BAB 2
TINJAUAN PUSTAKA

2.1 Aplikasi

Perangkat lunak aplikasi yaitu perangkat lunak yang digunakan untuk membantu pemakai komputer untuk melaksanakan pekerjaannya. Jika ingin mengembangkan program aplikasi sendiri, maka untuk menulis program aplikasi tersebut, dibutuhkan suatu bahasa pemrograman yang dapat berbentuk *assembler, compiler* ataupun *interpreter* [2].

Bahasa pemrograman berfungsi untuk menulis program dengan bahasa yang lebih mudah, dan selanjutnya akan diterjemahkan ke dalam bahasa mesin supaya bisa dimengerti oleh komputer. Bila hendak mengembangkan suatu program aplikasi untuk memecahkan permasalahan yang besar dan rumit, maka supaya program aplikasi tersebut dapat berhasil dengan baik, maka dibutuhkan prosedur dan perencanaan yang baik dalam mengembangkannya. Saat ini terdapat program-program aplikasi yang tersedia dalam bentuk paket-paket program. Beberapa perusahaan perangkat lunak telah memproduksi paket-paket perangkat lunak yang mempunyai reputasi internasional. Program-program paket tersebut dapat diandalkan, dapat memenuhi kebutuhan pemakai, dirancang dengan baik, relatif bebas dari kesalahan-kesalahan, *user friendly* (mudah digunakan), mempunyai dokumentasi manual yang memadai, mampu dikembangkan untuk kebutuhan mendatang, dan didukung perkembangannya [2].

2.2 Smartphone

Menurut Williams dan Sawyer [3] *smartphone* adalah telepon selular dengan mikroprosesor, memori, layar dan modem bawaan. *Smartphone* merupakan ponsel multimedia yang menggabungkan fungsionalitas *Personal Computer* (PC) dan *handset* sehingga menghasilkan *gadget* yang mewah dimana terdapat pesan teks, kamera, pemutar musik, video, *game*, akses email, tv digital, *search engine*,

pengelola informasi pribadi, fitur GPS, jasa telepon internet, dan bahkan terdapat telepon yang juga berfungsi sebagai kartu kredit.

2.3 Android

Pembahasan mengenai *android, tools,* dan bahasa pemrograman yang digunakan dalam pembuatan aplikasi akan dibahas pada sub bab berikut :

2.3.1 Pengertian Android

Android adalah sebuah sistem operasi pada handphone yang bersifat terbuka dan berbasis pada sistem operasi *Linux*. *Android* bisa digunakan oleh setiap orang yang ingin menggunakannya pada perangkat mereka. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk bermacam peranti bergerak.

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 peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile*, dan *Nvidia*.

Pada saat perilisan perdana *Android,* 5 November 2007, *Android* bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, *Google* merilis kode–kode *Android* di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler [4].

2.3.2 Sejarah dan Perkembangan Android

Pada mulanya terdapat berbagai macam sistem operasi pada perangkat selular, diantaranya sistem operasi *Symbian, Microsoft Windows Mobile*, *Mobile Linux, iPhone,* dan sistem operasi lainnya. Namun diantara sistem operasi yang ada belum mendukung standar dan penerbitan *API* yang dapat dimanfaatkan secara keseluruhan dan dengan biaya yang murah. Kemudian *Google* ikut berkecimpung

didalamnya dengan *platform Android*, yang menjanjikan keterbukaan, keterjangkauan, open source dan framework berkualitas.

Pada tahun 2005, Google mengakuisisi perusahaan *Android Inc.* untuk mamulai perkembangan *platform Android*. Dimana terlibat pengembangan ini adalah Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada pertengahan 2007 sekelompok pemimpin industri bersama-sama membentuk analisis aliansi perangkat selular terbuka, *Open Handset Alliance (OHA).* Bagian dari tujuan aliansi ini adalah berinovasi dengan cepat dan menaggapi kebutuhan konsumen dengan lebih baik, dengan produk awalnya adalah *platform Android*. Dimana *Android* dirancang untuk melayani kebutuhan operator telekomunikasi, manufaktur *handset*, dan pengembangan aplikasi, *OHA* berkomitmen untuk membuat *Android open source* dengan lisensi *Apache Versi 2.0* [4].

