BAB II

TINJAUAN PUSTAKA

II.1. Perancangan

Menurut Al-Bahra (2005:39), perancangan adalah suatu tahapan yang memiliki tujuan untuk mendesign sistem baru yang dapat menyelesaikan masalah – masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik. Kegiatan yang dilakukan dalam tahap perancangan ini meliputi perancangan *output*, *input* dan *file*.

Sedangkan definisi perancangan menurut John Burch dan Gary Grudnitski yang telah diterjemahkan oleh Jogiyanto Hartono (2005:196) dalam buku yang berjudul *Analisis dan Desain Sistem Informasi* menyebutkan sebagai tahapan setelah analisa siklus pengembangan sistem, pendefinisian dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi, serta menggambarkan bagaimana suatu sistem dibentuk.

Berdasarkan penjelasan di atas penulis dapat mengambil kesimpulan bahwa perancangan merupakan kegiatan mendesain sistem baru yang bertujuan untuk menyelesaikan masalah yang dihadapi perusahaan atau suatu kegiatan yang memiliki tujuan untuk mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik.

II.2. Aplikasi

Aplikasi berasal dari kata *application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju. (www.totalinfo.or.id)

Perangkat lunak aplikasi adalah suatu sub kelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna.(Prajna : 2008)

Menurut Dhanta (2009 : 32), aplikasi (*application*) adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya *Microsoft Word, Microsoft Excel*.

Dari 2 (dua) definisi aplikasi di atas dapat penulis simpulkan pengertian aplikasi adalah suatu program (*software*) yang ditulis atau dirancang untuk menangani masalah tertentu.

II.3. *Game*

Game adalah media untuk melakukan aktifitas bermain. Aktifitas bermain merupakan suatu aktifitas yang meliputi pemecahan masalah yang menjadi tantangan dari *game* tersebut, dengan mengikuti suatu aturan tertentu. (Dadang : 2010)

Game juga dapat diartikan sebuah permainan *interactive* yang membutuhkan komputer untuk bermain. Program komputer menerima *input* dari si pemain melalui pengendali dan menampilkan lingkungan buatan melalui TV atau layar monitor. (Katharina : 2010 : 2)

Penulis menyimpulkan bahwa *game* atau permainan biasanya dilakukan untuk kesenangan dan kadang – kadang digunakan sebagai alat pendidikan. Untuk membuat sebuah *game*, terlebih dahulu pembuat game harus membuat deskripsi yang menceritakan *game* yang akan dibuat. Selain itu dibutuhkan juga design game yang sederhana untuk mempermudah pembuatan *game*. Dari *design* yang telah dibuat dapat diketahui semua elemen – elemen yang dibutuhkan dalam pembuatan *game*, misalnya karakter *user*, karakter musuh, animasi serangan dan sebagainya.

II.3.1. Jenis-Jenis Games

Menurut Marc Saltzman (2000,p1) ada beberapa jenis *game* seperti :

a. *Action Game*, secara umum lebih bergantung pada koordinasi tangan atau mata dibandingkan cerita atau strategi. Pada umunya mengandalkan ketangkasan dan refleks.

b. *Strategy*, *game* strategi menekankan pada pemikiran dan perencanaan logis. *Game* strategi cenderung menitikberatkan pada manajemen sumber daya dan waktu yang biasanya didahulukan sebelum respon cepat dan keterlibatan karakter. Perencanaan dan eksekusi taktis sangat penting, dan pencipta *game* biasanya menempatkan kemampuan pembuatan keputusan dan pengiriman perintah ke tangan pemain. Bertentangan dengan *game* bersifat *turn-base* seperti *Civilization* buatan *Microprose* atau *Heroes of Might* and *Magic* buatan 3DO, *game real-time*. *Strategy* menambahkan elemen aktif dan memaksa pemain untuk merespon beberapa kejadian yang terjadi dalam waktu yang hampir bersamaan. Contohnya *Starcraft* buatan *Blizzard* dan *Age of Empire* buatan *Ensemble Studios*.

c. *Adventure*, melibatkan perjalanan dan ekspedisi dari sebuah eksplorasi dan pemecahan teka-teki. *Game* seperti ini biasanya mempunyai jalan cerita yang linier dimana pemain sebagai *protagonist* harus menyelesaikan sebuah tujuan utama melewati interaksi karakter dan manipulasi inventaris.

