MODUL PRAKTIKUM



IMPLEMENTASI
PERANGKAT LUNAK

UNIVERSITAS TEKNOLOGI MATARAM

Jl. Kampus Universitas Teknologi Mataram, Kekalik, Mataram

KATA PENGANTAR

Modul praktikum ini disusun sebagai pedoman bagi mahasiswa Universitas Teknologi Mataram yang mengikuti praktikum **Implementasi Perangkat Lunak.** Tujuan dari pelaksanaan praktikum implementasi perangkat lunak ini adalah untuk mendukung mata kuliah Rekayasa Perangkat Lunak yang diberikan kepada mahasiswa.

Di dalam kegiatan praktikum ini, akan dipelajari dan dipraktekan tahapan-tahapan dalam proses rekayasa perangkat lunak. Susunan modul ini terdiri dari tujuan, pembahasan teori praktis, tugas-tugas praktikum dan tugas-tugas pendahuluan/rumah yang harus dikerjakan oleh para praktikan. Diharapkan para praktikan telah mempersiapkan materi yang akan diberikan pada praktikum demi kelancarannya.

Modul praktikum Implementasi Perangkat Lunak ini terdiri dari enam modul dengan topik bahasan sebagai berikut :

- **Modul 0** : Pendahuluan, berisi tujuan umum praktikum, pembahasan singkat mengenai rekayasa perangkat lunak dan referensi.
- **Modul 1** : Proposal Proyek Perangkat Lunak, berisi pembahasan penyusunan proposal proyek perangkat lunak.
- **Modul 2** : Analisis Sistem, berisi pembahasan tentang analisis sistem perangkat lunak
- **Modul 3** : Metode perancangan, berisi pembahasan tentang perancangan suatu perangkat lunak.
- **Modul 4** : Implementasi, berisi tentang penyusunan dokumen implementasi
- **Modul 5** : Pengujian, berisi tentang pembahasan pengujian perangkat lunak yang telah dibangun.

Materi yang diberikan dalam modul dan pada saat praktikum masih belum lengkap dan untuk itu praktikan diharapkan dapat mencari referensi tambahan yang diperlukannya baik di perpustakaan maupun melalui media internet. Selain itu praktikan diharapkan mengikuti mata kuliah Rekayasa Perangkat Lunak dengan baik, karena salah satu sumber selain modul adalah materi yang diberikan pada saat kuliah.

Modul ini masih belum sempurna, sehingga perlu dikaji baik oleh dosen pengajar, instruktur, asisten maupun praktikan yang terlibat dalam praktikum. Oleh karena itu penyusun berharap agar para pemakai modul ini dapat memberikan sumbangan saran untuk perbaikan modul rekayasa perangkat lunak ini.

Semoga modul ini dapat bermanfaat bagi para personil yang terlibat dalam praktikum rekayasa perangkat lunak, serta dapat meningkatkan kemampuan mahasiswa dalam menguasai proses-proses dalam rekayasa perangkat lunak.

Mataram, Agustus 2019

Penyusun

DAFTAR ISI

LEMBAR PENGESAHAN i

KATA PENGANTAR ii

DAFTAR ISI iv

DAFTAR TABEL viii

DAFTAR GAMBAR ix

DAFTAR LAMPIRAN x

TEKNIK PELAKSANAAN PRAKTIKUM xi

TEKNIK PENILAIAN xiii

MODUL 0 PENDAHULUAN

0.1 Tujuan 0-1

0.2 Teori 0-1

0.3 Referensi 0-3

MODUL I PROPOSAL PROYEK PERANGKAT LUNAK

1.1 Tujuan I-1

1.2 Teori I-1

1.2.1 Fungsi Dalam Pengembangan Perangkat Lunak I-1

1.2.2 Dokumen Rencana Pengembangan Perangkat Lunak (RPPL) I-2

1.3 Proyek Perangkat Lunak I-6

1.4 Tugas Pendahuluan I I-10

1.5 Tugas Rumah I I-10

1.6 Latihan Praktikum I I-10

MODUL II ANALISIS SISTEM II-1

2.1 Tujuan II-1

2.2 Teori II-1
2.2.1 Penggalian *Requirement* II-1
2.2.1.1 Analisis Kebutuhan Perangkat Lunak II-7
2.2.1.2 Prinsip Analisis II-8
2.2.2 Pemodelan Sistem II-8
2.2.2.1 Pemodelan dalam Analisis Terstruktur II-8
2.2.2.2 Dokumen yang Terlibat dalam Fase Analisis atau Spesifikasi II-10
2.2.2.3 CSPEC (*Control Specification*) II-10
2.2.2.4 PSPEC (*Process Specification*) II-11
2.2.2.5 Kamus Data (*Data Dictionary*) II-11
2.2.2.6 Pemodelan Kelakuan (*Behaviour Model*) II-12
2.2.2.7 SRS (*Software Requirement Specification*) II-13
2.2.2.8 Software Specification Review II-14
2.3 Tugas-Tugas Pendahuluan II-15
2.3.1 Tugas Pendahuluan II II-15
2.3.2 Tugas Pendahuluan III II-15
2.3.3 Tugas Pendahuluan IV II-16
2.3.4 Tugas Pendahuluan V II-16
2.3.5 Tugas Pendahuluan VI II-16
2.4 Latihan - Latihan Praktikum II-16
2.4.1 Latihan Praktikum II (Pertemuan ke dua) II-16
2.4.2 Latihan Praktikum III (Pertemuan ke tiga, dan ke empat) II-16
2.4.3 Latihan Praktikum IV (Pertemuan ke lima) II-17

MODUL III METODE PERANCANGAN

3.1 Tujuan III-1
3.2 Teori III-1
3.2.1 Perancangan Awal (*Preliminary Design*) III-1
3.2.1.1 Perancangan Data III-1
3.2.1.2 Perancangan Arsitektural III-2
3.2.1.2.1 Proses Perancangan Arsitektural III-2

3.2.1.2.2 Pemetaan Pengubahan (*Transform Mapping*) III-3
3.2.1.2.3 Pemetaan Transaksi (*Transaction Mapping*) III-8
3.2.2 Perancangan Rinci (*Detail Design*) III-10
3.2.2.1 Perancangan Antarmuka III-10
3.2.2.2 Perancangan Prosedur III-11
3.2.3 SDD (*Software Design Document*) III-14
3.3 Tugas-Tugas Pendahuluan III-16
3.3.1 Tugas Pendahuluan VII III-16
3.3.2 Tugas Pendahuluan VIII III-16
3.3.3 Tugas Pendahuluan IX III-16
3.4 Latihan - Latihan Praktikum III-16
3.4.1 Latihan Praktikum IV (Pertemuan Ke Enam) III-16
3.4.2 Latihan Praktikum V (Pertemuan Ke Tujuh Dan Ke Delapan) III-17
3.4.3 Latihan Praktikum VI (Pertemuan Ke Delapan Dan Ke Sembilan) III-17

MODUL IV IMPLEMENTASI

4.1 Tujuan IV-1
4.2 Teori IV-1
4.3 Tugas Pendahuluan X IV-5
4.4 Latihan Praktikum VII (Pertemuan Ke Sembilan Dan Ke Sepuluh) IV-5

MODUL V PENGUJIAN (TESTING)

5.1 Tujuan V-1
5.2 Teori V-1
5.2.1 Teknik Pengujian Perangkat Lunak V-1
5.2.2 Objektif Pengujian V-1

5.2.3 Prinsip Pengujian V-2
5.2.4 *Testability* V-2
5.2.5 Perancangan Kasus Uji V-3
5.2.6 Pengujian *White Box*/*Glassbox* V-4
5.2.7 *Basis Path Testing* V-4
5.2.8 *Black Box Testing* V-6
5.2.9 Strategi Pengujian Perangkat Lunak V-8
5.2.10 Dokumen Pengujian (STP [*Software Test Plan*] dan STR [Software Test Result]) V-10
5.3 Tugas Pendahuluan XI V-13
5.4 Latihan Praktikum VIII (Pertemuan Ke Sebelas Dan Ke Duabelas) V-13

LAMPIRAN

DAFTAR TABEL

Tabel 1.1	Kerangka Dokumen Rencana Pengembangan Perangkat Lunak (RPPL)	I-2
Tabel 2.1	Notasi DFD	II-8
Tabel 2.2	Tabel CSPEC	II-10
Tabel 2.3	Tabel Kamus Data	II-12
Tabel 2.4	Kerangka SRS	II-13
Tabel 3.1	Notasi Flowchart	III-13
Tabel 3.2	Notasi box Diagram/Nassi-Shneiderman Chart (N-S Chart)/Chapin Chart	III-13
Tabel 3.3	Kerangka SDD	III-14
Tabel 4.1	Kerangka Dokumen Implementasi	IV-1
Tabel 4.2	Tabel penjelasan struktur program	IV-4
Tabel 5.1	Notasi *Flow Graph*	V-4
Tabel 5.2	Kerangka Dokumen Pengujian	V-10
Tabel 5.3	Contoh *Software Test Plan* (STP)	V-12
Tabel 5.4	Contoh *Software Test Result* (STR)	V-12

DAFTAR GAMBAR

Gambar 3.1	*Data Context Diagram Software* Pengamanan Rumah	III-3
Gambar 3.2	*Data Flow Diagram* level 1 *Software* Pengamanan Rumah	III-4
Gambar 3.3	*Data Flow Diagram* level 2 (Proses Monitor Sensor)	III-5
Gambar 3.4	*Data Flow Diagram* level 3 (Proses Monitor Sensor)	III-5
Gambar 3.5	*Factoring* level awal/pertama untuk Proses Monitor Sensor	III-6
Gambar 3.6	*Factoring* level kedua Proses Monitor Sensor	III-7
Gambar 3.7	Iterasi pertama dari Struktur Program Monitor Sensor	III-7
Gambar 3.8	Perbaikan dari Struktur Program Monitor Sensor	III-8
Gambar 3.9	Data Flow Diagram level 2 (Proses *Interaction*)	III-9
Gambar 3.10	*Factoring* level awal/pertama untuk Proses *Interaction*	III-9
Gambar 3.11	Iterasi Pertama dari Struktur Program untuk Proses *Interaction*	III-10
Gambar 4.1	Struktur Program Sistem Informasi Penggajian	IV-4
Gambar 4.2	Contoh Form *Input Password*	IV-5
Gambar 4.3	Contoh Form Pesan Kesalahan Password	IV-5

MODUL 0

PENDAHULUAN

(0 kali pertemuan, untuk dipelajari di rumah)

1 TUJUAN

Praktikum Rekayasa Perangkat Lunak bertujuan untuk :

- Memberikan pengetahuan kepada praktikan tahapan-tahapan dalam pembuatan Rekayasa Perangkat Lunak.
- Praktikan bisa menyelesaikan proyek Perangkat Lunak.
- Praktikan bisa membiasakan diri untuk menyelesaikan Proyek Perangkat Lunak secara terstruktur baik dalam satu tim maupun individu.
- Praktikan bisa menerapkan metodologi Rekayasa pembuatan Perangkat Lunak pada suatu kasus tertentu yang diberikan.
- Praktikan dapat membuat dokumen-dokumen yang diperlukan dalam rekayasa perangkat lunak
- Menunjang mata kuliah Rekayasa Perangkat Lunak
- Memberikan wawasan kepada praktikan untuk menghadapi mata kuliah Manajemen Proyek Perangkat Lunak, Kerja Praktek dan Tugas Akhir.
- Membiasakan praktikan untuk menyelesaikan tugas/pekerjaan tepat waktu sesuai yang telah dijadwalkan.

2 TEORI

Perangkat lunak dibangun untuk memproses data, mentransformasikan data dari satu bentuk ke bentuk yang lain baik untuk proses "*batch*" maupun proses *real time*. Selain itu perangkat lunak juga melakukan pemrosesan kejadian/*event*. Sebuah *event* mewakili beberapa aspek dari kendali sistem yang sebenarnya adalah data Boolean. Data dan kendali tersebut berada pada domain informasi dari suatu masalah yang terdiri dari :

- **Kandungan informasi,** menggambarkan data dan objek kendali yang terdiri dari kumpulan informasi yang ditransformasikan oleh *software*
- **Aliran informasi**, menggambarkan cara bagaimana data dan kendali berubah di seluruh sistem
- **Struktur informasi**, menggambarkan organisasi internal dan berbagai data dan kendali.

Software development process :

1. *To specify what the software shall do*
2. *To define how the software shall be done*
3. *To implement Test*
4. *To Integrate/Test module by module*
5. *To validate the software as a whole*

Tahapan pembangunan *software*/perangkat lunak, pada umumnya terdiri dari :

1. Analisis

Tahap analisis sistem merupakan salah satu tahap yang sangat penting, karena banyak kesalahan dalam suatu perangkat lunak yang terjadi sejak dari fase ini dan terlambat terdeteksi sehingga akan memerlukan waktu dan biaya yang tidak sedikit untuk memperbaikinya.

Panduan untuk tahap ini diantaranya :

- Pahami masalahnya sebelum menciptakan model analisisnya
- Kembangkan *prototype* yang membuat pengguna mengerti bagaimana interaksi antara mesin dan manusia akan terjadi
- Catat awal dan alasannya dari setiap kebutuhan
- Gunakan berbagai tinjauan/pandangan kebutuhan
- Buat urutan prioritas dari kebutuhan
- Lakukan upaya untuk menghilangakan *ambiguitas*

2. Perancangan

Setelah fase analisis selesai dilakukan perancangan berdasarkan hasil analisis. Pada fase ini dilakukan perancangan data, struktur program, antar muka dan prosedur-prosedur perangkat lunak.

3. **Pengkodean (Implementasi)**

   Selanjutnya setelah fase perancangan selesai dilakukan implementasi atau pengkodean berdasarkan hasil perancangan dengan menggunakan bahasa pemrograman tertentu.

4. **Pengujian**

   Pada fase ini dilakukan pengujian apakah perangkat lunak yang dibangun telah sesuai yang diharapkan, jika telah sesuai maka perangkat lunak ini dapat di integrasikan untuk digunakan. Sebaliknya jika masih ada kekurangan atau kesalahan maka dapat mengulangi mulai dari fase analisis sampai pengujian.

Pembangunan/pengembangan perangkat lunak pada umumnya dilakukan oleh lebih dari satu orang atau sekelompok orang (tim) walaupun tidak menutup kemungkinan dikerjakan oleh satu orang saja (untuk kasus-kasus tertentu). Masing-masing anggota dalam tim memiliki tugas dan tanggung jawab terhadap perangkat lunak yang akan dibangun. Masalah yang dialami oleh salah satu anggota dalam tim dapat mempengaruhi anggota tim lain dan dapat mengakibatkan keterlambatan penyelesaian perangkat lunak tersebut. Setiap anggota tim memiliki harus bekerjasama untuk menyelesaikan perangkat lunak sesuai fungsinya dan tepat waktu.

3 REFERENSI

Referensi-referensi yang dapat digunakan selain modul praktikum dan materi mata kuliah Rekayasa Perangkat Lunak diantaranya :

- Pressman, Roger S., "*Software Engineering : A Practitioner's Approach 4th Edition*", Mc-Graw Hill, 1997.
- Yourdon, Edward, "*Modern Structured Analysis*", Prentice Hall, 1989.
- Davis, Allan M.," *Software Requirements : Analysis & Specification*", Prentice Hall.
- Sommerville, Ian, *"Software Engineering 6th Edition"*, Addison-Wesley, 2001.

- Behforooz, Ali; Hudson, Frederick J., *"Software Engineering Fundamentals"*, Oxford University Press, 1996.
- Mayhew, Deborah J., *"The Usability Engineering Life Cycle (A Practitioner's Handbook For User Interface Design)"*, Morgan Kaufmann Publisher, 1999.
- Suharto, Toto, "*Rekayasa Perangkat Lunak Template Dokumen & Contoh Dokumentasi*", ITB, Bandung, 2002
- Laporan Praktikum Rekayasa Perangkat Lunak, Laboratorium Komputer Sains Jurusan Teknik Informatika Fakultas Teknik – Univerisitas Widyatama, 2001 - 2002.
- Jurnal – Jurnal Rekayasa Perangkat Lunak.
- Internet.

MODUL I

RENCANA PENGEMBANGAN PERANGKAT LUNAK (RPPL)

(1 kali pertemuan)

1.1 TUJUAN

Tujuan modul ini, adalah:

- Praktikan bisa membuat dokumen rencana pengembangan perangkat lunak.
- Praktikan dapat membiasakan diri untuk menyusun Dokumen Rencana Pengembangan Perangkat Lunak (Proposal) secara terstruktur baik dalam satu tim maupun individu.
- Praktikan memahami organisasi tim dalam proyek perangkat lunak

1.2 TEORI

1.2.1 Fungsi dalam Pengembangan Perangkat Lunak

Software Development Management (terdiri dari banyak fungsi dan tim), yaitu :

1. ***Software Project Manager***: pertama berhubungan dengan konsumen, menetapkan anggaran dan jadwal pelaksanaan proyek perangkat lunak.
2. ***Software Engingeering***

   Analyst : berhubungan dengan konsumen secara lebih rinci; bertugas mendeskripsikan atau menggali fungsi dan unjuk kerja *software* yang akan dibangun.

