PERANCANGAN PERANGKAT LUNAK

(reference Burch, chp. 15 – 20)

A. SW development
B. SW designing
C. SW coding (tidak perlu)
D. SW testing
E. SW implementation

A. SW Development

SW development dibedakan 2 :
1. Spreadsheet for every application (ready-made)
2. Reinventing the wheel (develop from scratch)

Sumber2 aplikasi SW :
- ü SW komersial dari vendor
- ü SW yang dibuat sendiri (dicustom) atau kontraktor pemrograman bebas.

SW harus dibuat sesuai dengan spesifikasi perancangan yang ada di Detailed Systems Design Report yang disusun dari permintaan user. SDLC cocok untuk membangun sistem, meskipun SW tsb. merupakan software jadi atau dibuat sendiri.

Paket SW Komersial
Kebanyakan diperuntukkan untuk kalangan bisnis. Contoh : spreadsheet, DBMS. Paket lain merupakan aplikasi khusus. Seperti GL, word processing. Paket ini merupakan fungsi bisnis dasar yang hampir tidak berbeda banyak dalam organisasi.

Keuntungan	Kekurangan
ü Rapid implementation ü Cost saving ü Time and cost estimation ü Reliability	ü Poor systems design match ü Vendor dependent ü Indirect cost from crashing SDLC

Dalam menyiapkan proposal untuk SW komersial langkah yang diambil adalah dalam pemilihan vendor yang tepat dan paket SW komersial perlu menyiapkan RFP (request for proposal) berorientasi kinerja. Faktor evaluasi yang utama meliputi spesifikasi desain yang rinci (O, I, P, Db, C, cost & time).

Menilai Paket
Setiap paket perlu dinilai, dapat dilihat dari benchmark. Sejumlah publikasi tentang penilaian adalah :
- ü Operating performance

- ü Documentation
- ü Ease of learning
- ü Ease of use
- ü Control & error handling
- ü Support

Memilih Paket
Yang dilihat adalah paket SW dari vendor yang mana yang memberikan keuntungan yang besar dengan cost yang kecil. Untuk tujuan ini, total penilaian dan total biaya perlu diperhitungkan.
Contoh :

General performance factors	Weight	Vendor A		Vendor B	
		Rating	Score	Rating	Score
Vendor assesment	10	6	60	8	80
Operating performance	20	7	140	8	160
Documentation	10	8	80	9	90
Ease of learning	20	7	140	6	120
Ease of use	10	5	50	6	60
Controls & error handling	20	4	80	6	120
support	10	7	70	8	80
T o t a l	100		620		710

Bobot untuk setiap faktor kinerja diberikan berdasarkan kepentingannya, jumlah total 100. Penilaian diberikan (1 = poor; 10 = excellent). Bobot dikalikan dengan penilaian dan didapatkan score. Hasilnya dijumlah untuk melihat nilai setiap vendor.

	Total cost	Total rating point	Cost per rating point
Vendor A	$ 22,700	620	$ 37
Vendor B	$ 27,690	710	$ 39

Berdasarkan cost kedua vendor ini, mana yang dipilih ?
Meskipun paket vendor A memiliki nilai rendah, biaya per nilainya $ 37, lebih baik untuk cost/ benefit dibandingkan vendor B.

Program SW Pesanan (Customized SW)
Program SW pesanan dibangun oleh orang yang bekerja baik di perusahaan maupun di kontraktor.

Software Development Life Cycle (SWDLC)
Membangun program terdiri dari 3 fase (SWDLC), yaitu :
- ü Design
- ü Code
- ü Test

Beberapa alasan SWDLC sebagai komponen dari SDLC :

- ü SDLC mencakup pembangunan dari sistem keseluruhan yang memerlukan komponen lain selain SW
- ü Dalam sistem yang membutuhkan pembangunan SW berdasarkan perancangan sistem dari SDLC, SWDLC diinisialisasi
- ü Saat SWDLC dibuat, perlu fase
- ü SWDLC menyediakan prosedur operasi standar dan frame kerja yang mendukung pendekatan teknik dari pembangunan SW.

Dalam fase SWDLC distribusi yang direkomendasikan adalah aturan 40-20-40. Rule ini menekankan pada perancangan front-end dan testing back-end, dimana pengkodean tidak terlalu ditekankan. Aturan ini hanya sebagai guideline. Jika perancangan bebas kesalahan, pemrograman dan testing dapat berjalan dengan sedikit kesulitan. Semakin kritis dan besar program, semakin ada upaya yang diberikan untuk fase perancangan dan testing.

