PROYEK SISTEM INFORMASI

giving and caring the world

TELKOM POLYTECHNIC

PROYEK SISTEM INFORMASI

giving and caring the world

POLITEKNIK TELKOM
BANDUNG
2009

Tim Penyusun & Editor:

Agus Pratondo
Eddy Prasetyo Nugroho
Ine Gartina
Siska Komala Sari

KATA PENGANTAR

Assalamu'alaikum Wr. Wb

Segala puji bagi Allah SWT karena dengan karunia-Nya *courseware* ini dapat diselesaikan.

Atas nama Politeknik Telkom, kami sangat menghargai dan ingin menyampaikan terima kasih kepada penulis, penerjemah dan penyunting yang telah memberikan tenaga, pikiran, dan waktu sehingga *courseware* ini dapat tersusun.

Tak ada gading yang tak retak, di dunia ini tidak ada yang sempurna, oleh karena itu kami harapkan para pengguna buku ini dapat memberikan masukan perbaikan demi pengembangan selanjutnya.

Semoga *courseware* ini dapat memberikan manfaat dan membantu seluruh Sivitas Akademika Politeknik Telkom dalam memahami dan mengikuti materi perkuliahan di Politeknik Telkom.
Amin.

Wassalamu'alaikum Wr. Wb.

Bandung, Maret 2009

Christanto Triwibisono
Wakil Direktur I
Bidang Akademik & Pengembangan

DAFTAR ISI

I PENDAHULUAN

Overview

Mata Kuliah Proyek SI adalah sebuah matakuliah rangkuman dari mata kuliah yang pernah diperoleh di semester sebelumnya dan merupakan pengantar mata kuliah di semester yang akan datang. Rangkuman tersebut disajikan dalam sebuah aplikasi berdasarkan pada kasus nyata. Bab ini menitikberatkan kepada hal-hal yang akan dilakukan dalam mata kuliah ini.

Tujuan

1. Mahasiswa mengetahui apa dan bagaimana menjalankan mata kuliah Proyek SI
2. Mahasiswa mampu menentukan langkah awal untuk mengerjakan Proyek SI
3. Mahasiswa dapat memperoleh gambaran tema proyek yang akan dibuat

1.1 Pendahuluan

Proyek Sistem Informasi merupakan sebuah mata kuliah yang bertujuan untuk mengarahkan mahasiswa kepada penerapan ilmu teori dan praktik ke dalam dunia nyata, serta sebaliknya menuntun dan mengarahkan mahasiswa agar dapat menerjemahkan kasus nyata ke dalam sebuah model, desain, atau aplikasi.

Sebagai media untuk menerapkan ilmu teori dan praktik yang diperoleh sebelumnya akan menjadikan kuliah ini menarik, karena dengan demikian mahasiswa akan menjadi lebih proaktif dalam meningkatkan pemahaman mereka terhadap mata kuliah yang pernah diperoleh sebelumnya. Sedangkan Proyek SI sebagai media pengantar pemahaman mata kuliah yang ada di semester selanjutnya akan menjadikan mahasiswa menaruh perhatian dan konsentrasi yang optimal kepada mata kuliah ini.

Tidak bisa dipungkiri bahwa perkembangan teknologi informasi dari hari ke hari semakin membuat kita selalu merasa 'tertinggal' jika kita tidak mengikuti perkembangan dan berusaha untuk terlibat di dalam proses perkembangan tersebut.

Saat ini penggunaan teknologi informasi sudah mutlak diperlukan dalam dunia bisnis, pemerintahan, pendidikan, kesehatan dan lainnya. Bisa dikatakan bahwa hampir seluruh sendi kehidupan manusia dapat didukung dengan adanya teknologi informasi.

Sebagai mahasiswa jurusan Manajemen Informatika anda selayaknya menjadi bagian dari pengembangan pemanfaatan teknologi informasi tersebut, bukan hanya sekedar *user* atau pengguna produk teknologi informasi.

Mata Kuliah Proyek Sistem Informasi ini merupakan sarana latihan untuk terlibat aktif dalam pengembangan teknologi informasi tersebut.

1.2 Sikap yang Diharapkan dari Mahasiswa

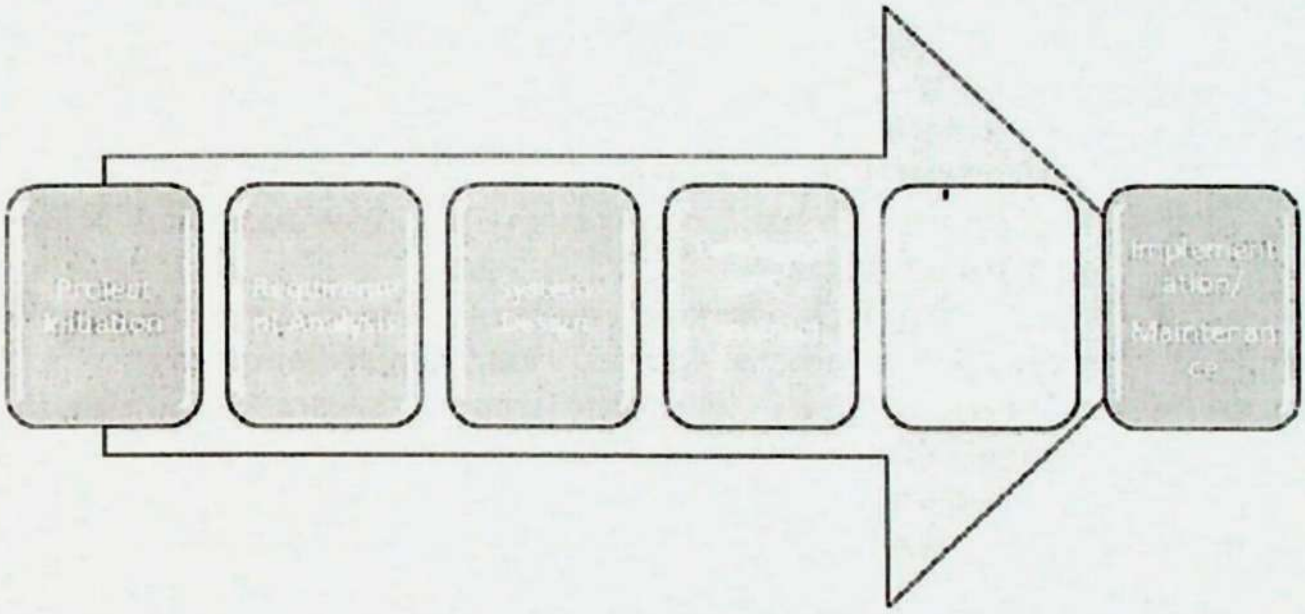
Selama mengikuti kuliah ini, mahasiswa diharapkan dapat berpartisipasi aktif dengan bersikap proaktif, inovatif, kreatif dan mengembangkan kerjasama tim yang baik.

1.3 Alur Proyek

Dalam mengerjakan tugas mata kuliah proyek Sistem Informasi ini, mahasiswa akan diarahkan untuk menerapkan tahap-tahap pembangunan perangkat lunak, yaitu tahapan-tahapan SDLC (*System Development* Life Cycle).

SDLC dapat berupa pembuatan suatu sistem baru yang tadinya belum ada menjadi ada, atau merupakan penambahan atau pengubahan modul pada sistem yang telah ada.

Secara umum, tahapan SDLC dapat dilihat pada gambar berikut:



Gambar 1.1 Tahapan umum SDLC

Hal-hal yang dilakukan pada tahapan-tahapan SDLC adalah sebagai berikut:

a. *Project Initiation* (Inisiasi proyek)
Pada tahap ini dilakukan pendefinisian proyek, penentuan tema dan proses bisnis secara umum dari proyek yang akan dikerjakan.

b. *Requirement Analysis* (Analisa Kebutuhan)
Pada tahap analisa kebutuhan sistem, terdapat kegiatan-kegiatan studi kelayakan terhadap sistem yang akan dibangun. Disini tim proyek akan mencari tahu mengenai sistem yang sudah berjalan, mengidentifikasi permasalahan-permasalahan yang muncul pada

sistem yang ada, kemudian mencari solusi atas masalah yang ada untuk kemudian diterjemahkan sebagai kebutuhan sistem yang baru berdasarkan hasil studi kelayakan.
Pada tahap ini, tim proyek akan mendefinisikan kebutuhan sistem yang baru berdasarkan hasil analisa terhadap kondisi sistem yang sedang berjalan.

c. *System Design* (Perancangan Sistem)
Setelah dokumen analisa kebutuhan sistem telah terdefinisi dengan baik dan sesuai dengan keinginan *user,* maka kebutuhan sistem tersebut diterjemahkan kedalam sebuah desain sistem yang akan memudahkan tim proyek untuk menerjemahkannya ke dalam bentuk program.
Perancangan sistem ini dapat terbagi menjadi dua macam, yaitu:
   a. Perancangan Logis (*Logical Design)*
      Perancangan logis merupakan proses pendefinisian objek-objek sistem dan keterhubungannya dari dunia nyata berdasarkan kebutuhan informasi serta sistem dari organisasi atau badan yang bersangkutan.
   b. Perancangan Fisik (*Physical Design*)
      Merupakan proses untuk mengimplementasikan hasil perancangan logis ke dalam model secara fisik dengan menggunakan perangkat lunak pemodelan yang dipilih.

d. *Development/Construction* (Pengembangan/Pembangunan Proyek)
Pada tahap ini dilakukan proses penerjemahan desain/pemodelan sistem ke dalam bahasa pemrograman tertentu yang dipilih.

e. *System Testing/Quality Assurance* (Pengujian sistem/Jaminan Kualitas)
Sistem yang telah dibangun pada tahap *Developement/Construction* akan diuji fungsi-fungsinya sebelum diimplementasikan.
Pengujian sistem atau penjaminan kualitas ini dapat dilakukan bertahap. Tahapan pengujian secara umum adalah sebagai berikut:
   a. Pengujian unit/modul (*Unit Test*). Pengujian dilakukan per unit/modul dari sistem yang telah dibuat
   b. Pengujian Sistem (*System Test*). Pengujian dilakukan secara terintegrasi pada sistem yang modul-modulnya telah lengkap sehingga diketahui apakah sistem secara keseluruhan telah berjalan dengan baik atau belum.

c. Pengujian penerimaan pengguna (*User Acceptance Test*). Pada pengujian ini, *user* akan menguji sistem secara langsung apakah sistem yang dibuat telah sesuai dengan kebutuhan *user* atau tidak.

f. *Implementation/Maintenance* (Implementasi/Perawatan)
Setelah melalui pengujian, sistem akan diimplementasikan di lingkungan *user* yang membutuhkan sistem tersebut untuk kemudian digunakan sebagai alat bantu berjalannya organisasi atau perusahaan yang bersangkutan. Selama sistem diimplementasikan di lingkungan *user*, sebaiknya selalu dilakukan perawatan sistem secara rutin.

1.4 Ruang Lingkup Proyek SI

Proyek Sistem Informasi yang dapat dikerjakan oleh mahasiswa meliputi hal-hal berikut:

1. Pembangunan Aplikasi
2. Pengembangan Aplikasi
3. Perancangan Basis Data

Untuk tingkat kesulitan atau kompleksitas dari proyek yang dikerjakan akan membutuhkan kesepakatan antara dosen pengajar dan mahasiswa.

1.5 Pengerjaan Proyek

Mahasiswa akan mengerjakan proyek ini dengan berkelompok, dimana jumlah anggota kelompok yang diharapkan adalah berkisar antar 3 hingga 5 orang per kelompok.

Dosen Pembimbing untuk pengerjaan proyek ini adalah dosen pengajar mata kuliah yang bersangkutan.

Presentasi hasil proyek akan dilakukan di akhir perkuliahan.

Secara umum, alur pengerjaan proyek Sistem Informasi adalah sebagai berikut:

1. Pembentukan Kelompok
2. Penentuan Tema dan Lokasi Survey
3. Pembuatan Surat Pengantar Survey
4. Survey ke Lokasi
5. Melakukan tahap Analisis Kebutuhan
6. Melakukan Tahap Perancangan Sistem
7. Melakukan Tahap Implementasi /*Coding*
8. Melakukan Tahap Pengujian
9. Melakukan Presentasi hasil pembuatan Proyek

1.6 Faktor Penilaian

Penilaian untuk Mata Kuliah ini diambil sesuai dengan proses yang dilalui, yaitu mulai dari pembentukan kelompok hingga presentasi ada faktor penilaiannya. Bobot penilaian tergantung hasil kesepakatan pada rapat koordinasi dosen pengajar mata kuliah Proyek Sistem Informasi.

1.7 Contoh Tema

Contoh Tema yang dapat diambil, sebagai berikut:

1. Sistem Mini Market
2. Sistem Agen Perjalanan
3. Sistem Layanan Laundry
4. Sistem Toko Bangunan
5. Sistem Warnet
6. Dan lain-lain

2 Manajemen Proyek

Overview

Dalam Pengelolaan Proyek Perangkat Lunak, seringkali terdapat mendapat masalah manajemen perangkat lunak seperti penyelesaian produk perangkat lunak yang terlambat, tidak handal sesuai dengan harapan *user*, sering melebihi dari estimasi awal pada budgeting dan banyak karakter dari perilaku pengembang yang kurang sesuai harapan proyek. Hal ini terjadi dikarenakan sistem perangkat lunak yang dibangun seringkali berkembang dan bersifat inovatif.

Untuk itu sebagai manajer Perangkat lunak tidak mungkin membuat deskripsi pekerjaan yang standar. Hal ini tergantung terhadap besarnya organisasi pengembang dan produk perangkat lunak yang dikembangkan. Tapi paling tidak manajer harus mengikuti beberapa tahapan dan aktifitas sebagai berikut:

- Penulisan Proposal (*Proposal Writing*)
- Perkiraan Anggaran Proyek (*Project Costing*)
- Perencanaan dan Penjadwalan Proyek (*Project Planning & Scheduling*)
- Monitor dan review Proyek (*Project monitoring and reviews*)
- Pemilihan dan Evaluasi Proyek (*Personnel Selection and Evaluation*)
- Penulisan dan Presentasi Laporan Keseluruhan Proyek (*Report Writing and presentation*)

Tujuan

1. Bab ini akan memberikan beberapa tahapan manajemen proyek yang akan dilakukan oleh mahasiswa
2. Mahasiswa dapat menjadi pelaku proyek sebagai manajer proyek pada kegiatannya, dari pembuatan proposal sampai pembuatan laporan proyek

2.1 Penulisan Proposal (*Proposal Writing*)

Proposal menggambarkan objektifitas atau tujuan dari proyek dan bagaimana proyek akan dikelola dan dilaksanakan. Biasanya akan memberikan perkiraan penjadwalan dan anggaran yang sehingga dapat memperkirakan pula kebutuhan orang yang ahli yang dapat masuk dalam tim proyek. Dalam mata kuiah ini kita tidak membahas budgeting atau perkiraan anggaran dan kebutuhan Tenaga Ahli.

Secara sederhana, sebuah proposal proyek terdiri dari:

a. **Judul**
   Halaman ini berisi judul proposal. Halaman ini adalah pintu gerbang bagi proposal. Diharapkan dengan membaca halaman ini, para pemangkukepentingan tertarik untuk membacanya. Untuk itu, judul proposal disarankan bersifat atraktif atau *eye catching* dan menimbulkan rasa penasaran untuk membaca isinya.

b. **Ringkasan Eksekutif**
   Halaman ini berisi ringkasan isi proposal. Dengan membancanya, para pemangkukepentingan sudah mendapatkan gambaran tentang isi dari proposal. Sebaiknya, halaman ini berisi ringkasan tentang latar belakang proyek, rumusan masalah yang akan diselesaikan dalam proyek, tujuan atau produk akhir dari proyek, tahapan penyelesaian proyek, dan jadwal penyelesaian proyek. Bila diperlukan, disajikan juga sumber daya yang diperlukan untuk menyelesaikan proyek.

c. **Lembar Pengesahan**
   Halaman ini berisi pernyataan persetujuan para sponsor proyek. Persetujuan ini diperlukan untuk mendapatkan komitmen dukungan dari seluruh pemangkukepentingan proyek.

d. **Pendahuluan**

   a. **Latar Belakang Proyek**

      Bagian ini berisi latar belakang pemikiran diperlukannya proyek ini. Bagian ini menjawab pertanyaan "Mengapa proyek ini dilaksanakan"

   b. **Rumusan Masalah**

      Bagian ini berisi permasalahan yang muncul sehingga proyek ini perlu dilaksanakan. Ia biasanya berupa pertanyaan atau pernyataan yang secara umum mencoba menjawab latar belakang proyek.

c. **Batasan Masalah**

Bila diperlukan, bagian ini dapat dihadirkan. Bagian ini berisi batasan-batasan dari proyek yang akan dikerjakan, misalnya: apa yang dikerjakan dan apa yang tidak dikerjakan

d. **Tujuan**

Bagian ini berisi tujuan dilaksanakannya proyek berikut produk akhir dari proyek.

e. **Manfaat Proyek**

Bila diperlukan, bagian ini dapat dihadirkan. Bagian ini berisi keuntungan yang akan dirasakan oleh para *stakeholder* dengan dilaksanakannya proyek ini.

e. **Landasan Teori, Literatur Penunjang, atau Kajian Pustaka**

Judul landasan teori digunakan bila produk proyek ini bersifat penciptaan teori baru, misalnya proyek algoritma. Untuk proyek yang lebih bersifat umum, lebih baik digunakan judul literatur penunjang atau kajian pustaka.

Bagian ini berisi uraian yang melengkapi latar belakang masalah, sekaligus memberikan *review* tentang pustaka yang telah dibaca selama masa pencarian solusi terhadap masalah (lihat sub bagian Merencanakan Proyek).