2.3.3 Versi Android

Android pertama kali diluncurkan pada 5 November 2007, dan smartphone pertama yang menggunakan sistem operasi *Android* dikeluarkan oleh *T-Mobile* degnan sebutan G1 pada bulan September 2008. Hingga saat ini *Android* telah merilis beberapa versi *Android* untuk menyempurnakan versi sebelumnya. Selain berdsarkan penomoran, pada setiap versi Android terdapat kode nama berdasarkan nama-nama kue. Hingga saat ini sudah terdapat beberapa versi yang telah diluncurkan, diantaranya:

1. Android 1.0 (API level 1) dirilis pada 23 September 2008.
2. Android 1.1 (API level 2) dirilis pada 09 Februari 2009.
3. Android 1.5 Cupcake (API level 3) dirilis pada 30 April 2009.
4. Android 1.6 Donut (API level 4) dirilis pada 15 September 2009.
5. Android 2.0 Eclair (API level 5) dirilis pada 26 Oktober 2009.
6. Android 2.0.1 Eclair (API level 6) dirilis pada 3 Desember 2009.
7. Android 2.1 Eclair (API level 7) dirilis pada 12 Januari 2010.
8. Android 2.2 s/d 2.2.3 Froyo (API level 8) dirilis pada 20 Mei 2010, 18 Januari 2011, 22 Januari 2011, 21 November 2011.

9. Android 2.3 s/d 2.3.2 Gingerbread (API level 9) dirilis pada 6 Desember 2010, Desember 2010, Januari 2011.
10. Android 2.3.3 s/d 2.3.7 Gingerbread (API level 10) dirilis pada 9 Februari 2011, 28 April 2011, 25 Juli 2011, 2 September 2011, 21 September 2011.
11. Android 3.0 Honeycomb (API level 11) dirilis pada 22 Februari 2011.
12. Android 3.1 Honeycomb (API level 12) dirilis pada 10 Mei 2011.
13. Android 3.2 Honeycomb (API level 13) dirilis pada 15 Juli 2011, 20 September 2011, 30 Agustus 2011, Desember 2011, Januari 2012, Februari 2012.
14. Android 4.0 s/d 4.0.2 Ice Cream Sandwich (API level 14) dirilis pada 19 Oktober 2011, 21 Oktober 2011, 28 November 2011.
15. Android 4.0.3 s/d 4.0.4 Ice Cream Sandwich (API level 15) dirilis pada 16 Desember 2011, 29 Maret 2012.
16. Android 4.1 s/d 4.1.2 Jelly Bean (API level 16) dirilis pada 9 Juli 2012, 23 Juli 2012, 9 Oktober 2012.
17. Android 4.2 s/d 4.2.2 Jelly Bean (API level 17) dirilis pada 13 November 2012, 27 November 2012, 11 Februari 2013.
18. Android 4.3 Jelly Bean (API level 18) dirilis pada 24 Juli 2013.
19. Android 4.4 KitKat (API level 19) dirilis pada Oktober 2013 [6].
20. Android versi 5.0.1/5.1.1 Lollipop dirilis pada 15 Oktober 2014.
21. Android versi 6.0 Marshmallow dirilis pada 19 Agustus 2015.
22. Android versi 7.0/7.1.1.1 Nougat dirilis pada 22 Agustus 2016.

2.3.4 Kelebihan Android

Sudah banyak *platform* untuk perangkat selular saat ini, termasuk didalamnya *Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiMo),* dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan *Android*. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada *platform* lain, *Android* adalah yang pertama menggabungkan hal seperti berikut :

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan *Linux* dan *open source*. Pembuat perangkat menyukai hal ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar *royality*. Sementara pengembang *software* menyukai karena *Android* dapat digunakan diperangkat manapun dan tanpa terikat oleh vendor manapun.
2. Arsitektur komponen dasar *Android* terinspirasi dari teknologi internet Mashup. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.
3. Banyak dukungan *service,* kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, *database SQL, browser* dan penggunaan peta. Semua itu sudah tertanam pada android sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja *system* menjadi lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.
5. Dukungan grafis dan suarat terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh *Flash* menyatu dalam 3D menggunakan *OpenGL* memungkinkan membuat aplikasi maupun *game* yang berbeda.

Portabilitas aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman *Java* dan dieksekusi oleh mesin [4].

