

BAB II

LANDASAN TEORI

2.1 Penelitian Terkait

1. Perancangan Aplikasi Penjualan dan Pengiriman Barang Berbasis *Web* Dengan Model *Waterfall* Pada CV. Pratama Tirta Jaya oleh Sari, N. P. N. (2015)

Pada skripsi ini mengambil studi kasus pada CV. Pratama Tirta Jaya yang berlokasi di jl. Wijaya Kusuma no. 44 Pondok Betung yang bergerak dibidang penjualan bahan kertas tulis dan jasa pengiriman barang. Pada saat itu, pengelolaan informasi penjualan dan pengiriman barang yang dilakukan masih terbilang cukup sederhana, yaitu melalui media telekomunikasi atau telepon yang dinilai kurang efektif. Model yang digunakan dalam perancangan tersebut menggunakan model *Waterfall*, yaitu model yang sering digunakan dalam *Software Engineering (SE)*. Dengan adanya perancangan dan implementasi tersebut, terjadi perkembangan proses pengolahan data sehingga proses kerja menjadi lebih terkomputerisasi yang diharapkan dapat membantu meningkatkan serta mempercepat proses penjualan dan pengiriman. Selain itu, pengolahan data dan transaksi menjadi lebih optimal dengan adanya sistem yang terintegrasi, sehingga waktu yang diperlukan menjadi lebih cepat.

2. Perancangan Sistem Informasi Penjualan, Pembelian dan *Inventory* pada CV. Bambi oleh Hartono, S. (2013)

CV. Bambi sebagai perusahaan yang bergerak di bidang penjualan makanan ringan mulai menyadari bahwa penerapan sistem informasi dan teknologi informasi sangat penting dalam pengembangan bisnisnya. Oleh karena itu, CV. Bambi membutuhkan sebuah sistem yang memanfaatkan teknologi informasi yang berfungsi untuk mencatat seluruh transaksi bisnis yang berkaitan dengan sistem penjualan, pembelian dan *inventory* dalam sebuah *database*. Tujuannya agar data mudah dicari oleh unit yang berbeda dan redundansi data akan

terminimalisasi. Penelitian ini dilakukan untuk membuat sebuah desain sistem informasi dari proses bisnis penjualan, pembelian dan *inventory* yang pada akhirnya akan menghasilkan laporan yang dapat digunakan oleh pihak manajemen. Pada perancangan sistem informasi tersebut menggunakan metode *unified process* dengan pendekatan *object oriented*. Hasil dari penelitian tersebut adalah sebuah rancangan sistem informasi yang dapat mengatasi masalah dalam pencarian data pada proses penjualan, pembelian dan *inventory* karena semua data telah tersimpan dalam sebuah *database* dan diperbarui secara *real time*. Dimana setiap terjadi transaksi penjualan, stok di sistem *inventory* akan langsung berkurang. Demikian halnya setiap terjadi pembelian barang, stok pada *inventory* akan otomatis bertambah. Selain itu, sistem ini juga memudahkan manajer untuk mendapatkan laporan penjualan, pembelian dan *inventory* melalui fitur laporan.

3. Perancangan Sistem Informasi Penjualan, Produksi dan Persediaan pada PT. Triwarna Eka Multimedia oleh Surja, S. & Lius S. (2014)

Munculnya ide untuk merancang sebuah sistem informasi pada PT. Triwarna Eka Multimedia lebih dikarenakan sistem informasi yang ada sekarang masih bersifat tradisional dan sangat bergantung akan data-data yang bersifat fisik. Perancangan sistem informasi pada PT. Triwarna Eka Multimedia bertujuan untuk mengidentifikasi kebutuhan yang diperlukan oleh perusahaan dalam pengelolaan operasional proses bisnis mereka yang terkait dengan data penjualan, produksi dan persediaan barang yang berjalan. Hal ini bertujuan agar terciptanya sebuah sistem yang terintegrasi yang dapat menjawab seluruh kebutuhan perusahaan di dalam menjalankan operasional bisnisnya dan menghadapi persaingan para kompetitor. Metode yang digunakan dalam penulisan makalah tersebut adalah survei, tinjauan pustaka, serta analisis proses bisnis berjalan yang ada dengan menggunakan metode *unified modeling language*. Dengan adanya sistem informasi ini maka seluruh kegiatan operasional dalam perusahaan dipermudah dengan otomatisasi yang lebih sempurna dibandingkan dengan sistem yang sebelumnya,

disamping itu juga dapat meminimalisir kehilangan atau kesalahan data yang sering terjadi yang diakibatkan dari penyimpanan data transaksi yang dilakukan dalam bentuk fisik dan tidak terorganisir.

2.2 Tinjauan Perusahaan

2.2.1 Sejarah Perusahaan

PT. Oxyplast Indonesia didirikan pada tahun 1994 yang merupakan sebuah perusahaan manufaktur yang bergerak di bidang *powder coating* berupa cat bubuk. Produk yang dihasilkan diproduksi langsung di pabrik yang bertempat di Pasuruan, Jawa Timur. Produk yang dihasilkan cocok digunakan untuk keperluan *indoor* maupun *outdoor* dan memiliki kualitas terbaik yang cocok untuk dijadikan pilihan untuk berbagai macam industri seperti logam fabrikasi, meubel, produk-produk kendaraan berupa suku cadang maupun komponen lainnya, peralatan rumah sakit, komponen panel listrik dan lain sebagainya. Atas dasar spesialisasi PT. Oxyplast Indonesia dalam bidang *powder coating* yang juga didukung dengan kerja sama dan afiliasi dengan perusahaan mancanegara sebagai pemasok bahan baku utama, peralatan produksi, serta spesialis bidang teknik.

Seluruh staff PT. Oxyplast Indonesia berkomitmen untuk kepuasan pelanggan yang secara terus menerus selalu ditingkatkan kualitasnya, menjamin mutu dari produk yang disediakan, serta layanan kepada pelanggan yang terus ditingkatkan. Berdasarkan perjanjian lisensi eksklusif dengan perusahaan Protech-Oxyplast yang berasal dari Kanada dan Belgia, PT. Oxyplast Indonesia menjadi salah satu produsen terkemuka *powder coating* di Indonesia sejak 1994. Dengan ikatan antara teknologi dengan perusahaan afiliasi regional yang bertempat di Singapura, PT. Oxyplast mendapat kepercayaan lebih dari 300 kontraktor dan aplikator nasional.

2.2.2 Visi dan Misi Perusahaan

Visi dari PT. Oxyplast Indonesia yaitu menjadi perusahaan penyedia terbaik dalam bidang *powder coating* di dunia. Selain itu, terdapat beberapa misi dari PT. Oxyplast Indonesia, diantaranya:

1. Berkomitmen menjadi produsen *powder coating* yang berkualitas dan ramah lingkungan.
2. Bekerja secara proaktif untuk memahami kebutuhan pelanggan untuk memberikan jasa terkait.
3. Berdedikasi secara optimal untuk menjaga kualitas dan berfokus pada prinsip *zero-defects*.
4. Untuk mencapai standar tertinggi dalam sisi profesional, integritas dan menjadi industri terbaik dalam praktik dan operasional.

2.2.3 Struktur Organisasi

Penyusunan struktur organisasi pada Gambar 3.1 merupakan langkah awal dalam memulai pelaksanaan kegiatan perusahaan, dengan kata lain penyusunan organisasi merupakan langkah yang terencana dalam suatu perusahaan untuk melaksanakan fungsi perencanaan, pengorganisasian, pengarahan dan pengawasan. Berikut merupakan struktur organisasi dari PT. Oxyplast Indonesia:



Gambar 2.1 Struktur Organisasi Perusahaan

2.2.4 Uraian Tugas

1. Kepala Cabang
 - a. Mengawasi dan melakukan koordinasi dengan semua divisi dalam perusahaan.
 - b. Mengawasi dan memastikan operasional kantor cabang berjalan dengan lancar.
 - c. Memberikan evaluasi terhadap kinerja staff kantor cabang.