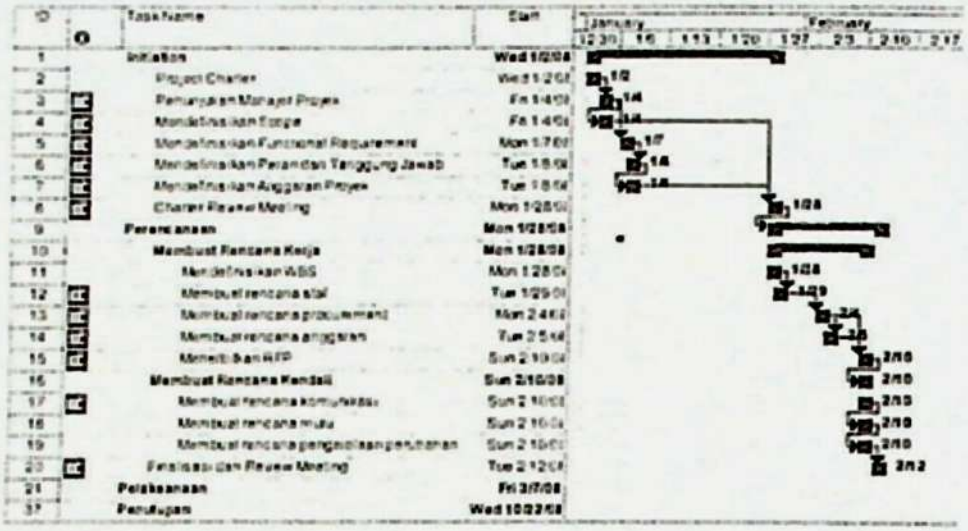
f. **Metode Pelaksanaan Proyek**

Bagian ini berisi tahapan yang akan dilaksanakan untuk menyelesaikan masalah atau untuk membuat produk. Tahapan yang dituliskan di sini bersumber dari proses yang dilakukan di dalam tahap Merencanakan Proyek. Tahapan ini dapat pula berisi siklus pengembangan *Software* atau siklus pengembangan produk seperti tahapan "pendefinisian kebutuhan, desain, penulisan kode sumber, pengujian, dan implementasi".

g. **Jadwal Pengerjaan Proyek**

Bagian ini menyajikan kalenderisasi tahapan pelaksanaan proyek. Terkait dengan itu, diperlukan pengetahuan tentang lama waktu suatu tahapan akan dikerjakan. Misalnya, tahapan pendefinisian kebutuhan akan dikerjakan 5 hari, dan sebagainya.

Bila sudah menguasai Microsoft *Project* atau Mr *Project*, silakan gunakan kedua alat bantu tersebut untuk membuat Gantt Chart. Boleh juga mempergunakan Microsoft Visio dengan *template Project Management* untuk membuat *Time Line*.



Boleh juga menggunakan Microsoft Excel seperti contoh di bawah ini:

h. Daftar Pustaka

Bagian ini berisi senarai buku, *hyperlink*, dan e-book yang digunakan sebagai referensi di dalam proyek ini. Buku-buku yang tidak digunakan di dalam dokumen ini sebaiknya tidak dicantumkan.

Lebih detilnya lihat pada contoh Proposal (Lampiran)

2.2 *Project Planning*

Manajemen Proyek Pembangunan Perangkat Lunak yang efektif tergantung atas perencaaan dan proses pelaksanaan dalam proyek. Manajer proyek harus dapat mengantisipasi masalah yang muncul dan menyelesaikan masalah tersebut dengan solusi yang tentative. Sebuah perencanaan (*Planning*), yang digambarkan di awal proyek harus digunakan untuk mengarahkan jalannya proyek.

Pada perencanaan awal proyek, membuat tim proyek yang handal untuk pengerjaaan proyek. Hal ini harus melihat besar skala proyek terlebih dahulu, Menjalin hubungan dengan konsumen dari proyek perangkat lunak, membuat perencaaan kegiatan proyek dengan prosedurnya.

Project Planning memungkinkan aktifitas-aktifitas yang diatur dengan manajemen waktu. Perencanaan yang diminta pada aktifitas tersebut dengan bentuk sistem terkirim suatu laporan pelaksanaan yang tercatat. Hal ini untuk memandu dan menjamin pelaksanaan proyek perangkat lunak.

Proses *Project Planning* pada pengembangan Perangkat lunak ditunjukkan pada *pseudocode* dibawah ini:

```
Establish 'the Project constraints
Make initial assessments of the prject parameters
Define Project milestone and deliverables
While Project has not been completed or cancelled loop
        Draw up Project schedule
        Initiate activities according to schedule
        Wait (for a shile)
        Review Project progress
        Revise estimates of Project parameters
        Update the Project schedule
        Re-negotiate Project constrainsts and deliverables
        If (problems arise) then
           Initiate technical review and possible revision
        End If
End loop
```

Algoritma di atas menunjukkan bahwa proses perencanaan itu merupakan proses iteratif yang lengkap bila proyek dinyatakan selesai sesuai dengan batasan proyek yang dijamin atas penjadwalan kegiatan, staff yang tersedia dan anggaran yang berakibat pada proyek tersebut.

Project Planning mencakup beberapa hal sbb:

1. Pendahuluan, menggambarkan tujuan atau sasaran proyek dan membuat ruang lingkup batasan (spt anggaran dan waktu) yang berpengaruh pada manajemen proyek
2. Organisasi Proyek, menggambarkan cara-cara dalam tim pengembang diorganisasi, orang-orang yang terlibat dan perannya dalam tim
3. Analisa Resiko, menggambarkan resiko-resiko proyek yang mungkin terjadi dan strategi yang diusulkan untuk mengurangi tau menanggulangi resiko tersebut.

4. Kebutuhan Sumber daya *Software* dan *hardware*, menjelaskan kebutuhan sumber daya *Software* dan *hardware* saat pengembangan dilakukan dengan memperkirakan harga sumberdaya tsb.
5. Work Breakdown, menjelaskan turunan proyek ke dalam bentuk aktifitas atau kegiatan dan mengidentifikasi *milestone* dan suatu yang dikirimkan terkait dengan dokumen yang dihasilkan dengan aktifitas tersebut.
6. Penjadwalan Proyek, menggambarkan ketergantungan antar aktifitas, waktu estimasi yang diminta untuk mecapai *milestone* dan alokasi atau penempatan orang/tenaga ahli ke aktifitas-aktifitas tersebut
7. Mekanisme monitor dan pelaporan, menggambarkan laporan manajemen yang diproduksi ketika laporan dibuat dan penggunaan mekanisme monitor terhadap proyek tersebut.

2.3 *Activity Organization*

Manajer membutuhkan informasi yang tersedia seperti dokumen yang menggambarkan kegiatan yang dikerjakan karena suatu *Software* itu bersifat intangible. Tanpa dokumen ini maka tidak akan mungkin dapat mengatur proses, memperkirakan anggaran dan penjadwalan tidak dapat di*update*.
Dalam Proses *Planning*, proyek berpijak pada *milestone* yang memberikan proses aktifitas yang ada dengan disertakan suatu yang *deliverable*.
Dalam Proyek Perangkat Lunak, *milestone* yang digunakan mengadopsi model proses pembangunan Perangkat Lunak seperti model *Waterfall*. (Dari Tahapan *Requirement Analysis, Analysis, Design, Coding, Testing* ...).

2.4 *Project Schedulling*

Penjadwalan Proyek ini, manajer akan memperkirakan waktu dan sumber daya yang diminta untuk menyelesaikan aktifitas dan mengaturnya sehingga menjadi kegiatan yang terurut terhadap *milestone* yang sudah disepakati pada model proses tertentu.
Pada Penjadwalan Proyek ini melibatkan pembagian keseluruhan pekerjaan ke dalam bagian-bagian aktifitas dan waktu yang diminta sampai aktifitas tersebut selesai. Dan biasanya terdapat beberapa aktifitas yang paralel prosesnya, hal inijuga harus bisa dikoodinasi sehingga optimal.

Cara Penjadwalan Proyek sbb:

1. Mengidentifikasi "what" , apa saja yang dikerjakan (dengan Work Breakdow Structure /WBS). WBS adalah suatu teknik untuk membagi keseluruhan proyek ke dalam komponen-komponen sampai dengan tugas-tugasyang dapat dikelola terhadap nama tugas, durasi waktu, jadwal dan anggaran.
2. Mengidentifikasi "how much", berapa besar (Teknik Estimasi ukuran beban kegiatan terhadap waktu yang dibutuhkan
3. Mengidentifikasi ketergantungan antar kegiatan (Graf Ketergantungan dan *Network* Diagram)
4. Estimasikan Durasi total pekerjaan yang dilakukan sampai selesai (Actual Schedule)

Rangkuman

1. Manajemen Proyek diperlukan untuk membangun suatu Pembangungan Proyek Perangkat Lunak yang diantaranya terdiri dari *Project Planning*, Organization activity dan *Project Schedulling*
2. Dalam *Project Planning* terdapat milestone-*milestone* yang disertai beberapa aktifitas-aktifitas dimana mengadopsi model proses tertentu seperti model Waterfall
3. *Project Schedulling* membuat estimasi waktu dan aktifitas ang diminta sampai selesai terhadap keseluruhan kegiatan Proyek sehingga terurut sesuai *milestone* yang telah disepakati pada organisasi tim pengembang.

Lampiran

Contoh Proposal Proyek Teknik Komputer

Mobile Voucher
Solusi e-Marketing Berbasis Mobile

Proposal Proyek Sistem Informasi

Oleh:
Kelompok 3 Kelas PIS-0809

Ketua Tim: AAA
Anggota 1: BBB
Anggota 2: CCC
Anggota 3: DDD

Politeknik Telkom
2009

Mobile Voucher
Solusi e-Marketing Berbasis Mobile

Ringkasan Eksekutif

Dalam dunia yang makin kompetitif ini penggunaan model-model pemasaran baru dipercaya mampu mendorong minat orang untuk membeli produk yang ditawarkan. Penggunaan *voucher* belanja telah menjadi salah satu cara bagi perusahaan untuk mempromosikan produk-produknya dan menarik minat calon pembeli untuk membeli. Salah satu kendala yang dihadapi dalam penggunaan voucher belanja ini adalah pada distribusi *voucher* kepada calon pembeli dan pengelolaan *voucher* oleh calon pembeli tersebut. *Mobile voucher* mencoba menjawab permasalahan tersebut dengan menghadirkan *voucher* belanja elektronik yang dapat didistribusikan melalui jaringan telepon seluler dan disimpan oleh calon pembeli sebagai teks SMS.

Sistem *mobile voucher* terdiri dari 2 bagian, yaitu *voucher management System* dan *customer wallet*. *Voucher management System* adalah sistem komputer yang mengelola *voucher*, terdiri dari sebuah server yang terhubung dengan SMS gateway, bertugas untuk mendistribusikan *voucher* belanja. *Customer Wallet* adalah aplikasi Java SIM Card yang berfungsi untuk memvalidasi dan memverifikasi *voucher* yang dikirimkan melalui SMS oleh *Voucher Management System* berdasarkan kode keamanan tertentu.

Proyek ini bertujuan untuk membuat aplikasi *Customer Wallet*, sebagai bagian dari keseluruhan aplikasi *Mobile Voucher*. Produk yang dihasilkan adalah perangkat lunak *Customer Walet* berbasis *Java Smart Card API*.

Proyek ini dilaksanakan dengan tahapan-tahapan: pendefinisian kebutuhan, desain, koding, dan pengujian, dengan jangka waktu pengerjaan selama 2 bulan dan biaya sebesar Rp. 2.000.000.

Lembar Persetujuan

Mobile Voucher
Solusi e-Marketing Berbasis Mobile

Oleh:
Kelompok 3 Kelas PIS-0809

Ketua Tim: AAA
Anggota 1: BBB
Anggota 2: CCC
Anggota 3: DDD

Disetujui,
Tanggal 01/01/2009
Project Sponsor

Ir. Sering Bijaksana
Dosen

Politeknik Telkom
2009

Pendahuluan

Latar Belakang

Dalam dunia yang makin kompetitif ini penggunaan model-model pemasaran baru dipercaya mampu mendorong minat orang untuk membeli produk yang ditawarkan. Penggunaan *voucher* belanja telah menjadi salah satu cara bagi perusahaan untuk mempromosikan produk-produknya dan menarik minat calon pembeli untuk membeli. Salah satu kendala yang dihadapi dalam penggunaan voucher belanja ini adalah pada distribusi *voucher* kepada calon pembeli dan pengelolaan *voucher* oleh calon pembeli tersebut. *Diperlukan sebuah sistem yang dapat mendistribusikan voucher belanja secara lebih luas dan mudah dibawa-bawa oleh calon pembeli.*

Rumusan Masalah

Dipandang dari bidang keahlian teknik komputer, bagaimanakah mekanisme pendistribusian *voucher* belanja yang dapat menjangkau lebih banyak orang dan *voucher* tersebut mudah dibawa-bawa oleh calon pembeli? Solusi yang teridentifikasi dan dapat menjadi solusi terbaik bagi masalah di atas adalah dengan penggunaan perangkat handphone sebagai alat penyimpan informasi *voucher*, dengan pertimbangan: pemilik handphone sudah mencapai 150 juta orang di Indonesia, tersebar dari Sabang sampai Merauke dan dari kota sampai ke desa-desa. Selain itu, handphone juga sudah merupakan barang bawaan yang hampir wajib bagi setiap calon pembeli potensial. Dengan demikian, menjadi pertanyaan:
"Bagaimanakah perangkat mobile seperti handphone dapat digunakan sebagai solusi *voucher* elektronik?" Lebih spesifik lagi, "Bagaimana membuat aplikasi di dalam handphone yang dapat mengelola voucher elektronik?"

Tujuan

Berdasarkan rumusan masalah di atas, tujuan proyek ini adalah **membuat aplikasi di dalam handphone yang dapat mengelola voucher belanja elektronik.**
Produk akhir dari proyek ini adalah *Software* aplikasi *voucher* elektronik.

Manfaat Proyek

Dengan proyek ini, perusahaan memiliki sebuah aplikasi *Software voucher* elektronik yang dapat ditawarkan kepada operator telepon seluler sebagai layanan nilai tambah. Dengan produk yang sudah teruji, perusahaan akan memiliki keunggulan kompetitif dibandingkan dengan perusahaan lain yang menawarkan solusi sejenis.

Tinjauan Pustaka

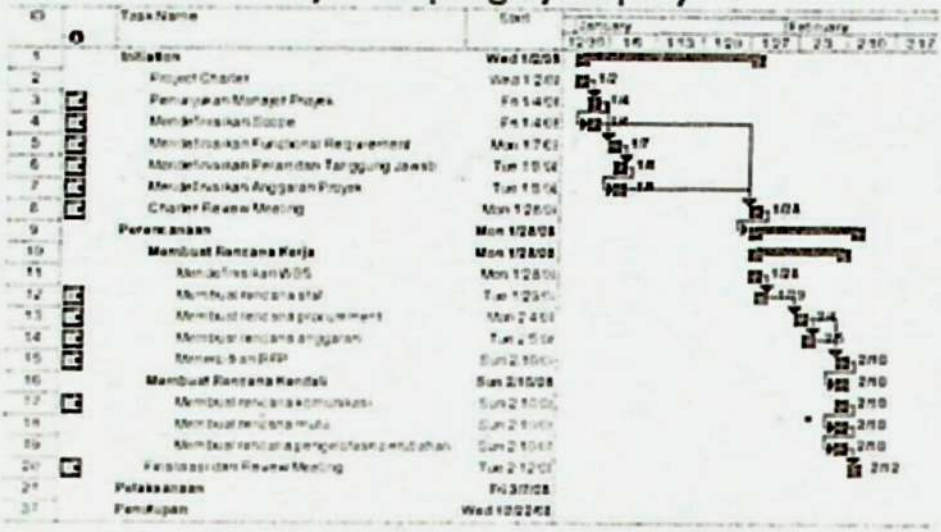
.....<blah><blah><blah>

Metode Pengerjaan Proyek

Proyek akan dilaksanakan dengan mengikuti siklus hidup pengembangan perangkat lunak yang terdiri dari:
Tahap I: pendefinisian kebutuhan
Di dalam tahap ini, tim akan mengerjakan .....
Tahap II: desain
Di dalam tahap ini, tim akan mengerjakan .....
Tahap III: koding dan unit testing
Di dalam tahap ini, tim akan mengerjakan .....
Tahap IV: pengujian
Di dalam tahap ini, tim akan mengerjakan .....
Tahap V: implementasi
Di dalam tahap ini, tim akan mengerjakan .....

Jadwal Pengerjaan Proyek

Berikut ini adalah jadwal pengerjaan proyek.



Daftar Pustaka

http://www.google.com
http://www.wikipedia.com
_____,"Panduan Proyek", Politeknik Telkom, 2009

3 ANALISIS SISTEM

Overview

'Jangan coba-coba untuk memperbaiki sesuatu kecuali kau memahaminya.' Pernyataan tersebut mendeskripsikan fase analisis sistem. Selalu ada sistem saat ini atau yang sudah ada (*current system*), baik yang manual atau berbasis komputer untuk dianalisis. Fase analisis sistem memberikan pemahaman tentang sistem yang sudah ada, dan memberikan pengetahuan kepada analis sistem mengenai peluang untuk terus dikembangkan menjadi sistem informasi yang memenuhi kebutuhan bisnis. Karena itu fase ini menjadi acuan dalam proyek pengembangan sistem informasi. Fase analisis sistem dikenal pula dengan sebutan fase studi, fase studi sistem saat ini, fase penyelidikan terinci, atau fase analisis kelayakan.

Tujuan

1. Mahasiswa memahami apa yang dimaksud fase analisis sistem.
2. Mahasiswa memahami kegiatan yang ada di fase analisis sistem.

3.1 Pengertian Analisis Sistem

Pemahaman terhadap sistem saat ini menjawab pertanyaan-pertanyaan sebagai berikut :

1. Adakah masalah pada sistem yang lama?
2. Apakah masalah yang ada harus dipecahkan?
3. Apakah sistem yang baru layak untuk dibangun?

Analisis sistem adalah bagian dari daur hidup pengembangan sistem (*System Development life cycle* / SDLC). Tujuan dari analisis sistem adalah mempelajari dan memahami sistem saat ini untuk kemudian menganalisis masalah yang ada, melihat peluang untuk mengembangkannya dan batasan-batasannya.

Umumnya, pemilik, pengguna, desainer dan pembangun sistem sering memiliki perpektif berbeda pada sistem informasi. Beberapa tertarik pada hal-hal general, sedang yang lain fokus pada hal-hal yang rinci. Beberapa pada nonteknis, sedang yang lain sangat teknis. Ini menunjukkan jurang komunikasi yang selalu ada pada mereka sehingga membutuhkan seseorang untuk membuat solusi bisnis berbasis komputer sekaligus memahami teknologi informasi. Jembatan untuk jurang komunikasi tersebut adalah analis sistem (*systems analyst*). Kata lain untuk analis sistem adalah konsultan sistem, analis bisnis (*business analyst*), analis proses bisnis, arsitek sistem, insinyur sistem, insinyur informasi (*information engineer*), analis informasi (*information analyst*) dan integrator sistem.

Tim proyek akan dipimpin oleh manajer proyek tapi difasilitasi oleh analis sistem. Studi yang dilakukan terhadap sistem saat ini hendaknya meliputi semua pihak termasuk manajemen dan para pengguna sistem.

