BAB II

TINJAUAN PUSTAKA

II.1.PT. Pelabuhan Indonesia I (Persero)

PT. Pelabuhan Indonesia I (Persero) didirikan berdasarkan Peraturan Pemerintahan No. 56 Tahun 1991 dengan akte Notaris Imas Fatimah, SH No. 1 tanggal 1 Desember 1992 sebagaimana dimuat dalam Tambahan Berita Negara RI No. 8612 Tahun 1994, beserta perubahan terakhir sebagaimana telah diumumkan dalam Tambahan Berita Negara RI tanggal 2 Januari 1999 No. 1.

Nama lengkap Perusahaan adalah PT. Pelabuhan Indonesia I (Persero) disingkat PT. Pelindo I (Persero), berkantor pusat di Jl. Krakatau Ujung No.100 Medan 20241, Sumatera Utara.

II.1.1.Visi, Misi dan Value Perusahaan

Visi dari PT. Pelindo I (Persero) adalah "Menjadi penyedia jasa kepelabuhan dan logistik terkemuka di tingkat regional", dan misi yang dilakukan oleh perusahan dalam upaya mencapai visinya adalah " Menyediakan jasa kepelabuhan dan logistik yang memenuhi harapan pelanggan dan memberikan nilai tambahan bagi pertubuhan ekonomi wilayah". (Personal Agenda, PT.Pelabuhan Indonesia I)

Untuk mendukung pencapaian visi dan misi, perusahaan memiliki *values* sebagai berikut :

1. *Custumer focus* : Menyediakan jasa layanan yang fokus kepada pelanggan.
2. *Leadership* : Sistem kepemimpinan dan SDM mampu menjamin efektifitas dan kualitas pemimpin dan personil untuk merealisir *custumer focus* serta *excelent operation*.
3. Inovasi : Membuat perubahan berarti untuk meningkatkan pelayanan dalam upaya menciptakan *new value* bagi *stake holder*.
4. *Valuing emplayee* : Komitmen manajemen atas kepuasan pengembangan dan perilaku yang baik bagi pegawai.
5. *System perspective* : Pengelolaan perusahaan sebagai sebuah sistem yang utuh sehingga pencapaian kesuksesan pengelolaan organisasi meliputi keseluruhan komponen organisasi tersebut.

Untuk mencapai maksud dan tujuan tersebut, PT. Pelabuhan Indonesia I(Persero) dapat melaksanakan kegiatan usaha utama sesuai Anggaran Dasar Perusahaan sebagai berikut :

1. Penyediaan dan/atau pelayanan kolam-kolam lintas dan tempat berlabuhannya kapal.
2. Penyediaan dan/atau pelayanan jasa-jasa yang berhubungan pemanduan (*pilotage*) dan penundaan kapal.
3. Penyediaan dan/atau pelayanan dermaga dan fasilitas lain untuk bertambat, bongkar muat peti kemas, curah cair, curah kering, multi *purpose*, barang termasuk hewan (*general cargo*), dan fasilitas naik turunnya penumpang dan/atau kendaraan.

4. Penyediaan pelayanan jasa bongkar muat, peti kemas, curah cair, curah kering (*general cargo*) dan kendaraan.
5. Penyedia dan/atau pelayanan jasa terminal peti kemas, curah cair curah kering, multi purpose penumpang pelayaran rakyat dan Ro-Ro.
6. Penyedia dan/atau pelayanan gudang-gudang dan lapangan penumpukan dan tangki/tempat penimbunan barang-barang, angkutan bandar , alat bongkat muat, serta peralatan pelabuhan.

II.2.Penjadwalan

Salah satu masalah yang cukup penting dalam sistem produksi adalah bagaimana melakukan penjadwalan pembuatan produk (pekerjaan), agar pesanan dapat selesai sesuai dengan kontrak dan sumber-sumber daya yangtersedia dapat dimanfaatkan seoptimal mungkin.

Masalah penjadwalan seringkali muncul jika terdapat n pekerjaan yang akan diproses pada m mesin, yang harus ditetapkan mana yang harus dikerjakan lebih dahulu dan pengalokasian operasi pada mesin sehingga diperoleh suatuproses produksi yang terjadwal.

