BAB II

LANDASAN TEORI

2.1 Sistem Informasi Kepegawaian

Sistem informasi kepegawaian adalah prosedur sistematik untuk mengumpulkan, menyimpan, mempertahankan, menarik dan memvalidasi data yang dibutuhkan oleh sebuah organisasi tentang sumber daya manusia, aktifitas–aktifitas personalia, karekteristik – karekteristik, dan unit - unit organisasi[1].

Di indonesia Sistem Informasi Kepegawaian merupakan suatu totalitas terpadu yang tediri dari perangkat pengolah meliputi pengumpul prosedur, tenaga pengolah dan perangkat lunak, perangkat penyimpanan meliputi pusat data dan bank data serta perangkat komunikasi yang saling berkaitan, saling ketergantungan dan saling menentukan dalam rangka penyediaan informasi di bidang kepegawaian[2].

Kegiatan yang berhubungan dengan sumber daya manusia yaitu *Human Resources information system* (HRIS) sebagai pendukung manajemen sumber daya manusia. HRIS merupakan sebuah sistem yang digunakan untuk memperoleh *(acquire),* menyimpan *(store),* memanipulasi *(manipulate),* menganalisis *(analyze),* mendapatkan kembali *(retrieve),* dan mendistribusikan *(distribute)* informasi yang berhubungan dengan sumber daya manusia untuk kepentingan organisasi.

Human Resource Information System (HRIS) dalam bahasa indonesia adalah sistem informasi sumber daya manusia atau lebih dikenal dengan istilah sistem informasi kepegawaian. Sistem informasi kepegawaian bertugas merancang format – format data kepegawaian dan mengatur sistem pengumpulan, pengolahan, penyimpanan, dan pelaporan informasi kepegawaian yang terdiri dari: data pegawai, data jabatan, data pendidikan, data keluarga dan lain-lain sehingga dapat dikelola informasi tentang kinerja pegawai, perencanaan kebutuhan pegawai, pembinaan dan pengembangan karirnya, kesejahteraaan, serta pemberhentian atau pensiun.

Sistem informasi kepegawaian dapat juga didefinisikan dengan segala sesuatu yang menyangkut perencanaan, pengembangan, pengelolaan dan penggunaan alat bantu teknologi infromasi untuk membantu manusia dalam menyelesaikan seluruh pekerjaan yang berhubungan dengan pengolahan dan pengelolaan informasi.

2.1.1 Tujuan Sistem Informasi Kepegawaian

Adapun tujuan sistem informasi kepegawaian diruang lingkup pemerintahan di antaranya :

a. Untuk mendukung sistem manejemen PNS yang rasional dan pengembangan SDM di aparatur Pemerintah.
b. Mewujudkan data kepegawaian yang mutakhir dan terintegeritas.
c. Menyediakan informasi yang akurat untuk keperluan perencanaan, pengembangan, kesejahteraan dan pengendalian pegawai.
d. Membantu kelancaran pekerjaan dibidang kepegawaian, terutama dalam pembuatan laporan.

2.1.2 Manfaat Sistem Informasi Kepegawaian

a. Pelacakan infromasi data seseorang pegawai akan mudah dan cepat.
b. Pembuatan laporan dapat mudah dikerjakan.
c. Mengetahui pegawai yang akan naik pangkat dan yang akan mendapat kenaikan gaji berkala.
d. Memudahkan suatu pekerjaan yang berhubungan dengan kepgawaian.
e. Mendapatkan informasi tentang keadaan pegawai (Profil Kepegawaian) yang cepat dan akurat.
f. Dapat merencanakan penyebaran (mutasi) pegawai sesuai pendidikan dan kompetensinya.
g. Merencanakan kebutuhan pegawai (Neraca Kebutuhan Pegawai)

2.1.3 Keuntungan Sistem Informasi Kepegawaian *Berbasis Online*

Adapun keuntungan Sistem Informasi Kepegawaian berbasis *online* (Berbasis *internet*) di antaranya :

a. Dapat memelihara satu data besar secara bersama-sama.
b. Kesalahan/ data yang kurang *valid* dapat dimonitor dan dikoreksi bersama.
c. Dapat melakukan pertukaran data dan *file*.
d. Berbagi sumber daya misalnya pemakaian satu *printer* untuk beberapa *komputer* yang terhubung dalam jaringan *komputer*.
e. Mempermudah komunikasi dalam suatu lingkungan kerja, misalnya dengan adanya program *E-mail* atau *Chatting* .
f. Apabila salah satu *unit komputer* terhubung ke *internet* melalui *modem* atau LAN, maka semua atau sebagian unit *komputer* dalam jaringan dapat mengakses dengan metode *sharing connection*.

2.2 *Website*

2.2.1 Sejarah *Website*

World Wide Web atau *WWW* atau juga dikenal dengan *WEB* adalah salah satu layanan yang didapat oleh pemakai computer yang terhubung ke *internet*. *Website* pertama kali ditemukan oleh Sir Timoyhy John. *Web*site sendiri tersambung dalam jaringan yaitu pada tahun 1991. Tujuan Sir Timoyhy John merancang *website* ini adalah untuk mempermudah pekerjaannya serta saling bertukar informasi bagi peneliti ditempatnya bekerja. Akhirnya pada tanggal 30 April 1993, *CERN*, perusahaan tempat Sir Timoyhy John bekerja mengumumkan jika *website* sudah bisa digunakan oleh publik dengan gratis.

