

Perancangan Software Sistem Informasi Akademik FTUI



December 06, 2001

AAG. Indra Pramana
Cipto Asido Sidabalok
Jefri C. Sormin
Minnarto Djojo

Arcle Technologies
<http://www.arcle.net>

SIMPLE RELIABLE SOLUTIONS



DAFTAR ISI

1. Pendahuluan	4
1.1. Fungsi Software	4
1.2. Unjuk kerja dan keandalan	4
1.3. Batasan	4
1.4. Antarmuka	4
2. Perencanaan Software Project	5
2.1. Model estimasi	5
2.2. Susunan organisasi	6
3. Manajemen Resiko	7
3.1. Resiko skala produk	7
3.2. Resiko pengaruh bisnis	7
3.3. Resiko yang berhubungan dengan pelanggan	8
3.4. Resiko proses	9
3.5. Resiko teknologi	10
3.6. Resiko peralatan pengembangan	10
3.7. Resiko yang berhubungan dengan jumlah staf dan pengalaman.	10
3.8. Resiko komponen dan pengendali.	11
4. Metoda Desain Software	13
4.1. Data Design	13
4.2. Architectural Design	14
4.3. Interface Design	17
4.4. Procedural Design	17
5. Testing Software Seminar / Skripsi	19
5.1. Interface	20
5.2. Input	20
5.3. Dokumentasi	20
5.4. Pengujian pemulihan terhadap kegagalan software	21
5.5. Pengujian security	21
5.6. Pengujian unjuk kerja	21
5.7. Stress testing	22



6. Maintenance	22
6.1. Bersifat korektif	22
6.2. Bersifat adaptif	22
6.3. Bersifat perbaikan (enhancement)	22
6.4. Bersifat reengineer	23
7. Kesimpulan	23



1. Pendahuluan

1.1. Fungsi Software

Software untuk sistem informasi akademik khususnya seminar dan skripsi ini bertujuan untuk memberikan informasi tentang segala sesuatu yang berkaitan dengan seminar / skripsi yang ada di jurusan Elektro FTUI (Fakultas Teknik Universitas Indonesia). Software ini ditujukan untuk staf karyawan administrasi jurusan Elektro FTUI dan mahasiswa/i yang ingin menggunakan informasi tersebut untuk setiap kegiatan yang berkaitan dengan seminar / skripsi sehingga dapat membantu memudahkan mahasiswa/i dalam mengurus seminar / skripsi.

1.2. Unjuk kerja dan keandalan

Unjuk kerja dari software yang akan dibuat diharapkan lebih baik dari software yang telah ada sebelumnya, terutama dalam hal kecepatan.

1.3. Batasan

Proses pengembangan software ini dihadapkan pada beberapa masalah, antara lain:

- Waktu pengembangan yang terbatas.
- Dana yang disiapkan untuk pengembangan software sangat terbatas.
- Software yang akan dikembangkan diharapkan tidak mengubah struktur sistem informasi akademik secara keseluruhan.
- Ruang lingkup software yang akan dikembangkan hanya terbatas pada seminar / skripsi.

1.4. Antarmuka

Antarmuka yang disiapkan dalam software ini antara lain:

- *User interface* berupa *computer graphics display*.
- Antarmuka dengan perangkat keras lainnya seperti printer dan *database server*.
- Antarmuka dengan perangkat lunak lainnya.



2. Perencanaan Software Project

2.1. Model estimasi

Dalam sistem informasi akademik khususnya seminar dan skripsi terdapat beberapa masalah, antara lain seperti ditunjukkan pada Tabel 1.

Tabel 1. Masalah-masalah yang dihadapi pada seminar / skripsi

No.	Masalah	Skala Prioritas
1	Penilaian	20
2	Topik seminar / skripsi	19
3	Pendaftaran periode sidang skripsi	18
4	Pembimbing	17
5	Jadwal sidang skripsi	17
6	Jadwal seminar	16
7	Jadwal pertemuan bimbingan	14
8	Konsultasi	14
9	Progress pengerjaan	14
10	Ijin penggunaan fasilitas untuk penelitian	14
11	Syarat awal (batas SKS)	13
12	Pendaftaran seminar / skripsi	12
13	Penyerahan / pengumpulan buku seminar / skripsi	12

Dari masalah-masalah yang dihadapi di Tabel 1, maka dapat dibuat fungsi-fungsi yang mendukung sistem, antara lain:

- *User interface and control facilities (UICF)*
- *Database management (DBM)*
- *Computer graphics display facilities (CGDF)*
- *Peripheral Control (PC)*

Berdasarkan fungsi-fungsi pendukung dapat dibuat estimasi LOC (*Line of Codes*) dari fungsi-fungsi tersebut, seperti yang ditunjukkan Tabel 2.

Tabel 2. Fungsi-fungsi pendukung

Fungsi	Estimasi (LOC)
User interface and control facilities (UICF)	1200
Database management (DBM)	1500
Computer graphics display facilities (CGDF)	1700
Peripheral control (PC)	1000
<i>total</i>	5400



Berdasarkan estimasi LOC dari tiap-tiap fungsi, maka dapat dihitung model estimasi dengan menggunakan Basic COCOMO model tipe *software project organic*. Persamaan Basic COCOMO model dinyatakan sebagai berikut:

$$E = a_b KLOC^{b_b}$$

$$D = c_b E^{d_b}$$

Tabel 3. Basic COCOMO model

Software Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Berdasarkan Tabel 3 maka diperoleh:

$$E = 2,4 (KLOC)^{1,05}$$

$$= 2,4 (5,4)^{1,05}$$

$$= 14 \text{ orang-bulan}$$

$$D = 2,5 E^{0,38}$$

$$= 2,5 (14)^{0,38}$$

$$= 6,8 \text{ bulan}$$

Dengan demikian dapat ditentukan jumlah orang yang akan terlibat pembuatan software (N), dengan menggunakan persamaan:

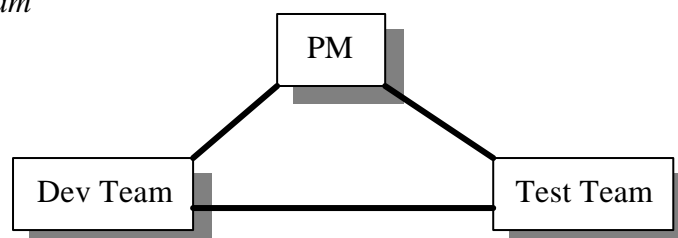
$$N = \frac{E}{D} = \frac{14}{6,8} \approx 2 \text{ orang}$$

Pada kenyataannya, perencana software akan menambah orang menjadi empat orang dan mempersingkat waktu pengerjaan software.