2.4 Android SDK (Software Development Kit)

Android SDK (Software Development Kit) adalah *tools API (Application Programming Interface)* yang diperlukan untuk mulai mengembangkan aplikasi pada *platform Android*. *Android SDK* disediakan sebagai alat bantu untuk mulai

mengembangkan aplikasi pada platform *Android* menggunakan bahasa pemrograman *Java*. Sebagai platform *aplikasi-netral*, *Android* memberi kesempatan untuk membuat aplikasi yang dibutuhkan [5].

Android SDK versi resmi dapat ditemukan di situs resmi *Google*. Namun terdapat juga *SDK* buatan pihak ketiga seperti *App Inventor* untuk *Android, Basic4android, Corona SDK, RubyMotion*, dan *Xamarin*. *SDK* tersebut dapat digunakan untuk menulis program *Android*, namun tentunya bahasa pemrograman yang digunakan menyesuaikan dengan *SDK* masing-masing.

Setiap kali *Google* merilis versi baru dari *Android* maka sebuah *SDK* yang sesuai juga dirilis. Untuk dapat menulis program dengan fitur terbaru maka pengembang aplikasi *Android* harus mengunduh dan menginstal *Android SDK* sesuai versi masing-masing untuk smartphone tertentu. Didalam penelitian ini penulis menggunakan *Android SDK* resmi dari *Google* dengan versi minimum 25.

2.5 Android Studio

Android Studio adalah *Integrated Development Environment (IDE)* untuk pengembangan aplikasi *Android*, berdasarkan *IntelliJ IDEA*. Selain merupakan editor kode *IntelliJ* dan alat pengembang yang berdaya guna, *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas saat membuat aplikasi *Android*, misalnya:

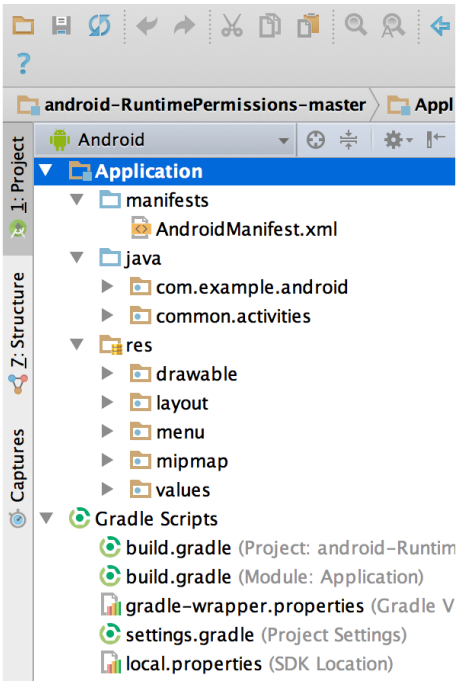
1. Sistem pembuatan berbasis *Gradle* yang fleksibel.
2. *Emulator* yang cepat dan kaya fitur.
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat *Android.*
4. Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat *APK* baru.
5. Template kode dan integrasi *GitHub* untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
6. Alat penguji dan kerangka kerja yang ekstensif.

7. Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
8. Dukungan *C++* dan *NDK*.
9. Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*.

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

1. Modul aplikasi *Android*
2. Modul perpustakaan
3. Modul *Google App Engine*

Secara default, *Android Studio* menampilkan file proyek Anda dalam tampilan proyek Android seperti yang ditunjukkan dalam gambar 2.2. Tampilan ini diatur menurut modul untuk memberi akses cepat ke file sumber kunci proyek Anda.



Sumber Gambar : https://developer.android.com/studio/intro/index.html

Gambar 2.1 File Proyek di Tampilan Android

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

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

Struktur proyek *Android* pada disk berbeda dari representasi rata ini. Untuk melihat struktur file sebenarnya dari proyek ini, pilih Project dari menu tarik turun Project (dalam gambar 2.2, struktur ditampilkan sebagai Android) [6].

Android Studio digunakan peneliti dibandingkan *IDE* lain karena *Android Studio* telah memiliki dukungan penuh dari *Google*. Hal tersebut memudahkan pengembang dalam melakukan integrasi seperti penggunaan sebuah library yang melalui file gradle cukup dengan mendaftarkan satu baris program saja. Adapun versi *Android studio* yang digunakan peneliti yaitu versi 2.3.2.

