# JARINGAN DINAMIS
## Sebuah Model Mealy Automaton

**TESIS MAGISTER**

Oleh:
Sugiarto Pudjohartono
NIM: 23594014

**Bidang Khusus Rekayasa Perangkat Lunak**
**Program Studi Teknik Informatika**
**Program Pascasarjana**
**Institut Teknologi Bandung**
**1997**

# JARINGAN DINAMIS
## Sebuah Model Mealy Automaton

Nama : Sugiarto Pudjohartono
NIM : 23594014

Pembimbing

DR. Oerip Setiono Santoso, M.Sc
NIP 130357665

1997

# ABSTRAK

Pada Tesis ini telah berhasil dilakukan studi pembangunan sebuah prototipe Jaringan Dinamis, dan mengimplementasikannya dalam bahasa Borland Delphi versi 2.0.

Studi pembangunan dilakukan dengan menggunakan pendekatan objek, dan melalui tahapan analisis dihasilkan model dari permasalahan dalam bentuk diagram objek. Setelah memperhatikan dukungan teknologi secara khusus adalah dukungan fasilitas pemrograman Delphi hasil analisis disempurnakan menjadi sebuah rancangan yang siap diimplementasikan. Selanjutnya, setelah melalui proses penulisan kode program, pengeksekusian program dan penanganan kesalahan, dapat dihasilkan sebuah program jaringan dinamis yang siap beroperasi di bawah sistem operasi Microsoft Windows '95.

Selain kebutuhan sistem operasi tersebut, pengoperasian program DM memerlukan perangkat keras dengan spesifikasi: prosesor minimum 486 DX2/66; memori minimum 16 MB; resolusi layar tampilan minimum 800x600 *pixels* ; dan sebuah *mouse*.

Pada tesis ini juga telah dilakukan pengujian terhadap perangkat lunak jaringan dinamis. Pengujian dilakukan terhadap faktor kebenaran; faktor otorisasi; faktor integritas berkas; faktor pemenuhan tingkat layanan; dan faktor kontrol akses.

**Kata kunci**: Kelas, Kelas & Objek, *Instance*, *Object Oriented Analysis*; *Object Oriented Design*; *Object Oriented Programming*.

# KATA PENGANTAR

Puji dan syukur penulis hunjukkan kehadirat Nya. Karena kasihNya penulis dapat menyelesaikan tesis ini.

Tesis ini disusun sebagai salah satu syarat mengikuti sidang akhir untuk meraih gelar magister pada Program Studi Magister Informatika, Institut Teknologi Bandung.

Tesis ini dapat diselesaikan berkat dukungan dan bantuan dari berbagai pihak. Melalui tulisan ini penulis menyampaikan ucapan terima kasih dan penghargaan yang tinggi kepada:

1. Bapak DR. Oerip Setiono Santoso, M.Sc., selaku pembimbing tesis, ketua tim penguji tesis, dan dosen wali.
2. Bapak DR. Ing. M.Sukrisno Mardiyanto, selaku dosen mata kuliah Pembangunan Sistem Perangkat Lunak dan sekretaris tim penguji tesis, yang telah banyak memberi masukan berkaitan dengan pendekatan objek pada proses rekayasa perangkat lunak jaringan dinamis.
3. Ibu Ir. Sri Purwanti, M.Sc., selaku anggota tim penguji tesis.
4. Bapak DR. Ing. Farid Wazdi selaku Ketua Program Studi Magister Informatika.
5. Bapak. DR. Yansen Marpaung, Pembantu Dekan I FKIP Universitas Sanata Dharma, yang berkenan memberi masukan

topik jaringan dinamis dan membantu penulis untuk memahami paper / journal tentang jaringan dinamis yang ditulis oleh pencipta alat dalam bahasa Jerman.

6. Rekan dosen Jurusan Pendidikan Matematika FKIP Universitas Sanata Dharma, dan rekan mahasiswa Program Studi Magister Informatika angkatan 1994, yang telah memberi dukungan dan dorongan semangat.

Penulis menyadari bahwa masih banyak dijumpai kekurangan pada tesis ini. Hal itu disebabkan karena keterbatasan pengetahuan dan pengalaman penulis. Oleh karenanya, dengan rendah hati penulis sangat mengharap kritik dan saran membangun dari para pembaca. Meski dalam kondisi serba terbatas, penulis berharap dapat menyumbangkan sesuatu yang bermanfaat bagi pembaca.

Bandung, April 1997

Penulis

# DAFTAR ISI

## DAFTAR TABEL

## DAFTAR GAMBAR

## DAFTAR LAMPIRAN

# BAB I
# PENDAHULUAN

Pada bab ini penulis akan menyajikan : latar belakang pemilihan topik tugas akhir; tujuan tugas akhir; ruang lingkup tugas akhir; metodologi penyusunan tugas akhir; dan ikhtisar dari laporan tugas akhir.

## 1.1 Latar Belakang Tugas Akhir

Sebuah penelitian yang berjudul "Improving the Understanding of Mathematical Concepts Through Computer Paradigm in Students' Mind" belum lama ini dilaksanakan oleh Marpaung. Penelitian itu merupakan proyek kerjasama antara Pusat Penelitian Sanata Dharma bidang Pendidikan Matematika dan Ilmu Pengetahuan Alam di Yogyakarta, dan Universitaet Osnabrueck di Jerman.

Penelitian di atas hendak mempelajari proses kognitif anak (siswa Sekolah Lanjutan Tingkat Pertama) dalam menyusun algoritma. Algoritma yang dimaksud adalah algoritma dalam bidang Aritmatika. Proses kognitif anak digali melalui proses pengerjaan terhadap sekumpulan tugas menyusun algoritma yang diberikan kepadanya. Tugas-tugas itu dapat diselesaikan oleh siswa dengan memanfaatkan modus-modus representasi yang disediakan, yaitu: modus enaktif; modus ikonik; dan modus simbolik, sesuai dengan minat mereka.

Alat bantu yang disediakan untuk menyelesaikan tugas adalah: kotak dan batang untuk mewakili modus enaktif; jaringan dinamis untuk mewakili modus ikonik; dan program komputer untuk mewakili modus simbolik. Alat bantu pertama, yaitu: kotak dan batang, sangatlah mudah dan murah untuk dibuat. Alat bantu ketiga, yaitu: program komputer, telah dibuat dan dengan bebas dapat digandakan. Sedangkan alat kedua, yaitu: jaringan dinamis, telah juga dibuat tetapi masih dalam jumlah sangat terbatas dan cukup mahal. Harga alat kedua cukup mahal disebabkan karena tuntutan ketepatan alat dan biaya produksi yang masih tinggi (alat diproduksi secara *home-industry*).

Hasil penelitian sementara menunjukkan adanya kecenderungan pemilihan modus representasi dan proses pengerjaan siswa yang mencerminkan struktur kognitif siswa. Di samping hasil itu, penelitian juga memperoleh hasil lain, yaitu: adanya gambaran bahwa tidak setiap siswa memiliki kecenderungan dan kepekaan yang sama terhadap setiap bentuk modus representasi; adanya indikasi korelasi positif antara tingkat abstraksi dari variasi modus representasi yang disukai siswa dan kemampuan siswa untuk memahami konsep-konsep Matematika, penyelesaian soal-soal Matematika, serta algoritma. [Mar-1992, p.i-2]. Hasil penelitian semacam ini sangat dibutuhkan bagi perkembangan Pendidikan Matematika.

Penelitian di atas hingga sekarang masih akan dikembangkan lebih lanjut. Mengingat sumbangan yang dapat diberikan oleh penelitian itu terhadap dunia Pendidikan Matematika, dan keterbatasan alat, maka penulis bermaksud mengembangkan sebuah perangkat lunak komputer

yang dapat menggantikan fungsi alat jaringan dinamis. Perangkat lunak tersebut selanjutnya akan dinamakan **DM** ***(Dynamic Maze)***. Harapan penulis agar salah satu kendala penelitian, yaitu kendala tentang alat, dapat diatasi.

## 1.2 Tujuan Tugas Akhir

Tugas akhir ini bertujuan membangun sebuah perangkat lunak DM, yang dapat mendukung penelitian untuk mengenal struktur kognitif siswa.

Kemampuan perangkat lunak DM adalah:

1. Menyediakan fasilitas bantuan tentang cara mengkonstruksi jaringan dinamis pada perangkat lunak DM.
2. Menyediakan fasilitas bagi siswa untuk mengkonstruksi jaringan dinamis, termasuk fasilitas gambar-hapus komponen jaringan dinamis , *zooming*, dan *scrolling*.
3. Menangani pengelolaan berkas data (*save, retrieve*) hasil konstruksi siswa.

## 1.3 Ruang Lingkup Tugas Akhir

Tugas akhir ini mencakup sejumlah usaha, yaitu:

1. Merancang perangkat lunak DM yang dapat mendukung usaha penelitian tentang struktur kognitif siswa, sesuai dengan tahapan rekayasa perangkat lunak dan menggunakan pendekatan objek.

2. Mengimplementasikan rancangan perangkat lunak DM pada bahasa pemrograman Borland Delphi versi 2.0.

3. Menyusun laporan tugas akhir dalam bentuk tesis.

## 1.4 Metodologi Tugas Akhir

Tugas akhir ini disusun dengan melibatkan sejumlah metoda, yaitu:

1. Studi Lapangan

   Pada metoda ini penulis melakukan pengamatan terhadap pemanfaatan jaringan dinamis pada penelitian untuk mengenal struktur kognitif siswa. Di samping itu, penulis mempelajari juga kaidah-kaidah yang harus diperhatikan oleh anak selama proses penyusunan jaringan dinamis.

   Pengamatan penulis lakukan dengan cara langsung, artinya penulis pernah melibatkan diri membantu penelitian tersebut. Selain cara langsung, penulis juga melakukan pengamatan dan mempelajari kaidah penyusunan jaringan dinamis melalui penggalian informasi dari peneliti.

2. Studi Literatur

   Pada metoda ini penulis melakukan studi pustaka yang berkaitan dengan: teori dasar jaringan dinamis, yaitu teori *Automata*; teori rekayasa perangkat lunak berorientasi objek;

teori perancangan antarmuka pemakai; dan bahasa pemrograman Borland Delphi versi 2.0.

3. Studi Kasus

Pada metoda ini penulis merancang perangkat lunak DM serta mengimplementasikannya pada bahasa pemrograman Borland Delphi versi 2.0, dengan memperhatikan kedua hasil studi di atas .

Pemilihan bahasa pemrograman didasarkan pada pertimbangan bahwa penulis telah mengenal bahasa pemrograman Pascal. Bahasa pemrograman Borland Delphi merupakan pengembangan dari bahasa pemrograman Pascal. Dengan demikian diharapkan penulis cukup mudah menyesuaikan diri dengan bahasa pemrograman Delphi.

Perancangan perangkat lunak DM menggunakan metoda objek dan paradigma *prototyping*.

Pemakaian metoda objek didasarkan pada pertimbangan kemudahan melakukan *reuse-analysis*, *reuse-design*, dan *reuse-code*.

Pemakaian paradigma rekayasa perangkat lunak *prototyping* didasarkan pada pertimbangan situasi pengguna dan pengembang. Meski pengguna dapat menentukan tujuan / kemampuan perangkat lunak DM, namun pengguna tidak dapat mendeskripsikan detil kebutuhan masukan / keluaran,

dan kebutuhan model interaksi pemakai-sistem. Sementara itu pengembang, dalam hal ini adalah penulis, memiliki keterbatasan pengalaman merekayasa perangkat lunak berorientasi objek.

*Prototype* perangkat lunak DM pada tugas akhir ini dibatasi sampai dengan pemenuhan kemampuan perangkat lunak DM pada butir 1.2 di atas.

## 1.5 Sistematika Pembahasan

Pembahasan pada buku tesis ini akan mengikuti sistematika pembahasan sebagai berikut.

**Bab I Pendahuluan**

Bab ini berisi uraian tentang latar belakang pemilihan topik tugas akhir; tujuan yang hendak diraih pada pembuatan tugas akhir; batasan lingkup tugas akhir; metodologi yang digunakan selama tugas akhir berlangsung; dan sistematika pembahasan laporan tugas akhir yang ditulis dalam bentuk tesis.

**Bab II Dasar Teori**

Bab ini berisi uraian teori yang langsung berkaitan dengan jaringan dinamis dan proses rekayasa perangkat lunak. Pembahasan dilakukan terhadap: teori rekayasa perangkat lunak berorientasi objek; teori perancangan antarmuka; teori automata

yang mendasari pembuatan alat jaringan dinamis; dan pembahasan tentang jaringan dinamis itu sendiri.

**Bab III Perancangan Perangkat Lunak DM**

Pada bab ini dibahas deskripsi umum terhadap perangkat lunak DM yang akan dibuat; hasil analisis berorientasi objek (OOA); dan hasil rancangan berorientasi objek (OOD). Alat bantu yang dipakai untuk menggambarkan hasil OOA dan OOD adalah diagram objek versi Peter Coad dan Edward Yourdon.

**Bab IV Implementasi Perangkat Lunak DM**

Pada bab ini dibahas tentang: deskripsi fungsional global dari kelas & objek hasil perancangan; pembagian modul terhadap hasil rancangan (OOD); implementasi dari setiap modul dengan memakai bahasa pemrograman Borland Delphi versi 2.0.

**Bab V Uji Perangkat Lunak DM**

Pada bab ini dibahas langkah pengujian dan hasil pengujian terhadap perangkat lunak DM. Pembahasan meliputi penentuan faktor uji yang sesuai; penyiapan kasus uji berikut kriteria uji; pengamatan hasil pengujian dan pengambilan kesimpulan hasil pengujian.

**Bab VI Kesimpulan dan Saran**

Pada bab ini dibahas hal penting yang didapati selama perancangan hingga implementasi DM, kesimpulan uji perangkat lunak, dan saran-saran bagi pengembangan DM lebih lanjut.

# BAB II
# DASAR TEORI

Pada bab ini penulis akan menyajikan: teori rekayasa perangkat lunak berorientasi objek; teori perancangan antarmuka pemakai; teori *automata* ; dan jaringan dinamis.

Deskripsi teori tersebut akan didahului oleh bagian pendahuluan. Bagian ini memberikan penjelasan singkat tentang dukungan teori terhadap pencapaian tujuan tugas akhir.

## 2.1 Pendahuluan

Teori rekayasa perangkat lunak berorientasi objek diperlukan sebagai dasar pijak penulis selama mengembangkan perangkat lunak DM. Penulis memilih pendekatan ini disebabkan oleh pertimbangan utama yaitu kemudahan untuk melakukan analisis ulang (*reuse-analysis*), desain ulang (*reuse-design*), dan koding ulang (*reuse-code*).

Pada bagian ini penulis akan menyajikan: definisi objek; terminologi istilah serta notasi yang dipakai; tahapan analisis yang dikenal sebagai *Object Oriented Analysis* (OOA); tahapan perancangan yang dikenal sebagai *Object Oriented Design* (OOD); serta tahapan implementasi yang dikenal sebagai *Object Oriented Programming* (OOP), yang secara khusus menggunakan bahasa pemrograman Delphi.

Notasi yang dipakai pada teori ini mengacu pada notasi yang dikembangkan oleh Peter Coad dan Edward Yourdon. Pemakaian notasi tersebut karena kesederhanaan notasi untuk merepresentasikan model.

Teori perancangan antarmuka-pemakai diperlukan sebagai rambu-rambu yang harus diperhatikan, dipertimbangkan serta diakomodasikan pada saat akan merancang antarmuka-pemakai.

Pada bagian ini penulis akan menyajikan rincian dari sejumlah 15 buah prinsip umum yang mendasari perancangan antarmuka-pemakai, menurut Deborah J. Mayhew [May-92].

Teori *Automata*, secara khusus adalah Mealy *Automaton*, merupakan teori yang mendasari penciptaan model-model bata penyusun sebuah jaringan-dinamis.

Pada bagian ini penulis akan menyajikan konsep *Automata* secara umum, konsep Mealy *Automaton* secara khusus, dan macam rangkaian Mealy *Automaton*.

Bagian terakhir dari bab ini adalah deskripsi tentang jaringan dinamis. Pada bagian ini penulis akan menyajikan: komponen jaringan-dinamis; mekanisme kerja setiap komponen jaringan-dinamis; serta penerapan konsep Mealy *Automaton* pada setiap komponen jaringan-dinamis.

## 2.2 Teori Rekayasa Perangkat Lunak Berorientasi Objek

Pada bagian ini akan disajikan: terminologi istilah dan notasi, pengertian objek sebagai pendekatan rekayasa perangkat lunak, OOA, OOD, dan OOP.

### 2.2.1 Terminologi Istilah dan Notasi Objek

Pada bagian ini akan disajikan sejumlah istilah dan notasi yang terkait dengan proses rekayasa perangkat lunak berorientasi objek.

Sejumlah istilah tersebut adalah: kelas; kelas & objek; struktur; subjek; atribut; layanan; *instance connection*; *message connection*. Sedangkan sejumlah notasi yang dimaksud adalah notasi kelas; notasi kelas & objek; notasi struktur; notasi subjek; notasi atribut; notasi layanan; notasi *instance connection*; dan notasi *message connection*.

**Definisi dan Notasi untuk Kelas, Kelas & Objek, *Instance***

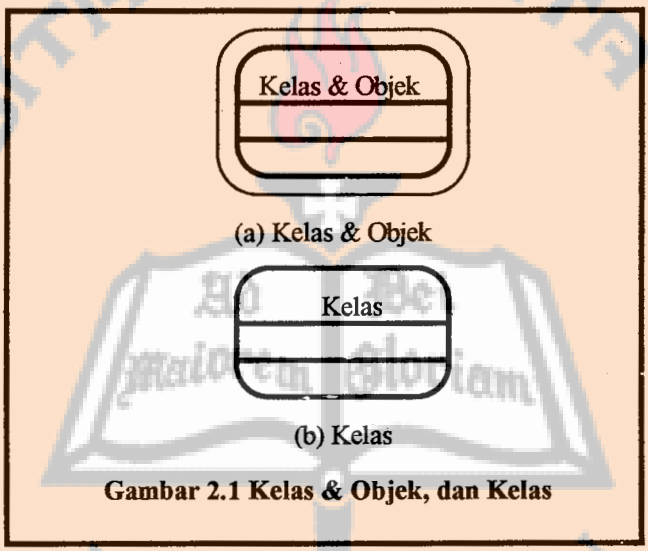
Objek merupakan abstraksi dari sesuatu pada ranah masalah yang menggambarkan kemampuan sistem dalam menyediakan informasi dan kemampuan berinteraksi dengan sesuatu tersebut. [CYo-91, p.26]

Kelas merupakan sebuah deskripsi dari satu atau lebih dari satu objek yang memiliki kesamaan atribut dan layanan. [CYo-91,p.26]

Selama tahapan analisis, perancangan, dan implementasi, dipakai istilah kelas & objek serta kelas. Istilah kelas & objek menunjuk pada sebuah kelas berikut objek-objeknya. Istilah kelas hanya menunjuk pada sebuah kelas saja tanpa menunjuk objek-objeknya.

Di samping istilah kelas & objek dan kelas, Coad memberikan istilah *instance* . Istilah *instance* menunjuk pada individu objek yang unik.

Notasi kelas & objek, serta kelas dapat dilihat pada gambar berikut:




**Gambar 2.1 Kelas & Objek, dan Kelas**

Kedua notasi hanya berbeda pada garis-luar yang melingkungi kelas. Pada notasi kelas & objek, garis-luar menunjukkan bahwa pada kelas tersebut didefinisikan (terdapat) sejumlah *instance*. Sedangkan pada notasi kelas tidak terdapat garis-luar.

**Definisi Struktur dan Notasi Struktur**

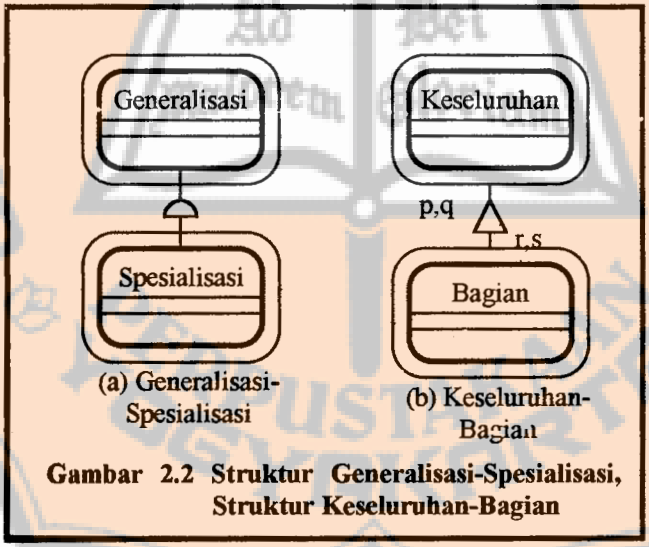
Struktur merupakan ekspresi dari kompleksitas ranah permasalahan yang terkait dengan responsibilitas sistem. [CYo-91,p.28] Terdapat 2 buah jenis struktur, yaitu: struktur generalisasi-spesialisasi

(*generalization-specialization structure*); dan struktur keseluruhan-bagian (*whole-part structure*).

Struktur generalisasi-spesialisasi merupakan struktur yang merelasikan kelas dengan kelas, dan dikenal sebagai struktur '*is a*' atau '*is a kind of*'.

Struktur keseluruhan-bagian merupakan struktur yang merelasikan *instance* dengan *instance*, dan dikenal sebagai struktur '*has a*'.

Notasi struktur generalisasi-spesialisasi (*Gen-Spec*)dan struktur keseluruhan-bagian (*Whole-Part*) dapat dilihat pada gambar berikut:




**Gambar 2.2 Struktur Generalisasi-Spesialisasi, Struktur Keseluruhan-Bagian**

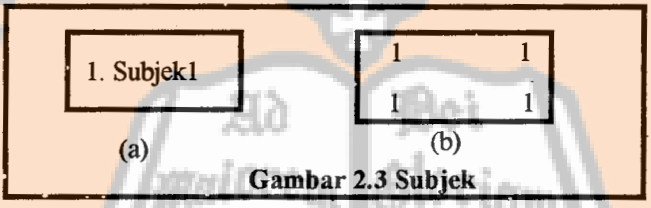
Catatan gambar 2.2.b. Diagram itu dibaca sebagai berikut: *Instance* dari Keseluruhan memiliki sejumlah p hingga q *instance* dari

Bagian, dan sebuah *instance* dari Bagian merupakan bagian dari r hingga s *instance* dari Keseluruhan.

**Definisi Subjek dan Notasi Subjek**

Subjek merupakan mekanisme yang dipakai untuk memandu pembaca (analis, manajer, pemakai) agar cepat memahami model yang cukup besar dan kompleks. Subjek memuat sejumlah kelas & objek dan atau kelas.

Notasi subjek dapat dilihat pada gambar berikut:



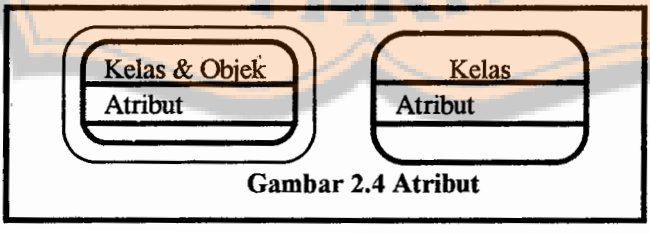

**Gambar 2.3 Subjek**

Catatan:Gambar 2.3.a merupakan notasi singkat. Gambar 2.3.b merupakan notasi yang dipakai jika notasi kelas&objek akan ditampilkan juga.

**Definisi Atribut dan Notasi Atribut**

Atribut merupakan data yang dimiliki oleh (terkait dengan) setiap kelas atau kelas & objek.

Notasi atribut dapat dilihat pada gambar berikut:

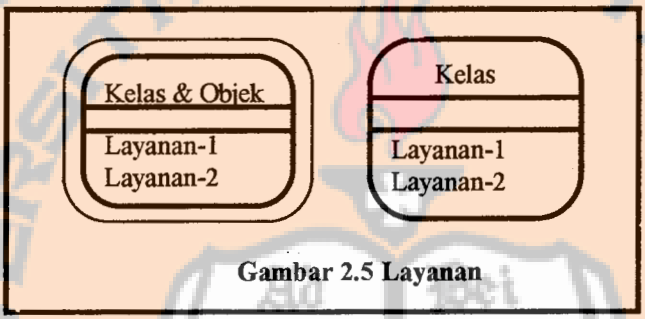



**Gambar 2.4 Atribut**

**Definisi Layanan dan Notasi Layanan**

Layanan merupakan kelakuan khusus yang ditunjukkan oleh kelas atau kelas & objek, dalam upaya mendukung responsibilitas sistem. Pada umumnya layanan berupa fungsi atau prosedur.

Notasi layanan dapat dilihat pada gambar berikut:

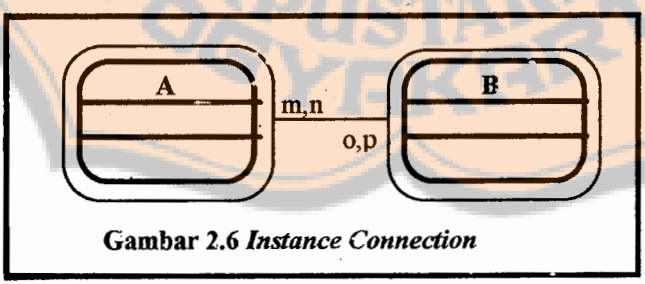



**Gambar 2.5 Layanan**

**Definisi *Instance Connection***

*Instance Connection* merupakan model relasi ranah masalah yang mencerminkan kebutuhan satu *instance* terhadap *instance* yang lain dalam upaya mendukung keberadaannya.

Notasi *Instance Connection* dapat dilihat pada gambar berikut:



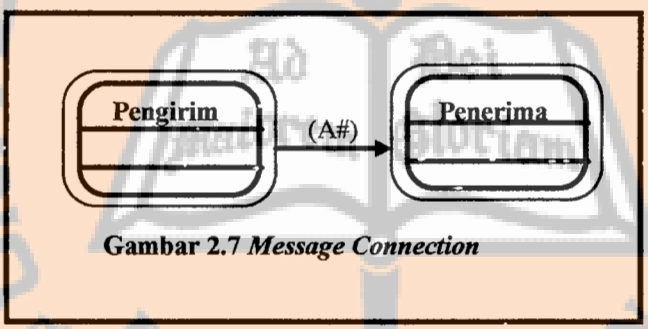

**Gambar 2.6 *Instance Connection***

Model relasi yang digambarkan pada gambar 2.6 dibaca sebagai berikut: setiap satu *instance* dari A berrelasi dengan m hingga n buah *instance* dari B, dan setiap satu *instance* dari B berrelasi dengan o hingga p buah *instance* dari A.

**Definisi *Message Connection***

*Message Connection* merupakan model keterkaitan proses sebuah *instance* yang mencerminkan kebutuhan akan layanan dalam upaya mendukung keberadaan *instance* tersebut.

Notasi *Message Connection* dapat dilihat pada gambar berikut:




**Gambar 2.7 *Message Connection***

Catatan: (A#) merupakan simbol dari pesan yang dikirim oleh pengirim menuju penerima. Karakter pertama merupakan abjad, dan karakter kedua merupakan bilangan asli. Simbol tersebut menunjukkan urutan pesan yang dikirim oleh *instance*.

### 2.2.2 Objek Sebagai Pendekatan Rekayasa Perangkat Lunak

Pada siklus pengembangan sebuah perangkat lunak, kegiatan analisis merupakan kegiatan awal yang menjadi acuan kerja bagi kegiatan selanjutnya.

Analisis merupakan proses mengekstraksi kebutuhan dari sebuah sistem. Dapat pula dikatakan bahwa analisis sebagai proses untuk menetapkan apa yang harus dilakukan oleh sistem untuk memenuhi permintaan pemakai [CoY-91,p.19 ; EYo-94,p.248].

Salah satu pendekatan yang dapat dilakukan pada kegiatan analisis adalah pendekatan objek. Pendekatan ini menghendaki agar kita melakukan pemetaan langsung dari ranah masalah dan responsibilitas sistem menuju sebuah model.

Model merupakan sesuatu yang diciptakan untuk menghadirkan , menggantikan, dan berperan sebagaimana peran dari sesuatu yang dimodelkan.

Model yang dibuat melibatkan sejumlah kelas dan kelas & objek, dan sejumlah prinsip untuk menangani kompleksitas masalah. Prinsip yang dimaksud adalah: abstraksi (*abstraction*) ; pembungkusan (*encapsulation*); pewarisan (*inheritance*); asosiasi (*association*); komunikasi pesan (*communication with message*); metoda berpikir; skala; dan klasifikasi kelakuan [CYo-91, p.6]

Abstraksi merupakan proses untuk mengabaikan sejumlah aspek yang kurang relevan terhadap tujuan yang akan dicapai, dan hanya

memperhatikan aspek yang relevan saja. Prinsip ini mencakup abstraksi prosedural dan abstraksi data.