Design

Bagian dari rancang sistem yang rinci yang dikonversikan ke program aplikasi, yang dirancang pada level yang dapat digunakan pemrogram untuk menuliskan kode. Untuk setiap kejadian, tool utama dari perancangan design adalah :

- ü Structured chart
- ü Structured english
- ü Decision tables
- ü Decision trees
- ü Equations
- ü Data dictionaries
- ü Warnier-Orr diagrams
- ü Jackson diagrams

Code

Fase koding (penulisan perintah2 ke bahasa pemrograman) yang akan dilaksanakan oleh pemrogram dan tidak otomatis digenerate oleh paket CASE, memetakan design ke prosedur program. Untuk menulis kode, pemrogram harus memiliki keahlian dalam menggunakan bahasa pemrograman.

Test

Semua modul koding, terpisah atau bersama, dites untuk melihat dan menghapus kesalahan2. Selama testing, program dioperasikan.

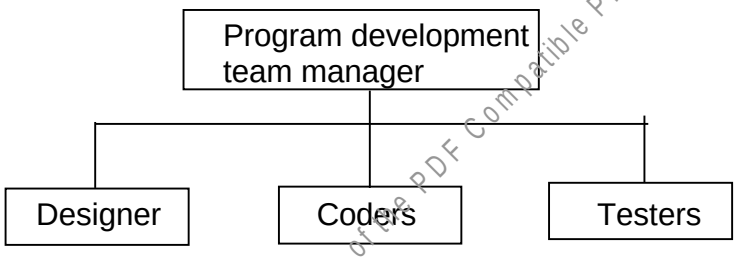
Mengorganisasikan Proyek Pembangunan SW

Sistem berbasis lokal dibangun dari perencanaan ke analisa dan perancangan umum ke rinci, untuk diprogramkan atau dikonversikan oleh seseorang yang disebut sebagai programmer/ analyst. Jika program pesanan diperlukan, normalnya merupakan program kecil. Tetapi jika pindah ke proyek yang lebih besar lagi, misalnya membutuhkan bermacam2 bidang keahlian yang dibutuhkan dalam suatu tim, maka perlu ada manajemen (koordinasi, integrasi dan

komunikasi). Yang diperlukan sebenarnya adalah bagaimana cara kodingnya dan bagaimana hasil akhirnya.

Pendekatan Organisasi
Banyak cara dalam mengorganisasikan tim pemrograman :

ü Program development team



ü Chief programmer team
Dibentuk dari pemrogram super (chief) yang memiliki kekayaan pengalaman dalam pengetahuan pemrograman. Orang ini dapat berkomunikasi secara efektif dengan analis sistem, perancang, user dan teknisi, bertindak sebagai manajer yang baik. Chief ini dibantu oleh asisten sebagai orang kedua dalam proyek dan mengkomunikasikan ke semua orang dalam tim, bertindak sebagai penerus ide dari chief pemrogram.
Bergantung pada ukuran dari proyek, 2 orang ini dibantu oleh :
§ Support programmer (pemrogram junior) : melakukan koding model berlevel rendah
§ Librarian : memelihara produksi program, mengindeks file yang dikompilasi dan dites, dan menjaga agar library kode sumber dan kode obyek up-to-date.
§ Administrator : menangani semua pendukung non teknik, seperti biaya, staf, birokrasi, dll)
§ Editor : bertanggungjawab atas dokumentasi hasil, penelitian, dan melihar semua reproduksi & distribusi dokumentasi.
§ Program clerk : menjaga agar semua record teknik untuk tim pemrograman dan menyediakan fasilitas kesekretariatan yang dibutuhkan.

ü Egoless programming team
Tim ini tanpa bos atau pemimpin. Tim ini terdiri dari orang2 yang bertanggungjawab atas pembangunan SW. Tim memanage setiap hari tanpa ada supervisi langsung. Anggota tim membuat keputusan secara demokratis untuk menghindari konflik. Setiap anggota mereview dan mengkritik pekerjaan anggota lain. Terdiri dari librarian, editor dan clerk program.

Yang Membedakan Ketiga Tim Tsb.
Lihat fotokopi gambar 15.6

Program development team melihat aturan pembangunan program 40-20-40, sedangkan tim chief programming dan egoless hanya menekankan pada fungsi pengkodingan.
Jumlah dan ukuran dari jalur komunikasi di antara anggota tim juga berbeda. Kesalahan perancangan dan pengkodingan terjadi pada interface, yaitu saat perancang A berkomunikasi dengan B atau pemrogram A berkomunikasi dengan pemrogram B. Jumlah jalur komunikasi dan interface untuk n orang pada tim egoless adalah n(n-1)/2.