Setelah memahami sistem saat ini, masalah yang ada dan peluang, maka tim proyek menentukan tujuan pengembangan sistem informasi (tujuan dari proyek tersebut).

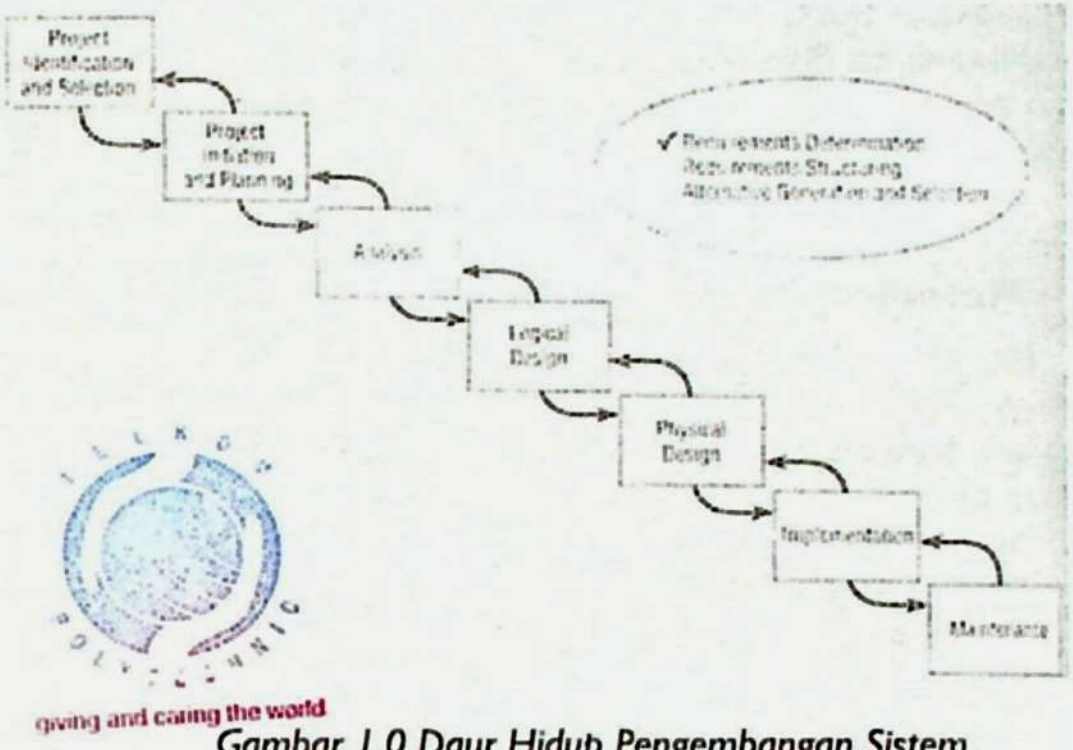
Fase analisis sistem terdiri dari tiga subfase (Hoffer George Valacich, 1999:240), yaitu :

1. Menentukan kebutuhan sistem (*determine System requirement*)
2. Menyusun kebutuhan sistem (*structuring System requirement*)
3. Membuat alternatif strategi desain sistem dan memilih yang terbaik.

3.2 Menentukan Kebutuhan Sistem

Setelah selesai mengidentifikasi dan memilih proyek, (ini terjadi saat fase *Project identification and identification* selesai) maka tim proyek melanjutkan

ke fase inisiasi dan perencanaan proyek. Setelah itu melanjutkan ke fase analisis. Lihat Gambar 1.0 Daur Hidup Pengembangan Sistem.



Gambar 1.0 Daur Hidup Pengembangan Sistem
Menentukan Kebutuhan Sistem

Di subfase menentukan kebutuhan sistem, manajer proyek dan tim analis sistem mengumpulkan informasi dari pengguna di dalam sistem, dari pengguna yang hanya mengamati, serta dari laporan, formulir dan prosedur. Saat mengumpulkan informasi tim proyek melakukan semacam investigasi. Tujuan dari mengumpulkan informasi adalah memahami komponen di dalam organisasi, seperti :

1. Tujuan bisnis yang menentukan apa dan bagaimana pekerjaan diselesaikan.
2. Informasi yang dibutuhkan semua pihak (manajemen, *user*) untuk mengerjakan pekerjaan masing-masing.
3. Data yang dimiliki organisasi.
4. Kapan bagaimana dan oleh siapa atau apa yang memindahkan, mengubah dan menyimpan data.
5. Urutan dan ketergantungan antar berbagai kegiatan menangani (mengolah) data.
6. Kebijakan dan panduan yang menjelaskan karakteristik bisnis dan pasar, serta lingkungan dimana bisnis dijalankan.

Pada saat mengumpulkan informasi, kita mungkin menyadari bahwa informasi yang akan dikumpulkan bisa saja sangat banyak, waktu yang dibutuhkan bisa saja sangat lama, terlalu banyak orang yang terlibat menyebabkan kerja tim tidak produktif.

Karena itu diperlukan metode khusus untuk mengumpulkan informasi. Kita mengenal metode tradisional dan modern untuk mengumpulkan informasi.

3.3 Metode Tradisional

Inti dari fase analisis sistem adalah mengumpulkan informasi. Kita harus mengumpulkan informasi tentang sistem informasi yang saat ini sedang digunakan, bagaimana keinginan *user* untuk memperbaiki sistem saat ini dan kegiatan operasional dengan sistem informasi yang benar-benar baru atau modifikasi dari sistem sebelumnya.

Metode tradisional untuk mengumpulkan kebutuhan sistem adalah :

1. Wawancara
2. Angket
3. Observasi
4. Analisis Prosedur dan Dokumen lain

3.3.1 Wawancara

Panduan saat mewawancara adalah :

1. Rencanakan Wawancara
   a. Siapkan *interviewee* : buat janji untuk bertemu, prioritas pertanyaan
   b. Siapkan agenda, pertanyaan dan *checklist*
2. Dengar dan catat hal-hal yang perlu (rekam pembicaraan bila dimungkinkan)
3. Tinjau catatan dalam 48 jam setelah wawancara
4. Bersikap netral
5. Mencari pandangan lain.

Di bawah ini adalah contoh bagan wawancara :

Interview Outline

Interviewee: *Name of person being interviewed*	Interviewer: *Name of person leading interview*
Location/Medium: *Office, conference room, or phone number*	Appointment Date Start Time End Time
Objectives: *What data to collect* *On what to gain agreement* *What areas to explore*	Reminders: *Background/experience of interviewee* *Known opinions of interviewee*
Agenda: Introduction Background on Project Overview of Interview Topics To Be Covered Permission to Tape Record Topic 1 Questions Topic 2 Questions ... Summary of Major Points Questions from Interviewee Closing	Approximate Time 1 minute 2 minutes 1 minute 5 minutes 7 minutes ... 2 minutes 3 minutes 1 minute
General Observations: Interviewee seemed busy — probably need to call in a few days for follow-up questions since he gave only short answers. PC was turned off — probably not a regular PC user.	
Unresolved Issues, Topics not Covered: He seem to look up sales figures from 1993. He raised the issue of how to handle returned goods, but we did not have time to discuss.	

Gambar 2.0 Contoh Bagan Wawancara

3.3.2 Angket

Angket umumnya berisi kumpulan *close-end* pertanyaan, artinya pertanyaan-pertanyaan di dalamnya dan pilihan jawabannya cukup singkat. Seorang analis sistem yang cukup pengalaman dan sering berlatih akan dapat membuat pertanyaan-pertanyaan yang baik. Karena pertanyaan-pertanyaan di angket adalah tertulis, maka harus benar-benar jelas dalam arti harfiah dan terurut secara lojik.

Angket lebih murah dibandingkan wawancara, karena angket tidak membutuhkan petugas pengawas secara langsung saat beberapa orang mengisi angket pada waktu yang sama.

3.3.3 Observasi

Manusia tidak selalu menjadi informan yang dapat diandalkan, meskipun manusia berusaha berkata jujur. Karena itu kita dapat menggali informasi dengan cara memperhatikan secara langsung apa yang mereka kerjakan, bukan dengan cara mendengarkan apa yang mereka katakan.

Sebagai contoh, manajer A menceritakan bagaimana dia menghabiskan waktunya dalam bekerja, bahwa dia mengerjakan setiap tugas sesuai rencana, bekerja sangat keras dan konsisten untuk memecahkan masalah, serta dapat mengendalikan setiap langkahnya dalam bekerja. Namun setelah diamati langsung oleh seorang analis sistem, ditemukan bahwa manajer A tidak bekerja dengan fokus, memecahkan masalah untuk jangka pendek saja, dan terlalu banyak 'gangguan' seperti seringnya telepon masuk, atau kunjungan dari bawahan atau manajer lain. Ini menunjukkan informasi yang didengar tidak sesuai dengan yang dilihat di lapangan.

3.3.4 Analisis Prosedur dan Dokumen Lain

Metode lain yang dapat digunakan untuk menentukan kebutuhan sistem adalah dengan cara mempelajari dokumentasi sistem dan organisasi untuk menemukan detil rincian mengenai sistem saat ini dan organisasi yang didukung sistem tersebut.

Informasi yang didapatkan dari menganalisis dokumen tentang kebutuhan sistem baru adalah :

1. Alasan mengapa sistem saat ini dibangun
2. Masalah yang ada pada sistem saat ini
3. Peluang untuk memenuhi kebutuhan baru
4. Arahan organisasi yang mempengaruhi kebutuhan sistem informasi, misalnya hubungan dengan pelanggan dan distributor
5. Data dan aturan pengolahan data.

GUIDE FOR PREPARATION OF INVENTION DISCLOSURE
(See FACULTY and STAFF MANUALS for detailed Patent Policy and routing procedures.)

(1) DISCLOSE ONLY ONE INVENTION PER FORM.

(2) PREPARE COMPLETE DISCLOSURE.

The disclosure of your invention is adequate for patent purposes ONLY if it enables a person skilled in the art to understand the invention.

(3) CONSIDER THE FOLLOWING IN PREPARING A COMPLETE DISCLOSURE:

(a) All essential elements of the invention, their relationship to one another, and their mode of operation.

(b) Equivalents that can be substituted for any elements.

(c) List of features believed to be new.

(d) Advantages this invention has over the prior art.

(e) Whether the invention has been built and/or tested.

(4) PROVIDE APPROPRIATE ADDITIONAL MATERIAL.

Drawings and descriptive material should be provided as needed to clarify the disclosure. Each page of this material must be signed and dated by each inventor and properly witnessed. A copy of any current and/or planned publication relating to the invention should be included.

(5) INDICATE PRIOR KNOWLEDGE AND INFORMATION.

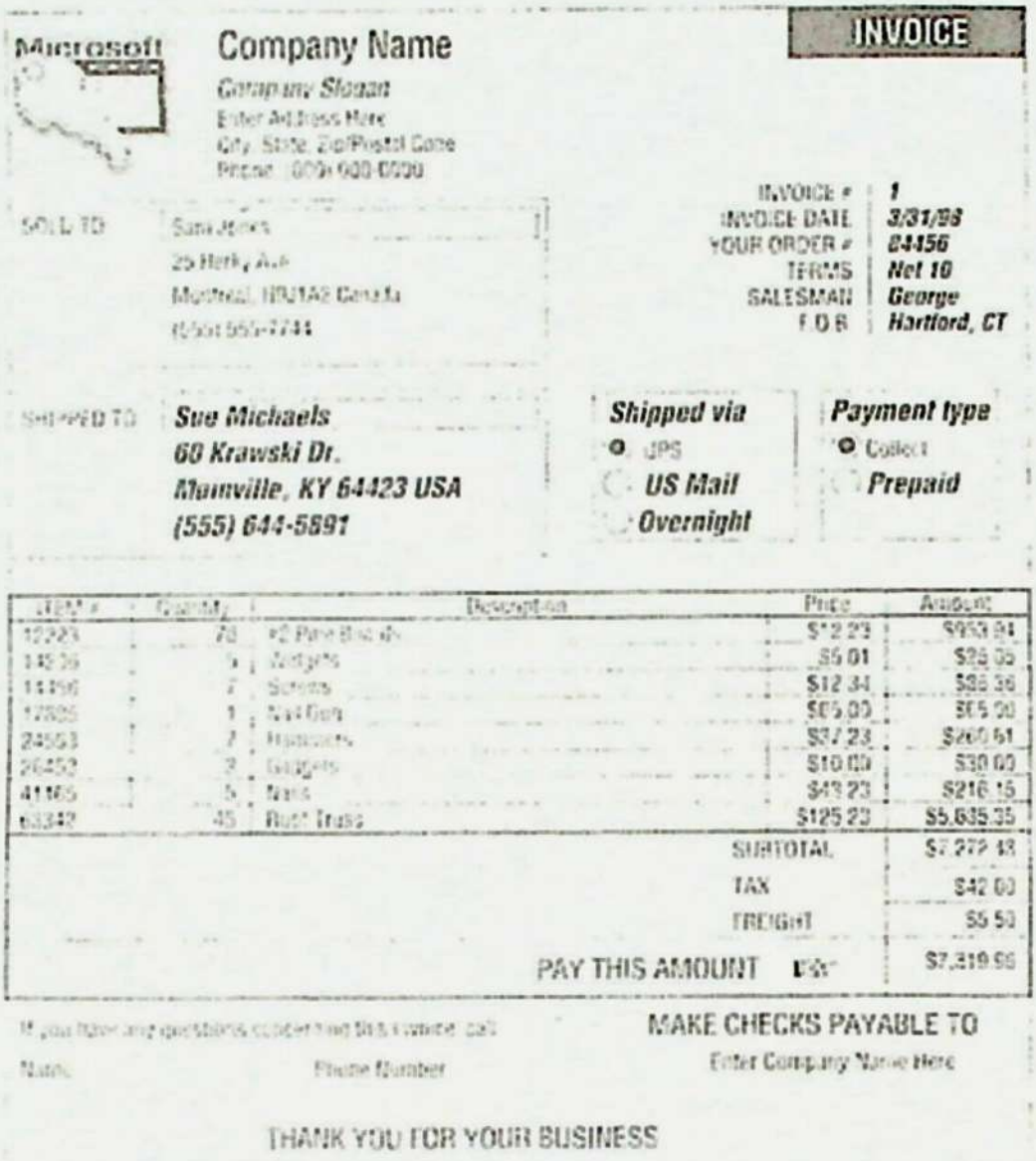
Pertinent publications, patents or previous devices, and related research or engineering activities should be identified.

(6) HAVE DISCLOSURE WITNESSED.

Persons other than co-inventors should serve as witnesses and should sign each sheet of the disclosure only after reading and understanding the disclosure.

(7) FORWARD ORIGINAL PLUS ONE COPY (two copies if supported by grant/contract) TO VICE PRESIDENT FOR RESEARCH VIA DEPARTMENT HEAD AND DEAN.

Gambar 1.0 Contoh sebuah prosedur



Microsoft

Company Name

Company Slogan

Enter Address Here

City, State, Zip/Postal Code

Phone: (000) 000-0000

INVOICE

INVOICE #	1
INVOICE DATE	3/31/98
YOUR ORDER #	84456
TERMS	Net 10
SALESMAN	George
F.O.B.	Hartford, CT

SOLD TO

Sam Jones

29 Herk, A.A

Montreal, H0J1A2 Canada

(514) 555-7744

SHIPPED TO

Sue Michaels
60 Krawski Dr.
Manville, KY 64423 USA
(555) 644-5891

Shipped via
- (•) UPS
- ( ) US Mail
- ( ) Overnight

Payment type
- (•) Collect
- ( ) Prepaid

Item #	Quantity	Description	Price	Amount
12223	78	#2 Pencil Sharpener	$12.23	$953.94
14536	4	Widgets	$5.01	$20.04
14556	7	Screws	$12.34	$86.38
12555	1	Nail Gun	$65.00	$65.00
24563	7	Hammers	$37.23	$260.61
26452	3	Gadgets	$10.00	$30.00
41165	5	Nails	$43.23	$216.15
63342	45	Rust Trays	$125.23	$5,635.35
			SUBTOTAL	$7,277.48
			TAX	$42.00
			FREIGHT	$5.50
			PAY THIS AMOUNT	$7,319.98

If you have any questions concerning this invoice, call

Name Phone Number

MAKE CHECKS PAYABLE TO

Enter Company Name Here

THANK YOU FOR YOUR BUSINESS

Gambar 4.0 Contoh sebuah faktur (invoice)

3.4 Metode Modern

Metode modern yang digunakan untuk mengumpulkan informasi penting telah banyak digunakan oleh analis-analis sistem. Metode tersebut menggunakan teknik yang disebut :

1. *Joint Application Design* (JAD)
2. Sistem Dukungan Kelompok (*Group System Support* / GSS)
3. CASE (*Computer-aided Software engineering*) tools
4. Rekayasa Ulang Proses Bisnis (*Business Process Reengineering* / BPR).

Teknik – teknik modern tersebut dapat mendukung pengumpulan informasi secara efektif dan terstruktur, sambil juga mengurangi penggunaan waktu untuk fase analisis.