Sebuah *website* bisa merupakan hasil kerja dari perseorangan, atau menunjukkan kepemilikan dari suatu organisasi atau perusahaan. Biasanya *website* menunjukkan beberapa topik khusus atau suatu kepentingan tertentu. *Website* ini menyediakan informasi bagi pemakai computer yang terhubung ke *internet* dari sekedar informasi "sampah" atau informasi yang tidak berguna sama sekali sampai informasi yang serius, dari informasi yang gratisan sampai informasi yang komersial. *Website* atau *situs* dapat diartikan sebagai kumpulan

halaman - halaman yang digunakan untuk menampilkan informasi *teks*, gambar diam atau gerak, *animasi*, suara, dan gabungan dari semuanya itu baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing - masing dihubungkan dengan jaringan - jaringan halaman (*hyperlink*).

Sebuah *website* dapat berisikan *hyperlink* yang dapat menghubungkan ke *website* atau *page* lain. *Website* ditulis atau secara dinamik dikonversi menjadi HTML dan diakses melalui *software* yang sering disebut dengan *web browser*, dan dikenal juga dengan HTTP *Client.*

2.2.2 Definisi *Website*

Website atau situs merupakan kumpulan halaman yang menampilkan informasi data, *teks*, gambar, data animasi, suara, dan gabungan dari semuanya, baik yang bersifat dinamis yang membentuk suatu rangkaian bangunan yang saling terkait dengan jaringan – jaringan halaman.

Website merupakan fasilitas internet yang menghubungkan dokumen dalam lingkup lokal maupun jarak jauh. Dokumen pada website disebut dengan web page dan link dalam website memungkinkan pengguna bisa berpindah dari satu page ke page lain (hyper text), baik diantara page yang disimpan dalam server yang sama maupun server diseluruh dunia[3].

2.2.3 Jenis –Jenis Situs *Website*

Secara umum *situs web* digolongkanmenjadi 3 jenis yaitu :

a. *Website* Statis

Website Statis adalah *web* yang mempunyai halaman tidak berubah. Artinya adalah untuk melakukan perubahan pada suatu halaman dilakukan secara manual dengan mengedit kode yang menjadi struktur dari situs itu.

b. *Website* Dinamis

Website Dinamis merupakan *web* yang secara struktur diperuntukan untuk *update* sesering mungkin. Biasanya selain menu utama yang bisa diakses

oleh user pada umumnya, juga disediakan halaman *backend* untuk mengedit kontent dari *website*. Contoh umum mengenai *website* dinamis adalah *web* berita atau *web* portal yang didalamnya terdapat fasilitas berita, *polling* dan sebagainya.

c. *Website* Interaktif

Website Interaktif adalah *web* yang saat ini memang sedang *booming*. Salah satu contoh *website* interaktif adalah *blog* dan *forum*. Di *website* ini *user* bisa berinteraksi dan beradu argumen mengenai apa yang menjadi pemikiran mereka. Biasanya *website* seperti memiliki *moderator* untuk mengatur supaya topik yang diperbincangkan tidak melenceng dari alur pembicaraan.

2.2.4 Unsur – Unsur Penunjang dalam *Website*

Menurut Rian Eko Wiliyanto (2012) untuk membangun sebuah *website,* diharuskan untuk menyediakan unsur penunjang lainnya di antaranya:

a. Nama*Domain*

Domain Name atau URL adalah alamat unik di dunia *internet* yang digunakan untuk mengenali sebuah *situs*, atau dengan kata lain nama *domain* adalah alamat yang digunakan untuk menemukan sebuah *website* di *internet*.

Nama *domain* diperjualbelikan secara bebas di *internet* dengan status sewa tahunan. Nama *domain* sendiri mempunyai identifikasi ekstensi/akhiran sesuai dengan kepentingan dan lokasi keberadaan *website* tersebut. Contoh nama domain ber-*ekstensi* internasional adalah *com, net, org, info, biz, name, ws*. Contoh nama *domain* ber-*ekstensi* lokasi Negara Indonesia adalah *co.id* (untuk nama *domain website* perusahaan), *ac.id* (nama *domain website* pendidikan), *go.id* (nama domain *website instansi* pemerintah), *or.id* (nama *domain website* organisasi).

b. *Web Hosting*

Web hosting adalah ruangan yang terdapat dalam *harddisk* tempat menyimpan berbagai data, *file - file*, gambar dan lainnya yang akan ditampilkan di *website*. Besarnya data yang bisa dimasukkan tergantung dari besarnya *web hosting* yang disewa/ dipunyai, semakin besar *webhosting* semakin besar pula data yang dapat dimasukkan dan ditampilkan dalam *website*. *Web Hosting* juga diperoleh dengan menyewa. Besarnya *hosting* ditentukan ruangan *harddisk* dengan ukuran MB (*Mega Byte*) atau GB (*Giga Byte*). Lama penyewaan *web hosting* rata-rata dihitung per tahun. Penyewaan *hosting* dilakukan dari perusahaan - perusahaan penyewa *web hosting* yang banyak dijumpai baik di Indonesia maupun Luar Negri.