- d. Membaca laporan dan mengatur biaya dan strategi pemasaran.
- 2. Keuangan
 - a. Menjalankan tugas dan memberikan laporan terkait keuangan kepada kepala cabang dan keuangan pusat.
 - b. Bertanggung jawab atas proses penagihan, pembayaran, kas operasional serta pencatatan laporan keuangan dan perpajakan.
 - c. Berhubungan dengan bagian keuangan pelanggan guna mengecek pembayaran.
- 3. *Sales*
 - a. Memasarkan produk yang berkaitan.
 - b. Menciptakan peluang kerjasama dengan pelanggan baru.
 - c. Mempertahankan kerjasama dengan pelanggan.
- 4. *Collector*
 - a. Melakukan penagihan dari data pelanggan yang bersangkutan sesuai dengan jatuh tempo.
 - b. Melakukan pekerjaan yang berkaitan dengan giro atau cek yang tersedia pada pelanggan.
- 5. *Sales Admin*
 - a. Membuat laporan penjualan.
 - b. Mengatur pengiriman supir.
 - c. Berkoordinasi dengan *sales* guna mengawasi penjualan barang.
 - d. Berkoordinasi dengan *stock admin* guna mengetahui ketersediaan barang.
- 6. *Stock Admin*
 - a. Membuat laporan stok penjualan baik bulanan maupun tahunan untuk keperluan gudang pusat.
 - b. Berkoordinasi dengan bagian gudang guna mengelola ketersediaan barang.
 - c. Berkoordinasi dengan *sales admin* guna keperluan penjualan barang.
- 7. Teknikal

- a. Mengecek atas komplain pelanggan jika ada produk yang mengalami gangguan.
 - b. Bekerjasama dengan bagian penjualan jika terdapat gangguan yang dialami oleh pelanggan.
- 8. Supir
 - a. Mengirim barang yang telah disiapkan oleh bagian administrasi penjualan dan administrasi gudang.
- 9. Gudang
 - a. Menyiapkan barang yang akan dikirim oleh supir.
 - b. Mengelola proses pengambilan barang yang akan dilakukan oleh pelanggan tertentu.

2.3 Definisi Perancangan Sistem

Menurut (Laudon K & Laudon J, 2012) perancangan sistem adalah detail yang menjelaskan bagaimana sistem akan memenuhi kebutuhan informasi sebagaimana telah ditentukan oleh analisis sistem. Sedangkan menurut (Subhan M, 2012) menerangkan bahwa perancangan adalah proses pengembangan spesifikasi baru berdasarkan rekomendasi hasil sistem analisis. Sedangkan menurut (Handoko T, 2010) mengemukakan bahwa perancangan adalah pemilihan atau penetapan tujuan-tujuan dari organisasi yang menentukan strategi, kebijaksanaan, proyek, program, prosedur, metode, sistem, anggaran dan standar yang dibutuhkan untuk mencapai tujuan.

2.4 Pengertian Sistem

Sistem berasal dari bahasa Latin *sys̄stēma* dan bahasa Yunani *sustēma*. Sistem merupakan kumpulan dari elemen-elemen atau komponen-komponen yang merupakan definisi yang lebih luas lebih banyak diterima, dimana komponen-komponen tersebut tidak dapat berdiri sendiri, semuanya saling berinteraksi dan saling berhubungan membentuk suatu kesatuan sehingga sasaran sistem dapat tercapai. Teori sistem meliputi definisi sistem, lingkungan sistem dan komponen sistem.

Menurut (Marlinda L, 2011) sistem adalah kumpulan elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Sistem sangat diperlukan dalam memproses masukan untuk menghasilkan keluaran. Sebuah sistem merupakan himpunan komponen-komponen atau variabel yang terorganisasi, saling berinteraksi, saling bergantung satu sama lain dan terpadu.

Menurut (Subhan M, 2012) menerangkan bahwa sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen atau variabel yang terorganisasi, saling berinteraksi, saling tergantung satu sama lain dan terpadu. Sistem juga merupakan kumpulan elemen-elemen saling terkait dan bekerja sama untuk memproses masukan (*input*) yang ditujukan kepada sistem tersebut guna mengolah masukan tersebut sampai menghasilkan keluaran (*output*) yang diinginkan. Pendekatan suatu sistem merupakan suatu jaringan prosedur yang lebih menekankan pada urutan-urutan operasi di dalam sistem, sedangkan pendekatan yang menekankan pada elemen-elemen atau komponen merupakan interaksi antar sistem atau komponen atau mencapai sasaran atau tujuan sistem.

Menurut (Subhan M, 2012), sistem memiliki karakteristik atau sifat-sifat yang tertentu, yaitu:

1. Komponen Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi satu sama lain yang saling berkaitan dan bekerjasama antara satu sama lain untuk membentuk suatu kesatuan. Komponen-komponen yang saling berinteraksi tersebut dapat berupa suatu subsistem yang memiliki sifat-sifat sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses secara keseluruhan.

2. Proses Pengolahan Sistem

Sistem bertujuan untuk menjalankan fungsi-fungsi tertentu yang secara keseluruhan mempunyai sistem yang lain yang berfungsi untuk mengolah suatu masukan yang akan merubah suatu masukan menjadi suatu keluaran yang diharapkan.

3. Batasan (*Boundary*) Sistem

Batasan dalam sistem yaitu merupakan ruang lingkup yang dapat berupa daerah yang membatasi antara suatu sistem dengan sistem yang lainnya

baik dengan lingkungan dalam sistem maupun dengan lingkungan luar sistem itu sendiri. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan yang tidak dapat dipisahkan.

4. Lingkungan Luar Sistem (*Environment*)

Lingkungan luar dalam suatu sistem adalah apapun di luar batasan sistem yang mempengaruhi operasi sistem, lingkungan luar sistem dapat bersifat menguntungkan atau juga dapat bersifat merugikan sistem tersebut. Untuk itu lingkungan luar sistem tersebut harus selalu dijaga dan dipelihara dengan baik agar tidak menimbulkan kerugian bagi sistem tersebut.

5. Penghubung (*Interface*) Sistem

Merupakan suatu media penghubung antara suatu subsistem dengan subsistem lain maupun dengan sistem yang lainnya yang memungkinkan sumber daya yang mengalir dari suatu subsistem ke subsistem yang lainnya. Dalam hal ini, keluaran (*Output*) dari suatu subsistem dapat menjadi masukan (*Input*) bagi subsistem lainnya dengan melalui media penghubung tersebut.

6. Masukan Sistem (*Input*)

Adalah energi yang dimasukkan ke dalam suatu sistem yang dapat berupa perawatan (*Maintenance Input*), atau masukan sinyal (*Signal Input*). *Maintenance input* adalah energi yang dimasukkan agar sistem tersebut dapat beroperasi dengan baik dan seharusnya. *Signal input* adalah energi yang diproses untuk mendapatkan suatu keluaran atau hasil dari sistem yang bersangkutan.

7. Keluaran Sistem (*Output*)

Adalah hasil dari energi yang dikelola dan diklasifikasikan menjadi sebuah keluaran atau hasil yang berguna dan sisa pembuangan keluaran dapat juga merupakan masukan untuk subsistem yang lainnya kepada sistem utama.

8. Sasaran Sistem

Sasaran dari suatu sistem sangat menentukan hasil dari masukan yang dibutuhkan oleh sistem serta keluaran yang dihasilkan oleh suatu sistem.

2.5 Konsep Dasar Penjualan

Penjualan merupakan salah satu kegiatan yang dilakukan perusahaan untuk mempertahankan bisnisnya untuk berkembang dan laba atau keuntungan yang diinginkan. Menurut (Assauri, 2011) menjelaskan bahwa penjualan merupakan kegiatan pelengkap atau suplemen dari pembelian, untuk memungkinkan terjadinya transaksi. Dalam kata lain, kegiatan penjualan dan pembelian merupakan suatu kesatuan untuk dapat melaksanakan transfer antara hak atau transaksi. Oleh karena itu, kegiatan penjualan sama halnya seperti pembelian yang terdiri dari serangkaian kegiatan yang meliputi terciptanya permintaan, menemukan calon pembeli, negosiasi harga, serta syarat-syarat pembayaran yang dalam hal ini penjual harus menentukan kebijakan dan prosedur yang akan diikuti untuk memungkinkan dilaksanakannya rencana penjualan yang telah ditetapkan.

