *language dependant*. Untuk suatu bahasa baru, bagian ini harus dikembangkansecara lengkap khusus untuk bahasa tersebut.

2. Bagian Konverter Fonem ke Ucapan (*Phoneme to Speech*)

   Bagian Konverter Fonem ke Ucapan menerima masukan berupa kode-kode fonem serta pitch dan durasi yang dihasilkan oleh bagian sebelumnya. Berdasarkan kode-kode tersebut, bagian Konverter Fonem ke Ucapan akan menghasilkan bunyi atau sinyal ucapan yang sesuai dengan kalimat yang ingin diucapkan. Ada beberapa alternatif teknik yang dapat digunakan untuk implementasi bagian ini. Dua teknik yang banyak digunakan adalah *formant synthesizer* serta *diphone concatenation*. Sub sistem ini harus memiliki pustaka setiap unit ucapan dari suatu bahasa.

Formant synthesizer bekerja berdasarkan suatu model matematis yang akan melakukan komputasi untuk menghasilkan sinyal ucapan yang diinginkan. *Synthesizer* jenis ini telah lama digunakan pada berbagai aplikasi. Walaupun dapat menghasilkan ucapan dengan tingkat kemudahan interpretasi yang baik, *synthesizer* ini tidak dapat menghasilkan ucapan dengan tingkat kealamian yang tinggi.

Synthesizer yang menggunakan teknik *diphone concatenation* bekerja dengan cara menggabung-gabungkan segmen-segmen bunyi yang telah direkam sebelumnya. Setiap segmen berupa *diphone* (gabungan dua buah fonem).

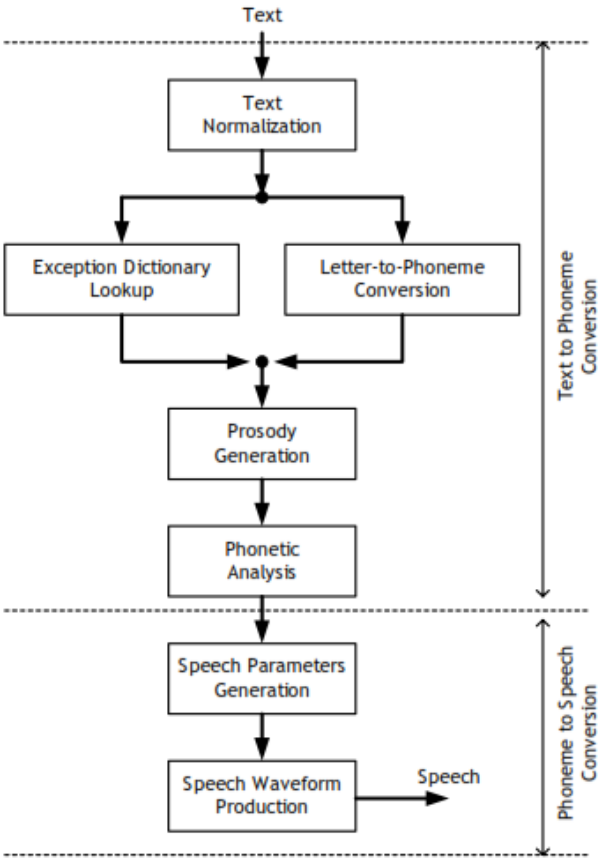
Synthesizer jenis ini dapat menghasilkan bunyi ucapan dengan tingkat kealamian (*naturalness*) yang tinggi.

Struktur sistem seperti di atas pada prinsipnya merupakan konfigurasi tipikal yang digunakan pada berbagai sistem *Text to Speech* berbagai bahasa. Namun demikian, pada setiap sub-sistem terdapat sifat-sifat serta proses-proses yang sangat spesifik dan sangat tergantung dari bahasanya.

Konversi dari teks ke fonem sangat dipengaruhi oleh aturan-aturan yang berlaku dalam suatu bahasa. Pada prinsipnya proses ini melakukan konversi dari simbol-simbol tekstual menjadi simbol-simbol fonetik yang merepresentasikan unit bunyi terkecil dalam suatu bahasa. Setiap bahasa memiliki aturan cara pembacaan dan cara pengucapan teks yang sangat spesifik. Hal ini menyebabkan implementasi unit konverter teks ke fonem menjadi sangat spesifik terhadap suatu bahasa. Untuk mendapatkan ucapan yang lebih alami, ucapan yang dihasilkan harus memiliki intonasi (*prosody*). Secara kuantisasi, prosodi adalah perubahan nilai *pitch* (frekuensi dasar) selama pengucapan kalimat dilakukan atau pitch sebagai fungsi waktu. Pada prakteknya, informasi pembentuk prosodi berupa data-data *pitch* serta durasi pengucapannya untuk setiap fonem yang dibangkitkan. Nilai-nilai yang dihasilkan diperoleh dari suatu model prosodi. Prosodi bersifat sangat spesifik untuk setiap bahasa, sehingga model yang diperlukan untuk membangkitkan data-data prosodi menjadi sangat spesifik juga untuk suatu bahasa. Beberapa model umum prosodi pernah dikembangkan, tetapi untuk digunakan pada suatu bahasa masih perlu banyak penyesuaian yang harus dilakukan.