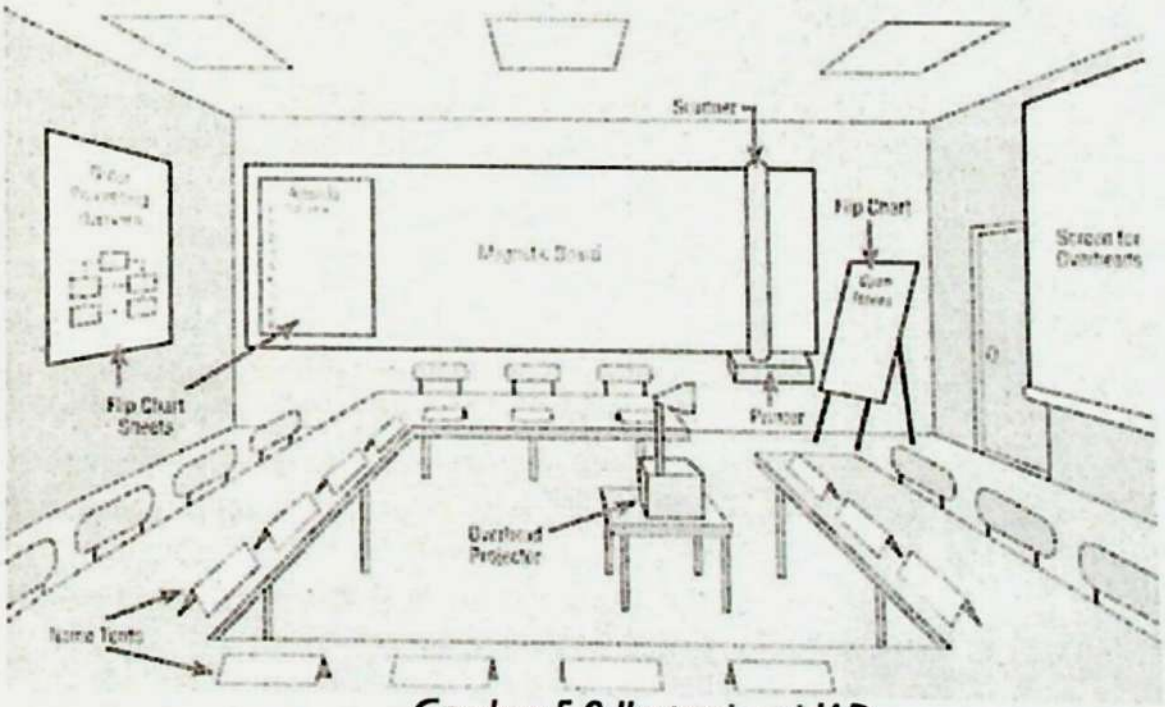
3.4.1 *Joint Application Design* (JAD)

JAD mengumpulkan *user* kunci, manajer dan analis sistem yang terlibat dalam analisis sistem saat ini. JAD mirip dengan wawancara berkelompok hanya menggunakan aturan dan agenda yang berbeda dengan wawancara pada umumnya.

Tujuan JAD adalah mengumpulkan informasi kebutuhan sistem secara simultan dari orang-orang di posisi penting yang terlibat di dalam sistem. Hasilnya adalah pada waktu yang sama analis sistem dapat melihat hal-hal apa yang menjadi kesepakatan di antara orang-orang penting tersebut, dan hal-hal apa yang menjadi konflik.

Sesi JAD diadakan di ruangan khusus dimana partisipan duduk mengeliling meja berbentuk sepatu kuda. Ruangan terdiri dari *whiteboard, overhead Projector, flip chart sheet, scanner, printer* dan *computer-generated display. Flip chart paper* digunakan untuk mencatat *track of issue* yang tidak dapat dipecahkan selama sesi JAD atau topik-topik yang membutuhkan informasi tambahan yang dapat dikumpulkan di luar sesi JAD.

Hasil dari JAD adalah dokumen tentang detil pekerjaan dari sistem saat ini dihubungkan dengan studi dengan sistem penggantinya.



Gambar 5.0 Ilustrasi sesi JAD

3.4.2 Sistem Dukungan Kelompok

Sistem Dukungan Kelompok (*Group System Support* / GSS) didesain khusus untuk mengurangi beberapa masalah dengan pertemuan secara berkelompok. Agar setiap orang di pertemuan / rapat memiliki kontribusi yang sama, maka setiap anggota kelompok mengetikkan komentar ke komputer dan bukan berbicara langsung. GSS diatur sehingga semua anggota grup dapat melihat apa yang diketik oleh anggota lain.

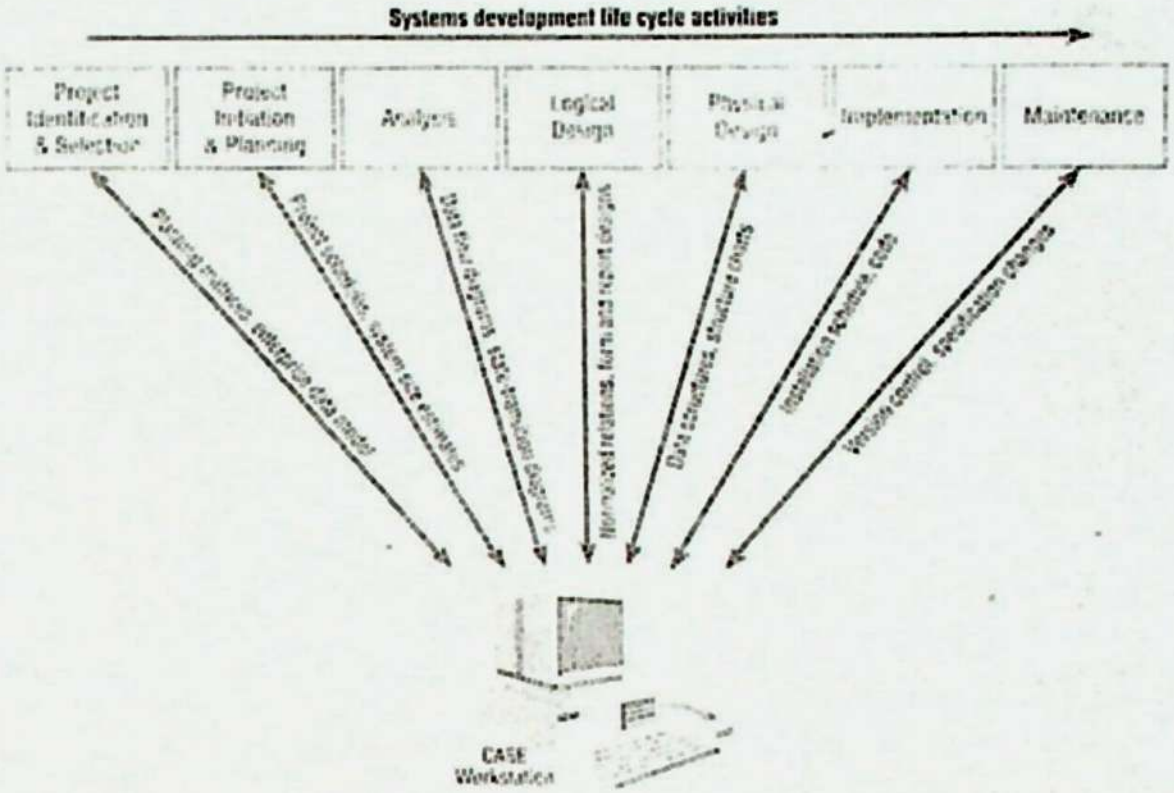
Apabila semua orang mengetik, bukan berbicara, maka semua orang memiliki kesempatan berkontribusi yang sama, dan kesempatan mendominasi pertemuan oleh satu individu dapat dihindari. Komentar yang diketikkan ke GSS adalah anonim, ini menghindari kritik yang ditujukan ke individu.

Teknik GSS dan JAD dapat saja digabung, agar menambah kekuatan dalam mengumpulkan informasi.

3.4.3 CASE tools

CASE (*computer-aided Software engineering*) *tool* adalah perangkat lunak untuk membangun sistem informasi secara otomatis. Ini mempercepat proses menganalisis kebutuhan sistem, dan orang-orang yang memiliki kemampuan menggunakan *tool* ini sangat dibutuhkan tim proyek pengembangan sistem informasi.

Di bawah ini gambaran CASE *tool* dalam mendukung hampir semua fase dalam daur hidup pengembangan sistem :



Gambar 6.0 CASE *tool* mendukung pengembangan sistem secara efektif

3.4.4 Rekayasa Ulang Proses Bisnis

Perhatikan analogi berikut. Dimisalkan Andi adalah seorang pemain golf terkenal di Indonesia. Dia belajar bagaimana mengendalikan bola di cuaca berangin kencang, menggiring bola di area rumput hijau yang luas, memukul bola di lapangan hijau bergelombang, dan dapat mencapai target tanpa dibantu *trainer*. Suatu hari Andi mengikuti pertandingan golf di USA, saat itu dia menyadari lingkungan persaingan di sana tidak cocok dengan gaya bermainnya selama ini. Andi perlu mengubah ulang semua taktik permainannya, belajar kembali bagaimana mencapat target, memutar dan menghentikan bola di lapangan hijau yang penuh 'gangguan' dari orang-orang asing dan wartawan. Apabila Andi dapat mengubah ulang *(reengineering)*, maka dia bisa bertahan, tanpa *reengineering* Andi takkan pernah menjadi pemenang.

Rekayasa Ulang Proses Bisnis (*business Process reengineering* / BPR) adalah mencari dan mengimplementasikan perubahan radikal dalam proses bisnis utuk mencapai terobosan baru pada produk dan jasa.

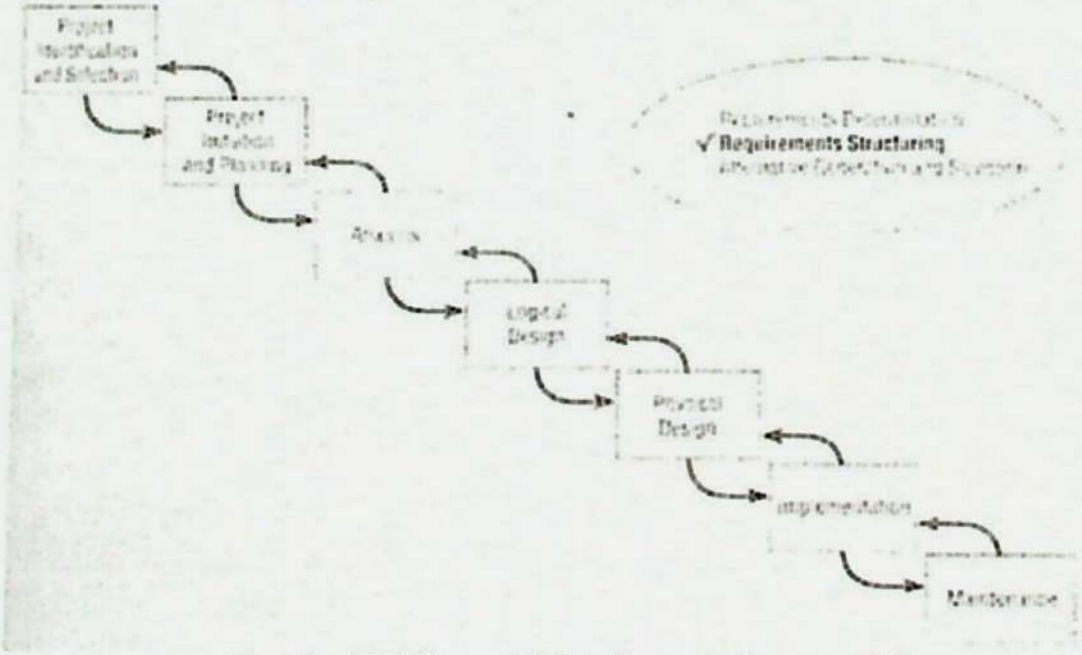
Langkah-langkah BPR :

1. Identifikasikan proses-proses yang akan di-rekayasa ulang
   Pahami proses yang menjadi kunci proses bisnis. Kunci proses bisnis adalah sekumpulan kegiatan terstruktur yang didesain unuk menghasilkan *output* tertentu bagi pelanggan atau pasar tertentu. Pahami kunci proses bisnis, kemudian ubahlah urutan dan struktur kegiatan-kegiatan tersebut untuk memperbaiki kepuasan pelanggan (pelayanan berkualitas dan cepat).
2. Tentukan teknologi
   Sekali kunci proses bisnis dan kegiatannya telah diidentifikasi, maka teknologi informasi harus diterapkan ke proses bisnis tersebut. Sebagai contoh, perusahaan Saturn menggunakan basis data penjadualan produksi dan EDI (*electronic data interchange*) yang memungkinkan para *supplier* dan perusahaan Saturn seperti 'satu' organisasi.

3.5 Menyusun Kebutuhan Sistem

Setelah menentukan kebutuhan sistem, maka kegiatan selanjutnya adalah menyusun kebutuhan sistem dengan cara **membuat model kebutuhan sistem**. Tujuan pemodelan adalah untuk menjelaskan tentang :

1. Bagaimana data mengalir
2. Proses perubahan data menjadi informasi
3. Relasi antar data (tabel)
4. Bagaimana data disimpan.



Gambar7.0 Daur Hidup Pengembangan Sistem
Memodelkan Kebutuhan Sistem

Kakas (*tools*) yang dapat digunakan untuk memodelkan kebutuhan sistem adalah :

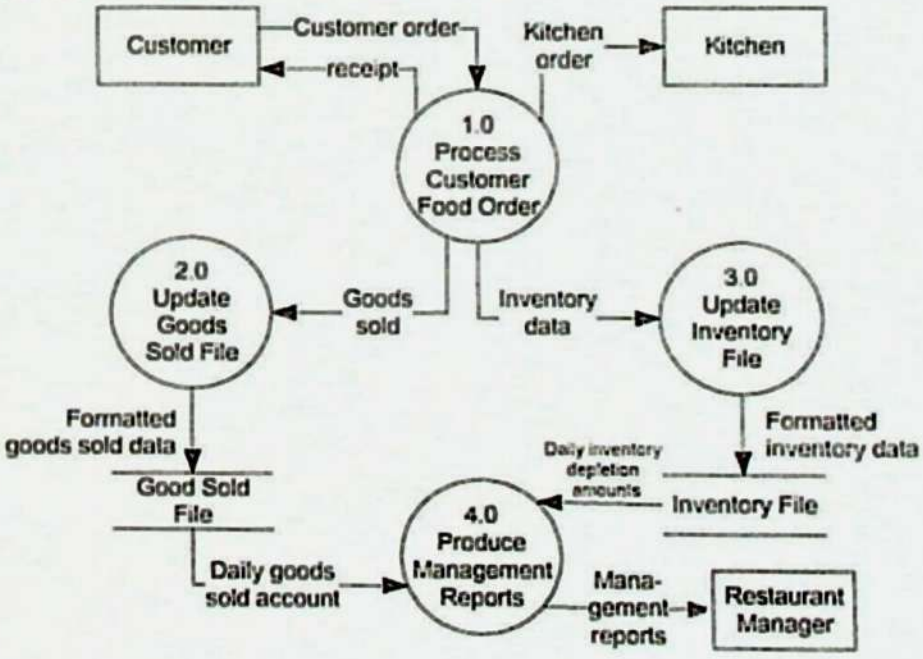
1. Diagram Alir Data (*Data Flow Diagram*), memodelkan aliran data dan proses di dengan metode terstruktur. Diagram ini memodelkan sistem berorientasi pada fungsi atau proses.
2. Diagram Use-Case, memodelkan kebutuhan sistem dengan metode *Object-oriented*. Diagram ini fokus pada objek atau konsep. *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.
3. Diagram *Entity-Relationship*, memodelkan konsep data dan relasi antar data.

3.5.1 Model Kebutuhan Sistem dengan Diagram Alir Data

Berikut ini adalah contoh diagram alir data yang menunjukkan kebutuhan sistem di Hoosier Burger.

Hoosier Burger adalah sebuah restoran cepat saji yang menyediakan berbagai macam makanan khususnya burger. Hoosier Burger terkenal dengan burger dagingnya yang paling enak di sebuah kota di salah satu negara bagian di Amerika Serikat. Penduduk di kota tersebut sering membeli burger di restoran Hoosier Burger.

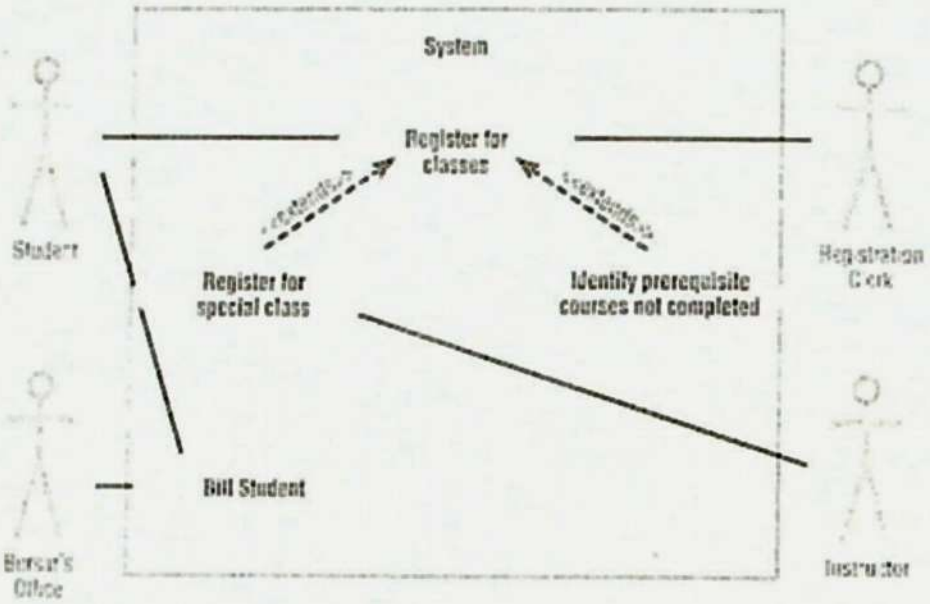
Setelah dianalisis, maka model kebutuhan Sistem Informasi Penyajian adalah seperti di bawah ini :



Gambar 8.0 Contoh Diagram Alir Data Sistem Informasi Penyajian

3.5.2 Model Kebutuhan Sistem dengan Diagram Use-Case

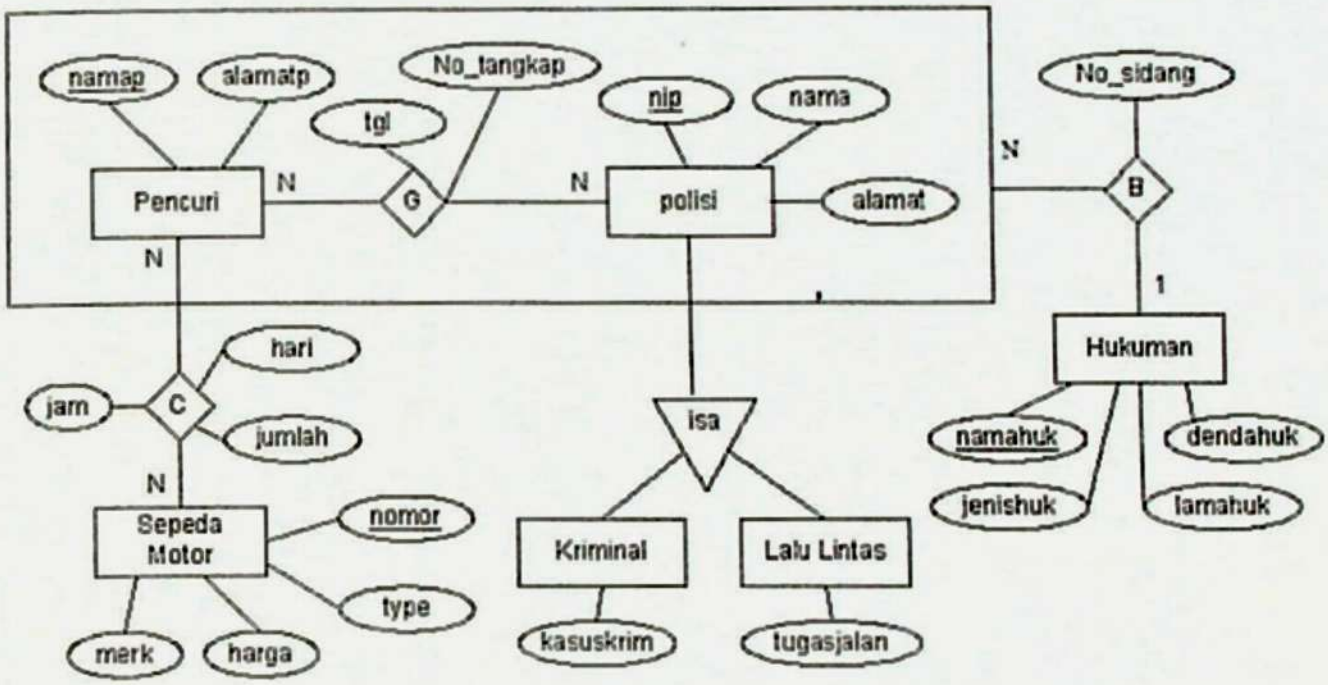
Berikut ini adalah Sistem Pendaftaran Mahasiswa Baru di sebuah universitas. Aktor adalah orang, proses atau sistem lain yang berinteraksi dengan sistem informasi. Aktor pada sistem ini mahasiswa (*student*), kantor bendahara (*bursar's office*), petugas pendaftaran (*registration clerk*) dan instruktur (*instructor*). Setelah dianalisis, maka model kebutuhan sistem informasi tersebut adalah seperti di bawah ini :



Gambar 9.0 Contoh Diagram Use-Case
Sistem Informasi Pendaftaran Mahasiswa Baru

3.5.3 Model Kebutuhan Sistem dengan Diagram *Entity-Relationship*

Setiap perkara pencurian sepeda motor ditangani polisi, baik polisi kriminal ataupun polisi lalu lintas. Pencuri yang ditangkap akan dijatuhi hukuman sesuai hukum yang berlaku. Untuk kasus tersebut dibuatkan model konsep data dan relasi antar data dengan diagran *Entity-Relationship* seperti di bawah ini :



Keterangan : C = pencurian, G = penangkapan, B= pemberian hukuman

Gambar 10.0 Contoh Diagram Entity-Relationship
Sistem Penanganan Pencurian Sepeda Motor

3.6 Membuat Alternatif Desain Strategi Sistem

Tahapan ini disebut juga desain strategi, di sini kita akan menyusun strategi untuk menggantikan sistem yang lama.