c. Bahasa Pemrograman

Bahasa pemrograman adalah bahasa yang digunakan untuk menerjemahkan setiap perintah dalam *website* yang pada saat diakses.Jenis bahasa pemrograman sangat menentukan statis, dinamis atau interaktifnya sebuah *website*. Semakin banyak ragam bahasa pemrograman yang digunakan maka akan terlihat bahwa *website*tersebut semakin dinamis dan interaktif serta terlihat bagus. Beragam bahasa pemrograman saat ini telah hadir untuk mendukung kualitas *website*. Jenis jenis bahasa program yang banyak dipakai para *desainer website* antara lain *HTML, ASP, PHP, JSP, Java Scripts, Java applets,* dsb.Bahasa dasar yang dipakai setiap *situs* adalah *HTML* sedangkan *PHP, ASP, JSP* dan lainnya merupakan bahasa pendukung yang bertindak sebagai pengatur dinamis, dan interaktifnya *situs*. Bahasa program *ASP, PHP, JSP* atau lainnya bisa dibuat sendiri. Bahasa program ini biasanya digunakan untuk membangun portal berita, *artikel, forum* diskusi, buku tamu, anggota organisasi, *email, mailing list* dan lain sebagainya yang memerlukan *update* setiap saat.

d. *Desain Website*

Unsur *website* yang penting dan utama adalah *desain. Desain website* menentukan kualitas dan keindahan sebuah *website*. Desain sangat berpengaruh kepada penilaian pengunjung akan bagus tidaknya sebuah *website*. Serta mempengaruhi dengan kenyamanan *konsumen* atau pembaca.Untuk membuat *website* biasanya dapat dilakukan sendiri atau menyewa jasa *website designer*. Saat ini sangat banyak jasa *web designer*, terutama di kota-kota besar. Perlu diketahui bahwa kualitas *situs* sangat ditentukan oleh kualitas *designer*. Semakin banyak penguasaan *web designer* tentang beragam *program/ software* pendukung pembuatan *situs* maka akan dihasilkan *situs* yang semakin berkualitas, demikian pula sebaliknya. Jasa *web designer* ini yang umumnya memerlukan biaya yang tertinggi dari seluruh biaya pembangunan *situs* dan semuanya itu tergantung kualitas *designer*.

e. Publikasi *Website*

Keberadaan seakan kekurangan makna dan nilai kegunaannya jika dibangun tanpa dikunjungi atau dikenal oleh masyarakat atau pengunjung *internet*. Karena *efektif* tidaknya *situs* sangat tergantung dari besarnya pengunjung dan komentar yang masuk. Untuk mengenalkan *situs* kepada masyarakat memerlukan apa yang disebut *publikasi* atau *promosi* tadi.

2.2.5 Fungsi *Website*

Secara umum *situs web* mempunyai fungsi sebagai berikut[4] :

a. Fungsi Komunikasi

Situs web yang mempunyai fungsi komunikasi pada umumnya adalah *situs web* dinamis. Karena dibuat menggunakan pemrograman *web (server side)*maka dilengkapi fasilitas yang memberikan fungsi – fungsi komunikasi, seperti *web mail, form contact, chatting form,* dan yang lainnya.

b. Fungsi Informasi

Situs web yang memiliki fungsi informasi pada umumnya lebih menekankan pada kualitas bagian *konten*-nya, karena tujuan *situs* tersebut adalah menyampaikan isinya. *Situs* ini sebaiknya berisi *teks* dan *grafik* yang dapat di *download* dengan cepat. Pembatasan penggunaan *animasi* gambar dan *elemen* bergerak seperti *shockwave* dan *java* diyakini sebagai langkah yang tepat, diganti dengan fasilitas yang memberikan fungsi informasi seperti *news, profile company, library, reference* dan lain – lain.

c. Fungsi *Entertainment*

Situs web juga dapat memiliki fungsi *entertainment*/ hiburan. Bila *situs web* yang berfungsi sebagai sarana hiburan maka penggunaan *animasi* gambar dan *elemen* bergerak dapat meningkatkan mutu *presentasi desain*-nya, meski tetap harus mempertimbangkan kecepatan *download*-nya. Beberapa fasilitas yang memberikan fungsi hiburan adalah *gameonline, film online, music online*, dan sebagainya.

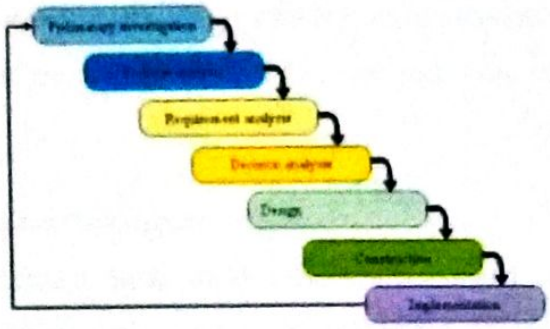
d. Fungsi Transaksi

Situs web dapat dijadikan sarana transaksi bisnis, baik barang, jasa, atau lainnya. *Situs web* ini menghubungkan perusahaan, konsumen, dan komunitas tertentu melalui transaksi elektronik. Pembayarannya bisa menggunakan kartu kredit, *transfer*, atau dengan membayar secara langsung.

2.3 Metode Fast (*Framework for the Applications of System Technology*)

FAST atau *Framework for the Applications of System Technology* mendefinisikan tahapan untuk mengidentifikasi dan mengevaluasi permasalahan-permasalahan, kesempatan - kesempatan, hambatan - hambatan yang terjadi, dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan - perbaikan. Tahapan pada *FAST* berdasarkan pada permasalahan dan kesempatan yang

dihadapi dengan peningkatan-peningkatan yang diharapkan dari sistem yang dikembangkan[5].