Penjadwalan adalah pengurutan pembuatan atau pengerjaan produk secara menyeluruh yang dikerjakan pada beberapa buah mesin. Tujuan dengan penjadwalan adalah meningkatkan penggunaan sumber daya, mengurangi persediaan barang setengah jadi atau sejumlah pekerjaan yang menunggu dalam antrian, dan mengurangi keterlambatan pada pekerjaan yang mempunyai batas

waktu penyelesaian.(Andri Sulaksmi, Annisa Kesy Garside, Dan Fithriany Hadziqah : 2014: 35-36)

II.3.*Java*

Java merupakan bahasa pemrograman untuk membangun aplikasi pada sistem operasi *Android*.Oleh karena itu, untuk membangun aplikasi pada sistem operasi *android* diperlukan dasar tentang pemrograman *java*.*Java* merupakan pemrograman berorientasi objek. Oleh karena itu, setiap konsep yang akan diimplementasikan dalam *java* berbentuk dalam kelas. Kelas ini mendefinisikan objek-objek yang memiliki kesamaan perilaku dan keadaan. Pada *java* terdapat kumpulan kelas standar yang dikenal dengan *Application Programming Interface (API) java*, selain itu dapat juga dideskripsikan kelas sendiri sesuai kebutuhan. *java* mengadopsi konsep *smart client*, konsep ini berbicara tentang kemampuan *client* untuk meminta dan menangkap pesan dari *server*. Dalam proses meminta dan menangkap itu *client* melakukan proses validasi dan verifikasi data.

Salah satu kelebihan *java* yang paling signifikan adalah *run everywhere*. Dengan kelebihan ini, para *developer* yang sudah terbiasa mengembangkan aplikasi dalam bingkai kerja *J2SE* dan *J2EE*, akan mampu bermigrasi dengan mudah untuk mengembangkan aplikasi *J2ME*. (Miftakhul Huda :2010 :9-10).

II.4. Android

Android adalah sebuah sistem operasi *mobile* yang berbasiskan pada versi modifikasi dari *Linux*.Pertama kali sistem operasi ini dikembangkan oleh

perusahaan Android.Inc.nama perusahaan inilah yang pada akhirnya digunakan sebagai nama proyek sistem operasi *mobile* tersebut, yaitu sistem operasi Android.

Pada tahun 2005, sebagian dari strategi untuk memasuki pasar *mobile*, Google membeli Android dan mengambil alih proses pengembangannya sekaligus *team developer Android*. Google menginginkan Android untuk menjadi sistem operasi *Open Source* dan gratis, kebanyakan *code* Android dirilis diabawah lisensi *Open Source Apache*yang berarti setiap orang bebas untuk menggunakan dan mengunduh *source code Android* secara penuh.

Android telah dikembangkan dan diupdate beberapa kali sejak rilis pertamanya.*Table* dibawah ini memperlihatkan versi Android semenjak pertama kali dirilis (*Wahana Komputer : 2013: 2 -3*)

Tabel II.1. Versi Android
(sumber : Wahana Komputer: 2013: 3)

Versi Android	Tanggal Rilis	Nama Kode
1.1	9 Februari 2009	-
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0 / 2.1	26 Oktober 2009	Éclair
2.2	20 Mei 2010	Froyo
2.3	6 Desember 2010	Gingerbread
3.0	Tidak Diketahui	Honeycomb
4.0	19 Oktober 2011	Ice Cream Sandwich

II.4.1. Aplikasi *Native*

Aplikasi *native* adalah aplikasi yang secara khusus ditujukan untuk *platform mobile* tertentu dan menggunakan bahasa pemrograman serta perangkat

lunak pengembangan sesuai dengan *platform* tersebut. Sebagai contoh , aplikasi *native Android* ditulis menggunakan bahasa pemrograman *Java* dan *Tool Eclipse*, sementara aplikasi iOS/iPhone dibuat dengan menggunakan bahasa *Objective-C* dan *tool Xcode. (*sumber : Didik Dwi Prasetsya : 2013 : 2)

Kelebihan :

1. Perfoma yang sangat baik karena ditulis secara *native* untuk *platform* spesifik.
2. Mampu mengakses semua fitur perangkat keras *smartphone*, seperti *info device, accelerator,* kamera, kompas, *file* dan lain sebagainya.
3. Menghasilkan antarmuka *look and feel* yang alami dengan sangat baik.

