BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori dari penelitian ini menguraikan proses analisis sistem serta mendukung proses pembangunan *Home Automation* pada *Lighting control* dengan Android.

2.1.1 *Home Automation*

Home Automation dapat dikatakan sebagai ekstensi perumahan dari "otomatis bangunan/*building automation*". Hal ini berhubungan dengan kegiatan dan pekerjaan di rumah yang otomatis terlaksana. *Home Automation* mungkin termasuk *control* yang terpusat pada pencahayaan, HVAC (pemanas, ventilasi dan AC), peralatan, dan *system* lainnya, untuk memberikan kenyamanan, keamanan, efisiensi energi dan kemudahan yang lebih baik.

Sebuah *system* otomatisasi rumah mampu mengintegrasikan perangkat listrik di rumah dengan satu sama lainnya. Teknik-teknik yang digunakan dalam *Home Automation* termasuk yang diotomatisasi bangunan dengan pengendalian kegiatan domestic seperti *system* hiburan rumah, *houseplant* dan penyiraman halaman dan lainya. Perangkat dapat dihubungkan melalui jaringan Komputer untuk memungkinkan *control* dengan komputer pribadi dan memungkinkan akses *remote* dari internet. Melalui integrasi teknologi informasi dengan lingkungan rumah, sistem dan peralatan dapat berkomunikasi secara terpadu yang menghasilkan kenyamanan, efisiensi energi dan manfaat keselamatan[1].

2.1.2 Mikrokontroler Arduino

Arduino adalah kit elektronik atau papan rangkaian elektronik *Open source* yang di dalamnya terdapat komponen utama, yaitu sebuah chip mikrokontroler dengan jenis *Advanced Versatile RISC* (AVR) dari perusahaan Atmel. Mikrokontroler itu sendiri adalah chip atau *intergrated circuit* (IC) yang bisa di program menggunakan komputer. Tujuan menanamakan program pada mikrokontroler adalah agar rangkaian elektronik dapat membaca *input*,

memproses *input* tersebut dan kemudian menghasilkan *output* sesuai dengan yang di inginkan. Jadi mikrokontroler sebagai "otak" *input*, *output* dan proses pada sebuah rangkaian elektronik.

Secara umum, Arduino terbagai dari dua bagian,yaitu:

1. *Hadware* berupa papan *input*/*output* (I/O) yang *Open source*.
2. *Software* arduino yang juga *Open source*, meliputi *software* arduino IDE untuk menulis program dan driver untuk koneksi dengan komputer.

2.1.2.1 Sejarah Arduino

Pembuatan Arduino di mulai pada tahun 2005, dimana sebuah situs perusahaan komputer *Oliveti* di *Ivrea* Italia, membuat perangakat untuk mengedalikan proyek desain interaksi siswa supaya lebih murah di bandingkan dengan sistem pada saat itu. Dilanjutkan pada bulan mei 2011, diamana sudah lebih dari 300.000 unit arduino terjual. Pendiri dari arduino itu sendiri adalah *Massimo Banzi* dan *David Cuarttielles* sebagai *Founder*. Awalnya mereka memberi nama proyek itu dengan sebutan arduin dari *Ivrea* tetapi seiring dengan perkembangan zaman, nama proyek itu di ubah menjadi Arduino yang berarti "teman yang kuat" atau dalam versi inggrisnya dikenal dengan sebutan "*Hardwin*".Proyek pengkabelan diciptakan oleh seniman sekaligus *programmer* asal kolombia bernama *Hernando Barragan*.

Pengkabelan ini adalah proyek tesis *Hernando* pada Desain Interaksi *Institute Ivrea*. Hal tersebut dimaksudkan untuk menjadi versi elektronik pengolahan yang digunakan di lingkungan pemrograman dan mengambil pola sintaks *Processing*. Dengan berkembangnya teknologi, Arduino menjadi sangat popular di kalangan mahasiswa dan pelajar saat ini. Mereka mengemabangkan Arduino dengan bootloder dan software yang user *friendly* sehingga menghasilkan sebuah board mikrokontroler yang bersifat *Open source* yang bisa di pelajari dan di kembangan oleh mahasiswa, pelajar, *professional*, pemula dan penggemar elektronika maupun robot di seluruh dunia *Integrated Development Environment* (IDE) diciptakan oleh *Cesey Reas* dan *Ben Fry*, beberapa *programmer* yang lain

juga terlibat seperti *Tom Igoe, Gianluca Martino, David Mellis, dan Nicholas Zambett.*

2.1.2.2 Kelebihan Arduino

Tentu saja ada banyak mikrokontroler maupun *platform* mikrokontroler tersedia, misalnya saja *Basic* Stamp-nya *Parralax, BX-24-nya Netmedia, Phidget, MIT's Handyboard,* dan lain sebagainya. Semua alat tersebut bertujuan untuk menyederhanakan berbagai macam kerumitan maupun detail rumit pada pemograman mikrokontroler sehingga menjadi paket mudah digunakan (*easy –to-use)* . Arduino juga menyederhanakan proses bekerja dengan mikrokontroler, sekaligus menawarkan berbagai macam kelebihan antara lain:

a. Murah. Papan (perangkat keras) Arduino biasanya di jual *relatife* murah dibandingkan dengan *platform* mikrokontoler lainnya. Jika ingni lebih murah lagi,tentu bisa di buat sendiri dan itu sangat mungkin sekali karena semua sumber website arduino bahkan di website-website komunitas arduino lainnya. Tidak hanya cocok untuk *Windows*, namun juga cocok bekerja di *linux, Mac.*
b. Sederhana dan mudah pemogramannya. Perlu diketahui bahwa lingkungan pemograman di arduino mudah di gunakan untuk pemula, dan cukup fleksibel bagi mereka yang sudah tinggkat lanjut. Untuk guru/dosen, Arduino berbasis pada lingkungan pemograman *Processing*, sehingga jika mahasiswa atau murid-murid terbiasa menggunakan *Processing* tentu saja kotak Arduino terdapat tulisan bahwa Arduino di pertuntukan bagi seniman, desainer, penghobi dan siapa saja. sungguh membesarkan hati dan membangkitkan semangat bahwa penggunaan tidak harus teknisi berpengalaman atau ilmuwan.
c. Perangkat lunaknya *Open souce.* Perangkat lunak Arduino IDE dipublikasikan sebagai *Open source*, tersedia bagi para pemogram berpengalaman untuk pengembangan lebih lanjut. Bahasanya bisa di kembangkan lebih lanjut melalui pustaka-pustaka C++ yang berbasis pada bahasa C dan AVR

d. Perangkat kerasnya *Open source*. Perangkat Keras Arduino berbasis mikrokontroler ATMEGA8, ATMEGA168, ATMEGA328 dan ATMEGA1280. Dengan demikian, siapa saja bisa membuat perangkat keras Arduino ini, apalagi bootloader tersedia langsung dari perangkat lunak arduino IDE-nya. Bisa juga menggunakan breadboard untuk membuat perangkat Arduino beserta *peripheral - peripheral* lain yang dibutuhkan.
e. Tidak perlu perangkat *chip programmer*. Karena di dalamya sudah ada bootlader yang akan menangani upload program dari komputer
f. Sudah memiliki sarana komunikasi USB. Sehingga pengguna laptop yang tidak memiliki port serial/RS323 bisa menggunakannya.
g. Bahasa pemograman relatif murah, karena *software* Arduino dilengkapi dengan kumpulan *library* yang cukup lengkap.
h. Memiliki *module* siap pakai (*shiled*) yang bisa ditancapkan pada board Arduino Misalnya *shield GPS*, *Ethernet*, *SD Card*, dan lain-lain.

2.1.2.3 Jenis – Jenis Arduino

Saat ini ada bermacam-macam bentuk dan jenis papan arduino yang disesuaikan dengan peruntukannya, tidak hanya board arduino yang disediakan juga terdapat modul siap pakai *(shield),* juga aksesoris seperti USB adapter dan sebagainya. Berikut jenis-jenis papan Arduino yang ada di pasaran.