Hasil kerja dalam penjualan dapat diukur dari volume penjualan yang dihasilkan dan bukan berasal dari laba pemasaran. Perusahaan yang berorientasi pada penjualan menganut sebuah konsep yang disebut konsep penjualan. Menurut (Suyanto, 2007), konsep penjualan menerangkan bahwa konsumen akan membeli produk ke perusahaan jika perusahaan melakukan promosi dan penjualan yang menonjol. Konsep penjualan adalah orientasi manajemen yang menganggap produk-produk perusahaan didasarkan pada pertimbangan usaha-usaha nyata yang dilakukan untuk mengunggah minat akan produk tersebut.

2.6 Sistem Informasi Penjualan

Menurut (Ladjamudin, 2005), sistem informasi penjualan adalah suatu sistem informasi yang mengorganisasikan serangkaian prosedur dan metode yang dirancang untuk menghasilkan, menganalisa, menyebarkan dan memperoleh informasi guna mendukung pengambilan keputusan mengenai penjualan.

2.7 Sistem Berbasis Web

Sistem berbasis *web* adalah aplikasi atau layanan yang berada dalam *server* dan dapat diakses dengan menggunakan penjelajah *web* atau *web browser* sehingga dapat diakses dari manapun dan kapanpun melalui jaringan *internet*.

Aplikasi berbasis *web* atau sering dikenal dengan *web based application* merupakan suatu aplikasi yang berjalan di *web browser*. Aplikasi ini dapat diakses dimanapun dan kapanpun asalkan koneksi *internet* yang mendukung. Kelebihan aplikasi *web based* ini adalah tidak perlu melakukan *install* aplikasi di komputer masing-masing seperti pada aplikasi *desktop*. *Web Based* merupakan aplikasi yang dalam proses dayanya membutuhkan koneksi *internet* kemudian beranjak menuju *browser* penyedia *server*. *Web based* tidak memerlukan alokasi ruang *hard drive* lokal komputer, dan tidak peduli kapasitas komputer yang tinggi untuk menjalankan fungsinya. *Web based* tidak tergantung pada spesifikasi dan kompatibilitas dari komputer yang digunakan, tetapi kebergantungan hanya kepada akses *internet*. Selama akses *internet* tidak mengalami kendala, maka selama itu pula *web base* dapat memenuhi kebutuhan pengguna (Oktavian, 2014). Menurut (Laine et al, 2011) pengembangan aplikasi *web* menggunakan proses pengembangan secara potensial dengan arsitektur *three-tier* yang terdiri dari presentasi (*user interface*) untuk *client*, logika (*server*), dan data (manajemen data). Komponen dari aplikasi *web* disimpan pada komputer *client* atau komputer *server* karena aplikasi *web* merupakan tipe aplikasi *client* atau *server*. *Web browser* dapat digunakan untuk mengakses aplikasi *web* yang dijalankan pada komputer *client*. Salah satu contoh *web browser* adalah Google Chrome. Untuk penyimpanan aplikasi *web* terletak pada komputer *server*. Komputer ini menjalankan piranti lunak aplikasi *web* yang akan mengirimkan halaman *web* ke *web browser*. Salah satu aplikasi *web server* yang terpopuler adalah *Apache Software Foundation's Apache HTTP Server* yang biasa dikenal dengan Apache. Sebagian besar aplikasi *web* beroperasi dengan menggunakan data yang disimpan pada sebuah basis data atau *database*. Kebanyakan *server* menjalankan sebuah *Database Management System (DBMS)* dimana salah satu nya dengan menggunakan MySQL.

2.8 Model Pengembangan *Unified Process Modeling*

Unified Process adalah suatu metodologi pengembangan sistem secara *object oriented* yang dikembangkan oleh Rational Software yang merupakan bagian divisi dari IBM yang didirikan sejak tahun 2003. *Unified Process* merupakan

salah satu metodologi untuk melakukan *problem solving* dan membutuhkan adanya peran serta partisipasi dari seluruh *stakeholder* yang dalam hal ini berarti pihak pengembang sistem maupun pihak perusahaan. Menurut (Satzinger et al, 2007), secara luas, *unified process* telah diakui sebagai standar metodologi pengembangan sistem berorientasi objek. Unified process memiliki beberapa karakteristik utama, diantaranya:

1. *Use Case Driven*

Use case merupakan sebuah potongan dari keseluruhan fungsionalitas yang dilakukan oleh sistem. *Use case* bukan hanya *tools* yang berguna untuk menggambarkan kebutuhan dari sistem, namun *use case* juga mengarahkan semua kebutuhan tersebut menuju proses desain, implementasi dan pengujian.

2. *Architectur Centric*

Arsitektur merupakan gambaran mengenai keseluruhan desain yang menonjolkan beberapa karakteristik. Arsitektur dengan *use case* haruslah seimbang karena apabila direalisasikan harus disesuaikan dengan arsitektur yang dikembangkan dan juga sebaliknya, arsitektur juga harus memberikan ruang untuk realisasi dari semua *use case* yang diminta, baik dari awal pengembangan, hingga langkah selanjutnya. *Architectur centric* juga berarti suatu arsitektur sistem dipergunakan sebagai suatu artifak utama dalam konsep, membangun dan pengelolaan sistem perangkat lunak.

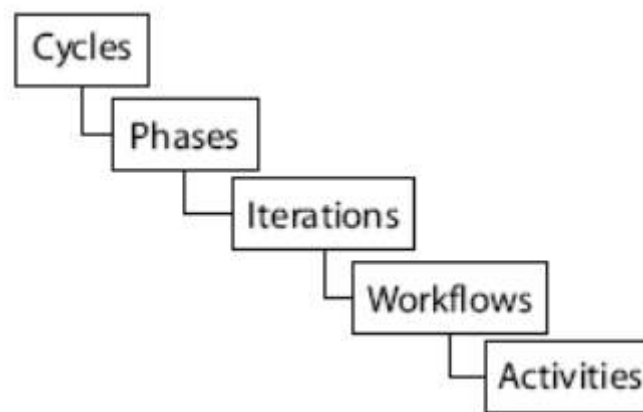
3. *Iterative & Incremental*

Iteratif mengacu pada semua tahapan pada *workflow* yang dilalui secara berulang-ulang hingga suatu *use case* tersebut berhasil terealisasikan. Dalam setiap iterasi, dilakukan identifikasi dan penetapan *use case* yang relevan yang membuat sebuah desain menggunakan arsitektur terpilih menjadi sebuah panduan, mengimplementasikan desain tersebut ke dalam komponen-komponen yang kemudian diverifikasi. Setiap perilsan produk selalu terdapat *review* yang bertujuan agar pengembangan selanjutnya dapat lebih baik. *Review* juga bertujuan untuk mengganti prosedur yang mungkin kurang sesuai.

Versi asli dari *unified process* didefinisikan dengan sangat rumit untuk setiap kegiatan. Namun, semakin hari metode ini semakin dikembangkan agar dapat semakin mudah untuk dimengerti. Untuk itu dilakukan pengembangan dari metode tersebut, salah satunya adalah *Rational Unified Process*. *Rational unified process* merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *test practice* yang terdapat dalam industri pengembangan perangkat lunak. *Unified process* terdiri dari 4 fase, diantaranya:

1. Fase Insepsi

Penekanan utama dari fase ini adalah pada *workflow requirement* dan sedikit analisis. Saat *project* beralih dari satu fase ke fase berikutnya, jumlah kerja yang dilakukan pada setiap *workflow* juga ikut berubah.



Gambar 2.2 Struktur Hirarki *Unified Process*

Selama fase ini berlangsung, sebuah ide atau asumsi dikembangkan untuk mengetahui permintaan tentang batasan dari perangkat lunak.