2.6 Bahasa Pemrograman Java

Java telah berkembang dari semula ditujukan untuk pemrograman applet di *web browser* menjadi bahasa pemrograman pengembangan aneka ragam aplikasi, mulai dari yang berjalan di *handheld devices* seperti *handphone*, PDA, sampai aplikasi tersebar skala *enterprise* di beragam komputer server. Java merupakan bahasa orientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat internet/jaringan komunikasi. Java tidak lagi hanya bahasa untuk membuat applet yang memperindah halaman web tapi Java telah menjadi bahasa untuk pengembangan aplikasi skala enterprise berbasis jaringan besar [7].

Menurut Budi Raharjo, Imam Heryanto, Arif haryono [8] java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsystems* saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun

dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis Java umumnya dikompilasi ke dalam p-code (bytecode) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (general purpose), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi Java mampu berjalan di beberapa platformsistem operasi yang berbeda, Java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini Java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

Java bukan merupakan satu-satunya bahasa pemrograman yang dapat digunakan untuk mengembangkan aplikasi berbasis Android. Adapun bahasa lain seperti Xamarin yang menggunakan bahasa pemgrograman C# yang dapat digunakan sebagai opsi alternatif dalam membangun aplikasi berbasis Android. Namun disini peneliti menggunakan bahasa pemrograman Java karena lebih mudah dalam mencari contoh dokumentasi dan merupakan bahasa yang didukung oleh IDE Android Studio resmi dari Google.

2.7 Raspberry Pi

Raspberry Pi adalah mini komputer mainboard yang bersifat *SoC (System on chip)* dimana ada penggabungan komponen dan fungsi-fungsi komputer serta elektronika kedalam satu chip atau bisa juga disebut dengan *embbedded systems*. Pada SoC memungkinkan pengguna menggunakan *GPIO (General Purpose Input / Output)* untuk keperluan bersifat elektronika misal melakukaan transmisi sinyal radio dari mainboard SoC yang dapat dikontrol melalui aplikasi tertentu yang berjalan pada suatu sistem operasi (OS) yang di gunakan. Ditunjang dengan dukungan prosessor tipe ARM membuat mainboard SoC sangat hemat listrik, murah dan hemat tempat.

Mini PC Raspberry ini dapat diaktifkan dengan catu daya eksternal. Daya dapat berasal dari adaptor DC 5V 2 Ampere. Adaptor dapat dihubungkan dengan menghubungkan konektor *male micro USB ke port female micro USB* yang telah tersedia di mini PC.

2.7.1 Sejarah Raspberry Pi

Raspberry Pi merupakan sebuah komputer yang berukuran sangat kecil yang dikembankan di Inggris oleh Raspberry Pi Foundation. Gagasan dibalik sebuah komputer kecil dan murah untuk anak – anak muncul pada tahun 2006. Ide ini muncul ketika beberapa mahasiswa Laboratorium Komputer di Universitas Cambridge, yakni Eben Upton, Rob Mullins, Jack Lang, dan Alan Mycroft, melakukan eksperimen pada anak – anak pada tahun 90-an mengenai pengalaman si anak menjadikan hobi mereka sebagai seorang *programmer*. Sedangkan, sekitar tahun 2000-an agak berbeda, rata- rata mereka hanya melakukan desain web saja. Ternyata perubahan tersebut terjadi karena dilatar belakangi oleh beberapa parameter, seperti penggunaan kurikulum pembelajaran tentang Microsoft Word dan Excel, menulis halaman web, atau munculnya PC rumah dan *games*.

2.7.2 Spesifikasi Raspberry Pi 3 Model B

Pada Tabel 2 akan menjelaskan spesifikasi hardware pada Raspberry Pi 3 Model B.

Tabel 2.1 Spesifikasi Raspberry Pi 3 Model B

Komponen	Spesifikasi
SoC	Broadcom BCM2837
CPU	4× ARM Cortex-A53, 1.2GHz
GPU	Broadcom VideoCore IV
RAM	1GB LPDDR2 (900 MHz)
Networking	10/100 Ethernet, 2.4GHz 802.11n
Bluetooth	Bluetooth 4.1 Classic
Storage	microSD
GPIO	40-pin header, populated

Komponen	Spesifikasi
Ports	HDMI, 3.5mm, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI).
Power Sources	5V, 2A



Gambar 2.2 Raspberry Pi

2.8 JavaScript

JavaScript adalah bahasa script berdasar pada objek yang memperbolehkan pemaki untuk mengendalikan banyak aspek interaksi pemakai pada suatu dokumen HTML. Dimana objek tersebut dapat berupa suatu window, frame, URL, dokumen, form, button, atau item yang lain. Yang semuanya itu mempunyai properti yang saling berhubungan dengannya dan masing-masing memiliki nama, lokasi, warna nilai dan atribut lain [9].

