BAB II

LANDASAN TEORI

2.1 Administrasi Pengiriman Barang

Menurut Suyono (2003:155) pengertian *freight forwarding* (jasa pengiriman barang) adalah badan usaha yang bertujuan memberikan jasa pelayanan/pengurusan atau seluruh kegiatan diperlukan bagi terlaksananya pengiriman, pengangkutan dan penerimaan barang dengan menggunakan multimodal trasport baik darat, laut dan udara.

Prosedur pengiriman barang berawal dari dikirimnya barang dari pengirim kepada penerima. Kemudian kedua belah pihak mengadakan kesepakatan mengenai barang yang dikirim tersebut, yang antara lain mencakup berat barang, jenis barang, tujuan barang serta layanan pengiriman barang, yang kesemuanya itu akan mendasari tarif yang dikenakan pada pengirim.

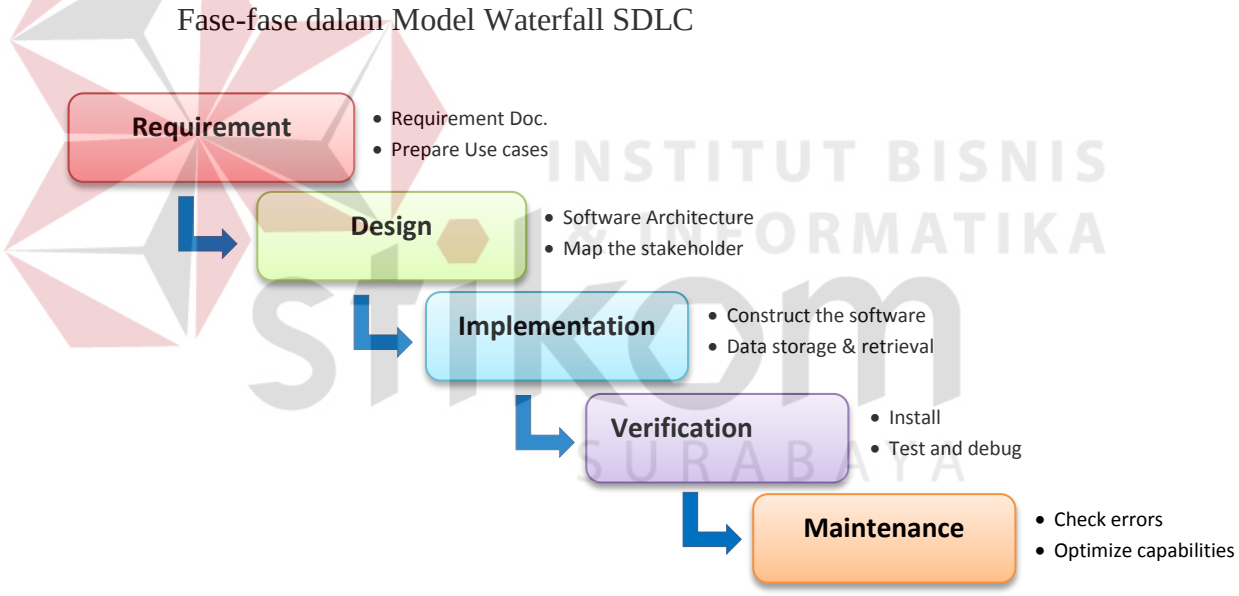
Kerjasama antara dua orang atau lebih yang didasarkan atas rasionalitas tertentu untuk mencapai tujuan yang telah ditentukan sebelumnya disebut administrasi. Pada Rush Courier, kegiatan administrasi dilakukan mulai dari pencatatan pengiriman barang dan pengaturan jadwal pengiriman. Pada proses pencatatan pengiriman barang dari pelanggan, bagian administrasi mempunyai ketentuan perhitungan biaya yang harus dibayar oleh pengirim. Jika pengirim melakukan pengiriman barang diatas tiga kilogram, maka dikenakan biaya tambahan per kilogram dua ribu rupiah seperti pada Lampiran 2. Perhitungan biaya total adalah sebagai berikut :

Total Biaya = harga + (berat – 3) * 2000

Misalnya pelanggan ingin mengirim barang dengan berat empat kilogram, perhitungannya menjadi harga (biaya kirim = 5000) ditambah dengan berat empat kilogram dikurangi tiga dan dikalikan harga kelebihan per kilogramnya total biayanya adalah tujuh ribu rupiah.

