

BAB II

LANDASAN TEORI

2.1 Android

2.1.1 Pengertian Android

Android adalah sistem operasi mobile yang paling cepat tumbuh (OS). Dengan lebih dari 800.000 aplikasi yang tersedia di *Google Play Store*, ekosistem Android berkembang juga. Ada cukup keragaman dalam fitur perangkat dan operator nirkabel untuk menarik hanya tentang siapa pun. Netbooks selalu platform alami untuk mengadopsi Android, tetapi keaktifan di belakang Android telah makan pertumbuhan lebih lanjut ke tablet, televisi, dan bahkan mobil. Banyak perusahaan terbesar di dunia dari perbankan sampai makanan cepat saji sampai penerbangan telah menetapkan kehadiran di Android dan menawarkan layanan yang kompatibel. Pengembang Android memiliki banyak kesempatan, dan aplikasi yang relevan menjangkau lebih banyak orang daripada sebelumnya, meningkatkan kepuasan membuat aplikasi yang relevan (Schwarz dkk, 2013).

2.1.2 Sejarah dan Perkembangan Android

Google melihat pertumbuhan besar dalam penggunaan Internet dan pencarian dalam perangkat mobile, mengakuisisi Android, Inc, pada tahun 2005 untuk fokus pengembangan pada platform. *Apple* mengenalkan perangkat mobile iPhone pada tahun 2007 dengan beberapa ide terobosan termasuk *multitouch* dan pasar terbuka untuk aplikasi. Android dengan cepat diadaptasi untuk menyertakan fitur ini dan menawarkan perbedaan yang pasti, seperti kontrol lebih untuk pengembang dan *multitasking*. Selain itu, Android menggabungkan persyaratan perusahaan, seperti dukungan pertukaran, *remote wipe*, dan *Virtual Private Network (VPN)* dukungan, untuk pergi setelah pasar perusahaan yang *Research In Motion* telah dikembangkan dan diselenggarakan dengan baik dengan model Blackberry-nya (Steele dan To, 2010).

commit to user

Keragaman perangkat dan adaptasi yang cepat telah membantu Android berkembang ke basis pengguna, tetapi ia datang dengan tantangan potensial untuk *developers*. Aplikasi perlu untuk mendukung ukuran layar ganda, rasio resolusi, *keyboard*, sensor hardware, versi OS, kecepatan data nirkabel, dan konfigurasi sistem. Masing-masing dapat menyebabkan perilaku yang berbeda dan tak terduga, tapi pengujian aplikasi di semua lingkungan merupakan tugas yang mustahil (Steele dan To, 2010).

Android karena itu telah dibangun untuk memastikan sebagai seragam pengalaman di platform mungkin. Dengan abstrak perbedaan *hardware*, Android OS mencoba untuk melindungi aplikasi dari modifikasi perangkat khusus sambil memberikan fleksibilitas untuk aspek lagu yang diperlukan. Pemeriksaan yang akan mendatang aplikasi pengenalan platform perangkat baru dan pembaruan OS juga mengalami pertimbangan. Hal ini kebanyakan bekerja selama *developer* sangat menyadari pendekatan sistematis ini. *Application Programming Interfaces* (API) umum yang menawarkan Android dan bagaimana memastikan perangkat dan kompatibilitas OS adalah benang utama yang dibahas dalam buku ini (Steele dan To, 2010).

Namun, seperti dengan platform tertanam, pengujian ekstensif dari aplikasi yang diperlukan. Google memberikan bantuan kepada pengembang pihak ketiga dalam berbagai bentuk seperti *Android Development Tool (ADT) plugin* untuk *Eclipse* (juga sebagai alat berdiri sendiri) termasuk *realtime* kemampuan *logging*, emulator realistis yang menjalankan kode *ARM* asli, dan di-laporan kesalahan lapangan dari pengguna untuk pengembang aplikasi *Android Market* (Steele dan To, 2010).

2.1.3 *The Dalvik Virtual Machine (DVM)*

Salah satu elemen kunci dari android adalah Dalvik Virtual Machine (DVM). Android berjalan di Dalvik Virtual Machine (DVM) bukan di Java Virtual Machine (JVM), sebenarnya banyak persamaan dengan Java Virtual Machine (JVM) seperti Java ME (Java Mobile Edition), tetapi android menggunakan Virtual Machine

sendiri yang diskostumisasi dan dirancang untuk memastikan bahwa beberapa fitur-fitur berjalan lebih efisien pada perangkat mobile (Felker, 2011).

2.1.4 Komponen Aplikasi

Android memiliki 4 komponen aplikasi. Setiap komponen memiliki peran yang berbeda-beda dan berdiri sendiri meskipun terkadang salah satu komponen bergantung pada komponen lainnya. Masing-masing komponen memiliki daur hidup masing-masing, yang mengatur kapan komponen tersebut dibuat dan bagaimana komponen tersebut dihapus. Untuk mengaktifkan komponen-komponen tersebut diperlukan sebuah pesan asinkron yang disebut *Intent* (Gargenta, 2011).

Berikut adalah 4 tipe komponen dalam aplikasi android:

1. *Activity*, merupakan suatu komponen yang merepresentasikan sebuah tampilan yang dilengkapi dengan *user interface*.
2. *Service*, merupakan suatu komponen yang berjalan di balik layar untuk menjalankan operasi yang menghabiskan banyak waktu.
3. *Content Provider*, merupakan suatu komponen yang mengatur sekumpulan data aplikasi.
4. *Broadcast receiver*, merupakan komponen yang merespon terhadap siaran (*broadcast*) pengumuman yang dikeluarkan oleh sistem.