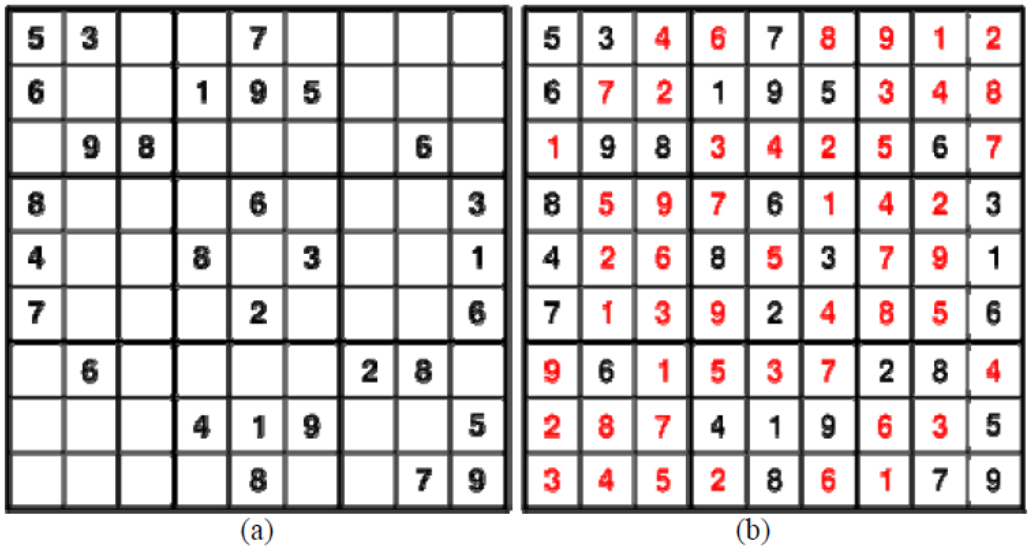
d. *Role Playing Games* (RPGs), mirip dengan *adventure game*, tetapi lebih bergantung pada pembangunan dan pengembangan karakter (biasanya disertai dengan statistik pemain), percakapan, dan strategi bertempur dibanding pemecahan teka-teki. Dunia fantasi yang luas dan *epic quest* dengan NPCs (*non-player characters*) merupakan sesuatu yang umum, dan jalan cerita tidak selalu linier seperti *adventure game*. *Side quest* merupakan hal yang tidak langka bagi RPGs.

e. *Sports*, menstimulasi sebuah permainan perorangan atau kelompok dari sudut pandang instruksional atau pemain. Realita merupakan sesuatu yang paling penting, sama seperti ketangkasan dan strategi
f. *Simulations* atau *Sims*, secara nyata menstimulasikan sebuah objek atau proses yang dianimasikan maupun tidak. Sebagian besar, *Sims* menempatkan pemain pada sudut pandang 3D orang pertama, atau menciptakan kembali mesin-mesin seperti pesawat, *tank*, helikopter, dan kapal selam.
g. *Puzzle* atau *"Classic" Games*, mencakup permainan-permainan yang lebih tua, bersejarah seperti kartu, permainan papan, *trivia*, atau kata-kata. (M.Iwan Januar & E.F Turmudzi : 2006 : 8)

II.4. Game Sudoku

Sudoku modern diterbitkan pertama kali dalam Dell Magazine pada tahun 1979. Teka-teki ini dibuat oleh Howard Garns, seorang pensiunan arsitek dan perancang teka-teki dari Amerika Serikat. Meskipun, demikian, cikal bakal sudoku sebenarnya telah muncul pada akhir abad ke-19 di Perancis. Pada bulan April 1984, teka-teki ini diperkenalkan ke Jepang dan mendapatkan nama "*sudoku*". Nama ini berasal dari kata "*Suuji wa dokushin ni kagiru*" yang dapat diterjemahkan menjadi angka harus sendiri atau muncul sekali. Pada April 2005, New York Post menerbitkan teka-teki sudoku sebagai fitur secara teratur dan mulai sekitar bulan Juli 2005, sudoku mencapai kepopuleran di Amerika dan muncul sebagai acara *live* di televisi. Di Indonesia sendiri, sudoku muncul dalam bentuk buku yang cukup digemari.

Bentuk sudoku yang paling umum berukuran 9 x 9 kisi dan terbagi menjadi 3 x 3 kotak yang disebut *subgrid*. Secara tradisional, tujuan permainan ini adalah mengisi semua kotak – kotak kecil dalam kolom, baris dan *subgrid* dengan angka satu sampai sembilan sedemikian sehingga dalam satu deret angka hanya muncul sekali saja demikian juga dalam satu kolom dan satu *subgrid*. Karena berhubungan dengan angka permainan ini terlihat lebih matematis. Contoh sudoku dengan angka dapat dilihat dalam gambar berikut ini :



Gambar II.1. Contoh sudoku (a) dan sudoku lengkap (telah diselesaikan) (b)
Sumber : Sukisman, Seminar Nasional Kimia dan Pendidikan Kimia 2007