Abstraksi prosedural menghendaki agar setiap operasi yang didefinisikan harus dapat diperlakukan sebagai sebuah entitas tunggal. Berkaitan dengan hal itu diperlukan upaya merinci sebuah proses menjadi proses-proses yang lebih kecil dan tunggal.

Abstraksi data menghendaki agar pendefinisian setiap tipe data terkait dengan kebutuhan operasi terhadap *instance*.

Pembungkusan merupakan upaya untuk mengisolasi komponen program terhadap campur tangan dari komponen lain yang tidak / kurang relevan.

Pewarisan merupakan upaya untuk mengekspresikan similaritas dari sejumlah kelas, dengan tetap menjaga kesederhanaan pendefinisian terhadap kelas yang telah terdefinisikan sebelumnya.

Asosiasi merupakan upaya untuk menghubungkan *instance* dari sejumlah berhingga kelas & objek yang aktif pada waktu yang sama atau situasi yang serupa.

Komunikasi pesan merupakan interaksi pesan yang terjadi di antara para *instance* yang aktif. Sebuah *instance* menjadi pengirim pesan, dan *Instance* yang lain menjadi penerima pesan.

Metoda berpikir mencakup penurunan (diferensiasi) pengalaman ke dalam kelas atau kelas & objek berikut atributnya; pembedaan antara

kelas & objek keseluruhan dengan kelas & objek bagian; dan pembentukan serta pembedaan antar kelompok dan anggota.

Skala merupakan prinsip yang mengaplikasikan relasi *whole-part* untuk sesuatu (masalah) yang lebih besar.

Klasifikasi kelakuan merupakan prinsip mengenali kelakuan dari kelas dan kelas & objek, yang mencakup: kelakuan berbasis suatu sebab langsung; kelakuan yang berubah karena waktu; kelakuan karena fungsi yang mirip (similar).

### 2.2.3 OOA

Pada tahap ini, kegiatan analisis mencakup 5 buah kegiatan pokok yaitu: pendefinisian kelas dan kelas & objek; pendefinisian struktur; pendefinisian subjek; pendefinisian atribut; dan pendefinisian layanan.

Penyebutan urutan dari kelima kegiatan tersebut bukan merupakan urutan kegiatan yang harus diselesaikan sebelum kegiatan pada urutan selanjutnya dilakukan.

Urutan penyebutan tersebut berkaitan dengan tingkat abstraksi yang diperlukan oleh setiap kegiatan. Kegiatan pendefinisian kelas dan kelas & objek merupakan kegiatan yang memerlukan tingkat abstraksi relatif tertinggi dibandingkan kegiatan selanjutnya. Demikian pula dengan kegiatan berikutnya.

Selanjutnya, hasil dari kelima kegiatan di atas dipresentasikan sebagai sebuah model lapisan-jamak (*multi-layer*), yaitu: lapisan subjek;

lapisan kelas & objek; lapisan struktur; lapisan atribut; dan lapisan layanan.

### 2.2.3.1 Tahap Pendefinisian Kelas dan Kelas & Objek

Pada tahap ini kita harus memahami benar ranah masalah. Melalui kecermatan dalam melakukan abstraksi terhadap ranah masalah, kita melakukan pendefinisian sejumlah kelas dan kelas & objek yang diperlukan.

Edward Yourdon [EYo-94,p.125-131] memberi 3 macam tinjauan untuk membantu menemukan kelas atau kelas & objek yang diperlukan. Kelas atau kelas & objek dapat ditemukan dengan mencermati seputar data yang berkaitan dengan sistem, mempertanyakan fungsi kehadiran nya, dan mempertanyakan kelakuannya dalam kaitan dengan responsibilitas sistem yang akan dimodelkan.

Hasil akhir pada tahapan ini adalah sebuah diagram yang berisi seluruh kelas dan kelas & objek yang telah didefinisikan, dan dinyatakan dalam notasi objek. Diagram ini merupakan presentasi bagian dari model sistem pada lapisan kelas dan objek (*class & object layer*).

### 2.2.3.2 Tahap Pendefinisian Struktur

Pada tahap ini kita mendefinisikan struktur yang dapat dibangun pada kelas dan kelas & objek yang telah didefinisikan. Struktur yang dapat dibangun adalah: struktur keseluruhan-bagian; dan struktur generalisasi-spesialisasi.

Hasil akhir pada tahapan ini adalah sebuah diagram yang berisi struktur dari seluruh kelas dan kelas & objek yang telah didefinisikan, dan dinyatakan dalam notasi objek. Diagram ini merupakan presentasi bagian dari model sistem pada lapisan struktur (*structure layer*).

### 2.2.3.3 Tahap Pendefinisian Subjek

Pada tahap ini kita mendefinisikan sejumlah subjek dari kelas, dan kelas & objek yang didefinisikan. Subjek didefinisikan dari kekumpulan kelas dan kelas & objek yang memiliki struktur terpisah dari struktur kumpulan kelas dan kelas & objek yang lain.

Kegiatan pada tahap ini bersifat fakultatif. Jikalau model OOA melibatkan cukup banyak kelas atau kelas & objek maka pendefinisian subjek sangat diperlukan Sedangkan jikalau model OOA tidak melibatkan cukup banyak kelas atau kelas & objek, maka kegiatan pada tahap ini dapat tidak dilakukan.

Hasil akhir pada tahapan ini adalah diagram yang memuat seluruh subjek yang telah didefinisikan, dan dinyatakan dalam notasi objek. Diagram ini merupakan presentasi bagian dari model sistem pada lapisan subjek (*subject layer*).

### 2.2.3.4 Tahap Pendefinisian Atribut

Pada tahap ini kita mendefinisikan atribut-atribut yang relevan dengan kelas dan kelas & objek yang telah didefinisikan, berikut *instance connection* yang dapat dibangun.

Setelah seluruh atribut didefinisikan, selanjutnya dilakukan penempatan atribut dengan memperhatikan kestabilan dan konsistensi pemodelan.

Hasil akhir pada tahapan ini adalah sebuah diagram yang memuat seluruh atribut yang telah didefinisikan berikut *instance connection* yang dapat dibangun, dan dinyatakan dalam notasi objek. Diagram ini merupakan presentasi bagian dari model sistem pada lapisan atribut (*attribute layer*).

### 2.2.3.5 Tahap Pendefinisian Layanan

Pada tahap ini kita mendefinisikan layanan-layanan yang relevan dengan kelas dan kelas & objek yang telah didefinisikan, serta *message-connection* yang dapat dibangun pada kelas & objek.

Edward Youdon [EYo-94,p.233] memberikan sejumlah macam layanan dasar yang dapat dijadikan acuan untuk mendefinisikan layanan. Layanan dasar tersebut adalah: layanan implisit (misal: *create; modify; search; delete*; ...); layanan yang berkaitan dengan *instance-connections*; layanan yang berkaitan dengan *message-connection*; layanan yang berkaitan dengan atribut; layanan yang dibutuhkan akibat perkembangan / perubahan pada diagram objek.

Hasil akhir pada tahapan ini adalah diagram yang memuat seluruh layanan yang telah didefinisikan berikut *message-connection* antar kelas & objek yang dapat dibangun, dan dinyatakan dalam notasi objek. Diagram ini merupakan presentasi bagian dari model sistem pada lapisan layanan (*service layer*).

## 2.2.4 OOD

Jika kegiatan analisis memusatkan perhatian pada upaya untuk menetapkan apa yang harus dilakukan oleh sistem, maka pada kegiatan desain perhatian dipusatkan pada upaya untuk membuat hasil analisis dapat diimplementasikan.

Kegiatan analisis biasanya didasarkan pada asumsi bahwa teknologi yang tersedia sempurna. Sedangkan kegiatan perancangan biasanya didasarkan pada asumsi bahwa sistem akan diimplementasikan pada *platform* X, di bawah sistem operasi Y, dan dengan bahasa pemrograman Z.

Peter Coad dan Edward Yourdon [EYo-94, p.306-309] memberikan sejumlah pedoman yang perlu dipertimbangkan oleh perancang pada tahap OOD. Sejumlah pedoman tersebut adalah: keterkaitan (*coupling*); keterikatan (*cohesion*); kejelasan rancangan; hirarki dan pemfaktoran; kesederhanaan kelas dan kelas & objek; kesederhanaan protokol pesan; kesederhanaan layanan; meminimumkan rancangan yang mudah berubah-ubah; meminimumkan ukuran sistem keseluruhan; kemampuan evaluasi melalui skenario.

Notasi objek yang dipakai pada tahap ini persis sama dengan notasi objek yang dipakai pada OOA.

Kegiatan pada tahap ini meliputi : perancangan komponen ranah masalah (PDC); perancangan komponen antarmuka pemakai (HIC); perancangan komponen manajemen data (DMC); dan perancangan komponen manajemen tugas (TMC).

### 2.2.4.1 Tahap Perancangan Komponen Ranah Masalah (PDC)

Kesatuan dari kelima lapisan hasil analisis pada OOA merupakan bagian pokok dari komponen ini (PDC). Pada tahap ini, perancang masih perlu memperhatikan kemungkinan perubahan terhadap hasil keseluruhan analisis. Perubahan itu dimungkinkan sebagai upaya untuk menyesuaikan dengan dukungan teknologi tanpa harus mengubah ranah masalah.

Perubahan yang mungkin diperlukan pada tahap ini seperti menambah, mengkombinasikan atau memecah kelas, kelas & objek, struktur, atribut, dan layanan yang telah didefinisikan pada tahapan analisis.

Hasil akhir pada tahap ini adalah sebuah diagram objek yang menyatakan ranah masalah. Diagram itu tidak lain merupakan gabungan kelima lapisan hasil analisis yang telah dilakukan perubahan secukupnya guna menyesuaikan rancangan dengan dukungan teknologi.

### 2.2.4.2 Tahap Perancangan Komponen Antarmuka Pemakai (HIC)

Pada tahap ini perancang perlu mendefinisikan kelas dan kelas & objek yang berfungsi sebagai antar-muka sistem dengan pemakai.

Perancang perlu mendefinisikan interaksi pemakai-sistem yang akan diakomodasikan pada perangkat lunak yang sedang dibangun. Peter Coad dan Edward Yourdon [CYo-91,p.57] memberikan strategi yang perlu diperhatikan oleh perancang pada tahap ini, yaitu:

klasifikasikan individu pemakai sistem; deskripsikan individu dan skenario tugasnya; rancang hirarki perintah / tugas; rancang interaksi yang lebih detil; buat prototipe; rancang kelas dan kelas & objek HIC; rancang *graphical user interfaces* (jika dipakai).

Hasil akhir pada tahap ini adalah sebuah diagram objek yang menyatakan rancangan komponen antarmuka. Meski dapat dipisahkan sebagai sebuah diagram lepas, namun sebagai satu tahapan perancangan , diagram tersebut tidak dapat lepas dari diagram objek pada komponen lainnya seperti PDC, DMC, dan TMC.

### 2.2.4.3 Tahap Perancangan Komponen Manajemen Data (DMC)

Pada tahap ini perancang perlu mendefinisikan kelas dan kelas & objek pengelola data yang langsung mendukung ranah masalah. Pengelolaan data yang terkait langsung dengan kelas dan kelas & objek bukan menjadi tugas dari komponen ini. Hal itu merupakan tugas dari kelas atau kelas & objek yang bersangkutan, dan telah terbungkus dalam kelas atau kelas & objek tersebut.

Proses merancang komponen ini membutuhkan rancangan struktur data, dan rancangan struktur berkas eksternal, serta membutuhkan rancangan layanan yang terkait dengan rancangan data dan rancangan struktur berkas eksternal.

Hasil akhir pada tahap ini adalah sebuah diagram objek yang menyatakan rancangan komponen manajemen data. Meski dapat dipisahkan sebagai sebuah diagram lepas, namun sebagai satu tahapan

perancangan, diagram tersebut tidak dapat lepas dari diagram objek pada komponen lainnya seperti PDC, HIC, dan TMC.

#### 2.2.4.4 Tahap Perancangan Komponen Manajemen Tugas (TMC)

Pada tahap ini perancang mendefinisikan kelas dan kelas & objek pengelola layanan yang langsung mendukung ranah masalah. Perancangan komponen ini sangat diperlukan untuk mengelola layanan-layanan yang telah didefinisikan pada PDC yang bersifat: *event-driven*; *clock-driven*; kritis dan butuh pengaturan prioritas.; layanan jamak.

Hasil akhir pada tahap ini adalah sebuah diagram objek yang menyatakan rancangan komponen manajemen layanan. Meski dapat dipisahkan sebagai sebuah diagram lepas, namun sebagai satu tahapan perancangan, diagram tersebut tidak dapat lepas dari diagram objek pada komponen lainnya seperti PDC, HIC, dan DMC.

### 2.2.5 OOP pada Delphi

Delphi merupakan bahasa pemrograman yang memakai Object-Pascal sebagai bahasa dasarnya. Bahasa ini menyediakan fasilitas pemrograman berorientasi objek, dan memiliki lingkungan pengembangan aplikasi yang cepat (*Rapid Application Development* dilingkat RAD).

Lingkungan RAD memberikan kesempatan pemrogram dapat menggunakan alat bantu lebih intuitif dan visual. Pemrogram Delphi dapat membuat sebuah aplikasi dengan memanipulasi layar secara langsung. Langkah yang dilakukan oleh pemrogram adalah: memilih

komponen *visual* yang telah disediakan Delphi, meletakkan dan mengatur komponen itu pada *form*, memilih aksi yang akan dikenakan kepada komponen itu, dan barulah menyunting kode aksi yang dilakukan.

Pada saat selesai proses penyuntingan kode dan penyimpanan kode pada media rekam, Delphi akan membangun berkas utama yaitu: berkas projek (.DPR); berkas teks program Pascal (.PAS); berkas pendefinisian form (.DFM).

Pada saat selesai melakukan kompilasi, Delphi akan membangun berkas hasil kompilasi (.DCU). Sedangkan setelah proses *linking* selesai dilakukan, Delphi akan membentuk berkas *executable* (.EXE) atau berkas *dynamic-link-library* (.DLL). Proses tersebut dilakukan oleh pemrogram dalam satu lingkungan terintegrasi (*Integrated Development Environment* disingkat IDE).

Seperti Object-Pascal, Delphi merupakan bahasa pemrograman yang mendukung OOP. Konsep fundamental dari sebuah bahasa OOP meliputi: *classes*; pewarisan, dan *polymorphism* [CaM, p.146]. Ketiga konsep tersebut diakomodasikan oleh Object-Pascal.

Sebuah kelas merupakan tipe data yang didefinisikan oleh pemakai, yang memiliki beberapa data internal dan beberapa *method* (berupa fungsi atau prosedur). Sebuah objek (*class instance*) merupakan sebuah variabel bertipe sesuai dengan tipe yang didefinisikan pada kelas. Objek mengandung sebuah referensi atau pointer yang menunjuk pada

lokasi memori dimana objek disimpan. Nilai data yang disimpan di dalam objek, disebut sebagai *object instance* , dibangun secara manual.

Kelas dan objek pada Delphi didukung oleh mekanisme jangkauan akses terhadap atribut dan layanan, yaitu: *private*; *protected*; *public*, dan mekanisme pewarisan.

Proses pewarisan dari sebuah tipe cukup dilakukan oleh pemrogram dengan menuliskan tipe tersebut pada saat mendefinisikan sebuah kelas.

Prinsip *polymorphism* memberikan fasilitas bagi pemrogram untuk memberi sebuah nama layanan untuk beberapa layanan yang kelakuannya tergantung pada masing-masing objek Prinsip ini didukung oleh Delphi melalui metoda *dynamic-binding* (*late-binding*). Kompiler dan Linker Delphi akan memberi alamat memori yang menyatakan lokasi dimana fungsi atau prosedur berada, pada saat *run-time*.

Berikut ini akan disajikan cara mendefinisikan kelas dan objek, menginisialisasi *object instance*, mendefinisikan atribut, mendefinisikan layanan, merealisasikan pewarisan, merealisasikan asosiasi, merealisasikan *message connection*, dalam bahasa Delphi.

### 2.2.5.1 Mendefinisikan Kelas dan Objek

Sebuah kelas didefinisikan sebagai sebuah tipe data dan mengikuti sintaks berikut:

```
type
    kelas = class
    end;
```

Sebuah objek dari kelas didefinisikan sebagai sebuah variabel dan mengikuti sintaks berikut:

```
var
    objek : kelas;
```

### 2.2.5.2 Menginisialisasi *Object Instance*

Sebuah *instance* dari objek didefinisikan sebagai berikut:

```
begin
    objek := kelas.Create;
    ..... {bagian instansiasi atribut }
end;
```

Jika sebuah *object instance* sudah tidak lagi digunakan, maka *object instance* tersebut sebaiknya harus dikembalikan kepada sistem operasi. Sintaks yang dipakai untuk keperluan tersebut adalah:

```
begin
    objek.Free;
end;
```

### 2.2.5.3 Mendefinisikan Atribut

Atribut dari sebuah kelas didefinisikan pada bagian definisi kelas yang terletak di antara nama kelas dan pernyataan "**end**". Pendefinisian atribut mengikuti sintaks sebagai berikut:

```
type
    kelas = class
```

```
    atribut : tipeatribut;
end;
```

Catatan: tipeatribut merupakan tipe data yang telah terdefinisi baik oleh Delphi maupun didefinisikan oleh pemrogram.

### 2.2.5.4 Mendefinisikan Layanan

Layanan dari sebuah kelas didefinisikan pada bagian definisi kelas yang terletak di antara nama kelas dan pernyataan "**end**", dan berada di bawah atribut. Layanan diwujudkan dalam bentuk prosedur atau fungsi. Pendefinisian layanan mengikuti sintaks sebagai berikut:

```
type
  kelas = class
    atribut : tipeatribut;
    layanan;
  end;
```

### 2.2.5.5 Merealisasikan Pewarisan (*Inheritance*)

Sebuah kelas A yang akan mewarisi seluruh atribut dan seluruh layanan dari kelas B dinyatakan sebagai berikut:

```
type
  A = class(B)
  end;
```

### 2.2.5.6 Merealisasikan Asosiasi (*Association*)

Asosiasi antara 2 buah objek dari 2 buah kelas berbeda dinyatakan pada pendefinisian masing-masing kelas dengan menggunakan penduplikasian atribut.

Objek a dari kelas A berasosiasi dengan objek b dari kelas B, dalam bentuk asosiasi berhingga, misalkan asosiasi 2-3, dinyatakan sebagai berikut:

```
type
  B = class
  A = class
    atribut_a : tipeatribut;
     b1       : B;
     b2       : B;
  end;
  B = class
    atribut_b : tipeatribut;
     a1       : A;
     a2       : A;
     a3       : A;
  end;
```

### 2.2.5.7 Merealisasikan *Message Connection*

*Message connection* antara 2 buah objek pada umumnya direalisasikan dengan menggunakan pemanggilan layanan (*method call*).

Pemanggilan layanan dari sebuah objek dilakukan dengan mengikuti sintaks berikut:

namaobjek.namalayanan[(daftar parameter)]

Bagian sintaks yang diapit oleh tanda '[' dan ']' disesuaikan dengan realisasi layanan. Jika layanan tidak merealisasikan parameter, maka bagian yang diapit kedua tanda kurung siku tidak diperlukan. Namun, jika layanan merealisasikan sejumlah parameter, maka bagian sintaks yang diapit oleh kedua tanda kurung siku perlu ditulis dan

seluruh parameter perlu dituliskan sesuai dengan definisi. Tanda '[' dan ']' tidak turut ditulis.

Bagian sintaks yang dicetak dengan huruf tebal merupakan kosa kata Delphi, sedangkan bagian sintaks yang lain merupakan kosa kata pemrogram.

## 2.3 Teori Perancangan Antarmuka Pemakai

Pada setiap pembangunan sistem perangkat lunak selalu terkait subsistem pemakai dan subsistem komputer. Kedua subsistem itu saling berinteraksi dalam kerangka sistem perangkat lunak.

Subsistem pemakai memiliki sifat relatif lebih luwes (*flexible*) dan lebih mudah beradaptasi (*adaptable*) daripada subsistem komputer. Perbedaan sifat tersebut seringkali menimbulkan gangguan interaksi yang pada akhirnya akan menurunkan unjuk kerja sistem perangkat lunak. Oleh karenanya, untuk menunjang interaksi pemakai dan komputer secara total dibutuhkan media yang mampu menunjang komunikasi antara pemakai dan komputer dengan baik. Media yang dimaksud adalah antarmuka-pemakai.

Deborah J. Mayhew [May-92] merinci sejumlah 15 buah prinsip umum yang mendasari perancangan antarmuka-pemakai. Ke-15 prinsip umum yang dimaksud adalah: kesesuaian pemakai; kesesuaian produk; kesesuaian tugas; kesesuaian aliran pekerjaan; konsistensi; keakraban; kesederhanaan; manipulasi langsung; kontrol; fleksibilitas; tanggapan cepat; teknologi tidak tampak; ketegaran; perlindungan; mudah dipelajari dan mudah dipakai.

Bagian berikut ini merupakan penjelasan singkat dari setiap prinsip di atas.

**Kesesuaian Pemakai (*User Compatibility*)**

Antarmuka-pemakai dirancang berdasarkan pemahaman terhadap calon pemakai perangkat lunak. Perancang antarmuka-pemakai tidak dibenarkan mendasarkan rancangannya pada asumsi bahwa: seluruh pemakai adalah sama, dan seluruh pemakai mirip dengan perancang. Mereka (para pemakai) adalah unik. Oleh karena itu, sebaiknya perancang cukup memahami dasar psikologi kognitif atau cukup memahami tentang kekuatan dan kelemahan pikiran pemakai.

**Kesesuaian Produk (*Product Compatibility*)**

Prinsip ini menghendaki agar antarmuka-pemakai dirancang dengan memanfaatkan pengetahuan / pengalaman yang telah dimiliki oleh para pemakai, dan meminimalkan kebutuhan pemakai untuk mempelajari sesuatu yang baru.

Pertimbangan terhadap prinsip ini muncul karena seringkali ditemui bahwa calon pemakai sebuah sistem baru telah menjadi pemakai dari sistem lain atau sistem serupa pada versi awal.

**Kesesuaian Tugas (*Task Compatibility*)**

Prinsip ini menghendaki agar antarmuka pemakai dirancang dengan memperhatikan keserasian terhadap struktur dan aliran sistem, serta mendukung tugas-tugas yang sedang dilakukan.

**Kesesuaian Aliran Pekerjaan (*Work Flow Compatibility*)**

Prinsip ini menghendaki agar antarmuka-pemakai dirancang dengan memperhatikan kebutuhan terhadap fasilitas perpindahan antar tugas yang dilakukan di dalam sistem. Fasilitas itu akan memudahkan pemakai untuk memberikan suatu tugas dan selanjutnya beralih pada tugas yang lain.

**Konsistensi (*Consistency*)**

Prinsip ini menghendaki agar antarmuka-pemakai dirancang secara konsisten, terutama pada pemakaian istilah. Konsistensi akan membuat pemakai menalar dengan cara analogi dan membuat perkiraan terhadap sesuatu yang baru, sehingga akan meminimalkan keinginan untuk selalu membuka manual.

**Keakraban (*Familiarity*)**

Prinsip ini menghendaki agar antarmuka-pemakai dirancang dengan memanfaatkan konsep, terminologi istilah dan susunan ruang yang telah akrab dengan pemakai.

**Kesederhanaan (*Simplicity*)**

Prinsip ini menghendaki kesederhanaan antarmuka-pemakai. Bagi sistem yang cukup kompleks, antarmuka-pemakai dirancang berlapis-lapis. Antarmuka-pemakai dirancang sesederhana mungkin pada setiap lapisan, sedemikian hingga pemakai tidak perlu mengetahui kompleksitas sistem. Selanjutnya, melalui pengaturan lapisan antarmuka -pemakai diharapkan memudahkan pemakai mempelajari ide dasar

sistem dan secara bertahap mempelajari fungsi-fungsi yang lebih kompleks dari sistem.

Salah satu contoh yang disarankan untuk membuat lapisan antarmuka-pemakai sederhana adalah dengan memanfaatkan nilai *default* . Nilai *default* dapat diganti sejauh diperlukan.

**Manipulasi Langsung (*Direct Manipulation*)**

Prinsip ini menghendaki agar antarmuka-pemakai disusun sedemikian hingga pemakai dapat memberikan aksi secara langsung pada objek yang nampak. Sebagai contoh, pada perangkat lunak pengolah kata, pemakai dapat menunjuk langsung bagian teks yang akan diperbaiki. Pemakai dapat melakukan perbaikan pada bagian tersebut tanpa harus direpotkan dengan mengisi sejumlah parameter yang diperlukan komputer untuk menanggapi aksi pemakai.

**Kontrol (*Control*)**

Prinsip ini menghendaki agar antarmuka-pemakai disusun sedemikian hingga cepat menimbulkan rasa pengusaan (*sense of mastery*) dari pemakai terhadap sistem. Untuk mendukung hal itu, antarmuka-pemakai harus disusun sederhana, dapat diperkirakan, dan konsisten.

Antarmuka-pemakai yang membuat para pemakai dikendalikan dan diarahkan oleh mesin hanyalah akan membuat para pemakainya merasa frustasi. Sebaliknya, antarmuka-pemakai seharusnya disusun

sehingga pemakai merasa dapat mengendalikan dan mengarahkan mesin.

**Fleksibilitas (*Flexibility*)**

Prinsip ini menghendaki agar antarmuka-pemakai disusun dengan memperhatikan keluwesan pada pemberian pilihan kepada pemakai. Keluwesan yang dimaksud berkaitan dengan penyediaan lebih banyak pilihan cara mengontrol sistem, dan mengakomodasikan berbagai variasi ketrampilan serta minat pemakai.

**Tanggapan Cepat (*Responsiveness*)**

Prinsip ini menghendaki agar antarmuka pemakai disusun dengan memperhatikan respon cepat yang harus dilakukan oleh komputer. Jika respon komputer terpaksa tidak dapat diberikan dalam waktu singkat, maka pesan dan kemajuan proses pengerjaan oleh komputer harus disampaikan kepada pemakai melalui antarmuka-pemakai.

**Teknologi Tidak Tampak (*Invisible Technology*)**

Prinsip ini menghendaki agar antarmuka-pemakai disusun sedemikian hingga pemakai tidak perlu mengetahui detil teknis tentang bagaimana sistem diimplementasikan serta beroperasi. Sedapat mungkin antarmuka-pemakai menyembunyikan teknologi, dan hanyalah menghadirkan fungsi-fungsi yang langsung dibutuhkan oleh pemakai.

**Ketegaran (*Robustness*)**

Prinsip ini menghendaki agar antarmuka-pemakai dirancang dengan memperhatikan bahwa sistem harus cukup tegar untuk menangani setiap masukan dari pemakai, termasuk di dalamnya adalah kesalahan manusia (*human error*). Kesalahan apapun diharapkan tidak akan menjadikan sistem menggantung (*hang*), namun tetap memberikan kemudahan solusi penyembuhan yang dilakukan oleh pemakai.

**Perlindungan (*Protection*)**

Prinsip ini menghendaki agar antarmuka-pemakai disusun dengan memperhatikan bahwa pemakai harus dilindungi dari akibat fatal yang timbul karena kesalahan aksi yang diberikan kepadanya. Sebagai contoh, pemberian fasilitas "*UNDO*" untuk membatalkan aksi yang terlanjur diberikan oleh pemakai. Prosedur penyembuhan harus disusun sesederhana dan semudah mungkin bagi pemakai.

**Mudah Dipelajari dan Mudah Digunakan (*Ease of Learning and Ease of Use*)**

Prinsip ini menghendaki agar antarmuka-pemakai dirancang dengan memperhatikan bahwa sistem harus mudah dipelajari oleh pemakai baru dan tetap berdayaguna (*efficient*) serta mudah dipakai oleh pemakai ahli.

Sasaran yang akan dituju dari sebagian besar prinsip di atas merupakan besaran kualitatif, seperti: mudah dipakai; mudah dipelajari; ketegaran; fleksibel; dan lain-lain. Hal itu berarti bahwa ukuran

pencapaian sasaran tidak dapat ditentukan kuantitasnya, namun dapat dinyatakan dalam skala kualitas.