2.2 Model *Waterfall System Development Life Cycle*

Menurut Basil (2011), Model *Waterfall System Development Life Cycle* adalah proses pengembangan perangkat lunak yang berurutan (*squential*) dimana prosesnya dari atas ke bawah (seperti air terjun) melalu tahapan-tahapan yang harus dijalankan untuk keberhasilan pembuatan perangkat lunak.



Gambar 2.1 SDLC Model Waterfall

1. Fase Analis *(Analysis Phase)*. Fase analis sering disebut juga sebagai spesifikasi kebutuhan perangkat lunak (*Software Requirements Spesification* atau SRS), yaitu deskripsi lengkap dan komprehensif tentang perilaku perangkat lunak yang akan dikembangkan. Ini berimplikasi sistem dan bisnis analis untuk menetapkan persyaratan fungsional dan non fungsional.

Biasanya, persyaratan fungsional didefinisikan dengan cara menggunakan kasus yang menggambarkan interaksi pengguna dengan perangkat lunak. Mereka mencakup persyaratan seperti tujuan, ruang lingkup, perspektif, fungsi, atribut perangkat lunak, karakteristik pengguna, spesifikasi fungsi, persyaratan antarmuka *(interface)* dan persyaratan basis data *(database)*. Sebaliknya, persyaratan non-fungsional mengacu pada pelbagai kriteria, kendala, keterbatasan dan persyaratan yang dikenakan pada desain dan pengoperasian perangkat lunak bukan pada perilaku tertentu. Ini mencakup properti seperti keandalan, skalabilitas, *testability*, ketersediaan, pemeliharaan, kinerja dan standar kualitas.

2. Fase Desain *(Design Phase)*

   Fase desain adalah proses perencanaan dan pemecahan masalah *(problem solving)* untuk sebuah solusi pengkat lunak. Ini berimplikasi pengembang perangkat lunak dan desainer untuk menentukan rencana untuk solusi yang meliputi desain algoritma, desain arsitektur perangkat lunak, skema *database* konseptual dan desain diagram logis, desain konsep, desain GUI *(Graphical User Interface)* dan definisi struktur *data*.

3. Fase Implementasi *(Implementation Phase)*

   Fase implementasi mengacu pada realisasi kebutuhan bisnis dan spesifikasi desain ke dalam bentuk program nyata, *database*, *website*, atau komponen perangkat lunak melalui pemrograman dan penempatan *(deployment)*. Pada tahap ini, kode ditulis dan disusun menjadi sebuah aplikasi operasional, dan dimana *database* dan *file* teks dibuat. Dengan kata lain, fase implementasi

adalah proses mengubah seluruh persyaratan *(requirements)* dan *blueprint* ke dalam sebuah lingkungan produksi.

4. Fase Pengujian *(Testing Phase)*

   Fase pengujian juga dikenal sebagai verifikasi dan validasi, yaitu sebuah proses untuk memeriksa bahwa solusi sebuah perangkat lunak memenuhi persyaratan dan spesifikasi dan itu menyelesaikan tujuan yang telah ditetapkan. Verifikasi adalah proses evaluasi perangkat lunak untuk menentukan apakah produk dari tahap pengembangan yang diberikan memenuhi kondisi yang dikenakan pada awal fase itu, sementara validasi adalah proses pengevaluasian perangkat lunak selama atau pada akhir proses pembangunan untuk menentukan apakah itu memenuhi persyaratan yang ditentukan. Selain itu, tahap pengujian adalah outlet untuk melakukan debugging dimana *bug* dan gangguan sistem ditemukan, dikoreksi dan disempurnakan.

5. Fase Perawatan *(Maintenance Phase)*

   Fase perawatan adalah proses memodifikasi solusi perangkat lunak setelah dibuat dan diterapkan untuk memperbaiki *output*, memperbaiki *error* dan meningkatkan kinerja dan kualitas. Kegiatan pemeliharaan tambahan dapat dilakukan dalam fase ini, termasuk beradaptasi perangkat lunak untuk lingkungannya, menampung kebutuhan pengguna baru dan meningkatkan keandalan perangkat lunak.

