BAB II
LANDASAN TEORI

II.1 Pengertian Mobile

Mobile berasal dari bahasa inggris yang artinya berpindah. *Mobile* dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain. Pada konsep ini, *mobile* lebih cenderung dengan aplikasi yang dapat digunakan kapanpun dan dimanapun dengan menggunakan perangkat *mobile* seperti telepon seluler, *pager*, PDA (*Portable Digital Assistant*), *smartphone* dan sejenisnya (Tri Rengganis Setyo Aji, 2010; hal 7).

II.1.1 Pengertian Aplikasi Mobile

Aplikasi *mobile* berasal dari kata *application* dan *mobile. Application* yang artinya penerapan, lamaran, penggunaan. Secara istilah aplikasi adalah program siap pakai yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh sasaran yang dituju sedangkan *mobile* dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain.

Maka aplikasi *mobile* dapat di artikan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah-pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil. Aplikasi *mobile* ini dapat di akses melalui perangkat nirkabel, *pager*, PDA (*Portable Digital Assistant*), telepon seluler, *smartphone*, dan perangkat sejenisnya.

II.1.2 Karakteristik Perangkat Mobile

Perangkat *mobile* memiliki banyak jenis dalam hal ukuran, *desain* dan *layout*, tetapi mereka memiliki karakteristik yang sangat berbeda dari sistem desktop (Elcom, 2010). Berikut karakteristik perangkat *mobile*, diantaranya :

1. Ukuran yang kecil

   Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.

2. Memory yang terbatas

   Perangkat *mobile* juga memiliki memory yang kecil, yaitu *primary* (RAM) dan *secondary* (*disk*). Primary RAM adalah memori yang digunakan untuk proses jalannya aplikasi sedangkan *secondary* (*disk*) merupakan media penyimpanan data. Pembatasan ini adalah salah satu faktor yang mempengaruhi penulisan program untuk berbagai jenis dari perangkat ini. Dengan pembatasan jumlah dari *memory*, pertimbangan-pertimbangan khusus harus diambil untuk memelihara pemakaian dari sumber daya yang mahal ini.

3. Daya proses yang terbatas

   Sistem *mobile* tidaklah setangguh rekan mereka yaitu *desktop*. Ukuran, teknologi dan biaya adalah beberapa faktor yang mempengaruhi status dari sumber daya ini. Seperti *harddisk* dan RAM, Anda dapat menemukan mereka dalam ukuran yang pas dengan sebuah kemasan kecil

4. Mengkonsumsi daya yang rendah

   Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin *desktop*. Perangkat ini harus menghemat daya karena mereka berjalan pada keadaan dimana daya yang disediakan dibatasi oleh baterai-baterai.

5. Kuat dan dapat diandalkan

   Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.

6. Konektivitas yang terbatas

   Perangkat *mobile* memiliki *bandwith* rendah, beberapa dari mereka bahkan tidak tersambung. Kebanyakan dari mereka menggunakan koneksi *wireless.*

7. Masa hidup yang pendek

   Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala. Coba ambil kasus sebuah *handphone*, mereka *booting* dalam hitungan detik dan kebanyakan orang tidak mematikan *handphone* mereka bahkan ketika malam hari. PDA akan menyala jika anda menekan tombol *power* mereka

II.2 Pengertian Doa

Doa adalah sari pati penghambaan diri kepada Allah. Allah telah menjanjikan akan mengabulkan doa atau permohonan hamba yang mau berdoa, sebagaimana dinyatakan dalam firman-Nya: “Berdoalah kamu sekalian kepada-Ku, niscaya aku kabulkan doamu.”(QS. Al-Mukmin [40]: 60).