Gambar 2.1(Whitten)
Fase - Fase Metode *FAST*.

FAST sendiri berkaitan erat dengan *analisis* dan *desain sistem* melalui cara *PIECES*(*Performance, Information, Economics, Control, Efficiency,* dan *Service)*. *PIECES* membantu *metode FAST* pada tahap analisis masalah dan kebutuhan *sistem*, meliputi:

a. *Performance* (kinerja), peningkatan terhadap kinerja *sistem* yang baru sehingga menjadi lebih efektif diukur dari jumlah pekerjaan yang dapat dilakukan pada saat tertentu (*throughput*) dan *response time.*
b. *Information* (informasi), peningkatan terhadap kualitas informasi yang disajikan.
c. *Economics* (ekonomi), peningkatan terhadap manfaat-manfaat atau keuntungan atau penurunan biaya yang terjadi.
d. *Control* (pengendalian), peningkatan terhadap pengendalian untuk mendeteksi dan memperbaiki kesalahan serta kecurangan yang akan terjadi.
e. *Efficiency* (efisiensi), peningkatan terhadap efisiensi operasi.
f. *Service* (pelayanan), peningkatan terhadap pelayanan yang diberikan oleh sistem.

Pengembangan sistem dengan *metode FAST* dilakukan secara berurutan yakni melalui tahapan *investigasi* atau *survei* awal, analisis masalah, analisis kebutuhan, analisis keputusan, pembuatan rancangan, mengkonstruksi. menerapkan *system*, mengoperasikan dan pemeliharaan sistem. Pengembangan ini

bersifat daur hidup karena setelah selesai tahapan implementasi dan pemeliharaan maka sistem tersebut akan memberikan umpan balik ke analisis *sistem* yang telah dirancang. Sehingga tahapan pengembangan diatas terus menerus dilakukan demi penyempurnaan *sistem*. Berikut akan dijelaskan *fase-fase* yang digunakan dalam *metode FAST*.

a. *Phase* 1 *:Preliminary Investigation Phase*

Tahap ini merupakan tahap awal dari pengembangan *sistem*. Fase ini berisikan *investigasi* awal ketika ingin merancang sebuah *sistem*, seperti wawancara, tinjauan langsung dan mempelajari *dokumen* perusahaan.Hasil *investigasi* harus mampu menjawab berbagai pertanyaan:

a) Apakah masalah atau peluang yang ada.
b) Berapa besar upaya pengembangan sistem yang kemungkinan akandilakukan.
c) Alternatif- altematif apa untuk memecahkan masalah yang ditemukan.
d) Beberapa besar biaya dan manfaat masing-masing pemecahan masalah tersebut.

Lingkup masalah yang ditetapkan dari tahap ini menyatakan seberapa besar proyek ini akan dilaksanakan. Dengan adanya lingkup seperti ini maka analis dapat menentukan tim proyek, *estimasi* biaya, dan menyiapkan jadwal untuk tahap-tahap selanjutnya. Kemudian akan ditentukan oleh pemilik *sistem* apakah ia menyetujui lingkup seperti ini dengan biaya dan jadwal yang telah dirancang atau lingkup yang ada perlu diperkecil lagi. *Output* dari tahap ini adalah *project charter*.

b. *Phase* 2 : *Problem Analysis Phase*

Problem Analysis ialah menganalisa masalah-masalah yang terdapat di lapangan. Tahap ini merupakan pengembangan dari tahap pertama. Pada tahap ini dilakukan analisis terhadap sistem yang telah ada saat itu. Analisis masalah dapat didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya, dengan maksud untuk

mengidentifikasikan dan mengevaluasi permasalahan, kesempatan, dan hambatan-hambatan yang terjadi serta kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan. Dalam tahapan ini dilakukan penelitian terhadap komponen - komponen sistem untuk memahami sistem yang ada, sebagai dasar untuk rancangan sistem yang diharapkan dengan cara melakukan wawancara pada pelaku - pelaku sistem meliputi:

a) Apa yang dilakukan dalam menjalankan fungsi dan peran dalam sistem informasi?
b) Bagaimana proses informasinya dari sejak penangkapan atau pencatatan data, proses pengolahannya dan informasi yang dihasilkan serta digunakan untuk apa informasi tersebut?
c) Dimana proses informasi pada sistem informasi terjadi dan digunakan oleh bagian-bagian atau fungsi-fungsi dimana saja?
d) Mengapa proses informasi tersebut terjadi dan mengapa keputusan tersebutditetapkan berdasarkan informasi yang dihasilkan?

c. *Phase* 3 : *Requirement Analysis Phase*

Pada tahap analisis kebutuhan dilakukan pengumpulan dan analisis data,terutama menyangkut kebutuhan para pengguna sistem, dan menilai kekuatan maupun kelemahan *metode* kerja yang telah diterapkan selama ini. Dalam *FAST*, ada 4 sumber informasi yang digunakan untuk analisis kebutuhan :

a) *Dokumen* : meliputi pedoman kerja (*protap*), *form* pemasukan data dan pelaporan,diagram kerja, bagian organisasi untuk melihat hirarki wewenang dan tanggung jawab, dan lain-lain.
b) *Kuesioner* : keuntungan *kuesioner* adalah besarnya data yang dapat dikumpulkan dengan cepat, meliputi wilayah yang luas, relatif tidak mahal dan identitas *responden* dapat disembunyikan (*anonymous*).
c) Wawancara dirancang untuk menangkap beberapa data sejenis yang diperoleh dari *kuesioner*, namun informasi yang dikumpulkan lebih mendalam.