Ukuran dan bentuk kolom dapat bervariasi. Dalam sudoku yang belum dipecahkan, beberapa angka diletakkan dalam kotak sebagai titik awal atau titik kunci. Angka ini dapat diletakkan di sembarang tempat. Pemain harus mengisi kotak-kotak kosong untuk melengkapi sudoku. Selain angka, kotak juga dapat diisi dengan warna, huruf, atau bentuk. Sudoku termudah memerlukan logika yang sederhana atau dipecahkan dengan coba-coba, sementara sudoku yang

semakin meningkat kesulitannya memerlukan ketelitian, analisis multilangkah, dan pemecahan masalah. (Sukisman : 2007)

II.5. Sekilas Tentang Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi. *Android* menyediakan *platform* yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, *Google Inc*. membeli *Android Inc*. yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel atau *smartphone*. Kemudian untuk mengembangkan *android,* dibentuk *Open Handset Alliance,* konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcom, T-mobile* dan *Nvidia*.

Pada saat perintisan perdana *Android,* 5 November 2007, *android* bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat mobile, dilain pihak, *google* meliris 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* (GSM) dan kedua adalah yang benar-benar bebas distribusinya tampa dukungan langsung *google* atau dikenal dengan *Open Handset Disrtribution* (OHD).

Sekitar September 2007 *Google* mengenalkan *Nexus One,* salah satu jenis *smartphone* yang menggunakan *android* sebagai sistem operasinya. Telepon seluler ini diproduksi oleh *HTC Corporation* dan tersedia dipasaran pada 5 januari

2008. Pada 9 desember 2008, diumumkan anggota baru yang bergabung dalam program kerja *android ARM Holdings, Atyheros Communications*, diproduksi oleh *Asustek Computer Inc, garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp* dan *Vodafone Group Plc*. Seiring pembentukan *Open Handset Allience*, OHA mengumumkan produk *android* perdana mereka, perangkat *mobile* yang merupakan modifikasi kernel *Linux* 2.6. Sejak *android* diliris telah dilakukan berbagai pembaruan berupa perbaikan *bug* dan penambahan fitur baru.

Pada masa saat ini kebanyakan *vendor-vendor smartphone* sudah memproduksi *smartphone* berbasis *android, vendor-vendor* ini antara lain *HTC, Motorola, Samsung, LG, HKC, Huawei, Archos, Webstation Camangi, Dell, Nexus, SciPhone, Wayteq, Soni Ericsson, LG, Acer, Philips, T-Mobile, Nexian, IMO*, Asus dan masih banya lagi *vendor smartphone* yang memproduksi *android*. Hal ini dikarenakan android itu adalah sistem operasi yang *open source* sehingga bebas didistribusikanb dan dipakai oleh *vendor* manapun.

Tidak hanya menjadi sistem operasi di *smartphone*, saat ini *android* menjadi pesaing utama dari Apple pada sistem Operasi *Tablet PC*, pesatnya pertumbuhan android selain faktor yang disebutkan diatas adalah karena *android* itu sendiri adalah *platform* yang sangat lengkap baik itu sistem operasinya, Aplikasi dan *Tool* pengembangan, *market* aplikasi *android* serta dukungna yang tinggi dari komunitas *Open source* di dunia, sehingga *android* terus berkembang pesat baik dari segi teknologi maupun dari segi *device* yang ada di dunia. *(Nazruddin Safaat H: 2011:1).*

Adapun logo *android* dapat pada gambar II.2. berikut ini :

Gambar II.2. Logo Android
Sumber : Nazruddin Safaat H (2011:1)

II.5.1. Versi Android

Menurut Wahana Komputer (2012,2), telepon pertama yang memakai sistem operasi android adalah *HTC Dream*, yang diliris pada 22 Oktober 2008. Pada Penghujung tahun 2010 diperkirakan Hampir semua *vendor* seluler didunia manggunakan *android* sebagai *operating system*. Adapun versi-versi *android* yang pernah diliris adalah sebagai berikut :

1. Android versi 1.1

   Pada 9 Maret 2009, *Google* merilis *Android* versi 1.1. *Android* versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan *Gmail*, dan pemberitahuan *email*. Adapun logo *Android* Versi 1.1 dapat pada gambar II.2.

Gambar II.3. Android Versi 1.1
Sumber: Wahana Komputer (2012:2)

2. Android versi 1.5 (*Cupcake*)

Android 1.5 dirilis pada tanggal 30 April 2009 yang memiliki kode nama *Cupcake*. Terdapat beberapa pembaruan, termasuk juga penambahan beberapa fitur, yakni kemampuan merekam dan menonton video, mengunggah video ke *Youtube* dan gambar *Picasa* langsung dari telepon seluler, dukungan Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan system. Adapun logo *Android* Versi 1.5 dapat pada gambar II.3.