2. Fase Elaborasi

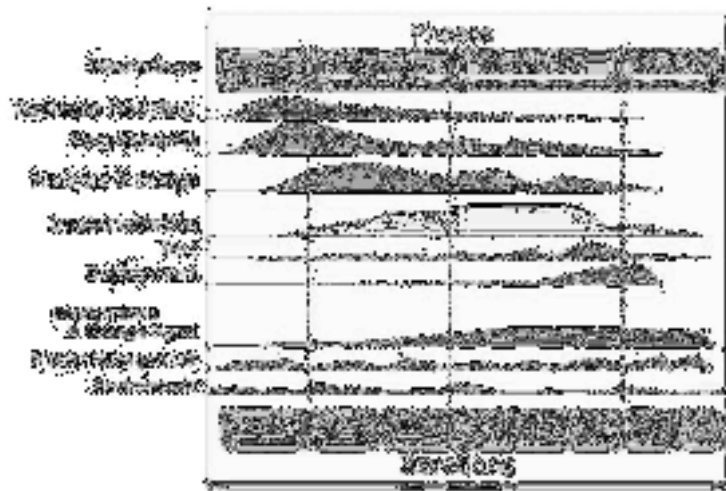
Fase elaborasi berfokus pada *workflow analysis*. Pada fase ini, pekerjaan yang dilakukan yaitu melengkapi *requirement*, baik fungsional maupun non-fungsional serta melakukan analisis dan pembuatan desain terhadap apa yang dibangun. Salah satu aktivitas utama dalam fase ini adalah membuat rancangan arsitektur sistem. Pada fase ini, juga telah dilakukan sedikit implementasi dan pengujian untuk mengeksekusi arsitektur yang telah diciptakan tersebut.

3. Fase Konstruksi

Pada fase ini, sebuah rancangan mulai direalisasikan. Tujuan dari fase ini adalah melengkapi seluruh *workflow requirement*, analisis dan desain serta membangun arsitektur yang akan ditransformasikan ke dalam sistem akhir. Pada fase ini seluruh *use case* sudah dipenuhi dan produk akan diluncurkan sesuai dengan kebutuhan.

4. Fase Transisi

Pada fase ini bertujuan untuk menempatkan sistem pada lingkungan *beta release*, dimana sejumlah *user* akan mencoba produk yang telah dihasilkan dan melaporkan kecacatan atau defisiensi yang ditemukan. Kemudian pengembang akan melakukan koreksi atas masalah yang dilaporkan untuk *general release* yang akan dihadirkan pada kumpulan *user* yang lebih luas.



Gambar 2.3 Fase dalam *Unified Process*

Proses yang dilakukan dalam metode ini tidaklah sederhana jika dibandingkan dengan metodologi klasik, seperti *waterfall* dan *iterative model*. Hal ini dikarenakan *unified process* lebih digunakan untuk membangun sebuah kerangka kerja yang dapat dikostumisasi untuk kepentingan organisasi dan proyek yang lebih spesifik. Dengan kerangka kerja tersebut, dapat tercipta beragam aplikasi dikarenakan adanya konsep *coding reuse*, dimana *coding* yang sama dapat digunakan untuk keperluan aplikasi yang sejenis.

Ada beberapa keuntungan yang didapat bila menggunakan *rational unified process* dalam perancangan dan pengembangan sebuah sistem, diantaranya:

1. Menyediakan akses yang mudah terhadap pengetahuan dasar bagi anggota tim.
2. Menyediakan petunjuk bagaimana menggunakan *Unified Modeling Language* secara efektif.
3. Mendukung proses pengulangan dalam pengembangan *software*.
4. Memungkinkan adanya penambahan pada proses yang dilakukan.
5. Memungkinkan untuk secara sistematis mengontrol perubahan yang terjadi pada sistem selama proses pengembangan berlangsung.
6. Memungkinkan untuk menjalankan *test case* dengan menggunakan *Rational Test Manager Tool*.

Selain itu, terdapat beberapa kekurangan dalam metode ini, diantaranya:

1. Metodologi ini hanya dapat digunakan pada pengembangan perangkat lunak yang berorientasi objek dengan berfokus pada *Unified Modeling Language*.
2. Membutuhkan waktu yang cukup lama dibandingkan dengan menggunakan metode lain.

2.9 Pengertian Basis Data

Basis data dapat didefinisikan dalam beberapa sudut pandang, yaitu:

1. Himpunan sekelompok data (arsip) yang saling berhubungan dan diorganisasi sedemikian rupa sehingga dapat dimanfaatkan kembali di kemudian hari dengan cepat dan mudah.
2. Sekumpulan data yang berhubungan dan disimpan secara bersama sedemikian rupa dan tanpa pengulangan dan redundansi yang tidak perlu yang bertujuan untuk memenuhi berbagai kebutuhan.
3. Kumpulan file atau arsip dalam bentuk table yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

Sedangkan menurut (Indrajani, 2011) beberapa pengertian basis data, yaitu :

1. Sebuah kumpulan data yang berhubungan secara logis dan merupakan penjelasan dari data tersebut yang dirancang dengan tujuan untuk menemukan data yang dibutuhkan oleh suatu perusahaan atau organisasi. Basis data juga dapat dikatakan sebagai kumpulan data yang saling

terintegrasi karena basis data dirancang untuk dapat digunakan oleh banyak pemakai, memegang data operasional dan juga penjelasan mengenai data tersebut, dan menghindari duplikasi data.

2. Sebuah kumpulan elemen data yang terintegrasi dan berhubungan secara logika. Basis data menggabungkan berbagai catatan yang sebelumnya disimpan dalam *file* terpisah ke dalam suatu elemen data.

Menurut (Connolly and Begg, 2010), basis data adalah sebuah kumpulan data yang saling berelasi secara logika dan dirancang untuk memenuhi informasi yang dibutuhkan oleh suatu organisasi. Dari beberapa pengertian diatas dapat disimpulkan bahwa basis data adalah sekumpulan data yang berkaitan dan saling berhubungan satu dengan yang lainnya yang disimpan didalam perangkat keras komputer yang menggunakan perangkat lunak untuk mengaksesnya. Data perlu disimpan dalam basis data untuk keperluan menyediakan informasi lebih lanjut yang dapat diolah di kemudian hari. Basis data dapat diakses dan dimanupulasi dengan menggunakan perangkat lunak paket yang disebut DBMS atau *Database Management System*.

2.10 Pengertian ERD

Model ERD atau juga dikenal sebagai *Entity Realtionship Diagram* adalah merupakan suatu model untuk menjelaskan hubungan antara data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan atau relasi. ERD didasarkan pada suatu persepsi atas keadaan nyata di dunia yang terdiri dari sekumpulan objek yang disebut *entity* dan relasi antara beberapa *entity* tersebut.

Menurut (Brady dan Loonam, 2010), *Entity Relationship Diagram* merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi yang biasanya dilakukan oleh seorang *system analist* dalam tahap analisis dari pengembangan suatu sistem. Sedangkan menurut (Sutanta, 2011), *Entity Relationship Diagram* merupakan suatu model yang dikembangkan berdasarkan objek yang digunakan untuk menjelaskan hubungan antara data dalam suatu basis data kepada pengguna secara logis.

Entity Relationship dikembangkan dalam rangka memberikan fasilitas dalam perancangan *database* dengan memberikan kesempatan untuk membuat

spesifikasi dari suatu skema yang merepresentasikan keseluruhan struktur logika dalam basis data. Komponen-komponen yang digunakan dalam *Entity Relationship Diagram* diantaranya adalah:

1. Entitas (*Entity*)

Menurut (Sutanta, 2011), entitas merupakan suatu objek yang dapat dibedakan dari yang lainnya yang dapat diwujudkan dalam basis data. Objek dasar tersebut dapat berupa orang, benda, atau hal lain yang keterangannya perlu atau telah disimpan dalam basis data. Sebuah objek yang dikategorikan sebagai entitas harus memiliki beberapa syarat, diantaranya:

- a. Merupakan objek yang memiliki lebih dari satu *entity instance* atau contoh dalam basis data
- b. Merupakan sebuah atau beberapa objek yang memiliki beberapa *attribute*.
- c. Objek yang digunakan bukan sebuah *user* dari suatu sistem juga *output* atau keluaran dari suatu sistem seperti sebuah laporan yang diberi nama dengan kata benda.



