

BAB IX

TAHAPAN PEMROGRAMAN

Membangun Bagian-Bagian

9.1 PENDAHULUAN

Pemrograman biasanya adalah bagian yang paling mudah-yaitulah yang kita paling kenal sebagai 'technical types'. Kenyataannya, sebagai manajer proyek Anda bisa mencari sendiri untuk mengendalikan staff anda dari memulai program yang terlalu cepat.. Selalu ada tekanan untuk 'melakukan dengan benar', tidak hanya dari tim proyek, namun dari manajemen tingkat yang lebih tinggi. Berhati-hatilah terhadap penyakit manajemen yang disebut WISCA : Why Isn't Sam Coding Anything ? (Constantine). Hal ini mengkhawatirkan manajer ketika programmer tidak mengerjakan apa-apa. Tidak pernah meng-koding sampai desainnya cukup tepat yang berarti tidak diperlukan kerja kembali.

Kegiatan dalam Tahapan ini adalah menulis program. Hal yang paling penting adalah mencoba program, Rencana Percobaan Program, dan paling tidak mulai pada dokumentasi pemakai.

Pada bab ini sepenuhnya lebih bersifat teknis, jadi para manajer yang terbatas bisa teringat ketika membaca The War Stories, Kesimpulan, dan kemudian menuju Bab 10.

Pemrograman War Stories

Cerita ini tentang sebuah koperasi ladang beras. Untuk orang-orang yang kurang mengenal usaha ladang beras, petani biasanya mengirimkan berasnya ke penggilingan beras yang terdekat. Dia kemudian mendapatkan kwitansi dan bayaran yang tergantung pada jenis beras, beratnya, kualitasnya, dan harga dasar beras pada masa mendatang. Kantor pusat dari perusahaan beras pasti menyimpan data beras pada seluruh penggilingan berasnya dibawah hak hukumnya, sehingga kapan sejumlah besar beras dibawa truk dan kereta yang mengambil beras dikirimkan ke lokasi yang benar.

Beberapa tahun yang lalu perusahaan beras ini, berkantor pusat di kota 'A', memutuskan untuk mengadakan komputerasi dalam kegiatannya. Mereka kemudian menggunakan hardware dan software jaringan FMMC (Famous Minicomputer

Manufacturing Co.), sehingga mengirimkan programmernya ke Pusat Latihan FMCC, yang berlokasi di kota 'B', untuk belajar software jaringan.

Perusahaan beras ini terkesan dengan instruktur kursus FMCC yang kemudian mereka sewa sebagai konsultan untuk mendesain sistem mereka. Desainnya sangat menarik. Sang Instruktur/Desainer menyarankan agar PC ditempatkan pada masing-masing penggilingan (diperlukan sedikit kemampuan komputer) untuk menangani kwitansi dan transaksi petani. Sebuah minicomputer pada kantor pusat menyimpan transaksi dan stok beras dan secara otomatis menghubungi masing-masing PC pada tengah malam untuk mendapatkan transaksi. Instruktur memerlukan waktu tiga minggu untuk merancang semuanya dan dia kembali pulang. Tetapi perusahaan beras sangat terkesan dengan rancangan dari instruktur yang mereka sewa sebagai konsultan untuk membuat program dengan baik. Pemrogramannya mengambil tempat di kota 'C', sejak itu menjadi rumah bagi kebanyakan programmer. Gambaran pada titik ini adalah; instruktur/programmer berada di kota 'C'; Pimpinan Proyek (Manajer Instruktur) berada di kota 'B'; dan Manajer Proyek, yang merupakan perwakilan dagang dari FMCC, berada di kantor pusat perusahaan beras di kota 'A'. Hal ini disebut 'Manajemen Proyek Terdistribusi'. Instruktur memulai program. Meskipun dijadwalkan satu modul perminggu, modul pertama tidak selesai dalam delapan minggu. Akhirnya PL menerima modul pertama – dikirimkan secara elektronik ke kota 'B'- dan panik. Programmernya sangat mengejutkan: tidak terstruktur dan penuh dengan kesalahan. Dia kemudian menelepon instruktur, berikut adalah percakapan mereka :

PL : Mengapa terlalu lama ? dan mengapa programnya sangat buruk ?