2.2. Susunan organisasi

Susunan organisasi tim pengembang software adalah:

- *Project Manager (PM)*
- *Development Team*
- *Testing Team*



Gambar 1. Struktur organisasi



3. Manajemen Resiko

3.1. Resiko skala produk

- Kesalahan perkiraan LOC

Resiko ini terjadi jika perkiraan LOC pada kenyataan yang ada jauh melebihi LOC perkiraan pada perhitungan COCOMO, yang mengakibatkan berubahnya jadwal pengerjaan dan biaya produksi.

Solusi :

- Melakukan re-scheduling dan melaporkannya kepada pelanggan, dan meminta pelanggan untuk melakukan penyesuaian jadwal dan biaya.
- Merekrut staff tambahan untuk membantu penyelesaian produksi agar sesuai jadwal.

- Kesalahan perancangan skala database

Resiko ini terjadi karena skala database yang telah dirancang ternyata tidak sesuai pada proses pengembangan software tersebut. Yang dapat berakibat berubahnya waktu penyelesaian produk.

Solusi :

- Memberitahu seluruh staff yang bersangkutan untuk melakukan meeting sehubungan dengan perancangan ulang skala database.
- Memberitahu pelanggan apabila terjadi perubahan waktu penyelesaian produk.
- Merekrut staff tambahan untuk membantu penyelesaian produksi agar sesuai jadwal.

3.2. Resiko pengaruh bisnis

- Dokumentasi untuk pelanggan dianggap kurang baik oleh pelanggan

Resiko ini terjadi apabila dokumentasi dari produk yang kita berikan kepada pelanggan tidak seperti yang diharapkan pelanggan. Sehingga pelanggan tidak puas dan kemungkinan tidak bisa mengoperasikan produk tersebut.

Solusi :

- Untuk pencegahannya sejak awal kita harus sudah melakukan riset bagaimana membuat dokumentasi yang baik, dan juga kita perlu untuk menanyakan pelanggan dokumentasi seperti apa yang diharapkan oleh pelanggan.



- Jika resiko tersebut sudah terjadi, kita dapat meminta kepada pelanggan untuk memberikan respon maupun review atas dokumentasi tersebut, dan berusaha memperbaikinya.

- Ketergantungan harga akibat keterlambatan produksi

Resiko ini dimungkinkan apabila terdapat kesepakatan yang menyatakan bahwa harga akan berubah jika terjadi keterlambatan produksi.

Solusi :

- Untuk menghindarinya proses produksi harus dijadwalkan dengan baik dan prosesnya diusahakan semaksimal mungkin sesuai dengan time table-nya.

- Ketergantungan harga akibat ketidaksempurnaan produk.

Resiko ini artinya harga akan berubah akibat produk yang dihasilkan tidak sesuai dengan kesepakatan (tidak sempurna/terdapat cacat).

Solusi :

- Sebelum diberikan kepada pelanggan maka produk harus terlebih dahulu menjalani tahap testing untuk melihat kelemahan-kelemahan yang mungkin terdapat dalam produk tersebut.

3.3. Resiko yang berhubungan dengan pelanggan

- Pelanggan belum/tidak bisa memastikan apa yang dibutuhkannya.

Resiko ini dikarenakan oleh pelanggan yang tidak tahu pasti apa yang dibutuhkannya, mungkin akibat pelanggan belum mempersiapkan diri atau memang produksi software tidak sesuai dengan bidang pelanggan.

Solusi :

- Menggali informasi tentang keadaan pelanggan, apa yang dibutuhkan. Sehingga kita bisa membantu memberikan saran tentang apa yang dia butuhkan.

- Meminta kepada pelanggan untuk mempersiapkan kebutuhannya terlebih dahulu, atau meminta kepada pelanggan untuk bertemu orang dari perusahaan pelanggan yang mengerti proses produksi software (staff EDP perusahaan pelanggan).

- Pelanggan tidak punya waktu untuk berkomunikasi dengan developer untuk saling memberi informasi.

Pelanggan tidak dapat berkomunikasi dengan developer untuk memberitahu apa yang pelanggan butuhkan dan apakah progress yang telah dilakukan telah sesuai dengan keinginan pelanggan.



Solusi :

- Meminta kepada pihak pelanggan beberapa contactperson, sehingga orang dari pihak pelanggan yang dapat dimintai keterangan lebih banyak.
- Membuat report dan simulasi yang sesingkat dan sedetil mungkin supaya pelanggan dapat cepat melakukan koreksi sehingga tidak menyita banyak waktunya.
- Pelanggan tidak mengerti proses pengembangan software.

Akibat pelanggan tidak mengerti proses pengembangan software, maka persepsi pelanggan tentang jadwal produksi, besar project, sulit tidaknya suatu project kemungkinan salah. Dan mereka dapat meminta sesuatu yang diluar sewajarnya.

Solusi :

- Memberikan gambaran global tentang proses pengembangan software, dan memberikan keterangan sesederhana dan sedetil mungkin.

3.4. Resiko proses

- Tidak semua staff bersedia untuk mengikuti proses yang telah ditentukan.

Resiko kemungkinan staff tidak mampu untuk mengikuti prosedur yang telah ditentukan, baik karena alasan skill, waktu, dan lain-lain.