Gambar 2.4 Simbol Entitas (*Entity*)

2. *Attribute*

Attribute merupakan keterangan-keterangan yang berkaitan dengan sebuah entitas yang disimpan dalam basis data yang berfungsi untuk memperjelas suatu entitas. Sebuah *attribute* terdapat suatu himpunan nilai yang dapat diberikan pada *attribute* tersebut yang dikatakan sebagai *domain* dari suatu *attribute*.

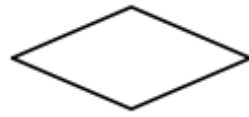


Gambar 2.5 Simbol *Attribute*

3. Relasi (*Relation*)

Relasi merupakan hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda (Sutanta, 2011). Suatu relasi adalah suatu

assosiasi diantara beberapa entitas. Sedangkan suatu himpunan relasi adalah suatu himpunan relasi yang memiliki tipe yang sama.



Gambar 2.6 Simbol Relasi (*Relation*)

4. *Link*

Sebuah garis yang digunakan untuk penghubung antara objek data, *attribute*, dan relasi.



Gambar 2.7 Simbol *Link*

5. *Mapping Cardinality*

Adalah hubungan antara sesama entitas dimana diantaranya terdapat relasi. *Mapping Cardinality* menspesifikasikan jumlah kejadian *relationship* dimana sebuah entitas dapat berpartisipasi. *Mapping Cardinality* berguna untuk mendeskripsikan relasi yang melibatkan dua atau lebih entitas. Terdapat beberapa jenis *Mapping Cardinality*, diantaranya:

a. *One to one* (1 : 1)

Dimana suatu entitas terhubung dengan maksimal satu entitas lainnya, dengan kata lain satu entitas hanya dapat memiliki satu relasi dengan entitas lain.



Gambar 2.8 *Mapping Cardinality One to One*

b. *One to many* (1 : M)

Hubungan *one to many relationship* adalah hubungan relasi yang merupakan tahap dimana hubungan antara file pertama dengan file kedua adalah satu berbanding banyak.



Gambar 2.9 *Mapping Cardinality One to Many*

c. *Many to many* (M : M)

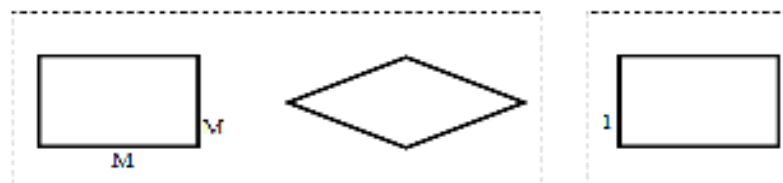
Hubungan *many to many relationship* adalah relasi antara suatu file dengan file yang lainnya dimana keduanya mempunyai relasi banyak berbanding banyak.



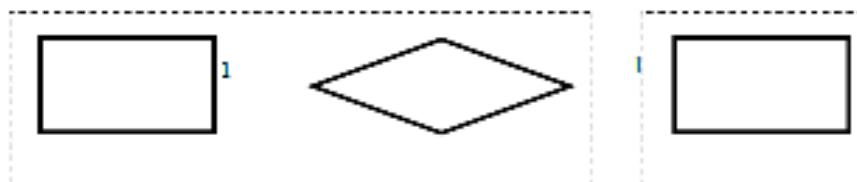
Gambar 2.10 Mapping Cardinality Many to Many

2.11 Transformasi ERD ke Bentuk LRS

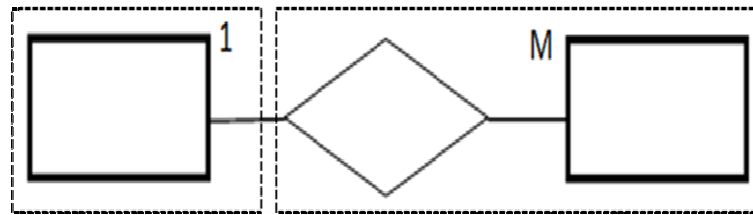
Menurut (Indrajani, 2011), transformasi yang dimaksud adalah transformasi diagram ERD menjadi diagram LRS yang merupakan suatu kegiatan untuk membentuk data-data dari diagram hubungan entitas ke suatu LRS. Diagram tersebut akan ditransformasikan kedalam bentuk LRS sehingga akan menghasilkan sebuah diagram yang sudah dapat menggambarkan basis data yang akan digunakan yang terdiri dari record dengan *field* yang dibutuhkan di dalamnya yang juga terdiri dari hubungan antara tipe *record* tersebut. Langkah pengelompokan pada diagram ER untuk menentukan entitas pada diagram LRS yaitu dimulai dari membuat konsep rancangan tabel relasi, yang kemudian dinormalisasikan untuk mendapatkan sebuah rancangan tabel relasi yang akan digunakan.



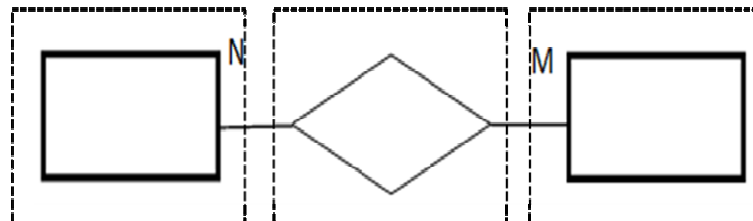
Gambar 2.11 Transformasi



Gambar 2.12 Transformasi *One to One*



Gambar 2.13 Transformasi One to Many

Gambar 2.14 Transformasi *Many to Many*

2.12 Logical Record Structure

Menurut (Indrajani, 2011), *Logical Record Structure* adalah representasi dari struktur record-record pada beberapa tabel yang terbentuk dari hasil antara himpunan entitas. *Logical Record Structure* bertujuan untuk menentukan kardinalitas, jumlah tabel dan foreign key pada database. Perbedaan LRS dan ERD adalah nama dan tipe *record* berada diluar *field* tipe *record* di tempatkan.

LRS terdiri dari beberapa *link* diantara tipe *record*. *Link* ini menunjukkan arah dari satu tipe *record* lainnya. Banyak *link* dari LRS yang diberi tanda *field* yang kelihatan pada kedua *link* tipe *record*. Penggambaran LRS mulai dengan menggunakan model yang dimengerti. Dua metode yang digunakan, dimulai dengan hubungan kedua model yang dapat dikonversikan ke LRS. Metode lain yang di mulai dengan ERD dan langsung dikonversikan ke LRS.

Tabel 2.1 Tabel *Logical Record Structure (LRS)*

No	Gambar	Nama	Keterangan
1		<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
2		<i>Association</i>	Merupakan hubungan struktural antara class yang saling berelasi.

Tabel 2.1 Tabel *Logical Record Structure (LRS)* lanjutan

No	Gambar	Nama	Keterangan
3	→	<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).

2.13 Spesifikasi Basis Data

Menurut (Sutanta, 2011) mendefinisikan data sebagai bahan keterangan tentang kejadian-kejadian nyata atau fakta-fakta yang dirumuskan dalam sekelompok lambang tertentu yang tidak acak, yang menentukan jumlah, tindakan atau hal. Data dapat berupa catatan dalam kertas, buku, atau file yang tersimpan dalam sebuah basis data. Data adalah kumpulan fakta yang masih mentah yang menjelaskan aktifitas-aktifitas yang terjadi dalam organisasi atau lingkungan fisik sebelum terorganisasi dan disusun menjadi sebuah bentuk yang dapat dimengerti dan digunakan oleh manusia (Laudon K & Laudon J, 2012). Terdapat beberapa macam tipe data, diantaranya:

1. Tipe Logika/Logis

Tipe data logika biasa disebut sebagai tipe data *Boolean* yang diambil dari nama seorang matematikawan asal Inggris, George Boole. Tipe data logika hanya mengenal dua nilai, yaitu benar (*True*) atau salah (*False*). Operasi yang dapat dilakukan pada data bertipe *Boolean* adalah operasi logika atau operasi *Boolean* dengan menggunakan operator logika.