2.1.2.3.1 Arduino Uno

Arduino uno adalah papan mikrokontroler berbasis ATMEGA328 yang memiliki 14 pin digital *input*/*output*, 6 diantarnya digunakan sebagai *output Pulse Width Modulation* (PWM), 6 *input* analog, clock speed 16MHz, koneksi USB, Jack listrik, *header* ICSP, dan tombol *reset*. Board ini menggunakan daya yang terhubung ke computer dengan kabel USB atau daya *external* dengan adapator AC-DC atau baterai.

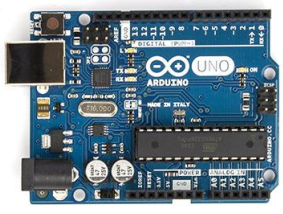
Arduino Uno adalah pilihan yang baik untuk pertama kali atau pemula yang ingin mengenal Arduino. Di samping sifat nya yang *reliabel* juga harga yang terjangkau

Tabel 2. 1 Spesifikasi Board Arduino Uno

Mikrokontroler	ATMEGA328
Tegangan Operasi	5V
Tegangan *Input*	7-12V
Batas tegangan *Input*	6-20V
Pin digital I/O	14 (6 pin PWM)
Pin analog *input*	6
Arus DC per I/O Pin	40 mA
Arus DC untuk pin 3.3V	50 mA
Flash memory	32KB(ATmega328), 0,5 KB bootloder
SRAM	2KB (ATmega328)
EEPROM	1KB(ATmega328)
Clock	16MHz

Arduino dapat di aktifkan melalui koneksi USB atau dengan catu daya *Eksternal*. Sumber daya dipilih secara otomatis. Sumber daya *Eksternal* (Non-USB) dapat berasal dari adapter AC ke DC atau baterai. Adaptor ini dapat dihubungkan dengan menancapkan Power Jack, dapat juga dihubungkan pada Power pin (Gnd dan Vin).

Board Arduino Uno dapat beroperasi pada pasokan *Eksternal* dari 6 sampai 20 volt. Jika disuplai kurang 7V. Meskipun pin 5V dapat di suplai kurang dari 5V, board mungkin tidak stabil. Jika menggunakan teganggan lebih dari 12V , regular tegangan bisa panas dan merusak board. Kisaran yang di sarankan adalah 7 sampai 12 Volt.



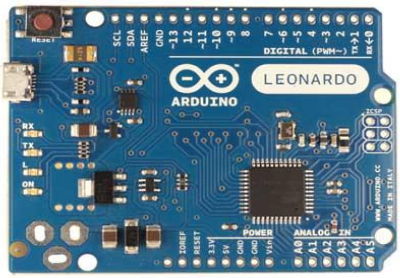
Gambar 2. 1 Arduino Uno

Adapun pin *power supply* pada arduino uno adalah:

- **VIN.** Tegangan *input* board aduino ketika menggunakan sumber daya (5volts dari sambungan USB atau dari sumber *regulator* lain). Anda dapat mensuplai tegangan pada pin ini, jika suplai tegangan lewat power jack, dapat melakukan akses melalui pin ini.
- **5V.** Keluaran pin ini telah diatur sebesar 5V dari *regulator* pada board. Board dapat di suplai melalui pin 5V atau 3.3V *bypasses regulator*, dapat merusask board.
- **3v3**. Suplai 3,3 volt dihasilkan oleh *regulator* pada board. Dengan menarik arus maksimal 50mA.
- **GND.** Pin ground.

2.1.2.3.2 Arduino Leonardo

Arduino Leonardo adalah sebuah papan mikrokontroler berbasis ATmega32u4. Yang mampu mempunyai 20 pin digital *input*/*output*, dimana 7pin dapat digunakan sebagai *output* PWM dan 12Pin analog *input*, *clock speed* 16MHz *crystal oscillator*, sambungan micro USB, power jack, ICSP *header*, dan sebuah tombol *reset*.



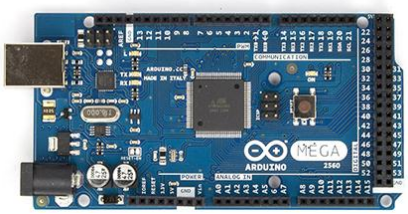
Gambar 2. 2 Arduino Leonardo

Board ini juga menggunakan daya terhubung ke komputer dengan kabel USB atau daya *Eksternal* dengan adaptor AC-DC atau baterai.

2.1.2.3.3 Arduino Mega 2560

Arduino mega 2560 adalah papan mikrokontroler berbasis ATmega2560. Yang mempunyai 54pin digital *input*/*output*, dimana 14 pin dapat digunakan

sebagai *output* PWM, 16 Analog *input*, 4 UARTs(*hadware* serial port), 16 MHz *crystal oscillator*, sambungan USB, power jack, ICSP *header* dan tombol *reset.*

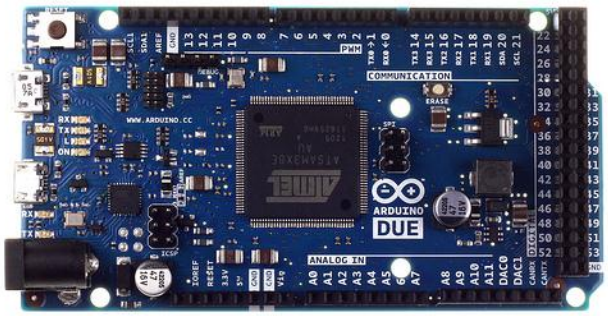


Gambar 2. 3 Arduino Mega 2560

Board ini juga menggunakan daya yang terhubung ke komputer dengan kabel USB atau daya ekternal dengan adaptor AC-DC atau baterai. Arduino mega *compatibel* dengan *shield* yang di desain untuk Arduino *Duemilanover or Diecimila.*

2.1.2.3.4 Arduino Due

Arduino Due adalah sebuah papan mikrokontroler berbasis atmel SAM3X8E ARM Cortex-M3 CPU. Arduino due merupakan mikrokontroler pertama dari arduino berbasis ARM 32bit. Mempunyai 54 pin digital *input*/*output*, 12 pin digunakan untuk *output* PWM, 12 analog *input*, 4 UARTs, *clock speed* 84MHz, sambungan OTG USB, DAC, 2 TWI, Power jack, SPI *header*, JTAG *header*, tombol *reset* dan tombol *erase.*



Gambar 2. 4 Arduino Due

2.1.2.3.5 Arduino Ethernet

Arduino *Ethernet* adalah mikrokontroler berbasis ATmega328 terdapat 14 pin digital *Input output*, 6 analog *input*, clock speed 16 MHz, sambungan RJ45, power jack, ICSP *header* dan tombol *reset*.



Gambar 2. 5 Arduino Ethernet

Arduino Ethernet berbeda dari board yang lain karena tidak mempunyai chip driver onboard USB-serial, tetapi mempunyai Wiznet Ethernet *interface*. Terdapat pembaca kartu memori microSD, yang dapat digunakan untuk menyimpan file-file untuk data akses jaringan, juga dapat di akses melalui *library* SD. Pin10 untuk Wiznet *interface*.

2.1.2.3.6 Arduino Mega ADK

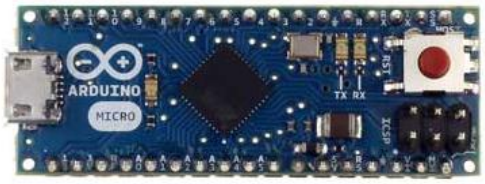
Arduino ADK adalah papan mikrokontroler berbasis ATmega2560. Terdapat USB host *interface* untuk koneksi pada *handphone* berbasis android, berbasis MAX3421e1C. Mempunyai 54 pin digital *input*/*output*, 14 pin digunakan untuk *output* PWM, 16 pin analog *input*, 4 UARTs, 16 MHz *crystal oscillator*, sambungan USB, power jack, ICSP *header* dan tombol *reset*.



Gambar 2. 6 Arduino Mega ADK

2.1.2.3.7 Arduino Micro

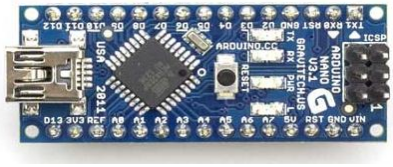
Arduino micro adalah papan mikrokontroler berbasis ATmega32u4. Mempunyai 20 pin digital *input*/*output*, 7pin digunakan, untuk outout PWM dan 12 analog *input* , 16 MHz crystal oscillator, sambungan USB, power jack, ICSP *header* dan tombol *reset* button. Arduino micro mirip dengan Arduino Leonardo yang berbasis ATmega32u4, di dukung dengan komunikasi USB, dan dapat dihubungkan dengan keyboard dan mouse computer.