PGR : John (bukan nama sebenarnya), saya tidak memprogram lagi selama enam tahun. Saya pikir akan sangat menyenangkan untuk kembali ke kegiatan itu. Tapi saya tidak menyukainya, dan saya benci kota 'C'. Biarkan saya keluar dari sini !

Komentar : Instruktur bahkan tidak mengomelinya. Dia adalah instruktur dan desainer yang baik, tetapi menyulitkan programmer. Hal ini biasa. Masalahnya adalah PL tidak bisa mengawasi. Selama mereka bersama, PL tidak memberitahukan masalah satu kalipun dan segera mengganti programmer. Untuk pegawai baru atau yang belum diketahui, awasi pekerjaan mereka pada minggu-minggu awal. Terutama untuk kontraktor dari luar.

Epilog : Instruktur telah pulang kembali, programmer lainnya pindah ke kota 'C' untuk menggantikannya, dan proyek ini berjalan dengan baik, meskipun terlambat sepuluh minggu.

Sekitar tiga bulan kemudian, FMCC memberitahukan bahwa komputer pengontrol beras mereka yang baru telah dapat dinikmati dan mendapatkan popularitas yang cukup besar. Setelah melakukan penelitian kecil, FMCC menemukan bahwa komputer yang tidak dipakai selama 905 dari waktu pemakaiannya, secara mengejutkan meningkatkan penggunaan game dan bahasa compiler seperti BASIC. Kursus-kursu komputer informal berkembang pesat. Para petani – terutama anak-anak mereka – sangat menikmati sistem yang baru ini.

9.2 DAFTAR PERTANYAAN SEBELUM PEMROGRAMAN

Sebelum Anda memulai pemrograman, jawablah pertanyaan berikut :

- Apakah review perancangan memerlukan pengerjaan kembali ? Jika ya, jadwalkan waktu mulai dan waktu tunggu dari pemrograman.
- Apakah sumberdaya yang direncanakan dan para programmer masih ada ? Jangan terlalu yakin dengan proyek orang lain yang diselesaikan tepat waktu. Jika ada staf yang telah diganti, apakah anda akan menguji kembali produktifitas mereka ? Statistik Industri telah menunjukkan bahwa programmer terbaik dapat lebih produktif delapan kali lipat dari yang terburuk.
- Apakah orang-orang ini telah dilatih ? Programmer harus mengetahui tentang sistem operasi, bahasa pemrograman, paket-paket program, dan alat-alat pemrograman yang digunakan. Mereka juga harus mengenal dengan baik aplikasi user dan masalah-masalah bisnis. Pastikan bahwa mereka telah membaca Requirements Documentation and Functional Specifications.
- Apakah lingkungan pemrograman cukup baik ? Anda memerlukan easy-to-use software pengembangan dan alat-alat pemrograman. (lihat Bagian 9.4). Komputer pengembangan harus mempunyai respon yang cepat, harus tersedia jika diperlukan, dan harus reliabel. Pastikan bahwa tersedia garansi dari penjual, seperti dokumentasi software pengembangan yang up-to-date. Sediakan tempat yang tenang, jauh dari gangguan.