2. Bilangan Bulat

Tipe data bilangan bulat lebih sering dikenal sebagai tipe data *Integer*. Secara teori, bilangan bulat memiliki nilai yang tidak terbatas. Dalam hal ini, pada kajian algoritma, nilai bilangan dengan tipe *integer* dibatasi sampai pada nilai tertentu.

3. Bilangan Riil

Nama tipe dari bilangan riil adalah *Real*. Bilangan riil digunakan untuk operasi bilangan-bilangan tidak bulat (pecahan atau desimal).

4. Tipe Karakter

Nama tipe untuk karakter adalah *Char*. Karakter adalah semua huruf dalam alphabet (a .. z, A .. Z), angka desimal (0 .. 9), operator aritmatika, tanda baca, dan karakter lainnya yang terdapat dalam daftar ASCII. Terdapat beberapa karakter yang dikecualikan, yaitu karakter-karakter yang telah memiliki kegunaan khusus atau telah digunakan pada sistem program.

5. Tipe *String*

String adalah rangkaian dari beberapa karakter yang membentuk suatu kata atau frase. Panjang dari string adalah tertentu tergantung dari kebutuhan. Karakter-karakter yang termuat pada data tipe *string* adalah seluruh karakter yang termuat dalam daftar ASCII, dengan pengecualian beberapa *string* yang sudah memiliki kegunaan khusus atau telah digunakan pada sistem program.

2.14 Normalisasi

Proses normalisasi merupakan proses pemilahan data elemen menjadi tabel-tabel yang menunjukkan entitas dan relasinya yang bertujuan untuk konsistensi basis data, validasi dan efisiensi manipulasi data. Menurut (Indrajani, 2011) normalisasi adalah teknik dengan pendekatan *bottom-up* yang digunakan untuk membantu mengidentifikasi hubungan yang dimulai dari menguji hubungan, yaitu secara *functional dependencies* antara *attribute*.

Pada proses normalisasi ini, ada beberapa pengujian yang dilakukan untuk membuktikan apakah tabel dan relasinya sudah merupakan *database* yang optimal atau masih dapat ditingkatkan lagi. Kesulitan tersebut sering ditemui ketika terjadi proses pemasukan (*insert*), penghapusan (*delete*), perubahan (*edit*), atau penampilan (*review*). Jika saat terjadi proses tersebut masih ditemukan kesulitan, maka dilakukan pemecahan tabel beserta relasinya sehingga mendapatkan *database* yang optimal. Namun, pada tabel yang telah dibuat tidak memerlukan normalisasi karena setiap *attribute* bukan lagi merupakan kunci yang tergantung

secara fungsional kepada *attribute* lain yang bukan merupakan kunci dalam relasi tersebut.

Tujuan utama proses normalisasi adalah mengidentifikasi kesesuaian hubungan yang mendukung data untuk memenuhi kebutuhan perusahaan. Menurut (Indrijani, 2011) proses normalisasi yang umum digunakan adalah sebagai berikut:

1. *Unnormalized Form* (UNF)

Merupakan kumpulan data yang direkam yang tidak ada keharusan untuk mengikuti format tertentu yang tersimpan dalam suatu tabel yang berisikan satu atau lebih grup yang berulang. Membuat tabel *unnormalized* yaitu dengan memindahkan data dari sumber informasi.

2. *First Norm Form* (1NF)

Merupakan sebuah relasi dimana setiap baris dan kolom berisikan satu dan hanya satu nilai. 1NF memiliki ciri setiap data dibentuk dalam file data, data dibentuk dalam suatu *record* dan nilai dari field-field berupa *automatic value*.

3. *Second Norm Form* (2NF)

Merupakan sebuah relasi dalam 1NF dan setiap *attribute non-primary key* bersifat *fully functionally dependent* pada *primary key*. Berdasarkan pada konsep *full functional dependency* dimana untuk membentuk 2NF dapat menentukan kunci fieldnya yang bersifat unik dan dapat memiliki *attribute* lain yang menjadi anggotanya.

4. *Third Norm Form* (3NF)

Berdasarkan pada konsep *transitive dependency*, dimana tidak terdapat *attribute non-primary key* yang bersifat *transitively dependent* pada *primary key*. Untuk menjadi sebuah 3NF, maka relasi harus berada dalam bentuk 2NF dan semua *attribute* bukan primer tidak mempunyai hubungan transitif.

5. *Boyce-Cood Norm Form* (BCNF)

Berdasarkan pada *functional dependencies* yang dimasukkan kedalam hitungan seluruh *candidate key* dalam suatu relasi. Bagaimanapun BCNF juga memiliki batasan-batasan tambahan yang disamakan dengan definisi

umum dari 3NF. Suatu relasi dikatakan BCNF jika dan hanya jika setiap determinan merupakan *candidate key*. Setiap relasi dalam BCNF juga merupakan 3NF, namun suatu relasi dalam 3NF belum tentu termasuk ke dalam BCNF. Dalam BCNF kesalahan jarang sekali terjadi. Kesalahan dapat terjadi pada relasi yang terdiri atas 2 atau lebih *composite candidate key* dan *candidate key overlap* yang memiliki sedikitnya satu *attribute*.

2.15 Perancangan UML

Menurut (Connolly and Begg, 2010) pendekatan *Object Oriented* adalah sebuah metode *model-driven* yang mengintegrasikan data dan proses yang dikonstruksi atau dikemas ke dalam sebuah *object*. *Object* adalah enkapsulasi sebuah data yang disebut *properties* dimana *properties* mendeskripsikan benda seperti orang, tempat, kejadian atau objek, dengan semua prosesnya yang disebut *methods*. Penjelasan mengenai Pendekatan *Object Oriented* dijabarkan dengan menggunakan diagram *Unified Modeling Language (UML)*.

UML dikembangkan oleh 3 orang yaitu Grandy Booch, Jim Rumbaugh dan Ivar Jacobson. UML menjadi bahasa yang bisa digunakan untuk berkomunikasi dalam perspektif obyek antara *user* dengan *developer*, antara sesama *developer*, antar *developer* analis dengan *design developer* dan antara *design developer* dengan *programmer developer*.

UML memungkinkan *developer* melakukan permodelan secara visual, yaitu penekanan pada penggambaran yang bukan didominasi oleh narasi. permodelan visual membantu untuk menangkap struktur dan kelakuan dari objek mempermudah penggambaran interaksi antara elemen dengan sistem, dan mempertahankan konsistensi antara desain dan implementasi dalam pemrograman.

Analisa dan perancangan berorientasi objek mempunyai beberapa konsep dasar yang berguna untuk mempermudah pemahaman yaitu:


1. *Object*, sesuatu yang dapat dilihat, disentuh atau diraba.
2. *Class*, sekumpulan objek yang sejenis yang memiliki perilaku dan *attribute* yang sejenis.
3. *Attribute*, sebuah data yang memiliki karakteristik yang dimiliki objek

4. *Behavior* (perilaku), kumpulan sesuatu yang dapat dilakukan oleh objek. Sering disebut juga sebagai *method*, *operation* dan *service*.
5. *Inheritance* (turunan), sebuah konsep di mana *method* dan *attribute* yang dimiliki oleh sebuah objek dapat diturunkan atau digunakan oleh objek lain.
6. *Encapsulation*, penggabungan dari beberapa *atribute* dan perilaku menjadi satu unit.
7. *Aggregation*, suatu hubungan dimana satu kelas yang lebih besar berisi satu atau lebih bagian kelas yang lebih kecil.
8. *Association*, suatu hubungan antara objek yang saling membutuhkan. Hubungan ini bisa satu arah ataupun lebih dari satu arah.
9. *Generalization*, suatu konsep dimana perilaku dan *atribute* yang umum dibagi ke dalam kelas mereka sendiri.