Konverter fonem ke ucapan berfungsi untuk membangkitkan sinyal ucapan berdasarkan kode-kode fonem yang dihasilkan dari proses sebelumnya. Sub sistem ini harus memiliki pustaka setiap unit ucapan dari suatu bahasa. Pada sistem yang menggunakan teknik *diphone concatenation*, sistem harus didukung oleh suatu *diphone database* yang berisi rekaman segmen-segmen ucapan yang berupa *diphone*. Ucapan dalam suatu bahasa dibentuk dari satu set bunyi yang mungkin berbeda untuk setiap bahasa, oleh karena itu setiap bahasa harus dilengkapi dengan diphone database yang berbeda.

Tahapan-tahapan utama konversi dari teks menjadi ucapan dapat dinyatakan dengan diagram seperti terlihat pada Gambar 2.3.



Gambar 2.3 Urutan Proses Konversi dari Teks ke Ucapan

2.9 *Speech To Text* (STT)

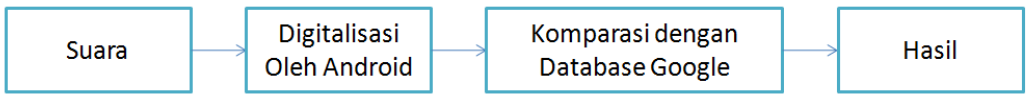
Speech to text atau dikenal sebagai *speech recognition* adalah suatu pengembangan teknik dan sistem yang memungkinkan komputer atau perangkat pintar lainnya untuk menerima masukan berupa kata yang diucapkan. Teknologi ini memungkinkan suatu perangkat untuk memahami kata-kata yang diucapkan dengan cara digitalisasi kata dan mencocokkan sinyal digital tersebut dengan *diphone database* yang tersimpan dalam suatu perangkat atau *server*. Kata-kata yang diucapkan diubah bentuknya menjadi sinyal digital dengan cara mengubah gelombang suara menjadi sekumpulan kode tertentu yang kemudian digunakan untuk mengidentifikasikan kata-kata tersebut. Hasil dari identifikasi kata yang diucapkan dapat ditampilkan dalam bentuk tulisan atau dapat dibaca oleh

perangkat teknologi sebagai sebuah perintah untuk melakukan suatu pekerjaan, misalnya penekanan tombol pada telepon genggam yang dilakukan secara otomatis dengan komando suara.

Untuk mengubah percakapan menjadi teks *on-screen* atau perintah tertentu, komputer melakukan beberapa langkah yang kompleks. Ketika berbicara, seseorang mengeluarkan getaran di udara. Kemudian, *analog-to-digital converter* (ADC) yang ada di *soundcard* menerjemahkan gelombang analog ini menjadi data digital yang dapat dimengerti oleh komputer.

Untuk melakukan hal tersebut, sistem *Speech Recognition* melakukan *sampling* atau *digitizing* suara dengan cara mengambil ukuran yang paling pas dari gelombang. Sistem menyaring suara yang telah didigitalkan tersebut dan membuang gangguan (*noise*), dan kadang-kadang memisahkannya ke dalam pita frekuensi yang berbeda. Frekuensi adalah panjang gelombang suara, yang terdengar oleh telinga manusia sebagai tinggi nada (*pitch*) yang berbeda.

Sistem ini juga menormalkan suara, atau mengaturnya ke dalam tingkat volume yang tetap, terkadang juga mendatarkan suara. Manusia tidak berbicara dalam kecepatan yang sama sehingga suara harus diatur dengan kecepatan yang sama dengan sampel-sampel template suara yang tersimpan dalam komputer atau *server*.



Gambar 2.4 Diagram Alir *Speech Recognition*

Langkah selanjutnya adalah memecah sinyal menjadi bagian-bagian kecil, dengan durasi seperseratus detik, atau bahkan seperseribu pada kasus bunyi konsonan atau mati. Konsonan memberhentikan produksi suara dengan menghalangi aliran gelombang pada bidang vokal, seperti “p” atau “t”.

Program di komputer kemudian mencocokkan bagian-bagian kecil ini dengan fonem yang dikenal dalam bahasa tertentu. Fonem adalah elemen terkecil dalam sebuah bahasa, merepresentasikan suara, dan merangkainya ke dalam bentuk ujaran yang memiliki makna.

Tahap berikutnya terlihat sederhana, tapi pada dasarnya merupakan proses yang paling susah diselesaikan, sekaligus merupakan inti dari sebagian besar penelitian di bidang *Speech Recognition*. Komputer memeriksa fonem-fonem dalam konteks (hubungan) dengan fonem-fonem lain yang menyertainya.

Komputer menjalankan alur (*plot*) melalui sebuah model statistika yang kompleks, dan membandingkannya dengan koleksi kata, frase, dan kalimat yang telah dikenal. Program *Speech Recognition* selanjutnya menentukan apa yang mungkin dikatakan oleh pengguna, dan juga mengetikkannya sebagai teks atau mengeluarkannya sebagai perintah pada komputer [6].

2.10 Google API

Google API merupakan bagian dari *Framework* Google. Google menyediakan berbagai API (*Application Programming Interface*) yang sangat berguna bagi pengembang aplikasi baik web, *mobile*, maupun desktop untuk memanfaatkan berbagai fitur yang disediakan oleh Google seperti: *AdSense*, *Search Engine*, *Translation, YouTube* dan lain sebagainya.