2.1.5 Google Maps API

Google menyediakan layanan API (*Application Programming Interface*) untuk menampilkan peta pada halaman *website*. Aplikasi ini diberi nama *Google Maps API* (*Gmaos API*). Peta yang ditampilkan diambil dari layanan *Google Maps*. Ada tiga jenis tampilan yang bisa dipilih dari *Google Maps*, yaitu : *Map*, *Sattelite*, dan *Hybrid*. *Map* menampilkan peta dalam bentuk peta garis, *Sattelite* menampilkan peta dalam bentuk citra/foto satelit dan *Hybrid* merupakan gabungan dari *Map* dan *Sattelite*. Aplikasi ini menggunakan *Ajax* yang merupakan gabungan dari *Javascript* dan *XML*. *Gmaps API* merupakan aplikasi yang *open source* sehingga dapat digunakan secara bebas, legal, dan gratis. Untuk menggunakannya

diperlukan registrasi terlebih dahulu untuk mendapatkan *API Key* dari google yang akan digunakan dalam kode program (Yuniar, 2008).

2.1.6 Latitude dan Longitude

Latitude adalah garis yang melintang diantara kutub utara dan kutub selatan, yang menghubungkan antara sisi timur dan barat bagian bumi. Garis ini memiliki posisi membentangi bumi, sama halnya seperti garis *equator*, tetapi dengan kondisi nilai tertentu garis lintang inilah yang dijadikan ukuran dalam mengukur sisi utara-selatan koordinat suatu titik di belahan bumi (Istiyanto, 2013).

Latitude dibedakan menjadi 2 wilayah, yaitu utara atau yang biasa kita sebut lintang utara dan selatan atau yang biasa kita sebut lintang selatan, dimana nilai koordinat di bagian utara selalu positif dan nilai koordinat di bagian selatan adalah negatif (Istiyanto, 2013).

Berikut nilai-nilai yang dijadikan patokan ukuran garis lintang ini :

1. Garis paling atas (kutub utara) = 90 derajat
2. Garis paling tengah (*equator*) = 0 derajat, dan
3. Garis paling bawah (kutub selatan) = -90 derajat

Dengan menyamakan derajat ke dalam bentuk satuan kilometer (km) maka ukurannya seperti ini :

1 derajat *latitude* = 111 km

1 menit *latitude* = 1.85 km

Sebaliknya, *longitude* adalah garis membujur yang menghubungkan antara sisi utara dan sisi selatan bumi (kutub). Garis bujur ini digunakan untuk mengukur sisi barat-timur koordinat suatu titik di belahan bumi (Istiyanto, 2013).

Sama seperti *equator* pada *latitude* yang berada di tengah dan memiliki nilai 0 (nol) derajat, pada *longitude*, garis tengah yang bernilai 0 (nol) derajat disebut garis *prime meridian*. Sedangkan garis yang berada paling kiri memiliki nilai -90 derajat, dan yang paling kanan memiliki nilai 90 derajat (Istiyanto, 2013).

Longitude juga dibedakan menjadi 2 wilayah, yaitu bujur timur dan bujur barat, dimana koordinat yang berada di timur selalu bernilai negatif, dan sebaliknya yang berada di barat selalu positif. Nilai suatu ukuran derajat menjadi kilometer pada *longitude* juga sama seperti pada *latitude* (Istiyanto, 2013).

Jadi, dalam metode pengukuran koordinat, suatu titik terlebih dulu diukur derajatnya, berdasarkan *latitude* dan *longitude*-nya, setelah itu barulah barulah ditranslasikan kedalam bentuk satuan kilometer, baik itu dalam format *degree* (DDD) maupun *degree-minutes-second* (DMS) (Istiyanto, 2013).

Pada cek lokasi terdapat fungsi perkalian *longitude* dan *latitude* dari lokasi yang bersatuan derajat dengan nilai 1E6 atau 1×10^6 (Istiyanto, 2013).

Double geoLat = location.latitude * 1E6

Double geoLng = location.longitude * 1E6

Perkalian tersebut diperlukan karena *object geopoint* sebagai penyimpan nilai koordinat *latitude* dan *longitude* menggunakan satuan *micro-degree* (mikro-derajat), titik koordinat tersebut akan digunakan sebagai referensi *mapview*. Dengan demikian, titik koordinat yang telah diperoleh dari kedua provider (*wireless network* dan GPS) harus dikonversikan ke satuan mikro-derajat, dengan cara dikalikan nilai 1×10^6 (Istiyanto, 2013).

2.1.7 GIS (*Geografis Information System*)

Sistem Informasi Geografis merupakan sejenis perangkat lunak yang dapat digunakan untuk pemasukan, penyimpanan, manipulasi, menampilkan, dan keluaran informasi geografis berikut atribut-atributnya (Prahasta, 2001).

Menurut sumber lain GIS adalah suatu sistem informasi yang dirancang untuk bekerja dengan data yang bereferensi spasial atau berkoordinat geografi, atau dengan kata lain suatu GIS adalah suatu sistem basis data dengan kemampuan khusus untuk menangani data yang bereferensi keruangan (spasial) bersamaan dengan seperangkat operasi kerja (Barus dan Wiradisastra, 2000).

commit to user

Sedangkan menurut (Anon, 2001) Sistem Informasi Geografis adalah suatu Sistem Informasi yang dapat memadukan antara data grafis (spasial) dengan data teks (atribut) objek yang dihubungkan secara geografis di bumi (georeference). Disamping itu, GIS juga dapat menggabungkan data, mengatur data dan melakukan analisis data yang akhirnya akan menghasilkan keluaran yang dapat dijadikan acuan dalam pengambilan keputusan pada masalah yang berhubungan dengan geografi.

2.1.8 Mobile GIS

Mobile GIS merupakan sebuah integrasi cara kerja perangkat lunak/keras untuk pengaksesan data dan layanan geospasial melalui perangkat bergerak via jaringan kabel atau nirkabel (Tso,1998).