Perancang antarmuka-pemakai diharapkan cukup cermat mengakomodasikan prinsip-prinsip di atas. Upaya untuk selalu mengakomodasikan secara optimal seluruh prinsip di atas seringkali menimbulkan kesulitan bagi perancang antarmuka-pemakai. Pada saat perancangan antarmuka-pemakai dilakukan tidak jarang akan timbul konflik berkaitan dengan optimasi prinsip. Sebagai contoh, untuk memaksimalkan prinsip mudah digunakan (*ease of use*) mungkin akan mengorbankan atau sekurangnya mengurangi kesesuaian produk antar versi. Memaksimalkan kontrol mungkin akan mengorbankan atau sekurangnya mengurangi prinsip kemudahan untuk dipelajari.

Perancang antarmuka-pemakai akhirnya harus melakukan tawar-menawar antar prinsip secara seksama. Upaya ini harus didasarkan pada pemahaman yang cukup cermat terhadap populasi calon pemakai.

## 2.4 Teori Automata

*Automata* adalah bentuk jamak dari *Automaton*. *Automaton* merupakan sebuah model matematis dari sebuah sistem yang memiliki sejumlah masukan-farik dan dengan atau tanpa keluaran-farik. Model tersebut memiliki: sejumlah kondisi (*state*); sejumlah masukan dan keluaran yang dikenalinya; fungsi yang mengendalikan kondisi; dan dengan atau tanpa fungsi yang mengendalikan keluaran.

*Automaton* yang memiliki: sejumlah berhingga kondisi; sejumlah berhingga masukan; dan sejumlah berhingga keluaran (jika ada), disebut sebagai *Automaton* Berhingga (*Finite Automaton*).

Terkait dengan teori *Automaton* yang dipakai pada jaringan dinamis, berikut penulis sajikan: definisi *Automaton* Berhingga, berikut sebuah contoh ; definisi Mealy *Automaton*, berikut sebuah contoh; serta rangkaian Mealy *Automaton*.

### 2.4.1 Definisi *Automaton* Berhingga

Sebuah *Automaton* Berhingga didefinisikan sebagai sebuah 5-tupel $(Q,\Sigma,\delta,q_0,F)$, dengan : $Q$ = himpunan berhingga kondisi; $\Sigma$ = himpunan berhingga abjad; $q_0 \in Q$ dan $q_0$ adalah kondisi-awal ; $F \subseteq Q$ dan $F$ = himpunan kondisi-akhir ; dan $\delta$ adalah sebuah fungsi transisi dari $Q \times \Sigma$ ke $Q$ ($\delta : Q \times \Sigma \rightarrow Q$). [HUI-79, p.17].

Contoh, sebuah *Automaton* $FA = (Q,\Sigma,\delta,q_0, F)$, dengan: $Q = \{ q_0, q_1 \}$ ; $\Sigma = \{ n, y \}$; $F = \{ q_0 \}$; dan $\delta$ sebuah fungsi yang ditabulasikan sebagai berikut:

**Tabel 2.1 Fungsi δ pada *Automaton* Berhingga**

| Kondisi (q) | Masukan (i) | δ(q,i) |
|---|---|---|
| $q_0$ | n | $q_0$ |
| $q_0$ | y | $q_1$ |
| $q_1$ | n | $q_1$ |
| $q_1$ | y | $q_0$ |

Pada perkembangan selanjutnya, himpunan masukan dan keluaran tidak harus berupa himpunan abjad. Kedua himpunan itu harus merupakan himpunan berhingga dari elemen-elemen yang terkait dengan kondisi sistem yang sedang dimodelkan.

### 2.4.2 Definisi Mealy *Automaton*

Mealy *Automaton* merupakan *Automaton* berhingga yang memiliki keluaran.

Sebuah Mealy *Automaton* didefinisikan sebagai sebuah 5-tupel $(Z,X,Y,\delta,\lambda)$ , dengan: Z = himpunan berhingga kondisi; X = himpunan berhingga masukan; Y = himpunan berhingga keluaran; $\delta$ adalah sebuah fungsi dari Z x X ke Z ($\delta$: ZxX$\rightarrow$Z); $\lambda$ adalah sebuah fungsi dari Z x X ke Y ( $\lambda$: Z x X $\rightarrow$ Y). [Coh-77, p.109]

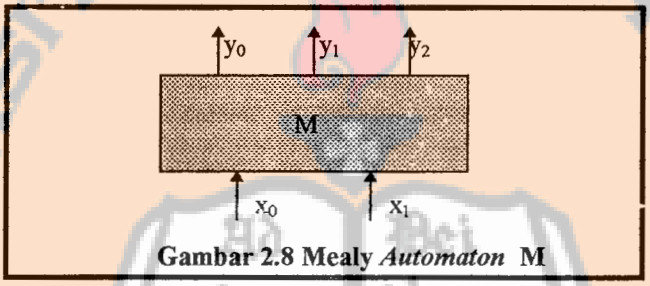
Contoh, sebuah Mealy *Automaton* M = $(Z,X,Y,\delta,\lambda)$, dengan : $Z = \{z_0,z_1,z_2,z_3\}$; $X = \{x_0,x_1\}$; $Y = \{y_0,y_1,y_2\}$; fungsi $\delta$ dan $\lambda$ yang memiliki aturan fungsi sebagai berikut:

**Tabel 2.2 Fungsi $\delta$ dan $\lambda$ pada Mealy *Automaton***

| Kondisi (z) | Masukan (i) | $\delta(z,i)$ | $\lambda(z,i)$ |
|---|---|---|---|
| $z_0$ | $x_0$ | $z_0$ | $y_0$ |
| $z_0$ | $x_1$ | $z_1$ | $y_2$ |
| $z_1$ | $x_0$ | $z_0$ | $y_1$ |
| $z_1$ | $x_1$ | $z_2$ | $y_2$ |
| $z_2$ | $x_0$ | $z_1$ | $y_1$ |
| $z_2$ | $x_1$ | $z_3$ | $y_2$ |
| $z_3$ | $x_0$ | $z_2$ | $y_1$ |
| $z_3$ | $x_1$ | $z_3$ | $y_2$ |

Sebuah Mealy *Automaton* dapat digambarkan sebagai sebuah kotak hitam yang memiliki sejumlah masukan, sejumlah keluaran, dan di dalamnya tersimpan sejumlah berhingga kondisi, fungsi yang mengatur transisi antar kondisi, dan fungsi yang mengatur keluaran.

Mealy *Automaton* M pada contoh di atas dapat digambarkan sebagai berikut:




**Gambar 2.8 Mealy *Automaton* M**

### 2.4.3 Rangkaian Mealy *Automaton*

Antar Mealy *Automaton* dapat dilakukan upaya merangkai menjadi sebuah Mealy *Automaton* baru. Cohors mendefinisikan 3 buah jenis rangkaian, yaitu: rangkaian paralel (*parallelschaltung*); rangkaian kopling-balik (*rueckkopplung*); rangkaian seri (*hintereinander-schaltung*). [Coh-79, p.113 - 115]

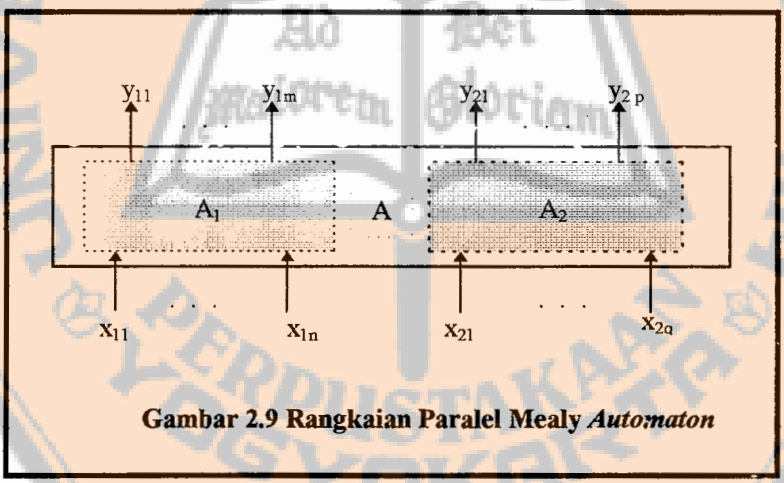
#### 2.4.3.1 Rangkaian Paralel

Andaikan $A_1 = ( Z_1, X_1, Y_1, \delta_1, \lambda_1 )$,dan $A_2 = ( Z_2, X_2, Y_2, \delta_2, \lambda_2 )$ masing-masing adalah Mealy *Automaton*. Rangkaian paralel dari $A_1$ dan $A_2$ disajikan sebagai $(A_1 \| A_2)$, dan merupakan sebuah Mealy *Automaton* $( Z_1 \times Z_2, X_1 \cup X_2, Y_1 \cup Y_2, \delta, \lambda)$ dengan $\delta$ serta $\lambda$ yang didefinisikan sebagai berikut:

$$\delta((z_1,z_2),x)=\begin{cases}(\delta_1(z_1,x),z_2) & ,untuk\,(z_1,x)\in D_{\delta_1}\\ (z_1,\delta_2(z_2,x) & ,untuk(z_2,x)\in D_{\delta_2}\\ tidak\ didefinisikan & ,selain\ syarat\ di\ atas\end{cases}$$

$$\lambda((z_1,z_2),x)=\begin{cases}\lambda_1(z_1,x) & ,untuk\,(z_1,x)\in D_{\lambda_1}\\ \lambda_2(z_2,x) & ,untuk(z_2,x)\in D_{\lambda_2}\\ tidak\ didefinisikan & ,selain\ syarat\ di\ atas\end{cases}$$

Jika rangkaian paralel antara Mealy *Automaton* $A_1$ dan $A_2$ dinyatakan dengan gambar, maka hasil rangkaian $A = (A_1 \,||\, A_2)$ seperti tampak pada gambar berikut.




**Gambar 2.9 Rangkaian Paralel Mealy *Automaton***

### 2.4.3.2 Rangkaian Kopling-balik (*Rueckkopplung*)

Andaikan A = ( Z, X, Y, δ, λ ) adalah Mealy *Automaton* dengan $X = \{x_1,x_2,\ldots,x_n\}$, $Y = \{ y_1,y_2,\ldots,y_m\}$, dan $m,n \geq 1$.

Kopling-balik (*rueckkopplung*) keluaran $y_j$ pada masukan $x_i$ dari Mealy *Automaton* A, disajikan sebagai:

$A_{x_i}^{y_j} := (Z, X - \{x_i\}, Y - \{y_j\}, \delta_i^j, \lambda_i^j)$, adalah sebuah Mealy *Automaton* dengan fungsi $\delta_i^j$ dan $\lambda_i^j$ yang memenuhi syarat:

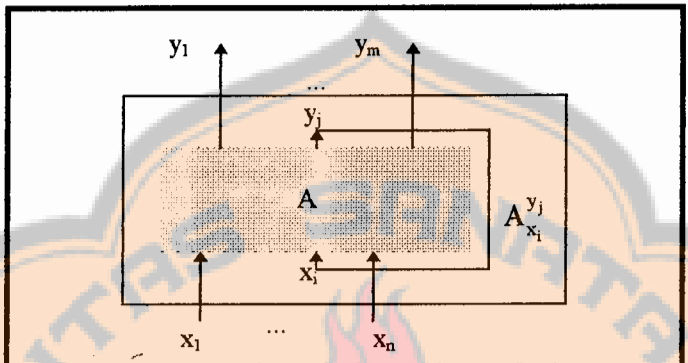
1. Jika untuk $z \in Z$, $x \in X - \{x_i\}$ nilai $\delta(z,x)$ dan $\lambda(z,x)$ tidak didefinisikan, maka $\delta_i^j(z,x)$ dan $\lambda_i^j(z,x)$ juga tidak didefinisikan.

2. Jika untuk $z \in Z$, $x \in X - \{x_i\}$ berlaku $\lambda(z,x) \neq y_j$, maka $\delta_i^j(z,x) \cong \delta(z,x)$ dan $\lambda_i^j(z,x) \cong \lambda(z,x)$.

3. Jika untuk $z \in Z$, $x \in X - \{x_i\}$ berlaku $\lambda(z,x) = y_j$, dan terdapat sejumlah berhingga barisan kondisi A, yaitu: $z_1, \ldots, z_{r-1}$, yang memiliki sifat: $z_1 := \delta(z,x)$ dan $z_{\rho+1} := \delta(z_\rho, x_i)$ untuk $1 \leq \rho \leq r$; berlaku $\lambda(z_\rho, x_i) = y_j$ untuk $1 \leq \rho < r$ dan $\lambda(z_r, x_i) \neq y_j$, maka $\delta_i^j(z,x) := z_{r+1}$ dan $\lambda_i^j(z,x) := \lambda(z_r, x)$.

4. Jika kondisi tidak memenuhi kondisi di atas, maka $\delta_i^j(z,x)$ dan $\lambda_i^j(z,x)$ tidak didefinisikan.

Jika rangkaian kopling-balik keluaran $y_j$ pada masukan $x_i$ dari Mealy *Automaton* A dinyatakan dengan gambar, maka hasil rangkaian seperti tampak pada gambar 2.10.

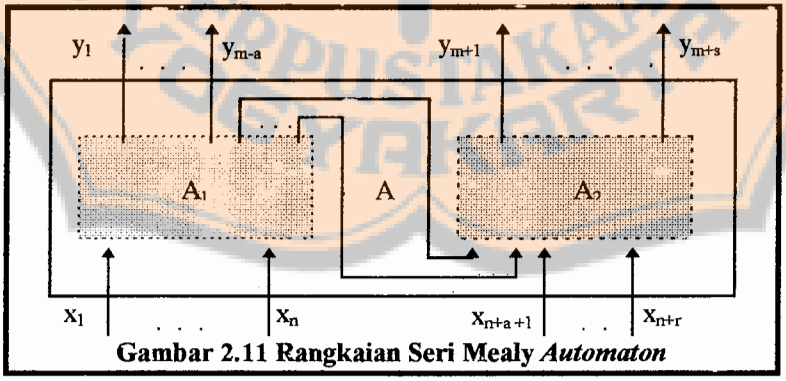



**Gambar 2.10 Rangkaian Kopling-balik Mealy *Automaton***

### 2.4.3.3 Rangkaian Seri

Andaikan $A_1 = ( Z_1, \{x_1,..., x_n\}, \{y_1,...,y_m\}, \delta_1, \lambda_1 )$, dan $A_2 = ( Z_2, \{x_{n+1},...,x_{n+r}\}, \{y_{m+1},...,y_{m+s}\}, \delta_2, \lambda_2 )$ masing-masing adalah Mealy *Automaton*. Rangkaian seri $A_1$ dan $A_2$ didefinisikan sebagai rangkaian yang mengkombinasikan rangkaian paralel dan rangkaian kopling-balik. Hasil rangkaian merupakan sebuah Mealy *Automaton* baru.

Jika rangkaian seri antara Mealy *Automaton* $A_1$ dan $A_2$ , yaitu A, dinyatakan dengan gambar, maka hasil rangkaian seperti tampak pada gambar berikut.




**Gambar 2.11 Rangkaian Seri Mealy *Automaton***

## 2.5 Jaringan Dinamis

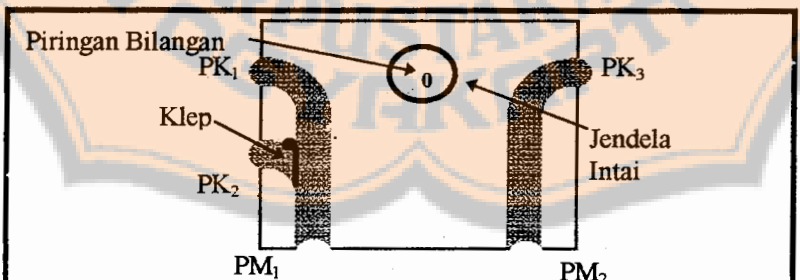
Jaringan dinamis merupakan seperangkat alat didaktik yang menunjukkan model Mealy *Automaton*. Alat tersebut dirancang dan dikembangkan oleh Prof. DR. Elmar Cohors-Fresenborg dari Universitaet Osnabrueck di Jerman.

Sebuah jaringan dinamis terdiri dari sekumpulan komponen yang dirangkai secara paralel, atau kopling-balik, atau seri. Jaringan tersebut dapat dipandang sebagai sebuah mesin otomat yang mengerjakan tugas manipulasi tertentu (terbatas operasi Aritmatika) terhadap bilangan cacah. Contoh dari sebuah jaringan dinamis dapat dilihat pada lampiran A.

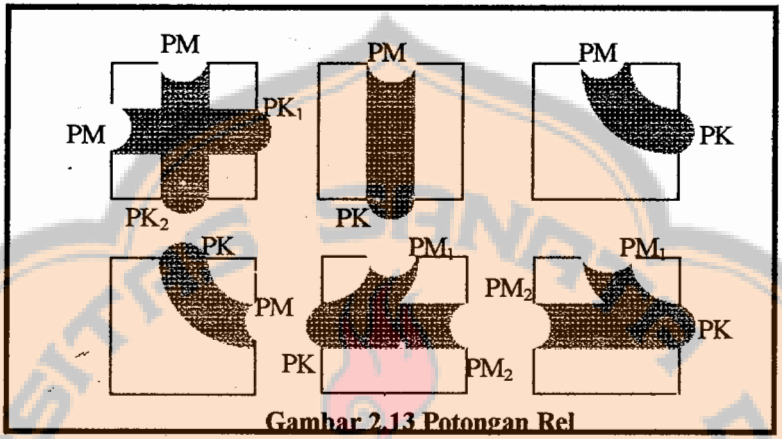
Berikut ini penulis akan menyajikan : komponen penyusun jaringan dinamis; mekanisme kerja dari setiap komponen; dan pemakaian konsep Mealy *Automaton* pada setiap komponen.

### 2.5.1 Komponen Jaringan Dinamis

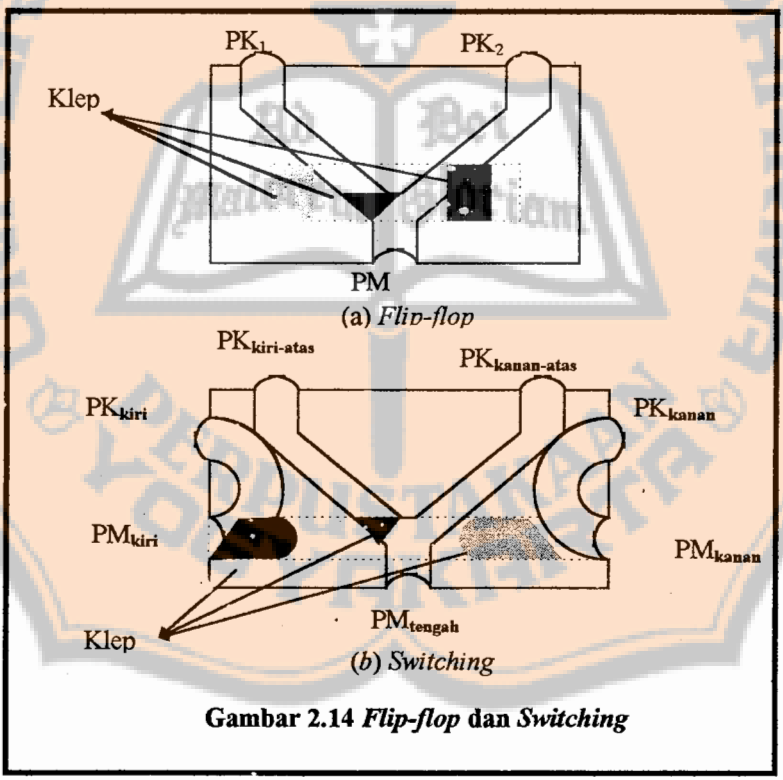
Komponen yang menyusun jaringan dinamis terdiri dari: *counter*; potongan-rel, *flip-flop*, dan *swicthing*. Gambar dari setiap komponen adalah sebagai berikut.




**Gambar 2.12 *Counter***




Gambar 2.13 Potongan Rel




**Gambar 2.14 *Flip-flop* dan *Switching***

***Counter***

*Counter* memiliki beberapa komponen yang cukup penting untuk diperhatikan. Komponen tersebut adalah: 2 buah pintu masuk, yaitu $PM_1$ dan $PM_2$; 3 buah pintu ke luar, yaitu $PK_1$ , $PK_2$, dan $PK_3$; 1 buah klep; 1 buah piringan bilangan (tertutup); serta 1 buah jendela-intai (lihat gambar 2.12).

Pintu masuk ditandai dengan lekukan ke arah dalam (disebut: ***nase***), sedangkan pintu ke luar ditandai dengan lekukan ke arah luar (disebut: ***loch***).

**Potongan Rel**

Terdapat 6 macam bangun potongan rel (lihat gambar 2.13), dan setiap potongan rel dikemas pada sebuah bujursangkar. Pada setiap potongan rel terdapat satu atau beberapa pintu masuk dan pintu ke luar.

Dua buah potongan rel dapat dihubungkan dengan cara melekatkan pintu ke luar potongan rel yang satu dengan pintu masuk potongan rel yang lain.

Potongan-potongan rel tersebut selanjutnya dihubungkan sedemikian hingga membentuk sebuah lintasan, seperti halnya lintasan rel kereta-api. Lintasan tersebut dibangun untuk menghubungkan pintu-pintu (pintu-masuk dan atau pintu ke luar) dari sekumpulan *counter / flip-flop / switching*, sehingga membentuk sebuah jaringan dinamis.

***Flip-flop***

*Flip-flop* memiliki: 1 buah pintu masuk, yaitu PM; 2 buah pintu ke luar, yaitu $PK_1$ dan $PK_2$ ; dan 1 buah klep (lihat gambar 2.14.a).

***Switching***

*Switching* memiliki 3 buah pintu masuk, 4 buah pintu ke luar, dan 1 buah klep. Ketiga pintu masuk tersebut adalah: pintu masuk kiri ($PM_{kiri}$); pintu masuk tengah ($PM_{tengah}$); dan pintu masuk kanan ($PM_{kanan}$).

Keempat pintu ke luar tersebut adalah: pintu ke luar kiri ($PK_{kiri}$); pintu ke luar kanan ($PK_{kanan}$); pintu ke luar kiri atas ($PK_{kiri\text{-}atas}$); pintu ke luar kanan atas ($Pk_{kanan\text{-}atas}$). Komponen *switching* dapat dilihat pada gambar 2.14.b.

## 2.5.2 Mekanisme Kerja Komponen Jaringan Dinamis

Pada seluruh komponen jaringan dinamis berlaku aturan bahwa jika memasuki komponen maka harus melalui pintu masuk, demikian juga jika akan meninggalkan komponen maka harus melalui pintu ke luar.

Berikut ini disajikan mekanisme kerja dari setiap komponen jaringan dinamis. Mekanisme ini menjadi dasar bagi mesin otomat yang akan dibentuk melalui pembentukan jaringan dinamis.

**Mekanisme Kerja *Counter***

- Jika kita memasuki mesin melalui $PM_2$ , maka kita akan ke luar dari mesin melalui $PK_3$ . Sebelum ke luar, piringan bilangan akan berputar berlawanan arah putaran jarum jam, sehingga bilangan yang terlihat pada jendela intai akan bertambah 1.

- Jika kita memasuki mesin melalui $PM_1$ , maka kita akan ke luar melalui tepat salah satu pintu ke luar $PK_1$ atau $PK_2$.

  Jika pada saat itu bilangan yang terlihat pada jendela intai adalah 0, maka klep akan membuka dan saluran menuju $PK_1$ akan tertutup. Dengan demikian, kita dipaksa untuk ke luar melalui $PK_2$ dan tanpa memutar piringan bilangan.

  Jika pada saat itu bilangan yang terlihat pada jendela intai tidak sama dengan 0, maka klep akan tetap menutup dan saluran menuju $PK_2$ akan tertutup. Dengan demikian, kita dipaksa untuk ke luar melalui $PK_1$ . Sebelum ke luar dari mesin melalui $PK_1$ piringan bilangan akan berputar searah putaran jarum jam, dan bilangan yang terlihat pada jendela intai akan berkurang 1.

**Mekanisme Kerja Potongan Rel**

- Jika kita masuk melalui sebuah pintu masuk dari sebuah potongan rel, maka kita akan ke luar melalui tepat 1 pintu ke luar yang sesuai dengan saluran. Kita tidak dapat ke luar melalui pintu ke luar yang lain.

**Mekanisme Kerja *Flip-flop***

- Jika kita masuk melalui pintu masuk PM dan ke luar melalui pintu ke luar ke-2 ($PK_2$), maka secara otomatis klep segitiga akan menutup saluran menuju $PK_2$. Dengan demikian saluran yang lain, yaitu saluran menuju $PK_1$ akan terbuka.

- Jika kita masuk melalui pintu masuk PM dan ke luar melalui pintu ke luar ke-1 ($PK_1$), maka secara otomatis klep segitiga akan menutup saluran menuju $PK_1$. Dengan demikian saluran yang lain, yaitu saluran menuju $PK_2$ akan terbuka.

**Mekanisme Kerja *Switching***

- Jika kita masuk melalui $PM_{tengah}$ dan saluran menuju $PK_{kiri\text{-}atas}$ terbuka, maka kita akan ke luar melalui $PK_{kiri\text{-}atas}$.

  Kondisi saluran itu akan tetap apabila dari $PK_{kiri\text{-}atas}$ kita langsung memasuki kembali $PM_{tengah}$. Akan tetapi, jika kemudian kita masuk melalui $PM_{kanan}$ dan ke luar melalui $PK_{kanan}$, maka secara otomatis klep segitiga akan menutup saluran menuju $PK_{kiri\text{-}atas}$ dan saluran menuju $PK_{kanan\text{-}atas}$ akan terbuka.

- Jika kita masuk melalui $PM_{tengah}$ dan saluran menuju $PK_{kanan\text{-}atas}$ terbuka, maka kita akan ke luar melalui $PK_{kanan\text{-}atas}$.

  Kondisi saluran itu akan tetap apabila dari $PK_{kanan\text{-}atas}$ kita langsung memasuki kembali $PM_{tengah}$. Akan tetapi, jika kemudian kita masuk melalui $PM_{kiri}$ dan ke luar melalui $PK_{kiri}$, maka secara otomatis klep segitiga akan menutup saluran menuju $PK_{kanan\text{-}atas}$ dan saluran menuju $PK_{kiri\text{-}atas}$ akan terbuka.

### 2.5.3 Mealy *Automaton* pada Komponen Jaringan Dinamis

Pada setiap komponen jaringan dinamis dapat didefinisikan: himpunan berhingga kondisi (himpunan Z); himpunan berhingga

masukan (himpunan X); himpunan berhingga keluaran (himpunan Y); fungsi $\delta$: Z x X $\rightarrow$ Z; dan fungsi $\lambda$: Z x X $\rightarrow$ Y. Dengan demikian, konsep Mealy *Automaton* diakomodasikan pada setiap komponen jaringan dinamis.

**Mealy *Automaton* pada *Counter***

Pendefinisian himpunan Z, himpunan X, himpunan Y, fungsi $\delta$ dan fungsi $\lambda$ sebagai berikut:

- Z = { 0,1,2,3,4,5, ..., n} , dengan n sebuah bilangan bulat positip berhingga.
- X = {$PM_1$, $PM_2$}. $PM_1$ dan $PM_2$ adalah kedua pintu masuk *counter*.
- Y = { $PK_1$, $PK_2$, $PK_3$}. $PK_1$, $PK_2$ dan $PK_3$ adalah ketiga pintu ke luar *counter*.
- $\delta$ adalah sebuah fungsi Z x X $\rightarrow$ Z yang memenuhi aturan sebagai berikut:

**Tabel 2.3 Fungsi Z x X $\rightarrow$ Z pada *Counter***

| Z | X | Z |
|---|---|---|
| 0 | PM1 | 0 |
| 0 | PM2 | 1 |
| 1 | PM1 | 0 |
| 1 | PM2 | 2 |
| . | . | . |
| n | PM1 | (n-1) |
| n | PM2 | n |

- λ adalah sebuah fungsi Z x X → Y yang memenuhi aturan sebagai berikut:

**Tabel 2.4 Fungsi Z x X → Y pada *Counter***

| Z | X | Y |
|---|---|---|
| 0 | $PM_1$ | $PK_2$ |
| 0 | $PM_2$ | $PK_3$ |
| 1 | $PM_1$ | $PK_1$ |
| 1 | $PM_2$ | $PK_3$ |
| . | . | . |
| n | $PM_1$ | $PK_1$ |
| n | $PM_2$ | $PK_3$ |

**Mealy *Automaton* pada Potongan Rel**

Pendefinisian himpunan Z, himpunan X, himpunan Y, fungsi δ dan fungsi λ sebagai berikut:

- Z = { c}, dengan c sebuah bilangan cacah (konstanta cacah).
- Rel Lurus dan Belok
  - X = {PM}. PM adalah pintu masuk potongan rel lurus atau rel belok.
  - Y = {PK}. PK adalah pintu ke luar potongan rel lurus atau rel belok.
  - δ adalah sebuah fungsi Z x X → Z, yaitu: δ(c,PM) = c.
  - λ adalah sebuah fungsi Z x X → Y, yaitu: λ(c,PM) = PK.