Gambar II.4. Android Versi 1.5 (*Cupcake*)
Sumber: Wahana Komputer (2012:3)

3. Android versi 1.6 (*Donut*)

Donut (versi 1.6) dirilis pada tanggal 15 September 2009 dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan *control applet* VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, *camcorder* dan *galeri* yang dintegrasikan, CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text-to-speech engine*, kemampuan dial kontak, teknologi *text to change speech* (tidak tersedia pada semua ponsel) pengadaan resolusi VWGA. Adapun logo *Android* Versi 1.6 dapat pada gambar II.4.

Gambar II.5 Android Versi 1.6 (Donut)
Sumber: Wahana Komputer (2012:3)

4. Android versi 2.0/2.1 (*Eclair*)

Pada 3 Desember 2009 kembali diluncurkan ponsel *Android* dengan versi 2.0/2.1 (*Eclair*), perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan *Google Maps* 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, *digital Zoom*, dan *Bluetooth* 2.1. Adapun logo *Android* Versi 2.0/2.1 *(Éclair)* dapat pada gambar II.6.

Gambar II.6 Android Versi 2.0/2.1 (Eclair)
Sumber: Wahana Komputer (2012:4)

5. Android versi 2.2 (*Froyo: Frozen Yoghurt*)

Pada 20 Mei 2010, *Android* versi 2.2 (*Froyo*) diluncurkan. *Android* ini yang sekarang banya beredar dipasaran, salah satunya adalah dipakai di Samsung FX tab yang sudah ada dipasaran. Fiktur yang tersedia di *android* versi ini sudah kompleks diantaranya adalah:

a. Kerangka aplikasi memungkinkan penggunaan dan penghapusan komponen yang tersedia.
b. *Dalvik Virtual Machine* dioptimalkan untuk perangkat *mobile*.
c. Grafik 2D dan 3D berdasarkan *libraries OpenGL*.
d. *SQLite* untuk penyimpanan data.
e. Mendukung media *audio, video,* dan berbagai *format* gambar.
f. GSM, *Bluetooth, EDGE,* 3D dan *WiFi*

g. Kamera, *Global Positioning System (GPS)*, kompas dan *accelerometer* (tergantung *hardware*).

Adapun logo *Android* Versi 2.2 (*Froyo*: *Frozen yoghurt*) dapat pada gambar II.7.

Gambar II.7 Android Versi 2.7 (Froyo: Frozen yoghurt)
Sumber: Wahana Komputer (2012:4)

6. Android versi 2.3 (*Gingerbread*)

Pada 6 Desember 2010, *Android* versi 2.3 (*Gingerbread*) diluncurkan. Perubahan-perubahan umum yang didapat dari *Android* versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antar muka (*User Interface*) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (*reverb, equalization, headphone virtualization,* dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC), dan dukungan jumlah kamera yang lebih dari satu. Adapun logo *Android* Versi 2.3 (*Gingerbread*) dapat pada gambar II.8.

Gambar II.8. Android Versi 2.3 (*Gingerbread*)
Sumber: Wahana Komputer (2012:5)

7. Android versi 3.0 (*Honeycomb*)

Android Honeycomb dirancang khusus untuk komputer tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk *tablet*. *Honeycomb* juga mendukung *multi prosesor* dan juga akselerasi perangkat keras (*hardware*) untuk *grafis*. Tablet pertama yang dibuat dengan menjalankan *Honeycomb* adalah *Motorola Xoom*. Perangkat komputer *tablet* dengan *platform Android* 3.0 akan segera hadir di Indonesia. Perangkat tersebut bernama *Eee Pad Transformer* produksi dari *Asus*. Rencana masuk pasar Indonesia pada Mei 2011. Adapun logo *Android* Versi *3.0 (Honeycomb)* dapat pada gambar II.9.

Gambar II.9. Android Versi 3.0 (*Honeycomb*)
Sumber: Wahana Komputer (2012:6)

8. Android versi 4.0 (*Ice Cream Sandwich*)

Versi dirilis pada 19 Oktober 2011. *Ice Cream*, tentu saja kita tahu karena ini adalah minuman atau tepatnya makanan yang sangat disukai terutama oleh anak kecil .*Ice Cream* dipakai sebagai nama alias dari Android versi 4.0. *Smartphone* yang pertama kali mengunakan OS Android ini adalah Samsung Galaxy Nexus. Secara teori semua perangkat seluler yang menggunakan versi Android sebelumnya, *Gingerbread*, dapat di-*update* ke Android *Ice Cream Sandwich*. Namun sayangnya sampai saat ini ke banyak *smartphone* yang menggunakan Android ICS merupakan *smartphone* kelas

high-end yang dijual dengan harga cukup mahal. Mungkin karena alasan inilah distribusi versi Android satu ini tidak lebih dari 8% sampai pertengahn tahun 2012 ini. Adapun logo versi 4.0 (*Ice Cream Sandwich*) dapat pada gambar II.10.