2.2 Spiral Development Model

Model spiral (*spiral model*) yang pada awalnya diusulkan oleh Boehm adalah model proses perangkat lunak yang evolusioner yang merangkai sifat iteratif dari prototipe dengan cara control dan aspek sistematis dari model sekuensial linier. Di dalam model spiral, perangkat lunak dikembangkan di dalam suatu deretan pertambahan. Selama awal iterasi, rilis inkremental bisa merupakan sebuah model atau prototipe kertas. Selama iterasi berikutnya, sedikit demi sedikit dihasilkan versi sistem rekayasa yang lebih lengkap (Pressman,2005).

Model spiral dibagi menjadi sejumlah aktifitas kerangka kerja, disebut juga wilayah tugas, diantara tiga sampai enam wilayah tugas :

1. Komunikasi pelanggan, tugas-tugas yang dibutuhkan untuk membangun komunikasi yang efektif diantara pengembang dan pelanggan. Perencanaan, tugas-tugas yang dibutuhkan untuk mendefinisikan sumber-sumber daya, ketepatan waktu, dan proyek informasi lain yang berhubungan. Analisis resiko, tugas-tugas yang dibutuhkan untuk menaksir resiko-resiko, baik manajemen maupun teknis. Perencanaan, tugas-tugas yang dibutuhkan untuk membangun satu atau lebih representasi dari aplikasi tersebut.

commit to user

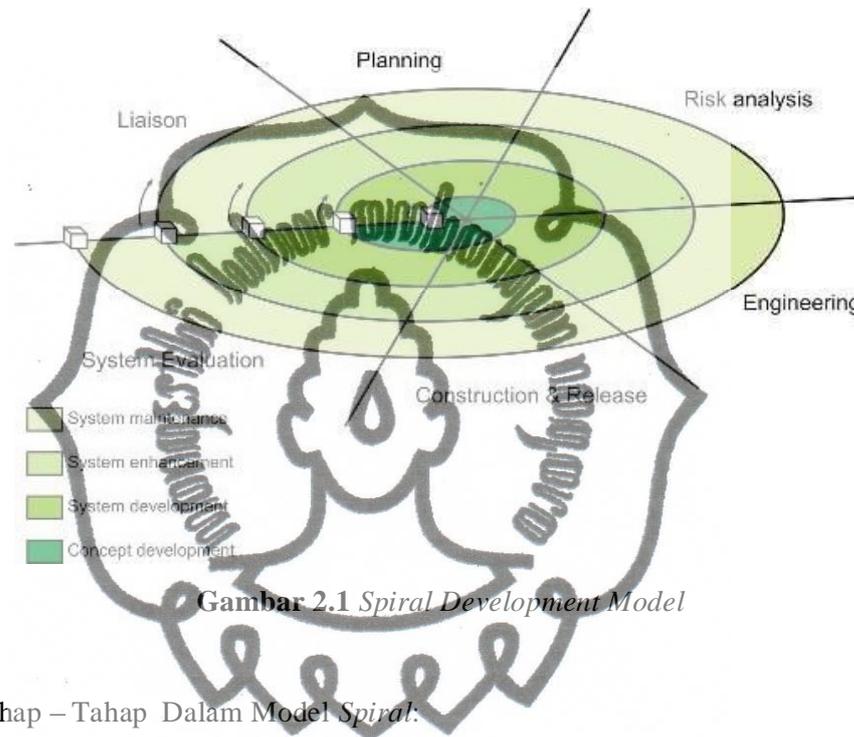
2. Konstruksi dan peluncuran, tugas-tugas yang dibutuhkan untuk mengkonstruksi, menguji, memasang (instal) dan memberikan pelayanan kepada pemakai (contohnya pelatihan dan dokumentasi).
3. Evaluasi pelanggan, tugas-tugas yang dibutuhkan untuk memperoleh umpan balik dari pelanggan dengan didasarkan pada evaluasi representasi perangkat lunak, yang dibuat selama masa perekayasaan, dan diimplementasikan selama pemasangan.
4. Model spiral menjadi sebuah pendekatan yang realistis bagi perkembangan system dan perangkat lunak skala besar. Karena perangkat lunak terus bekerja selama proses bergerak, pengembang dan pemakai memahami dan bereaksi lebih baik terhadap resiko dari setiap tingkat evolusi. Model spiral menggunakan prototipe sebagai mekanisme pengurangan resiko. Tetapi yang lebih penting lagi, model spiral memungkinkan pengembang menggunakan pendekatan prototipe pada setiap keadaan di dalam evolusi produk. Model spiral menjaga pendekatan langkah demi langkah secara sistematis seperti yang diusulkan oleh siklus kehidupan klasik, tetapi memasukkannya ke dalam kerangka kerja iterative yang secara realistis merefleksikan dunia nyata.

Prinsip dasarnya adalah :

1. Fokus dari model adalah pada resiko penilaian dan mengurangi resiko proyek dengan cara membagi proyek kedalam bagian yang lebih kecil sehingga memudahkan perubahan selama proses pengembangan.
2. Setiap siklus melibatkan kemajuan melalui langkah-langkah yang sama, untuk setiap bagian dari produk dan untuk masing-masing tingkat atas elaborasi, dari keseluruhan konsep dokumen operasi ke coding setiap individu.
3. Setiap proses melewati 4 kuadran :
 - a. Menentukan tujuan, alternatif, dan kendala dari iterasi.
 - b. Mengevaluasi alternatif, mengenali dan menyelesaikan resiko.

commit to user

- c. Mengembangkan dan mengecek kiriman dari iterasi.
- d. Merencanakan iterasi selanjutnya.
- e. Memulai setiap siklus lingkaran dengan keinginan pemangku kepentingannya, dan akhiri dengan review dan komitmen.