Solusi :

- Menyusun ulang pembagian tugas dari masing-masing staff.
- Mengganti staff tersebut dengan yang lebih mampu
- Kesulitan pengaturan jadwal untuk melakukan review teknis.

Para staff memiliki jadwal yang berbeda-beda sehingga mereka tidak dapat melakukan meeting.

Solusi :

- Project Manager harus mengatur jadwal yang tepat untuk masing-masing staff.
- Masing-masing staf sejak awal harus berkomitmen untuk meluangkan waktunya.
- Metode testing yang ada kurang sesuai.

Metode Pengetesan yang dilakukan ternyata tidak dapat diterapkan pada software tersebut.

Solusi :

- Mencari metode testing yang lebih baik, dengan cara merapatkannya dengan staff, mencarinya di internet dan sumber-sumber lainnya.



3.5. Resiko teknologi

- Tidak semua staff menguasai/telah mengenal tools yang akan digunakan.

Para staff memiliki kemampuan yang berbeda-beda sehingga tools yang mereka kuasai juga tidak sama.

Solusi :

- Memilih staff yang telah mengenal tools tersebut.
- Mengadakan training singkat untuk staff yang belum menguasainya.
- Project membutuhkan hal baru yang belum pernah dibuat oleh developer.

Solusi :

- Mengumpulkan bahan-bahan yang diperlukan dan melakukan riset untuk persiapan project tersebut.

3.6. Resiko peralatan pengembangan

- Software project/proses manajemen tidak tersedia.

Solusi :

- Mencari software tersebut terlebih dahulu.
- Tidak tersedianya software development untuk yang dibutuhkan.
- Mencari alternatif software yang dapat menggantikannya.
- Mencari software tersebut terlebih dahulu.
- Tidak tersedianya dokumentasi yang cukup untuk peralatan yang digunakan.
- Mencari dokumentasi dari buku, internet, majalah, dan sebagainya.
- Staf tidak terlatih untuk menggunakan tools yang ada.
- Mengadakan training singkat mengenai tools tersebut.
- Memberikan buku panduan yang harus dipelajari sendiri oleh staff tersebut.

3.7. Resiko yang berhubungan dengan jumlah staf dan pengalaman.

- Tidak tersedianya kombinasi staff dengan kemampuan yang tepat.
- Menyusun ulang pembagian tugas antar staf.
- Merekrut staf baru yang memiliki kemampuan seperti yang dibutuhkan.
- Jumlah staff tidak memadai.
- Merekrut staf baru.
- Mengefektifkan kerja para staf.
- Menambah jumlah jam kerja setiap staf.



- Staf mengundurkan diri.
- Merekrut staff baru.
- Mengefektifkan kerja para staf.
- Menambah jumlah jam kerja setiap staf.

3.8. Resiko komponen dan pengendali.

- Performa produk tidak seperti yang diharapkan.

Solusi :

- Melakukan kompilasi ulang dengan mengoptimalkan dan memperbaiki software.

- Harga produk di luar perkiraan.

Solusi :

- Melakukan pencegahan dengan menyediakan dana cadangan dalam estimasi.

- Software tidak bisa dikoreksi/diubah.

Solusi :

- Melakukan kompilasi software dari awal dengan terlebih dahulu memperbaikinya.

- Project di luar jadwal yang ditentukan.

Solusi :

- Melakukan pencegahan dengan menambahkan waktu pada estimasi.

- Meminta tambahan waktu pada pelanggan.

Rangkuman dari resiko-resiko yang telah dijelaskan di atas tampak pada Tabel 4.

Tabel 4. Tabel resiko

Resiko	Kategori	Kemungkinan	Dampak
Kesalahan perkiraan LOC	Skala produk	40%	2
Kesalahan perancangan database	Skala produk	30%	2
dokumentasi untuk pelanggan kurang baik	Pengaruh bisnis	20%	3
keterlambatan produksi	Pengaruh bisnis	50%	3
ketidaktepatan produk	Pengaruh bisnis	20%	3
pelanggan tidak bisa memastikan apa yang dibutuhkannya	Hub. Pelanggan	60%	4



Sulitnya komunikasi pelanggan dengan developer	Hub. Pelanggan	40%	4
Pelanggan tidak mengerti proses pengembangan software	Hub. Pelanggan	70%	4
Staff tidak bersedia untuk mengikuti proses yang telah ditentukan	Proses	40%	1
Kesulitan pengaturan jadwal untuk melakukan review teknis	Proses	30%	2
Metode testing yang ada kurang sesuai	Proses	30%	2
staff tidak menguasai tools yang akan digunakan	Teknologi	25%	1
project membutuhkan hal baru yang belum pernah dibuat oleh developer	Teknologi	25%	2
software project/proses manajemen tidak tersedia	Peralatan	20%	2
tidak tersedianya software development untuk yang dibutuhkan	Pengembangan		
tidak tersedianya peralatan	Peralatan	20%	2
dokumentasi yang cukup untuk peralatan yang digunakan	Pengembangan		
staff tidak terlatih untuk menggunakan tools yang ada	Peralatan	40%	2
tidak tersedianya kombinasi staff dengan kemampuan yang tepat	Pengembangan		
	Jumlah staff & pengalaman	40%	2



Jumlah staff tidak memadai	Jumlah staff & pengalaman	20%	2
staff mengundurkan diri	Jumlah staff & pengalaman	20%	1
performa produk tidak seperti yang diharapkan	Komponen & pengendali	50%	4
harga produk diluar perkiraan	Komponen & pengendali	30%	3
software tidak bisa dikoreksi / diubah	Komponen & pengendali	20%	4
project diluar jadwal yang ditentukan	Komponen & pengendali	60%	4

Nilai dampak :

- 1 – parah
- 2 – serius
- 3 – sedang
- 4 – dapat diabaikan

4. Metoda Desain Software

Desain software mencakup beberapa hal, antara lain :

- Desain data (data design)
- Desain arsitektur (architectural design)
- Desain antar muka (interface design)
- Desain prosedural (procedural design)

4.1. Data Design

Data design merupakan langkah awal dalam melakukan desain software. Tujuan dari dilakukannya data design ialah untuk mendapatkan struktur data yang baik sehingga diperoleh program yang lebih modular dan mengurangi kompleksitas pengembangan software.