Gambar II.10. Android versi 4.0 (*Ice Cream Sandwich*)
Sumber: Wahana Komputer (2012:6)

9. Android versi 4.1 (*Jelly Bean*)

Sampai tulisan ini di tulis Versi Android *Jelly Bean* adalah versi Android yang terbaru. Android *Jelly Bean* yang diluncurkan pada acara *Google I/O* lalu membawa sejumlah keunggulan dan fitur baru. Penambahan baru diantaranya meningkatkan *input keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Tak ketinggalan *Google Now* juga menjadi bagian yang diperbarui. *Google Now* memberikan informasi yang tepat pada waktu yang tepat pula. Salah satu kemampuannya adalah dapat mengetahui informasi cuaca, lalu-lintas, ataupun hasil pertandingan olahraga. Sistem operasi Android *Jelly Bean* 4.1 muncul pertama kali dalam produk tablet Asus, yakni Google Nexus 7. Adapun logo versi 4.1 (*Jelly Bean*) dapat pada gambar II.11.

Gambar II.11. Android versi 4.1 (*Jelly Bean*)
Sumber: Wahana Komputer (2012:6)

II.6. Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform-independent*). *Eclipse* awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM *Visual Age for Java* 4.0. Produk ini diluncurkan oleh IBM pada tanggal 5 November 2001, yang menginvestasikan sebanyak US$ 40 juta untuk pengembangannya. Semenjak itu *konsursium Eclipse Foundation* mengambil alih untuk pengembangan *Eclipse* lebih lanjut dan pengaturan organisasinya (Anang Triyono: 2012: 47).

Eclipse hanya dibutuhkan untuk mengembangkan aplikasi dengan bantuan sebuah *plugin Eclipse* yang bernama *Android Development Tools* (ADT). Versi yang direkomendasikan adalah "*Eclipse for Java Developers*" atau "*Eclipse for RCP Developers*". Untuk versi 3.5, versi yang direkomendasikan adalah "*Eclipse Classic*". *Eclipse* harus dipastikan memiliki Java *Development Tools* (JDT), biasanya JDT sudah terdapat pada kebanyakan paket *Eclipse* adapun logo *Eclipse* dapat pada gambar II.12.

Gambar II.12. Logo Eclipse
Sumber : Ivan Michael Siregar (2011: 2)

Berikut ini adalah sifat dari *Eclipse*:

1. *Multi-platform*

   Target sistem operasi *Eclipse* adalah *Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.*

2. *Mulit-language*

   Eclipse dikembangkan dengan bahasa pemrograman *Java*, akan tetapi *Eclipse* mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, *Cobol*, *Python*, Perl, PHP, dan lain sebagainya.

3. *Multi-role*

   Selain sebagai IDE untuk pengembangan aplikasi, *Eclipse* pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya (Anang Triyono: 2012: 47).

II.7. *SQLite Database Server*

Untuk struktur data yang lebih kompleks, database bias menjanjikan akses yang lebih fleksibel dibandingkan dengan file flat atau shared preferences. Dan

untunglah, android menyediakan database *built* in yang disebut *SQLite.* Database ini menyediakan kemampuan RDBMS menggunakan standar command *SQL*. Tiap aplikasi yang menggunakan *SQLite* memiliki *instance* database sendiri. Yang secara *default* hanya bisa diakses dari aplikasi itu sendiri. Database ini disimpan di data/data/nama_paket/database pada peranti *android* (Edi Winarno ST: 2011: 184).

II.7.1. Mengenal *SQLite*

SQLite merupakan database *Open Source* yang di-*embed* ke *Android. SQLite* mendukung fitur database RDBMS standar, seperti sintaks SQL, transaksi, dan *prepared statement.* Walapun demikian, *SQLite* hanya membutuhkan sedikit *memory* saat *runtime*, yaitu sekitar 250 *kByte*. *SQLite* mendukung tipe data berikut :

1. *TEXT* yang identik dengan *String* di *java*
2. *INTEGER* yang identik dengan *long* di *java*
3. *REAL* yang identik dengan *double* di *java*

Semua tipe lain harus dikonversi ke salah satu tipe *field* ini untuk dikonversi ke database. *SQLite* sendiri tidak memvalidasi apakah tipe yang ditulis ke kolom sesuai dengan tipe yang didefinisikan. Anda bisa menuliskan *integer* ke kolom *string. SQLite* tersedia ditiap peranti *android* mengunakan database *SQLite* di android tidak memerlukan setup database atau administrasi tertentu. Anda tinggal menentukan sql yang akan dipakai untuk bekerja dengan database, maka database sudah siap untuk langsung digunakan (Edi Winarno ST: 2011: 185).