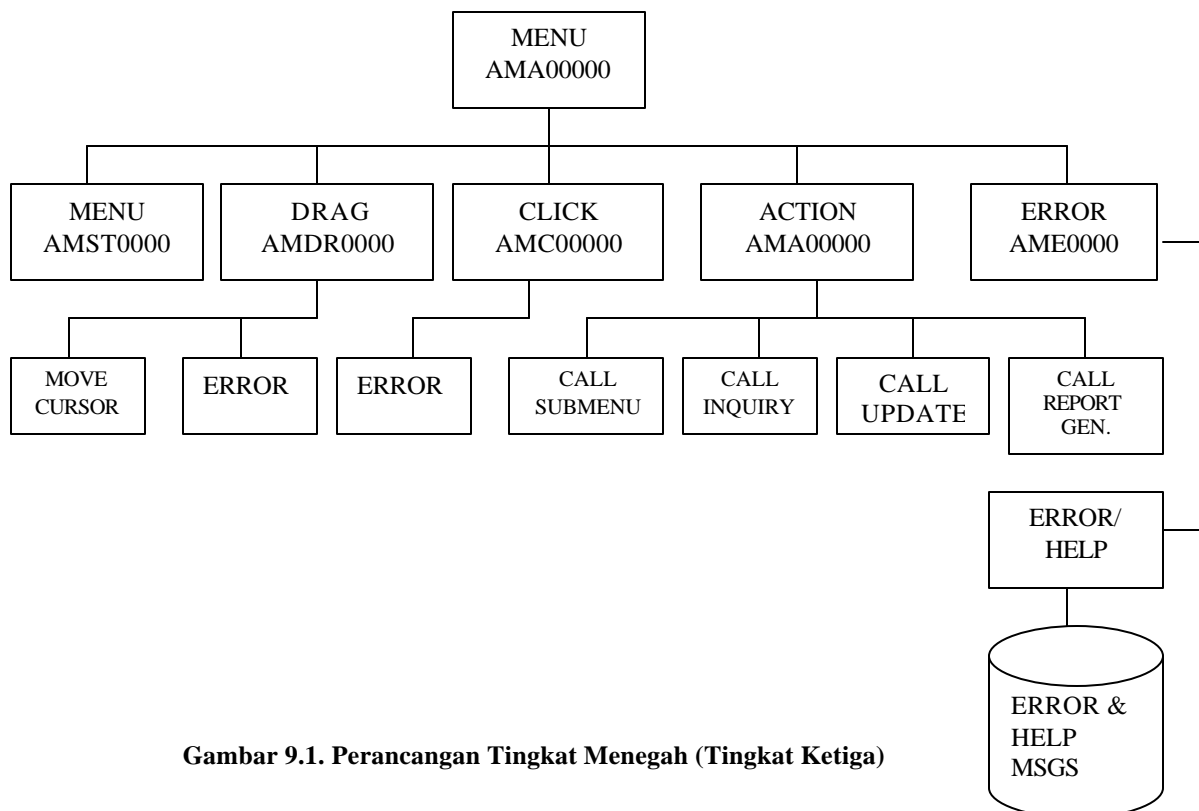
9.3 LANGKAH-LANGKAH PEMROGRAMAN

Langkah 1. Rencanakan Penggabungan

Kumpulkan catatan biasa yang tidak bisa anda program semuanya – diperlukan perangkaian langkah demi langkah. Rencanakan susunannya dimana anda akan menggabungkannya. Bab 9 memerinci beberapa metode untuk merangkai bagian-bagian, tetapi anda harus merencanakan urutan penggabungan ini sekarang, sejak *Anda tidak diharuskan untuk menuliskan program supaya mereka dapat digabungkan*. Ini disebut Rencana Percobaan Sistem.

Langkah 2. Perancangan Modul

Programer menerima *beberapa* tingkatan perancangan dari Tahapan Perancangan. Pekerjaannya adalah memecah modul ke detail-detail yang lebih rendah sampai mencapai keadaan programer siap untuk melakukan pemrograman. Ini disebut *Perancangan Modul*. Level Perancangan modul tingkat menengah seperti Gambar 9. 1. Berikut, telah dikembangkan dalam Tahapan Perancangan.

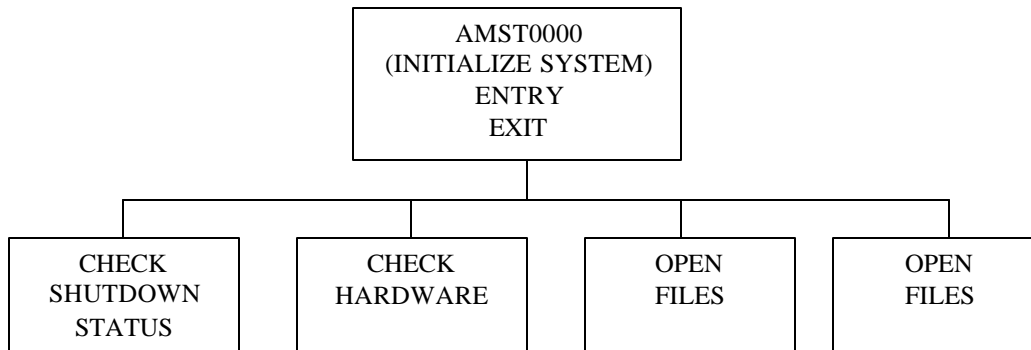


Gambar 9.1. Perancangan Tingkat Menengah (Tingkat Ketiga)