d) *Observasi* dilakukan untuk mengidentifikasi proses kerja dalam sistem lama yang kurang efektif dan tidak memuaskan bagi konsumen. Setelah data terkumpul, dilakukan analisis untuk mencapai kesimpulan – kesimpulan yang melandasi perancangan sistem. Diagram mengenai aliran data (*data flow diagram*), aliran kerja *system* (*system flowchart*), analisis terstruktur dan teknik perancangan (*structured analysis and design technique*) digunakan sebagai alat bantu untuk analisis sistem.

d. *Phase* 4 : *Decision Analysis Phase*

Analisis keputusan digunakan untuk menilai beberapa alternatif kemungkinan pengembangan sistem sesuai dengan kebutuhan. Beberapa pertanyaan penting yang harus dilakuka adalah :

a) Seberapa besar sistem akan di-*komputerisasi*?
b) Apakah sistem baru akan dibeli atau dibangun sendiri ?
c) Apakah sistem akan dibangun berdasarkan jaringan *internal* atau berbasiskan *web*?

Masing masing alternatif kemungkinan pengembangan sistem kemudian harus diputuskan dengan berbagai *studi* kelayakan seperti :

a) Kelayakan *teknis*, apakah SDM yang ada menguasai secara *teknis*?
b) Kelayakan *Operasional*, untuk menilai sudahkah sesuai dengan kebutuhan pengguna?
c) Kelayakan *ekonomis*, untuk menilai dari segi keuangan apakah sudah *ekonomis*?
d) Kelayakan jadwal, untuk menilai dari segi penjadwalan atau waktu yang ditetapkan?
e) Kelayakan resiko, untuk menilai kesuksesan alternatif yang dipilih?

e. *Phase* 5 : *Desain Phase*

Setelah diperoleh proposal sistem yang disetujui, maka dapat mulai dilakukan proses *desain* dari sistem target. Tujuan dari tahap ini adalah untuk men-*transformasi*-kan *business requirement statement* menjadi *spesifikasi* desain untuk proses *construksi*. Dengan kata lain, tahap *desain* menyatakan bagaimana *teknologi* akan digunakan dalam sistem yang baru. Tahap ini memerlukan ide dan *opini* dari pengguna, *vendor*, dan *spesialis IT*.Pada akhir tahap ini masih terdapat beberapa alternatif keputusan mengenai proyek walaupun pembatalan proyek jarang dilakukan pada tahap ini (kecuali benar - benar *over budget* atau sangat terlambat dari jadwal). Perubahan lingkup menjadi lebih kecil masih dapat terjadi. Selain itu, mungkin juga terjadi perubahaan ulang jadwal untuk menghasilkan solusi yang lebih lengkap.

f. *Phase* 6 : *Construction Phase*

Construction Phase ialah tahapan melaksanakan pengujian pada komponen sistem secara *individu* dan sistem secara keseluruhan. Tujuan dari tahap ini adalah :

a) Membangun dan menguji sistem yang memenuhi *business requirement* dan *spesifikasi desain.*
b) Meng-*implementasi*-kan penghubung antara sitem baru dan sistem lama, termasuk *instalasi* dari *software* yang dibeli atau disewa.

Pada tahap ini dilakukan *konstruksi basis data*, *program aplikasi*, dan penghubung antara sistem dan pengguna. Beberapa dari komponen ini telah ada sebelumnya. Setelah dilakukan pengujian, maka sistem dapat mulai di-*implementasi*-kan.

g. *Phase* 7 : *Implementation Phase*

Implementation ialah menerapkan hasil rancangan yang telah disusun sedemikian rupa ke dalam sistem perusahaan untuk mendapatkan kondisi yang sesuai dengan kebutuhan perusahaan. *Input* dari tahap ini adalah *sistem*

fungsional dari tahap *konstruksi*. Analis harus mampu menyediakan *transisi* yang sederhana dari sistem lama ke sistem baru dan membantu pengguna menghadapi masalah utama saat mulai menggunakan sistem baru. Selain itu, analis harus melatih pengguna, menuliskan cara-cara penggunaan *manual*, meng-*input file* dan basis data, dan melakukan tes akhir. Pengguna sistem akan memberikan *feedback* bagi tim proyek sebagai masalah baru dan isu baru. *Output* dari tahap ini adalah *sistem operasional* yang akan memasuki tahap operasi dan pendukung dalam siklus hidup perusahaan.

h. *Phase* 8 : *Operation and Support Stage Phase*

Sistem pendukung *teknis* berkelanjutan bagi para pengguna, seperti kebutuhan *maintenance* untuk memperbaiki kesalahan, penghilangan, dan kebutuhan - kebutuhan baru. Aktivitas - aktivitas dalam sistem pendukung :

a) *Assisting users* : tak peduli seberapa baiknya pelatihan yang diberikan pada pengguna, pasti tetap akan ada kebutuhan *asistensi* tambahan bagi para pengguna terutama saat muncul masalah baru, muncul tambahan pengguna, dan lain-lain.

b) *Fixing software defects* : memperbaiki kesalahan - kesalahan yang muncul saat *operasional* maupun pengujian.

c) *Recovering system* : kegagalan sistem dapat menyebabkan terjadinya kehilangan atau '*crash*' data yang memerlukan perbaikan pada sistemnya seperti pemasukan ulang *file* basis data dan me-*restart* ulang sistem.

d) *Adapting the system to new requirements* : kebutuhan yang selalu berkembang menimbulkan kebutuhan akan perbaikan berkelanjutan dalam sistem informasi agar sistem yagn ada dapat terus mengikuti perubahan yang sedang terjadi seperti munculnya kebutuhan bisnis baru, masalah *teknis* baru, atau kebutuhan *teknologi* baru.