Apa Konsep dari Pabrik SW
Tujuan :
- ü Aplikasi dari pendekatan teknik terhadap sistem dan pembangunan SW
- ü Tool pemodelan dan teknologi CASE
- ü Instalasi dari teknik manajemen proyek
- ü Penekanan faktor perancangan MURRE
- ü Pencapaian sistem yang optimum dan produktivitas pembangunan SW.

Jika aplikasi pesanan baru diperlukan, koding baru, jika perlu digabungkan dengan koding lama, waktu keseluruhan pembangunan SW jadi berkurang.

Kesalahan residual : kesalahan yang ditemukan setelah SW dikonversi ke operasi.

Mengukur Produktivitas dalam Pembangunan SW
Formula :

$$\text{Productivity} = \frac{\text{outputs produced}}{\text{Inputs consumed}}$$

Formula dapat ditingkatkan dengan menaikkan output atau menurunkan input atau keduanya. Input diukur dari buruh, workstation, supply dan fasilitas pendukung.
Untuk mengukur output ada 2 cara :
- ü LOEC = line of executable code
- ü Function points

Dengan pengukuran ini, memungkinkan untuk memanage proses pembangunan SW. Pengukuran apa pun yang dipakai, feature yang penting dari sistem pengukuran adalah reasonableness dan konsistensi.

Menghitung Lines of Executable Code (LOC)
Banyak yang menggunakan Source lines of code (SLOC) sebagai pengukuran. Baris dalam kode sumber adalah setiap baris dalam program yang bukan komentar atau baris kosong.
Perbaikan dari pengukuran SLOC adalah menggunakan lines of executable code (LOEC) saja, yang dianggap pengukuran yang paling baik termasuk function points karena sederhana dan banyak digunakan.

Beberapa keuntungan yang didapat dari pengukuran LOEC :

ü Easy to define and discuss with clarity
ü Widely quoted
ü Easy to measure
ü Easy to use for estimates.

Menggunakan Pengukuran Function Point
Dipakai untuk memperbaiki defisiensi dari pengukuran LOEC. 5 fungsi yang dianalisa :
ü Jumlah input, seperti form dan layar
ü Jumlah output, seperti laporan dan layar
ü Jumlah query oleh end user
ü Jumlah file logika yang diakses dan digunakan
ü Jumlah interface ke aplikasi lain.
Kelima fungsi ini yang akan dihitung jika function point dipergunakan. Function point mengukur apa yang SW development team berikan ke end user. Function point ini menekankan pada pengukuran perancangan, pengkodingan, dan tes, dimana LOEC hanya pengkodingan. Function point ini mengukur efisensi dan efektivitas.

Contoh cara menganalisas proyek pembangunan SW dengan function point :

Function point	Degree of complexity			Total
	Low	Average	High	
Input	12 x 2 = 24	3 x 5 = 15	12 x 8 = 96	135
Output	10 x 3 = 30	15 x 5 = 75	14 x 9 = 126	231
Inquery	10 x 3 = 30	16 x 6 = 96	17 x 8 = 136	262
File	9 x 4 = 36	20 x 7 = 140	10 x 10 = 100	276
Interface	12 x 4 = 48	20 x 6 = 120	20 x 10 = 200	368
	Total function points			1272

Rerata produktivitas membangun (rerata menghasilkan), diukur sebagai :

$$\text{Development productivity rate} = \frac{\text{number of function point delivered}}{\text{number of person-months}}$$

Untuk aplikasi SW secara umum, rerate produktivitasnya antara 5 – 10, berarti satu orang dapat menghasilkan sekitar 5 hingga 10 function point setiap bulan.

Dari contoh, total function point-nya 1272 dan jika rerata menghasilkan adalah 8 function point per person-month, proyek membutuhkan sekitar 159 person-month. Jika setiap bulan membutuhkan $ 10,000 untuk resource (input), proyek akan memakan biaya sekitar $ 1,590,000.
Jika perusahaan menginstal sistem CASE, rerata menghasilkan akan melonjak menjadi 15 atau 20 function point per person-month. Dan jika CASE digabungkan dengan prosedur pembangunan SW, reusability dari kode dan perusahaan mencapai 50% penggunaan kode ulang, yang memungkinkan

pencapaian rerata produktivitas membangun lebih dari 70 function point per person-month.

Ke-valid-an penggunaan person-month untuk estimasi waktu dan biaya dan evaluasi produktivitas bergantung pada orang yang mengerjakannya.

Beberapa kegiatan yang mengurangi produktivitas :
- ü Poor or nonexistent productivity metrics
- ü Poor planning and control
- ü Poor mix of skills
- ü Premature coding
- ü Inequitable rewards.