Beberapa fitur yang ada pada *SQLite*, antara lain :

1. *Serverless*, artinya *SQLite* tidak memerlukan proses pada *server* melainkan hanya sebuah *file* yang diakses oleh *library SQLite.*
2. *Zero Configuration*, artinya ketika membuat sebuah *database* seperti membuat *file* biasa.
3. *Cross-platfrom*, artinya semua *database* berada dalam sebuah *file cross-platform* dan tidak memerlukan administrasi.
4. *Sel-contained*, artinya terdapat *library* yang mengandung keseluruhan *database* dan langsung terintegrasi pada aplikasi program.
5. *Transactional*, artinya *SQLite* memperboleh aksi penyimpanan melalui beberapa proses *thread*.
6. *Full featured*, artinya *SQLite* mendukung sebagian besar standar SQL92 (SQL2).
7. *Highly reliable*, artinya tim pengembang *SQLite* telah mengembangkan dengan proses yang serius dan *testing* yang ketat. (Edi Winarno ST: 2011: 186).

II.8. Algoritma Genetika

Algoritma Genetika atau *Genetic Algorithm* (GA) sebagai cabang dari Algoritma Evolusi merupakan metode yang digunakan untuk memecahkan suatu pencarian nilai dalam sebuah masalah optimasi yaitu permasalahan – permasalahan yang tak linier. Algoritma ini pertama kali diperkenalkan oleh John Holland pada tahun 1975 sebagai *whitepaper* di Universitas Michigan yang

didasarkan pada proses genetik pada makhluk hidup. John Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika. Dengan meniru teori evolusi, Algoritma Genetika bekerja dengan sebuah populasi dari individu-individu yang masing-masing individunya merepresentasikan sebuah solusi bagi persoalan yang ada. GA telah diaplikasikan untuk berbagai permasalahan seperti bioinformatika, ilmu komputer, teknik, ekonomi, matematika, fisika, dan bidang lainnya. (Bhakti Yudho Suprapto dan Sariman : 2012).

Algoritma genetika adalah suatu algoritma pencarian yang berbasis pada mekanisme seleksi alam dan genetika. Algoritma genetika merupakan salah satu algoritma yang sangat tepat digunakan dalam menyelesaikan masalah optimasi kompleks, yang sulit dilakukan oleh metode konvensional (Anies Hannawati, Jurnal Teknik Elektro Vol.2, No. 2, September 2002).

Sifat algoritma genetika adalah mencari kemungkinan-kemungkinan dari calon solusi untuk mendapatkan yang optimal bagi penyelesaian masalah. Ruang cakupan dari semua solusi yang layak (*feasible*), yaitu obyek-obyek di antara solusi yang sesuai, dinamakan ruang pencarian (*search space*). Tiap titik dalam ruang pencarian mempresenttasikan satu solusi yang layak. Tiap solusi yang layak dapat ditandai dengan nilai *fitness*-nya bagi masalah.

Algoritma genetika bekerja dari populasi yang merupakan himpunan solusi yang dihasilkan secara acak. Setiap anggota yang mempresentasikan satu solusi masalah dinamakan kromosom. Kromosom dalam suatu populasi berevolusi dalam iteritas yang dinamakan generasi, tiap kromosom dievaluasi

berdasarkan fungsi evaluasi (*fitness function*). Pada algoritma genetika, *fitness* biasanya dapat berupa fungsi objektif dari masalah yang akan dioptimalisasi (Suyanto: 2005: 1).

II.8.1. Struktur Umum Algoritma Genetika

Pada Algoritma Genetika, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi disebut dengan istilah kromosom. Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen (kromosom) disebut dengan *genotype* (gen). Dalam Algoritma Genetika gen ini bisa berupa nilai biner, float, integer maupun karakter atau kombinatorial. Sedangkan nilai dari gen tersebut disebut dengan allele. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan generasi.

Pada setiap generasi, kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut dengan fungsi *fitness*. Nilai *fitness* dari suatu kromosom akan menunjukkan kualitas dari kromosom dalam populasi tersebut. Generasi berikutnya dikenal dengan istilah anak (*offspring*) terbentuk dari gabungan dua kromosom generasi sekarang yang bertindak sebagai induk (parent) dengan menggunakan operator penyilangan (*crossover*) dan operator mutasi. Populasi generasi yang baru dibentuk dengan cara menyeleksi nilai fitness dari kromosom induk (*parent*) dan nilai fitness dari kromosom anak (*offspring*), serta menolak kromosom-kromosom yang lainnya sehingga ukuran populasi (jumlah

kromosom dalam suatu populasi) konstan. Setelah melalui beberapa generasi, maka algoritma ini akan konvergen ke kromosom terbaik (Bhakti Yudho Suprapto dan Sariman : 2012).