- Rel Simpang-3
  - X = {$PM_1$,$PM_2$}. $PM_1$ dan $PM_2$ adalah kedua pintu masuk rel simpang-3.
  - Y = {PK}. PK adalah pintu ke luar rel simpang-3.
  - $\delta$ adalah sebuah fungsi Z x X $\rightarrow$ Z, yaitu: $\delta(c,PM_1)= c$, dan $\delta(c,PM_2)= c$.
  - $\lambda$ adalah sebuah fungsi Z x X $\rightarrow$ Y, yaitu: $\lambda(c,PM_1)$=PK , dan $\lambda(c,PM_2)$=PK
- Rel Simpang-4
  - X = {$PM_1$,$PM_2$}. $PM_1$ dan $PM_2$ adalah kedua pintu masuk rel simpang-4.
  - Y = {$PK_1$,$PK_2$}. $PK_1$ dan $PK_2$ adalah kedua pintu ke luar rel simpang-4.
  - $\delta$ adalah sebuah fungsi Z x X $\rightarrow$ Z, yaitu: $\delta(c,PM_1)= c$, dan $\delta(c,PM_2)= c$.
  - $\lambda$ adalah sebuah fungsi Z x X $\rightarrow$ Y, yaitu: $\lambda(c,PM_1)= PK_1$ , dan $\lambda(c,PM_2)= PK_2$ .

**Mealy *Automaton* pada *Flip-flop***

Pendefinisian himpunan Z, himpunan X, himpunan Y, fungsi $\delta$ dan fungsi $\lambda$ sebagai berikut:

- Z = {$K_i$ , $K_a$}. $K_i$ = status klep segitiga menutup jalur kiri; $K_a$ = status klep segitiga menutup jalur kanan.
- X = {PM}. PM adalah pintu masuk flip-flop.
- Y = {$PK_1$, $PK_2$}. $PK_1$ dan $PK_2$ adalah kedua pintu ke luar *flip-flop*.
- δ adalah sebuah fungsi Z x X → Z yang memenuhi aturan sebagai berikut:

**Tabel 2.5 Fungsi Z x X → Z pada *Flip-flop***

| Z | X | Z |
|---|---|---|
| $K_i$ | PM | $K_a$ |
| $K_a$ | PM | $K_i$ |

- λ adalah sebuah fungsi Z x X → Y yang memenuhi aturan sebagai berikut:

**Tabel 2.6 Fungsi Z x X → Y pada *Flip-flop***

| Z | X | Y |
|---|---|---|
| $K_i$ | PM | $PK_2$ |
| $K_a$ | PM | $PK_1$ |

**Mealy *Automaton* pada *Switching***

Pendefinisian himpunan Z, himpunan X, himpunan Y, fungsi δ dan fungsi λ sebagai berikut:

- Z= {$K_i$ , $K_a$}. $K_i$ = status klep segitiga menutup jalur kiri; $K_a$= status klep segitiga menutup jalur kanan.

- $X = \{PM_{kiri}, PM_{tengah}, PM_{kanan}\}$. $PM_{kiri}$, $PM_{tengah}$, dan $PM_{kanan}$ adalah ketiga pintu masuk *switching*.
- $Y = \{PK_{kiri}, PK_{kiri\text{-}atas}, PK_{kanan\text{-}atas}, PK_{kanan}\}$. $PK_{kiri}$, $PK_{kiri\text{-}atas}$, $PK_{kanan\text{-}atas}$, dan $PK_{kanan}$ adalah keempat pintu ke luar *switching*.
- $\delta$ adalah sebuah fungsi $Z \times X \rightarrow Z$ yang memenuhi aturan sebagai berikut:

**Tabel 2.7 Fungsi $Z \times X \rightarrow Z$ pada *Switching***

| Z | X | Z |
|---|---|---|
| $K_i$ | $PM_{kiri}$ | $K_a$ |
| $K_i$ | $PM_{tengah}$ | $K_i$ |
| $K_i$ | $PM_{kanan}$ | $K_i$ |
| $K_a$ | $PM_{kiri}$ | $K_a$ |
| $K_a$ | $PM_{tengah}$ | $K_a$ |
| $K_a$ | $PM_{kanan}$ | $K_i$ |

- $\lambda$ adalah sebuah fungsi $Z \times X \rightarrow Y$ yang memenuhi aturan sebagai berikut:

**Tabel 2.8 Fungsi $Z \times X \rightarrow Y$ pada *Switching***

| Z | X | Y |
|---|---|---|
| $K_i$ | $PM_{kiri}$ | $PK_{kiri}$ |
| $K_i$ | $PM_{tengah}$ | $PK_{kanan\text{-}}$ |
| $K_i$ | $PM_{kanan}$ | $PK_{kan n}$ |
| $K_a$ | $PM_{kiri}$ | $PK_{kiri}$ |
| $K_a$ | $PM_{tengah}$ | $PK_{kiri\text{-}atas}$ |
| $K_a$ | $PM_{kanan}$ | $PK_{kanan}$ |

# BAB III
# PERANCANGAN PERANGKAT LUNAK DM

Pada bab ini penulis akan menyajikan hasil perancangan perangkat lunak DM yang berorientasi objek, yaitu meliputi hasil perancangan terhadap analisis (OOA), dan hasil perancangan terhadap desain (OOD).

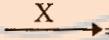
Deskripsi hasil perancangan akan didahului oleh bagian deskripsi umum. Bagian ini akan memberikan deskripsi singkat terhadap perangkat lunak yang akan dibangun.

## 3.1 Deskripsi Umum

Dari hasil studi lapangan diperoleh gambaran secara umum bahwa model sistem yang akan diakomodasikan oleh perangkat lunak mencakup 3 buah subsistem besar yang dapat dipisahkan. Ketiga subsistem tersebut adalah: subsistem sunting gambar jaringan dinamis (selanjutnya disebut subsistem sunting), subsistem penjelajahan jaringan dinamis (selanjutnya disebut subsistem jelajah), dan subsistem pemakai.

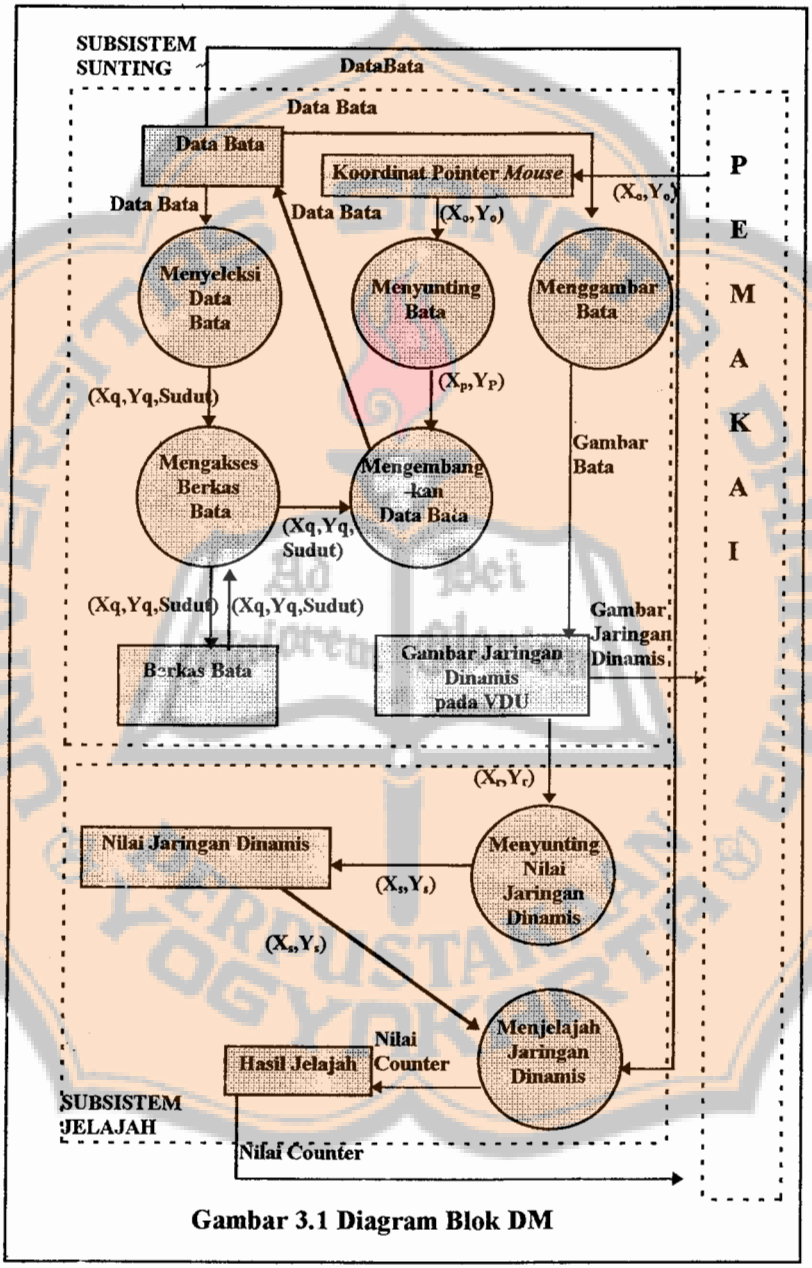
Deskripsi global dari ketiga subsistem tersebut akan disajikan pada gambar 3.1. Notasi yang dipakai bukan notasi objek. Notasi dan arti notasi yang dipakai adalah:

 = Masukan / keluaran

 = Aliran Data X

 = Proses

 = Subsistem




**Gambar 3.1 Diagram Blok DM**

Keterangan Diagram:

- (Xo,Yo) : koordinat pointer *mouse* yang menunjuk sebuah titik pada papan panel
- (Xp,Yp) : koordinat titik terpilih pada papan panel
- DataBata : sebuah *record* yang berisi nilai dari seluruh atribut *instance* sebuah bata.
- (Xq,Yq,Sudut) : 3-tupel data numerik, (Xq,Yq) adalah koordinat titik kiri atas gambar bata, dan Sudut adalah besar sudut rotasi bata.
- GambarBata : gambar dari bata
- Gambar Jaringan Dinamis : gambar dari rangkaian bata
- (Xr,Yr) : koordinat pointer *mouse* yang menunjuk gambar komponen sebuah jaringan dinamis
- (Xs,Ys) : koordinat titik terpilih pada gambar jaringan dinamis
- Nilai Counter : bilangan cacah non-negatif

Pada tugas akhir ini penulis hanya merealisasikan subsistem sunting, dan tidak merealisasikan subsistem jelajah. Keputusan ini diambil berdasarkan pada pertimbangan bahwa hanya subsistem sunting yang dibutuhkan oleh peneliti untuk mendukung penelitian tentang pengenalan struktur kognitif siswa. Subsistem pemakai secara otomatis akan direalisasikan juga bersama dengan subsistem sunting.

Kontribusi subsistem jelajah dibutuhkan jikalau perangkat lunak DM dipakai sebagai media pengajaran. Subsistem jelajah akan dikembangkan sebagai kelanjutan penelitian ini dan dilaksanakan di luar tugas akhir ini.

### 3.1.1 Kategori Pemakai Perangkat Lunak DM

Perangkat lunak ini dibangun untuk mendukung penelitian dalam bidang Pendidikan Matematika. Oleh karena itu, pemakai perangkat lunak ini dapat dipisahkan menjadi 2 kelompok, yaitu: peneliti dan siswa.

Meski dibedakan menjadi 2 kelompok pemakai, namun otoritas dari kedua kelompok tersebut tidak dibedakan pada subsistem sunting. Sedangkan pada subsistem jelajah, otoritas keduanya dibedakan. Untuk keperluan penelitian, siswa tidak perlu menggunakan subsistem jelajah. Dengan demikian, otoritas terhadap subsistem jelajah hanya dimiliki oleh peneliti saja.

### 3.1.2 Proses Pemakaian Perangkat Lunak DM

Pada bagian ini akan penulis sajikan proses pemakaian perangkat lunak DM, yang menyatu dengan proses penelitian.

Peneliti akan memberi sebuah tugas berkaitan dengan algoritma. Contoh tugas yang diberikan. "Bangunlah suatu algoritma untuk soal berikut. "

| $R_1$ | $R_2$ |
|---|---|
| $x_1$ | $x_2$ |

→

| $R_1$ | $R_2$ |
|---|---|
| 0 | $x_1+x_2$ |

Aturan yang harus dipakai oleh siswa adalah: pada setiap langkah siswa hanya boleh menambah 1 atau mengurangi 1 isi register; dan isi pada setiap register tidak boleh diketahui oleh siswa.

Jika siswa tertarik menggunakan alat bantu jaringan dinamis, maka perangkat lunak DM segera dioperasikan. Fungsi dari setiap

komponen fisik jaringan dinamis dapat ditemukan pada perangkat lunak DM.

Kegiatan menyusun jaringan dinamis yang dahulu dilakukan terhadap benda nyata, kini dilakukan terhadap gambar di layar VDU. Langkah yang harus dilakukan oleh siswa adalah:

- pilih gambar bata yang sesuai,
- lakukan manipulasi bata seperlunya (rotasi ke kiri / kanan),
- pasang gambar bata yang telah dimanipulasi pada tempat yang sesuai pada papan panel,
- jika diperlukan untuk menghapus gambar bata, hapuslah gambar bata yang dimaksud dari papan panel.

Diagram pada gambar 3.1, penjelasan kategori pemakai, dan proses pemakaian perangkat lunak DM, membantu kita untuk memahami ranah masalah yang akan dimodelkan secara keseluruhan.

Penulis akan melakukan sedikit perubahan pada penggambaran *message connetion*. Pada penyajian hasil OOA dan OOD penulis menggambar anak-panah yang menggambarkan *message connection* sebagai sepasang anak-panah. Anak-panah pertama adalah anak-panah yang ke luar dari sebuah kelas & objek pengirim, dan anak-panah kedua adalah anak-panah yang memasuki sebuah kelas & objek penerima. Pertimbangan yang mendasari langkah tersebut adalah keterbatasan bidang gambar serta menghindari agar gambar tidak menjadi kusut.

Selanjutnya, hasil OOA dan OOD disajikan sebagai berikut.

## 3.2 Perancangan OOA

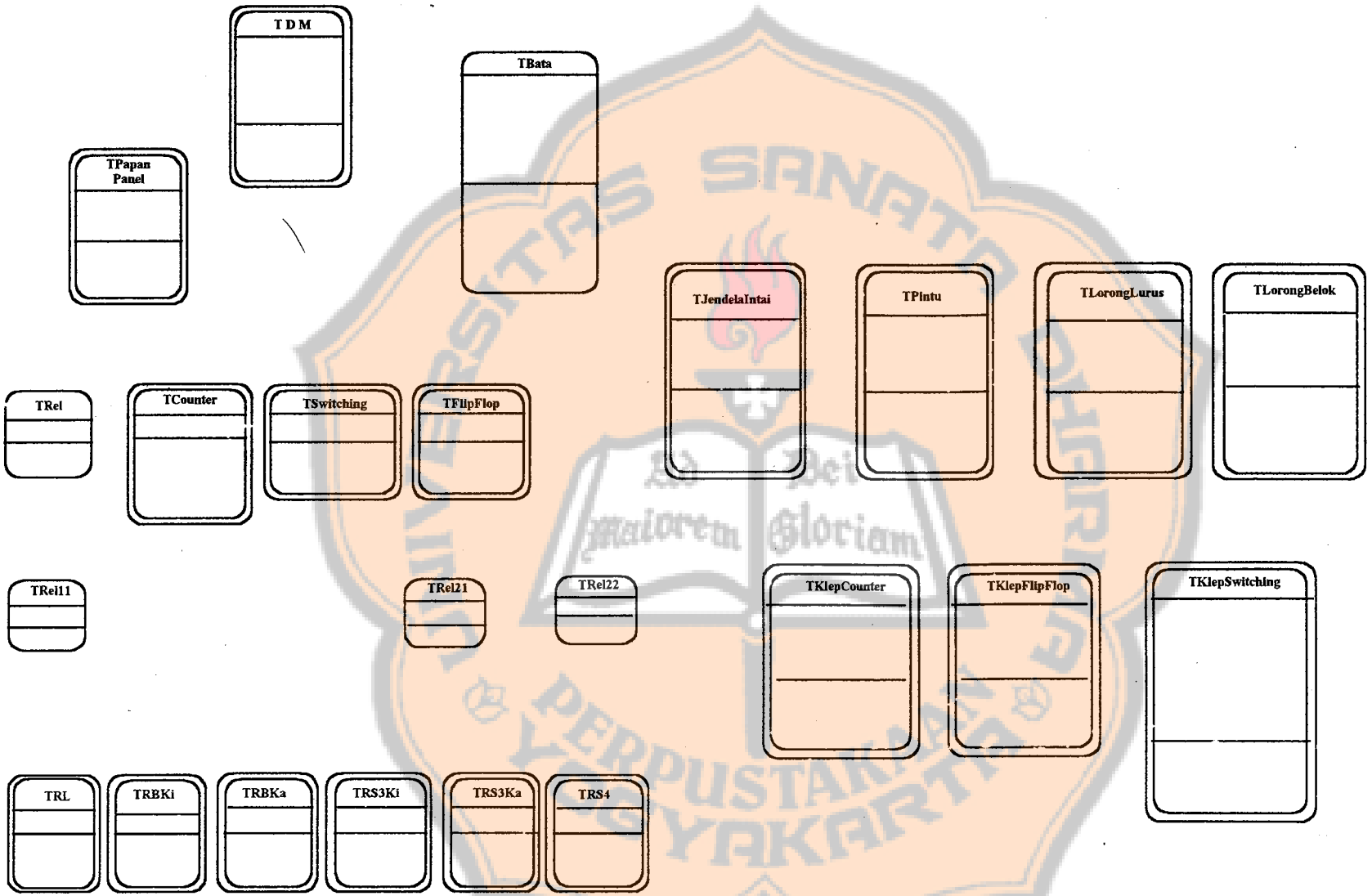
Pada bagian ini akan disajikan hasil akhir dari setiap tahapan analisis berorientasi objek.

Tahapan pendefinisian subjek tidak dilakukan. Hal ini disebabkan karena kelas dan kelas&objek yang diperoleh pada kegiatan analisis terhadap rekayasa perangkat lunak DM tidak cukup banyak, sehingga kehadiran subjek tidak terlalu diperlukan.

### 3.2.1 Tahap Pendefinisian Kelas dan Kelas & Objek

Perangkat lunak DM dibangun untuk keperluan substitusi fungsi dari alat jaringan dinamis. Oleh karena itu, benda-benda fisik dari alat jaringan dinamis secara otomatis akan menjadi kelas atau kelas & objek.

Keseluruhan kelas dan kelas & objek dapat dilihat pada gambar 3.2 pada halaman III-7.




Gambar 3.2 Lapisan Kelas & Objek

Keterangan kelas dan kelas & objek dari gambar 3.2 sebagai berikut:

- TDM adalah kelas & objek jaringan dinamis.
- TPapanPanel adalah kelas & objek papan panel.
- TBata adalah kelas bata.
- TRel adalah kelas rel.
- TCounter adalah kelas & objek *counter*.
- TSwitching adalah kelas & objek *switching*.
- TFlipFlop adalah kelas & objek *flipflop*.
- TRel11 adalah kelas rel yang memiliki 1 buah pintu masuk dan 1 buah pintu ke luar.
- TRel21 adalah kelas rel yang memiliki 2 buah pintu masuk dan 1 buah pintu ke luar.
- TRel22 adalah kelas rel yang memiliki 2 buah pintu masuk dan 2 buah pintu ke luar.
- TRL adalah kelas & objek rel lurus.
- TRBKi adalah kelas & objek rel belok kiri.
- TRBKa adalah kelas & objek rel belok kanan.
- TRS3Ki adalah kelas & objek rel simpang tiga kiri.
- TRS3Ka adalah kelas & objek rel simpang tiga kanan.
- TRS4 adalah kelas & objek rel simpang empat.
- TJendelaIntai adalah kelas & objek jendela intai pada *counter*.
- TPintu adalah kelas & objek pintu.

- TLorongLurus adalah kelas & objek lorong lurus.
- TLorongBelok adalah kelas & objek lorong belok.
- TKlepCounter adalah kelas & objek klep *counter*.
- TKlepFlipFlop adalah kelas & objek klep *flipflop*.
- TKlepSwitching adalah kelas & objek klep *switching*.

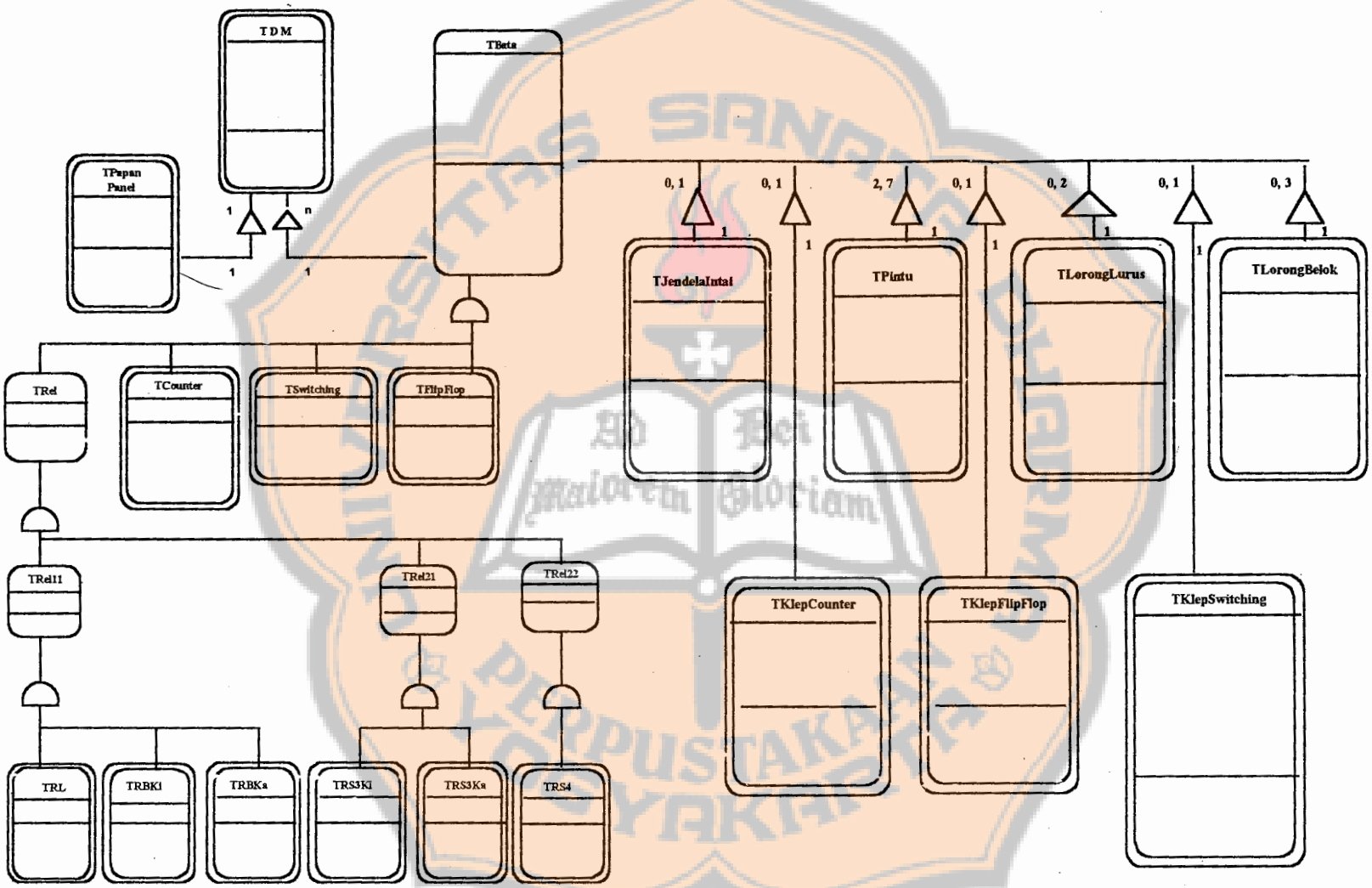
### 3.2.2 Tahap Pendefinisian Struktur

Pembangunan struktur di antara 23 buah kelas dan kelas & objek dilakukan berdasarkan pertimbangan berikut:

1. Sebuah jaringan dinamis terdiri dari: 1 buah papan panel, dan sejumlah berhingga bata.
2. Bata dapat diklasifikasikan menurut fungsinya menjadi 9 macam bata, yaitu:
   - Rel-Lurus (RL),
   - Rel-Belok-Kiri (RBKi),
   - Rel-Belok-Kanan (RBKa),
   - Rel-Simpang-3-Kiri (RS3Ki),
   - Rel-Simpang-3-Kanan (RS3Ka),
   - Rel-Simpang-4 (RS4),
   - *Counter*,
   - *FlipFlop*, dan
   - *Switching*.
3. Komponen yang dimiliki oleh setiap jenis bata adalah:

- RL memiliki: 2 buah pintu;dan 1 buah lorong lurus.
- RBKi memiliki: 2 buah pintu;dan 1 buah lorong belok.
- RBKa memiliki: 2 buah pintu;dan 1 buah lorong belok.
- RS3Ki memiliki: 3 buah pintu; 1 buah lorong lurus; dan 1 buah lorong belok.
- RS3Ka memiliki: 3 buah pintu; 1 buah lorong lurus; dan 1 buah lorong belok.
- RS4 memiliki: 4 buah pintu; 2 buah lorong lurus.
- *Counter* memiliki: 5 buah pintu; 2 buah lorong lurus; 3 buah lorong belok; 1 buah klep *counter*; dan 1 buah jendela intai.
- *FlipFlop* memiliki: 3 buah pintu; 2 buah lorong lurus; dan 1 buah klep *flipflop*.
- *Switching* memiliki: 7 buah pintu; 2 buah lorong lurus; 2 buah lorong belok; dan 1 buah klep *switching*.

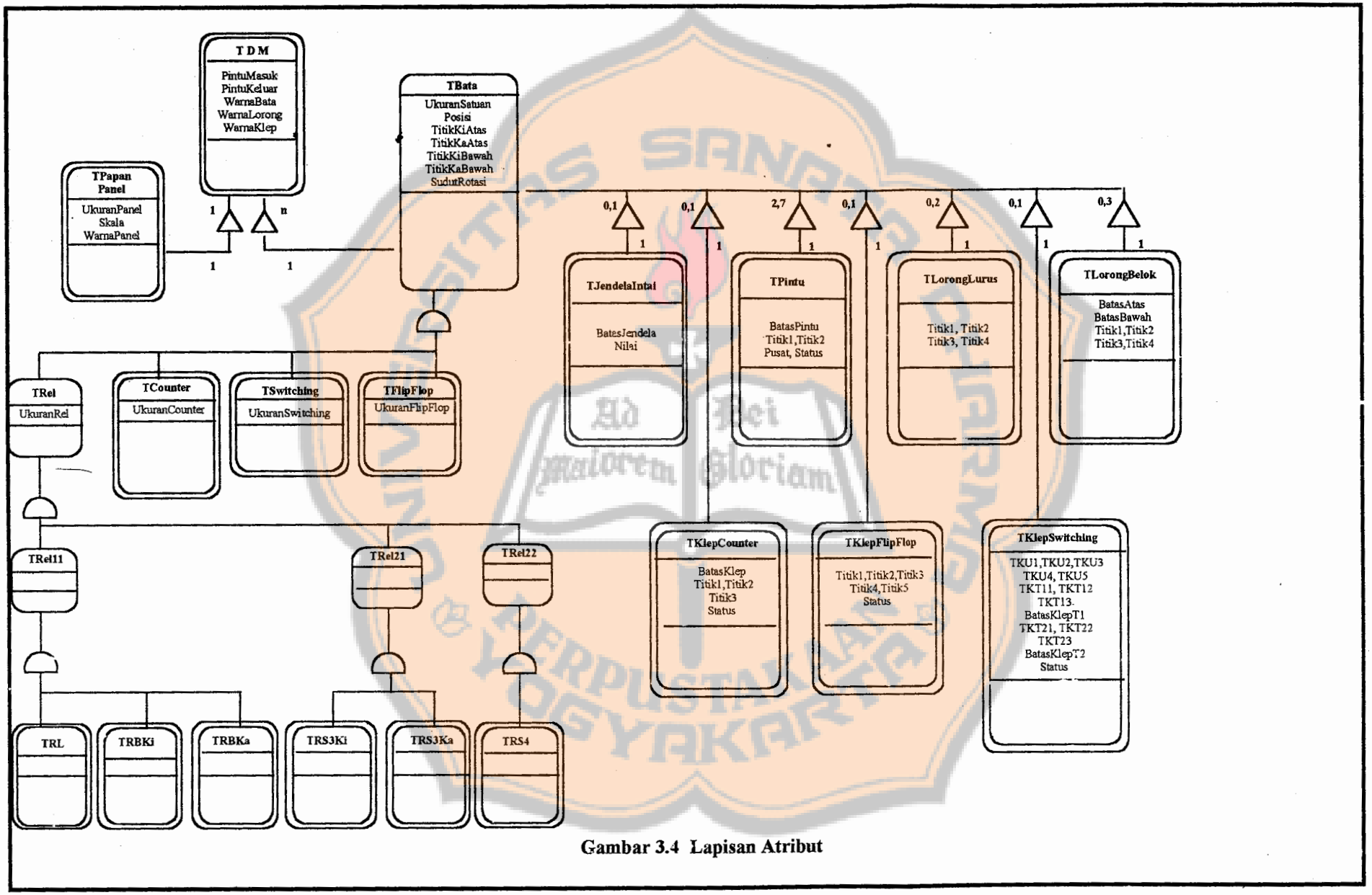
Struktur yang berhasil dibangun dapat dilihat pada gambar 3.3 halaman III-11.