Gambar 2. 7 Arduino Micro

2.1.2.3.8 Arduino Nano

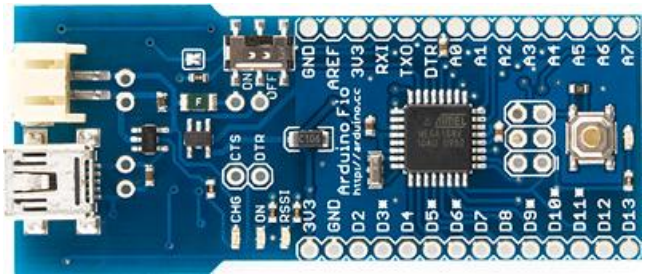
Arduino nano adalah board arduino berukuran kecil, lengkap berbasis ATmega328 untuk arduino nano 3.0 atau ATmega168 untuk arduino nano2.x mempunyai kelebihan yang sama fungsional dengan arduino duemilanove, namun dalam paket yang berbeda. Kekurangannya tidak mempunyai DC power jack, dan hanya dengan kabel Mini-B USB standar. Arduino Nano didesain dan diproduksi oleh *gravitech.*



Gambar 2. 8 Arduino Nano

2.1.2.3.9 Arduino Fio

Arduino fio adalah mikrokontroler berbasis ATmega328P, beroperasi pada tegangan 3.3v dan *clock* 8MHz. mempunyai 14 pin digital *input*/*output*, 6 pin dapat digunakan sebagai *output* PWM, 8 analog *input*, on-board resonator, tombol *reset* dan pin berlubang. Terdapat baterai *lithium Polymer* dan termasuk rangkaian charge circuit via USB. Juga terdapat soket XBee pada bagian bawah.

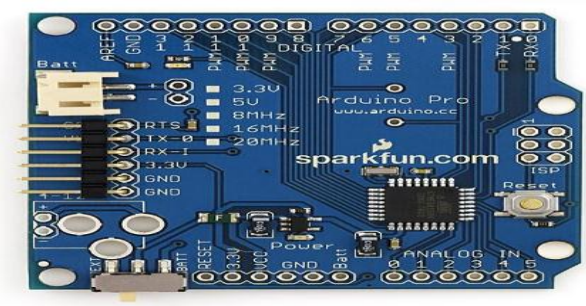


Gambar 2. 9 Arduino Fio

Arduino fio ditunjukan untuk aplikasi wireless. Pengguna dapat mengupload sketch /program dengan kabel FTDI atau *sparkfun breakout board*. Dengan memodifikasi adaptor USB-to-XBee seperti XBee explorer USB, pengguna dapat mengupload sketch melalui wireless. Arduino fio di desain oleh Shigeru Kobayashi dan di publikasi oleh SparkFun Electronics.

2.1.2.3.10 Arduino Pro

Arduino pro adalah papan mikrokontroler berbasis ATmega 168 atau ATmega328. Arduino pro terdiri dari versi 3.3V/8 MHz dan 5V/16 MHz.mempunyai 14 pin digital *input*/*output*, 6 pin digunakan sebagai *output* PWM, 6 analog *input*, baterai power jack, power switch, tombol *reset* dan lubang untuk pemasangan power jack, ICSP *header* dan Pin *header*, 6 pin *header* dapat dihubungkan pada kabel FTDI atau sparkfun breakout board untuk daya USB dan komunikasi ke board[8].



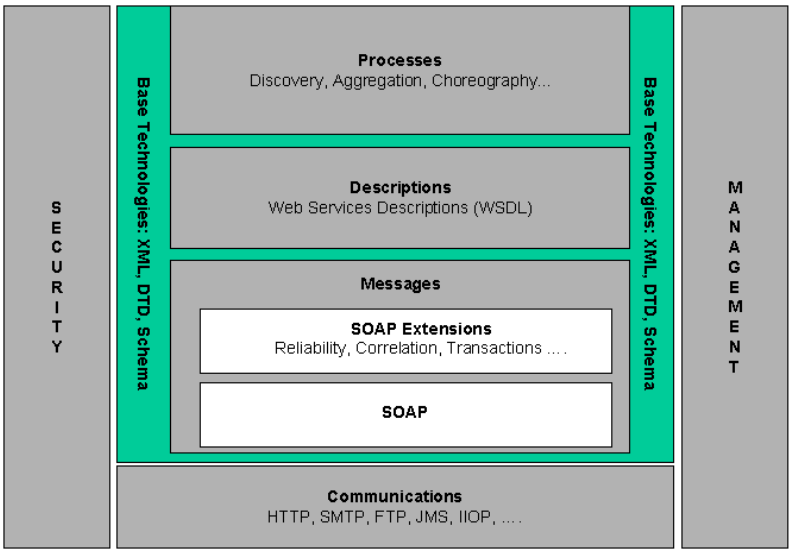
Gambar 2. 10 Android Pro

2.1.3 Web *Service*

Web *service* adalah sebuah entitas komputasi yang dapat diakses melalui jaringan internet maupun intranet dengan standar *protokol* tertentu dalam *platform* dan antarmuka bahasa pemograman yang independen. Tujuan pengembangannya adalah untuk menjembatani komunikasi antar program, sehingga aplikasi yang satu dan aplikasi yang lain yang terdapat pada suatu jaringan yang sama atau pada jaringan berbeda dapat saling berkomunikasi asalkan menggunakan standart protokol yang ditetapkan oleh web *service*. Hal ini bisa terjadi, karena standart protokol itu tidaklah terikat pada suatu *platform* atau bahasa pemograman.

Protokol itu sendiri dibangun oleh *Extensible Markup Language* (XML) yang memang kenyataannya telah didukung oleh banyak *paltform*, bahasa pemograman, dan oleh *developer* di seluruh dunia.

Arsitektur web *service* dibangun oleh beberapa *laye*r dan teknologi yang saling berhubungan. Banyak cara untuk memvisualisaskan *service*, sama banyaknya dengan cara untuk membangun dan menggunakan web *service* gambar berikut hanyalah salah satu cara untuk menggambarkan arsitekturnya.



Gambar 2. 11 Arsitektur web *service*

Web *service* disusun oleh tiga komponen standart, yaitu :

1. *Simple Object Access Protocol* (SOAP), yaitu protokol yang bertanggungjawab dalam pertukuran informasi dalam lingkungan jaringan terdistribusi.
2. *Web service definition Language*(WDSL), dokumen standart yang dituliskan pada format XML, dan mendifinikasan kehadiran web *service* dalam suatu jaringan.
3. *Universal description, Discovery and Inttegration*(UDDI), yaitu suatu lokasi direktori yang beriskan *service* (layanan) dan bersifat bebas *platform (platform independent)*, dituliskan berbasis XML dan dapat diakses oleh *entittas* yang berada di dalam dan luar jaringan.

Adanya standart tersebut membuat web *service* mudah di akses melalui berbagai antarmuka dan juga memberi peluang dimungkinkannya berbagai sistem yang dibangun pada *platform* yang berbeda dan bahasa yang berbeda untuk berkolaborasi pada dalam suatu pekerjaan[9].

2.1.4 Android

Android Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar *(Smartphone)* dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti begerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile dan Nvidia[12]. Bebrapa pengertian lain dari Android, yaitu :

1. Merupakan *platform* terbuka (*Open source*) bagi para *programmer* untuk membuat atau mengembangkan aplikasi.
2. Merupakan sistem operasi yang dibeli Google Inc dari Android Inc.
3. Bukan bahasa pemrograman, akan tetapi hanya menyediakan lingkungan hidup atau *runtime environment* yang disebut DVM *(Dalvik Virtual Machine)* yang telah dioptimasi untuk device dengan sistem memori yang kecil.

2.1.4.1 Versi Android

Banyak smartphone dan PC Tablet menggunakan sistem operasi dengan versi yang berbeda. Semakin tinggi versi, fiturnya semakin canggih dan banyak. Telepon pertama yang memakai sistem operasi Android adalah HTC Dream 13 yang dirilis pada tanggal 22 Oktober 2008. Beberapa uraian versi android seperti dibawah ini :

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.