Beberapa hal yang perlu diperhatikan dalam melakukan data design antara lain :

1. Prinsip analisis secara sistematis yang dapat dilakukan terhadap fungsionalitas dan tingkah laku software juga dilakukan terhadap data. Pada tahap ini tiap objek yang merepresentasikan data dalam software akan diperjelas keterhubungannya satu sama lain untuk mengetahui bagaimana aliran data dalam software.
2. Dilakukan identifikasi terhadap semua struktur data dan proses yang akan dioperasikan. Untuk tahap identifikasi ini sebaiknya digunakan tipe data abstrak untuk mempermudah dalam melakukan desain software.
3. Dibuat spesifikasi data yang menunjukkan keterhubungan dan aturan-aturan bagi tiap elemen dalam struktur data.
4. Data desain yang lebih mengarah ke tingkat implementasi sebaiknya ditunda sampai tahap akhir dari desain software.
5. Modularitas dari struktur data harus dirancang dengan baik. Dalam perancangan representasi suatu struktur data sebaiknya hanya dapat diakses oleh suatu modul yang benar-benar memerlukan akses secara langsung terhadap data yang terdapat dalam struktur tersebut.
6. Dibuat suatu *library* yang berisikan kumpulan struktur-struktur data yang sering dipergunakan beserta dengan operasi-operasi yang berkaitan dengan struktur data tersebut.
7. Desain software dan bahasa pemrograman yang digunakan harus mampu menangani spesifikasi dan implementasi dari bentuk tipe data abstrak yang telah dirancang.

4.2. Architectural Design

Tujuan dari dilakukannya desain arsitektur ialah untuk mengembangkan suatu struktur program yang modular dan memperjelas kontrol proses yang terjadi antar tiap modul. Desain arsitektur akan menggabungkan antara struktur program dan struktur data, sehingga didapatkan suatu antar muka yang mengatur aliran informasi dalam program.

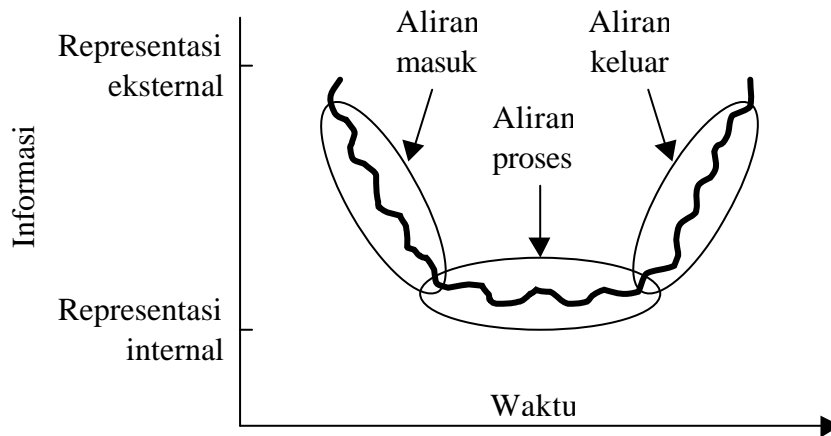
Secara umum, aliran informasi dalam suatu program terdiri dari aliran masuk (*incoming flow*), aliran proses (*transform flow*), dan aliran keluar (*outgoing flow*). Representasi data pada masing-masing tingkat aliran akan berbeda. Misalnya, pada tingkat aliran masuk dan aliran keluar (berupa input dan output), representasi data bersifat eksternal, yaitu data



berasal dari dunia luar. Input data diperoleh dari keyboard, mouse, dan perangkat eksternal lainnya. Sedangkan output data dikirimkan ke monitor, printer, dan sebagainya.

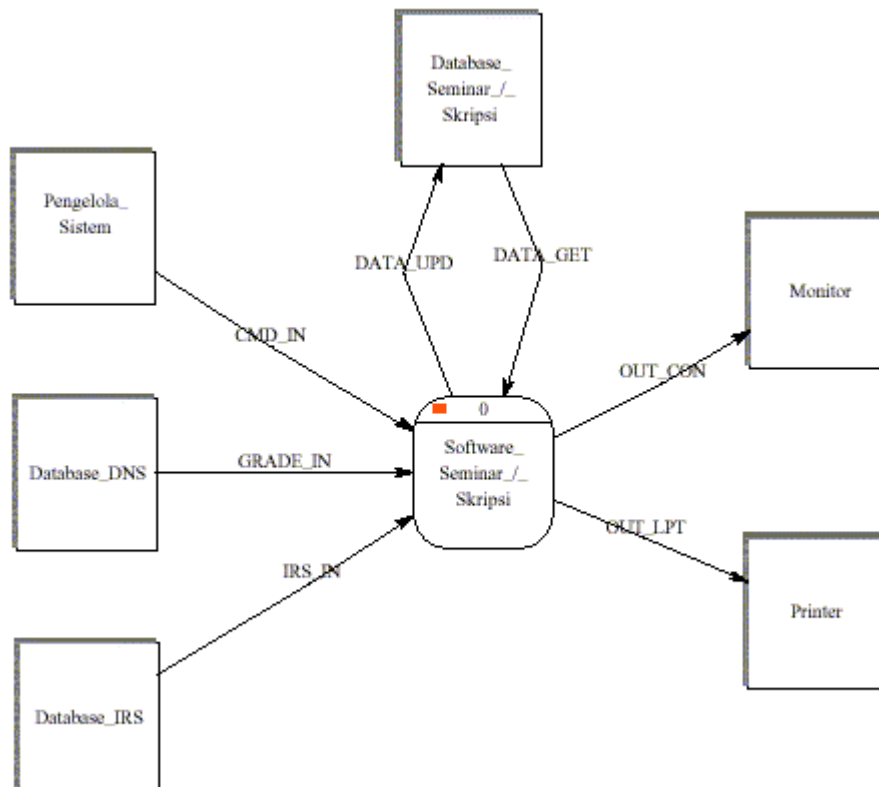
Pada aliran proses (transform flow), data merupakan representasi internal dalam program yang tergantung pada desain data yang telah dilakukan pada tahap desain sebelumnya.