Gambar 2.1 Spiral Development Model

Tahap – Tahap Dalam Model *Spiral*:

1) *Liason*.

Aktivitas yang dibutuhkan untuk membangun komunikasi yang efektif antara developer dengan user/costumer terutama mengenai kebutuhan dari costumer.

2) *Planning*.

Aktivitas perencanaan ini dibutuhkan untuk menentukan sumberdaya, perkiran waktu pengerjaan dan informasi lainnya yang dibutuhkan untuk pengembangan software.

3) *Risk Analysis*.

Aktivitas analisis resiko ini dijalankan untuk menganalisis baik resiko secara teknikal maupun secara manajaerial. Tahap inilah yang mungkin tidak ada

commit to user

pada model proses yang juga menggunakan metode iterasi, tetapi hanya dilakukan pada spiral model.

4) *Engineering.*

Aktivitas yang dibutuhkan untuk membangun 1 atau lebih representasi dari aplikasi secara teknikal.

5) *Construction & Release.*

Aktivitas yang dibutuhkan untuk develop software, testig, instalasi dan penyediaan user/costumer support seperti training penggunaan software serta dokumentasi seperti buku manual penggunaan software.

6) *System Evaluation*

Aktivitas yang dibutuhkan untuk mendapatkan feedback dari user/costumer berdasarkan evaluasi mereka selama representasi software pada tahap engineering maupun pada implementasi selama instalasi software pada tahap construction and release.

Kelebihan *Spiral Model* :

- a. Dapat disesuaikan agar perangkat lunak bisa dipakai selama hidup perangkat lunak komputer.
- b. Lebih cocok untuk pengembangan sistem dan perangkat lunak skala besar
- c. Pengembang dan pemakai dapat lebih mudah memahami dan bereaksi terhadap resiko setiap tingkat evolusi karena perangkat lunak terus bekerja selama proses .
- d. Menggunakan prototipe sebagai mekanisme pengurangan resiko dan pada setiap keadaan di dalam evolusi produk.
- e. Tetap mengikuti langkah-langkah dalam siklus kehidupan klasik dan memasukkannya ke dalam kerangka kerja iteratif.
- f. Membutuhkan pertimbangan langsung terhadap resiko teknis sehingga mengurangi resiko sebelum menjadi permasalahan yang serius.

Kekurangan *Spiral Model*.

commit to user

- a. Sulit untuk menyakinkan pelanggan bahwa pendekatan evolusioner ini bisa dikontrol.
- b. Memerlukan penaksiran resiko yang masuk akal dan akan menjadi masalah yang serius jika resiko mayor tidak ditemukan dan diatur.
- c. Butuh waktu lama untuk menerapkan paradigma ini menuju kepastian yang absolut.

2.3 Metode *Black Box Testing*

Metode *black box* dilakukan tanpa melihat source code program dan dijalankan oleh tester maupun user untuk mengamati apakah program tersebut telah menerima input, memproses dan menghasilkan output dengan benar (Jogiyanto,2004).

2.4 Basis Data

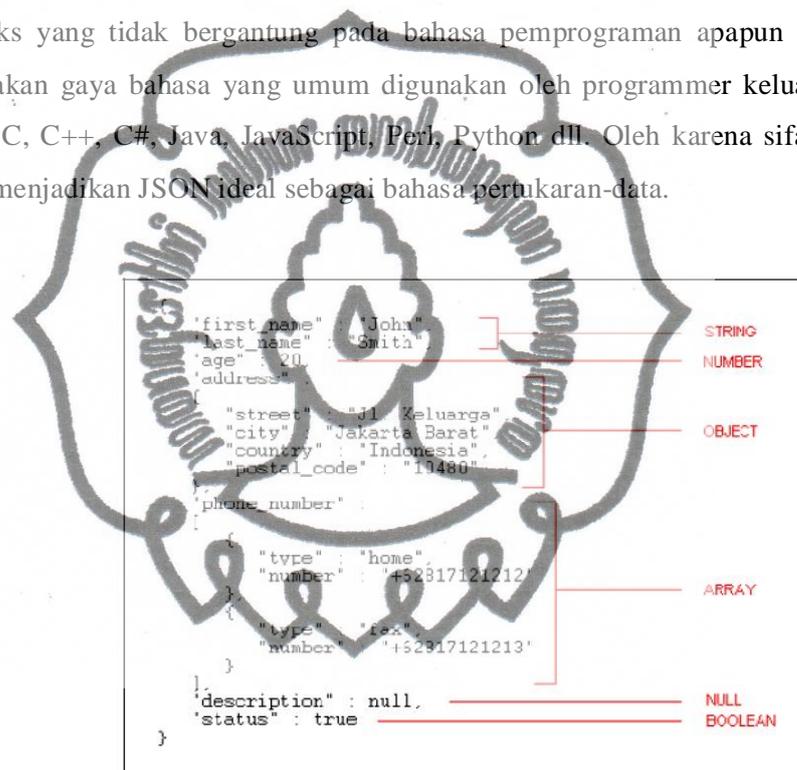
Database menurut Irmansyah (2003) adalah kumpulan dari item yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di hardware komputer dan dengan software untuk melakukan manipulasi untuk kegunaan tertentu.

Untuk membentuk suatu *Database* diperlukan jenjang data. Jenjang data dimulai dari :

- a. *Character* : merupakan bagian data yang terkecil, dapat berupa karakter numerik, huruf ataupun karakter-karakter khusus (*spesial character*) yang membentuk item data/*field*.
- b. *Field* : mempresentasikan suatu dari *record* yang menunjukkan suatu item dari data, misalnya nama, alamat dan lain sebagainya. Kumpulan dari *field* membentuk *record*.
- c. *Record* : *record* menggambarkan suatu unit data individu yang tertentu. Kumpulan dari *record* membentuk suatu *file*.
- d. *File* : *file* terdiri dari *record – record* yang menggambarkan satu kesatuan data yang sejenis.
- e. *Database* : kumpulan dari file/tabel membentuk suatu *Database*
commit to user

2.5 JSON (*JavaScript Object Notation*)

JSON 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 pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.