Dari delapan *phase* diatas, kami hanya membahas *phase* 1 sampai dengan *phase* 6, yaitu: *Preliminary Investigation Phase, Problem Analysis Phase,*

Requirement Analysis Phase, Decision Analysis Phase, Desain Phase, Construction Phase.

2.4 Metodologi Berorientasi Objek

Metodologi berorientasi objek diperkenalkan pada tahun 1980, menggunakan perangkat kerja dan teknik-teknik yang dibutuhkan dalam pengembangan sistem, yaitu *dynamic* dan *static object oriented model, state transition diagram* dan *case scenario*[6].

Fokus utama metodologi ini pada objek, dengan melihat suatu sistem terdiri dari objek yang saling berhubungan. Objek dapat digambarkan sebagai benda, orang, tempat dan sebagainya yang mempunyai *atribut* dan *metode*. Metodologi terdiri dari pembuatan model dari *domain aplikasi*, kemudian menambahkan *detail implementasi* pada saat *desain* dari suatu *sistem.*

2.4.1 Karakteristik Khusus Metodologi Berorientasi Objek

Tahap-tahap metodologi berdasarkan *Sistem Development Life Cycle* (SDLC) digunakan dengan memperhatikan karakteristik khusus berorientasi objek, diantaranya adalah :

a. Analisis

Analisis berorientasi objek dimulai dengan menyatakan suatu masalah, analis membuat model situasi dari dunia nyata, menggambarkan sifat yang penting. Analis harus bekerja dengan pihak yang membutuhkan sistem untuk memahami masalah tersebut. Model analisis adalah *abstraksi* yang ringkas dan tepat dari apa yang harus dilakukan oleh sistem, dan bagaimana melakukannya. Objek dalam model harus merupakan konsep *domain* dari aplikasi, dan bukan merupakan *implementasi komputer* seperti struktur data. Model yang baik harus dipahami dan ditanggapi oleh ahli aplikasi. Empat kesulitan yang merupakan gangguan utama sistem adalah memahami *problem domain*, komunikasi antara pihak yang berkaitan, perubahan *kontinyu*, dan *reuse* (penggunaan kembali).

b. *Desain*

Desain Berorientasi Objek atau *Object Oriented Design* (OOD) merupakan tahap lanjutan setelah Analisis Berorientasi Objek dimana tujuan sistem diorganisasikan ke dalam *sub-sistem* berdasar *struktur analisis* dan *arsitektur* yang dibutuhkan. *System designer* menentukan karakteristik penampilan secara optimal, menentukan strategi memecahkan masalah, dan menentukan pilihan alokasi sumberdaya. Sebagai contoh, *system designer* mungkin menentukan perubahan pada *screen* untuk *workstation* yang memerlukan kecepatan serta *resolusi* lebih tinggi. *Desain* model berdasarkan model analisa tetapi berisi *detail implementasi*. Fokus dari *object design* adalah perencanaan struktur data dan *algoritma* yang diperlukan untuk *implementasi* setiap kelas. *Objek domain aplikasi* dan *objek domain komputer* dijelaskan dengan menggunakan konsep dan *notasi* berorientasi objek yang sama.

c. *Implementasi*

Kelas, objek, dan *relasi*-nya dikembangkan dalam tahap pembuatan *desain objek* yang pada akhirnya diterjemahkan ke dalam bahasa pemrograman, basis data, dan *implementasi* perangkat keras. Hal yang penting dalam tahap *implementasi* adalah mengikuti penggunaan perangkat lunak yang baik.

Konsep berorientasi objek dapat berlaku pada siklus hidup dari analisis sampai *implementasi*. Kelas yang sama dipergunakan dari satu tahap ke tahap lain tanpa perubahan *notasi*, walaupun menambahkan *detail implementasi* pada tahap akhir. Beberapa kelas tidak merupakan bagian dari analisis, tetapi baru dikenali pada tahap *desain* atau *implementasi*.

2.5 *Tools* menggunakan UML

UML (*Unified Modeling Language*) adalah Metodologi kolaborasi antarametoda-metoda Booch, OMT (*Object Modeling Technique*), serta OOSE (*Object Oriented Software Enggineering*) dan beberapa metoda lainnya, merupakan metodologi yang paling sering digunakan saat ini untuk analisa dan

perancangan sistem dengan metodologi berorientasi objek mengadaptasi maraknya penggunaan bahasa "pemrograman berorientasi objek" (OOP)[7].