2.3 Website

Menurut Yuhefizar (2013) *website* adalah keseluruhan halaman-halaman *web* yang terdapat dari sebuah domain yang mengandung informasi. Sebuah *website* biasanya dibangun atas banyak halaman *web* yang saling berhubungan. Hubungan antara satu halaman dengan halaman *web* lainnya disebut dengan *Hyperlink* sedangkan teks yang dijadikan media penghubung disebut *Hypertext*.

Website merupakan suatu koleksi dokumen HTML pribadi atau perusahaan yang memuat informasi dalam *Web Server* (sistem komputer di suatu organisasi, yang berfungsi sebagai *server* (suatu unit komputer yang berfungsi untuk menyimpan informasi dan untuk mengelola jaringan komputer) untuk fasilitas *World Wide Web* atau *Web*, dan dapat diakses oleh seluruh pemakai internet).

2.4 Aplikasi Web *(Web Apps)*

Aplikasi *web* adalah suatu aplikasi yang sejak awal dirancang untuk dieksekusi di dalam lingkungan berbasis *Web*. Definisi ini mengungkapkan dua aspek penting dari aplikasi, yaitu:

a. Suatu aplikasi *Web* dirancang agar dapat berjalan di dalam lingkungan berbasis *Web*. Artinya, aspek-aspek hipermedia dalam kaitannya dengan hiperteks dan multimedia di dalam kombinasi dengan logika aplikasi tradisional harus diperhitungkan di seluruh siklus hidup aplikasi, yang membuatnya berbeda dengan aplikasi konvensional.

b. Aplikasi *Web* adalah suatu aplikasi yang tidak hanya berupa sekumpulan halaman-halaman *Web*.

c. Secara khusus, aplikasi *Web* menguatkan notasi sesi yang membedakannya dari paradigma *Web* permintaan-respons *(request-response)* yang biasa. Dalam konteks ini, *Web service* secara dinamis akan menghasilkan halaman yang tidak mungkin dipertimbangkan aplikasi *Web*. Sebagai contoh, suatu layanan jadwal yang diberi tujuan dan keberangkatan, dan tempat yang diinginkan akan mengembalikan sekumpulan halaman yang berisi koneksi dan kereta yang tersedia.

2.5 Database Management System

Menurut Connolly dan Begg (2010, p66), *Database Management Systems* (DBMS) merupakan sistem perangkat lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan kontrol akses ke *database*.

DBMS menyediakan fasilitas sebagai berikut:

a. Data Definition Language (DDL)

Memungkinkan pengguna untuk membuat spesifikasi tipe data, struktur data, dan *constraint* pada data untuk dapat disimpan di dalam *database*.

b. Data Manipulation Language (DML)

Memungkinkan pengguna untuk memasukkan, meng-*update*, menghapus, dan mengambil data dari *database*.

c. Akses Kontrol

DBMS menyediakan akses kontrol ke dalam *database* seperti:

i. Sistem keamanan, mencegah pengguna yang tidak sah untuk mengakses *database* tersebut.

ii. Sistem integritas, menjaga konsistensi data yang tersimpan.

iii. Sistem kontrol konkurensi, mengijinkan akses data untuk diakses oleh *database*.

iv. Sistem kontrol pemulihan, mengembalikan *database* ke keadaan yang konsisten dari sebelumnya setelah mengalami kegagalan perangkat keras atau perangkat lunak.

v. Sebuah katalog yang dapat diakses pengguna, berisi deskripsi dari data di dalam *database*.

2.6 Unified Modeling Language

Menurut Sholiq (2010), Notasi UML dibuat sebagai kolaborasi dari Grady Booch, DR. James Rumbaugh, Ivar Jacobson, Rebecca Wirfs-Brock, Peter Yourdon, dan lainnya. Jacobson telah menulis tentang bagaimana mendapatkan persyaratan-persyaratan sistem dalam paket-paket transaksi yang disebut *Use Case*. Ia juga mengembangkan sebuah metode untuk perancangan sistem yang disebut Object Oriented Software Engineering (OOSE) yang berfokus pada analisis. Booch, Rumbaugh, dan Jacobson disebut tiga sekawan.