Gambar 2.2 Contoh Representasi JSON pada sebuah *Object*

JSON terbuat dari dua struktur:

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*) (Crockford, n.d).

commit to user

2.6 MySQL

MySQL (*My Structured Query Language*) adalah sebuah program pembuat dan pengelola database atau yang sering disebut dengan DBMS (*Database Management Sistem*), sifat dari DBMS ini adalah *Open Source*, sehingga dapat diperoleh secara gratis. MySQL sebenarnya produk yang berjalan pada platform Linux, dengan adanya perkembangan dan banyaknya pengguna, serta lisensi dari database ini adalah *Open Source*, maka para pengembang kemudian merilis versi Windows.

Selain itu MySQL juga merupakan program pengakses database yang bersifat jaringan, sehingga dapat digunakan untuk aplikasi Multi User (banyak pengguna). Kelebihan lain dari MySQL adalah menggunakan bahasa query (permintaan) standar SQL (*Structured Query Language*) yang merupakan suatu bahasa permintaan yang terstruktur, SQL telah distandarkan untuk semua program pengakses database seperti Oracle, PostgreSQL, SQL Server dan lain-lain (Nugroho, 2008).

2.7 UML

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. (Dharwiyanti, 2003).

2.8 Use Case Diagram

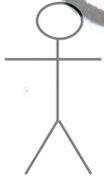
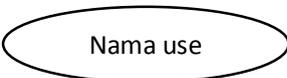
Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem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

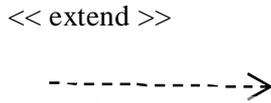
Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor. (Shalahuddin dan Rosa, 2011)

Berikut adalah simbol-simbol yang ada pada diagram *use case* :

Tabel 2.1 Simbol *Use Case*

Simbol	Nama	Keterangan
 Nama Aktor	<i>Actor/Aktor</i>	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi lain di luar sistem informasi itu sendiri; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
 Nama use	<i>Use Case</i>	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antarunit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frases nama <i>use case</i> .
	Asosiasi / <i>association</i>	Komunikasi antara aktor dan <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor.

Simbol	Nama	Keterangan
	Ekstensi / <i>extend</i>	Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu; mirip dengan prinsip <i>inheritance</i> pada pemrograman.
	Generalisasi / <i>generalization</i>	Hubungan generalisasi dan spesifikasi (umum -khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

2.9 Sequence Diagram

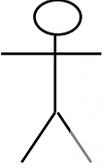
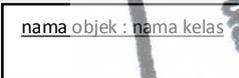
Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

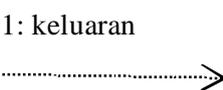
Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksinya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Shalahuddin dan Rosa , 2011).

Berikut adalah simbol-simbol yang ada pada diagram sekuen :

commit to user

Tabel 2.2 Simbol *Sequence Diagram*

Simbol	Nama	Keterangan
 Nama Aktor	Aktor/Aktor	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi lain diluar sistem informasi itu sendiri; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
	Garis hidup / <i>lifeline</i>	Menyatakan kehidupan suatu objek
	Objek	Menyatakan objek yang berinteraksi pesan
	Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan
<<create>> 	Pesan tipe <i>create</i>	Menyatakan suatu objek membuat objek lain, arah panah mengarah pada objek yang dibuat
l: nama method() 	Pesan tipe <i>call</i>	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek yang dibuat
l: masukkan 	Pesan tipe <i>send</i>	Menyatakan bahwa suatu objek mengirimkan data masukkan/informasi ke objek lainnya, arah panah mengarah ke objek yang dikirim

Simbol	Nama	Keterangan
	Pesan tipe <i>return</i>	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode yang menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
X	Pesan tipe <i>destroy</i>	Menyatakan akhir hidup suatu objek

2.10 Collaboration Diagram

Collaboration diagram menggunakan prinsip yang sama dengan *sequence* diagram, sama-sama memodelkan interaksi antar obyek-obyek, yang membedakannya hanya cara penggambarannya saja. Pada *collaboration* diagram ini, obyek-obyek dan *message* (pesan) yang ada digambarkan mirip seperti *flowchart*, hanya saja, untuk menjaga urutan pesan yang diterima oleh masing-masing obyek, pesan-pesan tersebut diberi nomor urutan pesan (Nugroho,2005).

2.11 Class Diagram

Class diagram atau diagram kelas merupakan suatu diagram yang menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

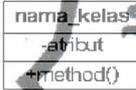
1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

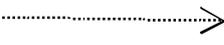
Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam 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 *cohesion* yang kuat dan kadar *coupling* yang lemah. (Shalahuddin dan Rosa , 2011).

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.3 Simbol *Class Diagram*

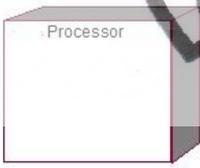
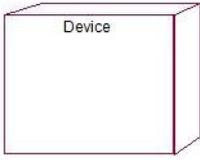
Simbol	Nama	Keterangan
	Kelas	Merupakan kelas pada struktur sistem yang terdiri dari atribut dan method.
	Antarmuka / <i>interface</i>	Sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek
	Asosiasi / <i>association</i>	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .
	Asosiasi berarah / <i>directed association</i>	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .
	Generalisasi / <i>generalization</i>	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)

Simbol	Nama	Keterangan
	Kebergantungan / <i>Dependency</i>	Relasi antar kelas dengan makna kebergantungan antar kelas
	Agregasi / aggregation	Relasi antar kelas dengan makna semua-bagian (whole-part)

2.12 Deployment Diagram

Diagram ini memperlihatkan pemetaan software kepada *hardware*. Diagram ini menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak, bagaimana kemampuan jaringan pada lokasi tersebut dan hal lain yang bersifat fisik (Dharwiyanti dan Wahono, 2003).