**Gambar 3.3 Lapisan Struktur**

### 3.2.3 Tahap Pendefinisian Atribut

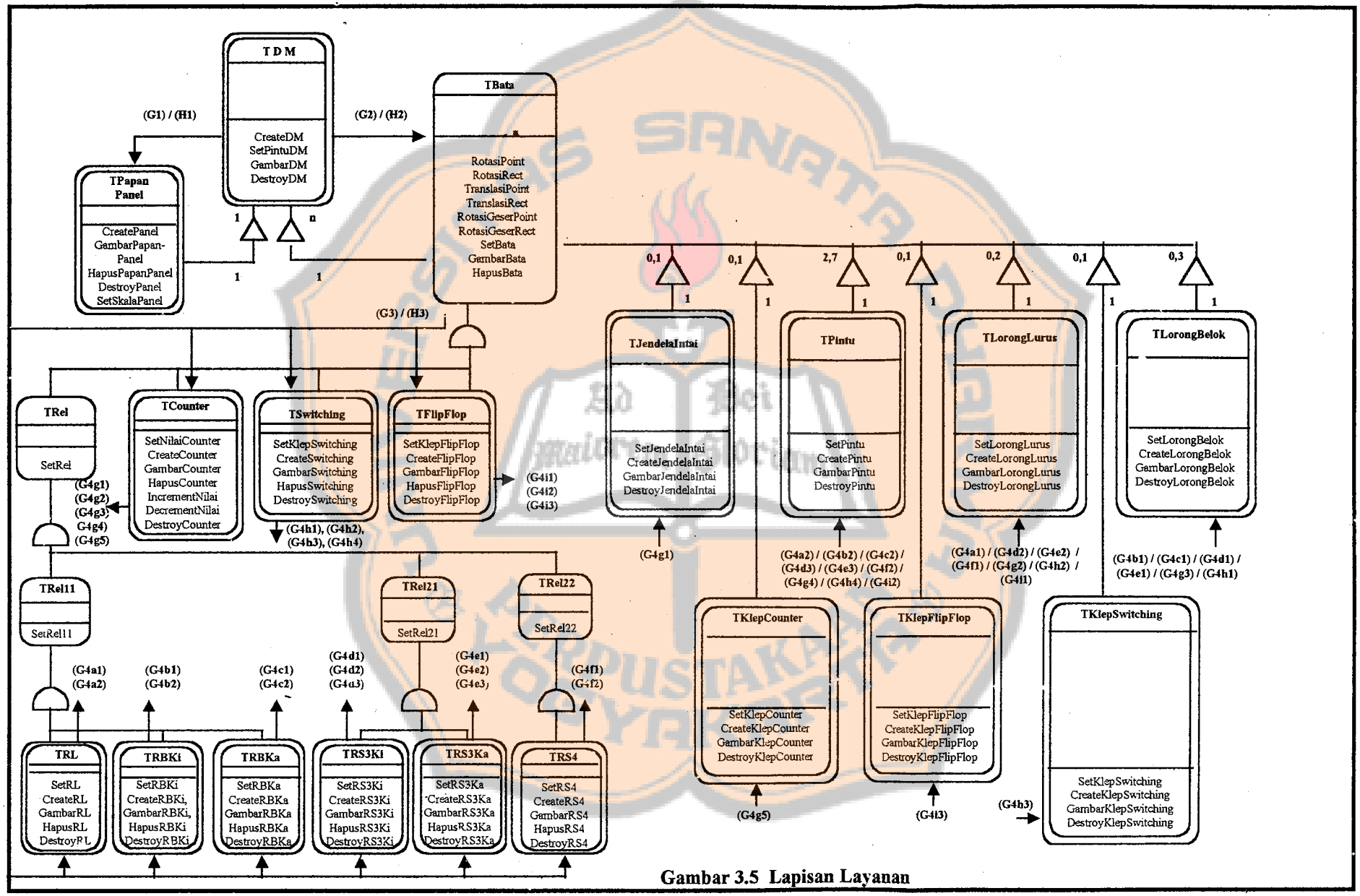
Atribut dari seluruh kelas dan kelas & objek dapat dilihat pada gambar 3.4 halaman III-13.




**Gambar 3.4 Lapisan Atribut**

### 3.2.4 Tahap Pendefinisian Layanan

Layanan dari seluruh kelas dan kelas & objek dapat dilihat pada gambar 3.5 halaman III-15.




**Gambar 3.5 Lapisan Layanan**

Keterangan yang lebih rinci tentang atribut dan layanan dari setiap kelas dan kelas & objek hasil analisis, dapat dilihat pada buku acuan teknis bagian A.

Layanan HapusPapanPanel tidak termuat pada lampiran tersebut. Layanan ini bertujuan untuk menghapus gambar papan panel dari VDU.

Pada lapisan layanan terdapat 2 buah serial *message-connection*, yaitu: serial (G#) dan serial (H#). Serial (G#) menunjuk pada serial *message* Gambar, sedangkan serial (H#) menunjuk pada serial *message* Hapus. Notasi untuk kedua serial *message connection* ditulis dengan memakai sebuah notasi yang sama, dan dituliskan sebagai: (G#) / (H#). Hal itu berarti bahwa notasi yang sama dipakai untuk serial (G#) maupun serial (H#). Cara penulisan tersebut semata bertujuan untuk menyederhanakan gambar, sehingga diagram tetap mudah dibaca.

Penulis juga menambahkan notasi *message connection* terhadap serangkaian (paralel) *message*. Tambahan notasi penulis lakukan dengan menduplikasi notasi yang telah ada tetapi menggunakan abjad dan angka yang menerangkan kedudukan *message*. Sebagai contoh, (G4a1) berarti *message* gambar pada urutan ke-4 dan *message* yang dikirim adalah *message* a urutan ke-1.

*Message* a menunjuk pada *message* gambar RL, *message* b menunjuk pada *message* gambar RBKi, *message* c menunjuk pada *message* gambar RBKa, *message* d menunjuk pada *message* gambar RS3Ki, *message* e menunjuk pada *message* gambar RS3Ka, *message* f menunjuk pada *message* gambar RS4, *message* g menunjuk pada

*message* gambar *counter*, *message* h menunjuk pada *message* gambar *switching*, *message* i menunjuk pada *message* gambar *flipflop*.

## 3.3 Perancangan OOD

Pada bagian ini akan disajikan hasil akhir dari setiap tahapan perancangan perangkat lunak DM.

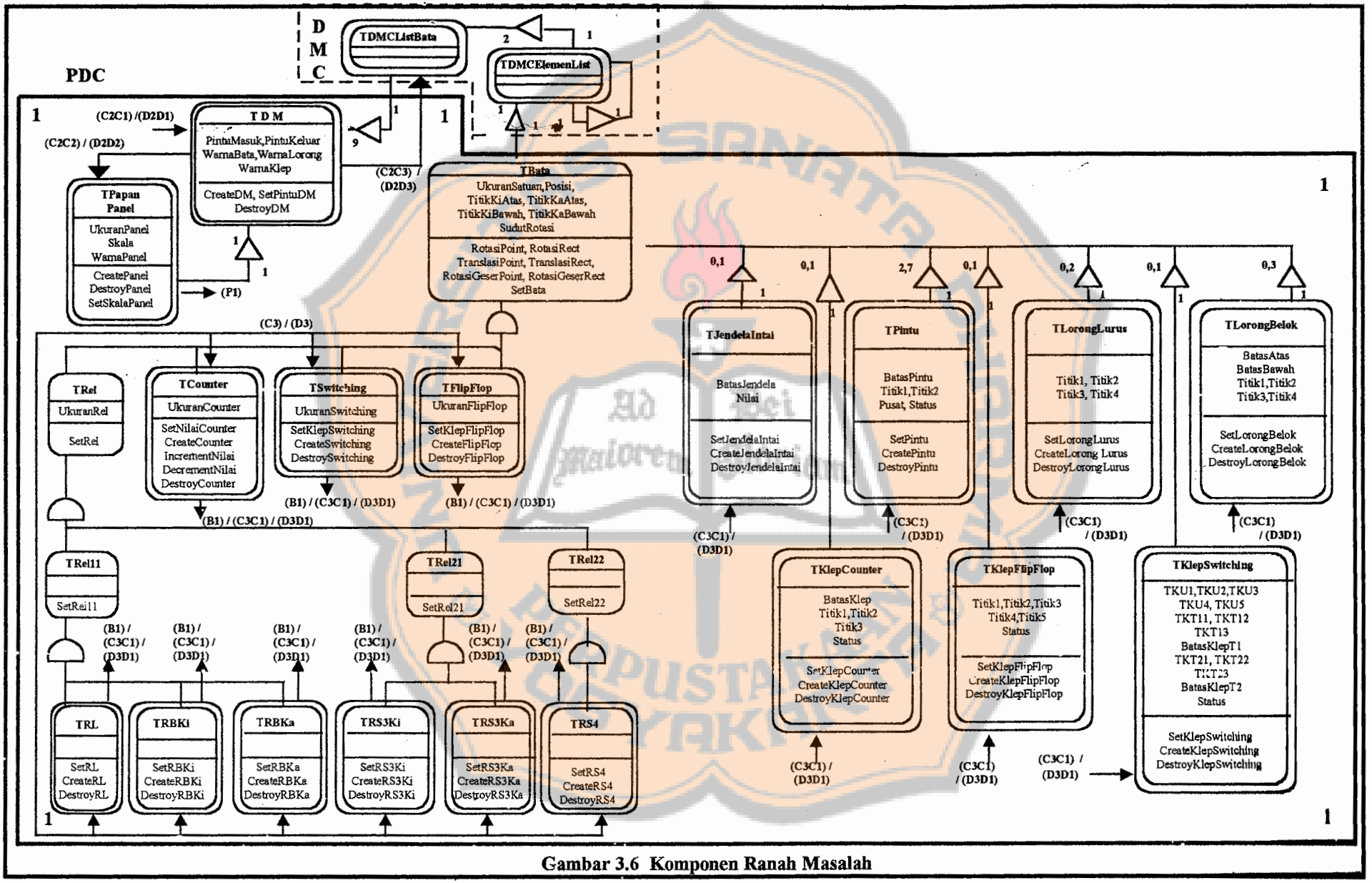
### 3.3.1 Tahap Perancangan Komponen Ranah Masalah (PDC)

Hasil perancangan pada tahap ini merupakan modifikasi dari hasil analisis. Modifikasi yang dilakukan terhadap hasil analisis tersebut adalah:

1. Layanan gambar dan hapus yang dimiliki oleh kelas dan kelas & objek hasil analisis dikeluarkan dari PDC. Layanan tersebut lebih tepat dilakukan oleh kelas dan kelas & objek pada HIC.

2. Jumlah setiap bata yang dapat dipilih oleh pemakai pada sebuah jaringan dinamis, tidak bisa diantisipasi sebelumnya baik jenis maupun banyak kali kemunculannya. Oleh karena itu, diperlukan kelas & objek list linier. Struktur keseluruhan-bagian antara TDM dan TBata dimodifikasi dengan menambah kelas & objek baru, yaitu: list-linier. Kelas & objek list-linier berfungsi sebagai pengelola seluruh bata pada jaringan dinamis, dan lebih tepat bila dikelompokkan dalam kelas & objek DMC.

Hasil akhir perancangan ranah masalah dapat dilihat pada gambar 3.6 halaman III-18.




**Gambar 3.6 Komponen Ranah Masalah**

### 3.3.2 Tahap Perancangan Komponen Antarmuka Pemakai (HIC)

Antarmuka pemakai pada perangkat lunak DM melibatkan 3 buah jendela. Jendela pertama merupakan jendela utama yang dipakai untuk melakukan penyuntingan jaringan dinamis. Jendela kedua dipakai untuk melakukan dialog pada proses baca dan tulis berkas eksternal jaringan dinamis. Sedangkan jendela ketiga dipakai untuk menayangkan bantuan bagi pemakai perangkat lunak. Ketiga jendela tersebut disajikan sebagai objek: THICForm1; THICForm2; THICForm3.

Meski melibatkan 3 buah jendela, antarmuka pemakai disusun dengan menggunakan model dokumen tunggal (*single document interface*, disingkat SDI). Pemilihan SDI dimaksudkan untuk mengakomodasikan prinsip kesederhanaan antarmuka pemakai.

Teknik interaksi yang diterapkan pada rancangan antarmuka pemakai adalah teknik interaksi *user originated*, dan teknik *computer originated* khusus pada bagian pesan kesalahan. Gabungan kedua teknik interaksi tersebut diharapkan agar antarmuka pemakai mampu mengakomodasikan prinsip mudah dipakai dan dipelajari.

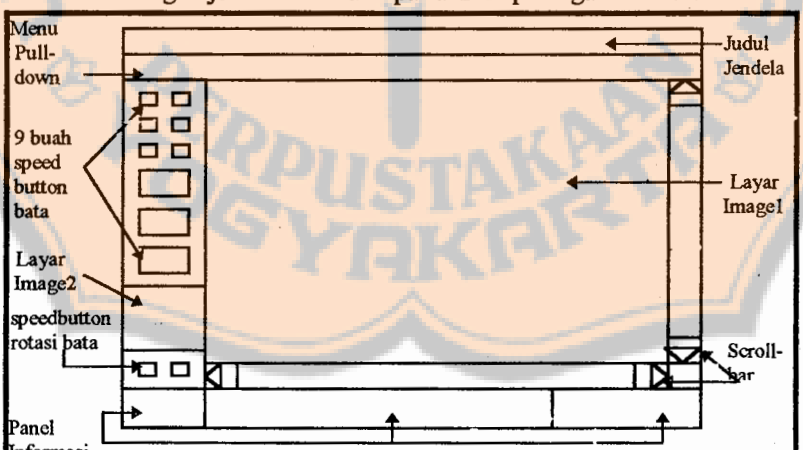
Pada saat perangkat lunak diaktifkan, jendela pertama akan diaktifkan secara otomatis. Jendela tersebut memuat menu *pull-down*, sejumlah *speedbutton* yang memberi fasilitas pilihan bata serta pilihan manipulasi bata, 2 buah layar *image* untuk melakukan penggambaran dan penghapusan gambar jaringan dinamis berikut fasilitas *scrolling*, dan panel informasi.

Menu dinyatakan dalam bahasa Indonesia, dan dilengkapi dengan fasilitas *shortcut* yang memberi kemudahan dan kecepatan memilih tugas bagi pemakai trampil. Pertimbangan ini dimaksudkan untuk mengakomodasikan prinsip kesesuaian terhadap pemakai dan prinsip fleksibilitas antarmuka pemakai.

Penggambaran dan penghapusan gambar jaringan dinamis pada layar *image* dapat dilakukan oleh pemakai secara langsung. Pertimbangan ini didasarkan pada prinsip manipulasi langsung pada antarmuka pemakai.

Panel informasi terdiri dari 3 buah sub-panel, yaitu: panel informasi modus sunting ; panel informasi tugas yang akan dikerjakan pada pilihan menu serta informasi bagian jendela; panel koordinat titik terpilih. Panel ini dirancang untuk mendukung prinsip mudah dipakai dan mudah dipelajari.

Rancangan jendela utama dapat dilihat pada gambar 3.7.




**Gambar 3.7 Rancangan Jendela Utama**

Menu utama berikut item-menu disusun sebagai berikut:

Berkas Rangkai Tampil Jelajah Petunjuk

Baru Ctrl+U
Baca Ctrl+A
Tulis Ctrl+S
KeluarCtrl+K

Hapus Del
Gambar Ins
Nilai
Counter Ctrl+C
Pintu Ctrl+P
Klep Ctrl+E

Skala
40x40Ctrl+F4
30x30Ctrl+F3
20x20Ctrl+F2
10x10Ctrl+F1

Untuk memulai tugas menyunting jaringan dinamis baru, pemakai akan berurusan dengan menu Berkas. Item Baru dipilih untuk memulai lembar kerja baru yang siap menyunting jaringan-dinamis. Item Baca dipilih untuk membaca berkas eksternal, membangkitkan data internal, dan melakukan penggambaran jaringan-dinamis yang sesuai. Item Tulis dipilih untuk menyimpan data kunci jaringan dinamis ke dalam berkas eksternal. Item Keluar dipilih untuk mengakhiri pemakaian perangkat lunak DM, dan kembali ke sistem operasi Windows '95.

Untuk mengubah status sunting, pemasangan nilai *counter*, lokasi pintu-masuk dan pintu-ke-luar, status klep jaringan dinamis, pemakai akan berurusan dengan menu Rangkai. Item menu Gambar dipakai untuk mengubah status sunting menjadi status gambar. Item menu Hapus dipakai untuk mengubah status sunting menjadi status hapus. Item Nilai dipakai untuk memasang nilai *counter* - yaitu melalui sub-item Counter -, pintu masuk dan ke luar jaringan dinamis - yaitu melalui sub-item Pintu - , posisi klep yaitu melalui sub-item Klep. Ketiga

sub-item yang termuat pada item Nilai tidak direalisasikan pada tugas akhir ini.

Untuk melihat keseluruhan rangkaian jaringan dinamis, pemakai dapat melakukan *zooming* melalui pemilihan menu Tampil. Sub-item 40x40 dipilih jika ingin dilakukan tampilan yang lebih besar (*zoom out*). Sub-item 30x30 dipilih untuk tampilan normal (*default*). Sub-item 20x20 dipilih untuk tampilan yang lebih kecil (*zoom in* -1). Sub-item 10x10 dipilih untuk tampilan yang paling kecil (*zoom in* - 2).

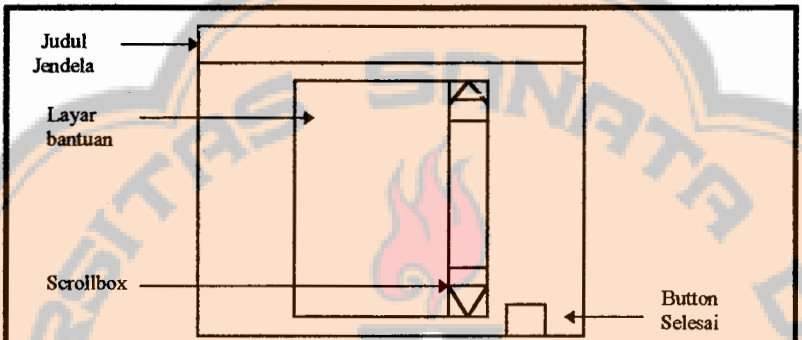
Untuk melakukan penjelajahan, penulis memilih menu Jelajah. Namun bagian ini tidak direalisasikan pada tugas akhir ini.

Untuk memperoleh bantuan cara menyusun jaringan dinamis pada DM, pemakai memilih menu Petunjuk.

Jendela kedua akan diaktifkan melalui pilihan tugas pada item menu Baca atau Tulis. Jendela ketiga akan diaktifkan melalui pilihan tugas pada item Petunjuk. Kedua jendela tersebut merupakan *modal-form*. Pemakaian *modal-form* untuk menjaga agar respon dari pemakai dapat benar-benar dikendalikan oleh perangkat lunak. Hal ini untuk menjaga prinsip kesesuaian aliran pekerjaan.

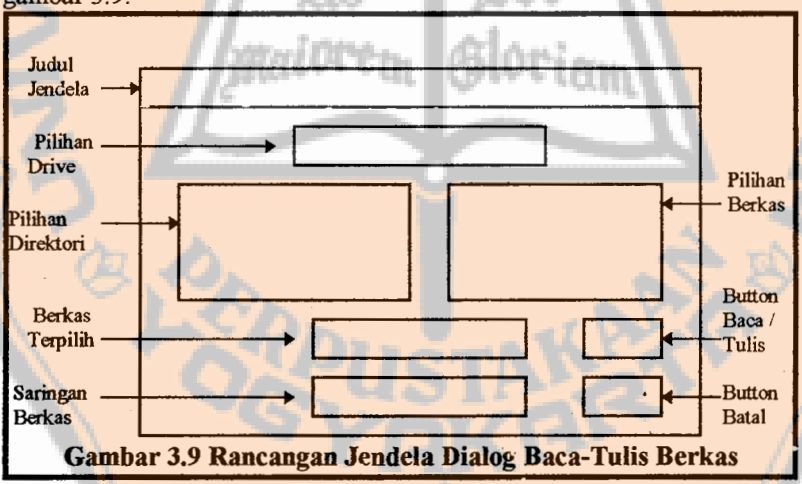
Rancangan jendela untuk dialog baca dan tulis berkas disesuaikan dengan format dialog yang sama pada perangkat lunak di bawah sistem operasi Windows. Hal ini dimaksudkan untuk mengakomodasikan prinsip keakraban antarmuka pemakai.

Rancangan jendela bantuan dapat dilihat pada gambar 3.8 di bawah ini.



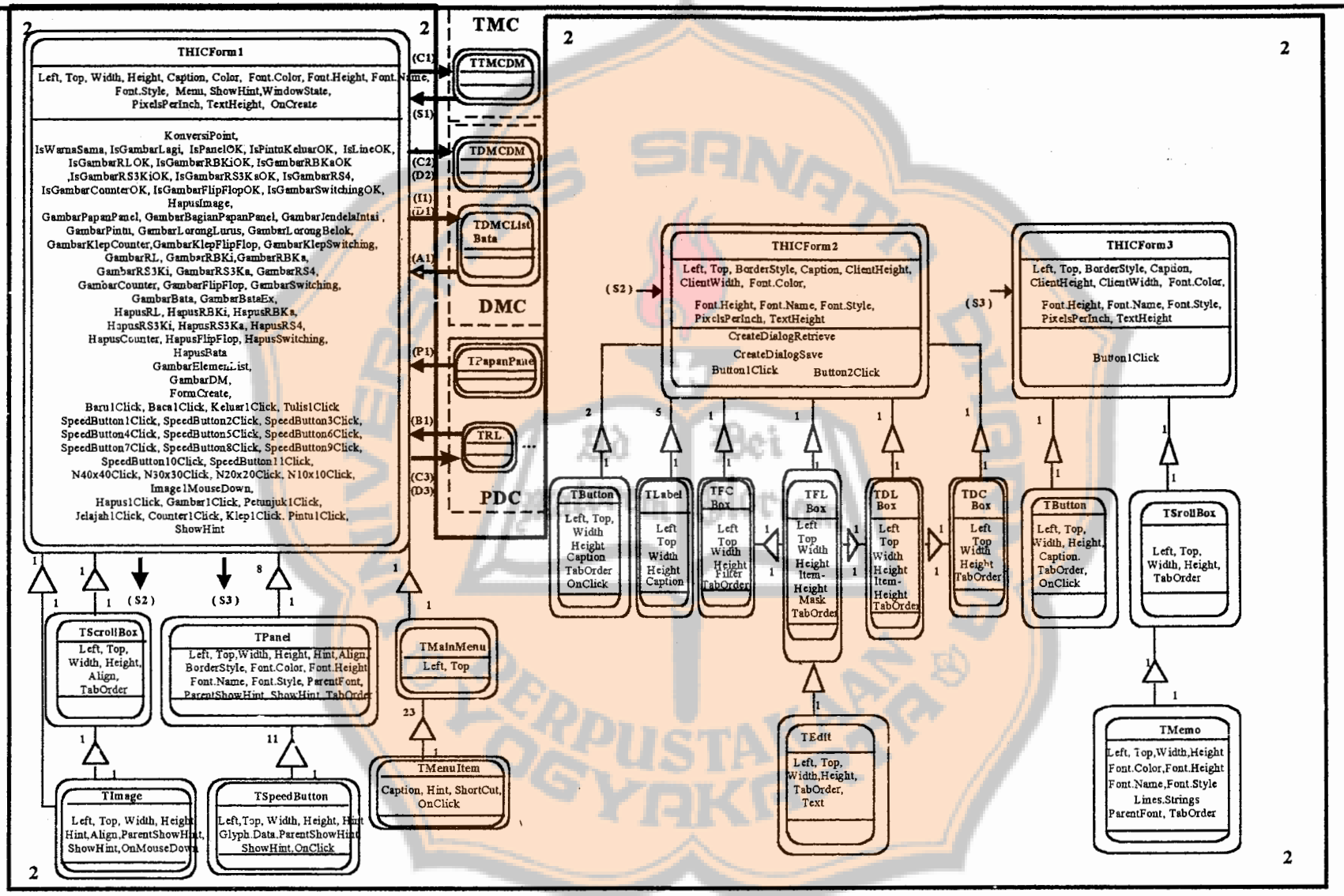

**Gambar 3.8 Rancangan Jendela Bantuan**

Rancangan jendela dialog baca dan tulis berkas dapat dilihat pada gambar 3.9.




**Gambar 3.9 Rancangan Jendela Dialog Baca-Tulis Berkas**

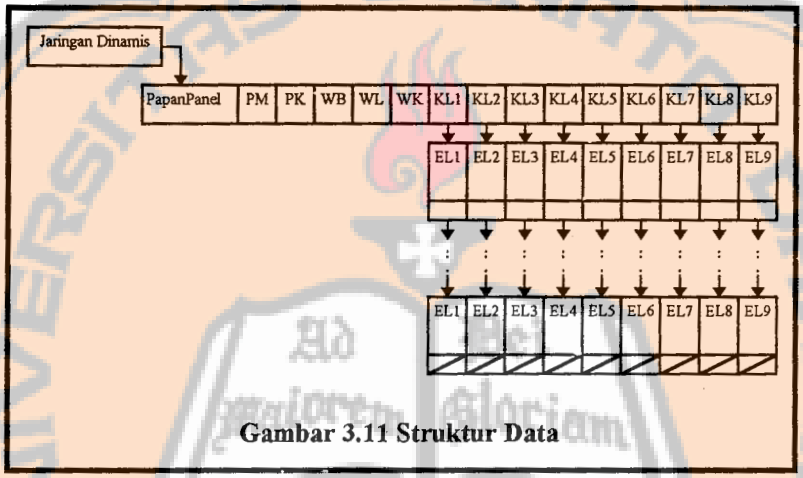
Rancangan ketiga jendela di atas diterjemahkan ke dalam kelas &objek HIC sebagai berikut. Lihat pada gambar 3.10 halaman III-24.




**Gambar 3.10 Komponen Antarmuka Pemakai**

### 3.3.3 Tahap Perancangan Komponen Manajemen Data (DMC)

Perangkat lunak DM melibatkan struktur data dan struktur berkas. Struktur data dapat dilihat pada gambar 3.11, dan struktur berkas dapat dilihat pada gambar 3.12.