Proses memilih strategi terbaik adalah : (1) membagi kebutuhan ke dalam beberapa kumpulan kemampuan / spesifikasi, diurutkan dari spesifikasi minimum dimana *user* dapat menerima, sampai ke spesifikasi menengah dan tinggi dimana perusahaan mampu mendanai dan mengembangkannya, (2) menghitung berbagai potensi lingkungan implementasi (seperti *hardware*,

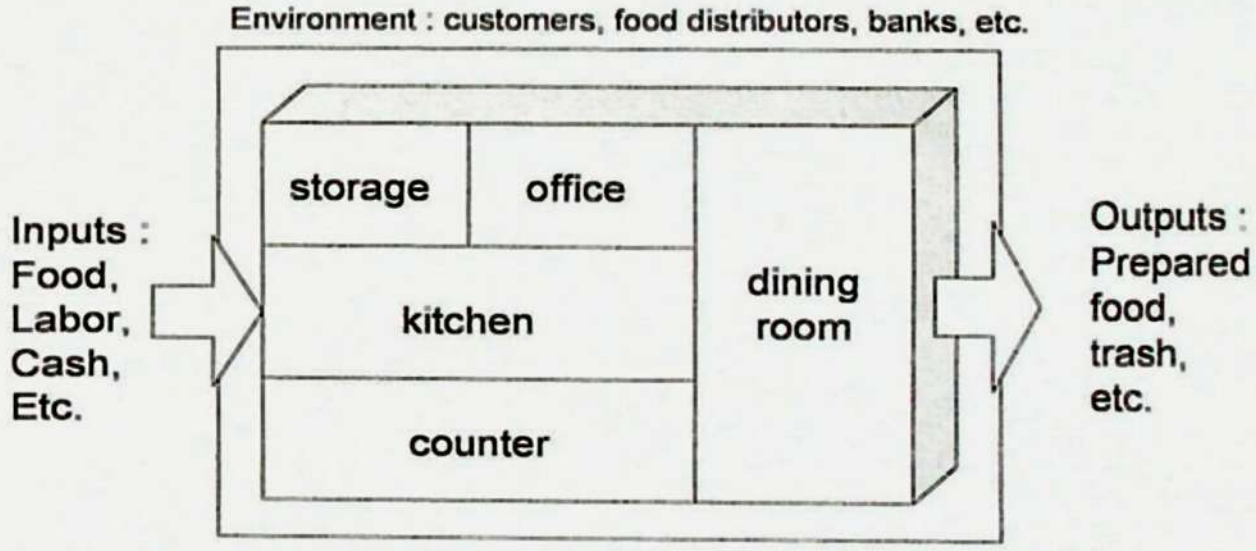
System software, network platform) yang memenuhi syarat dari beberapa kumpulan spesifikasi yang telah disebutkan sebelumnya, (3) menyusun cara untuk memperoleh beberapa kumpulan spesifikasi untuk lingkungan implementasi yang berbeda.

Hal-hal yang harus dipertimbangkan dalam menyusun strategi adalah :

1. *Outsourcing*. Apabila organisasi lain mengembangkan dan/atau menjalankan sebuah aplikasi untuk organisasi kita, maka hal ini disebut *Outsourcing*. Contoh, seorang analis mempertimbangkan bahwa untuk mengimplementasikan sebuah aplikasi bagi perusahaan klien, klien tersebut tidak harus membangun sendiri aplikasinya melainkan dapat dibuat oleh perusahaan lain.
2. Sumber daya *Software*. *Software* yang akan diimplementasikan dapat diperoleh dari pabrikan *hardware*, produser paket *software*, produser custom *software*, solusi *enterprise-wide*, pengembang dari dalam perusahaan (*in-house*) dan *off-the-shelf Software* (*Software* 'siap pakai') dibandingkan *in-house software*.
3. *Hardware* dan perangkat lunak sistem. Memilih strategi perangkat lunak sistem antara lain seperti sistem operasi, DBMS, bahasa pemrograman, dan perangkat lunak jaringan. Hal-hal yang harus dipertimbangkan antara lain adalah biaya pembelian dan instalasi, *user* sudah familiar dengan perangkat lunak sistem yang lama, integrasi dengan aplikasi baru menyebabkan kerancuan atau kemajuan?
4. Implemensi. Implementasi sistem informasi baru sama seperti merubah proses / prosedur kerja di dalam organisasi. Bahkan sistem baru menyebabkan perubahan dalam bekerja, relasi dengan pegawai dan pelanggan, dan membutuhkan keahlian baru.
5. Masalah Organisasi. Masalah pertama di dalam organisasi adalah pengelolaan biaya. Implementasi sistem informasi baru menyebabkan organisasi harus mengeluarkan biaya, demikian pula untuk *training* dan pembelian *hardware / Software* baru. Masalah kedua adalah organisasi dapat mendukung atau tidak dengan alternatif strategi sistem meskipun organisasi memiliki dana yang cukup. Misalnya sistem baru benar-benar berubah secara dramatis dari sistem lama karena akan menyebabkan perubahan proses bisnis organisasi.

3.7 Contoh Analisis Sistem Persediaan Bahan Makanan di Hooseir Burger Restaurant

Hoosier Burger Restaurant adalah rumah makan cepat saji di Bloomington, Indiana. Setiap harinya menjual burger paling enak di kota Bloomington, pelanggannya kebanyakan pelajar dan mahasiswa. Lingkungan Hoosier Burger terdiri dari elemen luar yang berinteraksi dengan sistem seperti pelanggan, bank, dan pesaing lain. Hoosier Burger memiliki *counter* depan tempat pelanggan memesan makanan, dan pintu belakang dimana bahan makanan dari *supplier* dikirim. Gambaran sistem Hoosier Burger seperti di bawah ini :



Gambar 12.0 Sistem di Hoosier Burger Restaurant

Sistem Informasi Persediaan Bahan Makanan di Hoosier Burger dimulai dari pembuatan data Stok Bahan Makanan yang disuplai oleh *supplier*. Kemudian pelanggan yang datang memesan burger akan dicatat pesanannya oleh petugas *Counter*. Catatan *order* diterima oleh petugas *Counter*, lalu disampaikan ke *Kitchen* (dapur). Koki di dapur mencatat bahan makanan yang dia gunakan untuk menyiapkan makanan. Proses ini meng*update* stok bahan makanan di *Storage* (gudang). Setelah selesai, makanan akan disajikan ke pelanggan di ruang makan. Selesai menyantap makanan, pelanggan membayar

makanan ke petugas kasir di *Office*. Petugas kasir akan mencatat makanan apa saja yang terjual. Setiap bulan, akan dibuatkan laporan Persediaan ke Manajer.

3.7.1 Analisis Kebutuhan Sistem

Proses analisis dilakukan dengan cara observasi/pengamatan langsung terhadap apa yang terjadi di rumah makan cepat saji Hoosier Burger. Terdapat kejadian operasional sebagai berikut :

a. Mencatat data Stok Bahan Makanan yang disuplai oleh *Supplier*.
b. Pelanggan memesan makanan.
c. Pesanan pelanggan disampaikan ke *kitchen* (dapur).
d. Makanan diantarkan ke meja pelanggan. Selesai makan, pelanggan akan melakukan pembayaran.
e. Setiap akhir bulan akan dibuatkan laporan Persediaan untuk Manajer.

Maka Kebutuhan Proses Berbasis Komputer berdasarkan kejadian operasional sebelumnya, adalah :

1. Mencatat Stok
2. Mencatat Pesanan / *Order*
3. Meng*update* Stok Bahan Makanan
4. Mencatat Pembayaran
5. Mencetak Kuitansi Pembayaran
6. Meng*update* Makanan yang Terjual
7. Membuat laporan Persediaan

Kebutuhan Dokumen dan Data (file) berdasarkan kejadian operasional sebelumnya, adalah :

1. Formulir Terima Bahan Makanan dari *Supplier*
2. Formulir *Order*
3. File Persediaan (Stok)
4. File Pembayaran
5. Kuitansi Pembayaran
6. File Daftar Barang Terjual
7. Laporan Persediaan

3.7.2 Menyusun Kebutuhan Sistem (Memodelkan Kebutuhan Sistem)

Pemodelan Kebutuhan Sistem tidak digambarkan dengan Tools apapun seperti pada subab 1.5. Melainkan hanya dituliskan saja input proses dan *output* yang ada pada Sistem Informasi Persediaaan.

Model Kebutuhan Sistem Informasi Persediaan di Hoosier Burger adalah sebagai berikut :

1. Proses : *Update* Stok Bahan Makanan
   Input : Formulir Terima Barang dari *Supplier*, File Persediaan
   Output : File Persediaan (ter*update*)
2. Proses : Mencatat Pesanan
   Input : Pesanan dari Pelanggan
   Output : Formulir Pesanan
3. Proses : Meng*update* Stok Bahan Makanan
   Input : Formulir Pesanan
   Output : File Persediaan (ter*update*)
4. Proses : Mencatat Pembayaran, Mencatat Barang Terjual
   Input : Formulir Pesanan
   Output : File Pembayaran, File Daftar Barang Terjual
5. Proses : Cetak Kuitansi
   Input : File Pembayaran
   Output : Kuitansi untuk Pelanggan
6. Proses : Membuat Laporan Persediaan
   Input : File Persediaan, File Daftar Barang Terjual
   Output : Laporan Persediaan untuk Manajer

3.7.3 Membuat Alternatif Strategi Sistem dan Memilih yang Terbaik

Pada tahanpan ini kita akan membahas mengenai batasan proyek pengembangan sistem informasi nya. Contoh batasan proyek pengembangan Sistem Informasi Persediaan adalah :

1. Dana pengembangan sistem tidak lebih dari $100,000.
2. Sistem baru harus sudah dapat dioperasikan dalam waktu 6 bulan sejak proyek dikerjakan.

Di bawah ini adalah contoh alternatif strategi sistem untuk Sistem Informasi Persediaan Hoosier Burger :

Kriteria		Alternatif A	Alternatif B
1.	Otomatisasi *Order*	Ya	Tidak
2.	Real-time *Update* Persediaaan / Stok	Untuk beberapa Item	Untuk semua Item
3.	Pengembangan *Software*	*In-house*	Off-the-self
4.	*Software*	Visual Basic	Borland Delphi
5.	Implementasi	2 hari	7 hari
6.	*Training*	*Training* sederhana	Full *Training*

Proses terakhir dari tahapan ini adalah pemilihan alternatif terbaik. Untuk otomatisasi *order* tidak perlukan karena *order* cukup dicatat di formulir pesanan. Pengembangan *Software* dilakukan *in-house*, karena sistem yang dibangun tidak terlalu kompleks, masih dapat ditangani oleh tim desain dan *programmer* proyek. *Software* yang dipilih adalah Borland Delphi versi 8.0 dengan DBMS Paradox yang sudah *built-in* dengan Borland Delphi. Pengembangan Sistem tidak lebih dari 6 bulan, implementasi 7 hari, dan akan diadakan *training* sederhana selama 3 hari.

4 Desain

Overview

Tahapan desain yang menggambarkan proses perancangan perangkat lunak dimana ide-ide informal yang ditransformasikan ke deskripsi implementasi yang lebih detil nantinya ke suatu pemrograman dengan beberapa strategi perancangan.

Tahapan desain ini mengejawantahkan segala kebutuhan yang dihasilkan pada saat Analisa dalam bentuk pendekatan pemodelan berorientasi objek yang menggantikan desain orientasi fungsional. Desain perangkat lunak ini terdiri dari :

- *System Design*
  - *Design Process*
    - Metode Desain
    - Deskripsi desain
  - *Design Strategies*
    - *Functional Design*
    - *Object-oriented Design*

Tujuan

1. Mahasiswa dapat mengartikan segala kebutuhan *user* pada saat analisa ke dalam perancangan perangkat lunak.
2. Mahasiswa mengetahui strategi desain dengan pendekatan berorientasi objek yang saling berkomplemen dengan berorientasi fungsional.
3. Mahasiswa mengetahui atribut kualitas desain

4.1 *System Design*

Perancangan yang bagus merupakan kunci proses rekayasa yang efektif. Akan tetapi tidak mungkin memformalkan proses desain dalam disiplin ilmu rekayasa. Proses desain merupakan proses kreatif yang meminta pengertian dan keahlian desainer yang sudah berpengalaman dan bertugas mempelajari *System* yang berjalan.

Perihal Desain yang harus ditangani adalah sbb:

1. Pelajari dan mengerti permasalahan
2. Mengidentifikasi solusi-solusi yang dimungkinkan dan mengevaluasinya untuk dijalankan.
3. Menggambarkan tiap deskripsi yang digunakan dalam solusi tersebut sebelum pembuatan dokumentasi formal.

Proses penyelesaian masalah tersebut diatas diulang-ulang selama tiap abstraksi diidentifikasi dalam perancangan awal. Proses perbaikan berlanjut sampai sebuah spesifikasi desain detill dengan berbentuk algoritma dan struktur data yang diimplementasikan pada tiap abstraksi dapat disiapkan.

4.1.1 Proses Desain

Sebuah model umum Desain *Software* adalah sebuah graf berarah. Target proses desain merupakan pembuatan sebuah graf tersebut tanpa ketidakkonsistenan. Node-node dalam Graf tersebut merepresentasikan entitas dlm desain yaitu proses, fungsi atau tipe. Hubungan atau Link antar node merepresentasikan relasi antar entitas desain spt call, menggunakan ... .

Aktifitas proses desain pada sistem *Software* adalah sebagai berikut:

1. *Architectural Design*, subsistem-subsistem membentuk suatu sistem dan hubungan diantaranya diidentifikasi dn didokumentasikan. Dalam hal ini menghasilkan Arsitektur Sistem.
2. *Abstract Specification*, Pada tiap subsistem, Spesifikasi abstraksi proses digambarkan sesuai batasan are sub sistem dan menghasilkan spesifikasi *software*. Aktifitas *Architectural Design* dan *Abstract Specification* ini merupakan komponen *Requirement Specification*.
3. *Interface Design*, Pada setiap subsistem, antarmuka dirancang dan didocumentasikan. Spesifikasi antarmuka harus tidak ambigu seperti subsistem digunakan tana dasar pengetahuan operasi subsistem tersebut. Aktifitas ini menghasilkan produk spesifikasi antarmuka.
4. *Component Design*, Proses layanan dialokasikan ke komponen-komponen yang berbeda dan antarmuka yang menggunakan komponen-komponen ini di rancang. Aktifitas ini menghasilkan produk spesifikasi komponen.

5. *Data Structure Design*, Penggunaan Struktur data dalam implementasi *System* dirancang lebih detil dan dispesifikasikan. Aktifitas ini menghasilkan produk spesifikasi Struktur Data.
6. *Algorithm Design*, Perancangan dengan spesifikasi Algoritma ang lebih detil pada setiap proses layanan dirancang. Aktifitas ini menghasilkan produk spesifikasi Algoritma.

Proses Desain di atas dlakukan berulang-ulang pada tiap subsistem sampai komponen teridentifikasi dapat dipetakan secara langsung ke dalam komponen bahasa pemrograman seperti paket prosedur atau fungsi.

Top-down Design merupakan salah satu cara penyelesaian suatu masalah desain dilakukan. Di dalamnya Masalah dibagi-bagi ke dalam sub masalah sampai submasalah terbaca diidentifikasi. Bentuk formalnya desain ini menggunakan suatu graf yang berhirarki seperti struktur modul. *Top-down Design* ini melakukan dekomposisi abstraksi *System* ke subsistem-subsistem sampai secara gamblang ke pendekatan perancangan komponen.