Tabel 2.4 Simbol *Deployment Diagram*

Simbol	Nama	Keterangan
	<i>Processor</i>	Prosesor adalah suatu mesin yang mempunyai kekuatan pemrosesan.
	<i>Devices</i>	Peralatan (<i>device</i>) adalah perangkat keras dengan tujuan tunggal atau maksud yang tertentu.
	<i>Connection</i>	Koneksi adalah suatu hubungan (link) secara fisik antara dua prosessor, dua peralatan, atau antara prosessor dan peralatan.

2.13 Activity Diagram

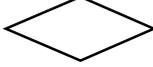
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan sistem.

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 (Shalahuddin dan Rosa, 2011).

Berikut simbol-simbol yang ada pada diagram aktivitas :

Tabel 2.5 Simbol Activity Diagram

Simbol	Nama	Keterangan
●	Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
	Aktivitas	Aktivitas yang dilakukan system, aktivitas biasanya diawali dengan kata kerja
	Percabangan / <i>decision</i>	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu

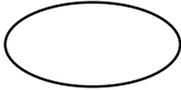
Simbol	Nama	Keterangan
	Penggabungan / <i>join</i>	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
	Status akhir	Status akhir aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status akhir

2.14 Entity Relationship Diagram (ERD)

ERD (*Entity Relationship Diagram*) adalah diagram yang berguna untuk memodelkan sistem yang nantinya akan dikembangkan dalam basis datanya. ERD terusan atas 3 (tiga) komponen, yaitu entitas, atribut dan keselarasan antar entitas. Secara garis besar, entitas merupakan objek dasar yang terlihat dalam sistem. Atribut berperan sebagai penjeles entitas, dan kerapian menunjukkan hubungan yang terjadi diantara 2 (dua) entitas (Sutanta, 2004). Simbol-simbol yang digunakan dalam ERD dapat digambarkan sebagai berikut :

Tabel 2.6 Simbol-simbol gambaran ERD (*Entity Relationship Diagram*)

Simbol	Arti	Keterangan
	Entitas	Suatu objek yang dapat didefinisikan dalam lingkungan pemakai dalam konteks system yang telah dibuat. Entity digunakan atau digambarkan persegi empat.

Simbol	Nama	Keterangan
	Atribut	Elemen–elemen yang ada dalam entity dan fungsi. Atribut mendeskripsikan karakter entity. Atribut digambarkan dengan symbol elips.
	Hubungan	Hubungan ini dinamakan relationship atau relasi. Hubungan dibedakan antara hubungan bentuk antara <i>entity</i> dengan isi dari hubungan ini sendiri. Hubungan digambarkan dengan simbol ketupat.
	Garis	Garis ini digunakan untuk menghubungkan <i>entity</i> dengan <i>entity</i> maupun <i>entity</i> dengan atribut.

2.15 PHP

PHP singkatan dari PHP *Hypertext Preprocessor* yang di gunakan sebagai *script server-side* dalam pengembangan web yang disisipkan pada dokumen HTML. PHP dikatakan sebagai sebuah *server-side embedded script language* artinya sintaks-sintaks dan perintah yang kita berikan akan sepenuhnya dijalankan oleh server tetapi disertakan pada halaman HTML biasa. Aplikasi-aplikasi yang dibangun oleh PHP pada umumnya akan memberikan hasil pada web browser, tetapi prosesnya secara keseluruhan dijalankan di server. Ketika menggunakan PHP sebagai *server-side embedded script language* maka server akan melakukan hal-hal sebagai berikut :

1. Membaca permintaan dari client/browser .
2. Mencari halaman/page di server.

commit to user

3. Melakukan instruksi yang diberikan oleh PHP untuk melakukan modifikasi pada halaman/page.
4. Mengirim kembali halaman tersebut kepada client melalui internet atau intranet. (Peranginangin,2006).

2.16 Yii Framework

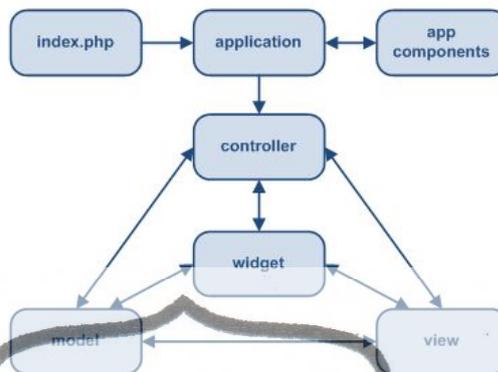
Yii adalah *framework* (kerangka kerja) PHP berbasis-komponen, berkinerja tinggi untuk pengembangan aplikasi Web berskala-besar. Yii menyediakan *reusability* maksimum dalam pemrograman Web dan mampu meningkatkan kecepatan pengembangan secara signifikan. Nama Yii (dieja sebagai /i:/) singkatan dari "*Yes It Is!*". Berikut merupakan respon yang paling tepat dan akurat untuk yang baru dengan Yii:

Yii mengimplementasikan pola desain *model-view-controller* (MVC), yang diadopsi secara luas dalam pemrograman Web. MVC bertujuan untuk memisahkan logika bisnis dari pertimbangan antar muka pengguna agar para pengembang bisa lebih mudah mengubah setiap bagian tanpa mempengaruhi yang lain. Dalam MVC, *model* menggambarkan informasi (data) dan aturan bisnis; *view* (tampilan) berisi elemen antar muka pengguna seperti teks, input form; sementara *controller* mengatur komunikasi antar *model* dan *view*.