Penggabungan beberapa metode menjadi UML dimulai tahun 1993. Setiap orang dari tiga sekawan mulai menggabungkan idenya dengan metode-metode lain yang ada saat itu. Akhir tahun 2005 Unified Method versi 0.8 diperkenalkan. Unified method diperbaiki dan diubah menjadi UML pada tahun 1996, UML 1.0 disahkan dan diberikan pada *Object Technology Group* (OGT) pada tahun 1997. Pada tahun yang sama UML 1.1 dirilis sebagai standar industri.

2.6.1 Pemodelan Bisnis

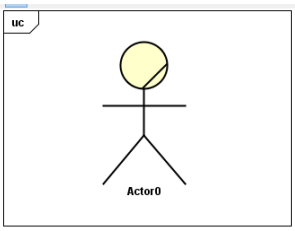
Menurut Sholiq (2010), kata pemodelan bisnis diterjemahkan dari kata Business berasal dari kata "busy+ness" yang berarti "kegiatan" dan modelling berasal dari kata "model" mendapatkan akhiran "-ing". *Modelling* berarti pemodelan, sehingga kata pemodelan bisnis secara singkat diterjemahkan menjadi "pemodelan kegiatan".

Mencermati makna harfiah pemodelan bisnis yang berarti proses memodelkan kegiatan-kegiatan, tentu saja, kegiatan pada kontek ini adalah kegiatan organisasi. Maka pemodelan bisnis adalah studi tentang organisasi dan aktivitasnya. Ketika sedang melakukan pemodelan bisnis, kita sedang menguji struktur organisasi, dan bagaimana mereka terhubungkan satu dengan lainnya.

Singkatnya, pemodelan bisnis mencoba memahami apa yang ada di dalam dan diluar bisnis, bagaimana mereka yang ada di dalam dan di luar organisasi berkomunikasi satu sama lain untuk menjalankan kegiatan-kegiatan tertentu. Informasi-informasi ini akan didokumentasikan ke suatu model bisnis.

A. Aktor Bisnis

Aktor bisnis adalah seseorang atau sesuatu yang ada di luar organisasi. Ia berinteraksi dengan organisasi dan terlibat dalam kegiatan bisnis organisasi. Contoh aktor bisnis, antara lain : pelanggan, kreditor, investor atau pemasok. Jadi posisi mereka di luar organisasi yang sedang dimodelkan, tetapi terlibat dalam kegiatan organisasi. Aktor bisnis dimodelkan dengan menggunakan ikon berikut :



Gambar 2.2 Aktor Bisnis

B. Pekerja Bisnis

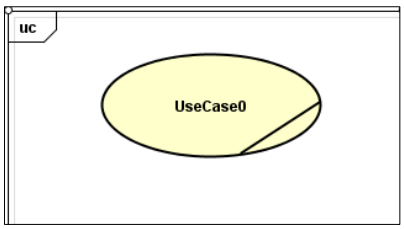
Pekerja Bisnis adalah suatu peran *(role)* di dalam organisasi, bukan posisi atau jabatan. Seseorang bisa memainkan banyak peran tetapi memegang hanya satu posisi.

Memodelkan pekerja bisnis digunakan untuk memahami peran di dalam aktivitas bisnis organisasi dan bagaimana peran tersebut berinteraksi dengan proses bisnis organisasi.

C. Use Case Bisnis

Sebuah *use case* bisnis adalah model yang digunakan untuk menggambarkan sebuah proses bisnis organisasi. Dengan kata lain, use case bisnis menginformasikan tentang aktivitas bisnis utama yang organisasi lakukan.