API secara sederhana bisa diartikan sebagai kode program yang merupakan antarmuka atau penghubung antara aplikasi atau web yang dibuat dengan fungsi-fungsi yang dikerjakan. Misalnya dalam hal ini Google API berarti kode program yang dapat ditambahkan pada aplikasi untuk memanfaatkan fungsi atau fitur yang disediakan Google. Contohnya *developer* dapat menambahkan fitur Google Map pada apliaksi yang sedang dibangun. Google API dapat dipelajari langsung melalui *Google Code*. Melalui *Google Code*, *developer* dapat belajar tentang Google API dan dapat mengimplementasikan pada aplikasi yang sedang dikembangkan. Ada banyak API yang disediakan oleh Google, beberapa diantaranya adalah [10]:

1. *Language API*: memanfaatkan fitur translation yang dimiliki Google
2. *Earth API*: memanfatkan fitur yang ada pada Google Earth
3. *Maps API*: memanfaatkan fitur yang ada pada Google Maps
4. *Search API*: memanfaatkan fitur pencarian pada Google Search
5. *Visualization API*: membuat grafik maupun chart dengan Google API

6. YouTube API: memanfaatkan fitur yang ada pada YouTube misalnya untuk pencarian video.

2.10.1 *Google Speech Recognition API*

Penggunaan *Google Speech Recognition API* memungkinkan *developer* untuk mengkonversi ucapan kedalam teks. Untuk dapat menggunakan fitur tersebut, *developer* android dapat menggunakan *interface* dan *class* yang telah disediakan Google API (*package android.speech)* [12].

1. *Interfaces*

   RecognitionListener digunakan untuk menerima pemberitahuan dari *SpeechRecognizer* ketika peristiwa pengenalan (*recognition*) terkait terjadi.

2. *Classes*

   Terdapat beberapa *class* yang terdapat pada *package android.speech*, diantaranya:

   a) *RecognitionService*, kelas ini menyediakan kelas dasar untuk implementasi layanan pengenalan
   b) *RecognitionService.Callback*, kelas ini menerima *callback* dari layanan pengenalan suara dan mengirimkannya ke pengguna
   c) *RecognizerIntent*, konstanta untuk mendukung pengenalan suara melalui memulai *Intent*
   d) *RecognizerResultsIntent*, konstanta untuk *intents* yang berkaitan dengan menunjukkan hasil pengenalan suara
   e) *SpeechRecognizer*, kelas ini menyediakan akses ke layanan pengenalan suara.

Untuk dapat memanfaatkan fitur *Google Speech Recognition API* pada program diperlukan beberapa langkah, diantaranya mengimpor *class* yang berhubungan dengan *speech recognition*, yaitu `android.speech.RecognizerIntent;`

Pada dasarnya memicu *Intent* (*android.speech.RecognizerIntent*) berfungsi untuk menunjukan kotak dialog untuk mengenali suara masukan. *Activity* ini kemudian mengubah suara menjadi teks dan mengirim kembali hasil dari

pemanggilan *Activity*. Ketika memanggil *Intent android.speech.RecognizerIntent*, *developer* harus menggunakan `startActivityForResult()` untuk menunjukkan hasil pengenalan suara.

2.10.2 *Google Text To Speech API*

Penggunaan *Google Text To Speech API* memungkinkan *developer* untuk mengkonversi teks kedalam ucapan (suara). Untuk dapat menggunakan fitur tersebut, *developer* android dapat menggunakan *interface* dan *class* yang telah disediakan Google API (*package android.speech.tts)* [13].

1. *Interfaces*

   Terdapat beberapa *interface* yang terdapat pada *package android.speech.tts*, diantaranya:

   a) *SynthesisCallback*
   b) *TextToSpeech.OnInitListener*
   c) *TextToSpeech.OnUtteranceCompletedListener*

2. *Classes*

   Terdapat beberapa *class* yang terdapat pada *package android.speech.tts*, diantaranya:

   a) *SynthesisRequest*, berisi data yang dibutuhkan oleh mesin untuk mensintesis ucapan
   b) *TextToSpeech*, mensintesis ucapan dari teks untuk diputar langsung atau untuk membuat file suara
   c) *TextToSpeech.*Engine, konstanta dan nama parameter untuk mengendalikan *text-to-speech*
   d) *TextToSpeech.*EngineInfo, informasi tentang terpasangnya mesin *text-to-speech*
   e) *TextToSpeechService,* kelas abstrak dasar untuk implementasi mesin TTS
   f) *UtteranceProgressListener*, *listener* untuk kejadian yang berkaitan dengan kemajuan suatu ucapan melalui antrian sintesis.