2.9 JSON (JavaScript Object Notation)

JSON *(JavaScript Object Notation)* adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll [10].

JSON terbuat dari dua struktur yaitu:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

2.10 Analisis dan Perancangan Berorientasi Object

Analisis dan perancangan berorientasi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah mempermudah *programmer* didalam mendesain program dalam bentuk objek-objek beserta keterhubungan antar objek untuk kemudian dimodelkan dalam sistem nyata.

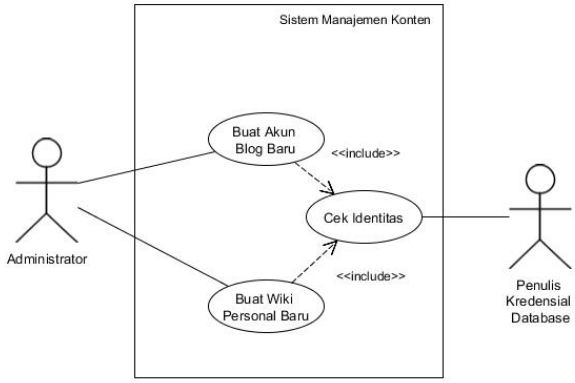
Unified Modelling Language (UML) adalah bahasa pemodelan standar untuk pengembangan perangkat lunak dan sistem [11]. Bahasa UML dapat digunakan untuk memodelkan sistem yang menggunakan pendekatan berorientasi objek. Bahasa UML memberikan banyak fitur opsional dalam mempresentasikan semua aspek penting dalam sebuah sistem seperti menggambarkan domain masalah, desain perangkat lunak yang diinginkan, atau bahkan menggambarkan implementasi perangkat lunak yang telah selesai dibangun. Standar yang saat ini digunakan adalah UML 2.0. Didalam UML 2.0 terdapat tiga belas diagram berbeda

yang dapat digunaan memodelkan perangkat lunak, diantaranya yaitu *use case diagram, activity diagram, sequence diagram, class diagram, component diagram*, dan *deployment diagram.*

2.9.1 *Use Case Diagram*

Use case adalah situasi/*case* dimana sebuah sistem digunakan untuk memenuhi satu atau lebih kebutuhan pengguna. *Use case* dapat digunakan untuk menggambarkan fungsionalitas pada bagian-bagian yang disediakan oleh sistem. Use case menspesifikasikan apa yang harus dilakukan oleh sistem tetapi tidak menspesifikasikan apa yang seharusnya sistem tidak lakukan [11].

Pada Gambar 2.3 adalah contoh diagram *use case.* Kotak besar adalah *system-boundary*. Segala hal yang berada didalam kotak tersebut adalah bagian dari sebuah pengembangan sistem. Dan yang berada diluar kotak dapat dilihat aktor yang bertindak terhadap sistem. Aktor adalah entitas diluar sistem yang memberikan stimulus terhadap sistem. Aktor umunya adalah pengguna. Kemudian didalam *system-boundary* dapat dilihat terdapat beberapa use case. *Use case* berbentuk oval dengan nama kata kerja didalamnya. Para aktor dihubungkan oleh *communication line* kepada *use case* yang berhubungan. Penggunaan garis panah perlu dihindari pada communication line karena tidak seorang pun mengetahui apa maksud dari kepala panah tersebut.



Sumber Gambar : Learning UML 2.0(2006) [11]

Gambar 2.3 Contoh Diagram *Use Case*

Namun, dalam use case diagram tersebut belum menjelaskan mendetail mengenai use case yang tertera. Untuk menjelaskan use case tersebut memerlukan skenario use case. Skenario use case mendeskripsikan secara detail tentang informasi dari suatu use case yang terdapat pada use case diagram. Berikut adalah tipe–tipe informasi yang digunakan dalam use case scenario dijelaskan pada Tabel 2.2 [11].