Umumnya, Algoritma Genetika memiliki lima komponen dasar menurut Michalewicz, yaitu :

1. Representasi genetika untuk solusi-solusi suatu masalah
2. Cara menentukan inisialisasi populasi awal
3. Penentuan fungsi evaluasi berdasarkan nilai fitness nya
4. Operator genetika yang mengubah gen-gen anak selama proses reproduksi

Representasi mengacu pada cara solusi untuk masalah tertentu yang dikodekan menjadi struktur data yang dapat diproses oleh komputer digital. Teknik pengkodean meliputi pengkodean gen dari kromosom, dimana gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variabel. Dengan demikian kromosom dapat direpresentasikan dengan menggunakan :

a. *String bit* : 10011, 01101, 11110 dan sebagainya
b. Bilangan real : 65.65 ; 45.78 ; 345.23 dan sebagainya
c. Elemen permutasi : E2, E10, E5 dan sebagainya
d. Daftar aturan : R1, R2, R3 dan sebagainya

II.9. Alat Bantu Perancangan Sistem

Adapun alat bantu yang digunakan dalam perancangan atau pembangunan sistem yang digunakan dalam penelitian umumnya berupa gambaran atau diagram yang tertera dan dijelaskan di bawah ini.

II.9.1. *Unified Modelling Language (UML)*

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML dapat dibuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek seperti *C++, Java, atau VB. NET* (Prastuti Sulistyorini : 2009).

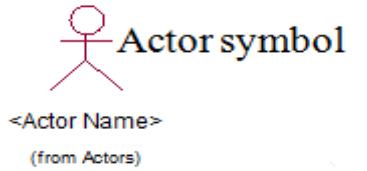
Use Case adalah deskripsi dari sebuah sistem dari perspektif pengguna. *Use Case* bekerja dengan cara mendeskripsikan *tipikal* interaksi antar *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai". *Use Case Diagram* menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana".

Sebuah *Use Case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use Case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah *entitas* manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use Case Diagram* dapat sangat membantu apabila kita sedang menyusun *requirement* sebuah sistem,

mengkomunikasikan rancangan dengan *client*, dan merancang *test case* untuk semua *feature* yang ada pada *system.*

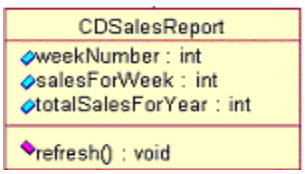
Untuk menggambarkan analisa dan desain diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, behaviour diagram dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

1. *Actor*, menentukan peran yang dimainkan oleh *user* atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya. Tugas *actor* adalah memberikan informasi kepada sistem dan dapat memerintahkan sistem untuk melakukan sesuatu tugas.



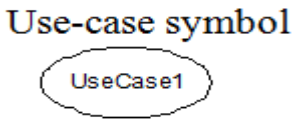
Gambar II.13. Notasi *Actor*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

2. *Class diagram*, notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek.



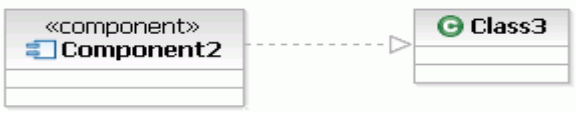
Gambar II.14. Notasi *Class*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

3. *Use Case* dan *use case specification*, *use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. *Use case* merupakan awal yang sangat baik untuk setiap fase pengembangan berbasis objek, *design*, *testing*, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang di luar sistem. Perlu diingat bahwa *use case* hanya menetapkan apa yang seharusnya dikerjakan oleh sistem, yaitu kebutuhan fungsional sistem dan tidak untuk menentukan kebutuhan non-fungsional, misalnya: sasaran kinerja, bahasa pemrograman dan lain sebagainya.



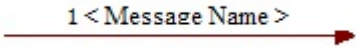
Gambar II.15. Notasi *Use Case*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

4. *Realization*, menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen yang ada di bagian dengan panah.



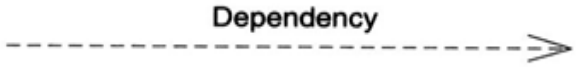
Gambar II.16. Notasi *Relaization*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

5. *Interaction*, digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.



Gambar II.17. Notasi *Interaction*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

6. *Dependency*, merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan ke dalam elemen yang ada di garis dengan panah.