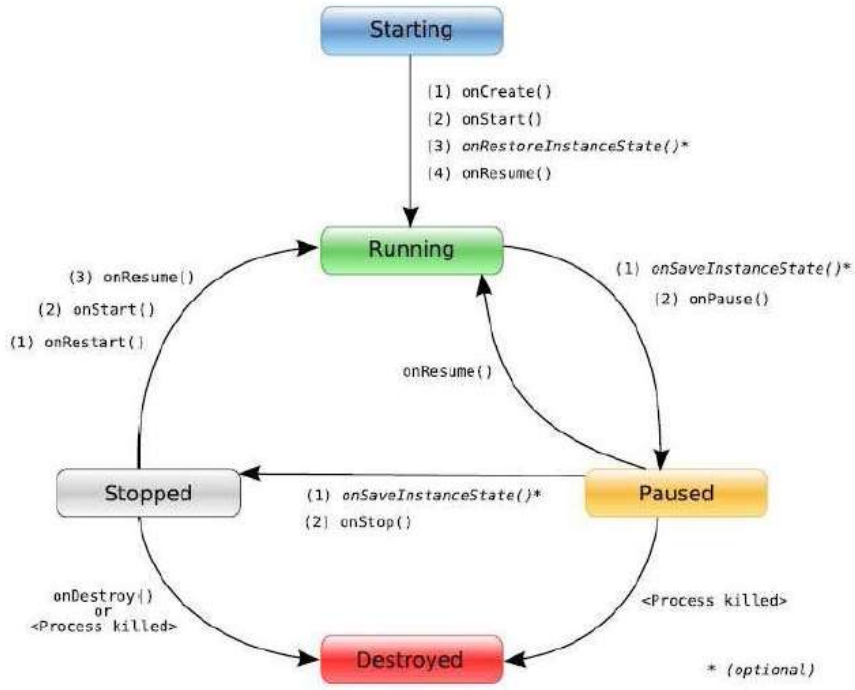
Kekurangan :

1. Pengembangan yang tidak mudah karena menggunakan lingkungan, bahasa dan API (*Application Programming Interface*) spesifik.
2. Aplikasi hanya berjalan pada *platform* yang sudah dispesifikasikan diawal pengembangan. Apabila ingin dikembangkan diplatform lain maka harus ditulis ulang dengan *tool* pengembangan yang sesuai.

II.4.2. Tool Pengembangan Android

Untuk mengembangkan sistem operasi Android dapat menggunkan Mac, Windows atau PC.*Linux,* semua *tool* yang dibutuhkan adalah gratis dan dapat di *download* dari web.

1. Java JDK : Andorid berjalan dengan menggunakan *resource* dari *Java SE Development Kit* (JDK).
2. Android SDK : Android SDK berikan *Debugger*, *library*, dokumentasi, kode contoh dan tutorial. Andorid SDK dapat didownload dari alamat :http://developer.andorid/sdk/index.html .
3. *Andorid Development Tools* (ADT) :*Plug-in Android Development Tools* (ADT) untuk mendukung pembuatan dan ;proses *debugging* dari aplikasi Android yang sedang buat.



Gambar II.1. Siklus Hidup Android

II.5. Definisi Aplikasi *Mobile*

Mobile dapat diartikan sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal

telepon yang dapat berpindah dengan mudah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi.

Aplikasi *mobile* merupakan aplikasi yang dapat digunakan walaupun pengguna berpindah dengan mudah dari satu tempat ketempat lain tanpa terjadi pemutusan atau terputusnya komunikasi. Aplikasi ini dapat diakses melalui perangkat nirkabel seperti pager, seperti telepon seluler, PDA, serta *Smartphone.*(*Hinova Rezha Ulinuha, 2013, 1-2*).

Adapun karakteristik perangkat *mobile* yaitu :

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*).
3. Daya proses yang terbatas : Sistem *mobile* tidaklah setangguh rekan mereka yaitu *desktop.*
4. Mengkonsumsi daya yang rendah : Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin *desktop.*
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.

II.6. Basis Data Terdistribusi

Sebuah sistem basis data terdistribusi hanya mungkin dibangun dalam sebuah sistem jaringan komputer. Dalam sebuah sistem jaringan komputer kita mengenal adanya topologi, yang akan menentukan bagaimana konfigurasi/keterhubungan antara satu simpul jaringan (*node/site*) dengan simpul-simpul lainnya. Setiap simpul, dalam kaitannya dengan sistem basis data terdistribusi mewakili sebuah *server*, yang memiliki *disk* dengan sistem basis data sendiri (lokal).Setiap *server* juga membentuk sebuah LAN (*Local Area Network*) sendiri untuk mengakomodasi sejumlah *workstation* dan sekaligus *user* lokal.(Fathansyah, 2012,337-338).