Selain implementasi MVC, Yii juga memperkenalkan *front-controller*(*controller*-depan), yang disebut Aplikasi, yang mengenkapsulasi konteks eksekusi untuk memproses sebuah *request*. Aplikasi mengumpulkan beberapa informasi mengenai request pengguna dan kemudian mengirimnya ke *controller* yang sesuai untuk penanganan selanjutnya.

Diagram berikut memperlihatkan struktur statis sebuah aplikasi Yii:

Struktur statis aplikasi Yii

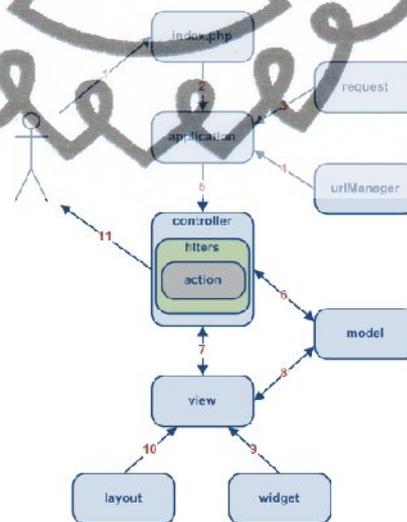


Gambar 2.3 Struktur Statis Yii

Alur kerja Umum

Diagram berikut memperlihatkan alur kerja umum sebuah aplikasi Yii saat menangani permintaan pengguna:

Alur kerja umum aplikasi Yii



Gambar 2.4 Alur Kerja Umum Yii

1. Pengguna membuat permintaan dengan URL `http://www.example.com/index.php?r=post/show&id=1` dan

commit to user

server web menangani permintaan dengan menjalankan skrip bootstrapindex.php.

2. Skrip bootstrap membuat sebuah instance aplikasi dan menjalankannya.
3. Aplikasi mendapatkan rincian informasi permintaan pengguna dari komponen aplikasi bernama request.
4. Aplikasi menentukan controller dan aksi yang diminta dengan bantuan komponen aplikasi bernama urlManager. Dalam contoh ini, controller adalah post yang merujuk pada kelas PostController; dan aksi adalah show yang arti sebenarnya ditentukan oleh controller.
5. Aplikasi membuat instance controller yang diminta untuk selanjutnya menangani permintaan pengguna. Controller menentukan aksi show merujuk pada sebuah metode bernama actionShow dalam kelas controller. Kemudian membuat dan menjalankan filter (contoh kontrol akses, pengukuran) terkait dengan aksi ini. Aksi dijalankan jika diijinkan oleh filter.
6. Aksi membaca Post model di mana ID adalah 1 dari database.
7. Aksi menyiapkan view(tampilan) bernama show dengan model Post.
8. View membaca dan menampilkan atribut model Post.
9. View menjalankan beberapa widget.
10. View menyiapkan hasil yang dipasangkan dalam layout(tata letak).
11. Aksi mengakhiri pembuatan view dan menampilkan hasil akhir kepada pengguna

(<http://www.yiiframework.com/doc/guide/1.1/id/quickstart.what-is-yii>, diakses pada 30 September 2013).

2.17 CSS (*Cascading Style Sheet*)

Merupakan suatu teknik design web yang memadukan antara html dengan style sheet. CSS menjadikan halaman web lebih interaktif, menarik, dan elegan hanya dengan menambahkan beberapa baris kode di dalamnya (Nugroho,2004).

2.18 SQLYog

SQLYog adalah salah satu *tool* administrasi untuk database MySQL. Jika kita biasanya menggunakan PhpMyAdmin yang *include* di dalam aplikasi Xampp untuk melakukan administrasi database, SQLYog adalah aplikasi alternatif untuk melakukan proses administrasi database MySQL. Banyak fitur yang disediakan oleh SQLYog yang tidak disediakan oleh PhpMyAdmin maupun *tool* administrasi database lainnya seperti MySQLQueryBrowser.

(<https://www.webyog.com/product/sqlyog>, diakses pada 30 September 2013).

2.19 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk pengembangan java/android yang *free*. Pada Eclipse terdapat *Android Development Tools* (ADT) yang merupakan *plugin* yang harus diinstal di Eclipse sehingga Android SDK yang sudah kita miliki dapat dihubungkan dengan IDE Eclipse dimana digunakan sebagai tempat coding aplikasi Android nantinya (Safaat, 2011).

2.20 Netbeans

NetBeans merupakan sebuah proyek kode terbuka yang sukses dengan pengguna yang sangat luas, komunitas yang terus tumbuh, dan memiliki hampir 100 mitra (dan terus bertambah!). Sun Microsystems mendirikan proyek kode terbuka NetBeans pada bulan Juni 2000 dan terus menjadi sponsor utama.

Saat ini terdapat dua produk : NetBeans IDE dan NetBeans Platform. The NetBeans IDE adalah sebuah lingkungan pengembangan - sebuah kakas untuk pemrogram menulis, mengompilasi, mencari kesalahan dan menyebarkan program. Netbeans IDE ditulis dalam Java - namun dapat mendukung bahasa pemrograman lain. Terdapat banyak modul untuk memperluas Netbeans IDE. Netbeans IDE adalah sebuah produk bebas dengan tanpa batasan bagaimana digunakan.

Tersedia juga NetBeans Platform; sebuah fondasi yang modular dan dapat diperluas yang dapat digunakan sebagai perangkat lunak dasar untuk membuat aplikasi desktop yang besar. Mitra ISV menyediakan plug-in bernilai tambah yang

commit to user

dapat dengan mudah diintegrasikan ke dalam Platform dan dapat juga digunakan untuk membuat kaskas dan solusi sendiri.