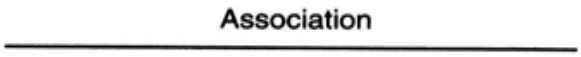


Gambar II.18. Notasi *Dependancy*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

7. *Note*, digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. *Note* ini bisa disertakan ke semua elemen notasi yang lain.

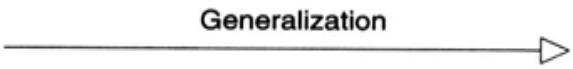
Gambar II.19. Notasi *Note*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

8. *Association*, menggambarkan navigasi antar *class* (*navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *class*) dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*).



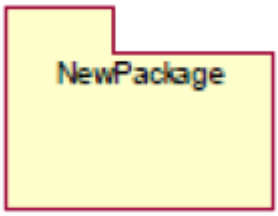
Gambar II.20. Notasi *Association*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

9. Generalization, menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.



Gambar II.21. Notasi *Generalization*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

10. *Package*, adalah mekanisme pengelompokkan yang dgunakan untuk menandakan pengelompokkan elemen-elemen model.



Gambar II.22. Notasi *Package*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

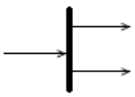
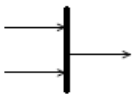
11. *Interface*, merupakan kumpulan operasi berupa implementasi dari suatu *class*. Atau dengan kata lain implementasi operasi dalam *interface* dijabarkan oleh operasi di dalam *class*.

Gambar II.23. Notasi *Interface*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

II.9.2. *Activity Diagram*

Activity Diagram adalah teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa (Prastuti Sulistyorini : 2009).

Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity diagram* yaitu :

Simbol	Keterangan
	Start Point
	End Point
	Activities
	Fork (Percabangan)
	Join (Penggabungan)
	Decision/Split Merge
Swimlane	Sebuah cara untuk mengelompokkan activity berdasarkan Actor (mengelompokkan activity dalam sebuah urutan yang sama)

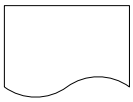
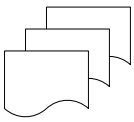
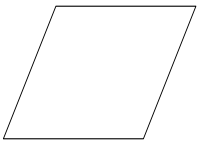
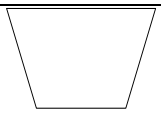
Gambar II.24. Simbol *Activity Diagram*
Sumber : http://kk.mercubuana.ac.id/files/15024-5-600173869778.pdf

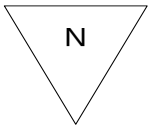
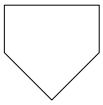
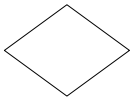
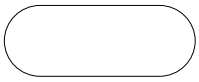
II.10. *Sytem Flowchart*

System flowchart merupakan bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan ini menjelaskan urut-urutan dari prosedur-prosedur yang ada di dalam sistem. Prosedur-prosedur yang ada akan di gambarkan melalui simbol-simbol tertentu, dan setiap simbol memiliki arti yang berbeda. Urutan prosedur yang digambarkan pada *flowchart*, digambarkan dari atas ke bawah. Untuk sebuah *flowchart* harus memiliki *start* dan *end*. (Arfeny Oktantia Mariena : 2011)

Di bawah ini terdapat tabel simbol – simbol bagan alir dan penjelasan tentang penggunaannya.

Tabel II.1. Simbol – Simbol Bagan Alir (*Flowchart*)

Simbol	Nama	Keterangan
	Dokumen	Dokumen tersebut dapat dipersiapkan dengan tulisan tangan atau dicatat dengan komputer
	Beberapa tembusan dari satu dokumen	Digambarkan dengan cara menumpuk simbol dokumen dan mencetak nomor dokumen di bagian depan sudut kanan atas
	*Input*/*Output*	Fungsi *input* dan *output* apapun di dalam bagan alir program, juga digunakan dalam mewakili jurnal dan buku besar dalam bagan alir dokumen
	Pemrosesan dengan komputer	Biasanya menghasilkan perubahan atas data atau informasi.
	Proses manual	Pelaksanaan pemrosesan yang dilakukan secara manual

	*File*	*File* dokumen secara manual disimpan. Huruf yang ditulis di dalam simbol menunjukkan urutan pengaturan *file* secara N = numeris, A = Alfabetis, D = Tanggal
	Arus dokumen atau proses	Arah pemrosesan atau arus dokumen
	*Off-page connector*	Suatu penanda masuk dari atau keluar ke halaman lain
	Keputusan	Langkah pengambilan keputusan
	*Terminal*	Titik awal, akhir atau pemberhentian dalam suatu proses.

Sumber : http://www.unhas.ac.id/rhiza/arsip/kuliah/Arsitektur-Komputer/sist%20dan%20analisis%20sist/Microsoft_Word_-_Modul_6_APSI_-_Flow_Chart.pdf