Programer menerima deskripsi modul dari desainer seperti berikut : (lihat appendix A)

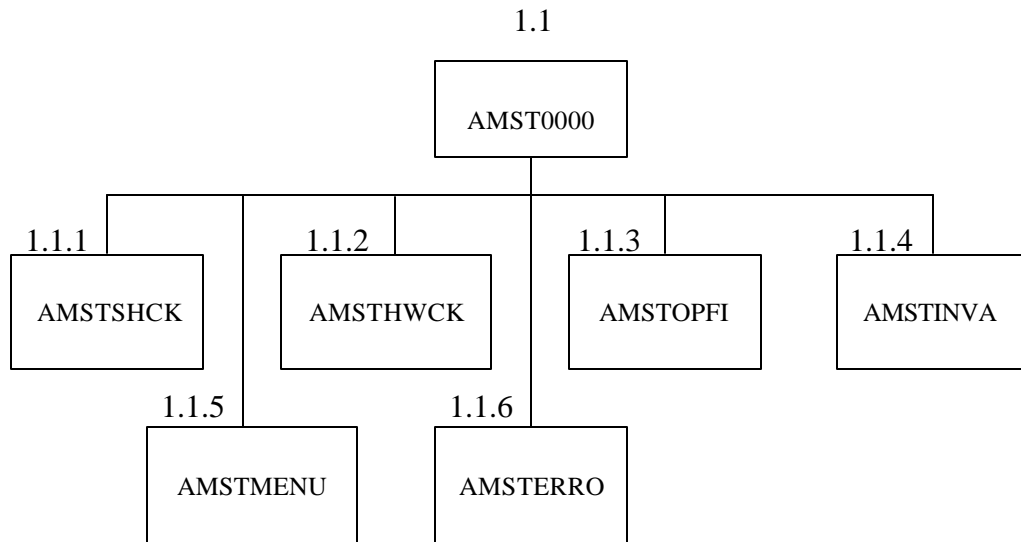
```
Module Name:  AMST0000
Called by:  AM000000
Subroutine called:  to be filled in by programmer
Input parameter:  none
Displays:  none
Returned parameter:  if no error exit code 0.  If error, exit code is error
number
External variabel used:  (list)
Files used:  STUDENT.DAT (open), COURSE.DAT (open), MATERIAL.DAT (open),
SYSTEM.DAT (open)
Functions:  open the files STUDENT.DAT, COURSE.DAT, MATERIAL.DAT,
SYSTEM.DAT.  If error, exit with code...  initialize variabel...
Check for abnormal shutdown by checking Record 1 of SYSTEM.DAT file.
Byte 1 = -1 means proper shutdown (see module AMSHUT00).  If not -1 , do
following...  On error exit with error code...
Ensure correct status of
  Mouse by checking...
    On error exit with error code...
  Screen by...
    On error exit with error code...
  Network by...
    On error exit with error code...
Normal exit error code 0
```

Programer pertama-tama menggambar diagram struktur dari modul. Terlihat seperti gambar 9.2.



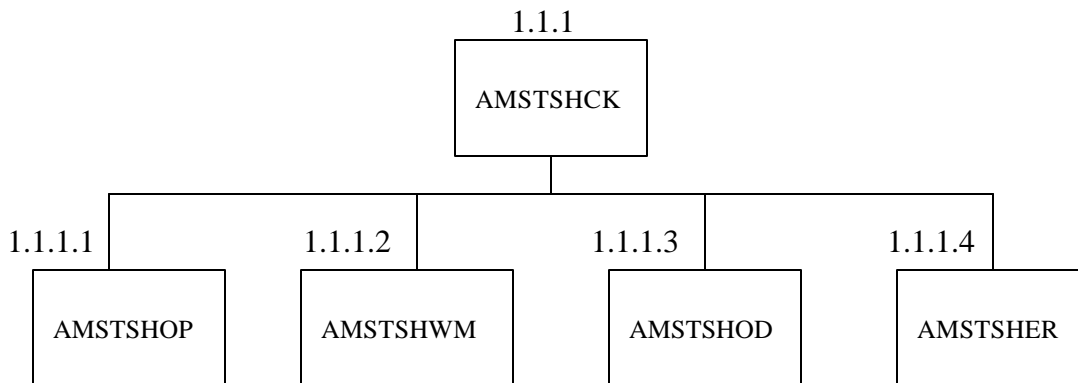
Gambar 9.2 Pemecahan Modul Tingkat Keempat

Perancangan Modul adalah pendekatan atas-bawah dimulai dengan kotak yang paling atas, AMST000 dan dipecah ke dalam sub-komponen yang tepat seperti gambar 9.3.