Untuk dapat memanfaatkan fitur *Google Text To Speech API* pada program diperlukan beberapa langkah, diantaranya mengimpor *class* yang berhubungan dengan *text to speech*, yaitu `android.speech.tts.TextToSpeech;`

Untuk menggunakan kelas *TextToSpeech*, perlu menginisialisasi sebuah objek dari *class TextToSpeech* dan menentukan *initListenere*.

```
private EditText write;
ttobj = new TextToSpeech(getApplicationContext(),
        new TextToSpeech.OnInitListener() {
   @Override
   public void onInit(int status) {
   }
}
);
```

Pada *listener* ini, harus menentukan properti untuk objek *text to speech*, seperti bahasa, *pitch* dan lain sebagainya. Bahasa dapat diatur dengan memanggil *method setLanguage()*, yaitu

```
ttobj.setLanguage(Locale.UK);
```

Method setLanguage mengambil objek lokal sebagai parameter. Berikut contoh daftar dari beberapa objek lokal (tergantung masing-amsing *device*): US, CANADA_FRENCH, GERMANY, ITALY, JAPAN, CHINA.

Setelah mengatur bahasa, dapat memanggil method speak.

```
ttobj.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null);
```

Terlepas dari *method speak*, ada beberapa *method* lain yang disediakan pada *class text to speech* yaitu *addSpeech(String text, String filename), getLanguage(), isSpeaking(), setPitch(float pitch), setSpeechRate(float speechRate), shutdown()*, dan *stop()*.

Selain menggunakan *class* `android.speech.tts.TextToSpeech;` yang disediakan oleh Google, developer dapat menggunakan HTTP GET (REST) request: http://translate.google.com/translate_tts?ie=UTF-8&tl=en&q=text untuk mengkonversi teks menjadi suara. Dimana *tl* merupakan bahasa yang digunakan, misalnya, *ja* untuk bahasa *Japanese* atau *en* untuk bahasa *English*, sedangkan *q* merupakan *query string* (teks) yang akan dikonversi menjadi suara (ucapan). Namun dalam penggunaannya, metode ini hanya terbatas pada 100 karakter.

2.11 Microsoft Translator API

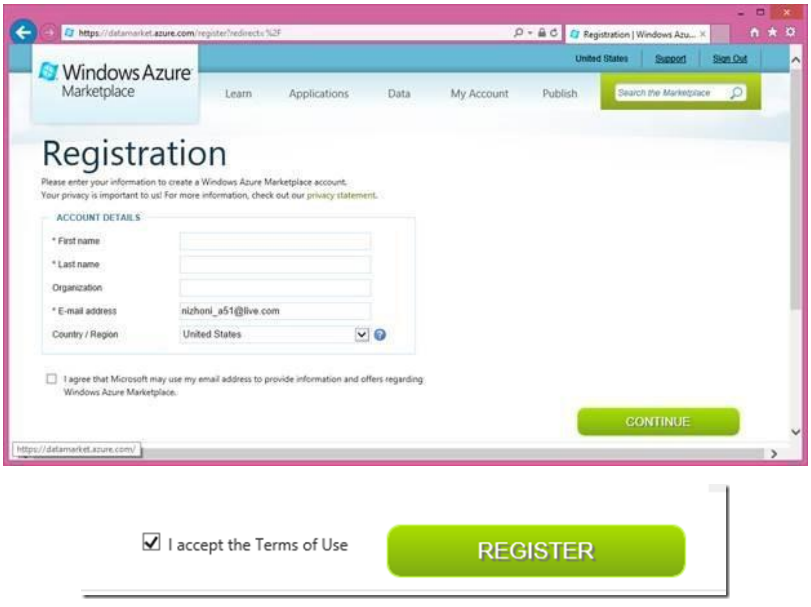
Microsoft Translator API atau *Bing Translator API* adalah sebuah layanan yang disediakan untuk menerjemahkan teks ke bahasa tertentu. Saat ini *Microsoft Translator API* menjadi alternatif yang banyak digunakan *developer* dalam mengebangkan aplikasi mengingat Google tidak lagi menggratiskan layanan *Google Translate API*.

Microsoft Translator API merupakan sebuah layanan *online* yang tersedia pada *Windows Azure Marketplace*. *Microsoft Translator API* menjual layanannya berdasarkan jumlah karakter teks yang dikirimkan ke API per bulan dan tersedia secara gratis untuk penggunaan sampai dengan 2 juta karakter per bulan.

Untuk dapat menggunakan layanan *Microsoft Translator API*, *developer* diharuskan terdaftar pada *Windows Azure Marketplace* untuk dapat memiliki *Client ID* dan *Client Secret*. *Client ID* dan *Client Secret* digunakan Microsoft untuk mengetahui aplikasi pihak ketiga yang sedang menggunakan layanannya tersebut. Berikut langkah-langkah untuk mendapatkan *account* pada *Windows Azure Marketplace*:

1. *Sign-In* dan *Register*

   Sign-in ke *Windows Azure Marketplace*. Jika ini pertama kali, maka akan tampil form registrasi. Isi kolom yang tersedia kemudian klik "*Continue*".



Gambar 2.5 Form registrasi *Windows Azure Marketplace*

Kemudian akan tampil halaman *Terms of Use*. Cek pada "*I accept the Terms of Use*" dan klik "*Register*".

2. Berlangganan ke *Microsoft Translator API*

   a) Cari layanan *Micrososft Translator API* di *Windows Azure Marketplace*

   Setelah menyelesaikan registrasi *Windows Azure Marketplace* (langkah 1), kemudian kunjungi https://datamarket.azure.com, pada kotak pencarian ketikan "*Translator*" dan tekan *enter*.