2. **Android versi 1.5 (*Cupcake*)**

   Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (*Software Development Kit*) dengan versi 1.5 (*Cupcake*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset bluetooth, animasi layar dan keyboard pada layar yang dapat disesuaikan dengan sistem.

3. **Android versi 1.6 (*Donut*)**

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus pada kamera, camcorder dan galeri yang diintegrasikan pada CDMA / EVDO, 802.1x, VPN, Gestures, dan Text-to-spech engine. Kemampuan dial kontak teknologi *text to change speech* (tidak tersedia pada semua ponsel).

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 mengoptimalkan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML 5, daftar kontak yang baru, dukungan flash untuk kamera 3.2 MP, digital zoom, dan Bluetooth 2.1. 14 Untuk bergerak cepat dalam persaingan perangkat generasi berikut, *Google* melakukan investasi dengan mengadakan kompetisi aplikasi mobile terbaik (*killer* apps-aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi

terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik. Dengan semakin berkembangnya dan semakin bertambahnya jumlah handset Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi Android. Aplikasi terkenal yang diubah ke dalam sistem operasi Android adalah *Shazam, Backgrounds dan WeatherBug*. Sistem operasi Android dalam situs internet juga dianggap penting untuk menciptakan aplikasi Android asli, contohnya oleh *MySpace* dan *Facebook.*

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

   Pada 20 Mei 2010, Android versi 2.2 (*Froyo*) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, integrasi dengan V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser,pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portabel dan kemampuan *auto update* dalam aplikasi Android Market.

6. **Android versi 2.3 (*Gingerbread*)**

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

7. **Android versi 3.1 (*Honeycomb*)**

   Android *Honeycomb* dirancang khusus untuk 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 peratam yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

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

   Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Honeycomb* untuk smartphone dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara *offline* dan berbagi informasi dengan menggunakan NFC.

9. **Android versi 4.1 (*Jelly Bean*)**

   Pada tanggal 27 Juni 2012 Google mengumumkan Android versi 4.1 (Jelly Bean) dalam konferensi Google I/O berdasarkan kernel Linux 3.0.31. Jelly Bean adalah pembaruan penting yang bertujuan untuk meningkatkan fungsi dan kinerja antarmuak pengguna (*User Interface*). Pembaruan ini diwujudkan dalam proyek *Butter*, perbaikan ini termasuk antisipasi sentuh, triple *buffering*, perpanjangan waktu *vsync* dan peningkatan *frame* rate hingga 60fps untuk menciptakan UI yang lebih halus. Perangkat pertama yang menggunakan sistem operasi ini adalah tablet Nexus 7, yang dirilis pada 13 Juli 2012.

10. **Android versi 4.4 (*KitKat*)**

    Android versi 4.4 (*KitKat*) direncanakan akan dirilis pada bulan Oktober 2013. Setelah sebelumnya beredar rumor bahwa Android versi berikutnya setelah Jelly Bean diperkirakan akan diberi nomor 5.0 dan dinamai Key Lime Pie. 16 [10].

2.1.5 Lighting control

Lighting control adalah sistem yang mengatur dan mengendalikan penerangan pada suatu bangunan secara terpusat dan terintegrasi. Tujuannya adalah untuk meningkatkan efektif dan efisien dalam pemakaian energi listrik (terkhusus pada penggunaan penerangan).

Lighting adalah suatu pencahayaan yang merupakan salah satu faktor untuk mendapatkan keadaan lingkungan yang aman dan nyaman yang berkaitan erat dengan produktivitas manusia. Pencahayaan yang baik memungkinkan orang dapat melihat objek-objek yang dikerjakannya secara jelas dan cepat, pencahayaan dapat dibagi menjadi 2 bagian yaitu :

a. Pencahayaan alami

Pencahayaan alami adalah sumber pencahayaan yang berasal dari sinar matahari. Sinar alami mempunyai banyak keuntungan, selain menghemat energi listrik juga dapat membunuh kuman. Untuk mendapatkan pencahayaan alami pada suatu ruang diperlukan jendela-jendela yang besar ataupun dinding kaca sekurang-kurangnya 1/6 dari pada luas lantai. Sumber pencahayaan alami kadang dirasa kurang efektif dibanding dengan penggunaan pencahayaan buatan, selain karena intensitas cahaya yang tidak tetap, sumber alami menghasilkan panas terutama saat siang hari. Faktor-faktor yang perlu diperhatikan agar penggunaan sinar alami mendapat keuntungan, yaitu:

1. Variasi intensitas cahaya matahari
2. Distribusi dari terangnya cahaya
3. Efek dari lokasi, pemantulan cahaya, jarak antar bangunan
4. Letak geografis dan kegunaan bangunan gedung.

b. Pencahayaan buatan

Pencahayaan buatan adalah pencahayaan yang dihasilkan oleh sumber cahaya selain cahaya alami. Pencahayaan buatan sangat diperlukan apabila posisi ruangan sulit dicapai oleh pencahayaan alami atau saat pencahayaan alami tidak mencukupi. Fungsi pokok pencahayaan buatan baik yang diterapkan secara tersendiri maupun yang dikombinasikan dengan pencahayaan alami adalah sebagai berikut:

1. Menciptakan lingkungan yang memungkinkan penghuni melihat secara detail serta terlaksananya tugasserta kegiatan visual secara mudah dan tepat
2. Memungkinkan penghuni berjalan dan bergerak secara mudah dan aman
3. Tidak menimbukan pertambahan suhu udara yang berlebihan pada tempat kerja- Memberikan pencahayaan dengan intensitas yang tetap menyebar secara merata, tidak berkedip, tidakmenyilaukan, dan tidak menimbulkan bayang-bayang.
4. Meningkatkan lingkungan visual yang nyaman dan meningkatkan prestasi.

Control atau pengedalian adalah salah satu bagian dari manajemen. Pengendalian dilakukan dengan tujuan supaya apa yang telah direncanakan dapat dilaksanakan dengan baik sehingga dapat mencapai target maupun tujuan yang ingin dicapai. Pengendalian memang merupakan salah satu tugas dari manager, satu hal yang harus dipahami, bahwa pengendalian dan pengawasan adalah berbeda, karena pengawasan merupakan bagian dari pengendalian, bila pengendalian dilakukan dengan disertai pelurusan (tindakan korektif), maka pengawasan adalah pemeriksaan di lapangan yang dilakukan pada periode tertentu secara berulang kali[11].

2.1.6 Relay

Prinsip Kerja Relay - Relay adalah Saklar *(Switch)* yang dioperasikan secara listrik dan merupakan komponen *Electromechanical* (Elektromekanikal) yang terdiri dari 2 bagian utama yakni Elektromagnet (Coil) dan Mekanikal (seperangkat Kontak Saklar/*Switch*). Relay menggunakan Prinsip Elektromagnetik untuk menggerakkan Kontak Saklar sehingga dengan arus listrik yang kecil (low power) dapat menghantarkan listrik yang bertegangan lebih tinggi. Sebagai contoh, dengan Relay yang menggunakan Elektromagnet 5V dan

50 mA mampu menggerakan Armature Relay (yang berfungsi sebagai saklarnya) untuk menghantarkan listrik 220V 2A[12].

2.1.7 Cron Job

Cron adalah software pengatur tugas pada sistem operasi komputer. Cron memungkinkan pengguna untuk mengatur tugas/ komando untuk berjalan pada waktu-waktu tertentu. Ini biasanya dipergunakan untuk mengotomatisasi sistem *maintenance* atau *administrasi*, selain itu juga bisa dipakai untuk tujuan lain seperti koneksi internet dan proses download email. Cron dikemudikan oleh file crontab (cron table), file konfigurasi yang khusus menangani command agar berjalan secara periodik pada jangka waktu yang sudah ditentukan. File crontab disimpan dimana daftar tugas dan instruksi lainnya tersimpan. Pengguna bisa membuat file crontab nya sendiri[13].

2.1.8 ACS712 - 30A

ACS712 atau disebut dengan Sensor arus 30A ini merupakan modul sensor untuk mendeteksi besar arus yang mengalir lewat terminal block menggunakan current sensor chip ACS712-30yang memanfaatkan efek Hall.