II.2.1 Manfaat Doa

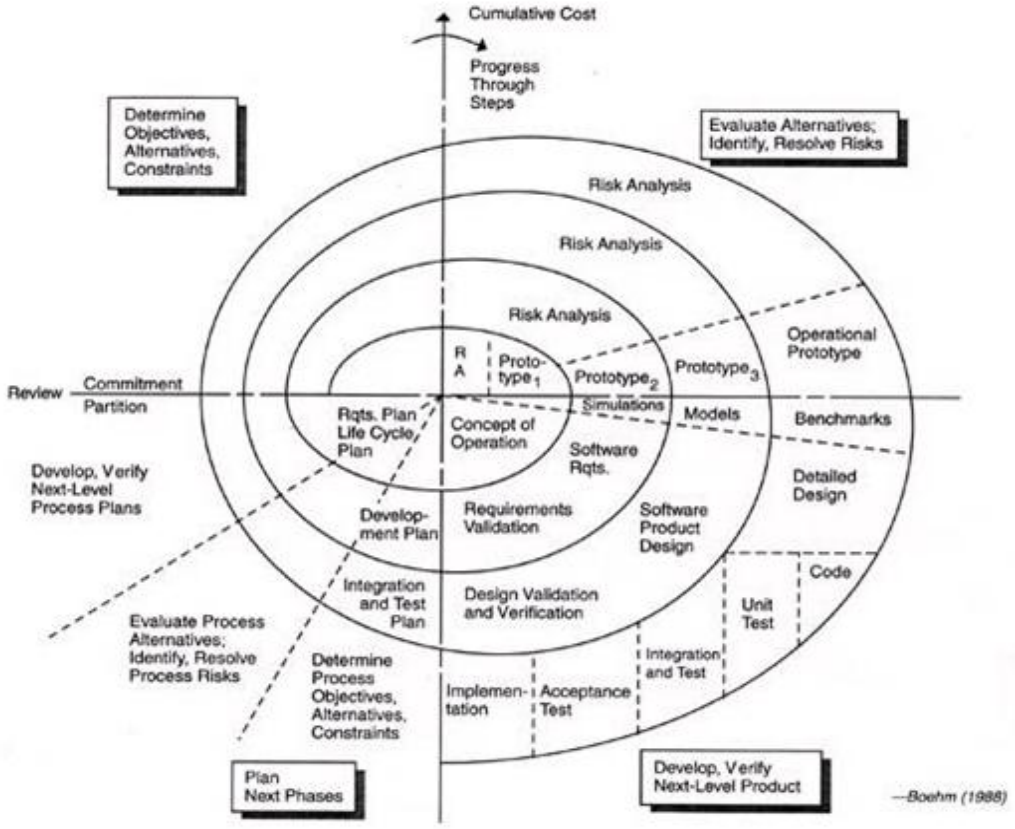
Manfaat doa sangatlah banyak, berikut salah satu manfaat doa diantaranya :

1. Doa berfungsi untuk menunjukkan keagungan Allah SWT kepada hamba-hambaNya yang lemah. Dengan doa seorang hamba menyadari bahwa hanya Allah yang memberinya nikmat, menerima taubat,yang memperkenankan doa-doanya.
2. Doa mengajarkan kita agar merasa malu kepada Allah. Sebab manakala ia tahu bahwa Allah akan mengabulkan doa-doanya, maka tentu saja ia malu untuk mengingkari nikmat-nikmatNya. Bahkan manakala manusia sudah berada dalam puncak keimanan yang kuat sekalipun, maka ia akan lebih dekat lagi (taqarrub) untuk mensyukuri nikmatNya
3. Doa mengajarkan kita agar selalu ingat kepada Allah SWT serta meningkatkan keimanan seseorang**.**

II.3 Metode Spiral

Model ini berbasiskan pada kebutuhan terhadap aplikasi secara keberlanjutan untuk menyaring kebutuhan-kebutuhan tersebut dan estimasi proyek secara keseluruhan. Model ini menerapkan perancangan model proses yang lebih dinamis dengan terus beradaptasi terhadap kebutuhan proses bisnis dimasa yang akan datang sehingga versi aplikasi terus berkembang dengan fitur-fitur yang mengalami peningkatan dari waktu ke waktu.

Kebutuhan waktu untuk pengembangan aplikasi yang cepat dengan kapasitas proyek yang relatif kecil sangat relefan dengan model spiral ini. Keterlibatan pelanggan dengan tim pengembang perangkat lunak akan sangat sering terjadi karena pelanggan akan memberikan *feedback* dan persetujuan setiap tahap dalam pengembangan aplikasi perangkat lunak. Dengan adanya *feedback* dari pelanggan maka estimasi waktu terhadap penyelesaian proyek perangkat lunak menjadi semakin jelas (Fathiah, 2014).