Dalam sebuah sistem data terdistribusi seperti itu ada 2 jenis transaksi yang mungkin terjadi:

1. Transaksi lokal. Transaksi yang mengakses basis data di *server* yang sama dengan *server* dari mana transaksi tersebut di jalankan.
2. Transaksi global. Transaksi yang membutuhkan pengaksesan data di *server* yang berbeda dengan *server* di mana transaksi tersebut dijalankan, atau transaksi dari sebuah *server* yang membutuhkan pengaksesan data ke sejumlah *server* lainnya

Keuntungan basis data terdistribusi:

1. Pembagian dan pemakaian data dan *control* yang tersebar.

Setiap *user* pada suatu lokasi dapat mengakses data yang berada di lokasi lainnya, sama halnya dengan *user-user* pada lokasi tempat data tersebut berada.

2. Keandalan dan Ketersediaan

   Jika ada sebuah simpul/lokasi mengalami kerusakan, simpul/lokasi yang lain akan tetap dapat beroperasi. Apalagi jika di dalam sebuah sistem terdistribusi digunakan mekanisme replikasi, maka ketersediaan data akan semakin tinggi.

3. Kecepatan Query

   Jika sebuah *query* melibatkan data di sejumlah simpul/lokasi, maka *query* tersebut dapat dipilah ke sejumlah *subquery* yang akan dijalankan di simpul-simpul yang bersesuaian.

Sedang kelemahan utama sistem basis data terdistribusi terletak pada meningkatnya kompleksitas yang diperlukan untuk menjamin koordinasi yang baik di antara simpul-simpul yang terlibat. Peningkatan kompleksitas ini berbentuk:

1. Biaya pembangunan perangkat lunak.

   Implementasi sistem basis data terdistribusi tentu akan lebih sukar, sehingga perlu biaya lebih besar.

2. Potensi sumber kesalahan program (*bug)* yang lebih banyak.

   Karena simpul-simpul dalam sistem basis data terdistribusi beroperasi secara parallel, maka akan lebih sulit menjamin kebenaran algoritma/program.

3. Peningkatan waktu proses (*overhead)*

   Waktu untuk pertukaran data dan tambahan komputasi yang diperlukan untuk mengupayakan koordinasi antarsimpul merupakan beban tambahan (*overhead)* yang tidak dijumpai dalam sistem terpusat.

II. 7.*Database*

Database adalah sekumpulan data yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. *Database* merupakan kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (*redundansi*) yang tidak perlu, untuk memenuhi berbagai kebutuhan. (Fathansyah, 2012:3)

II. 7. 1. MySQL

MySQL adalah suatu perangkat lunak database relasi (*Relational Database Management System* atau RDBMS), seperti halnya ORACLE, Postgresql, MS SQL , dan sebagainya. MySQL AB menyebut produknya sebagai *database open source* terpopuler di dunia. Berdasarkan riset dinyatakan bahwa bahwa di *platform* Web, dan baik untuk kategori *open source* maupun umum, MySQL adalah *database* yang paling banyak dipakai. Menurut perusahaan pengembangnya, MySQL telah terpasang di sekitar 3 juta komputer.Puluhan hingga ratusan ribu situs mengandalkan MySQL bekerja siang malam memompa data bagi para pengunjungnya.Penyebab utama MySQL begitu popular di kalangan Web adalah karena memang cocok bekerja di lingkungan tersebut.*(Iwan Sapta Yulianto, 2014: 3-4)*

1. Pertama, MySQL tersedia di berbagai *platform Linux* dan berbagai varian Unix.
2. Fitur-fitur yang dimiliki MySQL memang yang biasanya banyak dibutuhkan dalam aplikasi Web. Misalnya, klausa LIMIT SQL-nya, praktis untuk melakukan *paging*. Atau jenis indeks *field* FULLTEXT, untuk *full text searching*. Atau sebutlah kekayaaan fungsi-fungsi builtinnya, mulai dari memformat dan memanipulasi tanggal, mengolah string, regex, *enkripsi* dan *hashing*.
3. MySQL memiliki *overhead* koneksi yang rendah. Soal kecepatan melakukan transaksi atau kinerja di kondisi *load* tinggi mungkin bisa diperdebatkan dengan berbagai *benchmark* berbeda, tapi kalau soal yang satu ini MySQL-lah juaranya. Karakteristik ini membuat MySQL cocok bekerja dengan aplikasi CGI, dimana di setiap request skrip akan melakukan koneksi, mengirimkan satu atau lebih perintah SQL, lalu memutuskan koneksi lagi. Cobalah melakukan hal ini dengan *Interbase* atau bahkan *Oracle*. Maka dengan *load* beberapa *request* per detik saja *serverWeb/database* Anda mungkin akan segera menyerah karena tidak bisa mengimbangi beban ini.