Besar arus maksimum yang dapat dideteksi sebesar30A di mana tegangan pada pin keluaran akan berubah secara linear mulai dari 2,5 Volt (½×VCC, tegangan catu daya VCC = 5V) untuk kondisi tidak ada arus hingga 4,5V pada arus sebesar +30A atau 0,5V pada arus sebesar −30A (positif/negatif tergantung polaritas, nilai di bawah 0,5V atau di atas 4,5V dapat dianggap lebih dari batas maksimum). Perubahan tingkat tegangan berkorelasi linear terhadap besar arus sebesar 83,333 mV / Ampere[14].

2.1.9 Pemograman Bereorintasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan peralngkan lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya[11]. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientas objek

didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekata terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini beragam dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut :

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).
2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.
3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.
4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.
5. Meningkatkan kualitas perangkat lunak

Karena pendekatan pengembangan lebih dekat dengan dunai nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.1.9.1 Konsep Dasar 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 dan abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dihubungkus) menjadi suatu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancanggan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok daata dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainnya, dan dapat berinteraksi satu sama lain.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

1. Kelas (*Class*)

   Kelas adalah sekumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (metode/operasi), hubungan (*relationship*) dan arti. Suatu

kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara tekniks kelas adalah sebuah struktur dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek (*object*)

   Objek adalah abstraksi dari sesuatu yang mewakili dunia nyata benda, manusia, satua organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstra. 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.

   Secara teknis, sebuah kelas saat program dieksekusi makan akan dibuat sebuah objek. Objek dilihat darisegi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehinga sebuah objek hanya ada saat sebuah program dieksekusi. Jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat *runtime* (saat sebuah program dieksekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas.

3. Metode (*method*)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

   Metode atau operasi dapat berasal dari *event*, aktifitas atau aksi keadaan, fungsi, atau kelakuan dunnia nyata. Contoh metode atau operasi misalnya *Read*, *Write*, *Move*, *Copy*, dan sebagainya.

4. Atribut (*attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat beruoa nilai atau elemen-elemen data yang dimiliki oleh objek

dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusa atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya.

7. Pewarisan (*inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antarobjek

    Komunikasi antarobjek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

    Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

13. *Package*

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.1.9.2 Pengenalan UML

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

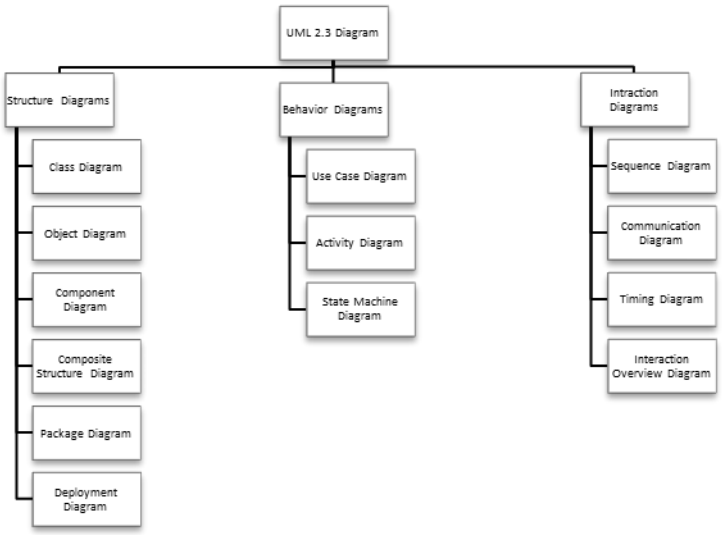
Pada perkembangan teknil pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan voisual untu menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan digaram dan teks-teks pendukung.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

Seperti yang kita ketahui di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkunan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi, belum tentu pandangan yang perbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin digambarkan. Secara analogi jika dengan bahasa yang digunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalahhal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan interpretasi di dalam bidang sistem informasi merupakan hal yang sangat wajar.

2.1.9.3 Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.1.9.1



Gambar 2. 12 Diagram UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut :

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

2.1.9.4 Use Case Diagram

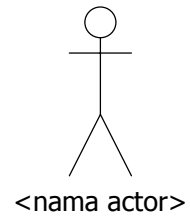
Use Case atau diagram *Use Case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *Use Case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *Use Case* yaitu pendefinisian apa yang disebut aktor dan *Use Case.*

2.1.9.4.1 Actor

Actor adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam use case. Actor menggambarkan orang, sistem atau entitas *Eksternal* yang secara khusus membangkitkan sistem dengan *input* atau masukan kejadian-kejadian, atau menerima sesuatu dari sistem. Actor dilukiskan dengan peran yang mereka mainkan dalam use case, seperti Staff, Kurir dan lain-lain.

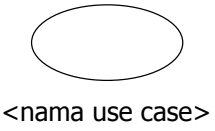
Dalam use case diagram terdapat satu aktor pemulai atau initiator actor yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau participating actor. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.



Gambar 2. 13 Bentuk Actor dalam UML

2.1.9.4.2 Use Case

Use case yang dibuat berdasarkan keperluan aktor merupakan gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. Use case diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor.

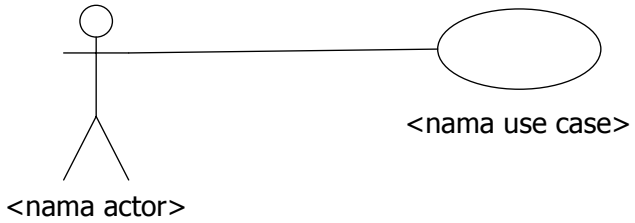


Gambar 2. 14 Bentuk Use Case dalam UML

2.1.9.4.3 Relationship

Relasi (relationship) digambarkan sebagai bentuk garis antara dua simbol dalam use case diagram. Relasi antara actor dan use case disebut juga dengan asosiasi (association). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya.

Relasi-relasi yang terjadi pada use case diagram bisa antara actor dengan use case atau use case dengan use case.



Gambar 2. 15 Bentuk Relationship dalam UML

Relasi antara use case dengan use case :

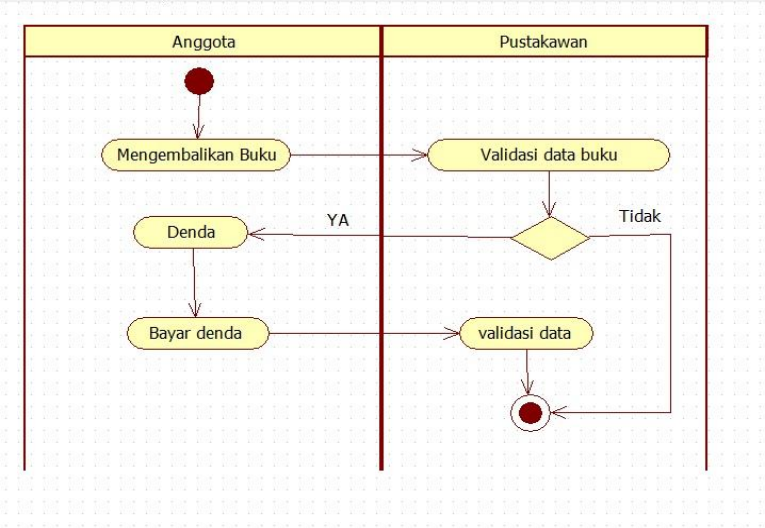
a. *Include*, pemanggilan use case oleh use case lain atau untuk menggambarkan suatu use case termasuk di dalam use case lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<include>>.
b. *Extend*, digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol form dan mendeklarasikan ekstension pada use case utama. Atau dengan kata lain adalah perluasan dari use case lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<extend>>.
c. *Generalization/Inheritance*, dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan base-parent use case. Digambarkan dengan garis berpanah tertutup dari base use case ke parent use case.