Gambar II.1 ***Model Proses Spiral***
Sumber: Fathiah, 2014

Pengembangan perangkat lunak dengan model spiral memiliki kelemahan karena tidak adanya *milestone* sebagai titik transisi dan pengujian maka

dikhawatirkan proses pengembangan sistem akan mengalami kekacauan dari segi waktu penyelesaian solusi sistem. Oleh karenanya model ini hanya sesuai untuk aplikasi-aplikasi kecil yang tidak terintegrasi dan terdistribusi (Fathiah, 2014).

Model spiral terbagi menjadi empat *quadrant*, dimana setiap quadrant merepresentasikan sebuah manajemen proses dengan tahapan-tahapan *identify*, *design, construct dan evaluate* (Fathiah, 2014).. Sistem akan melalui tahapan-tahapan proses yang akan berulang sebagai berikut:

1. Mendefinisikan tujuan dan kebutuhan bisnis, mengembangkan desain konseptual, rancangan konsep, rencana pengujian, dan analisis terhadap resiko dengan melibatkan pemakai.
2. Mendefinisikan kebutuhan sistem, mengembangkan desain logikal, mengkompilasi (*software-build*) rancangan awal, mengevaluasi hasil dengan melibatkan pemakai.
3. Mendifinisakan kebutuhan subsistem, menghasilkan desain fisikal, mengkompilasi rancangan berikutnya, mengevaluasi hasil dengan melibatkan pemakai.
4. Mendefinisikan kebutuhan setiap unit, menghasilkan desain akhir, mengkompilasi rancangan akhir, mengevaluasi keselur

II.4 Android

Android adalah sekumpulan perangkat lunak yang ditujukan bagi perangkat bergerak mencakup sistem operasi, *middleware*, dan aplikasi kunci. Android *Standart Development Kid* (SDK) menyediakan perlengkapan dan

Application Programming Interface (API) yang diperlukan untuk mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 4)

Android dikembangkan oleh *Google* bersama *Open Handset Alliance* (OHA) yaitu aliansi perngkat seluler terbuka yang terdiri dari 47 perusahaan *Hardware*, *Software* dan perusaan telekomunikasi ditujukan untuk mengembangkan *standar* terbuka bagi perangkat selular (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 4)

II.4.1 Sejarah dan Perkembangan Android

Pada mulanya terdapat berbagai macam sistem oprasi 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 di dalamnya dengan *platform* Android, yang menjanjikan keterbukaan, keterjangkauan, *open source* dan *framework* berkualitas (sumber: Elcom, Google Android - Sistem Operasi Masa Depan, Penerbit Andi, 2010)

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 (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 4)

Android pertama kali diluncurkan pada 5 November 2007, dan *smartphone* pertama yang menggunakan sistem operasi Android dikeluarkan oleh *T-Mobile* dengan 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: versi 1.1 dirlis pada 9 maret 2009, versi 1.5 dirilis pada 30 April 2009 diberi nama *Cupcake,* versi 1.6 dirilis pada 15 September 2009 diberi nama *Donut,* versi 2.0 dirilis pada 26 Oktober 2009 diberi nama *Éclair*, versi 2.2 dirilis pada 20 Mei 2010 diberi nama *Froyo* (*Frozen Yoghurt*), versi 2.3 dirilis pada 6 Desember 2010 diberi nama *Gingerbread*, versi 3.0 dirilis pada Mei 2011 diberi nama *Honeycomb*, versi 4.0 dirilis pada 19 Oktober 2011 diberi nama ICS (*Ice Cream Sandwich*) (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 4)

II.4.2 Kelebihan Android

Sudah banyak *platform* untuk perangkat selular saat ini, termasuk didalamnya Symbian, *iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile* (LiM0), 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 sistem 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.
6. 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 virtual Dalvik, sehingga kode program *portabel* antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard*, layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 7)

II.5 Java

Java adalah bahasa pemrograman yang disusun oleh James Gosling yang dibantu oleh rekan-rekannya seperti Patrick Naugton, Chris Warth, Ed Frank, dan Mike Sheridan di suatu perusahaan perangkat lunak yang bernama Sun Microsystem, pada tahun 1991. Bahasa pemrograman ini mula-mula diinisialisasi dengan nama “oak”, namun pada tahun 1995 diganti namanya menjadi “Java” (Tri Rengganis Setyo Aji, 2010; hal 4).