Gambar 9. 3. Pemecahan Modul Tingkat Kelima.

Kemudian masing-masing komponen dapat dibagi lagi seperti pada gambar 9.4.



Gambar 9. 4. Pemecahan Modul Tingkat Keenam

Dan kemudian modul tersebut dipecah-pecah lagi sampai tercapai sebuah tingkatan dimana mulai bisa diprogram.

Pertanyaan yang biasa diajukan adalah : “Pada tingkatan mana perancangan sistem berhenti dan perancangan modul dimulai ?”. Jawabannya adalah “Perancangan Sistem diturunkan di tingkatan yang mana programmer dapat memulainya” tingkatan ini dapat bermacam-macam dari proyek ke proyek dan bahkan dari satu bagian sistem ke bagian lainnya – tergantung pada programmer yang menerima bagian tersebut. Ada banyak pertimbangan :

- Jika modul yang dihasilkan adalah sangat memerlukan prioritas seperti adanya respon, tidak user-friendly atau tidak konsisten, perancang bisa pindah ke tingkat lebih rendah.
- Tingkatan hasil dari perancangan mungkin dinyatakan dengan kontrak. Departemen Pemerintah seperti Militer memberikan spesifikasi jumlah dari tingkatan tersebut.
- Jika programmer tidak mengetahui pada waktu perancangan, dapat diasumsikan pada sebuah tingkatan menengah dari pengetahuan programmer, dan perancangan dapat diambil alih oleh tingkatan programmer yang dapat mengatasinya.

Tetapi perlu diingat bahwa programmer tidak menyukai menerima perancangan terlalu terperinci seperti pemrograman itu sama mudahnya dengan menterjemahkan bahasa Inggris – seperti pernyataan kata demi kata ke dalam bahasa pemrograman.

Langkah 3. Telusuri Perancangan Modul

Seperti pada tingkat atas dan menengah dari perancangan, pertukaran harus dibuat pada tingkatan yang paling rendah. Telusuri perancangan masing-masing modul sebelum melakukan peng-kode-an. Penelusurannya sangat kecil: hanya programmer yang tepat, pengawasnya dan mungkin programmer lainnya yang perlu diperhatikan. Kegunaan dari penelusuran Perancangan Modul adalah untuk memastikan bahwa perancangan yang terbaik yang telah dilakukan., semua fungsi telah dialamatkan dan semua bagian telah ditangani.

Langkah 4. Perencanaan Bagaimana Menguji Sebuah Modul

Programmer harus menyiapkan rencana pengujian dan data pengujian untuk modul sebelum modul tersebut di-koding. Rencana pengujian dilakukan setelah kode ditebak. Mereka cenderung hanya menguji bagian kode yang paling 'sukar'. Pimpinan Proyek bisa saja melakukan tuntutan pada penelusuran rencana pengujian sepanjang perancangan modul sedang dilaksanakan.

Langkah 5. Meng-kode Masing-Masing Modul

Pengkodean secara standar akan disesuaikan pada saat perancangan sistem (lihat Bagian 7.12). Kita tidak membahas bagaimana pemrogramannya – lihat Referensi 12

(Tulisan ini membahas Perancangan sama dengan pemrograman) dan Referensi 13 untuk lebih jelasnya. Berikut ini adalah ringkasan dari sebuah program terstruktur :

- Program kecil. Aturan dasarnya adalah kira-kira 100 baris kode yang dapat dieksekusi, listingnya paling tidak dua halaman.
- Satu masuk, satu keluar.
- Referensi umum yang sedikit.
- Konstruksi struktur yang digunakan : Sequens, IF/THEN/ELSE, CASE, WHILE, UNTIL, CALL (bukan GO TO).