Dalam bahasa SQL pada umumnya informasi tersimpan dalam tabel- tabel yang secara logik merupakan struktur dua dimensi terdiri dari baris (*row* atau *record*) dan kolom(*column* atau *field*).Sedangkan dalam sebuah *database* dapat terdiri dari beberapa *table*.

II.8.Aplikasi Berbasis Client-Server

Client-server adalah model pemrograman yang membutuhkan komputer induk (*Server*) dan PC untuk *user* (*Client*). Model tersebut digunakan untuk bertukar data. Koneksi *client-server* biasa digunakan pemrograman *socket* yang telah dikembangkan oleh *Berkeley University* yang mempunyai IP (*Internet Protocol*) *address* dan *Port*(nomor *id* untuk dalam koneksi). *Socket* sudah menjadi bagian utama dalam jaringan seperti WSA (*Window Socket Agent*) untuk *Windows*, *Java Socket* untuk *Java* dan *Sock32* untuk pemrograman berbasis DOS. Telepon GSM (*Groupe Speciale Mobile*) digunakan oleh lebih dari satu milyar orang di lebih dari 200 negara. GSMdalam pensinyalan dan *channel* (saluran data) pembicaraan adalah digital sehingga muncul teknologi *General Paket Radio Service* (GPRS) yang merupakan sebuah layanan data untuk *mobile device*. Kelas GPRSsudah mencapai class10 yang mencapai *transfer rate* mulai dari 56 kbps-114 kbps(*kilo bit per second*) . *Mobile Information Device Profile* digunakan menambahkan fungsi *networking*, menyediakan standar komponen *user interface*, dan penyimpanan lokal untuk CLDC. Profil ini terbatas pada tampilan dan kemampuan penyimpanan pada *device*, menggunakan *networking* HTTP 1.1.

Connected Device Configuration/Connected Limited Device configuration mendefinisikan kebutuhan minimum *Java Libraries* dan kapabilitas yang dipunyai oleh para *developer* J2ME. CLDC ditujukan untuk pengunaan kelas bawah dari suatu perangkat elektronik, dengan pengalokasian memori sekitar 512 KB. Oleh karena itu CLDC ditujukan untuk *wireless java* seperti

handphone yang membuat *user* mampu mengunakan MIDlet. CDC digunakan mampu menjalankan aplikasi J2SE untuk perangkat yang mempunyai memori lebih dari 2 MB dan memiliki banyak kemampuan proses pada processor-nya. Biasa ditemui pada *high-end PDA, smart phone,*dan *web telephone* . (*Wiranto Herry Utomo, Theophilus Wellem, Tommy Bulyan : 2009 : 166*)

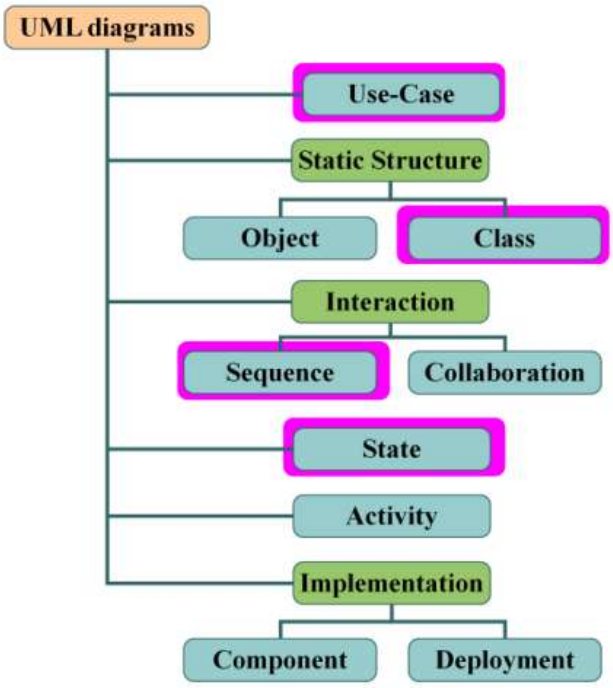
II.9. UML*(Unified Modelling Language)*

Unified Modelling Language merupakan alat perancangan sistem yang berorientasi pada objek. Secara filosofi kemunculan UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* (OO), karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik maka OO memiliki proses standard dan bersifat independen.