4.1.2 Metode Desain

Metode perancangan perangkat lunak terdapat beberapa pendekatan yaitu sbb:

- *Structured Design method* (Constantie and Yourdon, 1979)
- *Structured systems Analysis* (Gane and Sarson, 1999)
- *Jackson System Development* (Jackson, 1983)
- *Object-oriented Design*(Robinson,1992; Booch,1994)

Dalam Metode Struktur melibatkan sejumlah aktifitas, notasi, format laporan , aturan dan panduan desain. Metode Struktural ini didukung oleh beberapa model sistem berikut:

1. Model Aliran data, dimana sistem dimodelkan menggunakan transformasi data yang ditempatkan sebagai proses. Hal ini digambarkan dengan *Data Flow Diagram*
2. Model *Entity-Relationship*, digunakan untuk menggambarkan struktur data secara logika digunakan.
3. Model Struktural, dimana komponen sistem dan antarmukanya didokumentasikan. Pendekatan ini ke arah modular dengan structure charts-nya.
4. Model *Object-oriented*, yang melibatkan model *inheritance* pada sistem. Model ini menggambarkan bagaimana objek disusun dan digunakan oleh objek lain.

4.1.3 Deskripsi Desain

Proses Desain *Software* didokumentasikan dalam sejumlah dokumen desain yang menggambarkan desain pada perogrammer dan desainer yang lain. Terdapat 3 tipe notasi yang digunakan untuk mendeskripsikan dalam dokumen yaitu:

1. Notasi Grafis, digunakan untuk mennggambarkan hubungan antara komponen yang membangun desain dan menghubungkan desain ke sistem dunia nyata dengan pemodelan.
2. *Program Descripstion Language* (PDL), Bahasa ini menggunakan kontrol dan stuktur data yang membangun berdasarkan pembangunan bahasa pemrograman tertentu tetapi juga mengijinkan teks keterangan dan pernyataan yang ditambahkan untuk memberikan keterangan detil dari desain yang akan diimplementasikan.
3. Teks informal, sejumlah informasi yang berhubungan dengan desain tidak dapat diekspresikan secara formal. Informasi yang rasional atau yang non fungsional dapat diungkakan dengar teks bahasa narasi asli.

4.2 *Design Strategies*

Terdapat 2 strategi desain yaitu:

1. *Functional Design*
   Sistem dirancang dari sisi pandang fungsional, mulai dari sisi pandang high-level dan kemudian sampai penyempurnaan ke dalam desain yang lebh detil. Terdapat pendekatan desain struktural yang ditekankan di sini.

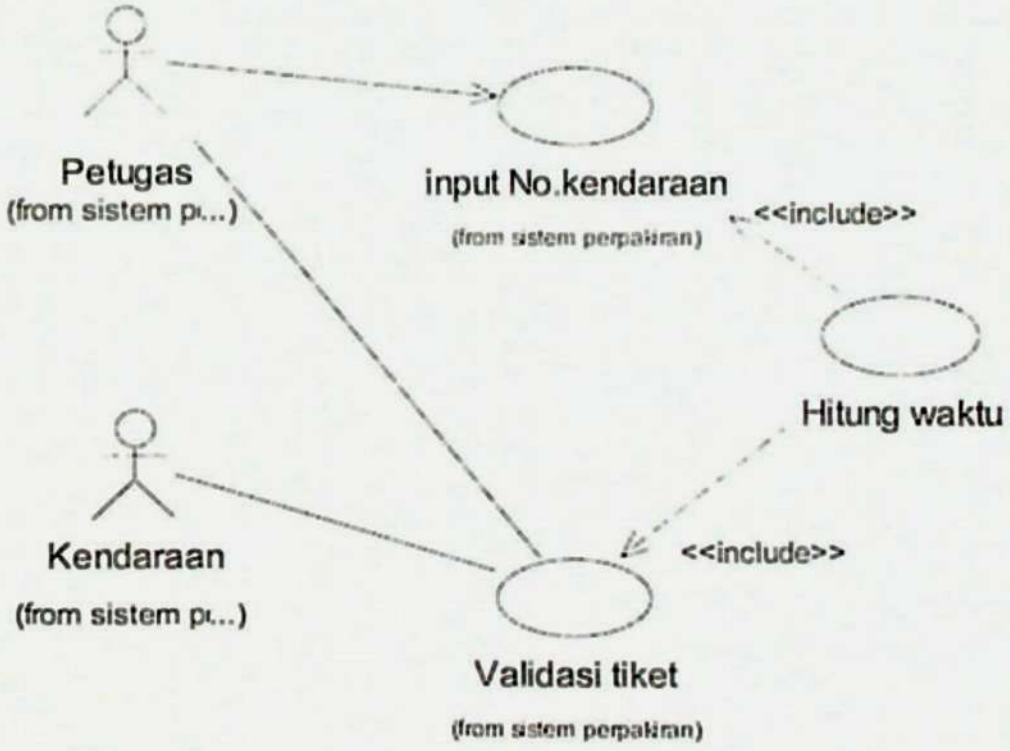
2. *Object-oriented Design*
   Sistem diperlihatkan sebagai kumpulan objekobjek daripada Fungsi-fungsi yang ada. Desain ini disasarkan atas ide informasi yang tersembunyi pada hubungan antar objek yang terjadi pad sistem.

Pada Desain *Object-oriented* (OOD), status sistem dibagi-bagi dan tiap objek mengatur informasi statusnya masing-masing. Objek mempunyai sejumlah atribut yang mendefinisikan status dan operasi yang dilakukan dalam atribut itu. Objek-objek biasanya anggota dari suatu kelas objek yang mendefinisikan atribut dan operasi anggota kelas tersebut. Hal ini memungkinkan diturunkan (*inheritance*) dari satu atau lebih *superclass* sehingga definisi suatu class butuh pembeda antar kelas itu dan *superclass*. Secara konsep, objek

mengkomunikasikan dengan pertukaran pesan. Dalam praktiknya, komunikasi objek diaktifkan dengan suatu objek memanggil sebuah prosedur yang berhubungan dengan objek yang lain. Hal ini bisa dilihat pada Diagram kelas yang terbentuk. (Dalam hal ini belum diperkenalkan).

Contoh Diagram Kelas Sistem Perpustakaan :

Pada OOD, diperlihatkan sesudah hasil analisa dibuat yang berasal dari *Use Case* diagram dan skenario yang dapat ditangkap oleh *System* Analist. (Hal ini sudah diungkap pada tahapan *User* Requirment dan Analisa). contoh *use case* diagram dan skenario sebagai dasar tahapan desain dapat dilihat contoh sebagai berikut:



Gambar Diagram *Use Case* Sistem Perparkiran

Berikut ini adalah langkah-langkah sederhana untuk mendefinisikan kebutuhan menggunakan *Use Case*:

a. Berdasarkan rumusan masalah dan tujuan proyek, tetapkan sebuah sistem yang akan dibuat

Sistem adalah suatu himpunan keadaan yang memetakan serangkaian masukkan ke dalam sistem menjadi serangkaian keluaran dari sistem. Sistem adalah himpunan terbatas yang memiliki batas-batas yang jelas yang memisahkannya dari sistem yang lain. Dalam langkah pertama ini, perjelaslah sistemnya, tentukan batas-batasnya, dan sistem apa saja yang berbatasan dengannya.

b. Menetapkan Aktor

Setelah jelas definisi sistem yang akan dibuat, langkah kedua adalah menetapkan aktor yang terlibat di dalamnya. Aktor bisa berupa orang atau sistem lain di luar sistem yang sedang dibuat. Misalnya, GSM Model adalah sistem di luar sistem *Software* SMS *Gateway*.
Catatan: bisa jadi banyak sekali aktor yang akan berinteraksi terhadap sistem yang dibuat, namun begitu tidak semua aktor perlu dipertimbangkan. Kadangkala ada faktor tertentu yang mengharuskan kita mengabaikannya. Jadikanlah itu sebagai batasan untuk menyederhanakan permasalahan.

c. Menetapkan *Use Case* untuk Setiap Aktor

Untuk setiap aktor, tetapkanlah kegiatan apa yang akan dia lakukan terhadap sistem. Misalnya, untuk aktor Mahasiswa, maka *Use Case*nya adalah 'memesan makanan'.

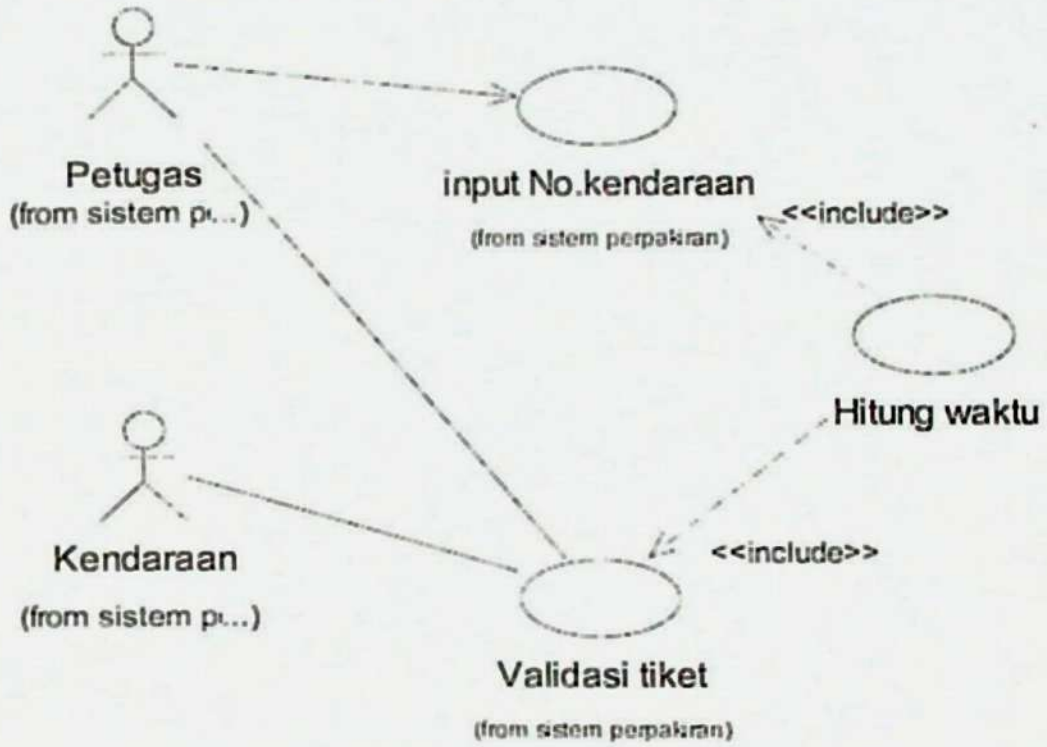
d. Menetapkan skenario untuk setiap *Use Case*

Untuk setiap *Use Case*, buatkan sebuah skenario. Skenario yang dibuat dapat mirip sebuah skenario di dalam film. Skenario tersebut harus menyebutkan prekondisi (kondisi aktor dan sistem sebelum aktor melakukan aksi), postkondisi (kondisi aktor dan sistem setelah aktor melakukan aksi), langkah-langkah kejadian normal, langkah-langkah kejadian alternatif (bila langkah normal tidak terpenuhi), skenario pengecualian, prioritas skenario, aturan-aturan yang harus dipenuhi (misalnya rumus-rumus, formula, algoritma, dll), dan frekuensi kejadian dari *Use Case* ini.

Langkah keempat inilah yang terberat karena kita diminta untuk membayangkan apa yang seharusnya terjadi, apa yang mungkin terjadi, dan apa yang tidak boleh terjadi untuk setiap *Use Case*. Untuk itu, dalam menyusun skenario ini perlu dilibatkan calon pengguna sistem yang memahami skenario-skenario yang mungkin terjadi.

4.3 Contoh Deskripsi *Use Case* dengan skenarionya

Deskripsi *Use Case* untuk Diagram *Use Case* pada



Gambar diatas dapat berbentuk sebagai berikut:

Use Case ID:	I		
Nama *Use Case*:	Input No. Kendaraan		
Created By:			Eddy Prasetyo Nugroho
Date Created:			27/03/2009
Aktor:	Petugas		
Deskripsi:	Petugas menginputkan nomor-nomor kendaraan yang masuk di area parkir ke dalam Sistem informasi perangkat lunak perpakiran. Sehingga mulai menjalankan		

	Hitung Waktu mulai dari kendaraan masuk hingga validasi dilakukan.
Normal Course/ Skenario Normal	1. Petugas melihat/memeriksa nomor kendaraan yang akan masuk ke area parkir 2. Sistem Perangkat Lunak menampilkan antarmuka untuk mendata kendaraan tersebut. 3. Petugas memasukkan nomor kendaraan pada form antarmuka data kendaraan. 4. Sistem akan mulai menghitung waktu parkir setelah petugas menekan tombol OK setelah pendataan untuk kendaraan tersebut. 5. Sistem akan mengeluarkan tiket parkir
Includes:	**Hitung Waktu**
Priority:	**Utama**
Kebutuhan khusus:	-
Catatan:	

Use Case ID:	2		
Nama *Use Case*:	Validasi Tiket		
Created By:			Eddy Prasetyo Nugroho
Date Created:			27/03/2009
Aktor:	Kendaraan		
Deskripsi:	Validasi kendaraan dilakukan pada saat kendaraan akan keluar dari area parkir. Dan dalam hal ini, menghitung waktu akhir dan durasi selama kendaraan parker serta memberikan validasi bahwa nomor kendaraan tersebut sudah tidak parkir.		
Normal Course/ Skenario Normal	1. Petugas melihat/memeriksa tiket dan nomor kendaraan yang akan KELUAR ke area parkir 2. Petugas memilih menu Kendaraan Keluar dalam Sistem Perangkat Lunak dan Sistem menampilkan antarmuka untuk verifikasi kendaraan tersebut. 3. Petugas memasukkan nomor kendaraan pada form		

	antarmuka kendaraan keluar. 4. Sistem memberitahukan bahwa kendaraan telah parkir dengan durasi waktu parkir yang didapat dari Hitung Waktu sebelumnya. 5. Sistem akan memvalidasi bahwa nomor kendaraannya sudah tidak parkir pada area perpakiran
Includes:	**Hitung Waktu**
Priority:	**Utama**
Kebutuhan khusus:	-
Catatan:	

Use Case ID:	3		
Nama *Use Case*:	Hitung Waktu		
Created By:			Eddy Prasetyo Nugroho
Date Created:			27/03/2009
Aktor:	-		
Deskripsi:	Sistem Menghitung waktu pada saat setelah input nomor kendaraan yang didata untuk masuk area parkir dan menghitung durasinya pada saat kendaraan akan keluar area parkir.		
Normal Course/ Skenario Normal	1. Sistem akan mulai menghitung saat setelah nomor kendaraan diinputkan 2. Sistem akan menghitung durasi waktu selama kendaraan parkir, pada saat diminta dalam menu Kendaraan_Keluar.		
Includes:	-		
Priority:	**Utama**		
Kebutuhan khusus:	-		
Catatan:			

Rangkuman

1. Bahwa Tahapan desain merupakan proses pengejawantahkan kebutuhan *user* pada sistem pada tahapan analisa yang akan dipandu untuk tahapan *coding*
2. Pada Desain terdapat modeling untuk mendapatkan deskripsi dokumen yang dikirim dengan 2 strategi yaitu *Functional Design* dan *Object-oriented Design*.

5 Coding

Overview

Untuk membangun program diperlukan isntruksi yang dimengerti oleh komputer. Sekumpulan isntruksi ini dapat berupa bahasa pemrograman dengan berbagai tingkatannya. Melalui bab ini akan dibahas bagaimana membuat kode yang baik untuk bahasa pemrograman khususnya bahasa pemrograman tingkat tinggi.

Tujuan

1. Mahasiswa mengetahui prinsip-prinsip dasar dalam pembangunan kode yang baik.
2. Mahasiswa mampu membuat kode yang memiliki kriteria baik.

5.1 Kebutuhan Kode yang Baik

Berbagai macam aplikasi dibangun dengan sekumpulan instruksi yang bahasa pemrograman. Sekumpulan ini instruksi ini sering disebut sebagai *code* atau *source code*. Semakin besar dan kompleks suatu aplikasi semakin besar pula jumlah *source code* yang dibutuhkan.

Dalam taraf aplikasi sederhana yang terdiri dari beberapa baris, kualitas kode tidaklah terlalu berpengaruh, namun dalam aplikasi yang komplek, kode akan sangat berperan secara signifikan dalam siklus hidup perangkat lunak yang dibangun.

Namun seringkali ditemui adanya anggapan bahwa kode yang baik bukanlah hal yang penting. Ungkapan berikut ini sering dilontarkan untuk orang yang berpandangan demikian:

"Kode yang baik tidak perlu, yang penting jalan!"

"Membuat kode yang baik hanya buang waktu saja"

"Customer tidak perlu melihat sour*code*, mereka hanya melihat aplikasi jalan, dan saya merasa cukup dengan itu"

"Kode yang baik itu untuk tugas perkuliahan saja, tidak perlu diimplementasikan dalam keseharian"

"Kode yang baik hanya mempersulit pembuatan kode itu sendiri"

Dalam kaidah rekayasa perangkat lunak, kode yang baik merupakan keniscayaan karena menentukan kualitas *Software* itu sendiri.

Untuk menilai suatu kode baik atau tidak terdapat beberapa kriteria yang dapat memandu untuk melakukan hal ini antara lain:

1. *Readibility*
2. *Bugfree*
3. *Modular*
4. Sesuai dengan waktu dan *budget*
5. Dapat dikembangkan lebih lanjut
6. Dapat di pelihara perkembangannya
7. Diimplementasikan berdasarkan desain yang jelas

5.1.1 Readable

Readable artinya orang lain dapat membaca dan memahami dengan mudah kode yang dibuat. Kemudahan ini dapat dilihat dari:

1. Tersedia cukup komentar
2. Mengikuti konvensi yang digunakan, berkaitan dengan penamaan variabel
3. Indentasi yang sama untuk kode yang memiliki derajat yang sama.

Readability bermanfaat untuk masa yang akan datang, bukan pada saat pembuatan *code*.