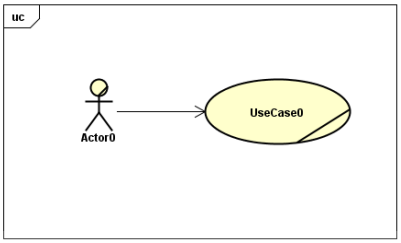
Dengan UML, digunakan simbol seperti di bawah ini untuk use case bisnis :



Gambar 2.3 Notasi Use Case Bisnis dalam UML

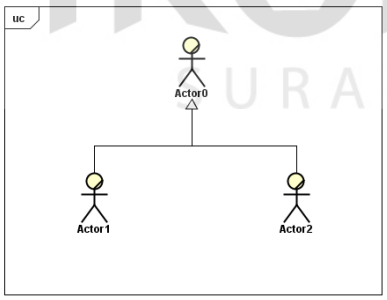
D. Relasi Asosiasi dan Generalisasi

Relasi asosiasi adalah relasi antara aktor bisnis atau pekerja bisnis dan use case bisnis. Relasi asosiasi mengindikasi bahwa aktor bisnis atau pekerja bisnis tertentu berkomunikasi terhadap fungsionalitas yang disediakan dalam *use case* bisnis. Simbol untuk relasi asosiasi adalah sebagai berikut :



Gambar 2.4 Relasi Asosiasi

Relasi generalisasi digunakan ketika ada dua atau lebih aktor bisnis, pekerja bisnis, atau *use case* bisnis yang serupa. Berikut ini simbol untuk relasi generalisasi :

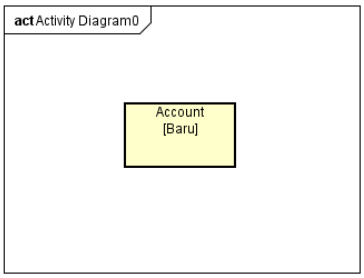


Gambar 2.5 Relasi Generalisasi

E. Entitas Bisnis

Entitas bisnis adalah obyek yang digunakan atau dihasilkan oleh organisasi saat melakukan aktivitas bisnis. Entitas bisnis meliputi sesuatu yang

pekerja bisnis hadapi sehari-hari. Misalnya daftar penjualan, daftar akun. Berikut adalah contoh entitas bisnis :



Gambar 2.6 Entitas Bisnis

F. Diagram Use Case Bisnis

Diagram *use case* bisnis menunjukkan interaksi antar aktor bisnis atau pekerja bisnis dan use case bisnis dalam sebuah organisasi. Diagram ini menggambarkan model bisnis lengkap tentang apa yang perusahaan lakukan, siapa yang ada di dalam organisasi, dan siapa yang ada diluar organisasi. Dengan diagram bisnis, dapat secara cepat memberikan informasi tingkat tinggi tentang bisnis apa yang organisasi lakukan tanpa semua rincian dari masing-masing proses bisnis membingungkan pembaca dengan menyajikan terlalu banyak notasi.

Kita dapat menggunakan use case bisnis untuk memahami dan mendokumentasikan kegiatan utama yang dilakukan oleh organisasi. Dokumentasi model bisnis juga sangat penting digunakan sebagai media analisis untuk perbaikan berkelanjutan (continously improvement).

G. Diagram Aktivitas

Diagram aktivitas menggambarkan aliran fungsionalitas sistem. Ada 2 kegunaan diagram aktivitas dalam pemodelan dengan UML :

1. Pada tahap pemodelan bisnis, diagram aktivitas dapat digunakan untuk menunjukkan alur kerja bisnis.
2. Pada tahap pemodelan sistem, diagram aktivitas dapat digunakan untuk menjelaskan aktivitas yang terjadi di dalam sebuah *use case*.

Diagram aktivitas mendefinisikan dari mana *workflow* di mulai, di mana *workflow* berakhir, aktivitas apa saja yang terjadi di dalam *workflow*, dan apa saja yang dilakukan saat sebuah aktivitas terjadi. Aktivitas adalah tugas yang dilakukan selama dalam *workflow*.

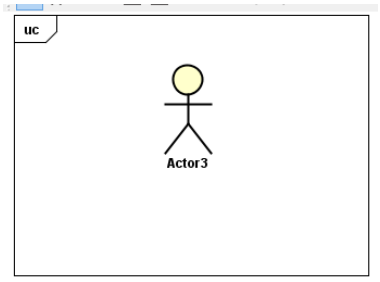
Diagram aktivitas adalah sebuah cara untuk memodelkan alur kerja *(workflow)* dari *use case* bisnis ke dalam bentuk grafik.