Gambar 3.11 Struktur Data

**Keterangan:**

PM=Pintu Masuk
PK=Pintu Keluar WB=Warna Bata
WL=Warna Lorong
WK=Warna Klep
KL1=Kepala ListRL
EL1=Elemen ListRL= RI
KL2=Kepala ListRBKi
EL2=Elemen ListRBKi= RBKi
KL3=Kepala ListRBKa
EL3=Elemen ListRBKa= RBKa
KL4=Kepala ListRS3Ki
EL4=Elemen ListRS3Ki = RS3Ki
KL5=Kepala ListRS3Ka
EL5=Elemen ListRS3Ka = RS3Ka
KL6=Kepala ListRS4
EL6=Elemen ListRS4 = RS4
KL7=Kepala ListCounter
EL7=Elemen ListCounter= *Counter*
KL8=Kepala ListFlipFlop
EL8=Elemen ListFlipFlop=*FlipFlop*
KL9=Kepala ListSwitching
EL9=Elemen ListSwitching= *Switching*

```
[DM-SUGI-ITB-23594014]
[MYPANEL]
1000;1000;30
[PINTUMASUK]
0;0
[PINTUKELUAR]
0;0
[TBATA]
[RL]
[ENDRL]
[RBKI]
[ENDRBKI]
[RBKA]
[ENDRBKA]
[RS3KI]
[ENDRS3KI]
[RS3KA]
[ENDRS3KA]
[RS4]
[ENDRS4]
[COUNTER]
[ENDCOUNTER]
[FLIPFLOP]
[ENDFLIPFLOP]
[SWITCHING]
[ENDSWITCHING]
```

**Gambar 3.12 Struktur Berkas**

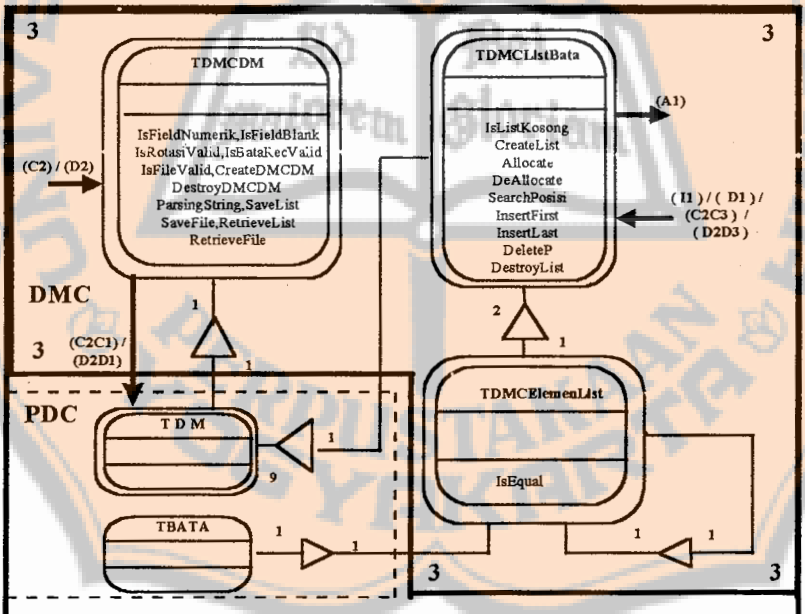
**Keterangan**

- [DM-SUGI-ITB-23594014] = header berkas
- [MYPANEL] = header yang mengawali data papanpanel
- 1000;1000; 30 = record data papanpanel (berarti lebar = 1000, tinggi = 1000, skala = 30), pembatas field adalah ';'
- [PINTUMASUK] = header yang mengawali data pintu-masuk
- 0;0 = record data pintu-masuk (berarti koordinat pintu-masuk = (0,0) )
- [PINTUKELUAR] = header yang mengawali data pintu-ke-luar
- 0;0 = record data pintu-ke-luar (berarti koordinat pintu-ke-luar = (0,0) )
- [TBATA] = header yang mengawali untaian bata
- [RL] = header yang mengawali untaian rel-lurus
- [ENDRL] = footer yang mengakhiri untaian rel-lurus, data rel-lurus disajikan dalam bentuk record di antara header dan footer, misal: 90;120;180 berarti absis = 90, ordinat = 120, dan sudut-rotasi = 180 derajat.
- [RBKI]-[ENDRBKI], [RBKA]-[ENDRBKA],[RS3KI]-[ENDRS3KI],[RS3KA]-[ENDRS3KA],[RS4]-[ENDRS4],[COUNTER]-[ENDCOUNTER],[FLIPFLOP]-[ENDFLIPFLOP], [SWITCHING]-[ENDSWITCHING] = idem [RL] - [ENDRL]

Pada tahap ini dibangun sebuah kelas & objek TDMCDM yang berfungsi untuk mengelola data dan berkas. Pengelolaan berkas dibutuhkan DM untuk mendukung proses baca dan tulis berkas. Sedangkan pengelolaan data dibutuhkan untuk mendukung proses gambar dan hapus jaringan dinamis.

Selain objek TDMCDM, dibangun pula sepasang kelas & objek yaitu: TDMCListBata dan TDMCElemenList. Kedua kelas & objek berfungsi sebagai sebuah list-linier berkait dengan pointer.

Hasil perancangan komponen manajemen data dapat dilihat pada gambar 3.13.




**Gambar 3.13 Komponen Manajemen Data**

### 3.3.4 Tahap Perancangan Komponen Manajemen Tugas (TMC)

Tugas utama yang dilakukan pada perangkat lunak ini adalah menggambar dan menghapus gambar bata. Kedua tugas itu dilakukan dengan cara yang sama, yaitu: melakukan klik *mouse* pada sebuah area *image1* yang terletak pada *form1*. Tugas dibedakan dengan memberi status sunting, yaitu: gambar dan hapus.

Tabel 3.1 menyajikan tugas yang tidak boleh dilakukan oleh pemakai pada masing-masing status sunting, dan perubahan yang dilakukan pada komponen jendela utama.

**Tabel 3.1 Status Sunting dan Larangan Tugas**

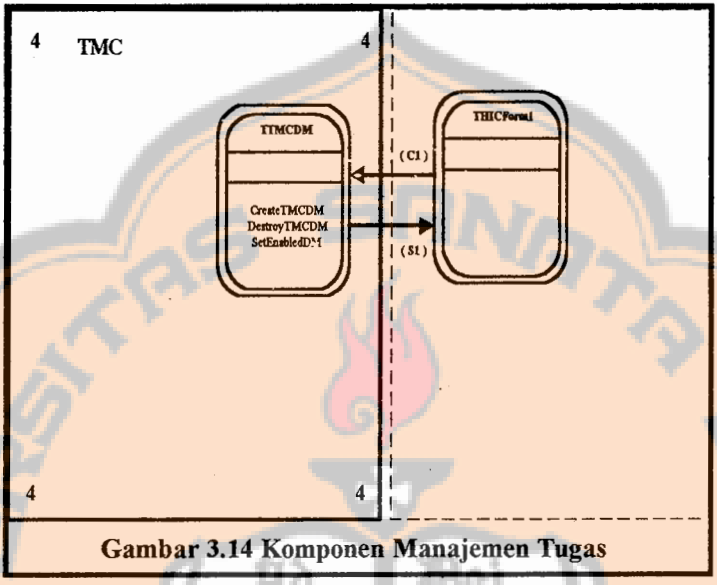
| No | Status Sunting | Tugas yang Dilarang | Perubahan Komponen Jendela Utama |
|---|---|---|---|
| 1 | Gambar | • klik item-menu Gambar | • Hapus1.Enabled = true<br>• Gambar1.Enabled = false<br>• Panel5.Enabled = true<br>• Panel7.Enabled = true<br>• Panel8.Caption = 'Modus = Gambar'<br>• Screen.Cursor = crDefault |

**Tabel 3.1 Status Sunting dan Larangan Tugas (sambungan)**

| No | Status Sunting | Tugas yang Dilarang | Perubahan Komponen Jendela Utama |
|---|---|---|---|
| 2 | Hapus | • klik item-menu Hapus<br>• Klik speedbutton bata<br>• Klik speedbutton rotasi bata | • Hapus1.Enabled = false<br>• Gambar1.Enabled= true<br>• Panel5.Enabled = false<br>• Panel7.Enabled = false<br>• Panel8.Enabled = 'Modus = Hapus'<br>• Screen.Cursor = crCross |

Pada tahap ini akan dibangun sebuah kelas & objek TTMCDM. Komponen itu akan mengatur tugas-tugas yang sah dilakukan pada masing-masing status sunting. Pengaturan tugas dilakukan dengan membuat sebagian komponen antarmuka pemakai dihidupkan atau dimatikan fungsinya. Di samping itu dilakukan pula perubahan tampilan pada panel informasi status sunting serta bentuk kursor, agar status tersebut dapat dikenali oleh pemakai.

Hasil perancangan komponen manajemen tugas dapat dilihat pada gambar 3.14 halaman III-30.




**Gambar 3.14 Komponen Manajemen Tugas**

Keterangan tentang atribut dan layanan yang didefinisikan pada setiap kelas dan kelas&objek OOD dapat dilihat pada buku acuan teknis bagian A.

Tabel 3.2 di bawah ini akan menyajikan arti dari setiap *message connection* (MC) yang dipakai pada keempat komponen hasil perancangan (OOD).

**Tabel 3.2 Message Connection pada OOD**

| No | MC | Rangkaian MC Terkait | Keterangan |
|---|---|---|---|
| 1 | (C1) | (S1) | *Create instance* TTMCDM |
| 2 | (S1) | - | *SetEnabled* komponen *form1* |

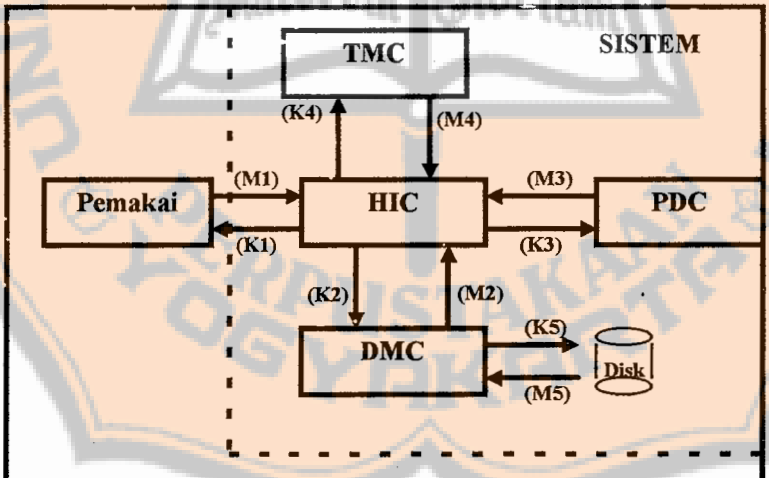
**Tabel 3.2 Message Connection pada OOD (sambungan)**

| No | MC | Rangkaian MC Terkait | Keterangan |
|---|---|---|---|
| 3 | (C2) | | *Create instance* TDMCDM |
| | | (C2C1) | *Create instance* TDM |
| | | (C2C2) | *Create instance* TPapanPanel |
| | | (C2C3) | *Create instance* TDMCListBata |
| 4 | (D1) | | *Delete (by value)* elemen list (*instance* TDMCListBata ) |
| 5 | (D2) | | *Destroy instance* TDMCDM |
| | | (D2D1) | *Destroy instance* kelas&objek bagian TBata |
| | | (D2D2) | *Destroy instance* kelas&objek spesialisasi TBata |
| | | (D2D3) | *Destroy instance* kelas&objek TDMCListBata |
| 6 | (C3) | | *Create instance* kelas & objek spesial dari TBATA |
| | | (C3C1) | *Create instance* kelas & objek bagian dari TBATA |
| 7 | (D3) | | *Destroy instance* kelas & objek spesial dari TBATA |
| | | (D3D1) | *Destroy instance* kelas & objek bagian dari TBATA |
| 8 | (I1) | - | *Insertlast* elemen list |

**Tabel 3.2 Message Connection pada OOD (sambungan)**

| No | MC | Rangkaian MC Terkait | Keterangan |
|---|---|---|---|
| 9 | (A1) | - | *passing* alamat sebuah elemen list |
| 10 | (P1) | - | *passing* data *instance* kelas & objek TPapanpanel |
| 11 | (B1) | - | *passing* data *instance* kelas & objek spesialisasi TBata |
| 12 | (S2) | - | *showmodal form2.* |
| 13 | (S3) | - | *showmodal form3.* |

Interaksi secara global dari keempat komponen hasil OOD dapat dilihat pada gambar 3.15.




**Gambar 3.15 Interaksi Komponen OOD dan Pemakai**

Keterangan:

- (M1) = Klik *mouse*, tekan tombol papan ketik
- (M2) = Data internal jaringan dinamis
- (M3) = Data bata
- (M4) = Data perubahan properti komponen jendela
- (M5) = Baca berkas dari *disk*
- (K1) = Tampilan jendela
- (K2) = *Create / destroy instance* objek DMC, SaveFile, RetrieveFile, DeleteP, InsertLast
- (K3) = *Create / destroy instance* objek PDC
- (K4) = *Create / destroy instance* objek TMC, SetEnabled
- (K5) = Tulis data ke dalam *disk*

# BAB IV
# IMPLEMENTASI PERANGKAT LUNAK DM

Pada bab ini penulis akan menyajikan implementasi dari hasil perancangan perangkat lunak DM. Sajian implementasi terdiri dari: deskripsi fungsional global kelas & objek hasil perancangan; pembagian modul; implementasi modul.

Hasil implementasi berupa kode program (berkas .PAS) maupun berupa definisi properti kelas & objek HIC (berkas .DFM) dapat dilihat pada buku acuan teknis bagian A.

## 4.1 Deskripsi Fungsional Global Kelas & Objek Hasil Perancangan

Pada bagian ini akan disajikan deskripsi fungsional dari kelas & objek hasil perancangan secara global. Deskripsi fungsional tersebut akan disajikan pada tabel 4.1 halaman IV-2.

Isi dari kolom ke-4 pada tabel tersebut, yaitu nama layanan Create*, Destroy*, dan Set*, berarti bahwa layanan tersebut direalisasikan pada kelas & objek dengan nama yang lebih spesifik. Misal Create* untuk kelas&objek TPintu, akan direalisasikan dalam layanan CreatePintu.

Isi dari kolom terakhir, yaitu kolom Keterangan, kata *instance** mempunyai arti instance dari kelas & objek yang bersangkutan. Jika isi kolom ke-3, yaitu Kelas & Objek, adalah TPintu, maka *instance** berarti sebagai *instance* dari kelas & objek TPintu.

**Tabel 4.1 Deskripsi Fungsional Global Kelas & Objek Hasil Perancangan**

| No | Fungsi | Kelas&Objek | Layanan | Keterangan |
|---|---|---|---|---|
| 1 | Pengembangan Data Komponen Bata | •TPintu<br>•TJendelaIntai<br>•TLorongLurus<br>•TLorongBelok<br>•TKlepCounter<br>•TKlepFlipFlop<br>•TKlepSwitching | •Create*<br>•Destroy*<br>•Set* | •menciptakan sebuah *instance**<br>• mengembalikan alamat sebuah *instance**<br>• mengubah nilai atribut sebuah *instance** |
| 2 | Pengembangan Data Bata | •TRL<br>•TRBKi<br>•TRBKa<br>•TRS3Ki<br>•TRS3Ka<br>•TRS4<br>•TCounter<br>•TFlipFlop<br>•TSwitching | •Create*<br>•Destroy*<br>•Set* | •menciptakan sebuah *instance**<br>• mengembalikan alamat sebuah *instance**<br>•mengubah nilai atribut sebuah *instance** |
| 3 | Pengelolaan List Linier | • TDMCListBata<br>• TDMCElemen List | •Create*<br>•Destroy*<br>•Allocate<br>•DeAllo-cate<br>•InsertLast<br>•DeleteP | •menciptakan sebuah *instance* TDMCList-Bata<br>•mengembalikan alamat sebuah *instance* TDMCList-Bata<br>•memesan alamat sebuah elemen list<br>•mengembalikan alamat sebuah elemen list<br>•Menyisipkan elemen list pada urutan akhir<br>•Menghapus elemen list berdasar nilai |

**Tabel 4.1 Deskripsi Fungsional Global Kelas & Objek Hasil Perancangan (sambungan)**

| No | Fungsi | Kelas&Objek | Layanan | Keterangan |
|---|---|---|---|---|
| 4 | Penggambaran Bata | • THICForm1 | •GambarRL<br>•GambarRBKi<br>•GambarRBKa<br>•GambarRS3Ki<br>•GambarRS3Ka<br>•GambarRS4<br>•GambarCounter<br>•GambarFlipFlop<br>•GambarSwitching | • menggambar bata pada layar VDU sesuai aturan pemasangan bata secara fisik |
| 5 | Penghapusan Gambar Bata | • THICForm1 | •GambarRL<br>•GambarRBKi<br>•GambarRBKa<br>•GambarRS3Ki<br>•GambarRS3Ka<br>•GambarRS4<br>•GambarCounter<br>•GambarFlipFlop<br>•GambarSwitching | • menghapus gambar bata pada layar VDU tanpa menghapus gambar bata lain yang telah ada sebelumnya |
| 6 | Pembacaan Berkas | • TDMCDM<br>• THICForm2 | • ParsingString<br>• RetriveFile | • melakukan parsing teks terhadap berkas data<br>• membaca berkas data dan membangun data seluruh bata |
| 7 | Penulisan Berkas | • TDMCDM<br>• THICForm2 | • SaveFile | •melakukan penulisan data papan panel,koordinat bata,sudut rotasi bata pada berkas data |

**Tabel 4.1 Deskripsi Fungsional Global Kelas & Objek Hasil Perancangan (sambungan)**

| No | Fungsi | Kelas&Objek | Layanan | Keterangan |
|---|---|---|---|---|
| 8 | Pemberian Bantuan | • THICForm3 | •showmodal | • mengaktifkan THICform3 sebagai *modal form* |
| 9 | *Zooming* | • THICForm1 | •HapusImage<br>•GambarPapanPanel<br>•GambarDM | • menghapus gambar papan panel<br>• menggambar papan panel sesuai skala<br>• menggambar semua bata penyusun jaringan dinamis sesuai skala |
| 10 | *Scrolling* | • THICForm1 | - | • menggunakan fasilitas *scroller* yang disediakan dalam HICForm1 |

## 4.2 Pembagian Modul

Sejumlah 46 buah kelas dan kelas & objek diperoleh sebagai hasil OOD. Organisasi dari ke-46 buah kelas dan kelas & objek dapat dilihat pada masing-masing komponen OOD.

Pembagian modul dilakukan dengan memperhatikan struktur yang terbentuk di antara ke-46 buah kelas dan kelas & objek, dan keterkaitan antar modul. Modul yang saling tergantung (modul siklik) harus dihindari.

Penulis memperoleh sejumlah 6 buah modul, yaitu: KompBata; Bata; ListBata; DM; Dialog; dan Bantuan.

### 4.2.1 Modul KompBata

Modul KompBata terdiri dari 7 buah kelas & objek komponen bata, yaitu: kelas & objek TJendelaIntai; kelas & objek TPintu; kelas & objek TLorongLurus; kelas & objek TLorongBelok; kelas & objek TKlepCounter; kelas & objek TKlepFlipFlop; dan kelas & objek TSwitching.

Modul KompBata berfungsi menyediakan *instance* dari seluruh komponen bata. *Instance* tersebut akan dipakai oleh modul Bata dan modul DM sebagai data masukan. Layanan yang diimplementasikan pada modul ini adalah: Create; Set; dan Destroy. Nama layanan masih disesuaikan dengan nama kelas & objeknya, misal: CreatePintu, SetPintu, dan DestroyPintu untuk kelas & objek TPintu.

### 4.2.2 Modul Bata

Modul Bata terdiri dari 1 buah kelas TBata dan 13 buah kelas & objek turunannya. Ke-13 kelas & objek turunan kelas TBata adalah: kelas TRel; kelas & objek TCounter; kelas & objek TFlipFlop; kelas & objek TSwitching; kelas TRel11; kelas TRel21; kelas TRel22; kelas & objek TRL; kelas & objek TRBKi; kelas & objek TRBKa; kelas & objek TRS3Ki; kelas & objek TRS3Ka; dan kelas & objek TRS4.

Modul Bata berfungsi menyediakan *instance* dari seluruh jenis bata. *Instance* tersebut akan dipakai oleh modul ListBata sebagai elemen

list, dan modul DM sebagai data masukan. Layanan yang diimplementasikan pada modul ini adalah: Create, Destroy, Set, IncrementNilai, DecrementNilai. Nama layanan Create, Destroy, dan Set disesuaikan dengan nama kelas & objek, misal: CreateRL untuk kelas & objek TRL, DestroyRL untuk kelas & objek TRL, dan seterusnya.

### 4.2.3 Modul ListBata

Modul ListBata terdiri dari 2 buah kelas & objek, yaitu: kelas & objek TDMCElemenList dan kelas & objek TDMCListBata. Kedua kelas & objek ini merupakan realisasi kebutuhan akan list-linier berkait dengan pointer.

Modul ListBata berfungsi mengelola untaian ke-9 jenis bata. Layanan yang diimplementasikan pada modul ini adalah: CreateList, DestroyList, Allocate, Deallocate, SearchPosisi, InsertFirst, InsertLast, DeleteP.

### 4.2.4 Modul Dialog

Modul Dialog terdiri dari 8 buah kelas & objek *form2*, yaitu: THICForm2 berikut 7 buah kelas & objek bagiannya. Ketujuh kelas & objek bagian tersebut adalah TButton; TLabel; TFilterComboBox; TFileListBox; TDirectoryListBox; TDriveComboBox; TEdit.

Modul Dialog merupakan modul dialog sistem-pemakai untuk keperluan tugas baca dan tulis berkas. Layanan yang diimplementasikan pada modul ini adalah: CreateDialogSave; CreateDialogRetrieve; Button1Click; dan Button2Click.

### 4.2.5 Modul Bantuan

Modul Bantuan terdiri dari 4 buah kelas & objek *form3*, yaitu: THICForm3 berikut 3 buah kelas &objek bagiannya. Ketiga kelas&objek bagian tersebut adalah: TButton; TScrollBox; TMemo.

Modul Bantuan merupakan modul dialog sistem-pemakai untuk keperluan pemberian bantuan cara menyusun jaringan dinamis pada DM. Layanan yang diimplementasikan pada modul ini adalah: Button1Click.

### 4.2.6 Modul DM

Modul DM terdiri dari 11 buah kelas & objek. Kelas & objek tersebut adalah: TDMCDM; TDM; TPapanPanel; TTMCDM; THICForm1 berikut 6 buah kelas & objek bagiannya. Ke-6 buah kelas & objek bagian dari THICForm1 adalah: TScrollBox; TPanel; TMainMenu ; TImage; TSpeedButton; dan TMenuItem.

Modul DM merupakan modul utama perangkat lunak DM. Modul ini berfungsi sebagai: antar muka pemakai, manajemen tugas yang dapat diaktifkan oleh pemakai, manajemen data jaringan-dinamis, dan melakukan proses menggambar serta menghapus jaringan-dinamis.

Untuk menjalankan fungsi di atas, modul ini memerlukan modul Dialog, modul Bantuan, modul KompBata, modul Bata, modul ListBata, dan implementasi dari sejumlah layanan utama.

Selain layanan create, destroy, dan set pada kelas & objek, diperlukan layanan utama yaitu: FormCreate; SetEnabledDM; SaveFile; RetrieveFile; GambarBata; HapusBata.

Layanan lain yang mendukung keberhasilan tugas dari layanan utama dapat dilihat pada buku acuan teknis bagian A.

## 4.3 Implementasi Modul

Setiap modul diimplementasikan sebagai sebuah unit. Unit pada Delphi disimpan dalam sebuah berkas berekstensi .PAS. Sedangkan *form* yang disusun pada sebuah unit akan disimpan dalam berkas terpisah yang berekstensi .DFM.

Modul KompBata disimpan dalam berkas KompBata.PAS, modul Bata disimpan dalam berkas Bata.PAS, modul ListBata disimpan dalam berkas ListBata.PAS, modul Dialog disimpan dalam berkas Dialog.PAS, modul Bantuan disimpan dalam berkas Bantuan.PAS, dan modul DM disimpan dalam berkas DM.PAS.

Tidak semua modul di atas memerlukan *form*. Modul-modul yang memerlukan *form* adalah: modul Bantuan; modul Dialog ; dan modul DM. Dengan demikian implementasi dari ke-3 buah modul tersebut selain disimpan dalam 3 buah berkas .PAS disimpan juga dalam 3 buah berkas .DFM. Ketiga buah berkas .DFM tersebut adalah: Bantuan.DFM; Dialog.DFM; dan DM.DFM.

Seluruh berkas .PAS dan berkas .DFM dikelola oleh sebuah proyek. Proyek pada Delphi disimpan dalam berkas berekstensi .DPR.

Berkas proyek yang dibangun untuk mengelola seluruh berkas .PAS dan berkas .DFM di atas adalah PDM.DPR

Berkas-berkas yang terkait dengan implementasi keenam modul dapat dilihat pada tabel 4.2.

**Tabel 4.2 Implementasi Modul-modul DM**

| No. | Nama Modul | Nama Berkas Proyek | Nama Berkas Implementasi | Keterangan Implementasi |
|---|---|---|---|---|
| 1 | KompBata | PDM.DPR | KompBata.PAS | berkas Pascal |
| 2 | Bata | PDM.DPR | Bata.PAS | berkas Pascal |
| 3 | ListBata | PDM.DPR | ListBata.PAS | berkas Pascal |
| 4 | Bantuan | PDM.DPR | Bantuan.PAS<br>Bantuan.DFM | berkas Pascal<br>berkas *form* Delphi |
| 5 | Dialog | PDM.DPR | Dialog.PAS<br>Dialog.DFM | berkas<br>berkas *form* Delphi |
| 6 | DM | PDM.DPR | DM.PAS<br>DM.DFM | berkas Pascal<br>berkas *form* Delphi |

# BAB V
# UJI PERANGKAT LUNAK DM

Pada bab ini penulis akan menyajikan uji perangkat lunak DM yang terdiri dari: faktor pengujian; kasus pengujian; hasil pengujian dan kesimpulan pengujian.

Tujuan dilakukan pengujian terhadap perangkat lunak adalah untuk mengurangi resiko yang melekat pada sistem karena kehadiran perangkat lunak tersebut di dalam sistem. Resiko diartikan sebagai kondisi yang dapat menimbulkan kerugian.

Terdapat 3 dimensi yang berkaitan dengan metodologi pengujian, yaitu: faktor pengujian, fase pengujian, dan proses pengujian. Faktor pengujian berkaitan langsung dengan resiko yang harus diperhatikan pada pengujian. Fase pengujian berkaitan dengan fase pengembangan perangkat lunak. Proses pengujian berkaitan dengan pemakaian alat bantu (*tools*) pengujian, teknik pengujian dan kriteria evaluasi yang dipakai dalam melakukan validasi, verifikasi dan atau pemeriksaan sistem aplikasi.

## 5.1 Faktor Pengujian

Faktor pengujian yang akan diakomodasikan pada pengujian DM adalah: Kebenaran (*Correctness*); Otorisasi (*Authorization*); Integritas Berkas (*File Integrity*); Tingkat Layanan (*Service Levels*); dan Kontrol Akses (*Acces Control*).

Faktor Kebenaran berkaitan dengan jaminan bahwa data dapat dimasukan, diproses, dan dikeluarkan secara lengkap dan akurat oleh sistem aplikasi. Pada sistem aplikasi DM, faktor tersebut ditunjukkan dengan keakuratan dan kelengkapan hasil proses menggambar dan menghapus gambar jaringan dinamis.

Faktor Otorisasi berkaitan dengan jaminan bahwa pemrosesan data berlangsung sesuai dengan keinginan manajemen. Pada sistem aplikasi DM, faktor itu ditunjukkan dengan otorisasi tugas terhadap status sunting. Pada saat status sunting adalah hapus, terdapat sejumlah tugas yang tidak boleh diaktifkan oleh pemakai. Demikian pula sebaliknya pada saat status sunting adalah gambar.

Faktor Integritas Berkas berkaitan dengan jaminan bahwa data yang disimpan dalam sistem aplikasi dapat diambil kembali tanpa terjadi perubahan. Pada sistem aplikasi DM, faktor ini ditunjukkan dengan proses SaveFile dan proses RetriveFile yang bekerja secara berbalikan.

Faktor Tingkat Layanan berkaitan dengan jaminan bahwa hasil diharapkan dari sistem aplikasi telah sesuai keinginan pemakai. Pada aplikasi DM, faktor ini ditunjukkan dengan pemeriksaan tugas-tugas yang dapat dilakukan oleh DM berikut hasilnya, serta membandingkan hasil pemeriksaan terhadap kemampuan yang semula diharapkan oleh pemakai terhadap DM.

Faktor Kontrol Akses berkaitan dengan prosedur keamanan yang mampu menjamin integritas data aplikasi serta program dari aktifitas yang tidak sah. Pada aplikasi DM, faktor ini ditunjukkan dengan

kemampuan sistem untuk mengantisipasi berkas eksternal yang telah diubah tidak sesuai dengan rancangan.

## 5.2 Kasus Pengujian

Pengujian dapat dilakukan melalui 2 buah pendekatan, yaitu: *black-box*; dan *white-box*. Pendekatan itu mempengaruhi penentuan kasus uji.

Pendekatan *black-box* merupakan pendekatan fungsional yang menguji apakah fungsi-fungsi sistem aplikasi telah bekerja dengan baik tanpa mempedulikan proses yang berlangsung di dalamnya. Sedangkan *white-box* merupakan pendekatan prosedural yang menguji bagaimana fungsi sistem dilakukan (struktur kendali).

Pada pengujian DM, penulis memakai pendekatan *black-box*. Hal ini lebih dikarenakan keterbatasan waktu. Pengujian terhadap kendali secara otomatis selalu penulis lakukan bersamaan dengan pengembangan kode program, meski tidak didokumentasikan.