2.1.9.5 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya



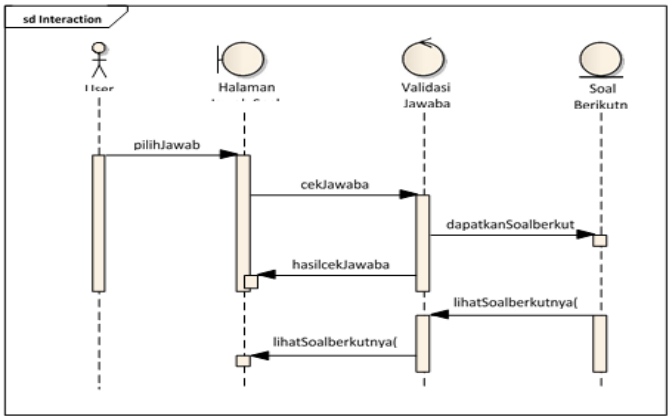
Gambar 2. 16 contoh activity diagram

2.1.9.6 Sequence Diagram

Diagram sequence 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 sequence maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

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

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



Gambar 2. 17 Contoh Sequence diagram

2.1.9.7 *Class* Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas ebaiknya memiliki jenis-jenis kelas berikut :

1) Kelas Main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2) Kelas yang menangani tampilan sistem

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3) Kelas yang diambil dari pendefinisian *usecase*

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.

4) Kelas yang diambil dari pendefinisian data

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada

struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

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

Dalam *class* diagram terdapat beberapa relasi (hubungan antar *class*) yaitu:

1. *Generalization* dan *Inheritence*

   Diperlukan untuk memperlihatkan hubungan pewarisan (*inheritance*) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi, relasi, dari kelas yang berada dalam hirarki pewarisannya.

2. *Associations*

   Hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui ekstensi *class* lain. Dalam notasi UML kita mengenal asosiasi 2 arah *(bidirectional)* dan 1 arah *(undirectional).*

3. *Aggregation*

   Hubungan antar-*class* dimana *class* yang satu *(part class)* adalah bagian dari *class* lainnya *(whole class).*

4. *Composition*

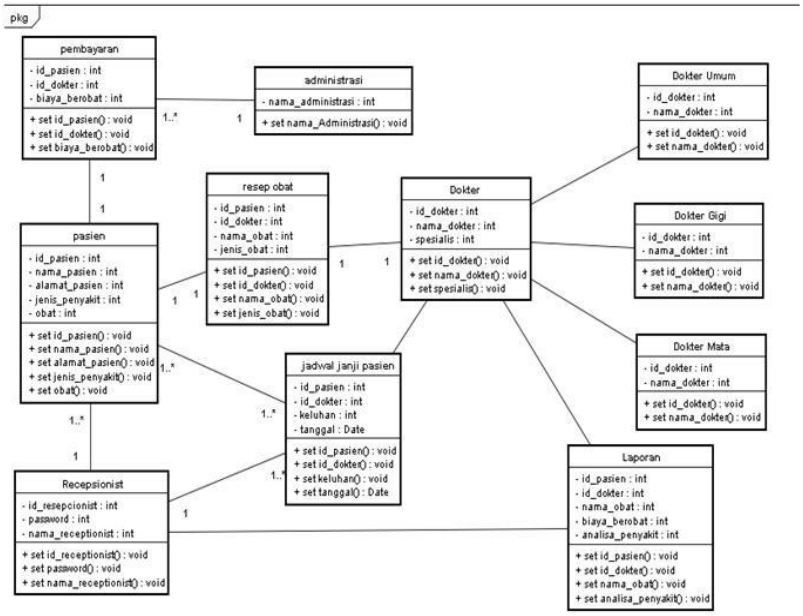
   Aggregation dengan ikatan yang lebih kuat. Di dalam *composite aggregation*, siklus hidup *part class* sangat bergantung pada *whole class* sehingga bila objek *instance* dari *whole class* dihapus maka objek *instance* dari *part class* juga akan terhapus.

5. *Depedency*

   Hubungan antar-*class* dimana sebuah *class* memiliki ketergantungan pada *class* lainnya tetapi tidak sebaliknya.

6. *Realization*

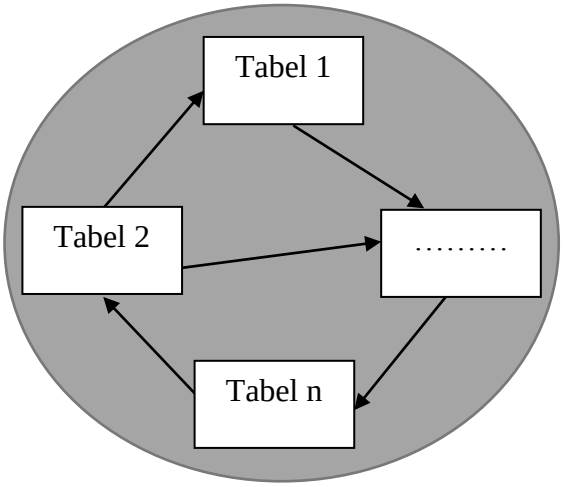
   Hubungan antar-*class* dimana sebuah *class* memiliki keharusan untuk mengikuti aturan yang ditetapkan *class* lainnya. Biasanya realization digunakan untk menspesifikasikan hubungan antara sebuah *interface* dengan *class* yang mengimplementasikan *interface* tersebut[15].



Gambar 2. 18 Contoh gambar *class* diagram

2.1.10 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 menyimpan data agar dapat diakses dengan mudah dan cepat. Pada buku ini menggunakan basis data relasional yang diimplementasikan dengan table-tabel yang saling memiliki relasi seperti pada gambar berikut.



Gambar 2. 19 Ilustrasi Basis data

Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa *file* teks ataupun *Database Management System* (DBMS).

Kebutuhan basis data dalam sistem informasi meliputi:

1. Memasukan, menyimpan dan mengambil data
2. Membuat laporan berdasarkan data yang telah disimpan

Tujuan dari buatnya table-tabel disini adalah untuk menyimpan data ke dalam tabel-tabel agar mudah diakses. Oleh karena itu, untuk merancang tabel-tabel yang akan dibuat maka dibutuhkan pola piker penyimpanan data nantinya jika dalam bentuk baris-baris data *(record)* dimana setiap baris terdiri dari beberapa kolom.

2.1.10.1 SQL

SQL *(Structured Query Language)* adalah bahasa yang digunakan untuk mengelola data pada RDBMS SQL awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus.

SQL mulai berkembang pada tahun 1970an. SQL mulai digunakan sebagai standar yang resmi pada tahun 1986 oleh ANSI *(American National Standards Institute)* dan pada tahun 1987 oleh ISO *(International Organization for Standardization)* dan disebut sebagai SQL-86. Pada perkembangannya, SQL beberapa kali dilakukan revisi. Berikut ini sejarah perkembangan SQL sampai saat ini:

Tabel 2. 2 Sejarah Pekembangan SQL

No	Tahun	Nama
1	1986	SQL-86
2	1989	SQL-89
3	1992	SQL-92
4	1999	SQL:1999
5	2003	SQL:2003
6	2006	SQL:2006
7	2008	SQL:2008
8	20011	SQL:2011

Meskipun SQL diadopsi dan diacu sebagai bahasa standar oleh hampir sebgian besar RDBMS yang beredar saat ini, tetapi tidak semua standar yang tercantum dalam SQL diimplementasikan oleh seluruh DBMS tersebut, sehingga kadang-kadang ada perbedaan perilaku (hasil yang ditampilkan) oleh DBMS yang berbeda padahal *query* yang dimasukan sama.

Berikut ini adalah contoh pengaksesan data pad DBMS dengan SQL yang secara umum terdiri dari 4 hal sebagai berikut:

Memasukan data (*insert*)

```
INSERT INTO Tabel_mahasiswa
(nim, nama, tanggal_lahir)
VALUES (‘13501058’, ‘Rosa’, ‘1986-
```

Query diatas digunakan untuk memasukan data mahasiswa dengan NIM 13501058 nama Rosa dan lahir tanggal 1 januari 1986 ke tabel “Table_mahasisswa”.

Mengubah data (*Update*)

```
UPDATE Tabel_mahasiswa
SET tanggal_lahir =’1990-02-04’
WHERE nim = ‘13501058’;
```

Query diatas digunakan untuk mengubah data tanggal lahir mahasiswa dengan NIM 135010558 menjadi 4 Maret 1990 data tabel “Tabel_mahasiswa” .

Menghapus data(*delete*)

```
DELETE FROM Tabel_mahaiswa
WHERE nim = ‘135010558’;
```

Query diatas digunakan untuk menghapus data mahasiswa dengan NIM =13501058 dari “Tabel_mahasiswa”.