Langkah 6. Menguji Modul

Programer menguji modul dengan mengatur lingkungan yang tepat, menyediakan beberapa input, membiarkan modul langsung memproses secara logis dan mendapatkan hasilnya. Beberapa input mungkin harus dimanipulasi, terutama jika modul tersebut tidak menyediakan inputnya.

Modul seharusnya diuji dalam dua tahap. Tahap pertama disebut pengujian ‘White Box’. Programer harus mengetahui isi di dalam modul, dan menyediakan data pengujian sehingga masing-masing path logikal dalam program dapat dieksekusi. Sesudah itu, Tahap Kedua, atau pengujian “Black Box” dapat dilakukan. Dalam pengujian Black Box programer mengabaikan bagian dalam dari modul, data disediakan secara berurut dan dianggap seperti pemakaian sebenarnya.

Langkah 7. Menguji Tingkatan Paling Rendah dari Penggabungan

Jika sebuah modul memanggil sub-modul, programer harus menggabungkan dan menguji semua modul secara bersama-sama. Bahkan jika dia tidak bertanggungjawab untuk menulis sub-modul, dia harus menguji perintah CALL dan RETURN dari seluruh modul. Metode terbaik untuk melakukan hal ini adalah membuat sebuah “Program Stub” untuk menggantikan sub-modul. Potongan program ini bisa saja empat baris program yang mengindikasikan bahwa kontrol sudah diterima dengan baik, tampilkan parameter penerima, jika perlu lakukan pegontrolan kembali dengan beberapa parameter yang dimanipulasi.

Langkah 8. Menyimpan Hasil Seluruh Pengujian; Menyerahkan Modul Akhir

Hasil pengujian digunakan untuk mendapatkan statistik penyebab, inti dan biaya dari perbaikan error. Pimpinan Proyek biasanya mengenakan biaya dari penggabungan ini dalam sistem ukuran kecil sampai sedang. Software seperti CMS(Code Management System) adalah sangat berguna untuk manajemen penggabungan – menyimpan versinya dan menggantinya dalam sumber kode Lihat Bagian 9.4).

Langkah 9. Memulai User Documentation

Apakah programmer bertanggungjawab untuk user documentation atau tidak, tahapan ini adalah waktu terbaik untuk menjaawabnya. Dokumen berikut mungkin harus ditulis :

User's Guide : Dokumen ini dapat ditulis oleh programmer, penulis teknis atau bahkan pemakai sendiri. Tampilkan kembali FS yang mempunyai bagian rinci mengenai menu, layar, form dan user interface lainnya.

USER'S GUIDE yang baik adalah terbagi dalam bagian-bagian yang menunjukkan tingkatan pemakai yang berbeda-beda. Dalam USER'S GUIDE sistem ABC, sebagai contoh, harus ada bagian yang disebut 'Registrar's Functions', atau "Warehouse Functions' atau lainnya. Materinya harus disesuaikan agar pemakai dapat menggunakan secara normal. Hali ini bisa membuat USER'S GUIDE berguna untuk mempelajari sistem. Urutan populer lainnya untuk USER'S GUIDE adalah dengan menelusuri menumenu perintah secara logika. Pada akhir dari USER'S GUIDE ini disediakan referensi dari masing-masing perintah, menu, form dan pesan yang ditampilkan secara alfabetis.

Maintenance Guide : Bagaimana Anda menemukan programmer di dalam dokumen yang memerinci program mereka untuk perbaikan-perbaikan berikutnya ?. Kebanyakan PM megalami kesulitan dalam hal berikut : programmer enggan untuk melakukan dokumentasi sebelum program ditulis; dan beruntunglah menemukannya setelah semuanya selesai dikerjakan. Programmer berpikir bahwa perbaikan memerlukan penjelasan secara terperinci dari logika pemrograman. Sangat membosankan untuk emnulisnya dan tidak perlu sebenarnya. Berikut adalah penyelesaian yang cukup mudah : *Lebih baik, memerinci spesifikasi perancangan tahap modul dengan struktur, mendokumentasikan sendiri kode dirasa cukup untuk perbaikan sistem.* MAINTENANCE GUIDE akan berisi Spesifikasi

Perancangan, listing program, dan penjelasan bagaimana semuanya disesuaikan, bagaimana mengganti pendekatan, dan bagaimana menghubungkan dan menguji semuanya.