   Designer : bertugas merancang algoritma/prosedur yang tepat untuk fungsi tersebut disesuaikan dengan *hardware* atau *software* pendukung yang ada.

   Programmer : mengimplementasikan algoritma dalam bentuk kode-kode program menggunakan bahasa pemograman.

3. ***Software Configuration Management*** : memantau fungsi-fungsi/prosedur-prosedur yang telah ditentukan, mencatat konfigurasi pada tahap-tahap/ waktu-waktu tertentu berdasarkan kenyataan yang ada.

   System Administrator : bertugas melakukan pengelolaan terhadap sistem pada saat diimplementasikan.

4. ***Software Quality***

   Software Test Engineer : bertugas melakukan pengujian sistem.

   Software Quality Assurance: bertugas melakukan pengawasan apakah *software* yang dibangun telah berjalan sesuai dengan fungsi dan kebutuhannya.

1.2.2 Dokumen Rencana Pengembangan Perangkat Lunak (RPPL)

Pada umumnya sebelum melakukan pengembangan atau pembangunan suatu perangkat lunak, terlebih dahulu dibuat proposal proyek pengembangan atau pembangunan perangkat lunak tersebut. Hal ini bertujuan untuk memberikan gambaran secara ringkas mengenai perangkat lunak yang akan dikembangkan atau dibangun.

Format/kerangka dari dokumen Rencana Pengembangan Perangkat Lunak (RPPL) adalah sebagai berikut :

Tabel 1.1 Kerangka Dokumen Rencana Pengembangan Perangkat Lunak (RPPL)

Kerangka Dokumen	Keterangan
Abstraksi	Abstraksi/Rangkuman dokumen (RPPL)
Daftar Isi Daftar Gambar Daftar Tabel	Daftar Isi, Daftar Gambar dan Daftar Tabel dalam Dokumen RPPL
1 Pendahuluan	
1.1 Gambaran Umum Proyek	Ringkasan dari latar belakang dan lingkup proyek (serta hubungannya dengan proyek lain bila ada)
1.2 Tujuan	Tujuan penyusunan dokumen RPPL
1.3 Daftar Definisi dan Singkatan	Menjelaskan definisi dan singkatan dalam RPPL
1.4 Referensi	Referensi/dokumen/bahan acuan yang digunakan
2 Organisasi Proyek	
2.1 Struktur Organisasi	Struktur organisasi tim pengembang dengan mengidentifikasi dan menggambarkan jalur komunikasi dan pertanggungjawaban tim
2.2 Otoritas, Hak dan Tanggung Jawab	Otoritas, hak dan tanggung jawab tiap anggota tim

Kerangka Dokumen	Keterangan
Anggota Tim	
3 Proses Manajerial	
3.1 Tujuan dan Prioritas Manajemen	Tujuan dan prioritas proyek pengembangan perangkat lunak
3.2 Asumsi, Ketergantungan dan Kendala	Menjelaskan asumsi yang digunakan pada pelaksanaan proyek, kebergantungan pada hal yang eksternal dan kendala yang perlu dipertimbangkan
3.3 Batasan Pengembangan Proyek	Menjelaskan batasan pengembangan proyek
3.4 Dokumentasi Perangkat Lunak	Menjelaskan dokumen yang akan dilaporkan dan waktu penyerahan dokumen
3.5 Rencana Penugasan	Berdasarkan struktur organisasi (poin 2.1), sebutkan jumlah dan tipe/jenis personalia yang dibutuhkan, menyangkut : • Keahlian • Saat mulai • Cara memfungsikan (*retaining*) dan memberhentikan personalia
4 Proses Teknis	Menjelaskan tentang rencana penggunaan : • Sistem Komputer (*Hardware* dan *Software*) • Metode pengembangan (pemodelan) • Notasi, alat Bantu, teknik dan metode lain yang digunakan.
5 Paket Kerja dan Jadwal	Menjelaskan : • Paket kerja (menjelaskan tugas dari masing-masing anggota tim pada setiap tahap) • Jadwal Pelaksanaan
Lampiran	Berisi penjelasan tambahan pada laporan ini

Berikut ini adalah beberapa contoh isi dokumen rencana pengembangan perangkat lunak untuk kasus Mesin Jual Otomatis

1.1 Gambaran Umum Proyek

Sebuah perusahaan yang bergerak dibidang pemasaran produk makanan, minuman, rokok dan surat kabar, dalam rangka meningkatkan hasil penjualan dan mutu pelayanannya bermaksud untuk menggunakan Mesin Jual Otomatis (MJO) untuk melayani konsumen yang ingin membeli produk-produk yang dipasarkannya.

Mesin Jual Otomatis (MJO) tersebut memiliki panel kendali yang berfungsi sebagai interface antara konsumen dengan MJO. Pada panel kendali tersebut terdapat tombol yang berfungsi untuk memilih jenis produk yang akan dibeli dan tombol yang digunakan untuk memasukan jumlah produk yang dibeli, selain itu panel kendali ini mempunyai layar yang berfungsi untuk memberikan pesan-pesan yang berkaitan dengan penjualan produk yang dipasarkannya. Pada panel kendali ini juga terdapat fasilitas untuk memasukan koin dari konsumen.

Mesin ini mampu mendeteksi jumlah uang dan nilai uang yang dimasukan. Bila konsumen telah memasukan sejumlah koin tertentu, maka mesin akan menghitung nilai uang tersebut. Jika nilai uang tersebut cukup untuk membayar produk yang dipilih, maka mesin akan mengeluarkan produk yang diinginkan. Jika nilai koin yang dimasukan lebih dari nilai produk yang dipilih, maka mesin akan mengeluarkan koin kembaliannya.

Sedangkan jika nilai koin tidak cukup maka mesin akan memberikan pesan bahwa uang yang dimasukan tidak cukup. Selain itu mesin juga akan memberikan pesan jika produk yang diinginkan habis, koin kembalian tidak cukup, dan penampung uang telah penuh.

Jenis koin yang dideteksi adalah uang dengan pecahan seribuan, lima ratusan dan seratusan. Tabel harga produk disimpan dalam file database yang menyimpan informasi tentang jenis produk, harga dari masing- masing produk serta stok dari masing-masing produk tersebut dan kapasitas penyimpanan uang.

1.2 Tujuan

Dokumen ini mendefinisikan aktifitas dan tanggung jawab dari perusahaan yang memberikan kontrak dan pihak pengembang

Klien dibagi menjadi dua komponen funsional utama dalam sistem yang dikembangkan yaitu sistem antar muka dan sistem intelegensia

……dst

1.3 Referensi

Untuk penanganan proyek ini digunakan acuan dokumen sebagai berikut:

- Pressman, Roger S., "*Software Engineering : A Practitioner's Approach 4th Edition*", Mc-Graw Hill, 1997.
- Yourdon, Edward, "*Modern Structured Analysis*", Prentice Hall, 1989
- Davis, Allan M.," *Software Requirements : Analysis & Specification*", Prentice Hall
- … dan seterusnya

….dst

3.1 Tujuan dan Prioritas Manajemen

Membangun aplikasi MJO (Mesin Jual Otomatis) untuk memenuhi kebutuhan perusahaan dalam rangka meningkatkan hasil penjualan dan mutu pelayanannya terhadap konsumen.

3.3 Batasan Masalah

Mesin ini dibuat untuk melayani konsumen yang ingin membeli produk makanan, minuman, rokok dan surat kabar secara otomatis. Uang yang digunakan untuk melakukan transaksi pada mesin ini adalah uang koin dengan pecahan 100, 500 dan 1000.

Proyek ini meliputi pengembangan secara lengkap dari MJO meliputi spesifikasi, perancangan, implementasi serta pengujian dari sebagian produk MJO tersebut.

3.4 Dokumentasi Perangkat Lunak

Proyek ini harus menyerahkan dokumen-dokumen sebagai berikut :

Dokumen Analisis (SRS)

Dokumen Perancangan (SDD)

Dokumen Implementasi

Dokumen Pengujian (STP dan STR)

Software Aplikasi dan Code Program

3.5 Rencana Penugasan

Proyek ini dikerjakan oleh tim pengembang yang terdiri dari :

Software Project Manager : Amir
Software Analyst : Budi
Eni
Software Designer : Cici
Amir
Software Quality Assurance : Deny
Software Developer : Eni
Deny

5 Paket Kerja dan Jadwal Pelaksanaan

Proyek ini dilaksanakan selama praktikum RPL, review dan tanggal penyerahan akan diatur sesuai dengan persetujuan project manager.

No	Deskripsi	Pertemuan											
		1	2	3	4	5	6	7	8	9	10	11	12
1	Persiapan												
2	Analisis												
3	Perancangan												
4	Implementasi												
5	Pengujian												

1.3 PROYEK PERANGKAT LUNAK

Kasus 1: Aplikasi pada Distributor Furniture

Sebuah Distributor Furniture membutuhkan sebuah aplikasi yang dapat digunakan dalam kegiatan usahanya. Aplikasi tersebut diharapkan dapat menangani:

1. Pengelolaan barang masuk dan keluar
2. Pengelolaan stok gudang
3. Pengelolaan data supplier
4. Pengelolaan penjualan (Pendapatan) dan pembelian (Utang)
5. Laporan penjualan, pembelian, dan stok per periodik

Skenario :

Petugas Gudang mengecek stok gudang, bila ada barang yang tinggal sedikit atau habis maka dilakukan pemesanan kepada supplier yang biasa menyuplai barang tersebut. Barang pesanan yang datang disimpan di gudang (Stok). Pembayaran kepada supplier dilakukan sebulan setelah barang diterima (Utang).

Bila ada pemesanan/penjualan barang, periksa stok barang yang di minta, bila tersedia buat surat jalan. Bila barang tidak ada/kurang lakukan pemesanan kepada suplier. Pembayaran oleh konsumen dapat dilakukan dengan DP (Down Payment) terlebih dulu, sisanya dilunasi setelah barang sampai di alamat (Cash Only).

Kasus 2: Rental CD/VCD/DVD

Sebuah Rental CD/VCD/DVD menyediakan layanan peminjaman CD/VCD/DVD kepada anggotanya. Untuk mendukung kegiatan usaha ini diperlukan sebah perangkat lunak yang dapat digunakan untuk memudahkan kegiatan operasional. Kegiatan operasional tersebut diantaranya

1. Pengelolaan data anggota (pendaftaran) dan koleksi (input, edit, hapus)
2. Transaksi peminjaman dn pengembalian
3. Penghitungan denda keterlambatan (Rp 1000/koleksi/hari)
4. Laporan data anggota, data koleksi yang sedang di pinjam dan yang tersedia.
5. Laporan peminjaman dan pengembalian / periodik
6. Laporan keuangan

Ket. Pendaftaran anggota pertama Rp 20.000; Biaya pinjam / koleksi Rp 2500; Periode keanggotaan selama 1 tahun. Untuk biaya perpanjangan anggota tahun berikutnya Rp 10.000;

Kasus 3 : Flight Reservation System

Gatotkaca airline merupakan pelayanan airline yang menyediakan penerbangan domestik. Sebagai improvisasi terhadap layanan kepada penumpang dan memberikan jadwal penerbangan yang berjalan baik, maka perlu dibuat sebuah perangkat lunak komputer yang dapat melayani pemesanan tiket penerbangan.

Perangkat lunak ini dapat dijalankan dengan multiuser sistem dimana bisa memberikan tambahan pemesanan sekitar 15%. Polis pemesanan ini berdasarkan asumsi bahwa sedikitnya 15% dari total angka pemesanan penumpang pada sebuah penerbangan akan batal atau tidak diperlihatkan, sehingga biasanya setiap penerbangan akan membawa penumpang dengan jumlah maksimum.

Pada pengaplikasiannya banyak penumpang yang dapat melihat jumlah kursi kosong, penumpang yang batal berangkat dengan biaya beban sebesar 10% untuk pemesanan pada penerbangan berikutnya. Biaya beban ini merupakan kompensasi terhadap transaksi yang telah dilakukan oleh calon penumpang dengan pihak Gatotkaca Airline.

Perangkat lunak ini ditujukan untuk memudahkan calon penumpang melakukan transaksi-transaksi yang berupa pemesanan tiket, pembatalan keberangkatan bahkan calon penumpang dapat mengetahui jumlah kursi yang kosong. Sehingga tidak ada keraguan bagi penumpang untuk melakukan transaksi dan tanpa harus menunggu antrian di loket pemesanan.

Kasus 4 : Aplikasi Simpan Pinjam di Koperasi

Sebuah lembaga koperasi yang bergerak di bidang simpan pinjam membutuhkan sebuah aplikasi yang dapat meningkatkan aktifitas kinerja koperasi. Yang melatarbelakangi dibutuhkannya aplikasi tersebut adalah dikarenakan sering terjadi salah pencatatan karena pencatatan masih bersifat manual, lamanya proses transaksi simpan pinjam, dan lambatnya pembuatan laporan yang diserahkan kepada pihak manajemen.

Berdasarkan latar belakang tersebut, buatlah sebuah aplikasi simpan pinjam yang mudah digunakan dengan ketentuan sebagai berikut:

Aplikasi dapat melakukan transaksi simpan pinjam (penyimpanan, pengambilan, peminjaman dan pembayaran)

Apliasi yang dibuat dapat mempercepat proses transaksi simpan pinjam.

Aplikasi yang dibuat dapat menampilkan laporan berupa transaksi simpan pinjam (penyimpanan, pengambilan, peminjaman, dan pembayaran).

Kasus 5 : Aplikasi Penjualan dan Pembelian Kaset

Sebuah toko kaset yang sedang berkembang, membutuhkan aplikasi untuk meningkatkan kinerja karyawan di bagian penjualan dan pembelian. Dengan aplikasi baru diharapkan dapat mengurangi permasalahan yang timbul dari poses pencatatan yang digunakan toko tersebut. Masalah yang timbul diantaranya adalah: kesalahan penulisan, penumpukan dokumen, lambatnya pembuatan laporan.

Aplikasi baru tersebut berisi

Aplikasi dapat melakukan proses transaksi penjualan dan pembelian yang akan berpengaruh pada stock kaset yang ada.

Aplikasi dapat mempermudah pembuatan laporan penjualan dan pembelian serta stock barang yang ada.

Aplikasi dapat menampilkan laporan barang dengan ketentuan tertentu (yang paling laris, banyaknya stock, dan lain-lain).

Kasus 6 : Sistem Persediaan Barang pada Instalasi Farmasi di Rumah Sakit "A"

Beberapa permasalahan yang timbul terhadap persediaan barang di rumah sakit "A" adalah:

1. Pada fungsi Perencanaan :

   Seringkali terjadi pemesanan barang yang masih memilki cukup banyak persediaan di lokasi penyimpanan.

   Sistem saat ini tidak mampu menyediakan informasi mengenai jumlah persediaan semua barang dalam waktu yang cepat. Sehingga pada saat dirasa perlu untuk memesan barang, bagian perencanaan tidak memiliki dasar yang tepat dalam menentukan barang – barang yang akan dipesan. Dari contoh daftar persediaan barang di atas diasumsikan bahwa yang diperlukan untuk segera dipesan adalah "Obat B" dan "Obat C" namun karena keterbatasan informasi yang didapat, "Obat A" dan "Obat D" pun masuk dalam daftar pemesanan barang.

   Banyak persediaan di Instalasi Farmasi memiliki masa kadaluwarsa. Pemesanan barang yang tidak tepat seperti dijelaskan di atas menimbulkan resiko terjadinya penyimpanan barang-barang yang sudah melewati masa kadaluwarsa sebelum persediaan itu terjual. Dengan demikian, RS harus melakukan Pemusnahan terhadap barang – barang tersebut (menimbulkan kerugian bagi RS).

2. Pada Fungsi Penyimpanan
   - Kerap terjadi *human error* dalam pencatatan keluar masuknya barang. Sehingga menimbulkan selisih antara jumlah barang sebenarnya dengan jumlah barang yang dicatat.
   - Adanya kesulitan dalam mendapatkan informasi mengenai jumlah total semua persediaan saat ini. Hal ini terjadi karena lokasi penyimpanan ada di beberapa tempat, antara lain Gudang Farmasi A, Gudang Farmasi B, Farmasi A, Farmasi B.

3. Pada Fungsi Penjualan

   Jika jumlah pengunjung Rumah Sakit meningkat, maka terjadi antrian yang panjang di Instalasi Farmasi. Banyaknya antrian ini mengakibatkan pengunjung segan untuk membeli obat di Intalasi Farmasi Rumah Sakit. Hal ini pun cukup merugikan karena

terkesan bahwa pelayanan Rumah Sakit buruk. Dan hal ini dapat mengakibatkan berkurangnya jumlah pengunjung.