Gambar 2.6 Kotak Pencarian *Windows Azure Marketplace*

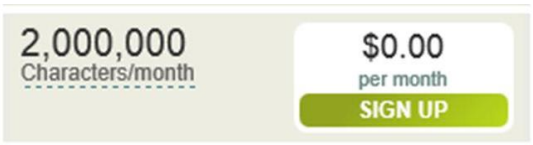
Pada hasil pencarian akan ditemukan *Microsoft Translator API*.



Gambar 2.7 Hasil Pencarian

   b) Berlangganan ke *Microsoft Translator API*

   Klik *Microsoft Translator*, kemudian akan tampil halaman tawaran yang diberikan oleh *Microsoft Translator API*. Di samping kanan layar akan terlihat berbagai paket pemakaian perbulan. Pilihlah salah satu paket yang diberikan, mulai dari yang gratis ataupun yang berbayar. Pada penelitian ini penulis memilih paket gratis dengan *volume* pemakaian 2 juta karakter per bulan.



Gambar 2.8 Paket Gratis Layanan *Microsoft Translator*

Klik "*Sign Up*".

3. Mendapatkan *Developer Credentials*.

   Pada tahap ini diasumsikan telah menyelesaikan langkah 1 dan 2, telah terdaftar pada *Windows Azure Marketplace*, dan telah menggunakan *account* azure marketplace untuk berlangganan layanan *Microsoft Translator API*.

   a) Mendaftarkan Aplikasi pada *Windows Azure Marketplace*

      Sign-in ke https://datamarket.azure.com/, pada bagian bawah halaman akan terlihat beberapa link. Klik "*Register Your Application*".

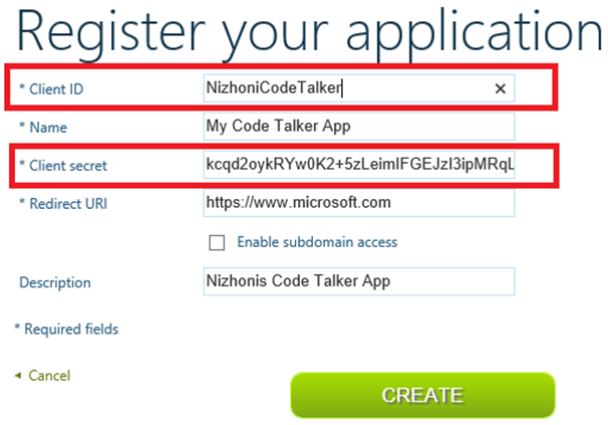


Gambar 2.9 *Register Your Appplication*

   b) Mendapatkan *Client ID* dan *Client Secret*

      Client ID dan *Client Secret* digunakan untuk mengotentikasi aplikasi pada saat menggunakan layanan *Microsoft Translator API*.

      1) Isi *field Client ID* dan *Name*
      2) *Field Client Secret* akan terisi secara otomatis dan jangan diubah.
      3) Isi "*Redirect URL*" dengan URL https yang *valid* seperti https://microsoft.com



Gambar 2.10 *Client ID* dan *Client Secret*

4) Klik "*Create*"

Untuk dapat menggunakan fungsi *Microsoft Translator API* pada pemrograman android, *developer* android memutuhkan suatu *library* yaitu *microsoft-translator-java-api-0.6.2-jar-with-dependencies.jar* yang merupakan *library* yang dibangun oleh pihak ketiga dengan *licensed* dibawah *Apache License, Version* 2.0 yang dibuat oleh Jonathan Griggs pada tahun 2013 lalu guna menjembatani *developer* android agar dapat mengakses *Microsoft Translator API* yang pada pemrogramannya menggunakan bahasa pemrograman ASP.

Dukungan bahasa yang terdapat pada *library microsoft-translator-java-api-0.6.2-jar-with-dependencies.jar* adalah 41 bahasa, yaitu: *Arabic*, *Bulgarian*, *Catalan*, *Chinese Simplified*, *Chinese Traditional*, *Czech*, *Danish*, *Dutch*, *English*, *Estonian*, *Finnish*, *French*, *German*, *Greek*, *Haitian Creole*, *Hebrew*, *Hindi*, *Hmong Daw*, *Hungarian*, *Indonesian*, *Italian*, *Japanese*, *Korean*, *Latvian*, *Lithuanian*, *Malay*, *Norwegian*, *Persian*, *Polish*, *Portuguese*, *Romanian*, *Russian*, *Slovak*, *Slovenian*, *Spanish*, *Swedish*, *Thai*, *Turkish*, *Ukrainian*, *Urdu*, dan *Vietnamese*.

Pada program, untuk dapat memanfaatkan fitur *Microsoft Translator API* diperlukan beberapa langkah, diantaranya mengimpor *class* `com.memetix.mst.language.Language` untuk mengambil bahasa dan `com.memetix.mst.translate.Translate` untuk menerjemahkan percakapan.

2.12 Android

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. Awalnya, Google Inc. membeli Android Inc., yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan *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. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD). Tidak hanya menjadi sistem operasi di *smartphone*, saat ini Android menjadi pesaing utama dari Apple pada sistem operasi Tablet PC.