Berdasarkan penjelasan di atas, maka aliran informasi dan representasi data dalam program akan dapat digambarkan dalam bentuk diagram sebagai berikut :



Hasil dari desain arsitektur ialah berupa suatu Data Flow Diagram (DFD) yang merupakan diagram yang menunjukkan aliran data dalam program.

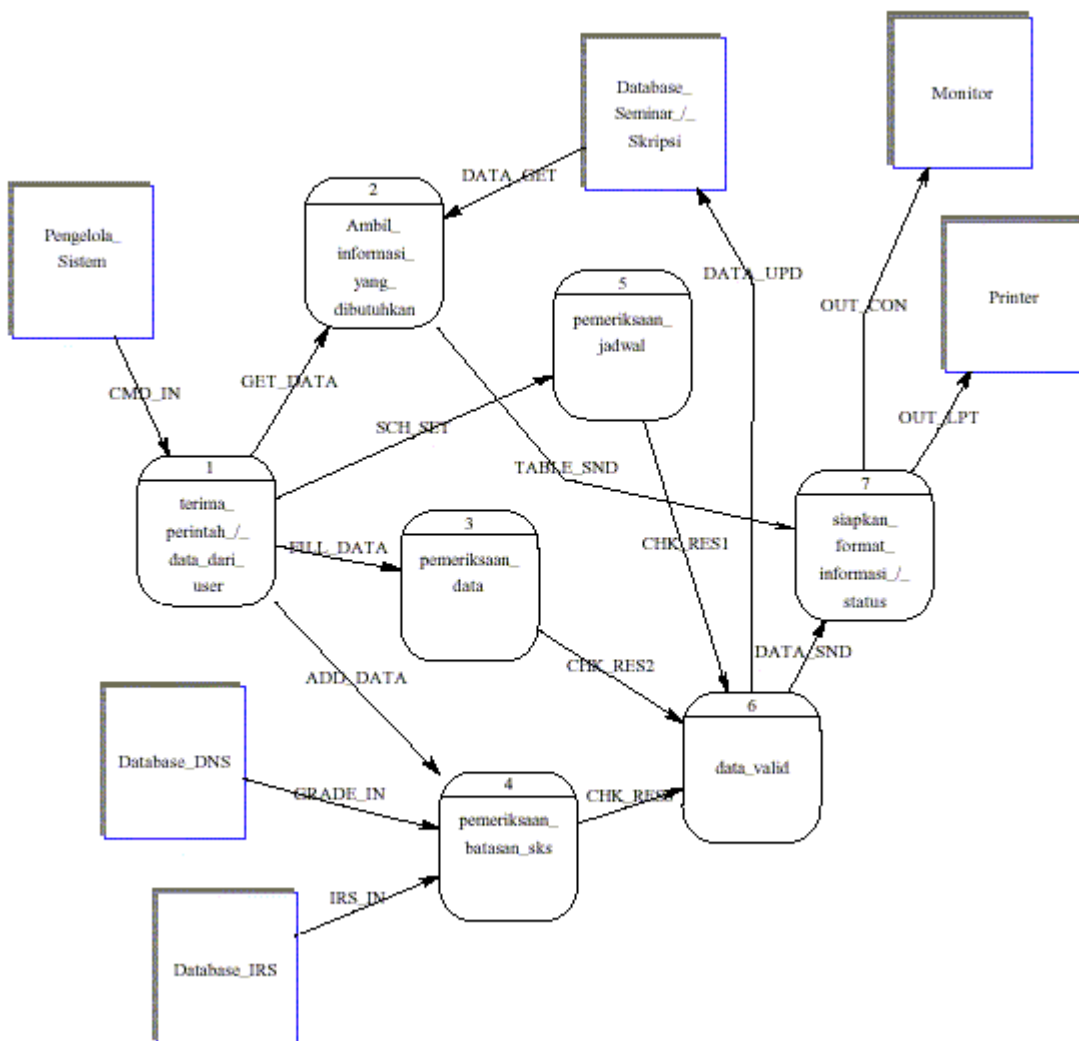
Untuk software seminar / skripsi yang dibuat, didapatkan rancangan DFD level 0 sebagai berikut :





DFD dibuat dalam beberapa tingkatan. Level utama dari DFD (level 0) menunjukkan keterhubungan antara program utama dengan konteks eksternal di luar program. Pada DFD level 0, informasi masuk dan keluar dari software dalam bentuk representasi eksternal. Pada diagram di atas, input berupa perintah dapat diterima dari keyboard dan mouse, data tambahan didapat dari database server, sedangkan output ditampilkan ke monitor atau printer.

Level DFD berikutnya merupakan pengembangan dari DFD level 0 yang memberikan deskripsi aliran data yang lebih terinci terhadap tiap-tiap entity yang terdapat pada level sebelumnya. Untuk software seminar / skripsi yang dibuat, pengembangan DFD dilakukan untuk memperlihatkan aliran data internal pada bagian software utama, sehingga didapatkan DFD level 1 sebagai berikut :





4.3. Interface Design

Desain antar muka difokuskan pada beberapa hal, antara lain :

1. Desain antar muka antara tiap modul dalam software.
2. Desain antar muka antara software dengan entity eksternal.
3. Desain antar muka antara user (manusia) dengan software.