Operator's Guide/System Manager's Guide : Sama seperti USER'S GUIDE bagi orang-orang yang menhidupkan sistem di pagi hari, mematikannya, melakukan back-up, menangani permasalahan, melakukan perhitungan dan lain sebagainya. Dokumentasi yang disediakan oleh pembuat hardware dan sistem operasi mungkin cukup – hanya prosedur untuk software tertentu yang harus ditulis ulang.

Training Documentation : Jika Anda membuka kursus bagaimana menggunakan sistem, rencanakan apakah bahan dari training akan diperlukan. USER'S GUIDE yang baik harusnya menambahkan hal ini. Anda mungkin harus membuat bantuan pelatihan seperti; transparansi, buku latihan, pengujian dan lain sebagainya.

9.4. ALAT-ALAT PEMROGRAMAN CASE

Berikut ini adalah produk software yang membantu programmer untuk melakukan pekerjaannya dengan lebih baik. Software ini disebut CASE (Computer Aided Software Engineering) karena membantu proses pemrograman secara otomatis. Lihat Referensi 2.1 untuk produk tersebut.

Bahasa Pemrograman

Bahasa pemrograman dan kompiler adalah alat yang sangat penting. Jika sesuai dengan aplikasinya, programmer akan dapat mempelajarinya dengan cepat, gunakan jenis yang diperlukan secara tepat, dan lakukan pemrograman tanpa canggung. Kompiler harus cepat dan tidak ada error.

Language Sensitive Editor (LSE)

LSE menyediakan template untuk masing-masing pernyataan dalam bahasa pemrograman. Sebagai contoh, dalam bahasa PASCAL, user dapat mengetikkan "FOR" dan LSE menghasilkan :

```
FOR  %{ctrl-var}% := %{exp}% %{TO | DOWNTO}% %{exp}% DO
      %{statement}%
END;
```

Programer mengisikan variabelnya dan LSE memastikan sintaksnya benar. LSE juga dapat memanggil kompilator. Jika ada error yang ditemukan oleh kompilator, LSE dapat mengontrol kembali, dan programer dapat berada dalam modus edit – pada pesan dan baris error tersebut. LSE dapat membuat program header dari template.

LSE membantu dalam pemeriksaan sintaks, kompilasi program dan memastikan kekompatibelan format asalnya terhadap sistem

Debugger

Debugger membantu memeriksa dan memperbaiki error. Debugger dapat memberhentikan program, menelusuri kesalahan, dan memeriksa error berikutnya. Debugger yang baik dapat menyesuaikan dan menampilkan variabel pada semua titik, seperti pada peng-eksekusian bagian spesifik dari program.

Code Management System (CMS)

Seringkali disebut manajer konfigurasi, CMS tidak tersedia untuk semua bahasa pemrograman. CMS adalah ‘perpustakaan’ yang memiliki sendiri seluruh sumbernya. CMS dapat menertibkan orang-orang yang melakukan update dan memastikan tidak terjadi konflik jika dua orang meng-update modul yang sama pada saat yang bersamaan. CMS menyimpan semua catatan ke semua modul sehingga history dari suatu modul dapat mudah dilihat. Dan sebagai tambahan, CMS menyediakan regresi yang mudah untuk versi sebelumnya.

CMS dapat menangani semua file ASCII. Sehingga berguna tidak hanya untuk menelusuri file sumber, tapi juga menyimpan file dokumen, file pengujian, dan file-file yang membangun sistem. Bayangkan keadaan pada pabrik hardware/software seperti DEC, dimana biasanya terdapat 20 sampai 30 versi sistem operasi yang digunakan, masing-masing jenis tergantung pada beratus-ratus permutasi dan kombinasi hardware dan software yang beragam. CMS tentunya diperlukan untuk menanganinya. (maafkan saya karena terlalu antusias, tapi CMS telah menyelamatkan saya berkali-kali).

Module Management System (MMS)

MMS digunakan untuk proses kompilasi dan link secara otomatis, atau membangun sebuah sistem. MMS hanya dapat membangun kembali semua komponen tersebut yang dirubah sejak pembangunan yang terakhir. MMS dapat digunakan untuk menjalankan secara otomatis sekumpulan pengujian modul. MMS sangat berguna ketika Anda membangun sebuah ‘release’ sistem : menyatukan sumber-sumber yang benardan meng-eksekusi image, seperti seluruh dokumen yang terdapat dalam satu paket. MMS bekerja hand-in-hand dengan

CMS dimana semua sumber, file dokumen dan file perintah yang berjalan di MMS dapat disimpan.