1.4 TUGAS PENDAHULUAN I

Dikumpulkan pada pertemuan ke satu

Tugas individu :

Jelaskan tentang Rekayasa Perangkat Lunak (**min 2 halaman**)

Tugas Kelompok ;

1 Buat satu tim yang terdiri dari minimum 5 orang dan maksimum 6 orang.
2 Pilih salah satu kasus proyek perangkat lunak di atas (point 1.3) dengan syarat tidak boleh sama dengan tim yang lain.(Opsional)

1.5 TUGAS RUMAH I

Dikumpulkan pada pertemuan ke dua (Individu)

1 Jelaskan manfaat proposal dalam proyek perangkat lunak.
2 Buat Proposal proyek perangkat lunak (kasus bebas kecuali kasus yang telah dipilih oleh tim) **boleh di print, kasus tidak boleh sama.**

1.6 LATIHAN PRAKTIKUM I

Latihan Praktikum 1 (pertemuan ke satu)

1 Buat organisasi tim pengembang proyek perangkat lunak dan pilih salah satu kasus diatas (opsional)
2 Buat Proposal proyek perangkat lunak sesuai kasus yang telah dipilih atau ditetapkan oleh tim/instruktur

Catatan :

- Kasus dan tim pengembang proyek perangkat lunak bisa dipilih sendiri oleh kelompok praktikan atau ditetapkan oleh instruktur.
- Dalam satu kelas tidak ada kasus proyek perangkat lunak yang sama

MODUL II

ANALISIS SISTEM

(5 kali pertemuan)

2.1 TUJUAN

Tujuan modul ini, adalah:

- Memperkenalkan penggunaan perangkat lunak bahasa pemrograman untuk mengimplementasikan struktur data (tipe data abstrak, list berkait linear).
- Praktikan dapat menerapkan teknik komunikasi dan pemodelan sistem untuk memahami sistem yang dibangun.
- Praktikan dapat mendokumentasikan *requirement* (SRS) dan mampu menerapkan pemodelan fungsional.
- Praktikan dapat melakukan *review* dokumen analisis.

2.2 TEORI

2.2.1 Penggalian *Requirement*

Requirement adalah (1) Kondisi atau kemampuan yang dibutuhkan oleh user untuk mengatasi suatu masalah atau untuk mencapai tujuan tertentu;(2) Kondisi atau kemampuan yang harus dimiliki oleh sistem untuk memenuhi sebuah kontrak, standar, spesifikasi atau dokumen resmi yang ditetapkan.

Analisis *Requirement* adalah (1) Proses untuk menemukan, menyempurnakan, memodelkan dan menspesifikasikan;(2) pekerjaan rekayasa perangkat lunak yang menjembatani jurang antara sistem *engineer* dan perancang *software*. Atau boleh dikatakan bahwa Analisis *Requirement* adalah tahap interaksi intensif antara analis sistem dengan komunitas pemakai sistem (*end-user*), dimana team pengembangan sistem menunjukkan keahliannya untuk mendapatkan tanggapan dan kepercayaan pemakai, sehingga mendapat partisipasi yang baik. Merupakan pekerjaan sulit untuk

mendapatkan kesepakatan (skeptical) pemakai tentang kebutuhan mereka dari sebuah sistem informasi, karena mungkin pemakai mengalami kegagalan sistem informasi sebelumnya.

Untuk itu perlunya metode untuk menggali kebutuhan user. Metode tersebut antarala lain:

- *interviews,*
- *questionnaires,*
- *observation,*
- *procedure analysis*, dan
- *document survey*

Penjelasan dari masing-masing metode adalah sebagai berikut:

1) Tanya Jawab (*interviews*)

a) Bagaimana metode itu digunakan

Pemilihan potential interviewees.

Membuat perjanjian terhadap potential interviewees.

Menyiapkan struktur pertanyaan yang lengkap dan jelas.

Memilih person yang diinterview secara pribadi dan merekamnya.

b) Target dari metode

Kunci pribadi dalam proses DFD.

Kadangkala melibatkan orang luar, seperti pelanggan atau vendors.

c) Keuntungan metode

Pewawancara dapat mengukur respon melalui pertanyaan dan menyesuaikannya sesuai situasi yang terjadi.

Baik untuk permasalahan yang tidak terstruktur, seperti mengapa anda berpikir hal ini dapat terjadi ?.

Menunjukkan kesan interviewer secara pribadi.

Memunculkan respons yang tinggi sejak penyusunan pertemuan.

d) Kerugian metode

Membutuhkan waktu dan biaya yang tidak sedikit.

Membutuhkan pelatihan dan pengalaman khusus dari pewawancara.

Sulit membandingkan laporan wawancara karena subyektivitas alamiah.

e) Kapan metode tersebut baik digunakan

Mendapatkan penjelasan atau pandangan dari personel kunci.

Test kredibilitas dari interviewees.

Mencari interview yang unsureness atau contradictions.

Memantapkan kredibilitas team.

Beberapa faktor penting dalam interview yang baik, yaitu *objektives, audience, format, weighting* dan *combining responses, and documentation*.

2) Kuisoner (*questionnaires*)

a) Bagaimana metode itu digunakan

Mendisain dengan menggunakan standar kuesioner.

Kuesioner dikirimkan ke lingkungan kerja end-users.

Struktur respon diringkas dalam statistik distribusi.

b) Target dari metode

Semua end-user dengan wawasannya akan dilibatkan dalam proses solusi pemecahan sistem.

End-user dihubungkan dengan proses pemakaian simbol-simbol dalam DFD.

c) Keuntungan metode

Murah dan cepat dari pada interviews.

Tidak membutuhkan investigator yang terlatih (hanya satu ahli yang dibutuhkan untuk mendesain kuesioner untuk end-user yang terpilih.

Mudah untuk mensintesis hasil sejak pembuatan kuesioner.

Dengan mudah dapat meminimalkan biaya untuk semua end-user.

d) Kerugian metode

Tidak dapat membuat pertanyaan yang spesifik bagi end-user.

Analis melibatkan kesan sehingga tidak dapat menampakkan pribadi end-user.

Tanggapan yang rendah karena tidak adanya dorongan yang kuat untuk mengembalikan kuesioner.

Tidak dapat menyesuaikan pertanyaan ke end-user secara spesifik.

e) Kapan metode tersebut baik digunakan

Pertanyaannya sederhana, dan tidak memiliki arti mendua.

Membutuhkan wawasan yang luas dari end-user.

Bila memiliki sedikit waktu dan biaya.

3) Observasi (*observation*)

a) Bagaimana metode itu digunakan

Secara pribadi seorang analis mengunjungi lokasi pengamatan.

Analis merekam kejadian dalam lokasi pengamatan, termasuk volumen dan pengolahan lembar kerja.

b) Target dari metode

Lokasi proses secara geografis ditunjukkan dalam DFD (Data Flow Diagram)

c) Keuntungan metode

Mendapatkan fakta records daripada pendapat (opinion).

Tidak membutuhkan konstruksi pertanyaan.

Tidak menganggu atau menyembunyikan sesuatu (end-users tidak mengetahui bahwa mereka sedang diamati).

Analis tidak bergantung pada penjelasan lisan dari end-users.

d) Kerugian metode

Jika terlihat, analis mungkin mengubah operasi (end-user merasa diamati).

Dalam jangka panjang, fakta yang diperoleh dalam satu observasi mungkin tidak tepat (representative) dalam kondisi harian atau mingguan.

Membutuhkan pengalaman dan kehlian khusus dari analis.

e) Kapan metode tersebut baik digunakan

Membutuhkan gambaran kuantitatif seperti waktu, volume dan sebagainya.

Kecurigaan bahwa end-user mengatakan suatu kejadian yang sebenarnya tidak terjadi (dibuat-buat).

Tip praktis dalam melakukan observasi :

a) Jangan mengamati dalam waktu yang lama. Terdapat dua alasan, yaitu : dengan waktu yang lama akan mengacau operasi yang sedang diamati, dan akan membiaskan permasalahan yang sebenarnya.
b) Buat catatan yang ringkas.
c) Sebelum observasi, beritahukan kepada supervisor dan pemakai yang terlibat tentang apa yang akan dikerjakan dan mengapa dikerjakan, sehingga akan mengurangi gangguan.
d) Gunakan checklist yang singkat tentang informasi yang dibutuhkan bersama.
e) Jangan melakukan observasi tanpa rencana.

4) Prosedur Analisis (*procedure analysis*)

a) Bagaimana metode itu digunakan

Dengan prosedur operasi dapat mempelajari dan mengidentifikasikan aliran dokumen kunci melalui sistem informasi, yaitu dengan data flow diagram (DFD).

Setiap aliran dokumen kunci menjelaskan prosedur operasi sistem.

Melalui observasi, analis mempelajari kenyataan daripada mendeskripsikan volume distribusi (tinggi, rendah, sedang) dan apa yang selanjutnya dikerjakan terhadap salinan dari dokumen aslinya.

b) Target dari metode

Dokumen utama dalam DFD (Data Flow Diagram)

Proses dalam DFD.

c) Keuntungan metode

Evaluasi prosedur dapat dikerjakan dengan campur tangan (interferences) yang minimal dan tidak mempengaruhi operasi pemakai.

Prosedur aliran dapat dapat menjadi sebuah struktur checklist untuk melakukan observasi.

d) Kerugian metode

Prosedure mungkin tidak lengkap dan tidak -up to date lagi.

Mempelajari bagan aliran dokumen membutuhkan waktu dan keahlian analis.

e) Kapan metode tersebut baik digunakan

Memutuskan apakah masalah kegagalan sistem dapat membantu perancangan yang baik.

Tim analis tidak secara total familiar dengan aliran dokumen.

Mendeskripsikan aliran dokumen yang menganggu kerjanya fungsi.

5) Pengamatan Dokumen (*document survey*)

a) Bagaimana metode itu digunakan.

Mengidentifikasikan dokumen utama dan laporan (*physical data flow diagram*).

Mengumpulkan salinan dokumen aktual dan laporan.

Setiap dokumen atau laporan, digunakan untuk record data, meliputi field (ukuran dan tipe), frekuensi penggunaan dan struktur kodingnya (*coding structure*).

b) Target dari metode.

Aliran data kunci ditunjukkan dalam data flow diagram (DFD).

c) Keuntungan metode.

Meminimalkan interupsi dari fungsi operasionalnya.

Permulaan elemen kamus data.

Seringkali, dapat mempertimbangkan modifikasi major procedural.

d) Kerugian metode.

Membutuhkan waktu yang cukup (terdapat organisasi bisnis yang mengalami kebanjiran dokumen dan laporan).

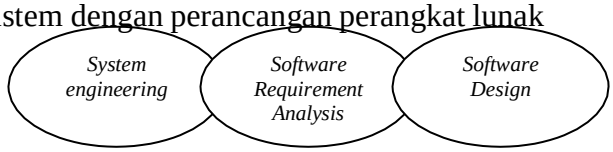
e) Kapan metode tersebut baik digunakan.

Harus dikerjakan jika sebuah sistem akan didesain (selama kegiatan analisis, dalam memperjelas desain sistem yang baru dan analisis dokumen dapat membantu untuk menentukan tugas perancangan selanjutnya).

2.2.1.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan bertujuan untuk menggali kebutuhan-kebutuhan (*requirement*) yang harus dipenuhi oleh *software* yang akan dibuat untuk memperoleh fungsi dan kelakuan *software.*

Pada fase analisis ini, *user* kadang akan memformulasi ulang fungsi dan unjuk kerja yang diinginkan dengan lebih detail. Sedangkan bagi pengembang/analis akan bertindak sebagai "interogator, konsultan dan pemecahan masalah". Analisis kebutuhan ini adalah pekerjaan rekayasa perangkat lunak yang menjembatani antara level spesifikasi sistem dengan perancangan perangkat lunak



Analisis kebutuhan dibagi menjadi lima bagian, yaitu : (1) Pengenalan Masalah;(2) Evaluasi Sintesa; (3) Pemodelan; (4) spesifikasi; (5) Peninjauan Ulang/(*review*).

Prinsip-prinsip Analisis :

a. Domain Fungsi dari masalah harus ditunjukan.
b. Fungsi yang akan dilaksanakan oleh perangkat lunak harus terdefinisi.
c. Kesalahan dari perangkat lunak harus ditunjukan.
d. Model yang menggambarkan informasi, fungsi dan kelakuan harus dibagi-bagi menjadi beberapa lapis tingkatan untuk mendapatkan informasi yang lebih detail.
e. Proses analisis hendaknya memindahkan dari informasi yang penting ke arah penerapan yang rinci.

2.2.1.2 Prinsip Analisis

a. Daerah informasi dari masalah harus dimengerti.
b. Fungsi yang harus dilakukan oleh *software* harus terdefinisi.
c. Kelakuan *software* (sebagai akibat dari pengaruh luar) harus terwakili atau ditampilkan.

d. Membuat pemodelan yang menggambarkan informasi, fungsi dan kelakuan *software.*
e. Proses analisis harus mengubah informasi yang diperoleh menjadi informasi yang siap dirancang untuk implementasi.

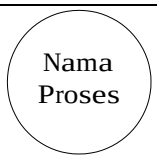
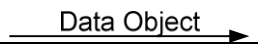
2.2.2 Pemodelan Sistem

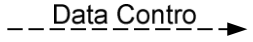
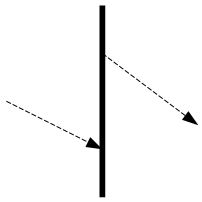
2.2.2.1 Pemodelan dalam Analisis Terstruktur

Pemodelan sistem disini menggunakan metode *Data Flow Oriented* dengan *tool Data Flow Diagram* (DFD). DFD adalah (1) Suatu teknik penggambaran atau pemodelan menggunakan notasi-notasi grafis yang menunjukan aliran informasi dan perubahannya yang diterapkan sebagai perubahan atau perpindahan data dari masukan(*input*) menjadi keluaran(*output*); (2) Peralatan pemodelan yang mengijinkan kita menggambarkan sistem sebagai suatu jaringan proses-proses yang dihubungkan dengan baris data dan tangki penyimpanan data.

A. Notasi DFD

Tabel 2.1 Notasi DFD

No.	Gambar Notasi	Nama Notasi	Keterangan
1		*External Entity* (Kata Benda)	Entitas luar yang terlibat dengan sistem, dapat berupa sumber asal datangnya data (*source*) atau yang menerima data (*sink*)
2		Proses (Kata Kerja)	Suatu kegiatan yang dilakukan oleh sistem, dapat berupa prosedur yang memanipulasi atau mengolah data
3		*Data Store* (Kata Benda)	Tempat penyimpanan data untuk direferensi atau diolah lagi lebih lanjut
4		*Data Flow* (Kata Benda)	Aliran data yang menunjukan transportasi data/informasi.

No.	Gambar Notasi	Nama Notasi	Keterangan
5		*Control Flow* (Kata Benda)	Aliran kontrol data atau kejadian, dapat berupa nilai boolean atau nilai diskrit
6		*Control Bar*	Merupakan acuan untuk CSPEC (*Control Specification*) yang menjelaskan kebiasaan sistem dan mendefinisikan bagaimana proses diaktifkan sebagai konsekuensi dari suatu kejadian

B. Aturan-aturan pembuatan DFD

- Suatu proses harus menghasilkan *output*.
- *Store* hanya muncul di DFD, tidak boleh di *Data Context Diagram* (DCD).
- Aliran data (*data flow*) tidak boleh dari *store* (penyimpanan data) ke *store* lain.
- Jumlah aliran data (data masuk dan data keluar) harus konsisten.
- Hindari proses yang hanya mempunyai aliran data masuk atau data keluar.
- Hati-hati terhadap *store* yang hanya mempunyai aliran data masuk atau data keluar.
- Hati-hati dengan aliran data yang tidak diberi nama, beri nama aliran data dengan kata benda.
- Hindari proses yang tidak diberi nama, beri nama proses dengan kalimat sederhana yang menunjukan apa yang akan diproses dan sebaiknya selain nama, suatu proses juga diberi nomor.
- **Sebaiknya jumlah proses pada satu bidang kertas = 7 ± 2**

C. Panduan untuk Membuat DFD

- Pilih nama yang bermakna untuk proses, *store* dan aliran data
- Berikan penomoran untuk setiap proses yang ada
- Hindari penggambaran DFD yang rumit (dapat diatasi dengan menggunakan pelevelan)
- Gambar beberapa kali untuk mendapatkan hasil yang enak untuk dilihat
- Yakini bahwa DFD konsisten secara internal

D. Tentang Pelevelan DFD

1) Bagaimana anda tahu berapa banyak level yang harus dimiliki DFD?
   - Tergantung pada jumlah prosesnya.
   - Berapa banyak proses yang optimum pada satu level DFD.
2) Apakah setiap bagian sistem harus dirinci sama banyak?
   - Tergantung proses yang dibutuhkan rincian lebih lanjut.
3) Bagaimana anda menunjukan level-level DFD ini ke user?
   - Tergantung user yang dibutuhkan DFD.
4) Bagaimana menggambarkan store pada berbagai level.
   - Tergantung simbol store yang akan digunakan.
   - Tergantung jumlah store yang digunakan oleh suatu proses.
5) Bagaimana menjamin level DFD konsisten dengan yang lain?
   - Jumlah data yang masuk dan keluar harus sama dengan DFD yang lain.

2.2.2.2 Dokumen yang Terlibat dalam Fase Analisis atau Spesifikasi

- IRS (*Interface Requirement Specification*) menjelaskan sistem secara global serta kaitannya dengan lingkungan sekitarnya.
- SRS (*Software Requirement Specification*) menjelaskan sistem secara detail termasuk fungsi-fungsi atau proses yang harus dipenuhi.