UML diagram memiliki tujuan utama untuk membantu tim pengembangan proyek berkomunikasi, mengeksplorasi potensi desain, dan memvalidasi *desain* arsitektur perangkat lunak atau pembuat program. Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu *Grady Booch*, OOD (*Object-Oriented Design*), Jim Rumbaugh, OMT (*Object Modelling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

UML mempunyai tiga kategori utama yaitu *struktur diagram, behaviour diagram dan interaction diagram*. Dimana masing-masing kategori tersebut memiliki diagram yang menjelaskan arsitektur sistem dan saling terintegrasi. (*Haviluddin , 2011 ; 1*)

Secara filosofi UML diilhami oleh konsep yang telah ada yaitu konsep permodelan *Object Oriented* karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik. Berikut gambar dari diagram UML



Gambar II.2. Diagram UML
(*Haviluddin , 2011 ; 2*)

II.9.1. Komponen-komponen UML

Sejauh ini para pakar merasa lebih mudah dalam menganalisa dan mendesain atau memodelkan suatu sistem karena UML memiliki seperangkat aturan dan notasi dalam bentuk grafis yang cukup spesifik.

Komponen atau notasi UML diturunkan dari 3 (tiga) notasi yang telah ada sebelumnya yaitu *Grady Booch*, OOD (*Object-Oriented Design*), *Jim*

Rumbaugh, OMT (*Object Modelling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). (*Haviluddin, 2011 ; 3*)

Pada UML versi 2 terdiri atas tiga kategori dan memiliki 13 jenis diagram yaitu :

1. Struktur Diagram

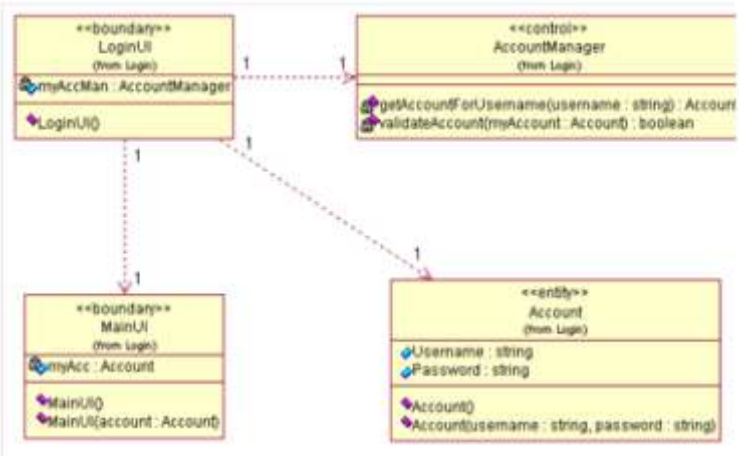
Menggambarkan elemen dari spesifikasi dimulai dengan kelas, obyek, dan hubungan mereka, dan beralih ke dokumen arsitektur logis dari suatu sistem. Struktur diagram dalamUML terdiri atas :

a. Class Diagram

Class diagram menggambarkan struktur statis dari kelas dalam sistem anda dan menggambarkan atribut, operasi dan hubungan antara kelas. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Class memiliki tiga area pokok :

a. Nama (dan *stereotype*)
b. Atribut
c. Metoda

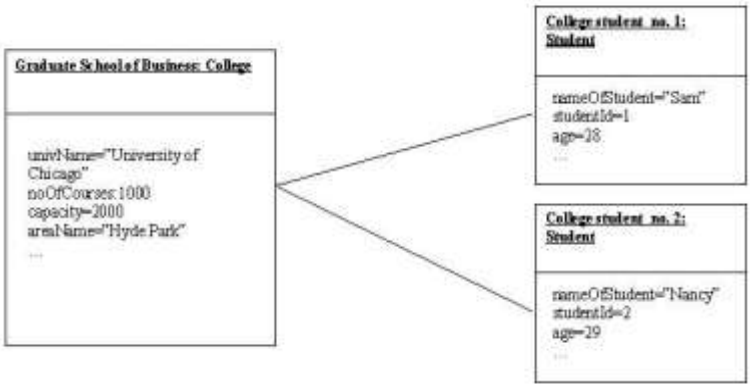


II.3. Contoh Notasi Class Diagram
(*Sumber : Haciluddin , 2011 ; 3*)