Desain antar muka internal dalam suatu program bergantung pada bentuk data yang mengalir antar tiap modul dan karakteristik bahasa pemrograman yang digunakan untuk mengimplementasi software. Secara umum, model analisis ini akan berisikan beberapa informasi yang diperlukan untuk melakukan desain antar muka secara modular.

Data flow diagram (DFD) yang telah dibuat sebelumnya akan digunakan untuk mengetahui bagaimana tiap objek data yang mengalir dalam software ditransformasikan. Tiap proses transformasi pada DFD (yang dilambangkan dengan bulatan) akan dipetakan menjadi modul-modul di dalam struktur program. Maka tiap objek data (yang dilambangkan oleh anak panah) yang mengalir masuk dan keluar dari tiap transformasi DFD akan diikutsertakan dalam desain antar muka bagi modul yang berkaitan dengan berkaitan dengan proses transformasi tersebut.

Desain antar muka juga dilakukan secara eksternal, yang dimulai dengan melakukan evaluasi terhadap tiap entity eksternal yang terdapat pada DFD. Spesifikasi data dan kontrol yang diperlukan untuk tiap entity eksternal akan dianalisa dan dibuat desain antar mukanya. Desain antar muka eksternal ini akan bertugas melindungi software dari kesalahan pemasukan data dan kesalahan kontrol terhadap data. Oleh sebab itu, desain antar muka ini harus mempertimbangkan proses validasi data dan algoritma penanganan error antar tiap modul. Dengan demikian, software akan memiliki ketahanan yang lebih terhadap kesalahan pemasukan data yang mungkin terjadi karena kesalahan pengoperasian oleh user.