2.2.2.3 CSPEC (*Control Specification*)

Digunakan untuk mengindikasikan bagaimana perlakuan *software* ketika suatu kejadian atau sinyal kontrol mulai terjadi dan proses apakah yang diaktifkan sebagai konsekuensi terjadinya suatu kejadian. CSPEC selalu berhubungan dengan kontrol bar.

Tabel 2.2 Tabel CSPEC

Nama Kontrol	Proses yang dipengaruhi	……..
OK	…..	…..
Not_OK	…..	…..

2.2.2.4 PSPEC (*Process Specification*)

Deskripsi tentang apa yang terjadi pada proses level paling bawah, pada suatu diagram aliran data. Disebut juga dengan "MINISPEC" (*Miniatur Specification*) [De Marco]. Maksud dari spesifikasi ini adalah untuk mendefinisikan apa yang harus dilakukan untuk mengubah aliran masuk (*Input*) menjadi keluaran (*Output*).

a. Macam-macam alat untuk spesifikasi proses :

1. ***Structured English*** adalah bagian dari bahasa inggris dengan pembatasan pada kalimat yang dipakai dan cara dalam hal pemakaian kalimat disini dikenal juga sebagai kalimat PDL (*Program Design Language*) dan PSL (*Program Statement Language*). Kalimat dalam *structured english* bisa berupa persamaan aljabar.
2. ***Pre-Conditioning/Past Conditioning*** (kondisi awal dan kondisi akhir) digunankan untuk menggambarkan fungsi yang harus ditangani oleh sebuah proses tanpa bertanya tentang algoritma atau prosedur yang digunakan.
3. ***Narrative English*** adalah pemaparan spesifikasi dengan menggunakan kalimat bahasa inggris sebagaimana mestinya.

b. Pertimbangan penggunaan alat untuk proses spesifikasi

- Spesifikasi proses harus dalam bentuk yang bisa diperiksa oleh user dan analisis sistem.
- Spesifikasi proses harus dalam bentuk yang bisa membuat komunikasi yang efektif bagi orang yang terlibat.

2.2.2.5 Kamus Data (*Data Dictionary*)

Kamus data adalah daftar terorganisir dari semua elemen data yang ada pada suatu sistem dengan definisi yang jelas/tepat, sehingga user dan analisis sistem bisa mendapat kesepahaman dari *input, output* dan komponen dari penyimpanan dan kalkulasi "*intermediate*" yang ada.

Kamus Data dibuat berdasarkan aliran data yang ada di DFD (Data Flow Diagram). Aliran data di DFD sifatnya adalah global, hanya ditunjukkan nama arus

datanya saja. Keterangan lebih lanjut tentang struktur dari suatu arus data di DFD secara lebih terinci dapat dilihat di kamus data.

Kamus data mendefinisikan elemen data dengan fungsi sebagai berikut:

a) Menjelaskan arti aliran data dan penyimpanan dalam DFD.
b) Mendeskripsikan komposisi paket data yang bergerak melalui aliran, misalnya alamat diuraikan menjadi kota, kodepos, propinsi, dan negara.
c) Mendeskripsikan komposisi penyimpanan data.
d) Menspesifikasikan nilai dan satuan yang relevan bagi penyimpanan dan aliran.
e) Mendeskripsikan hubungan detil antara penyimpanan yang akan menjadi titik perhatian dalam entity relationship diagram.

Notasi kamus data yang digunakan dalam analisis sistem, yaitu

Tabel 2.3 Tabel Kamus Data

No.	Notasi	Keterangan
1	=	Terdiri dari/didefinisikan sebagai/maksudnya adalah
2	+	Dan
3	(…)	Opsional
4	{…}	Iterasi/pengulangan
5	[…]	Pemilihan dari beberapa alternatif
6	*…*	Komentar
7	@	Identifier dari *state*
8	|	Pemisahan pada pemilihan (atau)

2.2.2.6 Pemodelan Kelakuan (*Behaviour Model*)

Digambarkan dengan menggunakan CSPEC dalam dua cara :

a. ***Process Activation Table*** **(PAT)** adalah tabel yang menggambarkan kapan suatu proses diaktifkan dan oleh kontrol apa pengaktifan tersebut terjadi

b. ***State Trantition Diagram*** **(STD)** adalah merupakan spesifikasi sekuensial (terurut) dari kelakuan suatu sistem. STD digambarkan dengan notasi tanda kotak yang menunjukan keadaan sistem dan panah yang menunjukan transisi keadaan.

2.2.2.7 SRS (*Software Requirement Specification*)

SRS adalah hasil akhir dari proses analisis. Fungsi dan kinerja yang harus dipenuhi sebagai bagian dari rekayasa sistem ditetapkan dengan deskripsi yang lengkap, baik deskripsi fungsional dan *behavioral* .

Format/kerangka SRS, adalah sebagai berikut :

Tabel 2.4 Kerangka SRS

Kerangka Dokumen	Keterangan
Abstraksi	Abstraksi/Rangkuman dokumen (SRS)
Daftar Isi Daftar Gambar Daftar Tabel	Daftar Isi, Daftar Gambar dan Daftar Tabel dalam Dokumen SRS
1 Pendahuluan	
1.1 Tujuan	Tujuan penyusunan dokumen SRS dan menentukan siapa yang akan menggunakan SRS ini
1.2 Ruang Lingkup Perangkat Lunak	Memberikan batasan pembuatan SRS
1.3 Daftar Definisi dan Singkatan	Menjelaskan definisi dan singkatan dalam SRS
1.4 Referensi	Referensi/dokumen/bahan acuan yang digunakan
1.5 Overview SRS	Menjelaskan isi dan organisasi dari SRS secara singkat
2 Deskripsi Umum	
2.1 Perspektif Produk	Menjelaskan : Identifikasi perangkat lunak Kemampuan perangkat lunak Tujuan dan keuntungan perangkat lunak
2.2 Fungsi-Fungsi produk	Menjelaskan kesimpulan dari fungsi yang umum yang akan dilakukan oleh perangkat lunak
2.3 Karakteristik Pengguna	Menjelaskan karakteristik umum dari *user* perangkat lunak
2.4 Batasan Umum	Menjelaskan item-item yang akan membatasi pilihan pengembangan perangkat lunak
2.5 Asumsi dan Ketergantungan	Menjelaskan factor-faktor yang dapat mengakibatkan perubahan pada perangkat lunak
3 Kebutuhan Spesifik	Berisi semua kebutuhan perangkat lunak hingga tingkat yang paling rinci
3.1 Kebutuhan Antarmuka	
3.1.1 Antarmuka Pengguna	Menjelaskan format layar, menu, tata letak dst
3.1.2 Antarmuka *Hardware*	Menjelaskan perangkat keras yang akan digunakan
3.1.3 Antarmuka *Software*	Menjelaskan perangkat lunak yang akan digunakan (basis data, sistem operasi dll)

Kerangka Dokumen	Keterangan
3.1.4 Antarmuka Komunikasi	Menjelaskan perangkat komunikasi yang akan digunakan (protokol jaringan lokal)
3.2 Kebutuhan Fungsional	Tujuan dan prioritas proyek pengembangan perangkat lunak
3.2.1 DCD (*Data Context Diagram*)	Menjelaskan *Context Diagram* perangkat lunak
3.2.2 DFD (*Data Flow Diagram*) Level 1	Menjelaskan DFD level 1 perangkat lunak (*data-in*, *data-out*, PSPEC, CSPEC, *Data Dictionary*)
3.2.3 DFD (*Data Flow Diagram*) Level 2	Menjelaskan DFD level 2 perangkat lunak (*data-in*, *data-out*, PSPEC, CSPEC, *Data Dictionary*)
3.2.4 DFD (*Data Flow Diagram*) Level n	Menjelaskan DFD level n perangkat lunak (*data-in*, *data-out*, PSPEC, CSPEC, *Data Dictionary*)
3.3 Unjuk Kerja	Menjelaskan unjuk kerja perangkat lunak
3.4 Batasan Perancangan	Menjelaskan batasan perancangan yang akan dihasilkan oleh standar lain, keterbatasan *hardware* dan lain-lain
3.5 Atribut	
3.5.1 Ketersediaan (*Availability*)	Menjelaskan faktor untuk menjamin tingkat ketersediaan seluruh sistem (*recovery* dll)
3.5.2 Keamanan (*Security*)	Menjelaskan faktor untuk menjamin tingkat keamanan perangkat lunak
3.5.3 Keterpeliharaan (*Maintainability*)	Menjelaskan atribut yang berhubungan dengan kemudahan perawatan dari perangkat lunak
3.5.4 Dst	
3.6 Kebutuhan Lain – Lain	
3.6.1 Basis Data	Menjelaskan kebutuhan logis untuk setiap informasi yang disimpan dalam basis data
3.6.2 Sistem Operasi	Menjelaskan kebutuhan sistem operasi dari perangkat lunak
3.6.3 Adaptasi Tempat	Menjelaskan kebutuhan tempat dan adaptasinya dari perangkat lunak
Lampiran	Berisi penjelasan tambahan pada laporan ini

2.2.2.8 *Software Specification Review*

Review diikuti oleh konsumen dan pengembang dengan tujuan untuk mendapatkan kesepahaman terhadap *software* yang akan dikembangkan. Panduan melakukan *review* yang lebih detail, adalah sebagai berikut :

- Perhatikan kata/*term* yang bermakna kabur (misalnya kadang, sebagian dll)

- Jika ada suatu daftar tapi tidak lengkap, yakinkan bahwa semua item terpenuhi
- Hati-hati dengan kalimat yang membingungkan
- Yakinlah dengan jangkauan keadaan
- Jika suatu *term* telah didefinisikan dengan jelas disuatu tempat, sebaikanya menggunakan pengacuan terhadap *term* tersebut tidak perlu mendefinisikan ulang.

2.3 TUGAS-TUGAS PENDAHULUAN

2.3.1 Tugas Pendahuluan II

Dikumpulkan pada pertemuan ke dua

1. Jelaskan tentang Analisis Sistem, *requirement* dan analisis *requirement*!
2. Jelaskan kegunaan proses analisis dalam tahapan rekayasa perangkat lunak!
3. Menurut Anda metode penggalian requirement yang mana yang palign baik? Jelaskan!

2.3.2 Tugas Pendahuluan III

Dikumpulkan pada pertemuan ke tiga

1. Jelaskan tentang DCD dan DFD!
2. Jelaskan perbedaan physical DFD dan logical DFD!
3. Buat DCD dan DFD untuk Kasus berikut ini!

Sistem Transmisi

Sistem ini digunakan untuk mengirimkan pesan antara sistem pemancar dan penerima. Pemancar akan mengirimkan pesan dalam bentuk teks dan identifikasinya. Sistem ini menghitung parity dari pesan dan mengirimkannya kembali ke pemancar. Sistem ini menggunakan identifikasi penerima untuk menemukan format pesan. Dengan berdasarkan pada informasi ini sistem akan memecah pesan kedalam paket dan mengirim paket ini ke penerima. Jenis-jenis format pesan yang mungkin sudah tersimpan dalam basis data.

2.3.3 Tugas Pendahuluan IV

Dikumpulkan pada pertemuan ke empat

1. Jelaskan tentang CPSEC, PSPEC dan Kamus Data!
2. Buat CSPEC, PSPEC dan Kamus data untuk Kasus pada tugas pendahuluan 2 !

2.3.4 Tugas Pendahuluan V

Dikumpulkan pada pertemuan ke lima

1. Jelaskan kegunaan proses *review* dalam perancangan suatu perangkat lunak!
2. Jelaskan tentang SRS!
3. Jelaskan tentang *Software Specification Review!*

2.3.5 Tugas Pendahuluan VI

Dikumpulkan pada pertemuan ke enam

1. Jelaskan tentang SDD!
2. Jelaskan tentang metode perancangan!
3. Jelaskan kegunaan proses perancangan dalam tahapan rekayasa perangkat lunak!

2.4 LATIHAN - LATIHAN PRAKTIKUM

2.4.1 Latihan Praktikum II (Pertemuan ke dua)

1. Analisis kasus proyek perangkat lunak pada pertemuan ke satu!
2. Dokumentasikan hasil analisis tersebut dalam dokumen SRS (Bab I dan Bab II)!

2.4.2 Latihan Praktikum III (Pertemuan ke dua, ke tiga, dan ke empat)

1. Dari hasil analisis pada pertemuan ke dua, buat pemodelan sistemnya dengan menggunakan metode *Data Flow Oriented* dengan *tools Data Flow Diagram*!
2. Buat CSPEC, PSEPC dan Kamus Datanya !

3. Dokumentasikan hasil analisis tersebut dalam dokumen SRS (Bab III point 3.1 dan 3.2)!

2.4.3 Latihan Praktikum IV (Pertemuan ke lima)

1. Dari hasil pemodelan sistem pada pertemuan sebelumnya, analisis kebutuhan antar muka untuk aplikasi, unjuk kerja, hambatan perancangan, basis data dan seterusnya!
2. Dokumentasikan dalam dokumen SRS (Bab III point 3.3 sampai point 3.6)!
3. Review dokumen SRS!

MODUL III
PERANCANGAN SISTEM

(4 kali pertemuan)

3.1 TUJUAN

Tujuan modul ini, adalah:

- Praktikan dapat menerapkan teknik dalam tahapan perancangan perangkat lunak.
- Praktikan dapat mendokumentasikan hasil perancangan (SDD)
- Praktikan dapat melakukan *review* dokumen rancangan.

3.2 TEORI

3.2.1 Perancangan Awal (*Preliminary Design*)

3.2.1.1 Perancangan Data

Perancangan data adalah aktifitas pertama dari empat aktifitas perancangan selama proses rekayasa perangkat lunak. Pengaruh arsitektur data pada struktur program dan kompleksitas prosedural akan berpengaruh juga terhadap kualitas *software*.

Aktifitas utama selama perancangan data adalah menyeleksi representasi logis dari objek data (struktur data) yang diidentifikasikan selama pendefinisian kebutuhan dan fase spesifikasi. Selain itu melakukan identifikasi modul program yang harus beroperasi secara langsung pada struktur data tersebut.

Menurut Wesserman dapat digunakan prinsip-prinsip spesifikasi perancangan data, yaitu :

a. Prinsip analisis sistematik yang diterapkan pada fungsi dan kelakuan sistem juga harus diterapkan kepada data.
b. Semua struktur data dan operasi yang akan dikerjakan terhadap data tersebut harus diidentifikasi.

c. Kamus data harus ada dan digunakan untuk mendefinisikan desain data dan program.
d. Pendefinisian perancangan data level terendah harus terus digunakan sampai akhir proses perancangan.
e. Representasi struktur data harus dikenal oleh modul-modul yang menggunakan data tersebut.
f. Sebuah *library* (kepustakaan) dari struktur data yang berguna dan operasi-operasi yang akan diterapkan pada struktur data tersebut harus dikembangkan.
g. Perancangan *software* dan bahasa pemrograman harus mendukung spesifikasi dan realisasi tipe data abstraknya.

3.2.1.2 Perancangan Arsitektural

Perancangan arsitektural bertujuan untuk mengembangkan sebuah struktur program yang modular dan menunjukan hubungan antar modul. Perancangan ini menyatukan struktur program dan struktur data, menentukan interface yang membaca data bisa mengalir disemua program.

3.2.1.2.1 Proses Perancangan Arsitektural

Perancangan yang berorientasi aliran data adalah metode perancangan arsitektural yang mengijinkan transisi dari model analisis ke sebuah deskripsi perancangan dari struktur program. Transisi dari aliran informasi ke struktur dicapai sebagai bagian dari proses lima tahapan sebagai berikut :

1. Tipe aliran informasi telah ditetapkan.
2. Batasan aliran telah ditunjuk.
3. DFD dipetakan ke seluruh program.
4. Hirarki kendali didefiniskan oleh "*factoring*".
5. Struktur gabungan diperbaiki dengan menggunakan ukuran-ukuran dan usaha-usaha perancangan.

A. **Aliran Pengubahan (*Transform Flow*)**

Pada model sistem yang paling dasar informasi harus masuk dan keluar dari *software* dalam bentuk sebuah "*external word*". Data tersebut harus diubah menjadi bentuk internal yang diidentifikasikan sebagai aliran datang (*incoming flow*). Data yang datang ini melewati sebuah pusat transformasi (*transform flow*) dan akan bergerak sepanjang jalur yang akan mengarah keluar dari *software*, data ini disebut aliran keluar (*outgoing flow*). *Transform flow* terjadi saat sebagian dari suatu DFD memiliki karakteristik ini.

B. **Aliran Transaksi (*Transaction Flow)***

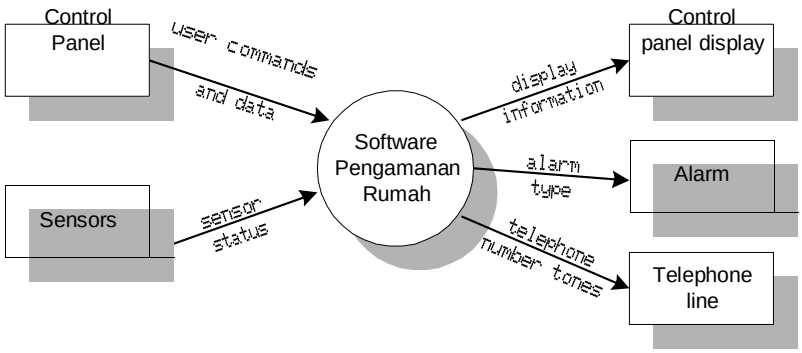
Aliran transaksi adalah aliran data yang memicu aliran data yang lain pada beberapa jalur. Aliran ini ditandai oleh adanya data yang bergerak sepanjang jalur kedatangan yang mengubah "*external word*" menjadi sebuah transaksi.