Test Manager (TM)

TM digunakan untuk menguji sebuah modul secara otomatis. Untuk menggunakan TM, Anda harus mendefinisikan serangkaian pengujian terhadap modul. TM akan menjalankan pengujian, dan memberitahukan programmer jika hasilnya berbeda dengan yang diharapkan.

9.5 HAK CIPTA

Subyek dari hak cipta software adalah tetap pada pengadilan, tetapi terdapat peraturan pemerintah yang tidak hanya merupakan bagian dari software yang memiliki hak cipta, tetapi juga 'look' dan 'feel' dari peraturan tersebut (apapun pengertiannya). Jika anda ingin melindungi kode anda, tambahkan pemberitahuan hak cipta pada masing-masing modul dan dokumen. "Copyright © 19nn, Company Name" yang biasanya diperlukan.

9.6 KESIMPULAN PADA TAHAPAN PEMROGRAMAN

Berikut ini beberapa pemikiran dalam pemrograman :

Pemrograman digunakan sebagai pertimbangan sebuah seni. Programmer boleh untuk 'melakukan hal-hal mereka sendiri'. Dapat dengan cepat diketahui bahwa hal tersebut sangat mahal. Haruslah dipertimbangkan untuk menjadi sebuah ilmu - dicatat dengan teliti.

Pemrograman adalah kesenangan - tetapi debugging bukanlah kesenangan. Perhatikan pernyataan seperti 'Coding is done; all that's left is to debug it, so I am 90% done!' Statistik menunjukkan bahwa hanya yang programmer kerjakan setelah peng-kodean.

Berikut beberapa pemikiran programmer :

Programmer selalu meremehkan tugas. Mengajari mereka pesimis - merupakan hal yang salah.

Programmer akan menikmati pekerjaan mereka jika Anda memotivasi mereka dengan sebuah tantangan. Masing-masing tugas harusnya lebih sulit atau berbeda dari sebelumnya. Jika Anda ingin belajar bagaimana memotivasi programmer; bacalah buku G. Weinberg, "The Psychology of Computer Programming" (Referensi 14).

Programer sangat mudah tertekan - mereka akan bekerja lembur jika diperlukan. Tetapi hati-hati terhadap lembur yang tetap. Sewaktu-waktu tidak ada lagi produktifitas extra yang diperoleh dan programer akan habis/punah.

Pada akhir tahapan pemrograman, lihatlah hal yang utama berikut ini :

1. Perancangan modul sudah dijalankan dan diselesaikan.
2. Program-program yang tersendiri sudah selesai di-kodekan, diuji dan diselesaikan oleh pimpinan proyek.
3. Susunan dari penggabungan telah ditentukan, ditulis dalam Rencana Pengujian Sistem (dan pemrograman telah dijalankan pada saat itu).
4. Tanggungjawab terhadap user documentation telah diberikan, dan jika Anda beruntung hal tersebut telah dilakukan!.

PERTANYAAN

1. Mengapa programer selalu memulai peng-kodean terlalu buru-buru (sebelum perancangan selesai) ? Mengapa hal ini dianggap tidak bijaksana ?
2. Kapan perancangan sistem diberhentikan dan perancangan modul dimulai ? Faktor apa yang mempengaruhinya ?
3. Mengapa programer harus merencanakan pengujian modul sebelum menuliskan kode ?
4. Apa yang dimaksud pengujian "White Box" dan "Black Box" ?
5. Buat daftar lima ciri-ciri Program terstruktur!.
6. Sebutkan dua hal, manfaat dari User's Guide yang baik ? Mengapa dua-duanya harus disediakan ?
7. Apa yang dimaksud dokumen perbaikan program secara tradisional, dan mengapa hal ini tidak disukai ? Apa penggantinya ?
8. Buatlah daftar, berdasarkan kepentingannya, enam alat-alat pemrograman CASE. Jelaskan mengapa Anda memilih tiga urutan pertama yang lebih penting.
9. Apakah hal penting dari Tahapan Pemrograman ?