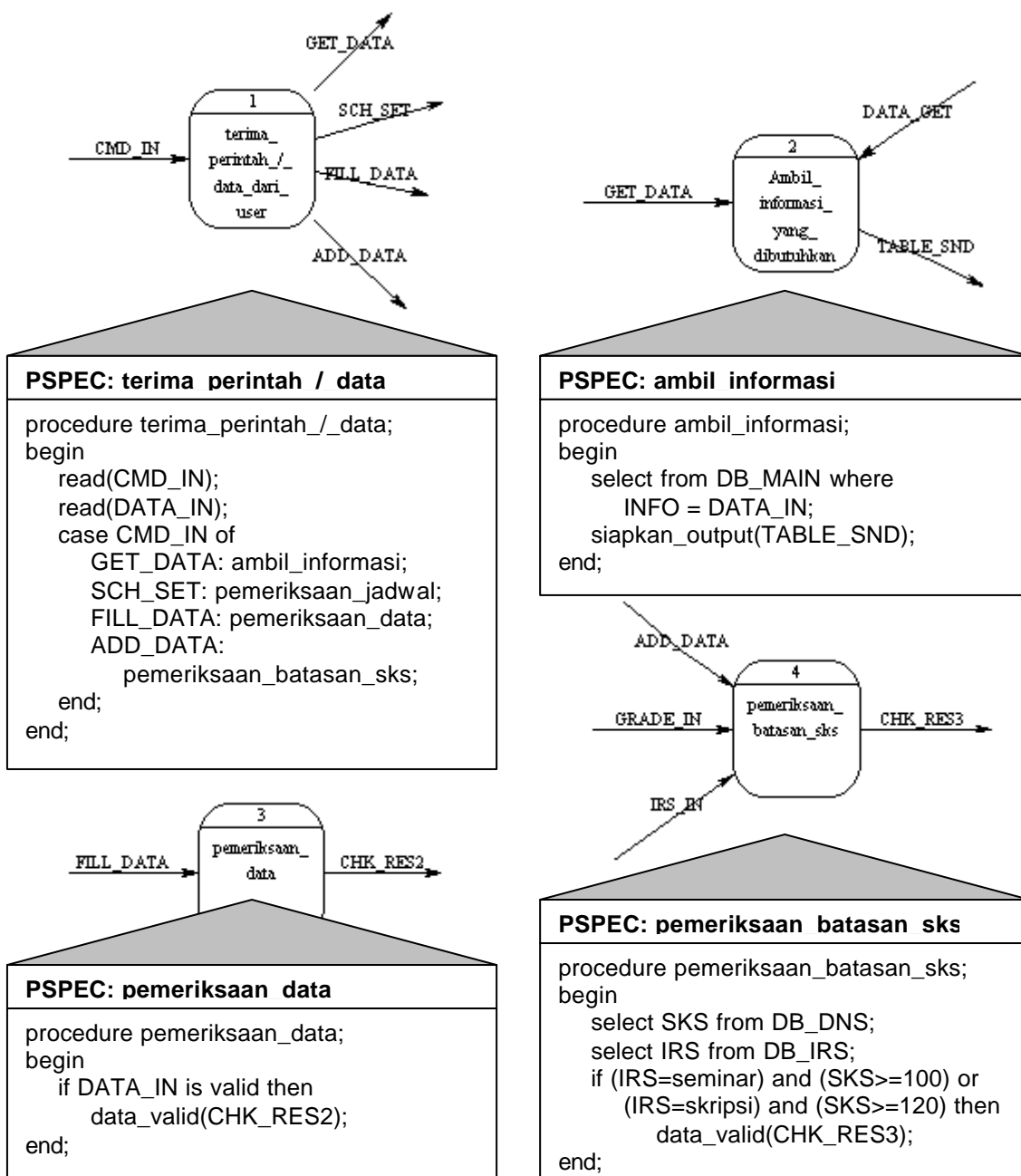
4.4. Procedural Design

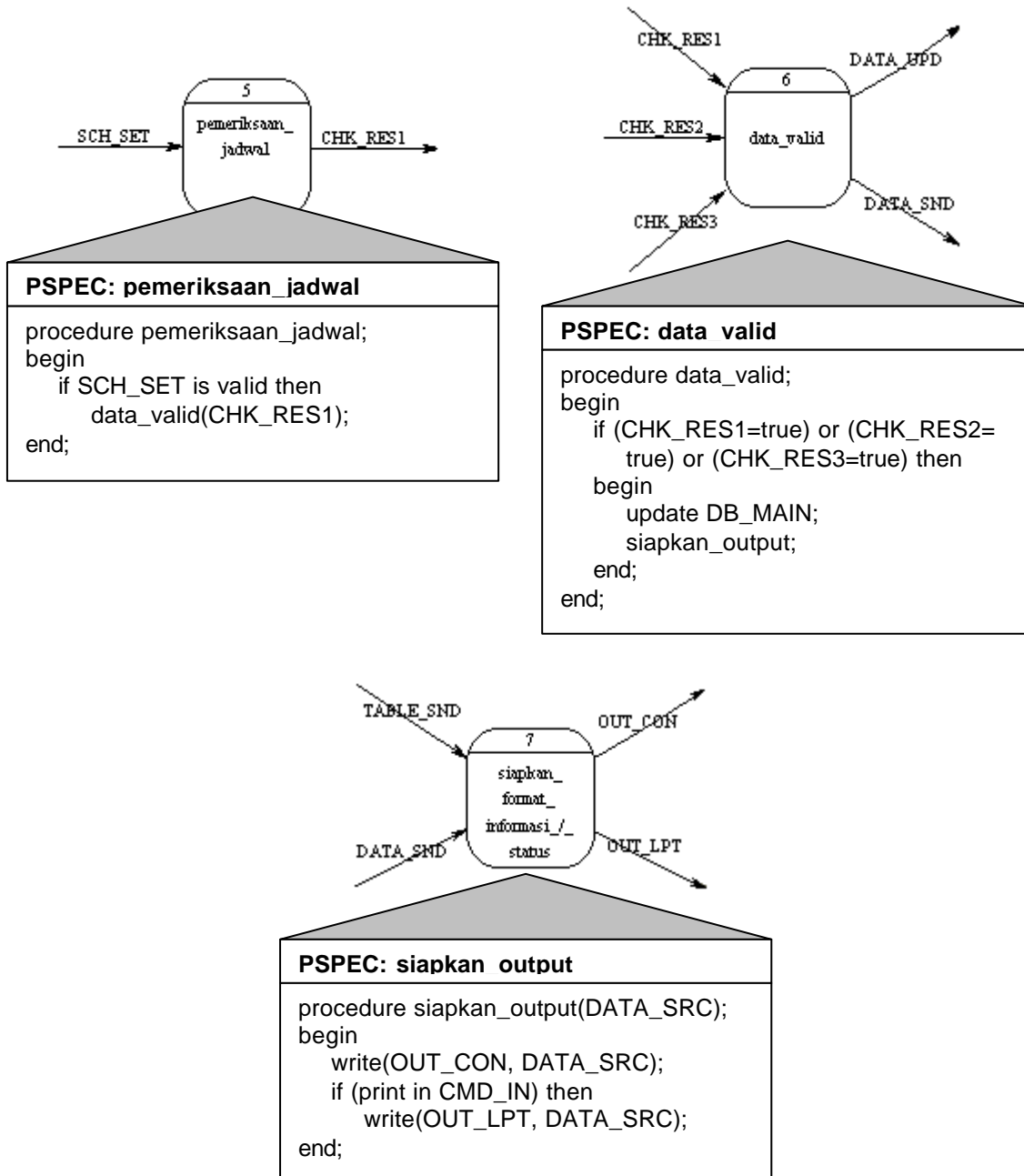
Desain prosedural dilakukan setelah diselesaikannya perancangan desain data, arsitektur, dan antar muka software. Desain ini akan berupaya mendefinisikan spesifikasi prosedural yang akan memberikan detail algoritma yang digunakan dalam implementasi program. Spesifikasi algoritma ini akan dibuat dalam bentuk notasi terstruktur berupa *sequence*, *condition*, dan *repetition*. Implementasi *sequence* akan merupakan urutan langkah-langkah



proses yang diperlukan untuk suatu operasi tertentu. *Condition* memberikan pilihan untuk melakukan proses-proses tertentu berdasarkan kondisi yang terjadi dalam program. Struktur yang dapat digunakan untuk keperluan ini ialah struktur if...then...else... atau case...of. Sedangkan *repetition* diperlukan untuk melakukan pengulangan proses (*looping*), yang dapat dilakukan dengan menggunakan struktur repeat...until, for...do, while...do, dan sebagainya.

Berikut ialah spesifikasi prosedural dari software seminar / skripsi yang dibuat untuk masing-masing proses pada DFD yang telah dirancang sebelumnya.





5. Testing Software Seminar / Skripsi

Testing dilakukan agar mengetahui apakah software tersebut berjalan sesuai dengan yang diinginkan atau tidak. Testing dilakukan meliputi seluruh aspek dari software tersebut mulai dari interface, output, input dan lain-lain. Pada software seminar / skripsi ini dalam dilakukan testing, antara lain :



5.1. Interface

a. Window

- Apakah window dapat melakukan resize, dipindahkan atau scroll ?
- Apakah item pada window dapat diklik, mempunyai function key tertentu ?
- Apakah semua fungsi yang dibutuhkan window tersedia ?
- Apakah multipel window dapat ditampilkan ?
- Bagaimana menampilkan window pada saat yang bersamaan ?
- Bagaimana menandakan suatu window yang sedang aktif ?
- Bagaimana respon window terhadap multitasking ?

b. Pull down dan operasi mouse

- Apakah bar yang ada ditampilkan dalam konteks yang tepat ?
- Apakah fungsi masing-masing pull down bekerja dengan baik ?
- Apakah aplikasi menu bar berhubungan dengan fitur lain ?
- Apakah nama dari masing-masing fungsi menu dapat menggambarkan fungsinya ?

5.2. Input

a. Pemasukan data

- Apakah pemasukan data seminar / skripsi dapat langsung diproses dan sebagai input ke sistem dan didukung oleh database ?
- Apakah mode grafik dari pemasukan data seminar skripsi berfungsi ?
- Apakah pemasukan data seminar / skripsi yang salah dapat dikenali ?
- Apakah pesan pemasukan data seminar / skripsi bersifat intelijen ?