b. Object diagram

Object diagram menggambarkan kejelasan kelas dan warisan dan kadang-kadang diambil ketika merencanakan kelas, atau untuk membantu pemangku kepentingan non-program yang mungkin menemukan diagram kelas terlalu abstrak.

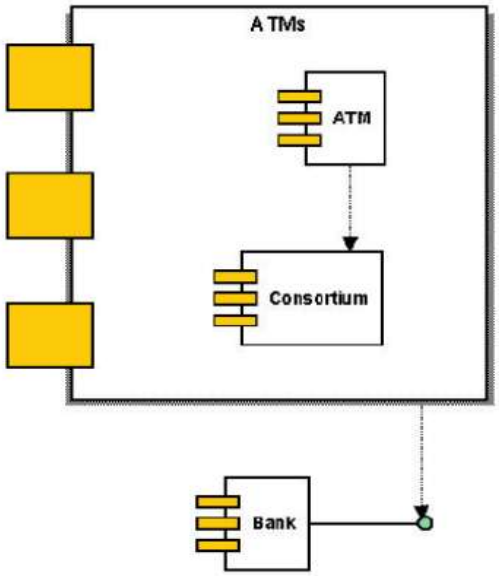
Berikut notasi *object diagram :*



II.4. Contoh Notasi *Object Diagram*
(*Sumber : Haviluddin , 2011 ; 3*)

c. Component diagram

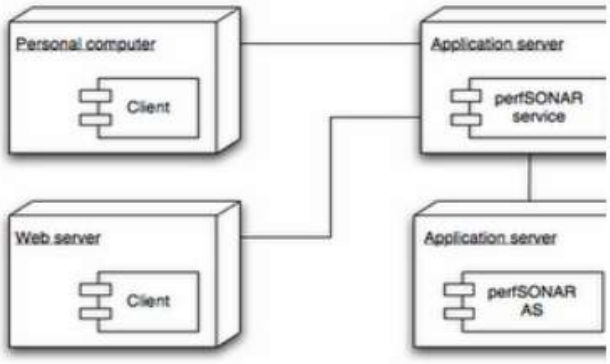
Component diagram menggambarkan struktur fisik dari kode, pemetaan pandangan logis dari kelas proyek untuk kode aktual di mana logika ini dilaksanakan.



II.5. Contoh Notasi *Object Diagram*
(*Sumber : Haviluddin , 2011 ; 3*)

d. Deployment diagram

Deployment diagram memberikan gambaran dari arsitektur fisik perangkat lunak, perangkat keras, dan artefak dari sistem. *Deployment diagram* dapat dianggap sebagai ujung spektrum dari kasus penggunaan, menggambarkan bentuk fisik dari sistem yang bertentangan dengan gambar konseptual dari pengguna dan perangkat berinteraksi dengan sistem.



II.6. Contoh Notasi *Deployment Diagram*
(*Sumber : Haviluddin , 2011 ; 4*)

e. Composite structure diagram

Sebuah diagram struktur komposit mirip dengan diagram kelas, tetapi menggambarkan bagian individu, bukan seluruh kelas. Kita dapat menambahkan konektor untuk menghubungkan dua atau lebih bagian dalam atau ketergantungan hubungan asosiasi.

f. Package diagram

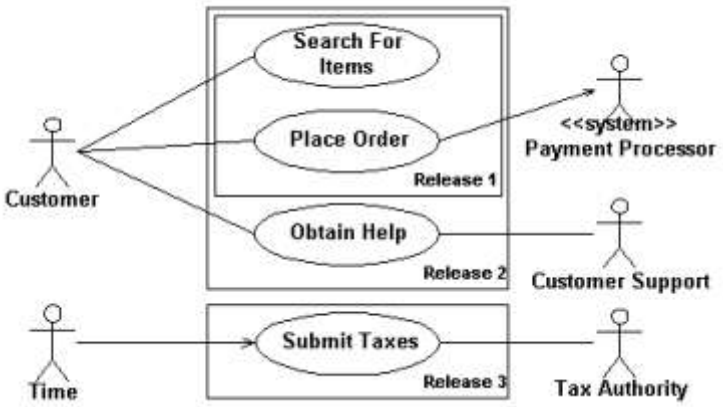
Paket diagram biasanya digunakan untuk menggambarkan tingkat organisasi yang tinggi dari suatu proyek *software*. Atau dengan kata lain untuk menghasilkan diagram ketergantungan paket untuk setiap paket dalam Pohon Model.