Tabel 2.2 Tipe Informasi Pada Use Case Scenario

Use Case Name	Memberikan penjelasan singkat tentang nama dari use case.
Related Requirements	Daftar use case yang berhubungan dengan use case tersebut.
Goal In Context	Menjelaskan apa yang aktor coba untuk dapatkan dari use case.
Precoditions	Kondisi sistem sebelum use case dijalankan.
Successful End Condition	Kondisi sistem jika use case berhasil dijalankan.
Failed End Condition	Kondisi sistem jika use case gagal dijalankan.
Primary Actors	Aktor utama yang berpartisipasi pada use case.
Secondary Actors	Aktor yang berpartisipasi pada use case tetapi tidak menjadi yang utama.
Trigger	Event yang dipicu oleh aktor yang menyebabkan use case dijalankan.
Main Flow	Mendeskripsikan langkah yang penting pada eksekusi normal sebuah use case.
Extensions	Mendeskripsikan langkah alternatif suatu langkah di Main Flow.

Sumber Tabel: Learning UML 2.0 (2006) [11]

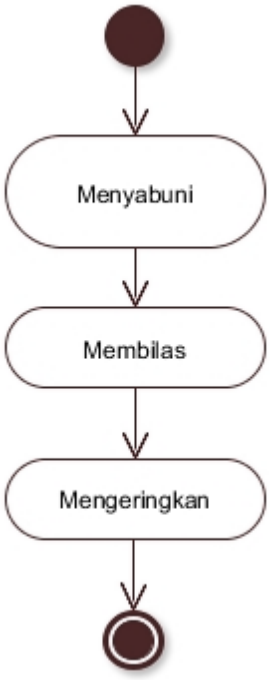
2.9.2 Activity Diagram

Apabila *use case* menunjukkan apa yang sistem harus lakukan, berbeda halnya dengan *activity diagram* yang lebih membahas tentang bagaimana sistem akan mencapai tujuannya. *Activity diagram* menjelaskan aksi rantai pada tingkat yang lebih tinggi untuk mewakili suatu proses yang terjadi didalam sebuah sistem [11].

Komponen utama dari suatu *activity diagram* adalah *action node*, diwakili oleh persegi panjang bulat, yang sesuai dengan tugas yang dilakukan oleh sistem

perangkat lunak. Panah dari satu action node ke action node yang lain menunjukkan aliran kontrol. Panah antara dua *action node* berarti bahwa setelah aksi pertama selesai, aksi kedua dimulai. Sebuah titik hitam yang solid membentuk simpul awal yang menunjukkan titik awal kegiatan. Sebuah titik hitam yang dikelilingi oleh lingkaran hitam merupakan simpul akhir menunjukkan akhir kegiatan.

Fork merupakan pemisah kegiatan menjadi dua atau lebih aktivitas bersamaan. Hal ini digambarkan sebagai bar hitam horisontal dengan satu panah menunjuk ke sana dan dua atau lebih anak panah keluar. Setiap panah keluar mewakili aliran kontrol yang dapat dieksekusi bersamaan dengan arus yang sesuai dengan panah keluar lainnya. Kegiatan bersamaan dapat dilakukan pada komputer dengan menggunakan thread yang berbeda atau bahkan menggunakan komputer yang berbeda. Sebagai contoh, *activity diagram* dapat dilihat pada Gambar 2.4 berikut.



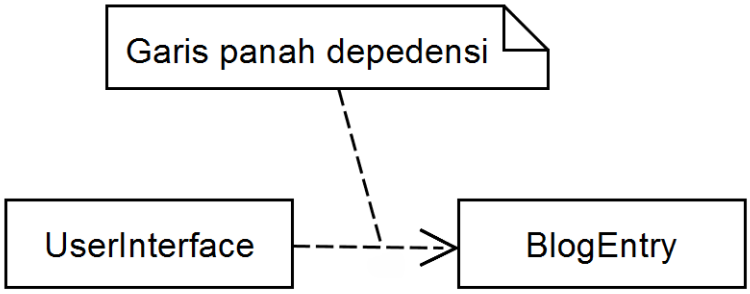
Sumber Gambar : Learning UML 2.0(2006) [10]

Gambar 2.4 Contoh *Activity Diagram* Mencuci Mobil

2.9.3 Class Diagram

Untuk memodelkan kelas, termasuk atribut mereka, operasi, dan hubungan dan asosiasi mereka dengan kelas-kelas lain, UML menyediakan class diagram. Sebuah class diagram memberikan pandangan statis atau struktural dari sebuah sistem. Class diagram tidak menunjukkan sifat dinamis dari komunikasi antara objek dari kelas dalam diagram.