3.2.1.2.2 Pemetaan Pengubahan (Transform Mapping)

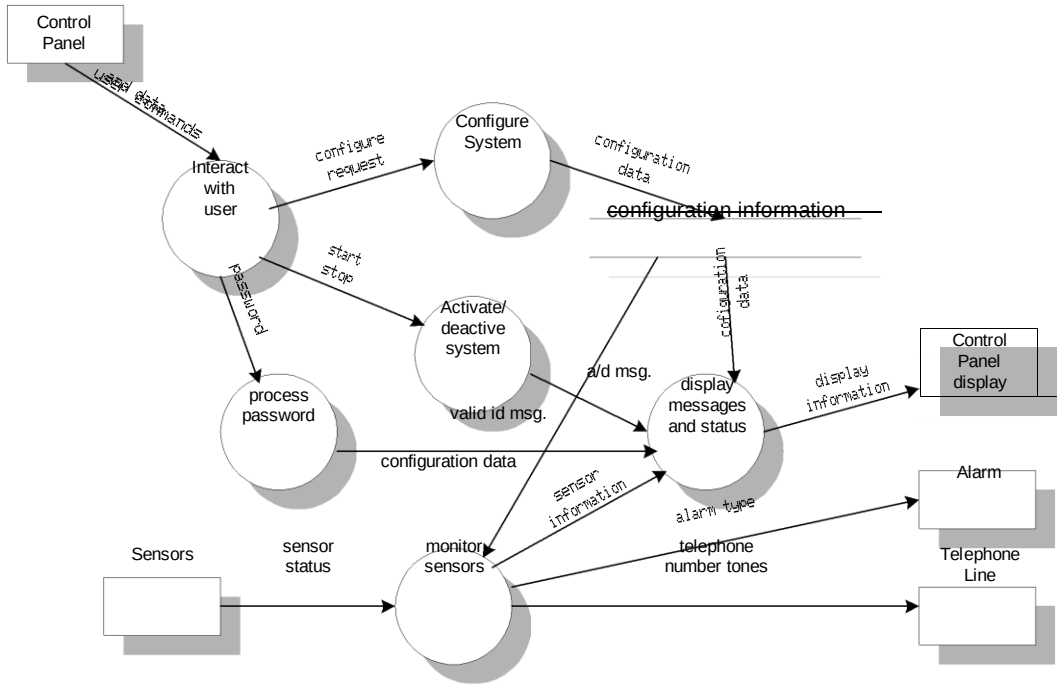
Transform Mapping adalah kumpulan langkah-langkah perancangan yang mengijinkan sebuah DFD dengan karakteristik pengubahan dipetakan kedalam "template" struktur program.

Contoh Kasus : *Software* Pengamanan Rumah

Sistem ini memonitor keadaan lingkungan dan bereaksi terhadap perubahan yang ditemukan. Produk ini juga berinteraksi dengan user melalui panel masukan yang diketikan dan sebuah display alpha numerik.



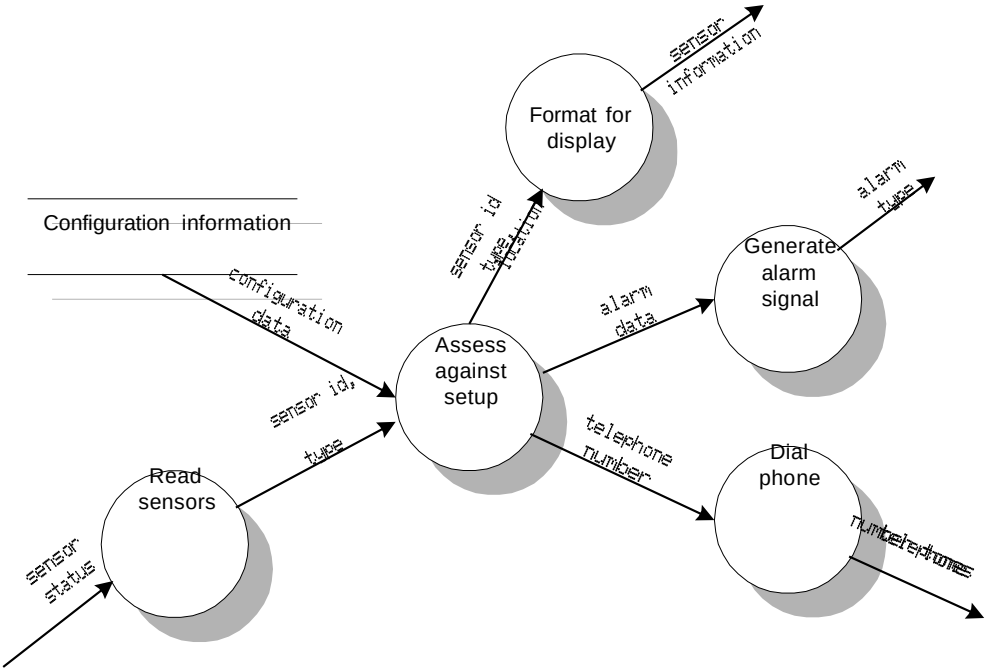
Gambar 3.1 *Data Context Diagram Software* Pengamanan Rumah



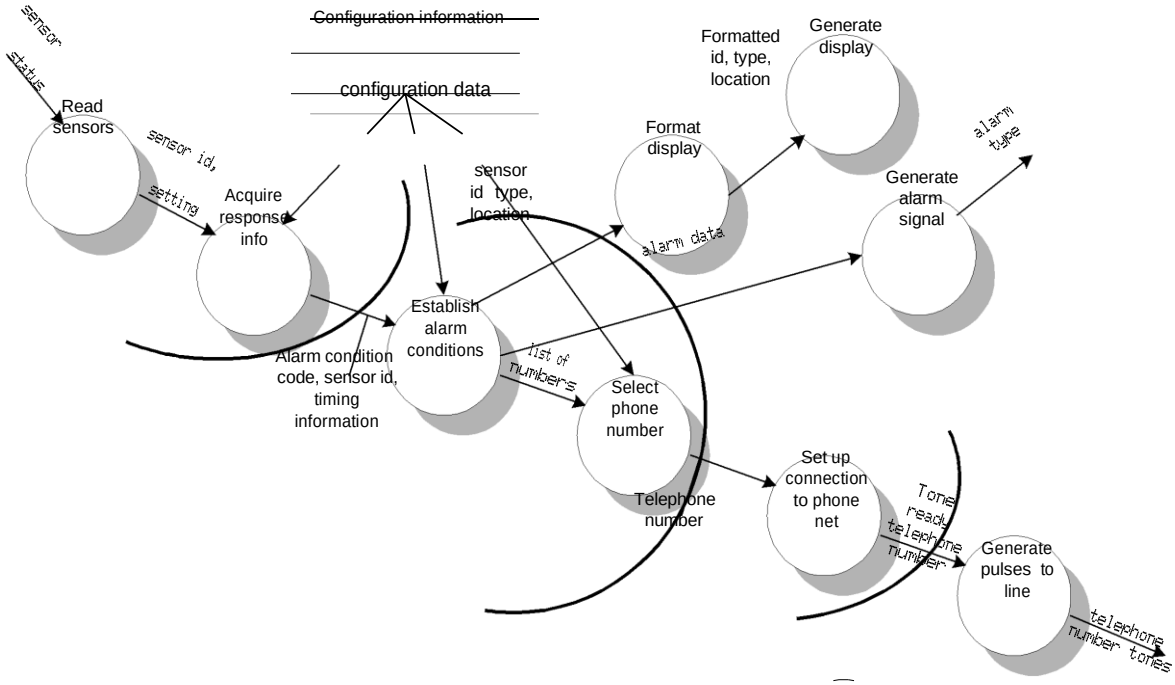
Gambar 3.2 *Data Flow Diagram* level 1 *Software* Pengamanan Rumah

Langkah-langkah perancangan :

1. Tinjau model dasar dari sistem (lihat DCD [gambar 3.1.] dan informasi pendukung lainnya)
2. Tinjau dan perbaiki diagram aliran data untuk *software* (lihat DFD level 2 [gambar 3.3.] dan DFD level 3 [gambar 3.4])

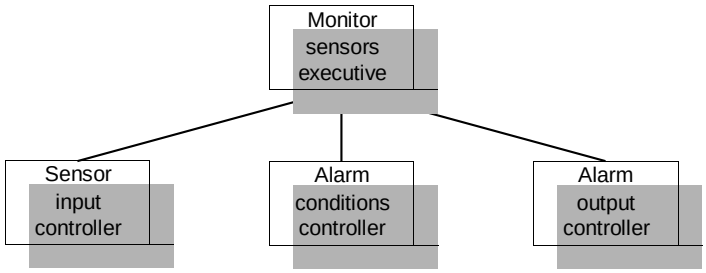


Gambar 3.3 *Data Flow Diagram* level 2 (Proses Monitor Sensor)



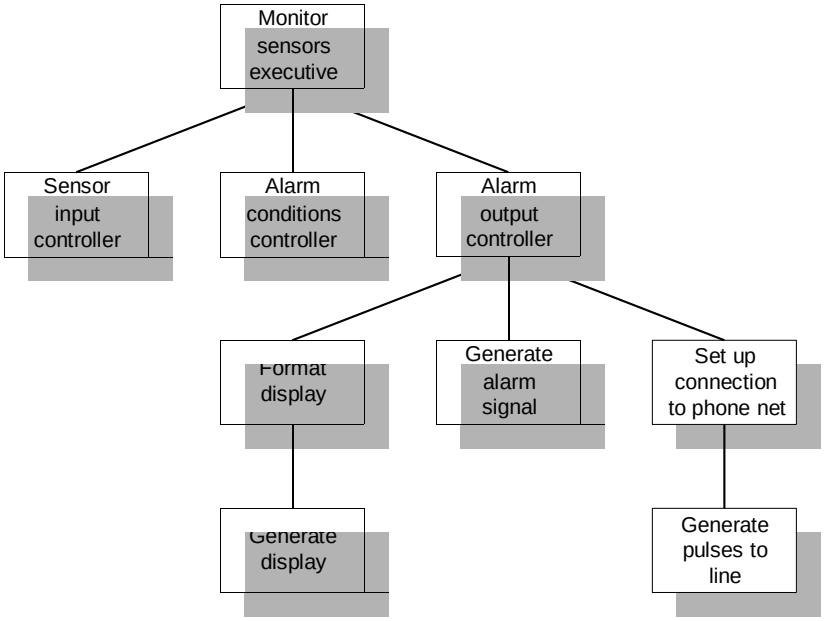
Gambar 3.4 *Data Flow Diagram* level 3 (Proses Monitor Sensor)

3. Tentukan apakah DFD memiliki karakteristik aliran pengubah bentuk atau aliran transaksi (evaluasi DFD level 3 [gambar 3.4.])
4. Isolasikan pusat pengubahan dengan menspesifikasikan batasan aliran kedatangan dan aliran keluar [gambar 3.4.]
5. Lakukan *factoring* tingkat awal (*First Level Factoring*) [gambar 3.5.]
   Struktur program menunjukan sebuah distribusi "*top-down*" dari kontrol. Hasil *factoring* pada struktur program :
   - pada modul level puncak menunjukan pengambilan keputusan
   - pada modul level menengah menunjukan beberapa pengedalian dan sejumlah kerja tertentu
   - pada modul level bawah menunjukan pekerjaan input, perhintungan proses dan output

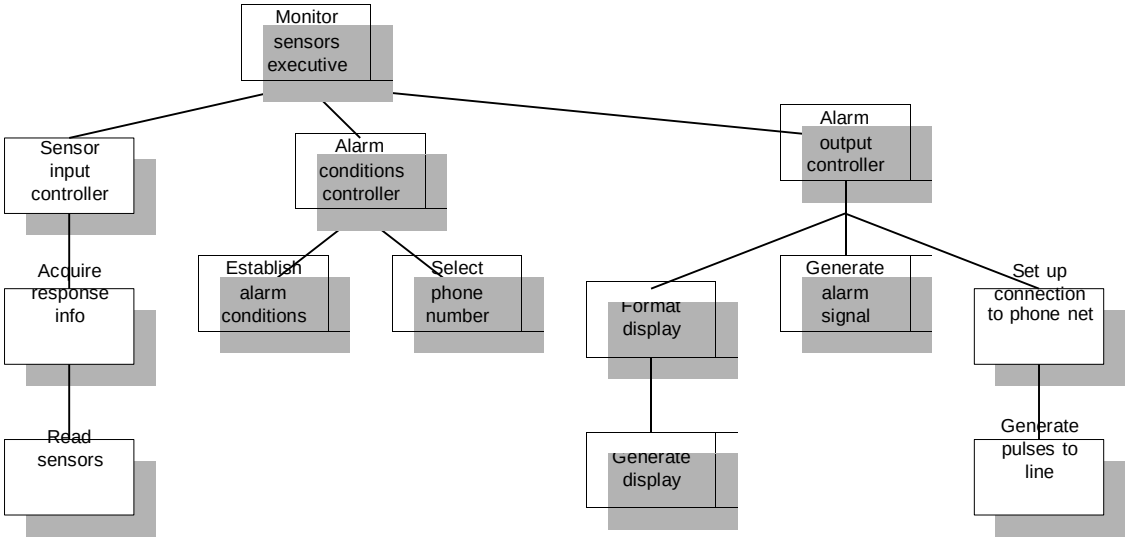


Gambar 3.5 *Factoring* level awal/pertama untuk Proses Monitor Sensor

6. Lakukan *factoring* level kedua [gambar 3.6]

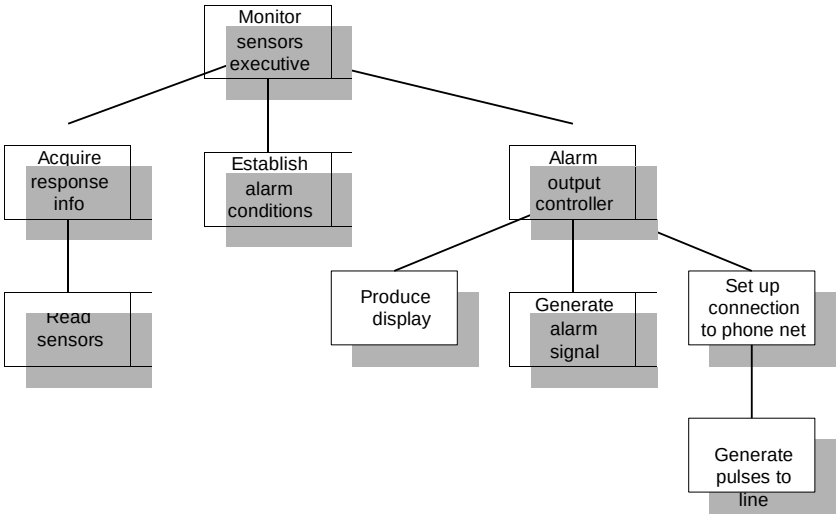


Gambar 3.6 *Factoring* level kedua Proses Monitor Sensor



Gambar 3.7 Iterasi pertama dari Struktur Program Monitor Sensor

7. Perbaiki iterasi pertama dari struktur program [gambar 3.7.] dengan menggunakan langkah-langkah perancangan untuk memperbaiki kualitas *software*. Iterasi pertama dari struktur program selalu diperbaiki dengan menerapkan konsep independensi modul. Modul-modul mungkin dipecah atau digabungkan untuk mendapatkan kohesi yang baik, kopling yang minimal dan yang terpenting adalah struktur tersebut bisa diimplementasikan tanpa kesulitan, bisa diuji tanpa membingungkan dan dipelihara tanpa menyusahkan. [gambar 3.8.]