2. Behavior Diagram

a. Usecase Diagram

Diagram yang menggambarkan aktor, *use case* dan relasinya sebagai suatu urutan tindakan yang memberikan nilai terukur

untuk aktor. Sebuah *use case* digambarkan sebagai *elips horizontal* dalam suatu diagram UML *use case.*



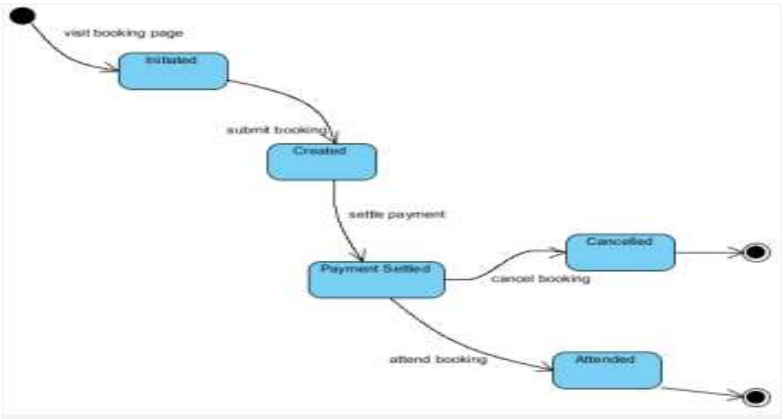
II.7. Contoh Notasi *Usecase Diagram*
(*Sumber : Haviluddin , 2011 ; 4*)

b. Activity diagram

Menggambarkan aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas.

c. State Machine diagram

Menggambarkan state, transisi state dan event.



II.8. Contoh Notasi *State Machine Diagram*
(*Sumber : Haviluddin , 2011 ; 4*)

3. Interaction diagram

a. Communication diagram

Serupa dengan *sequence* diagram, tetapi diagram komunikasi juga digunakan untuk memodelkan perilaku dinamis dari *use case*. Bila dibandingkan dengan *Sequence* diagram, diagram komunikasi lebih terfokus pada menampilkan kolaborasi benda daripada urutan waktu.

b. Interaction Overview diagram

Interaksi *overview* diagram berfokus pada gambaran aliran kendali interaksi dimana node adalah interaksi atau kejadian interaksi.

c. Sequence diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram.*

d. Timing diagram

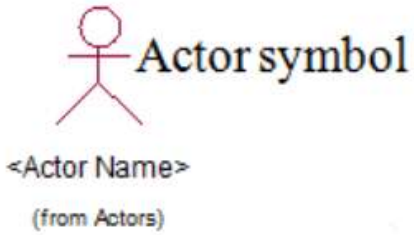
Timing diagram di UML didasarkan pada diagram waktu hardware awalnya dikembangkan oleh para insinyur listrik.

(Haviluddin , 2011 ; 3-5)

Untuk menggambarkan analisa dan desain diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu

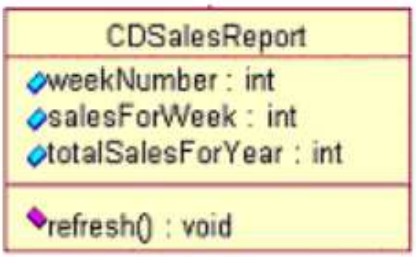
struktur diagram, *behaviour* diagram dan *interaction* diagram. Berikut beberapa notasi dalam UML diantaranya :

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.



II.9. Notasi *Actor*
(*Sumber : Haviluddin , 2011 ; 6*)

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.



II.10. Notasi *Class*
(*Sumber : Haviluddin , 2011 ; 6*)

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.
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.
5. *Interaction, Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar obyek maupun hubungan antar obyek.
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. (*Haviluddin , 2011 ; 6-7*)