Perhatikan contoh dua kode berikut

```
Function Hitung(x:integer):integer;
Var i,t:integer;
Begin
If x<= 0 then
t:= 0
else
for i:= 0 to x do
t=t+i;
Hitung := t;
End.
```

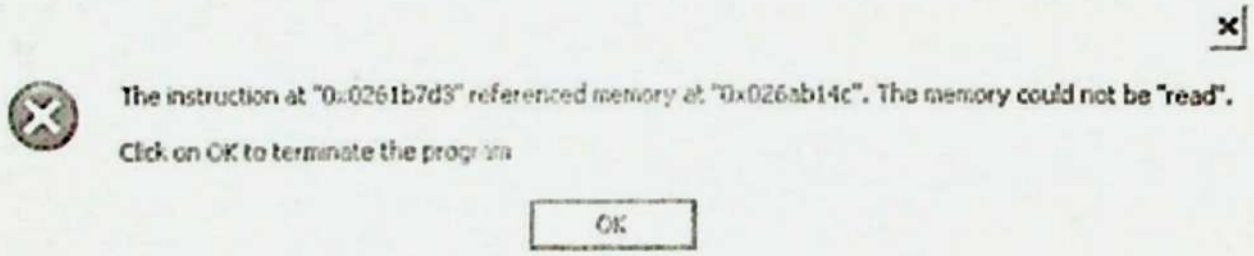
```
Function Hitung(x:integer):integer;
{Fungsi hitung menerima bilangan bulat x dan mengembalikan
nilai 0+1+2+3+...+x apabila x > 0 dan mengembalikan nilai
nol jika x <= 0}

Var
  i       :integer;{variabel iterasi}
  total   :integer;{penyimpan nilai 0+1+2+3...+x}

Begin
  If x<= 0 then
     total:= 0  {jika x<=0 maka total deretnya nol}
  else
     for i:=0 to x do   {jika x>0 dilakukan iterasi}
          total=total+i;{iterasi hitung 0+1+2+3+...x
                         disimpan di variabel total}
  Hitung := t;
End.
```

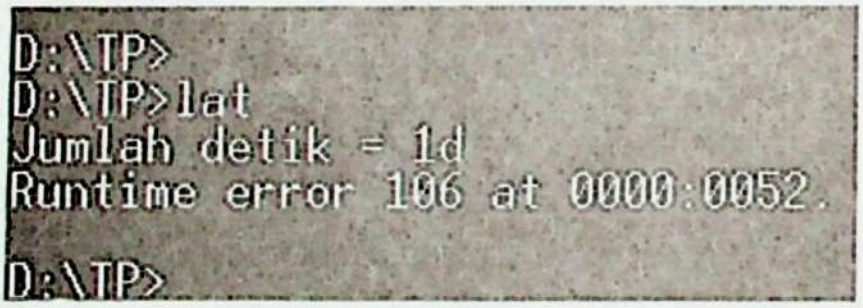
5.1.2 Bug Free

Bug Free artinya aplikasi bebas dari kesalahan (dalam batasan tertentu), dan menangani keterbatasan dan kesalahan proses pada level aplikasi, bukan oleh sistem. Pesan kesalahan tidak boleh dari sistem (yang biasanya sulit dimengerti oleh *user*)

Contoh: Pembagian dengan nol ditangani oleh aplikasi dengan pencegahan, atau dengan menangkap *exception*.

Bukti paling mutlak kebenaran *source code* adalah dapat dijalankannya *source code* tersebut (*Compiled and Runable*) sebelum pengecekan lebih lanjut dan Bug paling mutlak adalah tidak dapat dijalankannya *source code*.





5.1.3 Modular

Modular berarti fungsi, prosedur, (atau sebutan lain yang memiliki karakteristik fungsi/prosedur) dan kumpulan keduanya berkumpul berdasar kategori tertentu.

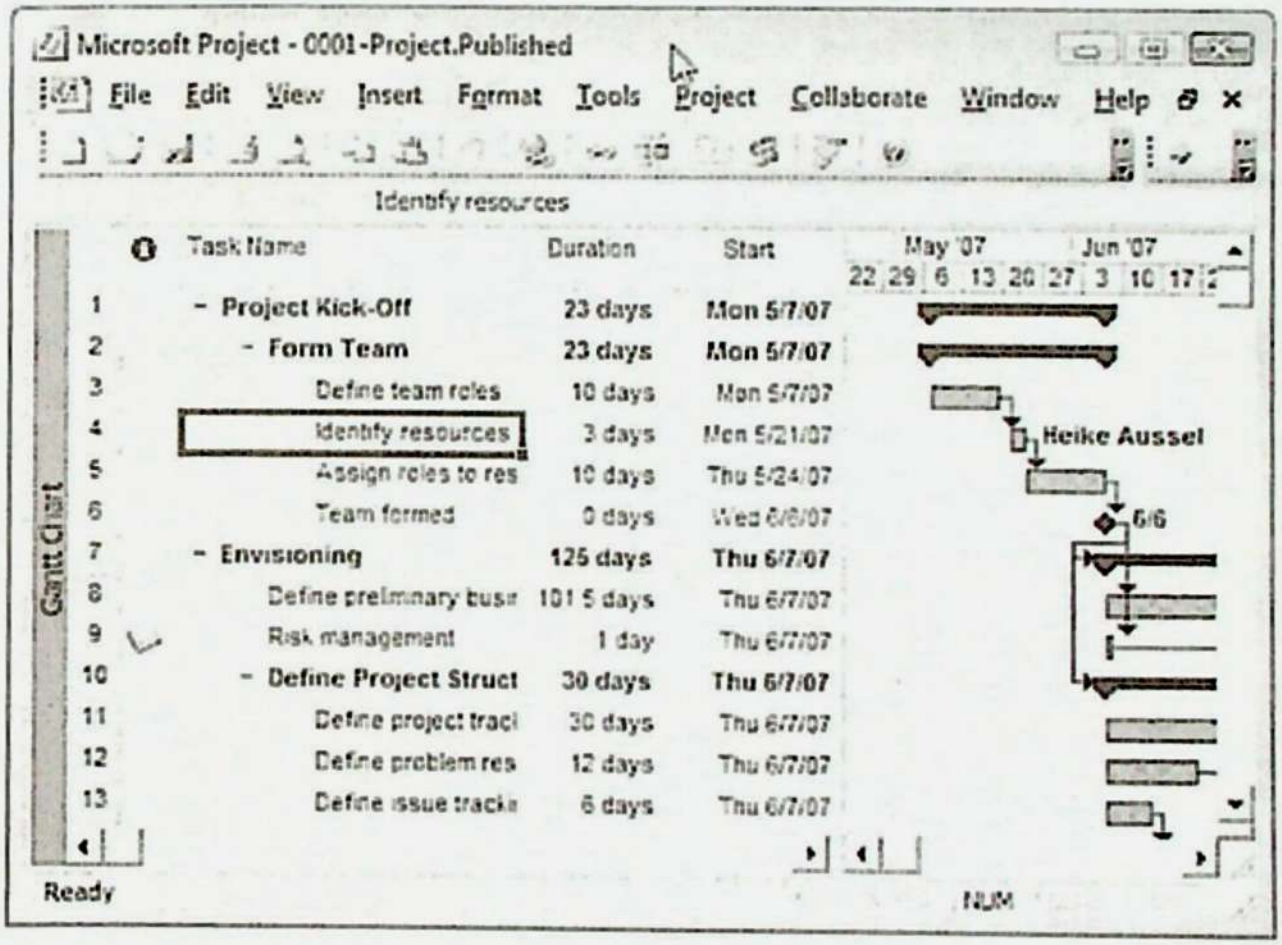
Dengan fungsi dan prosedur, menghindari pengulangan baris yang sama untuk suatu tujuan tertentu. Apabila terjadi kesalahan cukup memperbaiki satu tempat saja.

5.1.4 Delivered On Time And Within Budget

Dimensi paling mendasar pada sebuah proyek adalah batasan waktu dan biaya. Batasan yang lain merupakan turunan dari dua batasan tersebut.

Sebuah proyek perangkat lunak akan menjadi sia-sia apabila tidak sesuai waktu yang disepakati

Sebuah Perangkat lunak yang nyaris sempurna akan menjadi tidak ada nilainya jika budget yang dikeluarkan tidak sebanding dengan nilai yang diperoleh.



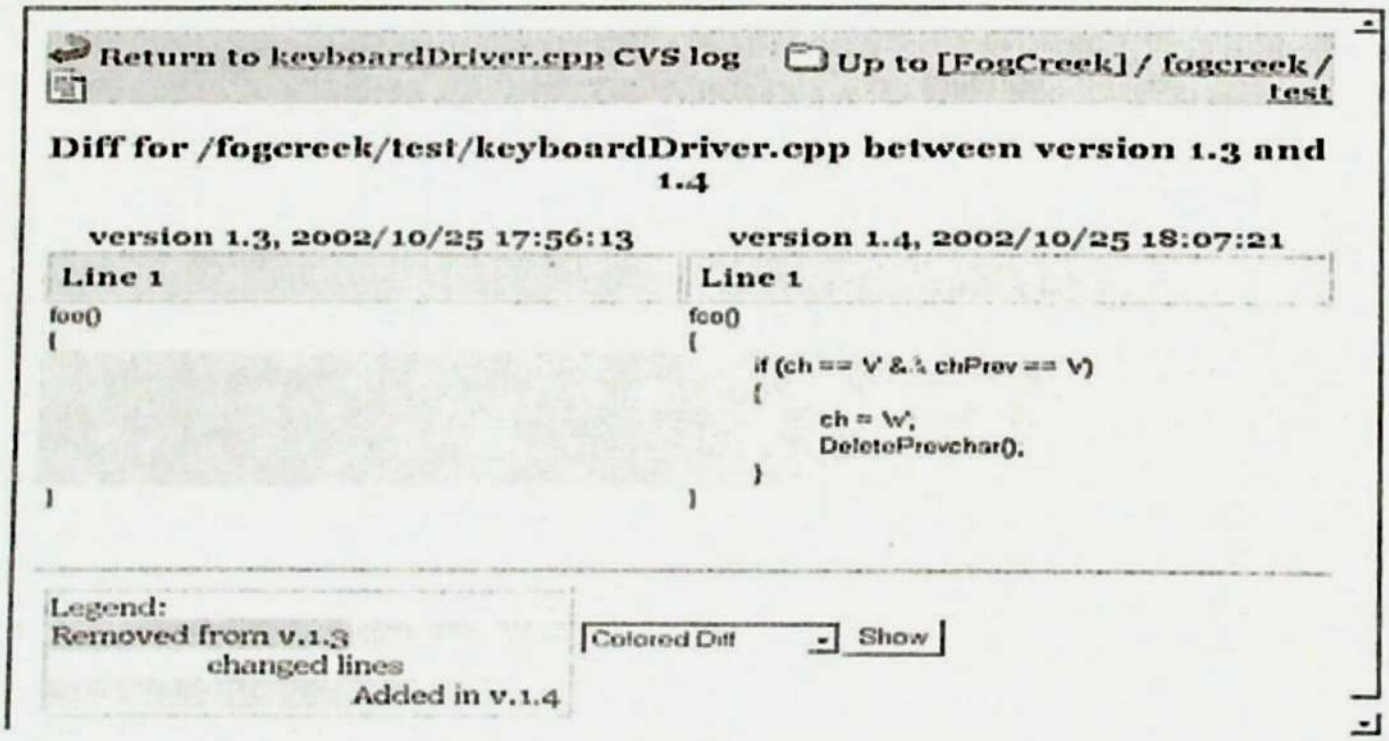
5.1.5 Expandable

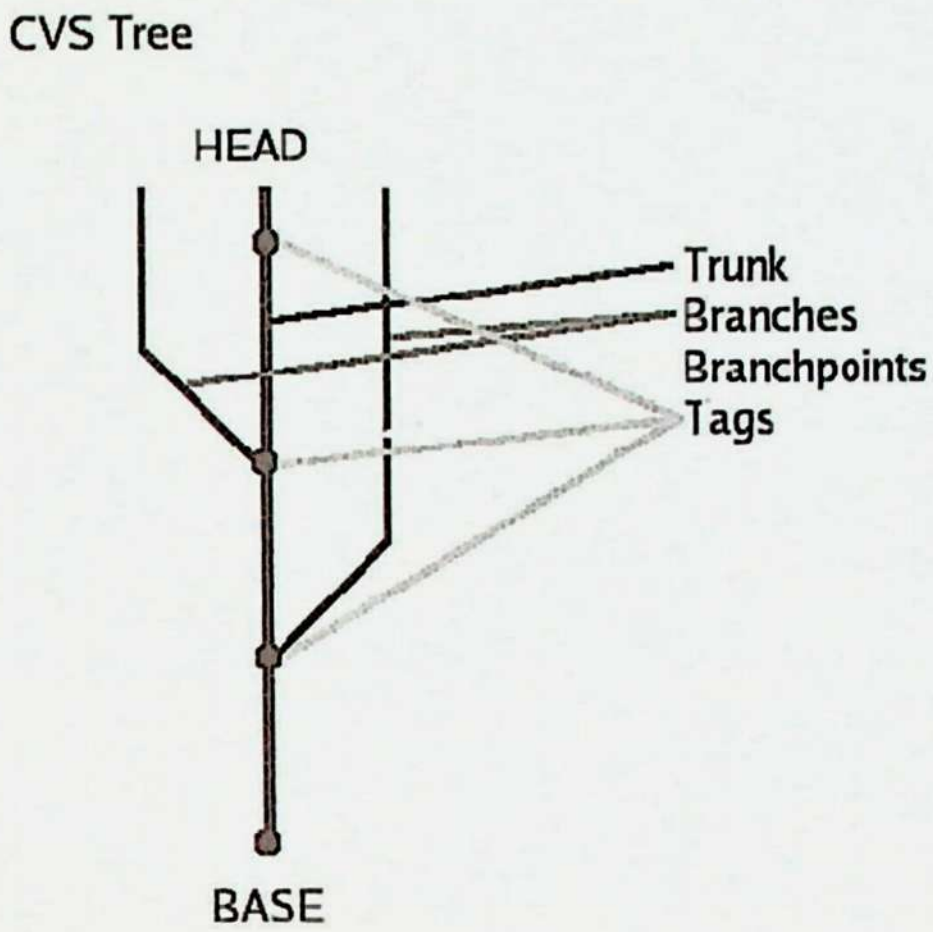
Expandable artinya *source code* dapat dikembangkan lebih lanjut untuk ditambahkan fitur-fitur baru tanpa melakukan perombakan besar-besaran.

Program yang terlalu spesifik dan tidak dapat menangani perkembangan kebutuhan akan menyebabkan keusangan yang pada akhirnya tidak digunakan lagi.

5.1.6 Maintanable

Maintanable: selain *source code* dapat dikembangkan lebih lanjut, perkembangan sour*code* harus dapat dipantau perkembangannya dari waktu ke waktu.
Saat ini tersedia berbagai tools untuk memantau perkembangan *source code* dari waktu ke waktu misalnya CVS.





5.1.7 By *Design*

Good code dibangung berdasarkan desain yang baik , yaitu desain yang terdokumentasi dan solutif terhadap permasahalan yang dihadapi).

6 Prinsip Desain Antarmuka

Overview

Untuk membangun antarmuka yang baik harus diperhatikan prinsip-prinsip dasar pada antarmuka antara *user* dengan komputer. Melalui bab ini akan dibahas prinsip-prinsip pembangunan antarmuka dari berbagai kalangan yang merupakan pakar di bidangnya.

Tujuan

1. Mahasiswa mengetahui prinsip-prinsip dasar dalam pembangunan antar muka dengan komputer.
2. Mahasiswa mampu memberikan penilaian seberapa baik suatu antar muka.
3. Mahasiswa dapat membangun antar muka yang baik berdasarkan prinsip-prinsip dasar.

6.1 Evaluasi Antarmuka

Sebagai pengguna aplikasi komputer, *user* akan memiliki penilaian terhadap aplikasi yang dijalankannya. Berbagai macam penilaian akan muncul terhadap suatu aplikasi yang sama. Seorang *user* akan mengatakan bahwa aplikasi Micorsoft Word itu sulit digunakan. Pernyataan ini akan sangat mungkin dilontarkan oleh seorang pengguna yang baru pertama kali menggunakan aplikasi tersebut. Pernyataan yang berbeda akan dilontarkan oleh *user* yang sudah lama berkecimpung dalam dunia pengolahan kata semenjak dari jaman aplikasi Chi-Writer dan Wordstar. Bagi pengguna yang terakhir ini, aplikasi Microsoft Word merupakan aplikasi yang sangat mudah dan menjadi suatu keajaiban dibandingkan aplikasi pengolah kata sebelumnya. Namun pengguna pertama pun dapat menganggap bahwa aplikasi Micorsoft Word itu sangat mudah setelah ia belajar sepekan.

Pada kasus di atas ada beberapa hal yang berpengaruh terhadap penilaian *user* kepada Microsoft Word yaitu:

- Pengalaman

  User kedua sudah berpengalaman dalam pengolah kata, sehingga mampu membuat perbandingan dengan segera
- Kebiasaan

  User kedua sudah terbiasa dengan aplikasi Microsfot Word selama sepekan belajar sehingga mengubah persepsinya dari semula aplikasi yang sulit menjadi aplikasi yang mudah
- Lama belajar

  Aplikasi Micrososft Word mendukung waktu pembelajaran yang relatif singkat (*learning time*) sehingga *user* dapat menyesuaikan dengan cepat dan memiliki tingkat kenyamanan yang tinggi.
- Kemudahan panduan

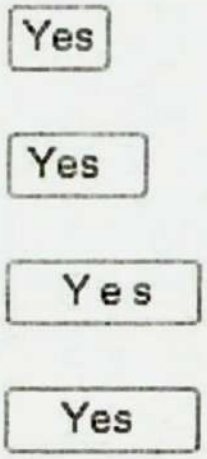
  Boleh jadi, pengguna yang pertama dapat lebih cepat mengubah persepsinya setelah dibantu penggunaan *help* yang interaktif dan tepat sasaran. *Help* yang panjang, bertele-tele, tidak visual dan berbentuk prosa, cenderung ditinggalkan oleh *user*.

Dengan pengalaman dan observasi, seseorang dapat menentukan kualitas desain suatu aplikasi. Berikut ini beberapa contoh berkaitan dengan desain aplikasi.