Sebuah kelas digambarkan dengan sebuah kotak yang dibagi menjadi tiga bagian. Pada bagian teratas berisi nama kelas, pada bagian tengah berisi atribut, dan pada bagian paling bawah berisi operasi-operasi yang mewakili karakteristik sebuah kelas. Sebuah atribut mengacu pada informasi yang mewakili status objek. Atribut biasanya diimplementasikan sebagai field dari sebuah kelas. Sedangkan operasi mengacu pada objek apa yang kelas dapat lakukan, biasanya diimplementasikan sebagai method dari sebuah kelas [35]. Sebagai contoh, *class diagram* dapat dilihat pada Gambar 2.5 berikut.



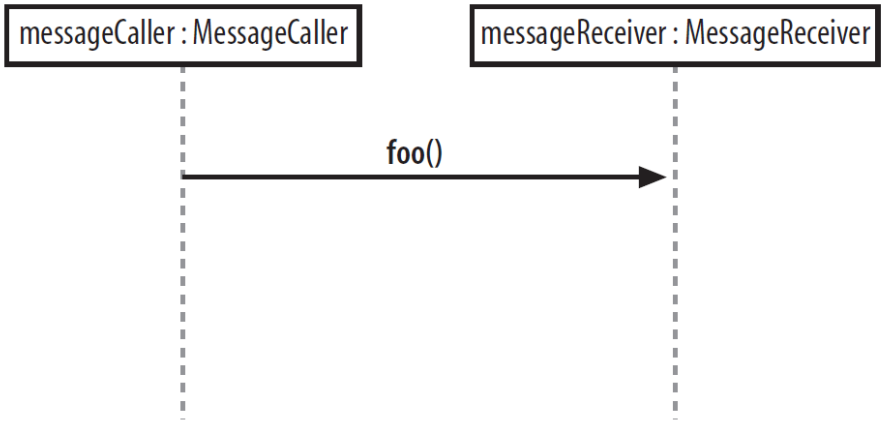
Sumber Gambar : Learning UML 2.0(2006) [11]

Gambar 2.5 Contoh *Class Diagram*

Setiap atribut dapat memiliki nama, jenis, dan tingkat visibilitas. Jenis dan visibilitas adalah opsional. Jenis ini mengikuti nama dan dipisahkan dari nama dengan titik dua. Visibilitas ditunjukkan dengan sebelumnya -, #, ~, atau +, menunjukkan *private, protected, package,* atau *public* [11].

2.9.4 Sequence Diagram

Sequence diagram berisi tentang urutan interaksi antara bagian-bagian didalam sebuah sistem. Menggunakan sequence diagram dapat digambarkan interaksi mana yang akan diberi trigger ketika sebagian use case dieksekusi dan dalam urutan seperti apa interaksi akan terjadi. Sequence diagram menunjukan banyak informasi lain tentang sebuah interaksi, tetapi intinya adalah menunjukan langkah sederhana dan efektif bagaimana mereka mengomunikasikan urutan-urutan kejadian(*events*) didalam sebuah interaksi [11]. Sebagai contoh, *sequence diagram* dapat dilihat pada Gambar 2.6 berikut.



Sumber Gambar : Learning UML 2.0(2006) [11]

Gambar 2.6 Contoh *Sequence Diagram*

Setiap kotak pada bagian atas diagram dulunya merupakan sebuah objek. Setiap objek adalah instansi dari sebuah kelas, dan untuk menandai objek biasanya diberi garis bawah. Didalam UML 2.0, nama kotak ini disebut partisipan untuk mendeskripsikan bagian-bagian yang terlibat dialam interaksi pada sebuah sequence diagram. Di bawah setiap kotak ada garis putus-putus yang disebut *lifeline* objek. Sumbu vertikal dalam sequence diagram sesuai dengan waktu dimana waktu meningkat saat bergerak ke bawah.

Sequence diagram menunjukkan pemanggilan metode menggunakan panah horisontal, diberi label dengan nama metode dan secara opsional termasuk

parameter, jenis, dan jenis kembali. Ketika sebuah objek mengeksekusi metode maka ditampilkan bar putih, disebut bar aktivasi, di bawah *lifeline* objek [11].