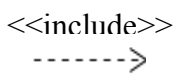
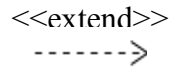

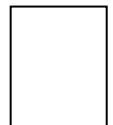
2.15.1 Use Case Diagram

Use Case Diagram menggambarkan interaksi antara sistem, *eksternal system* (sistem luar) dan *user*. Diagram ini menggambarkan secara grafikal siapa saja yang akan atau dapat menggunakan sistem dan dalam hal apa saja *user* dapat berhubungan dengan sistem yang ada. *Use case* digunakan sebagai tambahan untuk penggambaran secara tekstual terhadap urutan langkah-langkah dari setiap interaksi yang ada. Sedangkan menurut (Indrajani, 2011), *Use Case Diagram* digunakan pada saat perancangan sistem yang berfungsi untuk menjelaskan apa yang akan dilakukan oleh sistem serta aktor-aktor yang akan berhubungan dengan proses-proses yang ada pada sistem.


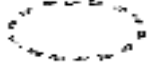

Tabel 2.2 Tabel *Use Case Diagrams*

No	Gambar	Nama	Keterangan
1		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i>

Tabel 2.2 Tabel *Use Case Diagrams* lanjutan

No	Gambar	Nama	Keterangan
2		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (<i>un-independent</i>).
3		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
4		<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i>
5		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.


Tabel 2.2 Tabel *Use Case Diagrams* lanjutan

No	Gambar	Nama	Keterangan
8		<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.





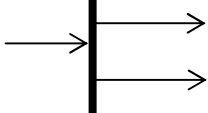
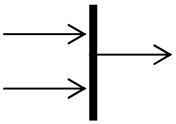
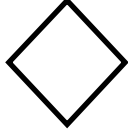
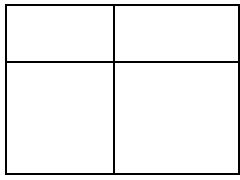
2.15.2 *Activity Diagram*

Activity Diagram digunakan dalam menggambarkan urutan aliran kegiatan secara grafis dari proses bisnis atau sebuah *use case* dan juga memodelkan kegiatan-kegiatan yang dilakukan ketika sebuah operasi dieksekusi dan hasil dari eksekusi kegiatan tersebut.

Tabel 2.3 Tabel *Activity Diagram*

No	Gambar	Nama	Keterangan
1		<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.


Tabel 2.3 Tabel *Activity Diagram* lanjutan

No	Gambar	Nama	Keterangan
2		<i>Action</i>	<i>State</i> dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali.
4		<i>Activity Final Node</i>	Bagaimana objek dibentuk dan dihancurkan.
5		<i>Fork Node</i>	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.
6		<i>Fork</i>	Mempunyai 1 transisi masuk dan 2 atau lebih transisi keluar.
7		<i>Join</i>	Mempunyai 2 atau lebih transisi masuk dan hanya 1 transisi keluar.
8		<i>Decision Point</i>	Menunjukkan dimana sebuah keputusan perlu dibuat dalam aliran kerja.
9		<i>Swimlane</i>	Sebuah cara untuk mengelompokkan <i>activity</i> berdasarkan <i>actor</i> (mengelompokkan <i>activity</i> dalam sebuah urutan yang sama).

2.15.3 Class Diagram

Class diagram menggambarkan struktur objek dari sistem. Diagram ini menggambarkan *object-object class* yang menyusun sistem dan juga relasi yang terjalin diantara *object-object class* itu.

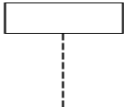



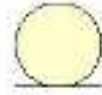
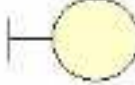

Tabel 2.4 Tabel *Class Diagram*

No	Gambar	Nama	Keterangan
1		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
2		<i>Nary Association</i>	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		<i>Class</i>	Himpunan dari objek-objek yang berbagi <i>attribute</i> serta operasi yang sama.
4		<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.

2.15.4 Sequence Diagram

Sequence diagram secara grafis menggambarkan bagaimana objek-objek berinteraksi satu sama lain lewat pesan dalam mengeksekusi sebuah *use case* atau sebuah operasi.

Tabel 2.5 Tabel *Sequence Diagram*

No	Gambar	Nama	Keterangan
1		<i>LifeLine</i>	Objek <i>entity</i> , antarmuka yang saling berinteraksi.
2		<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
3		<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
4		<i>Actor</i>	Menggambarkan orang yang sedang berinteraksi dengan sistem.
5		<i>Entity Class</i>	Menggambarkan kegiatan hubungan yang akan dilakukan.
6		<i>Boundary Class</i>	Menggambarkan sebuah penggambaran dari form.
7		<i>Control Class</i>	Menggambarkan penghubungan <i>boundary</i> dengan tabel.

2.16 PHP

2.16.1 Sejarah PHP

PHP Pertama kali ditemukan pada 1995 oleh seorang *Software Developer* bernama Rasmus Lerdorf. Ide awal PHP adalah ketika itu Rasmus ingin mengetahui jumlah pengunjung yang membaca *resume onlinenya*. *Script* yang

dikembangkan baru dapat melakukan dua pekerjaan pada awalan, yakni merekam informasi pengunjung, dan menampilkan jumlah pengunjung dari suatu *website*. Dan sampai sekarang kedua tugas tersebut masih tetap populer digunakan oleh dunia *web* saat ini. Kemudian, banyak orang yang di milis mendiskusikan *script* buatan Rasmus Lerdrof, hingga akhirnya Rasmus mulai membuat sebuah *tool* atau *script*, bernama *Personal Home Page*.

Kebutuhan PHP sebagai *tool* yang serba guna membuat Rasmus Lerdrof melanjutkan untuk mengembangkan PHP hingga menjadi suatu bahasa tersendiri yang mungkin dapat mengkonversikan data yang dimasukan melalui Form HTML menjadi suatu variabel, yang dapat dimanfaatkan oleh sistem lainnya. Untuk merealisasikannya, akhirnya Rasmus Lerdrof mencoba mengembangkan PHP menggunakan bahasa C ketimbang menggunakan Pearl. Tahun 1997, PHP versi 2.0 di rilis, dengan nama *Personal Home Page Form Interpreter (PHP-FI)*. PHP semakin populer, dan semakin diminati oleh *web programmer* dunia.

Rasmus Lerdrof benar-benar menjadikan PHP sangat populer, dan banyak sekali *Team Developer* yang ikut bergabung dengan Rasmus Lerdrof untuk mengembangkan PHP hingga menjadi sekarang, hingga akhirnya dirilis versi ke-3 pada Juni 1998, dan tercatat lebih dari 50.000 *programmer* menggunakan PHP dalam membuat *website* yang dinamis.

Pengembangan demi pengembangan terus berlanjut, ratusan fungsi ditambahkan sebagai fitur bahasa PHP, dan pada awal tahun 1999, Netcraft mencatat bahwa ditemukan 1.000.000 situs di dunia telah menggunakan PHP. Ini membuktikan bahwa PHP merupakan bahasa yang paling populer digunakan oleh dunia *web development*. Hal ini mengagetkan para pengembangnya termasuk Rasmus sendiri, dan tentunya sangat diluar dugaan sang pembuatnya. Kemudia Zeev Suraski dan Andi Gutsman selaku *core developer* atau programmer inti mencoba untuk menulis ulang PHP Parser, dan diintegrasikan dengan menggunakan *Zend Scripting Engine*, dan mengubah jalan alur operasi PHP. Dari kesemua fitur tersebut dirilis dalam PHP 4.

Pada 13 Juli 2004, PHP telah mengalami banyak sekali perbaikan di segala sisi. Netcraft mengumumkan bahwa PHP sebagai bahasa *web* terpopuler di dunia. Hal itu dikarenakan tercatat lebih dari 19.000.000 *domain* telah menggunakan PHP sebagai *Server Side Scripting*nya. PHP saat ini telah mendukung XML dan *Web Service* serta SQLite. Yang menjadikan PHP berbeda dengan HTML adalah proses yang berjalan pada PHP itu sendiri. HTML merupakan bahasa statis yang apabila kita ingin merubah konten dan isinya maka yang harus dilakukan pertama kalinya adalah membuka filenya terlebih dahulu, kemudian menambahkan isi kedalam file tersebut.