2.6.2 Pemodelan Use Case Sistem

Menurut Sholiq (2010), suatu pemodelan *use case* sistem berkonsentrasi pada sistem perangkat lunak yang sedang dikembangkan. Berikut adalah elemen-elemen yang terkandung dalam pemodelan *use case* sistem.

A. Aktor

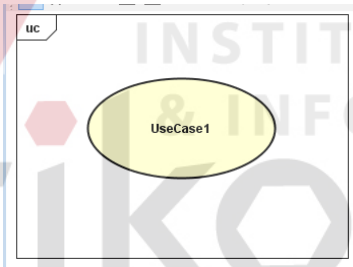
Pada pemodelan sistem, aktor memiliki arti yang berbeda dengan pemodelan bisnis, aktor bisa berupa seseorang atau apa saja yang berhubungan dengan sistem yang sedang dibangun. Berikut adalah notasi aktor :



Gambar 2.7 Notasi Aktor

B. Use Case Sistem

Use case sistem menggambarkan bagaimana seseorang sebagai pengguna berinteraksi dengan sistem. *Use case* sistem dapat dikatakan sebagai fungsi-fungsi atau fitur-fitur apa saja yang disediakan oleh sistem informasi yang akan dibangun kepada pengguna. *Use case* juga bisa meliputi fitur apa saja yang pengguna dapat lakukan terhadap sistem. Berikut ini adalah notasi *use case* sistem.



Gambar 2.8 Notasi *Use Case*

C. Flow of Event

Detail spesifik use case ditulis dalam *flow of event*. Tujuan *flow of events* adalah untuk mendokumentasikan aliran logika dalam *use case,* yang menjelaskan secara rinci apa yang pemakai akan lakukan dan apa yang sistem itu sendiri lakukan.

Sistematika *flow of event* terdiri dari :

1. Diskripsi singkat

Menjelaskan apa yang akan sistem lakukan. Diskripsi singkat harus singkat dan langsung ke fokus persoalan.

2. Prasyarat

Prasyarat adalah kondisi yang harus dipenuhi sebelum sebuah *use case* dijalankan.

3. Alur utama

Alur utama adalah jalur utama dalam *use case* yang membawa tercapainya tujuan utama sebuah use case.

4. Alur alternatif

Alur alternatif adalah penyimpangan dari alur utama dan bukan sebagai kondisi yang salah.

5. Alur salah

Alur salah adalah alur yang menyatakan penyimpangan dari alur utama atau alur alternatif yang menyatakan kondisi error dari sistem.

D. Diagram Kelas

Menurut Sholiq (2010), diagram kelas menunjukkan interaksi antar kelas-kelas dalam sistem. Kelas juga dapat dianggap sebagai cetak biru dari obyek-obyek di dalam sistem.

E. Diagram Statechart

Menurut Sholiq (2010), diagram statechart menunjukkan siklus hidup sebuah obyek tunggal, dari saat dibuat sampai obyek tersebut dihapus. Diagram ini adalah cara tepat untuk memodelkan perilaku dinamis sebuah kelas.

F. Diagram Komponen

Menurut Sholiq (2010), diagram komponen adalah diagram UML yang menampilkan komponen dalam sistem dan hubungan antar mereka. Dengan diagram komponen, seseorang yang bertanggung jawab untuk mengkompilasi dan men-deploy sistem akan tahu, kode pustaka mana saja yang dikompilasi lebih dulu sebelum yang lainnya dikompilasi. Jadi diagram komponen salah satunya berguna untuk mengetahui urutan kompilasi terhadap komponen-komponen yang akan dibuat.

G. Diagram Deployment

Menurut Sholiq (2010), diagram deployment adalah segala hal yang berkaitan dengan penyebaran fisik aplikasi. Hal ini termasuk persoalan *layout* jaringan dan lokasi komponen-komponen dalam jaringan.

Diagram deployment berisikan prosesor-prosesor, peralatan-peralatan, proses-proses, dan hubungan antar prosesor atau antar peralatan. Hanya ada satu diagram deployment dalam setiap sistem, sehingga hanya ada satu diagram deployment dalam setiap model.