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

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi *piranti lunak*, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan *piranti lunak* dalam bahasa - bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax/semantik*. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram *piranti lunak*. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: *Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique),* dan *Ivar Jacobson OOSE (Object-Oriented Software Engineering).*

Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: *metodologi booch,metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock*, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pen-*desain*-an berorientasi objek. Masing - masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/ perusahaan lain yang menggunakan metodologi yang berlainan.

Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikatan metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 di-*releasedraft* pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object Management Group* (OMG).Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML yang terdiri dari *structural classification, dynamic behavior,* dan model *management,* bisa kita pahami dengan mudah apabila *Main concepts* bisa kita pandang sebagai *term* yang akan muncul pada saat kita membuat *diagram*. Dan *view* adalah kategori dari *diagaram* tersebut.

Sejak itulah UML menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek, UML mendefinisikan *diagram - diagram* sebagai berikut :

a. *Activity Diagram*
b. *Use Case Diagram*
c. *Class Diagram*
d. *Sequence Digaram*
e. *Communication Diagram*
f. *State Machine Diagram*
g. *Component Diagram*
h. *Deployment Diagram*
i. *Composite Structure Diagram*
j. *Interaction Overview Daigram*
k. *Object Diagram*
l. *Package Diagram*
m. *Timing Diagram*

Namun penulis tidak membahas semua diagram yang diatas tetapi sesuai kebutuhan analisis dan perancangan sistem yang dibuat. Dan diagram yang kami gunakan, yaitu: *Activity Diagram, Use Case Diagram, ERD, dan Sequence Diagram.*

2.5.1 Analisis Berorientasi Objek

OOSE dikembangkan oleh Ivar Jacobson adalah metode desain berorientasi objek yang melibatkan *use case*. *Use case*merupakan skenario untuk memahami *requirement user* terhadap sistemmenggambarkan interaksi antara *user* dengan sistem, menggambarkan tanggung jawab dan keluaran sistem pada pengguna dapat digambarkan dengan teks tanpa aliran kejadian, teks dengan aliran data, dan formal dengan *pseudo code*. Metode yang paling banyak digunakan adalah menggunakan UML (*Unified Modelling Language*)[8].

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka UML lebih cocok untuk penulisan *piranti lunak* dalam bahasa–bahasa berorientasi objek seperti *C++, Java,* atau *VB.NET*. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C.

Seperti bahasa–bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram *piranti lunak*. Setiap bentuk memiliki makna tertentu dan UML *syntax* mendefinisikan bagaimana bentuk - bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari tiga notasi yang telah ada sebelumnya : Grady Booch OOD (*Object-Oriented Design*), Jim

Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*).

Dari penjelasan diatas penulis menjabarkan landasan teori *diagram - diagram* UML yang menjadi alat bantu pada tahap analisis berorientasi objek.

a. *Activity Diagram*

Activity diagram memiliki pengertian yaitu lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada *business modeling* untuk memperlihatkan urutan aktifitas proses bisnis. Memiliki struktur diagram yang mirip *flowchart* atau data *flow* diagram pada perancangan terstruktur. Memiliki pula manfaat yaitu apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. Dan *activity* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case* diagram.

(a) *Start Point (initial node),* adalah tanda yang mengawali sebuah kegiatan yang diletakan pada pojok kiri atas.

(b) *End Point (activity final node),* adalah tanda yang mengakhiri sebuah kegiatan dalam *activity diagram.*

(c) *Activities,* menggambarkan suatu proses/ kegiatan bisnis.

(d) *Fork* (pencabangan), digunakan untuk menunjukan kegiatan yang dilakukan secara pararel atau untuk menggabungkan dua kegiatan pararel menjadi satu.

(e) *Decision Point,* menggambarkan pilihan untuk pengambilan keputusan, *true* atau*false.*

(f) *Swimlane,* pembagian *activity diagram* untuk menunjukan siapa melakukan apa.

b. Analisa Dokumen Keluaran

Analisa dokumen keluaran merupakan analisa mengeai keluaran-keluaran dokumen yang dihasilkan melalui proses - proses yang ada dalam sistem berjalan

c. Analisa Dokumen Masukan

Analisa dokumen masukan adalah untuk mengetahui dokumen-dokumen apa saja yang digunakan sebagai masukan data pengolahan sistem informasi kepegawaian pada sistem yang berjalan.

d. Use Case Diagram

Use Case Diagram adalah *representasi visual* yang mewakili interaksi antara pengguna dan sistem informasi dalam UML. Dengan kata lain, *Use case diagram* dengan nyata menguraikan siapa yang akan menggunakan sistem dan dengan cara apa pemakai dapat saling berhubungan dengan sistem. *Use case diagram* terdiri dari[9] :

a) *Use Case* adalah deskripsi fungsi dari sebuah sistem dari 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. Penamaan *use case* sesuai dengan tujuan yang dicapai dari hasil interaksinya dengan *actor. Use case* biasanya menggunakan kata kerja.

b) *Actor* adalah *abtractions* dari orang atau sistem yang lain, yang mengaktifkan fungsi target sistem. Untuk mengidentifikasikan *actor*, harus ditentukan pembagian tenaga kerja dan tugas–tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol terhadap *use case*.

c) *Relationship*, relasi anatara *actor* dengan *use case* pada *use case diagram* digambarkan dalam bentuk garis. Relasi antara *actor* dengan *use case* disebut dengan asosiasi. Asosiasi adalah sebuah relasi antara *actor* dengan *use case* dimana sebuah interaksi terjadi diantara meraka.