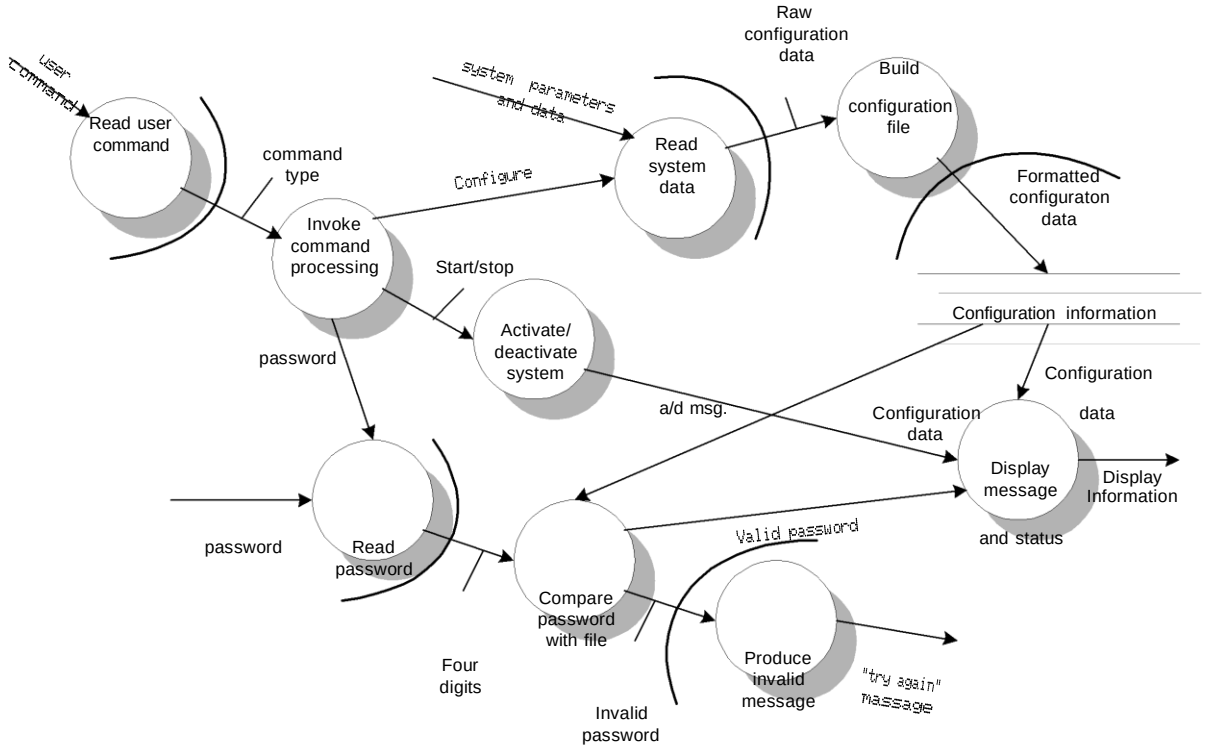


Gambar 3.8 Perbaikan dari Struktur Program Monitor Sensor

3.2.1.2.3 Pemetaan Transaksi (*Transaction Mapping*)

Langkah-langkah perancangan :

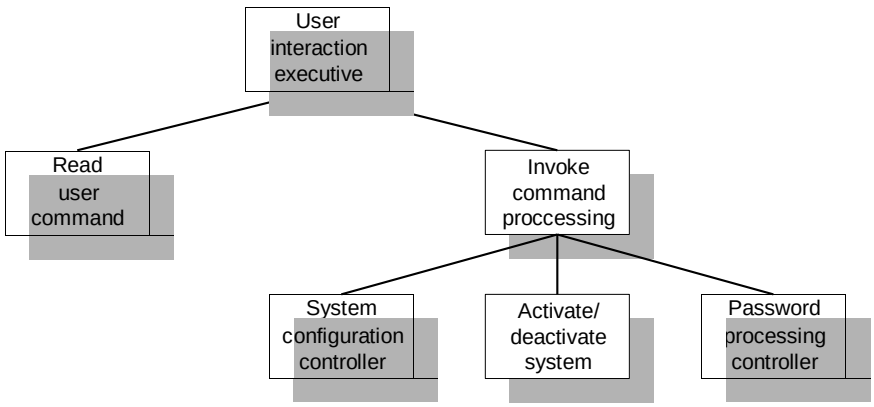
1. Tinjau model dasar dari sistem (lihat DCD [gambar 3.1] dan informasi pendukung lainnya)
2. Tinjau dan perbaiki diagram aliran data untuk *software* (lihat DFD level 2 [gambar 3.9]



Gambar 3.9 Data Flow Diagram level 2 (Proses *Interaction*)

3. Tentukan apakah DFD memiliki karakteristik aliran pengubah bentuk atau aliran transaksi (evaluasi DFD level 2 [gambar 3.9])
4. Identifikasikan pusat transaksi
5. Petakan DFD pada struktur program untuk transaksi

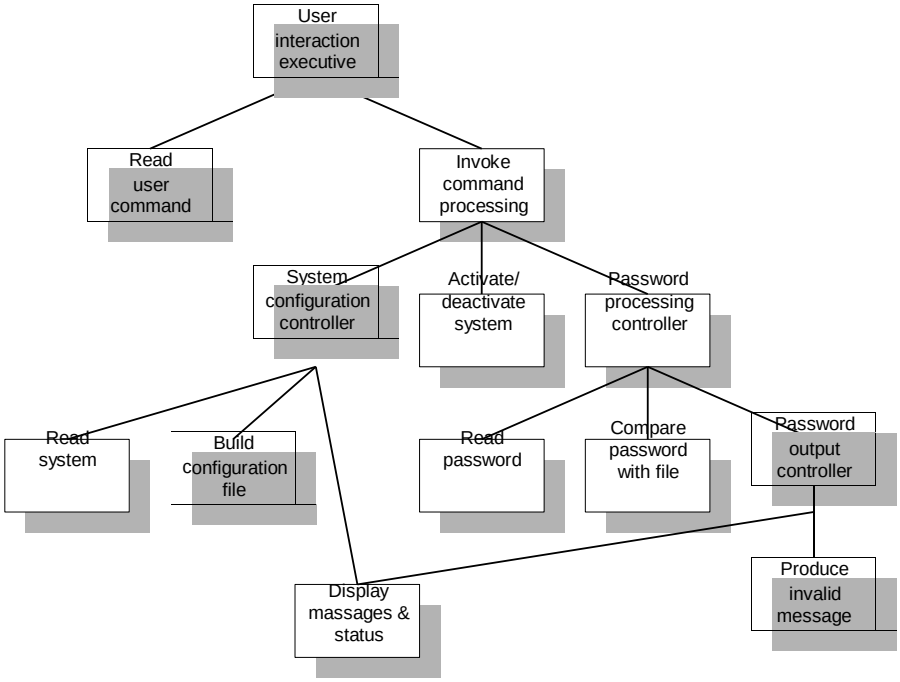
   Aliran transaksi dipetakan pada sebuah program yang mengandung cabang kedatangan dan pelepasan. Struktur cabang kedatangan dikembangkan dengan cara yang sama dengan pemetaan pengubahan. Hasil *factoring* tingkat awal (*First Level Factoring*) [gambar 3.10]



Gambar 3.10 *Factoring* level awal/pertama untuk Proses *Interaction*

6. Lakukan perbaikan terhadap struktur transaksi dan struktur dari masing-masing jalur

   Hasil *factoring* tingkat awal (*First Level Factoring*) [gambar 3.11]



Gambar 3.11 Iterasi Pertama dari Struktur Program untuk Proses *Interaction*

7. Perbaiki iterasi pertama dari struktur program dengan menggunakan langkah-langkah perancangan untuk memperbaiki kualitas *software*.

3.2.2 Perancangan Rinci (*Detail Design*)

3.2.2.1 Perancangan Antarmuka

Panduan Perancangan Antarmuka

1. Interaksi umum
   - Harus Konsisten.
   - Memberikan umpan balik yang bermakna.
   - Memberikan pertanyaan untuk verifikasi terhadap setiap aksi yang destruktif.
   - Memberikan kesempatan untuk mengulang atau membatalkan perintah sebelumnya.
   - Kurangi informasi yang harus diingat oleh user dari satu aksi ke aksi berikutnya.

- Cari efisiensi dalam dialog, pergerakan dan pemikiran.
- Gunakan kata-kata yang sederhana atau kalimat yang pendek untuk nama perintah.
- Memberikan fasilitas "*help*".
- "memaafkan" kesalahan.

2. Menampilkan informasi
   - Tampilan hanya menampilkan informasi yang relevan terhadap konteks saat itu.
   - User jangan "diberi" data yang banyak, gunakan format presentasi yang memudahkan untuk mendapatkan informasi.
   - Gunakan label yang konsisten, singkatan standar dan warna yang bisa diprediksi.
   - Berikan kemudahan bagi user untuk menjaga kontak visual (tidak kehilangan arah).
   - Gunakan pesan error yang bermakna.
   - Gunakan "window" untuk memisahkan informasi yang berbeda-beda.
   - Gunakan huruf besar/kecil, indentasi, pengelompokan untuk membantu kemudahan dalam pemahaman.
   - Gunakan *display* yang analog untuk menampilkan informasi.
   - Perhatikan kemampuan layar tampilan.
3. Masukan data
   - Minimalkan jumlah aksi yang berkaitan dengan pemasukan data oleh user.
   - Jaga konsistensi antara tampilan informasi dan masukan data.
   - Ijinkan user untuk memberikan masukan yang dikehendakinya.
   - Biarkan user mengendalikan aliran interaksi.
   - Biarkan fasilitas help untuk membantu semua aksi pemasukan data.

3.2.2.2 Perancangan Prosedur

Idealnya spesifikasi prosedur diperlukan untuk mendefinisikan algoritma yang rinci yang dinyatakan dalam bahasa sehari-hari. Perancangan prosedur harus memberikan prosedur yang rinci dan tidak bermakna ganda. Dengan menggunakan

bahasa sehari-hari, kita bisa menuliskan sekelompok langkah-langkah prosedural dalam berbagai cara.

1. Pemrograman terstruktur

   Djikstra mengusulkan penggunaan sekumpulan konstruksi logika dimana suatu program bisa berbentuk, terdiri dari :

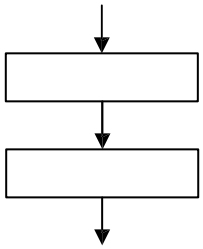
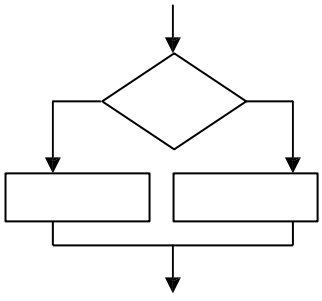
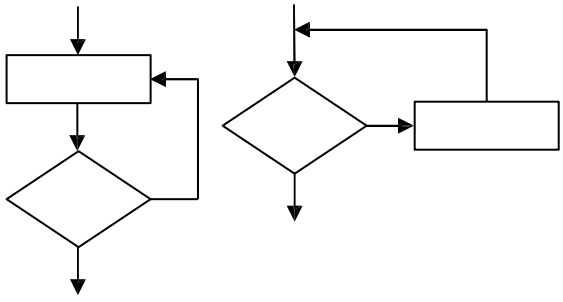
   - **Sequence/urutan** adalah implementasi dari tahapan pemrosesan yang merupakan bagian yang esensial dalam spesifikasi dari setiap algoritma.
   - **Kondisi**, memberikan fasilitas untuk proses pemilihan berdasarkan beberapa kejadian logika.
   - **Pengulangan**, memberikan kesempatan suatu perintah dijalankan berulang-ulang.

   Ketiga konstruksi di atas merupakan dasar dari pemrograman terstruktur yang merupakan teknik perancangan prosedur yang penting.

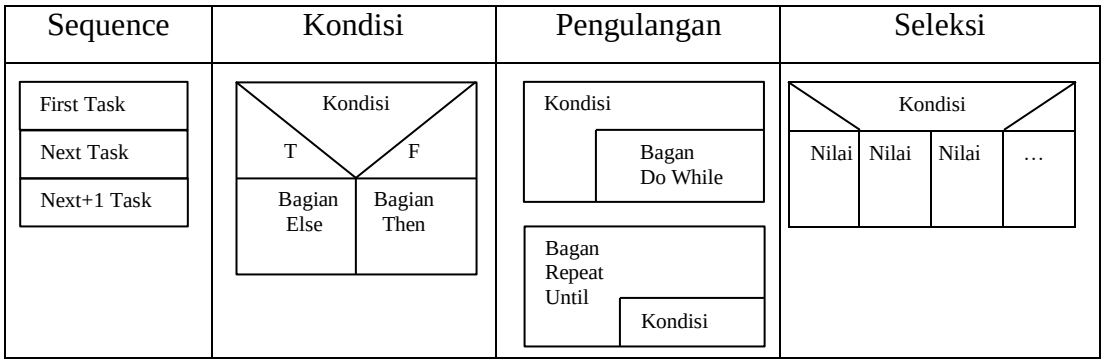
2. Notasi Perancangan dengan menggunakan grafik

   Notasi *Flowchart* adalah salah satu notasi untuk perancangan prosedur yang banyak digunakan.

Tabel 3.1 Notasi Flowchart

Sequence	Kondisi	Pengulangan
		

Tabel 3.2 Notasi box Diagram/Nassi-Shneiderman Chart (N-S Chart)/Chapin Chart



3. Bahasa perancangan program (*Program Design Language*/PDL)

   PDL disebut juga *Structured English or Pseudocode*, sepintas Bahasa perancangan program ini mirip dengan bahasa pemrograman modern.

4. Notasi perancangan berbentuk tabel

   Dalam beberapa *software* aplikasi, sebuah modul diinginkan untuk mengevaluasi kombinasi komplek dari kondisi dan harus memilih aksi yang tepat berdasarkan kondisi tersebut. Tabel keputusan (*Decision Table*) memberikan notasi yang memindahkan aksi dan kondisi ke bentuk tabel.

   Tabel keputusan ini dibagi menjadi empat bagian, yaitu :

   - Bagian kiri atas mengandung daftar dari semua kondisi.
   - Bagian kiri bawah berisi aksi yang muncul berdasarkan kombinasi dari kondisi.
   - Bagian kanan adalah matriks yang menunjukan kombinasi aksi dan kondisi yang berkaitan, yang akan terjadi untuk kombinasi tertentu.

   untuk itu setiap kolom dari matrik bisa diinterprestasikan sebagai hukum (*rule*) pemrosesan.

   Langkah-langkah untuk membuat tabel keputusan :

   - Daftarkanlah semua aksi yang berkaitan dengan prosedur khusus.
   - Buat daftar kondisi atau pengambilan keputusan selama eksekusi prosedur berlangsung.
   - Hubungkan sekumpulan kondisi tertentu dengan aksi tertentu, buanglah kombinasi yang tidak mungkin.

- Tentukan hukum/aturan dengan menunjukan aksi apa yang terjadi untuk sekumpulan kodisi tertentu.

3.2.3 SDD (*Software Design Document*)

SDD adalah hasil akhir dari proses perancangan. SDD merupakan penjelasan hasil proses perancangan yang termasuk didalamnya perbaikan hasil perancangan tersebut untuk merepresentasikan perangkat lunak yang sedang dibangun.

Kerangka SDD adalah sebagai berikut :

Tabel 3.3 Kerangka SDD

Kerangka Dokumen	Keterangan
Abstraksi	Abstraksi/Rangkuman dokumen (SDD)
Daftar Isi Daftar Gambar Daftar Tabel	Daftar Isi, Daftar Gambar dan Daftar Tabel dalam SDD
1 Pendahuluan	
1.1 Tujuan SDD	Tujuan penyusunan dokumen SDD dan menentukan siapa yang akan menggunakan SDD ini
1.2 Ruang Lingkup SDD	Memberikan batasan pembuatan SDD
1.3 Daftar Definisi dan Singkatan	Menjelaskan definisi dan singkatan dalam SDD
1.4 Referensi	Referensi/dokumen/bahan acuan yang digunakan
1.5 Overview SDD	Menjelaskan isi dan organisasi dari SDD secara singkat
2 Rancangan Lingkungan Implementasi	Menjelaskan *hardware*, *software*, basis data dst yang akan digunakan untuk implementasi
3 Perancangan Data	
3.1 Daftar Tabel	Menjelaskan tabel-tabel yang akan digunakan oleh perangkat keras (nama table, *primary key*, dan deskripsi table)
3.2 Conceptual Data Model (CDM)	Menjelaskan CDM atau E-R Diagram
3.3 Dekomposisi Fungsional Modul	Menjelaskan untuk suatu modul/proses tabel dengan data yang digunakan sebagai masukan dan keluaran untuk modul/proses tersebut
3.4 Tabel A	Menjelaskan struktur tabel A (Identifikasi tabel, deskripsi isi, jenis tabel, volume, laju, *primary key*, dst)

Kerangka Dokumen	Keterangan
3.5 Tabel B,… dst	Menjelaskan struktur tabel B,..dst (Identifikasi tabel, deskripsi isi, jenis tabel, volume, laju, *primary key*, dst)
4 Perancangan Arsitektural	
4.1 Kajian data dan Aliran data	Mengindikasikan bagaimana arsitektur program didapatkan dari model analisis
4.2 Struktur Program yang Diperoleh	Menjelaskan bagan struktur (representasi dari struktur program) yang digunakan untuk menunjukan hirarki modul tersebut
5 Perancangan Antarmuka	
5.1 Spesifikasi Antarmuka	
5.1.1 Spesifikasi Layar Utama	Menjelaskan spesifikasi layar utama (jenis, bentuk, ciri layar dst)
5.1.2 Spesifikasi Objek-Objek pada Layar	Menjelaskan objek-objek yang ada di layar
5.1.3 Spesifikasi Layar Pesan	Menjelaskan spesifikasi layar pesan (untuk suatu kasus akan ditampilkan pesan di layar dengan bentuk tertentu)
5.1.4 Spesifikasi Report	Menjelaskan spesifikasi laporan(jenis, bentuk, cirri laporan dst)
5.2 Aturan Perancangan Antarmuka	Menjelaskan aturan perancangan antarmuka
5.3 Perancangan Antar Muka Eksternal	
5.3.1 Antarmuka untuk data eksternal	Menjelaskan representasi antarmuka untuk data eksternal
5.3.2 Antarmuka untuk sistem dan peralatan eksternal	Menjelaskan representasi antarmuka untuk sistem dan peralatan eksternal
5.4 Perancangan Antarmuka Internal	Menjelaskan representasi antarmuka internal
6 Perancangan Prosedural	
6.1 Naratif Pemrosesan	Menjelaskan fungsi prosedural dari suatu modul
6.2 Deskripsi Antarmuka	Menjelaskan perancangan antarmuka
6.3 Deskripsi Perancangan Bahasa (atau lainnya)	Menjelaskan bahasa (atau lainnya) yang digunakan pada perancangan
6.4 Modul-Modul yang digunakan	Menjelaskan mdoul-modul yang digunakan
6.5 Struktur Data Internal	Menjelaskan struktur data internal

Kerangka Dokumen	Keterangan
6.6 Keterangan/Larangan/Batasan	Menjelaskan keterangan/larangan/batasan perancangan
Lampiran	Berisi penjelasan tambahan pada laporan ini

3.3 TUGAS-TUGAS PENDAHULUAN

3.3.1 Tugas Pendahuluan VII

Dikumpulkan pada pertemuan ke tujuh

1. Jelaskan tentang Perancangan Arsitektur dan perancangan data!
2. Jelaskan perbedaan perancangan arsitektur dengan perancangan data.
3. Apa manfaat perancangan arsitektur dan perancangan data pada tahap perancangan!