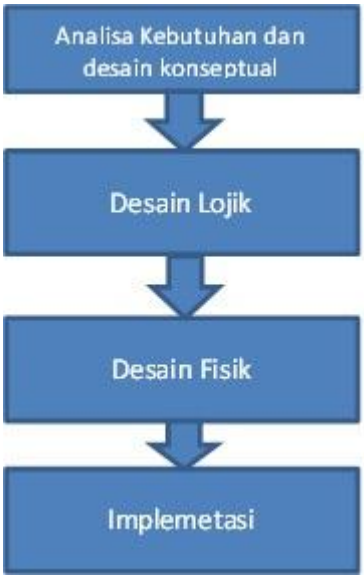
Menampilkan data (select)

```
SELECT nim,nama
FROM Tabel_mahasiswa
WHERE nim = ‘13501058’;
```

Query diatas digunakan untuk menampilkan data mahasiswa yang tersimpan dalam tabel “Tabel_mahasiswa” dengan NIM 13501058.

2.1.10.2 Alur Hidup Basis Data

Tidak hanya perangkat lunak yang memiliki alur hidup, dalam membuat perancangan basis data juga memiliki alur hidup atau *Database Life Cycle* (DBLC). Alur hidup basis data dapat di lihat gambar berikut:



Gambar 2. 20 DBLC

Fase-fase DBLC antara lain:

1. Analisis kebutuhan/*requiretment analysis*

   Hal-hal yang harus dilakukan pada tahap ini adalah:

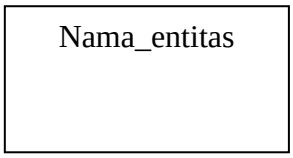
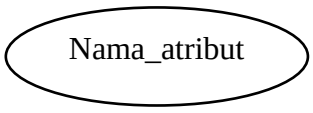
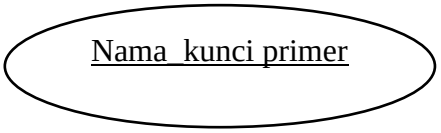
   1) Didefinisikan dengan mewawancarai produsen dan pemakai data, data apa sajakah yang butuh untuk disimpan dan terkait dengan aplikasi computer yang akan dikembangkan
   2) Membuat kontrak spesifikasi basis data
   3) *Entity Relationship Diagram* (ERD) sebagai bagian dari desain konseptual

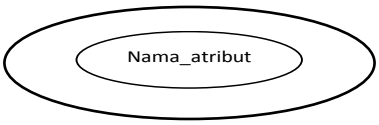
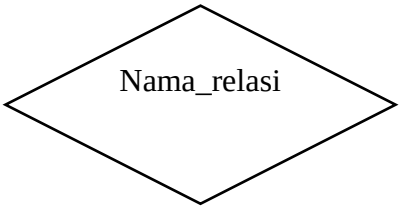
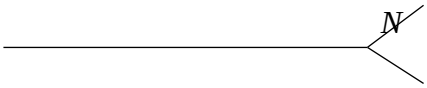
2. Desain Lojik basis data / *logical database design*
   Pada tahapan ini harus dibuat rancangan lojik basis data.biasanya pada tahap ini dibuat *Conceptual Data Model* (CDM).
3. Desain fisik basis data / *physical database design*
   Pada tahap ini harus dibuat rancangan fisik basis data. Biasanya pada tahap ini dibuat *Physical Data Model* (PDM).
4. Implementasi
   1) Membuat Query SQL
   2) Aplikasi ke DBMS atau file.

2.1.10.3 ERD

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagra* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Peter Chen), Barker (dikembangkan oleh Richard Barker, ian Palmer, Harry Ellis), notasi Crows Foot, dan beberapa notasi lain, namun yang banyak digunakan adalah notasi dari Chen. Berikut adalah simbol-simbol yang dignakan pada ERD dengan notasi Chen.

Tabel 2. 3 Simbol- simbo ERD

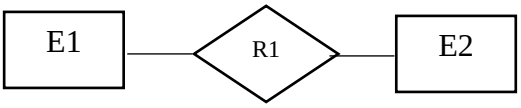
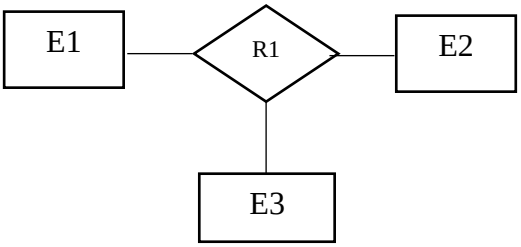
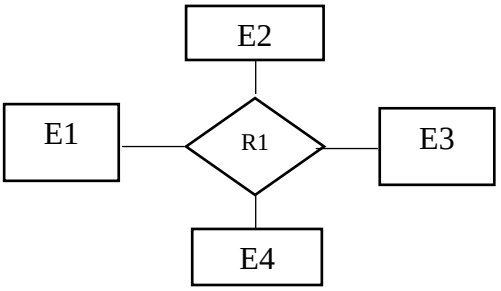
Simbol	Deskripsi
Entitas /*entity* 	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data; benda yang memilki data dan harus disimpan datanya agar dapat diakses oleh aplikasi computer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel
Atribut 	Field atau kolom data yang butuh disimpan dalam suatu entitas
Atribut kunci primer 	Field atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses record yang diingkan; biasnya berupa ID; kuci primer dapat lebih dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama)

Atribut multinilai/ *multivalue* 	Field atau kolom data yang butuh disimpan dalam suatu entitas; biasanya diawali dengan kata kerja
Relasi 	Relasi yang menghubungkan antara entitas; biasanya diawali dengan kata kerja
Asosiasi/ *association* 	Penghubung antara relasi dan entitas dan entitas dimana dikedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian Kemungkinan jumlah maksimun keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinatilas Misalkan ada kardinaltilas 1 ke N atau sering disebut dengan One to many menghubungkan entitas A dan Entitas B

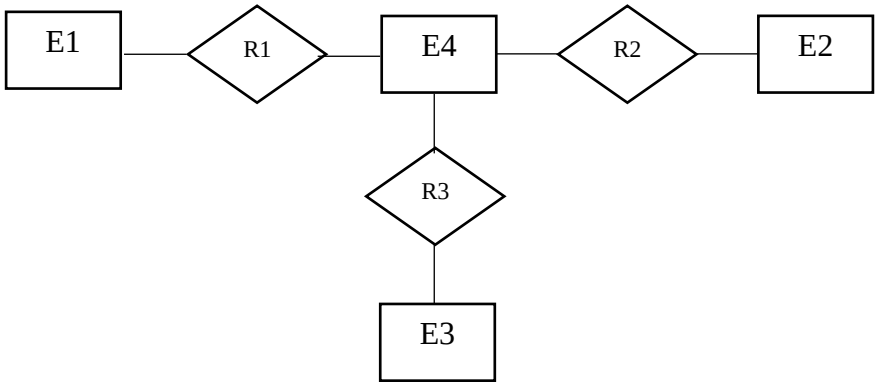
ERD biasanya memiliki hubungan binary (satu relasi menghubungkan dua buah entitas). Beberapa metode perancangan ERD menoleransi hubungan relasi *ternary* (satu relasi menghubungkan tiga buah relasi) atau N-ary (satu relasi menghubungkan banyak enttitas), tiap banyak metode perancangan ERD yang

tidak mengizinkan hubungan ternary atau N-ary. Berikut adalah contoh bentuk hubungan relasi dari ERD:

Tabel 2. 4 Relasi ERD

Nama	Gambar
Binary	
Ternary	
N-ary	

Beberapa metode perancangan ERD memberikan solusi untuk perancangan ERD yang tidak binary diubah sebagai berikut[16]:



Gambar 2. 21 menghindari relasi ternary

2.1.11 Rekayasa Perangkat Lunak

Rekayasa perangkat lunak (*software engineering*) merupakan pembangunan dengan menggunakan prinsip atau konsep rekayasa dengan tujuan menghasilkan perangkat lunak yang bernilai ekonomi yang dipercaya dan bekerja secara efisien menggunakan mesin. Perangkat lunak banyak dibuat dan pada akhirnya sering tidak digunakan karena tidak memenuhi kebutuhan pelanggan atau bahkan karena masalah non-teknis seperti keengan pemakai perangkat lunak (*user*) untuk mengubah cara kerja dari manual ke otomatis atau ketidakmampuan *user* menggunakan komputer. Oleh karena itu reakaya perangkat lunak dibutuhkan agar perangkat lunak dibuat tidak hanya menjadi perangkat lunak yang tidak dipakai.