Berikut ini akan disajikan sejumlah kasus uji yang dikembangkan pada setiap faktor uji. Tabel 5.1 memuat kasus uji untuk faktor kebenaran. Tabel 5.2 memuat kasus uji untuk faktor otorisasi. Tabel 5.3 memuat kasus uji untuk faktor integritas berkas. Tabel 5.4 memuat kasus uji untuk faktor tingkat layanan. Tabel 5.5 memuat kasus uji untuk faktor kontrol akses.

**Tabel 5.1 Kasus Uji Faktor Kebenaran DM**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 1 | RL-G-0 | TRL+THICForm1 | RL tergambar benar |
| 2 | RL-G-90 | TRL+THICForm1 | RL tergambar benar |
| 3 | RL-G-180 | TRL+THICForm1 | RL tergambar benar |
| 4 | RL-G-270 | TRL+THICForm1 | RL tergambar benar |
| 5 | RL-H-0 | TRL+THICForm1 | RL terhapus benar |
| 6 | RL-H-90 | TRL+THICForm1 | RL terhapus benar |
| 7 | RL-H-180 | TRL+THICForm1 | RL terhapus benar |
| 8 | RL-H-270 | TRL+THICForm1 | RL terhapus benar |
| 9 | RBKi-G-0 | TRBKI+THICForm1 | RBKI tergambar benar |
| 10 | RBKi-G-90 | TRBKI+THICForm1 | RBKI tergambar benar |
| 11 | RBKi-G-180 | TRBKI+THICForm1 | RBKI tergambar benar |
| 12 | RBKi-G-270 | TRBKI+THICForm1 | RBKI tergambar benar |
| 13 | RBKi-H-0 | TRBKI+THICForm1 | RBKI terhapus benar |
| 14 | RBKi-H-90 | TRBKI+THICForm1 | RBKI terhapus benar |
| 15 | RBKi-H-180 | TRBKI+THICForm1 | RBKI terhapus benar |
| 16 | RBKi-H-270 | TRBKI+THICForm1 | RBKI terhapus benar |
| 17 | RBKa-G-0 | TRBKA+THICForm1 | RBKA tergambar benar |
| 18 | RBKa-G-90 | TRBKA+THICForm1 | RBKA tergambar benar |
| 19 | RBKa-G-180 | TRBKA+THICForm1 | RBKA tergambar benar |
| 20 | RBKa-G-270 | TRBKA+THICForm1 | RBKA tergambar benar |
| 21 | RBKa-H-0 | TRBKA+THICForm1 | RBKA terhapus benar |
| 22 | RBKa-H-90 | TRBKA+THICForm1 | RBKA terhapus benar |

**Tabel 5.1 Kasus Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 23 | RBKa-H-180 | TRBKA+THICForm1 | RBKA terhapus benar |
| 24 | RBKa-H-270 | TRBKA+THICForm1 | RBKA terhapus benar |
| 25 | RS3Ki-G-0 | TRS3KI+THICForm1 | RS3KI tergambar benar |
| 26 | RS3Ki-G-90 | TRS3KI+THICForm1 | RS3KI tergambar benar |
| 27 | RS3Ki-G-180 | TRS3KI+THICForm1 | RS3KI tergambar benar |
| 28 | RS3Ki-G-270 | TRS3KI+THICForm1 | RS3KI tergambar benar |
| 29 | RS3Ki-H-0 | TRS3KI+THICForm1 | RS3KI terhapus benar |
| 30 | RS3Ki-H-90 | TRS3KI+THICForm1 | RS3KI terhapus benar |
| 31 | RS3Ki-H-180 | TRS3KI+THICForm1 | RS3KI terhapus benar |
| 32 | RS3Ki-H-270 | TRS3KI+THICForm1 | RS3KI terhapus benar |
| 33 | RS3Ka-G-0 | TRS3KA+THICForm1 | RS3KA tergambar benar |
| 34 | RS3Ka-G-90 | TRS3KA+THICForm1 | RS3KA tergambar benar |
| 35 | RS3Ka-G-180 | TRS3KA+THICForm1 | RS3KA tergambar benar |
| 36 | RS3Ka-G-270 | TRS3KA+THICForm1 | RS3KA tergambar benar |
| 37 | RS3Ka-H-0 | TRS3KA+THICForm1 | RS3KA terhapus benar |
| 38 | RS3Ka-H-90 | TRS3KA+THICForm1 | RS3KA terhapus benar |
| 39 | RS3Ka-H-180 | TRS3KA+THICForm1 | RS3KA terhapus benar |
| 40 | RS3Ka-H-270 | TRS3KA+THICForm1 | RS3KA terhapus benar |
| 41 | RS4-G-0 | TRS4+THICForm1 | RS4 tergambar benar |
| 42 | RS4-G-90 | TRS4+THICForm1 | RS4 tergambar benar |
| 43 | RS4-G-180 | TRS4+THICForm1 | RS4 tergambar benar |
| 44 | RS4-G-270 | TRS4+THICForm1 | RS4 tergambar benar |

**Tabel 5.1 Kasus Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 45 | RS4-H-0 | TRS4+THICForm1 | RS4 terhapus benar |
| 46 | RS4-H-90 | TRS4+THICForm1 | RS4 terhapus benar |
| 47 | RS4-H-180 | TRS4+THICForm1 | RS4 terhapus benar |
| 48 | RS4-H-270 | TRS4+THICForm1 | RS4 terhapus benar |
| 49 | Counter-G-0 | TCounter+THICForm1 | *Counter* tergambar benar |
| 50 | Counter-G-90 | TCounter+THICForm1 | *Counter* tergambar benar |
| 51 | Counter-G-180 | TCounter+THICForm1 | *Counter* tergambar benar |
| 52 | Counter-G-270 | TCounter+THICForm1 | *Counter* tergambar benar |
| 53 | Counter-H-0 | TCounter+THICForm1 | *Counter* terhapus benar |
| 54 | Counter-H-90 | TCounter+THICForm1 | *Counter* terhapus benar |
| 55 | Counter-H-180 | TCounter+THICForm1 | *Counter* terhapus benar |
| 56 | Counter-H-270 | TCounter+THICForm1 | *Counter* terhapus benar |
| 57 | FlipFlop-G-0 | TFlipFlop+THICForm1 | *FlipFlop* tergambar benar |
| 58 | FlipFlop-G-90 | TFlipFlop+THICForm1 | *FlipFlop* tergambar benar |
| 59 | FlipFlop-G-180 | TFlipFlop+THICForm1 | *FlipFlop* tergambar benar |
| 60 | FlipFlop-G-270 | TFlipFlop+THICForm1 | *FlipFlop* tergambar benar |
| 61 | FlipFlop-H-0 | TFlipFlop+THICForm1 | *FlipFlop* terhapus benar |
| 62 | FlipFlop-H-90 | TFlipFlop+THICForm1 | *FlipFlop* terhapus benar |
| 63 | FlipFlop-H-180 | TFlipFlop+THICForm1 | *FlipFlop* terhapus benar |
| 64 | FlipFlop-H-270 | TFlipFlop+THICForm1 | *FlipFlop* terhapus benar |
| 65 | Switching-G-0 | TSwitching+THICForm1 | *Switching* tergambar benar |
| 66 | Switching-G-90 | TSwitching+THICForm1 | *Switching* tergambar benar |

**Tabel 5.1 Kasus Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 67 | Switching-G-180 | TSwitching+THICForm1 | *Switching* tergambar benar |
| 68 | Switching-G-270 | TSwitching+THICForm1 | *Switching* tergambar benar |
| 69 | Switching-H-0 | TSwitching+THICForm1 | *Switching* terhapus benar |
| 70 | Switching-H-90 | TSwitching+THICForm1 | *Switching* terhapus benar |
| 71 | Switching-H-180 | TSwitching+THICForm1 | *Switching* terhapus benar |
| 72 | Switching-H-270 | TSwitching+THICForm1 | *Switching* terhapus benar |

Keterangan:

- RL-G-0 berarti: rel-lurus, status sunting = gambar, sudut rotasi = 0 derajat.
- RL tergambar benar berarti sesuai prosedur pemasangan bata RL yang sesungguhnya.
- RL terhapus benar berarti gambar RL terhapus tanpa meninggalkan bekas, dan tanpa menghapus gambar bata lain di sekitarnya.

**Tabel 5.2 Kasus Uji Faktor Otorisasi**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 1 | G-KlikItemGambar | THICForm1 | Item Gambar tidak dapat di-klik |
| 2 | H-KlikItemHapus | THICForm1 | Item Hapus tidak dapat di-klik |
| 3 | H-KlikButtonBata | THICForm1 | Button bata tidak dapat di-klik |
| 4 | H-KlikButtonRotasi | THICForm1 | Button rotasi tidak dapat di-klik |

Keterangan:

- G-KlikItemGambar berarti status = Gambar, klik item-menu Gambar
- H-KlikButtonBata berarti status = Hapus, klik button pilihan Bata

**Tabel 5.3 Kasus Uji Faktor Integritas Berkas**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 1 | Sunting-Tulis-Baru-Baca | THICForm1 + bata | Tampilan DM akhir = tampilan DM pertama disunting |

Keterangan:

- Sunting-Tulis-Baru-Baca berarti penguji melakukan penyuntingan jaringan dinamis, diikuti tugas Tulis ke dalam berkas, diikuti tugas Baru yang akan membersihkan memori, dan akhirnya diikuti tugas Baca terhadap berkas yang sama.
- Kriteria uji yang dipakai adalah kesamaan bentuk jaringan dinamis, antara gambar sebelum ditulis ke dalam berkas maupun gambar yang diambil dari berkas yang sama.

**Tabel 5.4 Kasus Uji Faktor Tingkat Layanan**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 1 | Idem Tabel 5.1 | Idem Tabel 5.1 | Idem Tabel 5.1 |
| 2 | KlikItemPetunjuk | THICForm3 | Isi teks cukup membantu pemakai untuk mengenal cara mengkonstruksi jaringan dinamis pada DM |
| 3 | JD-KlikItemTampil-KlikItemSkala-KlikItem40x40 | THICForm1 | Gambar jaringan dinamis akan ditampilkan dengan skala 40 (memperbesar) , tanpa mengubah susunan bata |
| 4 | JD-KlikItemTampil-KlikItemSkala-KlikItem30x30 | THICForm1 | Gambar jaringan dinamis akan ditampilkan dengan skala 30 (skala normal) tanpa mengubah susunan bata |

**Tabel 5.4 Kasus Uji Faktor Tingkat Layanan (sambungan)**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 5 | JD-KlikItemTampil-KlikItemSkala-KlikItem20x20 | THICForm1 | Gambar jaringan dinamis akan di-tampilkan dengan skala 20 (memperkecil) tanpa mengubah susunan bata |
| 6 | JD-KlikItemTampil-KlikItemSkala-KlikItem10x10 | THICForm1 | Gambar jaringan dinamis akan di-tampilkan dengan skala 10 (memperkecil) tanpa mengubah susunan bata |
| 7 | JD-DragScroller_ Vertikal | THICForm1 | Gambar jaringan dinamis akan terlipat sesuai ke arah atas-bawah |
| 8 | JD-DragScroller_ Horisontal | THICForm1 | Gambar jaringan dinamis akan terlipat sesuai ke arah kiri-kanan |
| 9 | JD-KlikItemTulis | THICForm1 | Berkas jaringan dinamis (.TXT) terbentuk |
| 10 | JD-KlikItemBaca (nama berkas=nama berkas no.9) | THICForm1 | Berkas dibaca, gambar jaringan semula akan ditampilkan persis sama baik ukuran maupun letak |

Keterangan:

- JD-KlikItemTampil-KlikItemSkala-KlikItem40x40 berarti kondisi jaringan dinamis tergambar, klik item-menu Tampil- subitem Skala - subitem 40x40.
- JD-DragScrollerVertikal berarti menyeret tombol *scroller* ke-atas atau ke-bawah.

**Tabel 5.5 Kasus Uji Faktor Kontrol Akses**

| No | Kasus Uji | Objek Uji | Kriteria Uji |
|---|---|---|---|
| 1 | UbahHeaderBerkas-KlikItemBaca (terhadap berkas tersebut) | TDMCDM | Sistem akan menolak. |
| 2 | UbahDataKoordinat_Berkas-KlikItemBaca (berkas sama) | TDMCDM | Jika absis dan ordinat masing-masing bukan kelipatan 30, maka sistem akan menolak. |
| 3 | UbahDataRotasiBer_kas-KlikItemBaca (terhadap berkas tersebut) | TDMCDM | Jika sudut rotasi ≠ 0 dan sudut rotasi ≠ 90 dan sudut rotasi ≠ 180, dan sudut rotasi ≠ 270 maka sistem akan menolak. |

Keterangan:

- UbahHeaderBerkas berarti mengubah header berkas eksternal dari sebuah jaringan dinamis (berkas .TXT)
- Sistem akan menolak berarti menolak perubahan itu, dan sistem tetap tidak gagal.

## 5.3 Hasil Pengujian dan Kesimpulan Pengujian

Seluruh pengujian berikut dilakukan pada fase uji (*test phase*). Hasil pengujian dapat dilihat pada tabel berikut. Tabel 5.6 memuat hasil pengujian faktor kebenaran. Tabel 5.7 memuat hasil pengujian faktor otorisasi. Tabel 5.8 memuat hasil pengujian faktor integritas berkas. Tabel 5.9 memuat hasil pengujian faktor tingkat layanan. Tabel 5.10 memuat hasil pengujian faktor kontrol akses.

Untuk pengujian faktor kebenaran, proses gambar dan hapus bata dilakukan pada bata yang diletakkan di samping sebuah bata sembarang.

Kesimpulan pengujian akan dideskripsikan di bawah setiap tabel dalam sejumlah paragrap.

**Tabel 5.6 Hasil Uji Faktor Kebenaran DM**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 1 | RL-G-0 | TRL+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 2 | RL-G-90 | TRL+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 3 | RL-G-180 | TRL+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 4 | RL-G-270 | TRL+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 5 | RL-H-0 | TRL+THICForm1 | gambar terhapus dengan baik |
| 6 | RL-H-90 | TRL+THICForm1 | gambar terhapus dengan baik |
| 7 | RL-H-180 | TRL+THICForm1 | gambar terhapus dengan baik |
| 8 | RL-H-270 | TRL+THICForm1 | gambar terhapus dengan baik |
| 9 | RBKi-G-0 | TRBKI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 10 | RBKi-G-90 | TRBKI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 11 | RBKi-G-180 | TRBKI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 12 | RBKi-G-270 | TRBKI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 13 | RBKi-H-0 | TRBKI+THICForm1 | gambar terhapus dengan baik |

**Tabel 5.6 Hasil Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 14 | RBKi-H-90 | TRBKI+THICForm1 | gambar terhapus dengan baik |
| 15 | RBKi-H-180 | TRBKI+THICForm1 | gambar terhapus dengan baik |
| 16 | RBKi-H-270 | TRBKI+THICForm1 | gambar terhapus dengan baik |
| 17 | RBKa-G-0 | TRBKA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 18 | RBKa-G-90 | TRBKA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 19 | RBKa-G-180 | TRBKA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 20 | RBKa-G-270 | TRBKA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 21 | RBKa-H-0 | TRBKA+THICForm1 | gambar terhapus dengan baik |
| 22 | RBKa-H-90 | TRBKA+THICForm1 | gambar terhapus dengan baik |
| 23 | RBKa-H-180 | TRBKA+THICForm1 | gambar terhapus dengan baik |
| 24 | RBKa-H-270 | TRBKA+THICForm1 | gambar terhapus dengan baik |
| 25 | RS3Ki-G-0 | TRS3KI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 26 | RS3Ki-G-90 | TRS3KI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |

**Tabel 5.6 Hasil Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 27 | RS3Ki-G-180 | TRS3KI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 28 | RS3Ki-G-270 | TRS3KI+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 29 | RS3Ki-H-0 | TRS3KI+THICForm1 | gambar terhapus dengan baik |
| 30 | RS3Ki-H-90 | TRS3KI+THICForm1 | gambar terhapus dengan baik |
| 31 | RS3Ki-H-180 | TRS3KI+THICForm1 | gambar terhapus dengan baik |
| 32 | RS3Ki-H-270 | TRS3KI+THICForm1 | gambar terhapus dengan baik |
| 33 | RS3Ka-G-0 | TRS3KA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 34 | RS3Ka-G-90 | TRS3KA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 35 | RS3Ka-G-180 | TRS3KA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 36 | RS3Ka-G-270 | TRS3KA+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 37 | RS3Ka-H-0 | TRS3KA+THICForm1 | gambar terhapus dengan baik |
| 38 | RS3Ka-H-90 | TRS3KA+THICForm1 | gambar terhapus dengan baik |
| 39 | RS3Ka-H-180 | TRS3KA+THICForm1 | gambar terhapus dengan baik |

**Tabel 5.6 Hasil Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 40 | RS3Ka-H-270 | TRS3KA+THICForm1 | gambar terhapus dengan baik |
| 41 | RS4-G-0 | TRS4+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 42 | RS4-G-90 | TRS4+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 43 | RS4-G-180 | TRS4+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 44 | RS4-G-270 | TRS4+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 45 | RS4-H-0 | TRS4+THICForm1 | gambar terhapus dengan baik |
| 46 | RS4-H-90 | TRS4+THICForm1 | gambar terhapus dengan baik |
| 47 | RS4-H-180 | TRS4+THICForm1 | gambar terhapus dengan baik |
| 48 | RS4-H-270 | TRS4+THICForm1 | gambar terhapus dengan baik |
| 49 | Counter-G-0 | TCounter+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 50 | Counter-G-90 | TCounter+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 51 | Counter-G-180 | TCounter+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 52 | Counter-G-270 | TCounter+THICForm1 | gambar benar, atau ditolak karena tidak cocok |

**Tabel 5.6 Hasil Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 53 | Counter-H-0 | TCounter+THICForm1 | gambar terhapus dengan baik* |
| 54 | Counter-H-90 | TCounter+THICForm1 | gambar terhapus dengan baik* |
| 55 | Counter-H-180 | TCounter+THICForm1 | gambar terhapus dengan baik* |
| 56 | Counter-H-270 | TCounter+THICForm1 | gambar terhapus dengan baik* |
| 57 | FlipFlop-G-0 | TFlipFlop+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 58 | FlipFlop-G-90 | TFlipFlop+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 59 | FlipFlop-G-180 | TFlipFlop+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 60 | FlipFlop-G-270 | TFlipFlop+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 61 | FlipFlop-H-0 | TFlipFlop+THICForm1 | gambar terhapus dengan baik* |
| 62 | FlipFlop-H-90 | TFlipFlop+THICForm1 | gambar terhapus dengan baik* |
| 63 | FlipFlop-H-180 | TFlipFlop+THICForm1 | gambar terhapus dengan baik* |
| 64 | FlipFlop-H-270 | TFlipFlop+THICForm1 | gambar terhapus dengan baik* |
| 65 | Switching-G-0 | TSwitching+THICForm1 | gambar benar, atau ditolak karena tidak cocok |

**Tabel 5.6 Hasil Uji Faktor Kebenaran DM (sambungan)**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 66 | Switching-G-90 | TSwitching+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 67 | Switching-G-180 | TSwitching+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 68 | Switching-G-270 | TSwitching+THICForm1 | gambar benar, atau ditolak karena tidak cocok |
| 69 | Switching-H-0 | TSwitching+THICForm1 | gambar terhapus dengan baik* |
| 70 | Switching-H-90 | TSwitching+THICForm1 | gambar terhapus dengan baik* |
| 71 | Switching-H-180 | TSwitching+THICForm1 | gambar terhapus dengan baik* |
| 72 | Switching-H-270 | TSwitching+THICForm1 | gambar terhapus dengan baik* |

Keterangan:

- Gambar benar atau ditolak karena tidak cocok, berarti jika memang letak bata benar maka bata akan digambar sesuai letak dan ukuran, namun jika letak bata tidak tepat maka sistem akan menolak dan gambar bata tersebut tidak dilakukan.
- Gambar terhapus dengan baik, berarti jika dilakukan klik di dalam area gambar bata maka gambar bata akan terhapus dengan baik tanpa menghapus gambar dari bata lain di sekitarnya.
- Gambar terhapus dengan baik*, berarti jika dilakukan klik di dalam area satuan bata pada sudut kiri atas gambar bata maka gambar bata akan terhapus dengan baik tanpa menghapus gambar dari bata lain di sekitarnya.

Hasil uji terhadap 72 buah kasus di atas menunjukkan bahwa proses menggambar dan menghapus jaringan dinamis dapat berlangsung dengan akurat dan lengkap.

**Tabel 5.7 Hasil Uji Faktor Otorisasi**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 1 | G-KlikItemGambar | THICForm1 | Item Gambar tertutup warna form dan tidak dapat di-klik |
| 2 | H-KlikItemHapus | THICForm1 | Item Hapus tertutup warna form dan tidak dapat di-klik |
| 3 | H-KlikButtonBata | THICForm1 | Button bata tetap tampak tetapi tidak berfungsi sebagaimana mestinya |
| 4 | H-KlikButtonRotasi | THICForm1 | Button rotasi tetap tampak tetapi tidak berfungsi sebagaimana mestinya |

Hasil uji terhadap seluruh kasus uji tersebut menunjukkan bahwa otorisasi tugas pada status gambar dan status hapus dapat berlangsung dengan baik.

**Tabel 5.8 Hasil Uji Faktor Integritas Berkas**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 1 | Sunting-Tulis-Baru-Baca | THICForm1 + bata | Tampilan rangkaian bata tidak berubah baik jumlah bata, jenis bata, ukuran bata dan letak susunan bata. |

Hasil uji terhadap kasus uji tersebut menunjukkan bahwa proses SaveFile dan RetrieveFile dapat bekerja berbalikan dengan baik. Dengan demikian, data yang disimpan oleh sistem aplikasi DM dapat diambil kembali tanpa perubahan.

**Tabel 5.9 Hasil Uji Faktor Tingkat Layanan**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 1 | Idem Tabel 5.6 | Idem Tabel 5.6 | Idem Tabel 5.6 |
| 2 | KlikItemPetunjuk | THICForm3 | Isi teks memang telah mencakup langkah utama untuk menggambar dan menghapus bata, meski tidak mendeskripsikan detil langkah yang harus dilakukan pemakai |
| 3 | JD-KlikItemTampil-KlikItemSkala-KlikItem40x40 | THICForm1 | Jaringan dinamis akan digambar dalam skala 40, tanpa mengubah jumlah bata, jenis bata, dan susunan bata. |
| 4 | JD-KlikItemTampil-KlikItemSkala-KlikItem30x30 | THICForm1 | Jaringan dinamis akan digambar dalam skala 30, tanpa mengubah jumlah bata, jenis bata, dan susunan bata. |
| 5 | JD-KlikItemTampil-KlikItemSkala-KlikItem20x20 | THICForm1 | Jaringan dinamis akan digambar dalam skala 20, tanpa mengubah jumlah bata, jenis bata, dan susunan bata. |
| 6 | JD-KlikItemTampil-KlikItemSkala-KlikItem10x10 | THICForm1 | Jaringan dinamis akan digambar dalam skala 10, tanpa mengubah jumlah bata, jenis bata, dan susunan bata. |

**Tabel 5.9 Hasil Uji Faktor Tingkat Layanan (sambungan)**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 7 | JD-DragScroller_ Vertikal | THICForm1 | Gambar jaringan dinamis akan terlipat ke-atas jika *scroller* diseret ke-bawah, dan akan terlipat ke-bawah jika *scroller* diseret ke-atas. |
| 8 | JD-DragScroller_ Horisontal | THICForm1 | Gambar jaringan dinamis akan terlipat ke-kanan jika *scroller* diseret ke-kiri, dan akan terlipat ke-kiri jika *scroller* diseret ke-kanan. |
| 9 | JD-KlikItemTulis | THICForm1 | Terbentuk sebuah berkas teks (.TXT) |
| 10 | JD-KlikItemBaca (nama berkas=nama berkas no.9) | THICForm1 | Berkas teks dibaca dan ditampilkan gambar jaringan dinamis yang sama dengan gambar jaringan dinamis pada saat akan dilakukan proses Tulis berkas. |

Hasil uji dari seluruh kasus uji menunjukkan bahwa fungsi bongkar-pasang bata, *zooming*, *scrolling*, dan penanganan berkas, dapat diakomodasikan dengan baik oleh DM.

**Tabel 5.10 Hasil Uji Faktor Kontrol Akses**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 1 | UbahHeaderBerkas-KlikItemBaca (terhadap berkas tersebut) | TDMCDM | Proses baca berkas dihentikan, dan muncul pesan dialog "Proses baca file gagal. File data anda telah berubah. Tekan OK". Setelah tombol OK ditekan maka sistem kembali ke kondisi semula. |

**Tabel 5.10 Hasil Uji Faktor Kontrol Akses (sambungan)**

| No | Kasus Uji | Objek Uji | Hasil Uji |
|---|---|---|---|
| 2 | UbahDataKoordinat_Berkas-KlikItemBaca (berkas sama), absis dan ordinat bukan kelipatan 30 | TDMCDM | Proses baca berkas dihentikan, dan muncul pesan dialog "Proses baca file gagal. File data anda telah berubah. Tekan OK". Setelah tombol OK ditekan maka sistem kembali ke kondisi semula. |
| 3 | UbahDataKoordinat_Berkas-KlikItemBaca (berkas sama), absis dan ordinat kelipatan 30 namun bata tidak dapat digambar karena menimpa bata lain | TDMCDM | Proses baca berkas diinterupsi sebentar dengan tampilan "..... menimpa bata lain. Bata XX tidak digambar. Tekan OK". Setelah tombol OK ditekan proses pembacaan dilanjutkan namun bata yang bersangkutan (yaitu: XX) tidak digambar.Sistem tidak gagal. |
| 4 | UbahDataRotasiBer_kas-KlikItemBaca (terhadap berkas tersebut) | TDMCDM | Proses baca berkas dihentikan, dan muncul pesan dialog "Proses baca file gagal. File data anda telah berubah. Tekan OK". Setelah tombol OK ditekan maka sistem kembali ke kondisi semula. |

Keterangan:

- Meski perubahan berkas tidak mengakibatkan sistem gagal, namun sistem tidak akan melakukan perbaikan berkas. Perbaikan berkas hanya bisa dilakukan oleh pemakai setelah melakukan perbaikan gambar jaringan dinamis, dan kemudian menyimpan ulang ke dalam berkas yang sama.

Hasil pengujian dari keempat kasus tersebut menunjukkan bahwa sistem aplikasi DM telah mengakodomasikan prosedur pengamanan sistem terhadap kerusakan atau perubahan berkas eksternal yang diperkirakan dapat mengganggu unjuk kerja sistem.

# BAB VI
# KESIMPULAN dan SARAN

Pada bagian ini akan disajikan kesimpulan dan saran yang dapat penulis rangkum selama proses pengembangan prototipe DM.

## 6.1 Kesimpulan

Kesimpulan penulis sajikan menjadi 2 bagian, yaitu: kesimpulan perancangan dan implementasi; dan kesimpulan uji perangkat lunak DM.

### 6.1.1 Kesimpulan Perancangan dan Implementasi

Selama proses OOA, OOD, dan OOP terhadap DM, penulis menarik beberapa kesimpulan. Beberapa kesimpulan tersebut adalah:

1. Penulis berhasil menyusun sebuah prototipe perangkat lunak DM yang dikembangkan berdasarkan metoda objek. Prototipe DM memiliki fungsi substitusi-alat terhadap sebuah jaringan dinamis, dan dirancang untuk memberikan dukungan terhadap sebuah penelitian tentang proses kognitif siswa.

2. Prototipe DM dikembangkan dengan menggunakan bahasa pemrograman Delphi versi 2.0. Seluruh berkas dikembangkan dengan memakai fasilitas Delphi secara utuh. Pemilihan bahasa Delphi didasarkan pada pertimbangan bahwa penulis telah cukup mengenal bahasa Pascal yang menjadi bahasa induk Delphi, dan penanganan antarmuka yang cukup baik.

File EXE yang dihasilkan dieksekusi di bawah sistem operasi Windows'95.

3. Metoda objek dipilih sebagai metoda pengembangan prototipe DM. Hal ini didasarkan pada pertimbangan utama terhadap karakteristik *reuseability*, yaitu kemudahan memakai kembali hasil analisis, rancangan, dan implementasi pada masa mendatang. Karakteristik ini sangat penulis butuhkan karena piranti (alat) jaringan dinamis itu sendiri masih berkembang, baik fisik maupun kemungkinan pemanfaatannya pada bidang pengajaran.

4. Pada fase OOA, analisis dilakukan dengan asumsi bahwa teknologi mendukung. Perhatian lebih difokuskan pada upaya memotret ranah masalah menjadi sebuah model.