Alasan utama pembentukan bahasa java adalah untuk membuat aplikasi-aplikasi yang dapat diletakkan diberbagai macam perangkat elektronik, seperti *microwave oven* dan *remote control*, sehingga Java harus bersifat *portable* atau

yang sering disebut dengan *platform independent* (tidak tergantung pada *platform*). Itulah yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah '*write once, run everywhere'*, yang berarti kode program hanya ditulis sekali, namun dapat dijalankan di bawah *platform* manapun, tanpa harus melakukan perubahan kode program (Tri Rengganis Setyo Aji, 2010; hal 4).

II.5.1 Arsitektur Java

Secara arsitektur, Java tidak berubah sedikitpun semenjak awal mula bahasa tersebut dirilis. Kompiler Java (yang disebut dengan **Javac** atau *Java Compiler*) akan mentransformasikan kode-kode dalam bahsa Java ke dalam suatu *bytecode*. Apa itu *bytecode*? *Bytecode* adalah sekumpulan perintah hasil kompilasi yang kemudian dapat dieksekusi melalui sebuah mesin komputer abstrak, yang disebut dengan JVM (*Java Virtual Machine*). JVM juga sering dinamakan sebagai *interpreter*, karena sifatnya yang selalu menerjemahkan kode-kode yang tersimpan dalam *bytecode* dengan cara baris demi baris (Tri Rengganis Setyo Aji, 2010; hal 7).

II.6 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platformindependent*) (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 7). 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, Perl, PHP, dan lain sebagainya.
3. *Multi-role*: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, *test* perangkat lunak, pengembangan *web*, dan lain sebagainya.

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 populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in* (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 7).

II.6.1 Android SDK

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

mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi netral, android *member* anda kesempatan unutk membuat aplikasi yang kita butuhkan yang merupakan aplikasi bawaan *Hadphone/Smartphone* (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 8). Beberapa fitur-fitur android yang paling penting adalah :

1. *Framework* : aplikasi yang mendukung pengganti komponen dan reusable. b. *Dalvik Virtual Machine* dioptimalkan untuk perangkat *mobile*
2. Integrated *Browser* verdasarkan engine *open source* WebKit.
3. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (Opsional Ekselerasi *hardware*)
4. SQLite untuk penyimpanan data.
5. Media *Support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PING, GIF), GSM *Telephony* (tergantung H*ardware*)
6. *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*)
7. Kamera, GPS, Kompas, dan *Accelerometer* (tergantung *hardware*)
8. Lingkungan *Development* yang lengkap dan termasuk pernagkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE Eclipse.

Untuk source SDK Android ini dapat dilihat dan di *download* langsung di situs resmi pengembang SDK Android di http://www.developer.android.com

II.7 SQLite

SQLite merupakan sebuah sistem manajemen basisdata *relasional* yang bersifat ACID-compliant dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. SQLite merupakan proyek yang bersifat *public domain* yang dikerjakan oleh D. Richard Hipp (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 6).

Tidak seperti pada paradigma client-server umumnya, Inti SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basisdata (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah *file*. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan *file* basis data pada saat sebuah transaksi dimulai (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 6).

II.8 Pemodelan UML

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual. Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan

sebuah sistem *software* yang terkait dengan objek.

Sejarah UML sendiri terbagi dalam dua *fase;* sebelum dan sesudah munculnya UML. Dalam *fase* sebelum, UML sebenarnya sudah mulai diperkenalkan sejak tahun 1990an namun notasi yang dikembangkan oleh para ahli analisis dan *desain* berbeda-beda, sehingga dapat dikatakan belum memiliki *standarisasi.*

Fase kedua; dilandasi dengan pemikiran untuk mempersatukan metode tersebut dan dimotori oleh *Object Management Group* (OMG) maka pengembangan UML dimulai pada akhir tahun 1994 ketika Grady Booch dengan metode OOD (*Object-Oriented Design*), Jim Rumbaugh dengan metode OMT (*Object Modelling Technique*) mereka ini bekerja pada *Rasional Software Corporation* dan Ivar Jacobson dengan metode OOSE (*Object-Oriented Software Engineering*) yang bekerja pada perusahaan *Objectory Rasional* (Haviluddin, Jurnal Informatika Mulawarman, 2011; hal 1)

II.8.1 Diagram UML

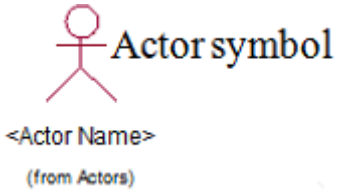
UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case* Diagram untuk memodelkan proses bisnis.
2. *Conceptual* Diagram untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. *Sequence* Diagram untuk memodelkan pengiriman pesan *(message*) antar objek.