Rekayasa perangkat lunak lebih fokus pada praktik pengembangan perangkat lunak dan mengirimkan perangkat lunak yang bermanfaat kepada pelanggan *(customer).* Adapun ilmu komputer lebih fokus pada teori dan konsep dasar perangkat komputer. Rekayasa perangkat lunak lebih fokus pada bagaimana membuat perangkat lunak yang memenuhi kriteria berikut:

1. Dapat terus dipelihara setelah perangkat lunak selesai dibuat seiring berkembanya teknologi dan lingkungan *(maintainability)*

2. Dapat diandalkan dengan proses bisnis yang dijalankan dengan perubahan yang terjadi *(dependability rebust)*
3. Efisien dari segi sumber daya penggunaan
4. Kemampuan untuk dipakai sesuai dengan kebutuhan *(usability)*

Dari kriteria diatas maka perangkat lunak yang baik adalah perangkat lunak yang dapat memnuhi kebutuhan pelanggan (custumer) atau *user* (pemakai perangkat lunak) atau berorientasi pada pelanggan atau pemakai perangkat lunak, bukan berorientasi pada pembuat atau pengembang perangkat lunak.

Pekerjaan yang terkait dengan rekayasa perangkat dapat dikategorikan menjadi tiga buah kategori umum tanpa melihat area dari aplikasi, ukuran proyek perangkat lunak, atau kompleksitas perangkat lunak yang akan dibuat. Setiap fase dialamatkan pada satu atau lebih pertanyaan yang diajurkan sebelumnya.

Fase pendefinisian fokus pada *"what"* yang artinya harus mencari tahu atau mendefinisikan informasi apa yang harus diproses, seperti apa fungsi dan performasi yang diingikan, seperti apa perilaku sistem yang diingikan , apa kriteria validasi yang dibutuhkan untuk mendifinisikan sistem.

Fase pengembangan yang fokus dengan *"how"* yang artinya selama tahap pengembangan perangkat lunak seorang perekayasa perangakt lunak *(software engineer)* berusaha untuk mendifinisikan bagaimana data distrukturkan dan bagaimana fungsi-fungsi yang dibutuhkan diimplementasikan didalam arsitektur perangkat lunak, bagaimana detail prosedural diimplementasikan, bagaimana karakter antarmuka tampilan, tampilan desain ditranslasikan ke bahasa pemograman dan bagaimana pengujian akan dijalankan.

Fase pendukung *(support phase)* fokus pada perubahan yang terasosiasi pada perbaikan kesalahan *(eror)* , adaptasi yang dibutuhkan pada lingkungan perangkat lunak yang terlibat dan perbaikan yang terjadi akibat perubahan kebutuhan pelanggan *(costumer)*. Fase pendukung terdiri dari empat tipe perubahan antara lain

1. Koreksi *(correction)*

   Walaupun dengan jaminan kualitas yang terbaik, akan selalu ada kecacatan atua keinginan pelanggan *(customer)* yang tidak ada tertangani

oleh perangkat lunak. Pemeliharaan dengan melakukan perbaikan terhadap kecacatan perangkat lunak.

2. Adaptasi (adaptation)

   Pada saat tertentu lingkungan asli (seperti CPU, sistem operasi, aturan bisinis, karakteristik produk luar) dimana perangkat lunak dikembangkan akan mengalami perubahan pemeliharaan adaptasi merupakan tahap untuk memodifikasi perangkat lunak guna mengakomodasi perubahaan lingkungan luar dimana perangakat luar dijalankan.

3. Perbaikan *(enchancement)*

   Sejalan dengan digunakannya perangkat lunak, maka pelanggan (*customer*) atau pemakai *(user)* akan mengenali fungsi tambahan yang dapat mendatangkan manfaat pemeliharaan perfektif atau penyempurnaan melakukan ekstensi atau penambahan pada keutuhan fungsional sebelumnya.

4. Pencegahaan *(prevention)*

   Keadaan perangkat lunak komputer sangat memungkinkan untuk perubhaan. Oleh karena itu, pemeliharaan pencegahan *(prevention)* atau sering disebut juga dengan rekayasa ulang sistem *(software reengineering)* harus dikonsisikan untuk mampu melayani kebutuhan pemakainya *(user).* Di lain sisi biasanya setelah perangakat lunak dikirimkan ke *user* maka masih dibutuhkan asistensi dan help desk dari pengembang perangkat lunak [16]

2.1.12 Perangkat Lunak Pendukung

Adapun beberapa perangkat lunak pendukung dalam penyusunan laporan Pembangunan *Home Automation* Ligthing Control dengan Android ini adalah

2.1.12.1 Eclipse

Eclipse adalah sebuah IDE (Integrated Development Environment) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform*-independent). 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. Multi-language: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, P erl, PHP, dan lain sebagainya.

   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.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *Open source,* yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.1.12.2 Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah kernel, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP:

1. Core *platform*
2. OSGi
3. SWT *(Standard Widget Toolkit)*
4. JFace
5. Eclipse Workbench

Secara standar Eclipse selalu dilengkapi dengan JDT *(Java Development Tools), plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in* Development Environment) untuk

mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka *(open)*, mudah diperluas *(extensible)* untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (C/C++ Development Tools). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, *plug-in* UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya.

2.1.12.3 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama Simultaneous Release. Setiap versi peluncuran terdiri dari Eclipse *Platform* dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise. Adapun versi Eclipse yang telah diluncurkan yaitu[17] :

Tabel 2. 5 tabel versi eclipse

No	Kode Peluncuan
1	Eclipse 3.0
2	Eclipse 3.1
3	Callisto
4	Europa
5	Ganymede
6	Galileo
7	Helios

8	Indigo
9	juno
10	Kepler

2.1.12.4 Mysql

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (database management system) atau DBMS yang multithread, multi-user,dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomastis.

MySQL juga memiliki beberapa kelebihan, antara lain :

1. Portability
   MySQL dapat berjalan stabil pada berbagai sistem operasi seperti windows, Linux, FreeBSD, Solaris dan lain-lain.
2. Open Source
   MySQL didistribusikan secara open source (gratis), dibawah lisensi GPL sehingga dapat digunakan cuma-Cuma.
3. Multi User
   MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. Performance Tuning
   MySQL memiliki kecepatan yang menakjubkan dalam menangani query sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. Coloumn Types

MySQL memiliki tipe kolom yang sangat kompleks, seperti integer, double, char, text, datedan lain-lain.

6. Command and Function
   MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah select danwhere dalam query.
7. Security
   MySQL memiliki beberapa lapisan sekuritas seperti level subnetmask, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta password terenkripsi.
8. Scability and Limits
   MySQL mampu menangani database dalam skala besar, dengan jumlah records lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. Connectivity
   MySQL dapat melakukan koneksi dengan clients menggunakan protokol TCP/IP, Unix socket (UNIX) atau Named Pipes (NT).
10. Localisation
    MySQL dapat mendeteksi pesan kesalahan pada client dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk didalamnya.

11. Interface
    MySQL memiliki interface (antar muka) terhadap berbagai aplikasi dan bahasa pemograman dengan menggunakan fungsi API (Aplication Programming Interface).
12. Clients and Tools
    MySQL dilengkapi dengan berbagai tools yang dapat digunakan untuk administrasi database dan pada setiap tool yang ada disertakan petunjuk online.
13. Struktur Tabel

MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani ALTER TABLE, dibandingkan database lainnya semacam PosgreeSQL ataupun Oracle.

Kelemahan MySQL dari dulu sampai saat ini adalah feature-creep artinya MySQL berusaha kompatibel dengan beberapa standar serta berusaha memenuhinya namun jika itu diungkapkan kenyataannya bahwa fitur-fitur tersebut belum lengkap dan belum berperilaku sesuai standar. Contoh fitur SUB-SELECT (nesting SELECT dalam SELECT) yang tidak optimal dan sering salah parsing query SQL dan jalan keluarnya dengan memecah menjadi beberapa query[18].