2.5.2 Perancangan Berorientasi Objek

Perancangan berorietasi objek adalah suatu pendekatan yang digunakan untuk memspesifikasi kebutuhan–kebutuhan sistem dengan mengkolaborasi objek – objek, atribut – atribut dan metode – metode yang ada. Fokus dari desain objek adalah perencanaan struktur data dan *algoritma* yang diperlukan untuk implementasi setiap kelas.

Diagram – diagram UML yang digunakan penulis dalam merancang sistem berorientasi objek adalah :

a. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/ properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/ fungsi).

Diagram kelas atau *class diagram* sangat membantu dalam visualisasi kelas dari suatu sistem. Hal ini disebabkan karena *class* adalah deskripsi kelompok objek - objek dengan *atribut*, prilaku dan relasi yang sama. Disamping itu class diagram bisa memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class* - *class* yang ada dan relasinya satu dengan lainnya.

Komponen - komponen *Class Diagram* :

a) Kelas

Kelas didefinisikan sebagai kumpulan atau himpunan objek-objek yang dengan *atribut* dan *operation* yang sama. Obyek adalah orang,

benda, tempat, kejadian atau konsep-konsep yang ada di dunia nyata yang penting bagi suatu aplikasi perangkat lunak/ perangkat keras. Setiap obyek memiliki keadaan sesaat prilaku. Sebuah obyek adalah kondisi obyek tersebut yang dinyatakan dalam *atribut/ propertis*. Sedangkan prilaku sebuah obyek mendefinisikan bagaimana sebuah obyek bertindak atau berinteraksi. Perilaku sebuah obyek dinyatakan dalam *operation*.

b) Atribut

Atribut adalah data yang dimiliki suatu obyek dalam suatu kelas, misalnya kelas manusia, yang memiliki atribut nama dan umur.

c) Operasi

Operasi *(Operation)* adalah sesuatu yang bisa dilakukan oleh sebuah kelas (tingkah laku sebuah objek) atau fungsi yang dapat diaplikasikan ke suatu objek dalam kelas. Misalnya suatu objek manusia pasti memiliki fungsi – fungsi seperti tersenyum, marah , makan, minum dan sebagainya. Operasi yang sama dapat diterapkan pada kelas yang berbeda, misalnya fungsi makan dapat diterapkan pada kelas manusia maupun pada kelas hewan.

d) *Association*

Association menunjukan hubungan antara masing–masing kelas. Setiap *association* mempunyai dua *association end*. Masing–masing *end* dihubungkan ke satu kelas dari kelas – kelas dalam *association*. Sebuah *end* dapat dibuat lebih jelas dengan memberikan nama dengan sebuah label. Label ini disebut dengan *role name (association end* sering disebut *role)*. Sebuah *association end* juga mempunyai atau memiliki *"multiplicity"*, *multiplicity* ini menunjukan berapa banyak objek yang berpartisipasi dalam suatu relasi.

b. *Squence Diagram*

Sequence diagram adalah suatu diagram UML yang memodelkan logika dari suatu *use case* dengan menggambarkan intraksi berupa pengirim pesan antar objek dalam urutan waktu[10].

Diagram ini secara khusus berasosiasi dengan *use case diagram*, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuai di dalam *use case.*

a) Obyek/ *Participant*

Objek diletakan di dekat bagian atas diagram dengan urutan dari kanan. Setiap *Participant* terhubung denag garis titik – titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation. Activation* mewakili sebuah eksekusi operasi dan *participant.* Panjang kotak ini berbanding lurus dengan durasi *activation.* Setiap *lifeline* mempunyai *activation bar* yang menunjukan kapan sebuah *participant* aktif pada intraksi. *Activation bar* adalah *optional* di UML, meskipun sangat berguna dalam klarifikasi perilaku.

b) *Actor*

Menggambarkan orang yang sedang berinteraksi dengan sistem.

c) *Message*

Sebuah *message* bergerak dari satu *participant* ke *participant* yang lain dan dari satu *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri. Sebuah *message* bisa jadi simple, *synchronous* atau *asynchronous. Message* yang *simple* adalah sebuah perpindahan *(transfe) control* dari satu *participant* ke *participant* yang lainnya. Jika sebuah *participant* mengirimkan sebuah *message synchronous*, maka jawaban atas message tersebut akan ditunggu sebelum diproses dengan urusannya. Namun jika *message asynchronous* yang dikirimkan, maka jawaban atas *message* tersebut tidak perlu ditunggu. *Message* datang dari sumber yang tidak ditentukan disebut dengan *found message.*

(a) *Loop*

Menggambarkan dari kegiatan yang dilakukan secara berulang – ulang.

(b) *Recursive* (Rekursi)

Kadang kala sebuah objek mempunyai sebuah *operation* kepada dirinya sendiri. Hal ini disebut dengan *recursive* dan menjadi arus utama banyak bahas pemograman. Asumsikan sebuah objek pada sistem adalah kalkulator dengan *operation*-nya menghitung bunga. Untuk menghitung bunga berbunga selama periode tertentu makan objek tersebut perlu melakukan *operation* terhadap dirinya sendiri guna menghitung bunga. Untuk menggambarkan hal tersebut, perlu ditambahkan lapisan kotak kecil pada *activation*. Arah panah perlu dibuat sedemikian rupa sehingga arahnya kembali ke kotak kecil tersebut.