5. Pada fase OOD, rancangan dilakukan dengan memperhatikan dukungan teknologi. Pada umumnya rancangan dilakukan dengan asumsi bahwa sistem akan diimplementasikan pada platform X, di bawah sistem operasi Y, dan dengan bahasa pemrograman Z.

6. Seluruh hasil pendefinisian pada fase OOA merupakan komponen PDC pada OOD. Pada fase OOD, perubahan hasil OOA masih diperlukan sebagai upaya menyesuaikan dengan dukungan teknologi, tanpa mengubah ranah masalah yang telah dimodelkan.

7. Pelaksanaan ketiga fase, yaitu: OOA, OOD, dan OOP, terhadap rekayasa DM tidak dapat lepas dari model *baseball* yang dikemukakan oleh Peter Coad [CoN-93,p.12]. Model ini mengakomodasikan prinsip pengembangan yang konkuren pada ketiga fase tersebut. Prinsip pengembangan yang konkuren penulis lakukan pada rekayasa DM lepas dari upaya untuk melakukan *trial & error*. Pertimbangan utama terhadap pemakaian prinsip tersebut adalah: pemakaian paradigma *prototyping*, dan keterbatasan pengalaman penulis dalam mengembangkan perangkat lunak yang berbasis objek.

### 6.1.2 Kesimpulan Uji Perangkat Lunak DM.

Berdasarkan rancangan faktor pengujian, kasus uji, dan hasil pengujian dapat ditarik beberapa kesimpulan berikut:

1. Prototipe DM dapat diharapkan cukup memberikan jaminan terhadap faktor kebenaran. Kesimpulan ini ditarik berdasarkan hasil uji pada tabel 5.6. Pada ke-72 kasus uji diperoleh hasil bahwa proses menggambar dan menghapus gambar jaringan dinamis dapat berlangsung dengan baik.

2. Prototipe DM dapat diharapkan cukup memberikan jaminan terhadap faktor otorisasi. Kesimpulan ini ditarik berdasarkan hasil uji faktor otorisasi pada tabel 5.7. Otorisasi tugas terhadap status sunting dapat berlangsung dengan baik.

3. Prototipe DM dapat diharapkan cukup memberi jaminan terhadap faktor integritas berkas data. Kesimpulan ini ditarik

berdasarkan hasil uji faktor tingkat integritas berkas pada tabel 5.8. Data yang telah disimpan dalam aplikasi dapat diambil kembali tanpa terjadi perubahan.

4. Prototipe DM cukup mengimplementasikan seluruh kebutuhan pemakai dengan baik. Hasil uji pada tabel 5.9 menunjukkan bahwa kemampuan yang diharapkan pada DM telah diimplementasikan dengan baik.

5. Prototipe DM dapat diharapkan cukup menjamin prosedur keamanan yang menjamin integritas data aplikasi dari aktifitas yang tidak sah. Meski aplikasi tidak melakukan tindakan perbaikan (*recovery*) terhadap berkas eksternal yang mengalami perubahan, kehadiran berkas tersebut sebagai berkas masukan tidak akan mengakibatkan sistem gagal.

## 6.2 Saran

Bertolak dari pengalaman mengembangkan prototipe DM, penulis menyajikan beberapa saran pengembangan. Beberapa saran tersebut adalah:

1. Melakukan eksplorasi lanjut pada Delphi versi 2.0 untuk mencari fasilitas yang dapat mengimplementasikan sebuah list-linier berkait dengan pointer. Penulis belum merasa puas terhadap implementasi list tersebut pada penulisan kode terhadap prototipe ini. Ketidakpuasan disebabkan pada cara mengkonsultasi sebuah *instance* yang dinilai agak panjang. Misalkan untuk mengkonsultasi *instance* RL yang diketahui

disimpan pada alamat L diperlukan perintah: L.Info.InfoRL, dan tidak cukup dengan L.Info saja.

2. Lengkapi prototipe DM dengan subsistem jelajah. Subsistem ini akan melakukan penjelajahan terhadap jaringan dinamis, dimulai dari pintu masuk jaringan dan seharusnya berakhir pada pintu ke luar jaringan. Bagian ini harus mengutamakan agar proses atau langkah yang dilakukan oleh subsistem dapat digambarkan dengan baik pada gambar jaringan dinamis yang telah ada. Dengan demikian dapat memberikan tambahan penjelasan proses bagi siswa.

3. Perlu ditambahkan fasilitas kemudahan lain yang sekiranya dapat lebih membantu pemakai untuk merancang jaringan dinamis. Fasilitas yang dimaksud adalah:

   - fasilitas UNDO, untuk membatalkan aksi yang dilakukan pemakai;
   - operasi terhadap blok gambar, seperti:
     - menghapus blok gambar,
     - menyisipkan blok gambar,
     - menggeser blok gambar,
     - menyisipkan file.

# DAFTAR PUSTAKA

[Mar-92] Marpaung Yansen : "Makalah Hasil Penelitian Peranan Modus Representasi dalam Proses Belajar Mengajar Algoritma di SMP", Pusat Penelitian Sanata Dharma Yogyakarta, Juni 1992.

[HUl-79] Hopcroft J.E., Ulman J.D. : "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley Publishing Company, 1979.

[Coh-77] Cohors Fresenborg E. : "Mathematik mit Kalkueien und Maschinen", Braunschweig: Vieweg, 1977.

[HoM-76] Hopkin David, Moss Barbara : "Automata", Elsevier North-Holland Publishing Co. Amsterdam, 1976.

[LeP-81] Lewis R. Harry, Papadimitriou H. Christos : "Elements of the Theory of Computation", Prentice Hall, 1981.

[May-92] Mayhew J. Deborah : "Principles and Guidelines in Software User Interface Design", Prentice Hall, 1992.

[CoY-91] Coad Peter, Yourdon Edward : "Object Oriented Analysis" second edition, Prentice Hall, 1991.

[CYo-91] Coad Peter, Yourdon Edward : "Object Oriented Design", Prentice Hall, 1991.

[CoN-93] Coad Peter, Nicola Jill : "Object Oriented Programming", Prentice Hall, 1993.

[EYo-94] Yourdon Edward : "Object Oriented Systems Design", Prentice Hall, 1994.

[CaM-95] Cantu Marco: "Mastering Delphi", SYBEX Inc., 1995.

[OGB] Osier Dan, Grobman Steve, Batson Steve : "Teach Yourself Delphi 2 in 21 Days", Sams Publishing.

[Rao-93] Rao R. Bindu: "C++ and the OOP Paradigm", McGrawHill, 1993.

[Per-88] Perry E.William: "A Structured Approach To Systems Testing", QED Information Sciences, Inc.

[Ano] Anonim: Borland Delphi for Windows , User's Guide, Borland International Inc.

[CoY-91] Coad Peter, Yourdon Edward : "Object Oriented Analysis" second edition, Prentice Hall, 1991.

[CYo-91] Coad Peter, Yourdon Edward : "Object Oriented Design", Prentice Hall, 1991.

[CoN-93] Coad Peter, Nicola Jill : "Object Oriented Programming", Prentice Hall, 1993.

[EYo-94] Yourdon Edward : "Object Oriented Systems Design", Prentice Hall, 1994.

[CaM-95] Cantu Marco: "Mastering Delphi", SYBEX Inc., 1995.

[OGB] Osier Dan, Grobman Steve, Batson Steve : "Teach Yourself Delphi 2 in 21 Days", Sams Publishing.

[Rao-93] Rao R. Bindu: "C++ and the OOP Paradigm", McGrawHill, 1993.

[Per-88] Perry E.William: "A Structured Approach To Systems Testing", QED Information Sciences, Inc.

[Ano] Anonim: Borland Delphi for Windows , User's Guide, Borland International Inc.

[CoY-91] Coad Peter, Yourdon Edward : "Object Oriented Analysis" second edition, Prentice Hall, 1991.

[CYo-91] Coad Peter, Yourdon Edward : "Object Oriented Design", Prentice Hall, 1991.

[CoN-93] Coad Peter, Nicola Jill : "Object Oriented Programming", Prentice Hall, 1993.

[EYo-94] Yourdon Edward : "Object Oriented Systems Design", Prentice Hall, 1994.

[CaM-95] Cantu Marco: "Mastering Delphi", SYBEX Inc., 1995.

[OGB] Osier Dan, Grobman Steve, Batson Steve : "Teach Yourself Delphi 2 in 21 Days", Sams Publishing.

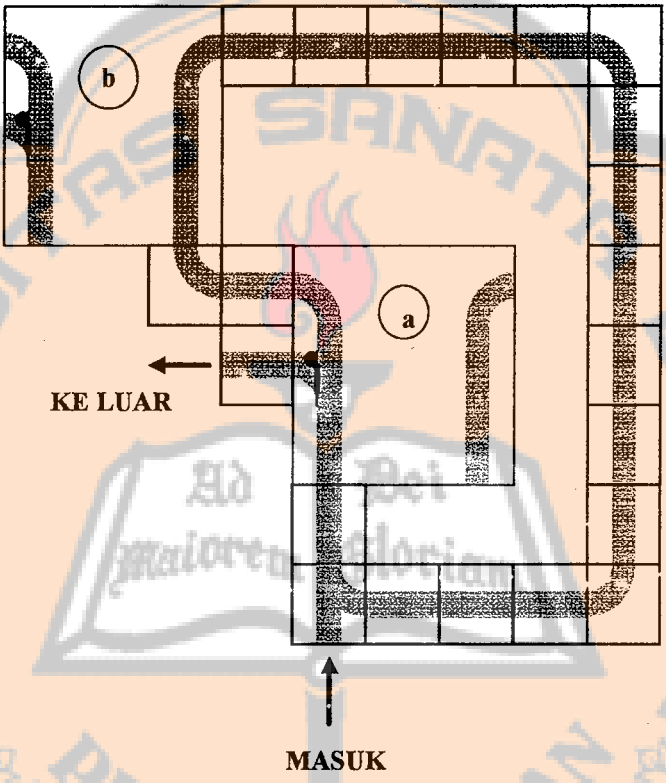
[Rao-93] Rao R. Bindu: "C++ and the OOP Paradigm", McGrawHill, 1993.

[Per-88] Perry E.William: "A Structured Approach To Systems Testing", QED Information Sciences, Inc.

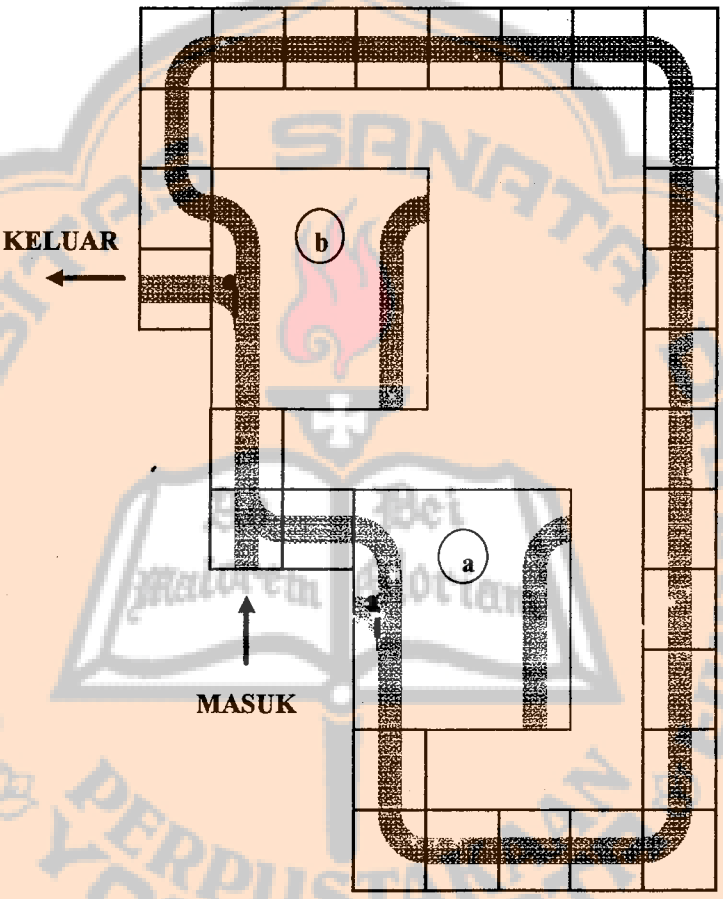
[Ano] Anonim: Borland Delphi for Windows , User's Guide, Borland International Inc.

# LAMPIRAN A

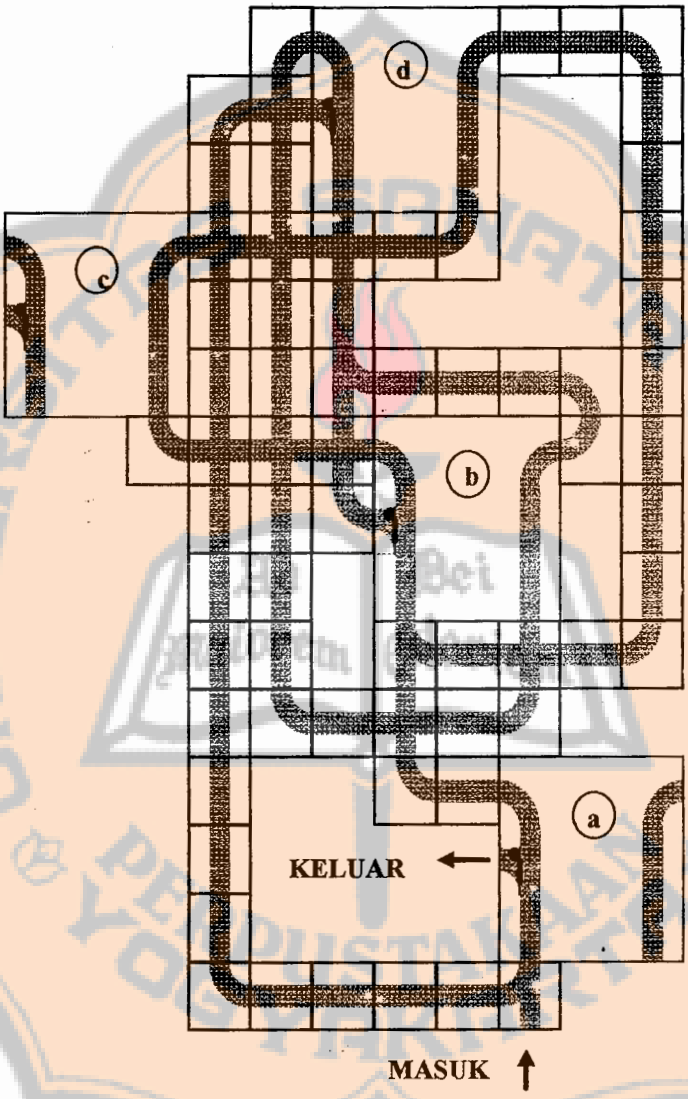
## Gambar Contoh Jaringan Dinamis



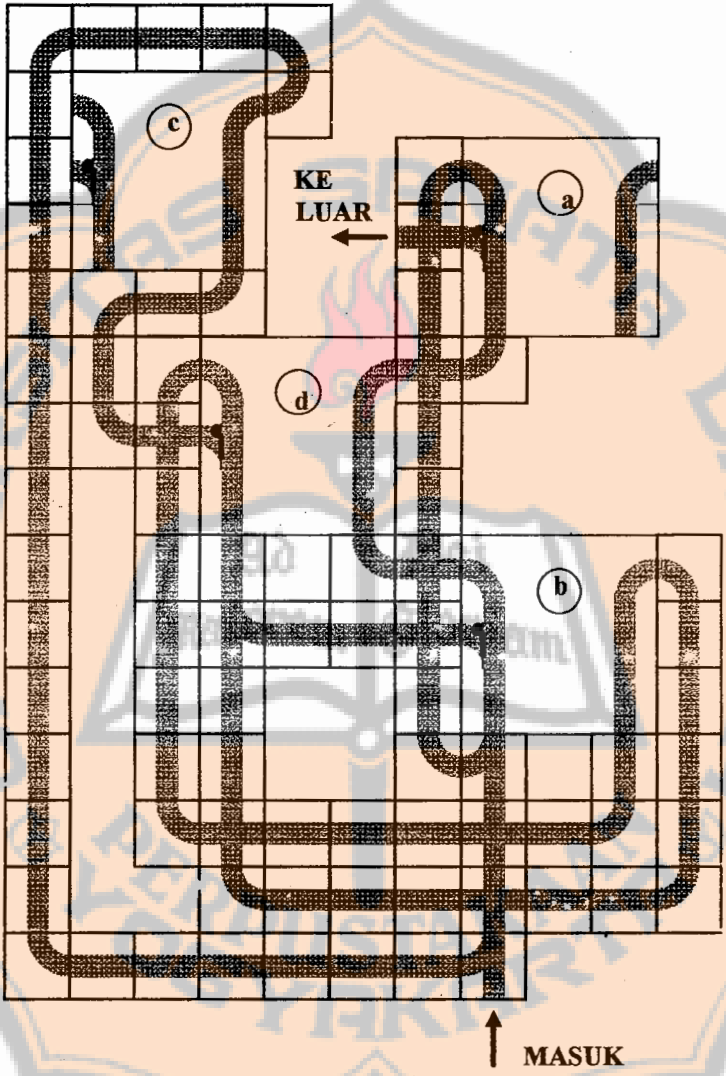

**Gambar A.1 Contoh Jaringan Dinamis untuk Operasi Penjumlahan ( b ← a + b)**




**Gambar A.2 Contoh Jaringan Dinamis untuk Operasi Pengurangan (a ← a - b , a >= b)**



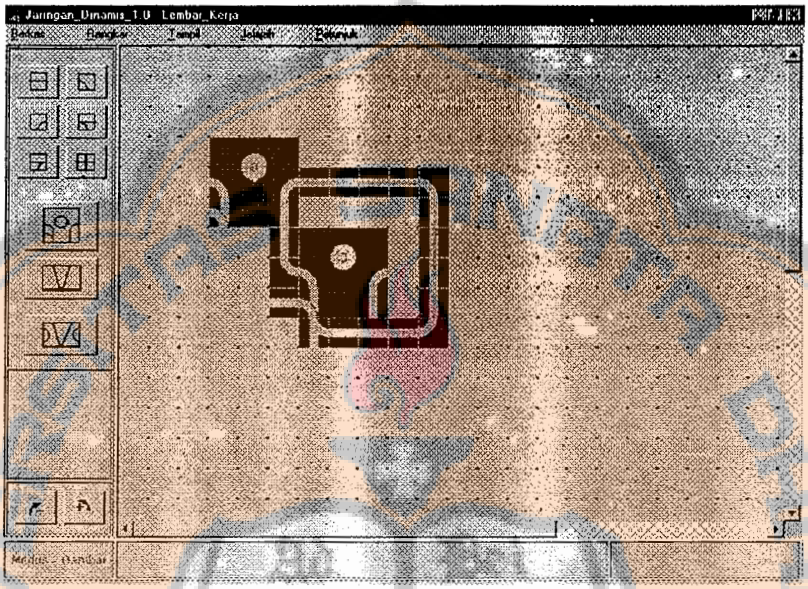

**Gambar A.3 Contoh Jaringan Dinamis untuk Operasi Perkalian ( c ← a x b)**



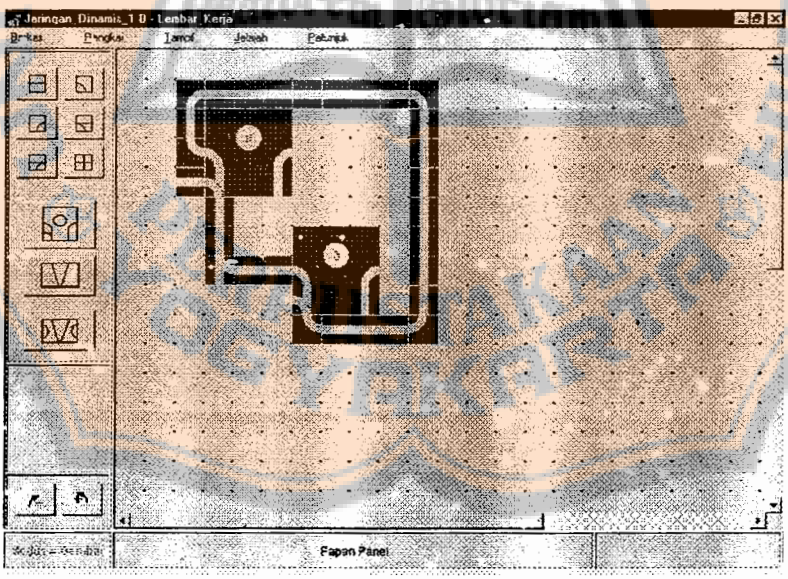

**Gambar A.4 Contoh Jaringan Dinamis untuk Operasi Pembagian (c←a:b, b≠0, a kelipatan b)**

**LAMPIRAN B**

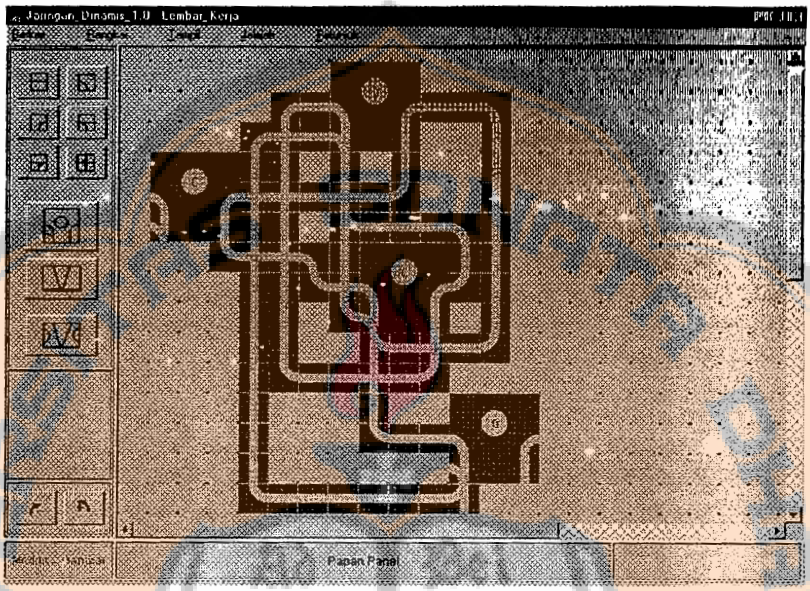
**Gambar Jaringan Dinamis pada DM**




**Gambar Jaringan Dinamis Operasi Penjumlahan pada DM**




**Gambar Jaringan Dinamis Operasi Pengurangan pada DM**




**Gambar Jaringan Dinamis Operasi Perkalian pada DM**

# LAMPIRAN C

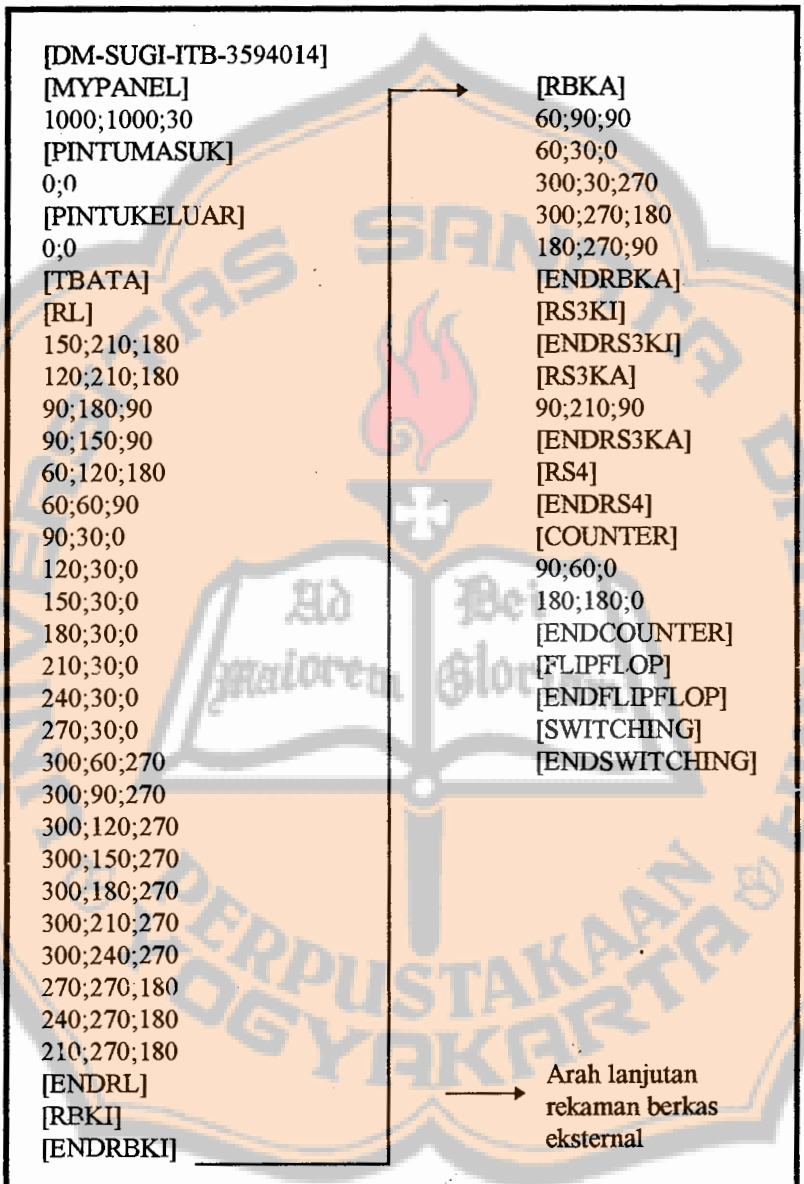
## Contoh Berkas Eksternal Jaringan Dinamis

```
[DM-SUGI-ITB-23594014]
[MYPANEL]
1000;1000;30
[PINTUMASUK]
0;0
[PINTUKELUAR]
0;0
[TBATA]
[RL]
150;180;90
210;120;0
240;120;0
270;120;0
300;150;270
300;180;270
300;210;270
300;240;270
270;270;180
240;270;180
210;270;180
150;240;180
180;120;0
[ENDRL]
[RBKI]
[ENDRBKI]
[RBKA]
150;210;90
300;120;270
300;270;180
[ENDRBKA]
[RS3KI]
[ENDRS3KI]
[RS3KA]
180;270;90
[ENDRS3KA]
```

→ [RS4]
[ENDRS4]
[COUNTER]
180;180;0
90;90;0
[ENDCOUNTER]
[FLIPFLOP]
[ENDFLIPFLOP]
[SWITCHING]
[ENDSWITCHING]

→ Arah lanjutan rekaman berkas eksternal

**Berkas Eksternal Jaringan Dinamis Penjumlahan pada DM**




**Berkas Eksternal Jaringan Dinamis Pengurangan pada DM**

```
[DM-SUGI-ITB-23594014]
[MYPANEL]
1000;1000;30
[PINTUMASUK]
0;0
[PINTUKELUAR]
0;0
[TBATA]
[RL]
300;450;0
270;450;0
240;450;0
210;450;0
180;450;0
150;450;0
120;420;270
120;390;270
120;360;270
120;330;270
120;300;270
120;270;270
120;240;270
120;180;270
120;120;270
180;60;180
180;90;180
150;120;270
180;150;0
210;120;90
240;150;0
270;120;90
210;210;90
240;180;180
270;180;180
300;180;180
330;210;90
150;180;270
150;240;270
150;270;270
150;300;270
180;330;0
210;330;0
300;390;180
270;390;180
240;360;90
270;300;180
270;330;0
330;300;180
360;270;270
360;240;270
360;210;270
360;180;270
360;150;270
360;120;270
360;90;270
300;60;0
330;60;0
[ENDRL]
[RBKI]
120;450;180
120;90;90
150;60;90
270;150;270
330;180;0
330;240;270
180;210;0
150;330;180
300;330;270
[ENDRBKI]
[RBKA]
90;210;90
210;270;90
180;240;90
240;390;90
360;300;180
360;60;270
[ENDRBKA]
[RS3KI]
330;450;270
[ENDRS3KI]
[RS3KA]
210;180;90
240;300;90
[ENDRS3KA]
[RS4]
120;210;180
120;150;270
150;90;180
150;150;270
210;150;0
210;240;90
150;210;180
240;330;0
300;300;90
[ENDRS4]
[COUNTER]
210;30;0
30;120;0
240;210;0
330;360;0
[ENDCOUNTER]
[FLIPFLOP]
[ENDFLIPFLOP]
[SWITCHING]
[ENDSWITCHING]
```

→ Arah lanjutan rekaman berkas eksternal

**Berkas Eksternal Jaringan Dinamis Perkalian pada DM**