3.3.2 Tugas Pendahuluan VIII

Dikumpulkan pada pertemuan ke delapan

1. Jelaskan tentang perancangan prosedur dan perancangan pengujian!
2. Jelaskan perbedaan perancangan prosedur dengan perancangan pengujian!
3. Apa manfaat perancangan prosedur dan perancangan pengujian pada tahap perancangan!

3.3.3 Tugas Pendahuluan IX

Dikumpulkan pada pertemuan ke sembilan

1. Jelaskan Tentang *Critically Design Review* (CDR)!
2. Apa manfaat CDR dalam pengembangan perangkat lunak!

3.4 LATIHAN - LATIHAN PRAKTIKUM

3.4.1 Latihan Praktikum IV (Pertemuan ke enam)

1. Dari hasil perancangan pada pertemuan ke lima dan ke enam, buat rancangan datanya!

2. Dokumentasikan hasil perancangan data tersebut dalam dokumen SDD (Bab 1 dan Bab 2)!

3.4.2 Latihan Praktikum V (Pertemuan ke tujuh dan ke delapan)

1. Berdasarkan pemodelan menggunakan *tools Data Flow* Diagram yang telah dibuat pada pertemuan sebelumnya, lakukan proses pada perancangan arsitektural!
2. Dokumentasikan dalam dokumen SDD (Bab 3)!

3.4.3 Latihan Praktikum VI (Pertemuan ke delapan dan ke sembilan)

1. Berdasarkan hasil pada perancangan arsitektur pada pertemuan sebelumnya, buat rancangan antar muka dan prosedur!
2. Dokumentasikan dalam dokumen SDD (Bab 4 sampai Bab 8)!
3. Review Dokumen SDD!

MODUL IV
IMPLEMENTASI

(2 kali pertemuan)

4.1 TUJUAN

Tujuan modul ini, adalah:

- Praktikan dapat mengimplementasikan hasil perancangannya dengan menggunakan salah satu bahasa pemrograman.
- Praktikan dapat membuat program sesuai dengan struktur program yang telah dirancang pada proses perancangan.
- Praktikan dapat membuat dokumentasi hasil implementasi

4.2 TEORI

Setelah tahap analisis sistem dan perancangan selesai dilaksanakan, dilanjutkan dengan mengimplementasikan hasil perancangan sistem tersebut kedalam aplikasi dengan menggunakan bahasa pemrograman yang telah ditetapkan sebelumnya.

Selanjutnya dibuat suatu dokumen yang berisi implementasi dari aplikasi yang telah dibangun yang disebut dengan Buku Manual/Pedoman. Dokumen ini digunakan sebagai acuan informasi untuk *user* yang akan menggunakan aplikasi/sistem ini. Selain itu buku manual/pedoman ini dapat digunakan sebagai bahan analisis untuk perbaikan/pengembangan sistem lebih lanjut.

Format/kerangka dokumen implementasi, adalah sebagai berikut:

Tabel 4.1 Kerangka Dokumen Implementasi

Kerangka Dokumen	Keterangan
Abstraksi	Abstraksi/Rangkuman dokumen Implementasi
Daftar Isi Daftar Gambar Daftar Tabel	Daftar Isi, Daftar Gambar dan Daftar Tabel dalam Dokumen Implementasi

Kerangka Dokumen	Keterangan
1 Pendahuluan	
1.1 Tujuan	Tujuan penyusunan dokumen Implementasi dan menentukan siapa yang akan menggunakan implementasi ini
1.2 Ruang Lingkup	Memberikan batasan pembuatan dokumen implementasi
1.3 Deskripsi Umum Perangkat Lunak	Memberikan penjelasan umum dari perangkat lunak yang diimplementasikan
1.4 Daftar Definisi dan Singkatan	
1.5 Referensi	Referensi/dokumen/bahan acuan yang digunakan
1.6 Overview	Menjelaskan isi dan organisasi dari dokumen implementasi secara singkat
2 Rancangan Lingkungan Implementasi	
2.1 Lingkungan Perangkat Keras	Menjelaskan *perangkat keras* yang digunakan untuk implementasi
2.2 Lingkungan Perangkat Lunak	Menjelaskan perangkat lunak yang digunakan untuk implementasi
3 Struktur Program	Menjelaskan modul-modul/struktur menu/urutan proses dari perangkat lunak yang dibangun
4 Penggunaan Aplikasi	Menjelaskan cara/langkah bagaiman menggunakan perangkat lunak yang dibangun
5 Implementasi dan Pemeliharaan	Menjelaskan mengimplementasikan (menginstall), menyebarkan, pelatihan dan pemeliharaan perangkat lunak yang dibangun
Lampiran	Menjelaskan informasi tambahan dari dokumen ini

Contoh dokumen implementasi :

1. Lingkungan Implementasi

   Lingkungan implementasi meliputi lingkungan perangkat keras (*Hardware*) dan lingkungan perangkat lunak (*Software*)

   1.1 Lingkungan Perangkat Keras (*Hardware*)

   Perangkat keras yang digunakan pada saat pengimplementasian Sistem Informasi Penggajian adalah sebagai berikut :

   a. Mikroprosessor : Pentium 100 MHz
   b. Memori : 32 MB

c. Monitor : SVGA 800x600
d. Media Penyimpanan : Harddisk 2.1 GB

1.2 Lingkungan Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan pada saat pengimplementasian Sistem Informasi Penggajian adalah sebagai berikut :

a. Sistem Operasi : Microsoft windows 98
b. Program : Visual Basic 6.0
c. Perangkat tambahan : Microsoft Access, Visual Data Manager

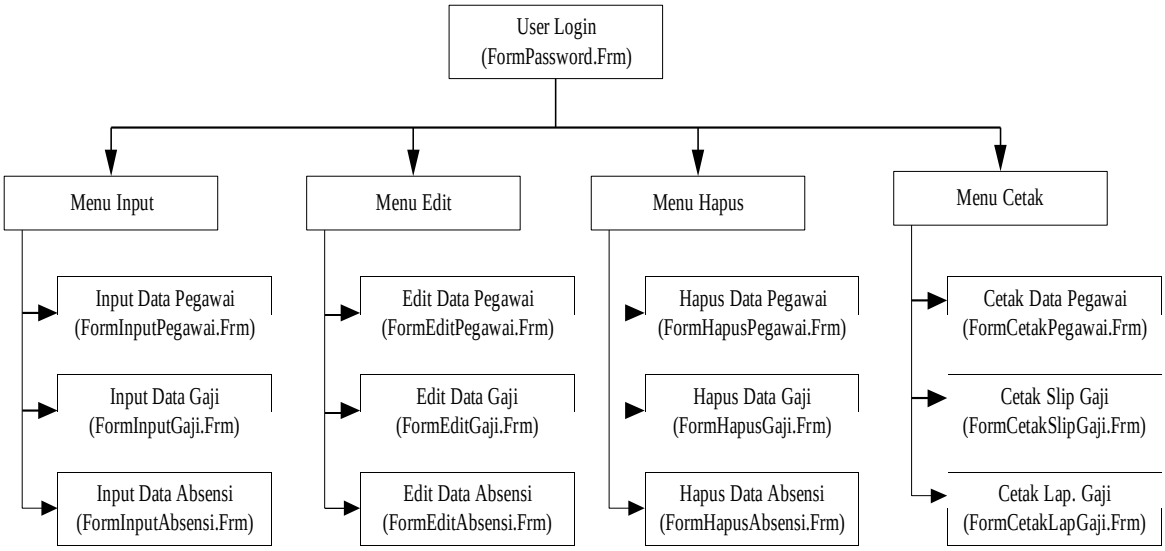
2. Struktur Program

Dalam Visual Basic, pembuatan sebuah program aplikasi harus dikerjakan dalam sebuah proyek yang berisi kumpulan file-file.

Sebuah proyek dapat terdiri satu file proyek (.vbp), satu file form untuk setiap form (.frm), satu file data binary untuk setiap form (.frx), satu file untuk setiap modul class (.cls), satu file untuk setiap modul standar (.bas) atau file resource tunggal (.res).

Selain modul dan file, beberapa tipe komponen dapat juga dimasukan ke dalam proyek seperti satu/lebih file yang terdiri kontrol ActiveX (.ocx), Insertable Object seperti objek worksheet Excel, Reference yang ditambahkan ke eksternal ActiveX, ActiveX Designer untuk merancang class pada objek,serta Standar Control seperti CommandButton.

Struktur Program Sistem Informasi Penggajian adalah sebagai berikut :



Gambar 4.1 Struktur Program Sistem Informasi Penggajian

Struktur program ini bisa juga disebut sebagai struktur menu. Struktur ini menjelaskan urutan proses berdasarkan menu dalam aplikasi yang dibangun

3. Penjelasan Struktur Program

   Penjelasan struktur program Sistem Informasi Penggajian (berdasarkan gambar di atas)

Tabel 4.2 Tabel penjelasan struktur program

Nama Menu / Form (*.frm)	**Keterangan**
FormPassword	Menampilkan *form user* login, digunakan untuk memasukan *user id* dan *password*-nya.
FormInputPegawai	Menampilkan form input data pegawai, digunakan untuk menambah data pegawai
FormInputGaji	Menampilkan *form* input gaji, digunakan untuk menambah data gaji.
FormInputData Absensi	Menampilkan form input data absensi, digunakan untuk menambah data absen para pegawai
…..	Dan seterusnya

4. Petunjuk Penggunaan Aplikasi

Berisi penjelasan penggunaan modul-modul yang terdapat dalam aplikasi yang dibangun. Contoh, Pertama kali aplikasi sistem informasi penggajian akan ditampilkan Form *User* login



Gambar 4.2 Contoh Form *Input Password*

- *User* memasukan *user* Id dan Password yang dimilikinya selanjutnya tombol OK diklik
- Jika Password salah akan ditampilkan pesan kesalahan



Gambar 4.3 Contoh Form Pesan Kesalahan Password

- Jika Password benar akan ditampilkan Form Menu Input.

……..Dan seterusnya.

Pada prinsipnya bagian ini menjelaskan penggunaan aplikasi yang dibangun, dari saat aplikasi dijalankan sampai aplikasi ini selsesai/ditutup.

4.3 TUGAS-TUGAS PENDAHULUAN X

Dikumpulkan pada pertemuan ke sepuluh

1. Jelaskan tentang tahapan implementasi
2. Jelaskan kegunaan implementasi dalam tahapan pengembangan perangkat lunak

4.4 LATIHAN PRAKTIKUM VII (Pertemuan ke sepuluh)

1. Berdasarkan hasil perancangan pada pertemuan sebelumnya lakukan implementasi hasil perancangan tersebut dalam bahasa pemrograman yang telah ditentukan.
2. Dokumentasikan dalam dokumen Implementasi

MODUL V
PENGUJIAN (*TESTING*)

(2 kali pertemuan)

5.1 TUJUAN

Tujuan modul ini, adalah:

- Praktikan dapat mempersiapkan tahapan pengujian.
- Praktikan dapat merancang kasus uji.
- Praktikan dapat melakukan pengujian *black box* dan pengujian *white box*.
- Praktikan dapat membuat dokumen pengujian

5.2 TEORI

5.2.1 Teknik Pengujian Perangkat Lunak

Pengujian *software* adalah elemen kritis dari jaminan kualitas *software* dan merupakan review akhir dari spesifikasi, perancangan dan pengkodean. Pada tahap awal dari pengembangan *software*, *engineer* berusaha untuk membangun *software* dari sebuah konsep abstrak menjadi implementasi nyata. Pada saat pengujian, *engineer* membuat serangkaian kasus uji yang bertujuan untuk "merusak" *software* yang telah dibuat.

5.2.2 Objektif Pengujian

Beberapa aturan objektif pengujian *software* menurut Glen Meyer :

- Pengujian adalah proses eksekusi sebuah program untuk menemukan "*error*"
- Kasus uji yang baik adalah sesuatu yang bisa mengungkapkan kemungkinan yang tinggi untuk menemukan *error-error* yang tidak ditemukan sebelumnya
- Pengujian yang sukses adalah sesuatu yang bisa mengungkapkan *error* tidak ditemukan sebelumnya

5.2.3 Prinsip Pengujian

Pada saat melakukan pengujian terhadap suatu *software* ada beberapa prinsip pengujian yang harus diperhatikan, diantaranya :

1. Semua pengujian harus terlacak kekebutuhan *user*
2. Pengujian harus direncanakan jauh sebelum pengujian dimulai
3. Pengujian dimulai dari kecil dan mengarah kepada yang besar
4. Pengujian yang sempurna adalah tidak mungkin
5. Agar pengujian berjalan dengan efektif, maka sebaiknya dilakukan oleh pihak ketiga yang netral

5.2.4 *Testability*

Kemampuan untuk diuji dari sebuah *software* (*software testability*) adalah kemudahan sebuah program komputer untuk diuji. Daftar pemeriksaan yang berkaitan dengan kemudahan untuk diuji dari sebuah *software*, diantaranya:

1. Operability
- "Lebih baik dia bekerja dan lebih efisien dia diuji"
- Sistem mempunyai sedikit "*bug*"
- Tidak ada *bug* yang menghalangi pengujian

2. Observability
- "apa yang anda lihat adalah apa yang anda uji"
- Bedakan *ooutput* yang dihasilkan oleh masing-masing *input*
- *State* sistem dan variabel terlihat selama eksekusi
- Status sistem yang lampau dan variabel
- Semua faktor yang mempengaruhi *ooutput* dapat dilihat
- *Output* yang salah mudah diidentifikasi
- *Error* internal otomatis dideteksi dengan mekanisme pengujian diri dan secara otomatis dilaporkan
- Kode program bisa diakses

3. Controlability
- Semua *ooutput* yang mungkin bisa dihasilkan melalui kombinasi beberapa *input*

- Semua kode dapat dieksekusi untuk beberapa kombinasi *input*
- *State* dari *software* dan *hardware* serta variabel dapat dikontrol secara langsung oleh *engineer*
- Format *input* dan *ooutput* konsisten dan terstruktur
- Pengujian bisa diidentifikasi, diotomatiskan dan diproduksi ulang

4. *Decomposability*
   - Sistem *software* dibangun dari modul-modul yang berdiri sendiri
   - Modul *software* diuji secara independen
5. *Simplicity*
   - *Functional simplicity*
   - *Structural simplicity*
   - *Code simplicity*
6. *Stability*
   - Perubahan terhadap *software* jarang, terkendali dan tidak membatalkan pengujian yang telah ada
   - *Software* bisa merecover dengan baik dari kegagalan
7. *Understandability*
   - Perancangan dipahami dengan baik
   - Ketergantungan antara internal, eksternal dan *shared* komponen dimengerti dengan baik
   - Perubahan terhadap perancangan diberitahukan kepada perancang lainnya
   - Dokumentasi teknik bisa diakses dengan cepat, diorganisasikan dengan baik, tertentu, rinci dan akurat.

5.2.5 Perancangan Kasus Uji

Perancangan untuk pengujian *software* adalah hal yang sama menantangnya dengan perancangan awal dari *software* itu sendiri. Setiap produk rekayasa perangkat lunak bisa diuji dalam dua cara:

1. Mengetahui fungsinya, sehingga pengujian dilakukan dengan mendemonstrasikan fungsi tersebut bisa berjalan dengan sempurna atau ada *error* (*Black box Testing*)
2. Mengetahui cara kerja internal dari produk tersebut (*White box Testing*)

5.2.6 Pengujian *White box*/*GlassBox*

Adalah sebuah metoda perancangan kasus uji yang menggunakan struktur kontrol dari perancangan prosedur. Pengguna metoda pengujian akan membuat *software engginer* dapat :

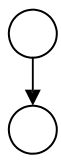
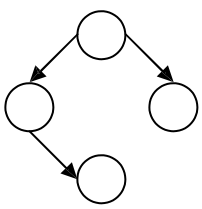
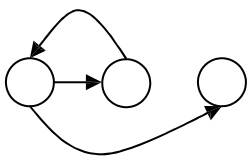
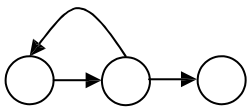
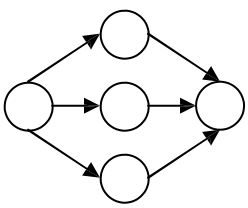
1. Menjamin bahwa semua jalur independen dalam sebuah modul telah dilewati paling tidak satu kali
2. Memeriksa semua keputusan logika baik pada sisi sebenarnya maupun pada sisi salahnya
3. Mengeksekusi semua *loop* pada nilai batasnya dan dalam nilai dimana dia harus beroperasi
4. Menguji struktur data internal untuk menjamin validasinya

5.2.7 *Basis Path Testing*

Metoda *Basis path testing* membuat perancang kasus uji bisa menurunkan sebuah ukuran kompleksitas logika dari sebuah perancangan prosedural dan ukuran ini selanjutnya digunakan untuk menentukan sekelompok data dasar (*basis set*) dari jalur eksekusi.