Pengembang memiliki beberapa pilihan ketika membuat apliaksi yang berbasis android. Sebagian besar pengembang menggunakan *Eclipse* yang tersedia secara bebas untuk merancang dan mengembangkan Aplikasi Android. *Eclipse* adalah IDE yang paling populer untuk pengembangan Android, karena memiliki Android *plug-in* yang tersedia untuk memfasilitasi pengembangan Android. Selain itu, *Eclipse* juga mendapat dukungan langsung dari Google untuk menjadi IDE pengembangan aplikasi Android, ini terbukti dengan adanya penambahan *plugin* untuk *eclipse* untuk membuat project android dimana *source software* langsung dari situs resmi Google. Akan tetapi, hal tersebut tidak menutup kemungkinan untuk menggunakan IDE yang lain seperti Netbeans untuk melakukan pengembangan android [16].

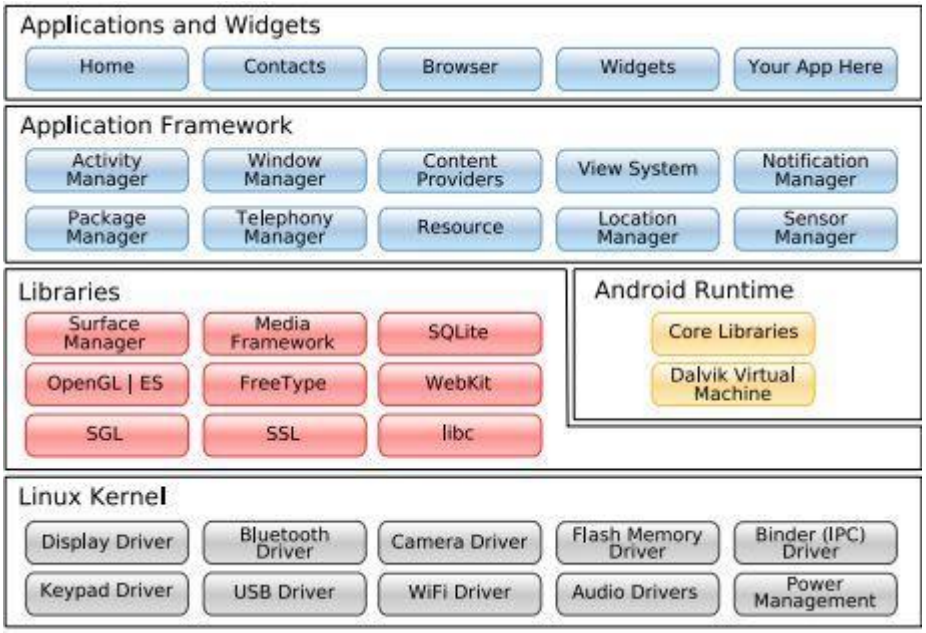
Beberapa versi Android yang telah dirilis diantaranya :

1. Android 1.0 / 1.1 (API level 1 / 2)
2. Android 1.5 *Cupcake* (API level 3)
3. Android 1.6 *Donut* (API level 4)
4. Android 2.0 / 2.0.1 / 2.1 *Eclair* (API level 5 / 6 / 7)
5. Android 2.2-2.2.3 *Frozen Yoghurt* (API level 8)
6. Android 2.3-2.3.2 / 2.3.3-2.3.7 *Gingerbread* (API level 9 / 10)
7. Android 3.0 / 3.1 / 3.2 *Honeycomb* (API level 11 / 12 / 13)
8. Android 4.0-4.0.2 / 4.0.3-4.0.4 *Ice Cream Sandwich* (API level 14 / 15)

9. Android 4.1 / 4.2 / 4.3 *Jelly Bean* (API level 16 / 17 / 18)
10. Android 4.4 *KitKat* (API level 19)

2.12.1 Arsitektur Android

Secara garis bersar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut [16]:



Gambar 2.11 Arsitektur Android

1. *Applications* dan *Widgets*

   Applications dan *Widgets* ini adalah *layer* dimana pengguna berhubungan dengan aplikasi saja, dimana biasanya pengguna men-*download* aplikasi kemudian melakukan instalasi dan menjalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

2. *Applications Frameworks*

   Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan

service background, mengatur *alarm*, menambahkan status *notificatios*, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi pada kategori inti. Arsitektur aplikasi dirancang supaya pengembang dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*).

Sehingga pengembang bisa menyimpulkan *Applications Frameworks* ini adalah *layer* dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada *layer* inilah aplikasi dapat dirancang, seperti *contont-providers* yang berupa sms dan lain sebagainya.

Komponen-komponen yang termasuk didalam *Applications Frameworks* diantaranya *Views*, *Content Provider*, *Resouce Manager*, *Notification* Manager, dan Activity *Manager.*

3. *Libraries*

   Libraries ini adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi kebanyakan mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas kernel, *layer* ini meliputi berbagai *library* C/C++ inti seperti sebagai libc dan SSL, serta:

   a) *libraries* media untuk pemutaran media audio dan video
   b) *libraries* untuk manajemen tampilan
   c) *libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
   d) *libraries* SQLite untuk dukungan *database*
   e) *libraries* SSL dan *WebKit* terintegrasi dengan *web browser* dan *security*
   f) *libraries* LiveWebcore mencakup *modern web browser* dengan *engine embeded web view*
   g) *libraries* 3D yang mencakup implementasi OpenGL ES 1.0 API's.