4. *Collaboration* Diagram untuk memodelkan interaksi antar objek.
5. *State* Diagram untuk memodelkan perilaku *objects* di dalam sistem.
6. *Activity* Diagram untuk memodelkan perilaku *Use Cases* dan objek di dalam *system.*
7. *Class* Diagram untuk memodelkan struktur kelas.
8. *Object* Diagram untuk memodelkan struktur objek.
9. *Component* Diagram untuk memodelkan komponen *object.*
10. *Deployment* Diagram untuk memodelkan distribusi aplikasi.

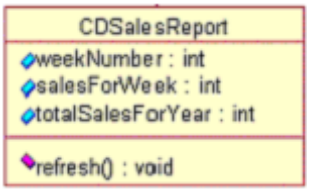
Untuk menggambarkan analisa dan *desain* diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, *behaviour* diagram dan interaction diagram. Berikut beberapa notasi dalam UML diantaranya :

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



Gambar II.2 *Actor*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

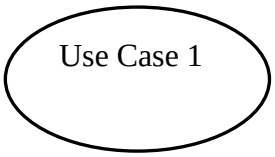
2. *Class* diagram; Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek.



Gambar II.3 *Class Diagram*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

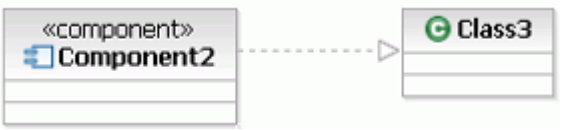
3. *Use Case* dan *use case specification*; *Use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai.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

Use-case symbol



Gambar II.4 *Use Case*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

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

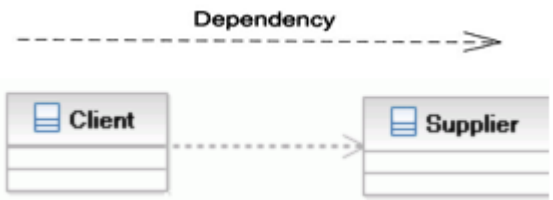


Gambar II.5 *Realization*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

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

Gambar II.6 *Interaction*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

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

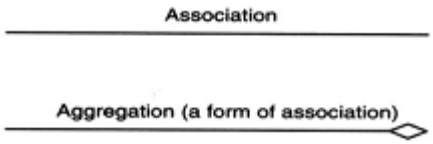


Gambar II.7 *Dependency*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

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

Gambar II.8 *Note*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

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

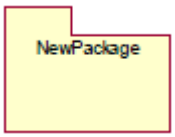


Gambar II.9 *Association*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

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

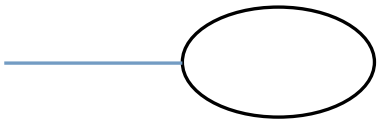
Gambar II.10 *Generalization*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

10. *Package; package* adalah mekanisme pengelompokkan yang digunakan untuk menandakan pengelompokkan elemen-elemen model.



Gambar II.11 *Package*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

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



Gambar II.12 *Interface*
Sumber : Haviluddin, Jurnal Informatika Mulawarman, 2011

Berikut akan dijelaskan empat macam diagram yang paling sering digunakan dalam pembangunan aplikasi berorientasi objek, yaitu *use case diagram, sequence diagram, collaboration diagram, dan class diagram.*

1. *Use Case Diagram*

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk *use case* dan aktor. Aktor merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi. *Use case* merepresentasikan operasi-operasi yang dilakukan oleh aktor. *Use case* digambarkan berbentuk *elips* dengan

nama operasi dituliskan di dalamnnya. Aktor yang melakukan operasi dihubungkan dengan garis lurus ke *use case* (Ardzi Firman Ihtiyar Rohanianto,2014; hal 7).

2. Sequence Diagram

Sequence diagram menjelaskan secara *detail* urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 7).

3. Class Diagram

Class diagram merupakan diagram yang selalu ada di permodelan system berorientasi objek. *Class diagram* menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 7).