Kedua produk adalah kode terbuka (*open source*) dan bebas (*free*) untuk penggunaan komersial dan non komersial. Kode sumber tersedia untuk guna ulang dengan lisensi *Common Development and Distribution License (CDDL)* (https://netbeans.org/index_id.html, diakses pada 30 September 2013).

2.21 XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri atas program Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam GNU General Public License dan bebas, merupakan web server yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis (Hakim,2008).

2.22 Adobe Photoshop

Adobe Photoshop, atau biasa disebut photoshop, merupakan *software* yang digunakan untuk desain grafis berbasis bitmap yang populer yang dikhususkan untuk pengeditan foto atau gambar dan pembuatan efek. Software ini digunakan oleh berbagai orang, mulai dari para desainer grafis profesional, desainer web, fotografer, para pekerja kantor, hingga pemula di bidang desain, semuanya mengakui *software* ini sebagai program pengolah gambar yang bisa diandalkan.

Fitur dan fasilitas Adobe Photoshop dikemas dalam *interface* yang *user-friendly* dan *fleksibel* untuk bekerja sama dengan berbagai software lain, baik untuk kepentingan *desktop publishing* maupun *printing*, menjadikan tiap versi program ini selalu dinanti-nanti (Suprayogo, 2004).

commit to user

2.23 Rational Rose

Rational Rose adalah *software* yang memiliki perangkat-perangkat pemodelan secara visual untuk membangun suatu solusi dalam rekayasa *software* dan pemodelan bisnis. *Rational Rose* dikeluarkan oleh perusahaan *software* bernama Rational Software, perusahaan yang mencetuskan ide pembentukan konsorsium bagi perusahaan-perusahaan yang memakai standar UML sebagai bahasa pemodelan di perusahaannya. *Rational Rose* memakai UML sebagai bahasa pemodelannya, ditambah beberapa fitur lain yang membuat *Rational Rose* menjadi *software* pemodelan visual yang terkemuka.

Beberapa fitur terkemuka diantaranya *Rational Rose* memiliki *Rational Unified Process* (RUP) yaitu proses yang paling terperinci yang ada saat ini dan akan memberikan pedoman secara umum dalam pembuatan *software* dan pemodelan bisnis. Selain itu, *Rational Rose* memiliki kemampuan membuat solusi client/server, yang kemudian dapat diterapkan dan didistribusikan dalam lingkungan perusahaan.

Keunggulan *Rational Rose*, diantaranya :

1. Bahasa yang digunakan adalah bahasa pemodelan standar yaitu UML, akan meningkatkan komunikasi intra tim.
2. *Rational Rose* mendukung *round-trip engineering* sehingga kita dapat meng-*generate* model kedalam kode (Java, C++, Visual Basic, dan sebagainya) dan melakukan *reverse engineering* untuk menampilkan arsitektur *software* dari kode yang ada. Hal ini dapat dilakukan secara bolak-balik sebagai proses iterative selama proses rekayasa *software*.
3. Model dan kode senantiasa sinkron selama dalam *development cycle*.
4. Membangun *software* menggunakan *Rational Rose* memudahkan dalam memperbaiki *software* tersebut karena apabila suatu saat ditemukan *requirement* baru, kita dapat lagi menggambarkan lagi *software* tersebut dalam UML.
5. Para user *Rational Rose* dapat berkomunikasi walaupun bekerja dalam sistem operasi yang berbeda (Windows atau UNIX) (Nugroho, 2005).

6. Dengan menggunakan Rose Web Publisher suatu tim dapat mengkomunikasikan model dan spesifikasinya dalam web browser.
7. Mendukung rekayasa software untuk sistem client/server sehingga Rational Rose merupakan software pemodelan visual yang tangguh dalam lingkungan client/server, e-business, dan lingkungan perusahaan terdistribusi (kantor-kantor terletak dalam tempat yang berbeda-beda).

2.24 Php MyAdmin

PhpMyAdmin disebut juga sebagai *tools* yang digunakan untuk mengakses *database* MySQL dalam bentuk tampilan *web*. *Tools* ini secara standar disertakan ketika menginstal XAMPP. PhpMyAdmin adalah suatu program Open Source berbasis web. Program ini berguna untuk mengakses database MySQL (Nugroho,2005).

2.25 Google Form

Membuat dan berkolaborasi dengan orang lain, berbagi dokumen, membangun spreadsheet dan membuat presentasi dengan cepat dengan Docs, Sheets dan aplikasi Slides. Quickoffice memungkinkan membuka dan mengedit dokumen Microsoft Office, spreadsheet dan presentasi di ponsel atau tablet. Google Forms memungkinkan menjalankan survei atau cepat membuat daftar tim dengan formulir online sederhana. Kemudian memeriksa hasil, disusun rapi dalam sebuah spreadsheet (<http://www.google.com/drive/using-drive/>, diakses pada 13 Juli 2014).

2.26 Genymotion

Genymotion adalah emulator x86 menggunakan virtualisasi arsitektur, sehingga jauh lebih efisien. Mengambil keuntungan dari akselerasi hardware OpenGL, memungkinkan untuk menguji aplikasi dengan kinerja 3D yang

commit to user

menakjubkan. Terintegrasi sempurna dalam lingkungan pengembangan dengan plugin Eclipse dengan interface user-friendly. Dikemas untuk Windows, Mac dan Linux. Kontrol simulasi sensor seperti baterai, GPS, dan accelerometer dengan antarmuka yang user-friendly (<http://www.genymotion.com>, diakses pada 13 Juli 2014).

2.27 Pseudocode

Pengertian pseudocode menurut (Waluya, 1997), “Pseudocode menggunakan kode instruksi program untuk member tanda aliran logika. Tidak seperti flowchart, pseudocode menggunakan kata-kata dari pada menggambarkan symbol untuk menunjukkan setiap langkah yang harus dilaksanakan”.