4. *Android Run Time*

   Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine*

(DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Didalam Android *Run Time* dibagi menjadi dua bagian yaitu:

a) *Core Libraries*: Aplikasi Android dibangun dalam bahasa java, sementara Dalvik sebagai virtual mesinya bukan Virtual Machine Java, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menerjemahkan bahasa java/c yang dihandle oleh *Core Libraries.*
b) *Dalvik Virtual Machine*: Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat linux kernel untuk *thereading* dan manajemen tingkat rendah.

5. Linux *Kernel*

   Linux *kernel* adalah *layer* dimana inti dari *operating sistem* dari Android itu berbeda. Berisi *file-file system* yang mengatur sistem *processing, memory, resource, drivers,* dan sistem-sistem operating sistem android lainnya. Linux *karnel* yang digunakan android adalah linux *karnel* 2.6.

2.12.2 *Java Development Kit* (JDK)

JDK (*Java Development Kit*) adalah sebuah perangkat peralatan yang digunakan untuk membangun perangkat lunak dengan menggunakan bahasa pemrograman Java. JDK berjalan diatas sebuah *virtual machine* yang dinamakan JVM (*Java Virtual Machine*).

Dokumentasi JDK berisi spesifikasi API, deskripsi fitur, panduan pengembang, referensi halaman untuk perkakas JDK dan utilitas, demo, dan *link* ke informasi terkait. Dokumentasi ini juga tersedia dalam bundel yang dapat diunduh dan diinstal pada komputer [11].

2.12.3 Eclipse *Integrated Development Environment* (Eclipse IDE)

IDE adalah program komputer yang memiliki beberapa fasilitas yang diperlukan dalam pembangunan perangkat lunak. Tujuan dari IDE adalah untuk menyediakan semua utilitas yang diperlukan dalam membangun perangkat lunak.

Eclipse adalah sebuah IDE untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*) [4].

2.12.4 Android *Software Development Kit* (Android SDK)

Android SDK adalah *tools* API (*Aplication Programming Interface*) yang diperlukan untuk memulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware*, dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java.

Beberapa fitur Android yang paling penting diantaranya:

1. *Framework* Aplikasi yang mendukung penggantian komponen dan *reusable*.
2. Mesin *Virtual Dalvik* dioptimalkan untuk perangkat *mobile*
3. *Integrated browser* berdasarkan *engine open source WebKit*
4. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi OpenGL ES 1.0 (Opsional akselerasi *hardware*)
5. SQLite untuk penyimpanan data
6. *Media Support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM *Telephony* (tergantung *hardware*)
7. *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*)
8. Kamera, GPS, kompas, dan *accelerometer* (tergantung *hardware*)
9. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat *emulator*, *tools* untuk *debugging*, profil, dan kinerja memori, dan *plugin* untuk IDE *Eclipse*.

2.12.5 *Android Development Tools* (ADT)

ADT adalah plugin yang didesain untuk IDE *Eclipse* yang memberikan kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE

Eclipse. Dengan menggunakan ADT untuk *Eclipse* akan memudahkan dalam membuat aplikasi *project* android, membuat GUI aplikasi, dan menambahkan komponen-komponen lainnya, selain itu dapat digunakan untuk menjalankan aplikasi menggunakan Android SDK melalui *Eclipse*. Dengan ADT dapat melakukan pembuatan package android (.apk) yang digunakan dalam distribusi apliaksi android.

2.12.6 SQLite

Android memiliki fasilitas untuk membuat *database* yang dikenal dengan SQLite. SQLite adalah salah satu *software* yang bersifat *embedded* yang sangat populer. SQLite memiliki kelebihan yaitu menggunakan kombinasi SQL *interface* dan penggunaan *memory* yang sangat sedikit dengan kecepatan yang sangat cepat. SQLite di android termasuk dalam Android *runtime*, sehingga setiap versi android dapat membuat *database* dengan SQLite.

Untuk membuat dan membuka *database* pada android yang paling baik adalah menggunakan *libraries* Importandroid.database.sqlite.SQLiteOpenHelper yang menyediakan tiga metode yaitu:

1. *Constructor*, menyediakan representasi versi dari *database* dan skema *database* yang digunakan
2. *onCreate(),* menyediakan SQLiteDatabase *object* yang digunakan dalam definisi tabel dan inisialisasi data
3. *onUpgrade(),* menyediakan fasilitas konversi *database* dari *database* versi lama ke *database* versi baru atau sebaliknya.