1. **Notasi Grafik Aliran (*Flow Graph*/*Program Graph*)** yang menggambarkan aliran kendali logika dengan menggunakan ilustrasi sebagai berikut :

Tabel 5.1 Notasi *Flow Graph*

Sequence	If	While	Repeat	Case
				

Lingkaran disebut *node* (N) digunakan untuk menunjukan pernyataan prosedural satu atau lebih.

Panah disebut *edge/link* (E) digunakan untuk menunjukan aliran dari kontrol.

Daerah yang dibatasi oleh node dan edge disebut sebagai *region*.

Setiap node yang mengandung kondisi disebut *predicate node* (P).

2. ***Cyclomatic Complexity*** adalah sebuah matriks *software* yang memberikan ukuran kuantitatif dari kompleksitas logika dari sebuah program. Ketika matrik ini digunakan dalam kontek metoda pengujian *basis path*, nilai yang dihitung dari *cyclomatic complexity* akan mendefinisikan jumlah jalur bebas (*independen path*) dari sekumpulan dasar program.

   Cyclomatic complexity dikembangkan berdasarkan teori graph. Cara menghitung *cyclomatic complexity* adalah sebagai berikut :

   - V(G) = Jumlah region
   - V(G) = E – N + 2
   - V(G) = P + 1

3. **Matrik Graph (*Matrix Graph*)** adalah matrik bujur sangkar yang mempunyai ukuran sama dengan jumlah node pada grafik aliran (*flow graph*). Masing-masing baris dan kolom berkaitan dengan node tertentu dan isi matrik berkaitan dengan hubungan (*edge*) antar node.

   Graph matrix ini akan menjadi alat yang amat beguna jika diberikan bobot hubungan (*link weight*) untuk masing-masing isi matrik. Pembobotan ini akan memberikan informasi tambahan tentang aliran kendali. Dalam bentuk Sederhananya isi nilainya adalah 1 (ada hubungan) dan 0 (tidak ada hubungan). Dengan cara penghitungan bahwa setiap yang ada hubungan, maka jumlah hubungan dikurangi satu, dan total dari jumlah hubungan tersebut ditambah satu akan menghasilkan *cyclomatic complexity*.

5.2.8 *Black Box Testing*

Black box testing memfokuskan pada kebutuhan fungsional dari *software*. Hal ini berarti bahwa pengujian ini memperbolehkan *software engineer* menurunkan sejumlah *input* yang ditujukan untuk menguji kebutuhan fungsional dari program tersebut. Pengujian ini berusaha menemukan *error* dengan kategori sebagai berikut :

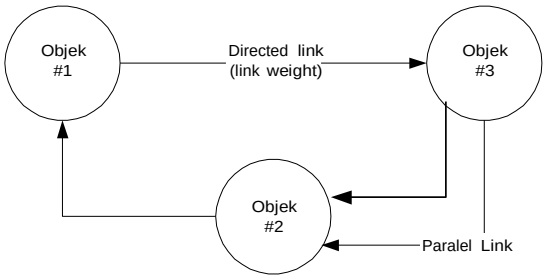
- Fungsi yang salah atau hilang
- Kesalahan antarmuka, struktur data atau pengaksesan data eksternal, unjuk kerja, inisialisasi dan penghentian

Tidak seperti pengujian *white box* yang dilakukan pada awal proses, pengujian *black box* diterapkan pada akhir tahapan proses pengujian. Hal ini dikarenakan pengujian ini tidak mementingkan struktur kontrol tapi lebih memfokuskan pada domain informasi.

A. **Metode Pengujian berdasarkan Grafik**

Langkah pertama pada pengujian *black box* adalah untuk memahami objek-objek yang dimodelkan dalam *software* dan hubungan dari objek tersebut. Langkah berikutnya adalah mendefinisikan serangkaian pengujian untuk memeriksa “semua objek mempunyai hubung satu terhadap yang lain sesuai dengan harapan”. Dengan kata lain pengujian *software* dimulai dengan membuat grafik dari objek yang penting dan hubungan dengan objek lain, kemudian melakukan serangkaian pengujian untuk melihat/mempelajari hubungan masing-masing objek dan melihat *error* yang ditemukan.

Untuk memenuhi langkah ini dibuat grafik sekumpulan *node* untuk menunjukan *link* yang menunjukan hubungan antar objek. Pembobot *node* (*node weight*) yang menggambarkan properti dari *node* (nilai data khusus atau kelakuan keadaan) dan *link weight* yang menggambarkankarakteristik dari hubungan.



B. **Pemisahan Persamaan (*Equivalence Partitioning*)**

Adalah metoda pengujian *black box* yang membagi *input domain* dari program ke dalam kelas-kelas data dimana kasus uji bisa diturunkan. *Equivalence Partitioning* didasarkan pada evaluasi persamaan kelas dari *input condition.* Sebuah persamaan kelas menunjukan sekumpulan keadaan valid atau tidak valid untuk syarat/kondisi masukan yang umumnya adalah nilai numerik tertentu, sebuah jangkauan nilai (*range value*), sebuah himpunan nilai-nilai yang berkaitan, atau kondisi boolean. Persamaan kelas bisa didefinisikan menurut panduan sebagai berikut:

- Jika kondisi *input* adalah sebuah range, satu kelas persamaan dan dua kelas persamaan didefinisikan
- Jika sebuah kondisi *input* memerlukan sebuah nilai khusus, satu kelas persamaan dan dua kelas persamaan didefinisikan
- Jika sebuah kondisi *input* adalah sebuah anggota himpunan, satu kelas persamaan yang valid dan satu kelas persamaan yang tidak valid didefinisikan
- Jika kondisi *input* adalah boolean, satu kelas yang valid dan satu kelas yang tidak valid.

C. **Analisis Nilai Batas (*Boundary Value Analysis*)**

Teknik analisis nilai batas ini dilakukan karena adanya fenomena bahwa kesalahan sering terjadi pada daerah batas dari suatu *input*. Teknik ini tidak hanya memperhatikan batas suatu nilai *input* tapi juga memperhatikan batas nilai *output*. Panduan untuk analisis nilai batas, adalah sebagai berikut:

- Bila kondisi *input* adalah sebuah range yang dibatasi nilai a dan b, kasus uji hendaknya dirancang dengan nilai a dan b, serta di atas atau di bawah sedikit dari nilai a dan b.
- Bila kondisi *input* adalah sebuah nilai, kasus uji harus dicoba nilai maksimum dan nilai minimum juga nilai sedikit di bawah/di atas dari nilai minimum dan maksimum
- Kedua panduan diatas diterapkan juga ke nilai keluaran
- Jika program mempunyai struktur data yang telah ditentukan batasannya.

D. **Pengujian Perbandingan**

Ada beberapa situasi dimana keadandalan *software* adalah kritis sekali. Pada aplikasi demikian ini *redundant hardware* dan *software* sering digunakan untuk meminimalkan *error*. *Redundant software* dikembangkan dengan cara memecahkan/memisahkan tim rekayasa perangkat lunak menjadi beberapa tim/versi yang tidak bergantung satu terhadap yang lain dengan menggunakan spesifikasi yang sama. Pada situasi seperti ini, masing-masing versi diuji dengan kasus yang sama untuk menjamin semua keluarannya adalah identik, kemudian semua versi ini dieksekusi secara paralel dengan perbandingan *real-time* untuk menjamin konsistensi.

Bila spesifikasi yang sama telah dikembangkan dengan banyak implementasi telah dihasilkan, perancangan kasus uji menggunakan teknik *black box* diberikan sebagai masukan untuk tiap-tiap versi *software* tersebut. Bila semua versi tersebut menghasilkan nilai keluaran yang sama, maka semua implementasinya benar. Bila tidak maka maing-masing versi perlu diuji lagi untuk mengetahui letak kesalahan.

5.2.9 Strategi Pengujian Perangkat Lunak

Strategi pengujian perangkat lunak adalah sebuah peta yang menggambarkan langkah-langkah yang harus dilakukan sebagai bagian dari pengujian, kapan pengujian direncanakan dan dilakukan, sejauh mana usaha, waktu dan sumber yang diperlukan. Jadi strategi pengujian harus menyatukan perencanaan pengujian,

perancangan kasus uji, pelaksanaan pengujian dan pengumpulan data hasil serta evaluasi terhadap data hasi uji tersebut

Verifikasi adalah sekumpulan aktifitas yang menjamin bahwa *software* telah diimplementasikan dengan benar untuk suatu fungsi tertentu. (*Are we building the product right*)

Validasi adalah sekumpulan aktifitas yang menjamin bahwa suatu *software* yang dibuat telah sesuai dengan kebutuhan user (*Are we building the right product*).

Faktor kualitas *software* bisa dikategorikan sebagai faktor secara langsung bisa diukur dan faktor yang tidak bisa diukur secara langsung. Faktor kualitas menurut Mc Call, diantaranya :

1. *Corectness* : apakah suatu program memenuhi spesifikasinya dan sesuai kehendak user
2. *Reliability* : apakah program tersebut cukup handal terhadap gangguan
3. *Efficiency* : perbandingan antara jumlah sumber dan kode yang diperlukan suatu program untuk melakukan suatu fungsinya
4. *Integrity* : ukuran dari kemampuan untuk mengontrol orang yang tidak diperbolehkan mengakses *software* tersebut
5. *Usability* : kemudahan untuk mempelajari, mengoperasikan,menyiapkan data *input*, menginterprestasikan *output* suatu *software*
6. *Maintainability* : usaha yang diperlukan untuk menemukan dan memperbaiki *error* dalam suatu program
7. *Testability* : usaha yang diperlukan untuk menguji sebuah program untuk menjamin bahwa program tersebut sesuai dengan fungsinya
8. *Flexibility* : usaha yang diperlukan untuk mengubah operasional dari program
9. *Portability* : usaha yang diperlukan untuk memindahkan program dari satu sistem *hardware* dan atau *software*
10. *Reusability* : kemudahan *software* untuk dipakai ulang untuk suatu paket *software* lain
11. Interoperability : usaha yang diperlukan untuk menggandeng suatu sistem dengan sistem yang lain

5.2.10 Dokumen Pengujian (STP [*Software Test Plan*] dan STR [Software Test Result])

Dokumen pengujian digunakan untuk memberikan laporan rencana dan rancangan pengujian (STP) serta hasil pengujian untuk suatu perangkat lunak (STR). STP (*Software Test Plan*) merupakan dokumen perancangan kasus uji. Dokumen ini dibuat sebelum dilakukan pengujian terhadap suatu aplikasi. Selain itu dokumen ini dibuat berdasarkan asumsi kemungkinan terdapat kesalahan dan atau kekurangan dalam suatu aplikasi.

STR (*Software Test Result*) merupakan dokumen hasil pengujian. Dokumen ini dibuat setelah dilakukan pengujian terhadap suatu aplikasi. Selain itu dokumen ini dibuat berdasarkan perancangan kasus uji dalam STP yang telah disusun sebelumnya.

Pada umumnya bentuk dokumen pengujian ini adalah :

Tabel 5.2 Kerangka Dokumen Pengujian

Kerangka Dokumen	Keterangan
Abstraksi	Abstraksi/Rangkuman dokumen pengujian
Daftar Isi Daftar Gambar Daftar Tabel	Daftar Isi, Daftar Gambar dan Daftar Tabel dalam Dokumen pengujian
1 Pendahuluan	
1.1 Tujuan Pengujian	Tujuan penyusunan dokumen pengujian dan menentukan siapa yang akan menggunakan pengujian ini
1.2 Ruang Lingkup Pengujian	Memberikan batasan pengujian
1.3 Deskripsi Umum Perangkat Lunak	Memberikan penjelasan umum dari perangkat lunak yang akan diuji
1.4 Daftar Definisi dan Singkatan	Menjelaskan definisi dan singkatan dalam dokumen pengujian
1.5 Referensi	Referensi/dokumen/bahan acuan yang digunakan
1.6 Overview STP	Menjelaskan isi dan organisasi dari STP secara singkat
2 Lingkungan Pengujian	
2.1 Perangkat Keras Pengujian	Menjelaskan *hardware* yang akan digunakan untuk pengujian
2.2 Perangkat Lunak Pengujian	Menjelaskan *software*, basis data dst yang akan digunakan untuk pengujian

Kerangka Dokumen	Keterangan
2.3 Sumber Daya Manusia	Menjelaskan sumber daya manusia yang diperlukan untuk pengujian
3 Prosedur Umum Pengujian	
3.1 Pengenalan dan Latihan	Menjelaskan tabel-tabel yang akan digunakan oleh perangkat keras (nama table, *primary key*, dan deskripsi table)
3.2 Persiapan Awal	Menjelaskan CDM atau E-R Diagram
3.2.1 Persiapan Prosedural	Menjelaskan persiapan prosedural
3.2.2 Persiapan Perangkat Keras	Mejelaskan persiapan perangkat keras
3.2.3 Persiapan Perangkat Lunak	Menjelaskan persiapan perangkat lunak
3.3 Pelaksanaan	Menjelaskan untuk suatu modul/proses tabel dengan data yang digunakan sebagai masukan dan keluaran untuk modul/proses tersebut
3.4 Pelaporan Hasil	Menjelaskan struktur tabel A (Identifikasi tabel, deskripsi isi, jenis tabel, volume, laju, *primary key*, dst)
4 Identifikasi dan Rencana Pengujian /*Software Test Plan* (STP)	Menjelaskan identifikasi dan rencana pengujian
5 Hasil Pengujian/*Software Test Result* (SRS)	Menjelaskan hasil pengujian
6 Pengujian *White Box*	Menjelaskan pengujian *white box* dari perangkat lunak (*flow graph* dan *cyclomatic complexity/ matrix graph*)
Lampiran	Berisi penjelasan tambahan pada laporan ini

Tabel 5.3 Contoh *Software Test Plan* (STP)

Kelas Uji	Kode Uji	Kasus Uji	Prosedur Pengujian	Masukan	Hasil yang diharapkan	Jenis Pengujian	Tanggal
Pengujian Login User	A-1	Pengujian untuk login ke aplikasi	- Jalankan aplikasi, akan ditampilkan form login - User mengisi textbox user id - User mengisi textbox password - User mengklik tombol OK	User ID =”ADA” Password = “ADA”	Layar menu ditampilkan	BlackBox	04-06-2003
Dst							

Tabel 5.4 Contoh *Software Test Result* (STR)

Kode Uji	A-1		
Nama Kasus Uji	Pengujian untuk Login ke Aplikasi		
Tujuan	Menguji fungsi login *user*		
Deskripsi	*User* melakukan pengujian login ke aplikasi		
Kondisi Awal	Layar login telah ditampilkan		
Tanggal Pengujian	4 Juni 2003		
Penguji	Kasep Thea		
Skenario Uji			
1. Isi textbox User ID 2. Isi textbox password 3. Klik tombol OK			
Kriteria Evaluasi Hasil			
Jika user id dan password benar layar menu akan ditampilkan, sebaliknya jika salah akan ditampilkan pesan kesalahan			
Kasus dan Hasil Uji			
Data Masukan	**Hasil yang Diharapkan**	**Pengamatan/Hasil Pengujian**	**Kesimpulan**
User id = “ADA” Password = “ADA”	Layar menu akan ditamplikan	Layar menu tidak ditampilkan dan tidak ada pesan kesalahan	ditolak
Catatan			
1. Proses login gagal karena user id belum terdaftar dalam tabel user 2. belum ada penanganan kesalahan jika user id belum terdaftar			

5.3 TUGAS PENDAHULUAN XI

Dikumpulkan pada pertemuan ke sebelas

1. Jelaskan Tentang Pengujian!
2. Jelaskan kegunaan pengujian dalam tahapan pengembangan perangkat lunak!
3. Jelaskan tentang *Black Box Testing* dan *White Box Testing!*

5.4 LATIHAN PRAKTIKUM VIII (Pertemuan ke sebelas dan ke duabelas)

1. Berdasarkan hasil implementasi pada pertemuan sebelumnya, buat identifikasi dan rencana pengujian **minimum 50 kasus uji**!
2. Lakukan pengujian hasil implementasi tersebut sesuai rencana pengujian di atas!
3. Buat dokumentasi pengujian !
4. Jika dalam pengujian masih terdapat kesalahan lakukan perbaikan dan perbaiki pula dokumen-dokumen yang terkait!