2.16.2 Karakteristik PHP

Menurut (Oktavian, 2010), pemrograman PHP memiliki karakteristik sebagai berikut:

1. Dapat dijalankan menggunakan sebuah *web server* seperti Apache.
2. Dapat dijalankan dan hanya dapat dijalankan pada *web server*.
3. Dapat digunakan untuk mengakses basis data seperti MySQL, Oracle, dan lain-lain.
4. Bersifat *open source* yang berarti dapat dikembangkan oleh siapapun tanpa adanya lisensi dan sebagainya.
5. Bebas untuk diunduh maupun diinstall tanpa ada biaya.
6. Bersifat *multiplatform* yang dapat dijalankan menggunakan berbagai macam sistem operasi seperti Linux, Windows, MacOS, maupun yang lainnya.

2.16.3 Keunggulan PHP

Dibalik segala kemudahannya dalam melakukan pengembangan maupun pemanfaatannya, PHP memiliki beberapa keunggulan yang sangat membantu para *programmer* untuk membangun suatu sistem (Oktavian, 2010), diantaranya:

1. Bahasa pemrograman PHP adalah sebuah bahasa pemrograman yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. *Web Server* yang mendukung PHP dapat ditemukan dimana-mana, mulai dari IIS sampai dengan Apache dengan konfigurasi yang sangat mudah.

3. Lebih mudah dikembangkan karena banyaknya milis-milis maupun forum yang berisi banyak *developer* yang siap membantu dalam proses pengembangannya.
4. PHP adalah bahasa pemrograman yang paling mudah dipelajari dikarenakan banyaknya referensi yang tersedia di *internet*.
5. Bersifat *open source* yang dapat digunakan di berbagai sistem operasi seperti Linux, Windows, dan Mac OS serta dapat dijalankan secara *runtime* melalui *console* serta dapat juga menjalankan perintah-perintah sistem.

2.16.4 Kelemahan PHP

Sedangkan terdapat beberapa kelemahan PHP dari bahasa pemrograman lain, diantaranya:

1. Dalam pengembangan berskala besar, PHP dinilai kurang mendetail.
2. Memiliki pemrograman berorientasi objek yang sesungguhnya.
3. Tidak dapat memisahkan antara tampilan antar muka dengan logika secara lebih baik.

2.17 Definisi Bootstrap

Bootstrap adalah sebuah *framework* atau kerangka kerja yang terdiri dari CSS dan *javascript* yang dapat digunakan untuk membuat suatu *website* yang responsif dengan cepat, mudah dan gratis. Menurut (Prianggoro, 2015), dalam Bootstrap terdapat struktur *grid*, yaitu struktur dua dimensi yang terdiri dari sumbu horizontal dan vertikal sehingga dengan menggunakan Bootstrap akan membentuk kolom dan baris seperti adanya kolom dan baris pada media cetak. Struktur tersebut terdiri dengan lebar baris 940px dan memiliki 12 kolom pada layar *desktop*. Struktur *grid* yang diterapkan tersebut menjadikan Bootstrap memiliki beberapa kelebihan, diantaranya:

1. Waktu pembuatan *website* yang lebih cepat dikarenakan elemen yang digunakan untuk membuat sebuah *website* telah disediakan oleh Bootstrap sehingga pengguna hanya tinggal memanggil *class* yang telah ditetapkan oleh Bootstrap tersebut.

2. Rancangan yang digunakan oleh Bootstrap lebih rapi dikarenakan Bootstrap telah membuat struktur dengan baik sehingga pengguna dapat menggunakan rancangan yang dibuat dengan seperlunya.
3. Rancangan yang dibuat oleh Bootstrap lebih ringan untuk dijalankan pada *website*. Hal ini menjadi titik berat dari pengguna yang menginginkan *website* yang dibuat memiliki desain antar muka yang ramah, responsif namun tetap ringan untuk diakses.
4. Dokumentasi yang disediakan oleh Bootstrap sangat lengkap sehingga pengembang tidak perlu repot untuk mempelajari. Hal ini menjadikan Bootstrap lebih dipilih oleh pengguna ketimbang kerangka kerja lainnya.

2.18 Definisi JQuery

Menurut (Hakim, 2009), JQuery adalah suatu *framework* atau kerangka kerja dari *javascript* yang menekankan bagaimana interaksi antara *javascript* tersebut dengan HTML. JQuery memudahkan para pengguna dalam pembuatan kode *javascript* yang jika pada umumnya memerlukan baris kode yang panjang dan bahkan terkadang akan sulit untuk dipahami. Disinilah peranan JQuery sebagai kerangka kerja berguna bagi pengguna yang dapat langsung menggunakan fungsi yang telah disediakan.

Selain itu pengguna dapat menggunakan fungsi yang dapat mengambil informasi dari *server* tanpa melakukan pembaruan pada halaman tersebut. dalam artian lain halaman tersebut akan terlihat berganti secara otomatis. Hal itu dapat memungkinkan dikarenakan JQuery telah mengaplikasikan fungsi yang digunakan untuk memanggil AJAX (*Asynchronous Javascript and XML*). Hal tersebut tentunya akan memudahkan kerja dari pengguna dimana pengguna tidak perlu menuliskan kode dengan jumlah yang banyak untuk menjalankan fungsi tersebut.

2.19 Pengertian MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data yang dibuat dan dikembangkan oleh MySQL AB yang memiliki lisensi GNU *General Public License (GPL)* dan berdasarkan pada SQL yang dikenal sebagai *Database*

Management System (RDBMS) yang memiliki *multithread* dan *multi-user* dengan instalasi sekitar 6 juta di seluruh dunia. MySQL AB adalah perusahaan komersial asal Swedia yang didirikan oleh David Axmark, Allan Larsson dan Michael “Monty” Widenius. MySQL AB memegang penuh hak cipta hampir atas semua kode sumbernya.

Menurut (Marlinda L, 2011), MySQL merupakan software *database* yang paling populer dikarenakan performa *query* dari *database* yang bisa dikatakan paling cepat dan jarang bermasalah. Hal tersebut yang menjadikan MySQL sebagai perangkat lunak yang bersifat *open source* yang dapat digunakan untuk kepentingan komersil maupun personal. MySQL memiliki beberapa kelebihan dibandingkan dengan sistem *database management* lainnya, diantaranya:

1. Merupakan *database* yang bersifat *open source* yang berarti bebas digunakan untuk keperluan pribadi maupun komersil tanpa harus membayar lisensinya.
2. Mampu menerima *query* dalam jumlah yang bertumpuk dalam satu permintaan *multi threading*.
3. Dapat menyimpan data berkapasitas sangat besar hingga berukuran *GygaBite*.
4. Dapat diakses dan dipadukan dengan aplikasi apapun.
5. Menggunakan enkripsi yang mejadikan MySQL sebagai *database* yang aman.
6. Dapat digunakan secara *multi user*.
7. Mendukung *field* yang dapat dijadikan sebagai *primary key* dan *unique key*.
8. Memiliki kecepatan yang cepat dalam proses pembuatan serta pengisian tabel.

Menurut (Marlinda L, 2011) MySQL juga memiliki beberapa kelemahan, diantaranya:

1. Jarang digunakan dalam pemrograman visual sebabkan kurang medukung untuk koneksi dengan bahasa pemrograman visual, seperti Visual Basic, Delphi dan FoxPro karena *field* yang dibaca harus sesuai dengan koneksi dari program visual tersebut.

2. Data yang dapat ditangani belum terlalu besar dan rumit.
3. *Feature-creep* yang berarti MySQL berusaha kompatibel dengan beberapa standar serta berusaha memenuhinya, namun dengan fitur-fitur belum lengkap dan berperilaku sesuai standar.