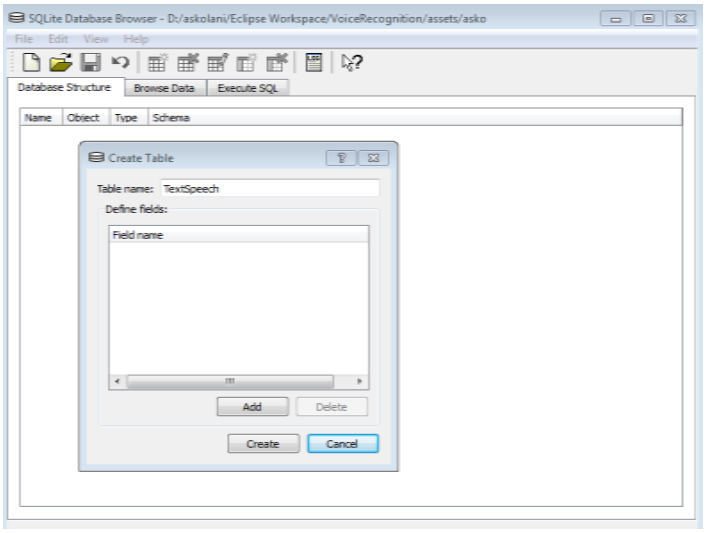
Adapun fitur-fitur yang ada pada SQLite adalah [6]:

1. *Serverless,* SQLite tidak memerlukan proses pada *server* atau sistem untuk menjalankannya, melainkan hanya sebuah *file* yang diakses oleh *library* SQLite.
2. *Cross Platform*, semua instan *database* berada dalam sebuah *file* yang *cross-platform*, tidak memerlukan administrasi.
3. *Zero Configuration,* Tidak ada *server* berarti tidak perlu setup, membuat sebuah *database* instan adalah semudah anda membuat *file* biasa.

4. *Small Runtime Footprint*, untuk membangun *database* SQLite hanya membutuhkan kurang dari satu *megabyte library* (kode program) dan hanya membutuhkan beberapa *megabyte memory*, bahkan dengan beberapa *adjustment* baik ukuran *library* maupun memory dapat diperkecil.
5. *Self-Contained*, sebuah *library* mengandung keseluruhan dari sistem *database*, yang langsung terintegrasi pada sebuah aplikasi program.
6. *Ransactional*, SQLite transaction memperbolehkan aksi penyimpanan melalui beberapa proses *thread*.
7. *Full Featured*, SQLite mensupport hampir sebagian besar standar SQL92 (SQL2).
8. *Highly Reliable*, Tim pengembang SQLite mengembangkan melalui kode program yang sangat serius serta telah melewati proses *testing*.

Eclipse tidak mendukung pembuatan *database* SQlite di dalam fitur-fiturnya. Untuk itu diperlukan *software* pembantu yang dapat membuat sebuah *database* kosong yang nantinya *database* tersebut dapat digunakan oleh Android untuk dimodifikasi atau diakses oleh suatu program Android tertentu.

Contoh *software* pendukung untuk pembuatan *database* SQLite kosong yang dapat digunakan adalah SQLite *Database Browser*. SQLite *Database Browser* merupakan aplikasi *open source* yang dapat memvisualisasikan struktur *database* dan datanya. Tampilan GUI yang *user friendly* membuat pengguna tidak harus mengerti tentang bahasa SQL. Selain itu mudah dalam memodifikasi data yang sangat membantu *programmer*. *Software* ini tersedia pada *platform* Windows, Linux dan Mac OS X. Berikut adalah tampilan antarmuka dari SQLite *Database Browser*:



Gambar 2.12 Tampilan Antarmuka SQLite Database Browser

Pembuatan *database* dilakukan dengan membuat *file* SQLite baru lalu di dalam *file* tersebut diciptakan tabel dengan nama yang nantinya diakses menggunakan *method* yang ada di Eclipse untuk Android.

Berikut adalah langkah-langkah pembuatan *database* SQLite pada Eclipse:

1. Pembuatan Kelas *DataHelper*

   Sebuah kelas yang berfungsi untuk mengatur pembuatan *database* dan *update*-nya.

2. Pembuatan Kelas *DataWrite*

   a) Deklarasi variabel untuk *field* data/kolom, contoh:

   ```
   public static final String KEY_ROWID = "_id";
   public static final String KEY_TRIGGER = "trigger";
   public static final String KEY_OUTPUT = "output";
   public static final String KEY_PROTOKOL = "protokol";
   public static final String KEY_FEEDBACK = "feedback";
   ```

   b) Deklarasi nama *database*, tabel dan versi, contoh:

   ```
   private static final String DATABASE_NAME =
   "textspeechmain";
   private static final String DATABASE_TABLE =
   "textspeech1";
   private static final int DATABASE_VERSION = 1;
   ```

   c) Membuat *adapter database*

   ```
   public DatabaseWrite(Context ctx)
   ```

```
{
  this.context = ctx;
}
```

d) *Inserting* data baru untuk *database*, contoh:

```
public Cursor getById(long rowId) throws SQLException
{
  Cursor mCursor = db.query(true, DATABASE_TABLE, new
String[] {
                    KEY_ROWID,
                    KEY_INPUT,
                    KEY_OUTPUT,
                    KEY_KAMERA,
                    KEY_KECEPATAN,
                    KEY_JARAK,
                    KEY_LENGAN},
                    KEY_ROWID + "=" + rowId,
                    null,
                    null,
                    null,
                    null,
                    null);
        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
```

e) Membuat *method* untuk membuka dan menutup *database*

```
public DatabaseWrite open() throws SQLException
{
  DBHelper = new DatabaseHelper(context);
  db = DBHelper.getWritableDatabase();
    return this;
}
public void close()
{
  DBHelper.close();
}
```

f) Menghapus Data

```
public boolean deleteById(long rowId)
{
  return db.delete(DATABASE_TABLE, KEY_ROWID +
            "=" + rowId, null) > 0;
}
```