6.2 *Eight Golden Rules of Dialog Design*

Ben Shneiderman mengusulkan delapan prinsip yang didasarkan pada pengalamannya secara heuristik namun dapat diterpkan pada berbagai macam sistem yang interaktif setelah melalui proses yang panjang dengan pemilahan, perluasan, dan penginterpretasian. Kedelapan prinsip tersebut adalah:

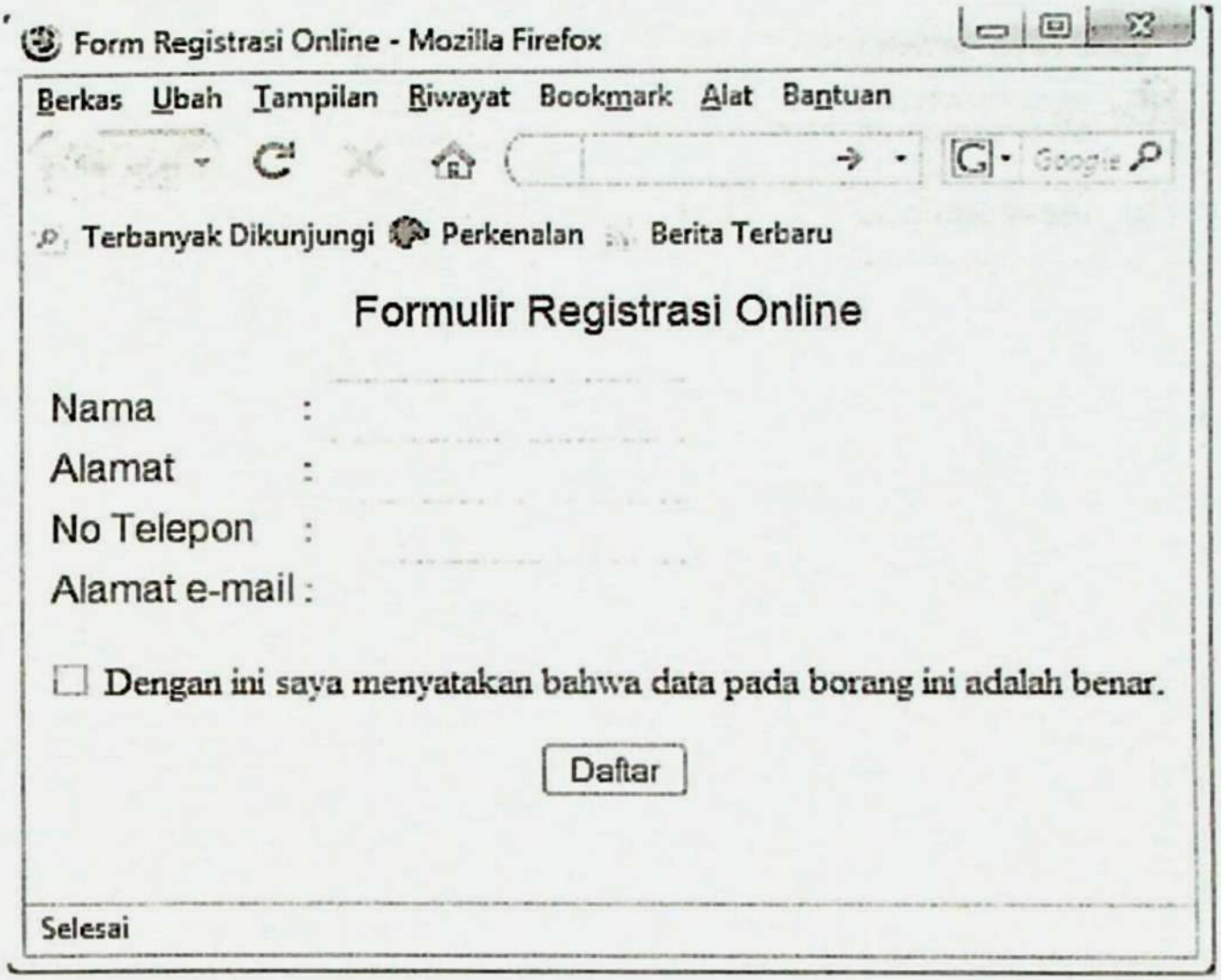
1. Upayakan untuk tetap konsisten
   Konsisten pada serangkaian aksi yang mirip merupakan hal yang penting. Konsistensi ini berkaitan dengan tampilan, menu, bantuan, perintah, pesan-pesan yang disampaikan, dan istilah yang digunakan di layar.



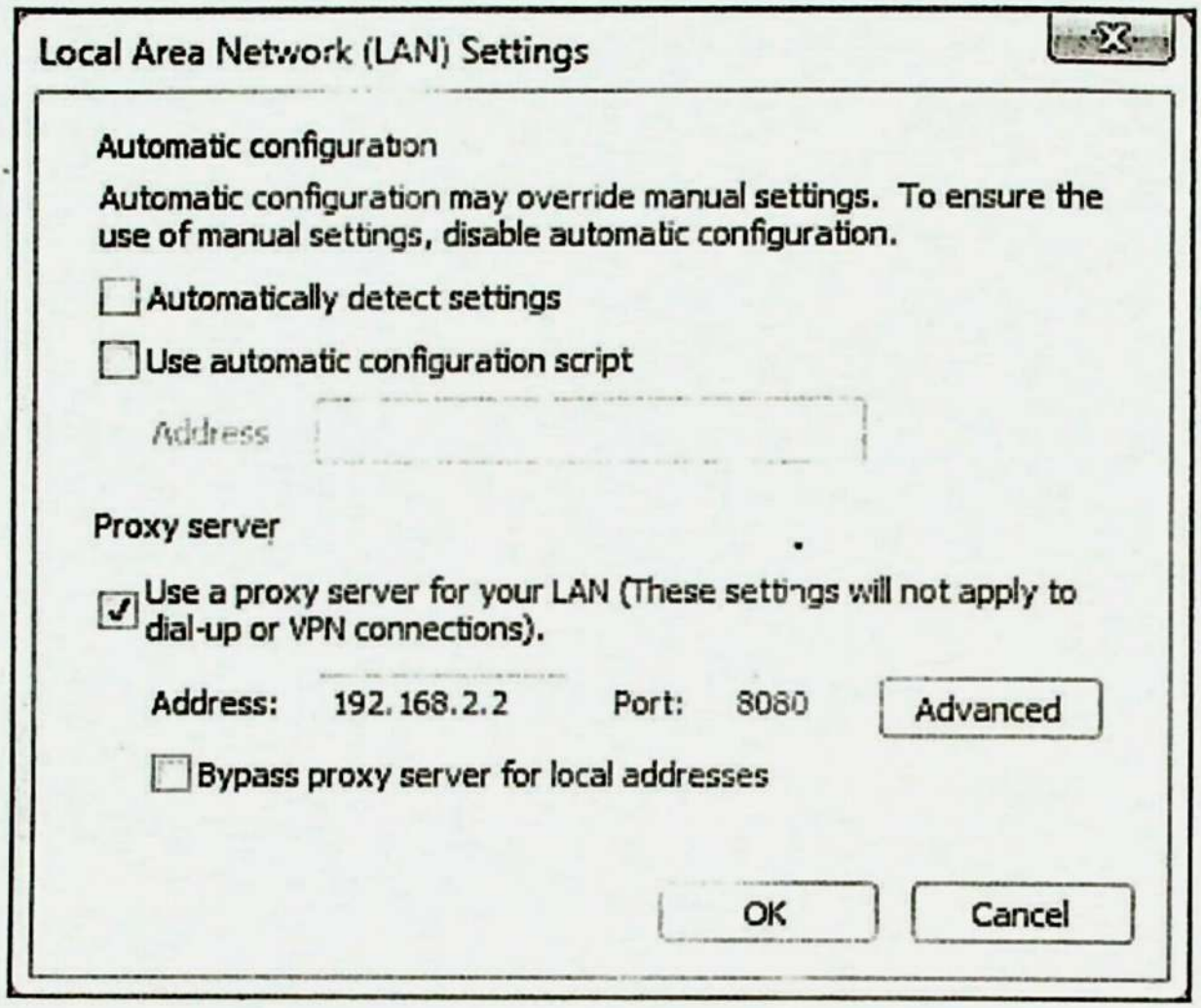
Pada gambar tersebut terdapat berbagai macam *button* 'Yes' dengan berbagai macam variasinya. Namun aplikasi yang konsisten, sekali memilih bentuk *button* 'Yes' maka bentuk tersebut harus senantiasa digunakan pada setiap kali dibutuhkan.

Pada gambar berikut sebuah *form* untuk melakukan registrasi secara online. Perhatikan bahwa terdapat inkonsistensi pada form tersebut yaitu:

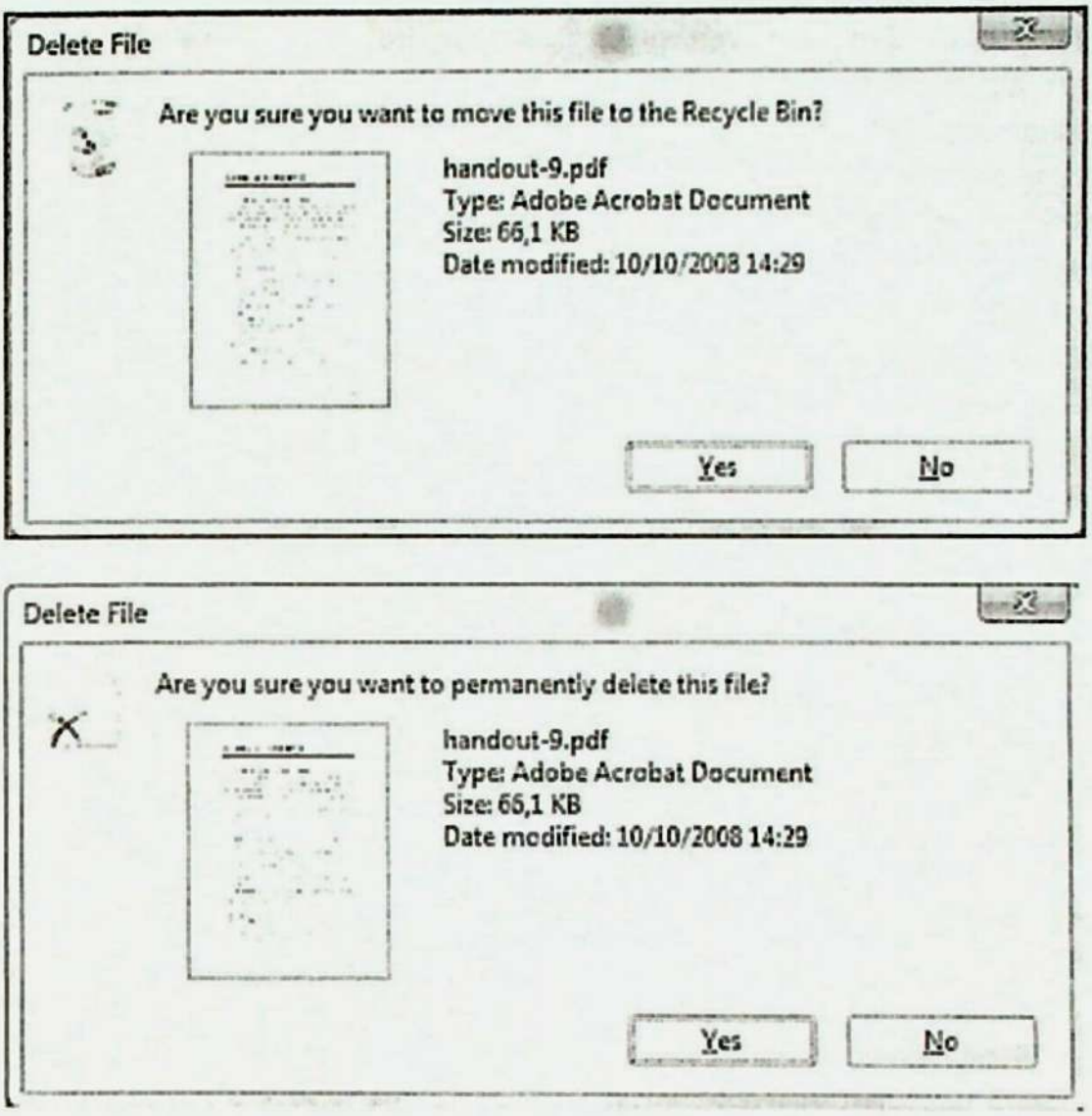
a. Penggunaan istilah *Form* (pada judul window), *Formulir* (pada judul halaman), dan *borang* (pada pernyataan keaslian) untuk mengacu kepada terminologi yang sama.
b. Penggunaan istilah *registrasi* (judul halaman) dan *daftar* (pada *button* bawah) untuk terminologi yang sama.



2. Gunakan *short cut* pada bagian yang sering digunakan.
   Dengan *short cut*, banyaknya interaksi dapat ditekan seminimal mungkin. Semakin sedikit interaksi semakin mudah penggunaan, semakin kecil pula kesalahan. Berbagai macam singkatan, tombol, perintah cepat dengan menggunakan *key* tertentu (*Function, shift, control, tab*, dsb.) sangat berguna bagi *user* yang mahir dengan aplikasi yang digunakannya.
   Contoh short cut dengan *key*:
   - [Alt] + [tab]
   - [Windows]+[e],
   - [Windows]+[d],
   - [Windows]+[r]
   - [alt]+[f4].
   - [Control]+[Alt]+[Del]



3. Sediakan *feedback* yang informatif.
Untuk operasi yang umum berikan feedback yang umum sedangkan untuk operasi yang khusus apalagi berbahaya berikan *feedback* dengan penekanan. Desain yang dibuat harus tetap menginformasikan kepada pengguna mengenai aksi atau interpretasi, perubahan status, adanya kesalahan, atau adanya ekspresi yang relevan dan menarik secara melalui bahasa yang jelas, singkat, tidak ambigu, dan familiar bagi pengguna.



4. Dialog memiliki lingkup tertentu.
   Serangkaian aksi harus diatur kedalam kelompok sedemikian sehingga terdapat bagian awal, pertengahan, dan penutup. *Feedback* yang informatif pada sekelompok aksi tersebut memberikan kenyamanan bagi pengguna dan selanjutnya menjadikan petunjuk untuk serangkaian langkah berikutnya.
5. Sediakan penanganan kesalahan yang sederhana
   Dalam mendesain sistem, harus dibuat sedapat mungkin pengguna tidak melakukan kesalahan yang fatal. Apabila terjadi kesalahan yang serius, sistem harus dapat mendeteksi dan melakukan penanganan kesalahan secara sederhana dan komprehensif.
6. Perbolehkan *user* melakukan aksi mundur atau pembatalan.
   Fitur ini menghilangkan kekhawatiran pengguna karena pengguna melakukan kesalahan, maka aksi tersebut dapat di batalkan. Pembatalan

ini mungkin dapat berupa pembatalan sebuah aksi, sekelompok data entri, ataupun sekelompok aksi.

7. Berikan kontrol internal
   Pengguna cenderung merasa tahu dan mereka mampu mengendalikan sistem. Oleh karena itu desain yang baik harus memposisikan pengguna sebgaia inisiator ketimbang sepagai respoender dari sistem.
8. Kurangi aktifitas mengingat.
   Keterbatasan manusia yang berkaitan dengan memori jangka pendek menuntut desainer sistem membuat antarmukayang sederhana. Sedapat mungkin *user* tidak perlu mengignat terlalu banyak.

6.3 General Principles of *User Interface Design*

Deborah J. Mayhew, memperkenalkan *General Principles Of UI Design*, atau Prinsip Umum Desain *User Interface*. Ada 17 prinsip yang harus dipahami para perancang sistem, terutama untuk mendapatkan hasil maksimal dari tampilan yang dibuat.

1. ***User Compatibility***
   User Compatibility artinya kesesuaian tampilan dengan tipikal dari *user*. karena berbeda *user* bisa jadi kebutuhan tampilannya berbeda. misalnya, jika aplikasi diperuntukkan bagi anak-anak, maka jangan menggunakan istilah atau tampilan orang dewasa.
2. ***Product Compatibility***
   Istilah ini mengartikan bahwa produk aplikasi yang dihasilkan juga harus sesuai, memiliki tampilan yang sama/serupa, baik untuk *user* yang awam maupun yang ahli.
3. ***Task Compatibility***
   Berarti fungsional dari task/tugas yang ada harus sesuai dengan tampilannya. misal untuk pilihan report, orang akan langsung mengartikan akan ditampilkan laporan, sehingga tampilan yang ada bukanlah tipe data (dari sisi pemrogram).
4. ***Work Flow Compatibility***
   Aplikasi bisa dalam satu tampilan untuk berbagai pekerjaan. Jika tampilan yang ada hanya untuk satu pekerjaan saja. Misal untuk kirim mail, maka kita harus membuka tampilan tersendiri untuk daftar alamat.

5. ***Consistency***
   Konsisten. Contohnya, jika anda menggunakan istilah *save* yang berarti simpan, maka gunakan terus istilah tersebut.
6. ***Familiarity***
   Icon disket akan lebih familiar jika digunakan untuk perintah menyimpan.
7. ***Simplicity***
   Aplikasi harus menyediakan pilihan *default* untuk suatu pekerjaan.
8. ***Direct Manipulation***
   Manipulasi secara langsung. Misalnya untuk mempertebal huruf, cukup dengan ctrl+B.
9. ***Control***
   Berikan kontrol penuh pada *user*, tipikal *user* biasanya tidak mau terlalu banyak aturan.
10. **WYSIWYG**
    What You See Is What You Get, buatlah tampilan mirip seperti kehidupan nyata *user*. dan pastikan fungsionalitas yang ada berjalan sesuai tujuan.
11. ***Flexibility***
    Tool/alat yang bisa digunakan *user*. jahgan hanya terpaku pada *keyboard* atau *mouse* saja.
12. ***Responsiveness***
    Tampilan yang dibuat harus ada responnya. misal, yang sering kita lihat ketika ada tampilan *please wait... 68%...*
13. ***Invisible Technology***
    User tidak penting mengetahui algoritma apa yang digunakan. Contohnya untuk mengurutkan pengguna tidak perlu mengetahui algoritma yang digunakan *programmer* (*max sort, bubble sort, quick sort,* dst)
14. ***Robustness***
    Handal. Dapat mengakomodir kesalahan *user*. jangan malah *error*, apalagi sampai *crash*.
15. ***Protection***
    Melindungi *user* dari kesalahan yang umum dilakukan. misalnya dengan memberikan fitur *back* atau *undo*.
16. ***Ease of Learning***
    Aplikasi mudah dipelajari.
17. ***Ease of use***
    Aplikasi harus mudah digunakan

DAFTAR PUSTAKA

1. Langer, Arthur M. 2008. *Analysis and Design of Information Systems 3rd edition*. Springer.
2. Ian Sommerville, *Software Engineering, 5th Edition*, Addison Wisley

Tgl. Kembali		Tgl. Kembali		Tgl. Kembali	
1		25		49	
2		26		50	
3		27		51	
4		28		52	
5		29		53	
6		30		54	
7		31		55	
8		32		56	
9		33		57	

giving and caring the world

Jl. Diponegoro No. 18 Bandung 40115
Jl. Telekomunikasi Dayeuhkolot Bandung 40257
Phone : +62 22 421 6644 Fax : +62 22 426 5091
Website : www.politekniktelkom.ac.id