5.3. Dokumentasi

Setiap software menyertakan dokumentasi dari software tersebut yang mencakup penggunaannya, instalasinya dan troubleshooting. Maka perlu diadakan pengujian, antara lain :

- Apakah dokumentasi menerangkan dengan jelas mengenai penggunaan software seminar / skripsi ini ?
- Apakah disertai contoh ?
- Apakah mudah dalam penggunaan dokumentasi ?



- Apakah troubleshooting yang terjadi dapat pada software seminar / skripsi dapat diatasi dengan dokumentasi ?
- Apakah error message ditampilkan jika terjadi kesalahan dalam penggunaan software seminar / skripsi ini ?

5.4. Pengujian pemulihan terhadap kegagalan software

Program seminar / skripsi harus dapat merecover dirinya dari kesalahan dan melanjutkan proses dalam waktu yang telah ditentukan. Sistem harus memiliki fault tolerant, yang menyebabkan kegagalan tidak berkibat pada seluruh proses yang dilakukan. Juga kegagalan harus dapat diperbaiki.

Pengujian dapat dilakukan dengan memaksa software seminar / skripsi ini mengalami kegagalan dan memastikan dilakukan recovery dengan tepat. Dilihat jenis recovery yang dilakukan apakah secara otomatis atau memerlukan bantuan manusia.

5.5. Pengujian security

Software seminar / skripsi ini harus dirancang dengan system security yang baik sehingga tidak semua orang dapat secara seenaknya memasuki database dan menggunakannya. Penetrasi seperti ini dapat menimbulkan kerugian baik dikalangan mahasiswa atau administrasi jurusan.

Pengujian dapat dilakukan untuk memastikan tidak ada celah yang memungkinkan orang yang tidak berhak dapat mengakses dan menggunakan software ini.

Testing dapat dilakukan dengan mengizinkan seseorang untuk mencoba menembus system security dengan cara apa saja, lalu dilihat celah yang dapat ditembus tersebut dan kemudian diperbaiki.

5.6. Pengujian unjuk kerja

Pengujian unjuk kerja dilakukan untuk menguji unjuk kerja run-time dari software seminar / skripsi dalam konteks system yang terintegrasi. Pengujian dilakukan terhadap semua unit-unit software tersebut.

Pengujian unjuk kerja sering dilakukan juga dengan pengujian stress dan membutuhkan instrumentasi software dan hardware.



5.7. Stress testing

Pengujian ini dilakukan dengan keadaan dimana permintaan sumber daya dalam kapasitas yang besar yang dapat berupa interupt dalam jumlah yang banyak, kecepatan input yang tinggi dibandingkan kemampuan proses, membutuhkan memori yang besar.

Dalam pengujian ini dapat dilakukan sekaligus beberapa materi pengujian sehingga dapat dipelajari perilaku software seminar / skripsi ini jika dipaksa untuk memproses data yang membutuhkan sumberdaya yang besar.

Semua testing yang dilakukan ini untuk menemukan error yang mungkin terjadi dalam software seminar / skripsi, mengetahui tingkah laku software dan melakukan perbaikan terhadap bug-bug yang masih ada.

6. Maintenance

Software seminar / skripsi setelah jadi membutuhkan maintenance berupa :

- Bersifat korektif
- Bersifat adaptif
- Bersifat perbaikan
- Bersifat reengineering.

6.1. Bersifat korektif

Maintenance yang dilakukan disini menitikberatkan pada perbaikan dari software seminar / skripsi ini dari bug-bug yang diporeleh selama proses pengujian. Artinya jika ditemukan bug, maka dilakukan peninjauan terhadap coding dan melakukan penyempurnaan.

6.2. Bersifat adaptif

Maintenance ini dilakukan merespon kebutuhan user, artinya memperhatikan kenyamanan user dan memperkecil kemungkinan penggunaan yang salah dari software ini. Sehingga pada akhirnya software ini menjadi suatu software seminar / skripsi yang user friendly.

6.3. Bersifat perbaikan (enhancement)

Maintenance ini dapat berupa penambahan fungsi-fungsi baru dalam software seminar / skripsi. Artinya cakupan dan fungsinya dapat mengakomodir kebutuhan mengenai seminar / skripsi secara sepenuhnya baik dari segi database.



6.4. Bersifat reengineer

Proses ini memakan waktu, sumber daya dan biaya yang banyak. Biasanya proses ini dilakukan jika pengembangan software yang telah ada mencapai titik jenuh padahal masih ada fungsi-fungsi penting lain yang harus ditambahkan, sehingga caranya adalah dengan melakukan reengineer ini.

7. Kesimpulan

Langkah-langkah yang sebaiknya dilakukan untuk membuat suatu software:

- Merumuskan masalah dari software yang dibuat
- Membuat estimasi LOC dari software yang akan dibuat
- Membuat manajemen resiko
- Membuat DFD / CFD
- Membuat *coding* software
- Mengadakan pengujian
- Mengadakan *maintenance*

Proyek untuk pembuatan software sistem informasi akademik seminar / skripsi ini lebih baik diimplementasikan dalam bentuk database. Proyek ini diperkirakan akan selesai dalam 3 – 4 bulan dengan jumlah SDM empat orang. Tools yang sebaiknya digunakan untuk pembuatan software ini antara lain:

- Untuk program dapat menggunakan Visual Basic, Delphi, Visual C++, C++ Builder, dan lain-lain.
- Untuk database dapat menggunakan MS-Access, mysql, SQL Server, dan lain-lain.

Software ini dapat dikembangkan menjadi *web-based application* dengan penambahan tabel baru pada database dan implementasi pada web server yang memiliki aplikasi *server side programming* seperti PHP, JSP, ASP, ColdFusion, dan sebagainya, serta *database connectivity* seperti ODBC dan JDBC.