Memproduksi SW yang Berkualitas Tinggi

Untuk mendapatkan suatu SW yang berkualitas tinggi dan bagaimana pencapaiannya, ada 3 dimensi yang perlu dilihat :
- ü Dari sudut end user, kualitas SW diukur dari faktor kinerja, yaitu :
  - § Overall operating performance
  - § Ease of learning
  - § Controls and error handling
  - § Support from developers and maintainers
- ü Faktor perancangan MURRE
- ü Faktor strategi RDM.

What is Quality Assurance (QA)

QA memastikan bahwa SWDLC yang digunakan dalam membangun produk SW yang berkualitas mengikuti standard dari produk. Pada skala yang lebih besar, QA meliputi pemantauan yang terus menerus terhadap semua sistem dan fase membangun SW dari perencanaan sistem ke implementasi. Juga pemeriksaan terhadap proses pembangunan sehingga kualitas dari sistem dan SW yang dihasilkan terbukti.

What is Quality Control (QC)

QA difokuskan pada proses sistem dan pembangunan SW. QC difokuskan pada produk, yaitu yang dihasilkan. Meskipun proses berhasil, kadang2 produk gagal. Hal ini karena QC mengevaluasi sistem dan SW setelah dibangun, bukan jumlah aktivitas QC yang dibuktikan berkualitas.Kualitas harus dilihat dari sistem dan SW pada saat ssedang dibangun, bukan setelah pembangunan selesai.
QA menerapkan teknik error-prevention, QC menerapkan teknik error-removal. QA tidak bisa menjadi QC, begitu pun sebaliknya.

Membentuk Grup QA

Grup ini terdiri dari wakil2 end user, analis sistem, perancang sistem dan pemrogram2 berpengalaman, semuanya bukan merupakan bagian dari perusahaan. Tugasnya adalah :
- ü Mencapai standar sistem dan SW, yaitu mengacu ke SDLC & SWDLC

- ü Mengevaluasi dokumen yang dapat dihasilkan
- ü Melaksanakan pembangunan sistem dan SW
- ü Mengkodingkan
- ü Melaksanakan tes.

QC akan berlaku setalah semua prosedur yang dites diaplikasikan dan grup QA dan perusahaan percaya bahwa SW siap untuk direlease ke implementasi dan konversi ke operasi. QC berperan selama tes sistem dan acceptance.

Perencanaan Proyek SWDLC

Perencanaan ini ditujukan bagi manajer proyek untuk menjadualkan dan memantau semua tugas yang diperlukan untuk menyelesaikan SWDLC. Tool yang dipakai adalah Program Evaluation and Review Technique (PERT).

PERT bertujuan untuk menentukan urutan pembangunan SW yang mana yang lebih dahulu dan mengestimasikan berapa lama proyek berjalan hingga selesai. Ada 4 langkah pada PERT untuk membangun SW :

1. Identifikasi semua tugas membangun SW yang harus dilaksanakan
2. Estimasi waktu yang dibutuhkan untuk menyelesaikan setiap tugas
   Formula yang digunakan :

$$TE = \frac{O + 4M + P}{6}$$

   TE = expected time
   O = optimistic time estimate
   M = most likely time estimate
   P = pessimistic time estimate
3. Menentukan urutan tugas
4. Menentukan jalur kritis yang mendikte total waktu pembangunan SW.

Lihat fotokopi gambar 15.8 & 15.9.

Para manajer proyek disarankan untuk menggunakan PERT dalam meningkatkan kontrol dari proyek dan mencapai rerata produktivitas yang tinggi.

B. SW Designing

Pembangunan / perancangan SW dibuat untuk mencapai transisi perancangan design-to-software menjadi sebuah sistem yang akurat.
Ada 2 pendekatan :
Structured SW design
Object-oriented SW design

Alasan membuat Fase Perancangan SW

Semakin banyak waktu dan dukungan untuk merancang SW, semakin baik kualitas SW dan semakin sedikit terjadi masalah pada akhir fase, semakin sedikit sumber daya yang dibutuhkan untuk siklus kehidupan sistem secara total.

Biaya perbaikan sebuah kesalahan lebih besar setelah program diimplementasikan dibandingkan fase perancangan dan pengkodingan.

Lihat fotokopi gambar 16.2.

Perancangan SW Terstruktur

Pada fase perancangan SW terstruktur ini, level terendah dari DFD dibuat untuk sebuah diagram struktur dengan deskripsi kode bahasa pemrograman.

Karakteristik perancangan

Program terstruktur memiliki karakteristik :

- ü Modul secara hirarki
- ü CALL-based atau PERFORM-based
- ü Perancangan top-to-down dan aliran kontrol, dan top-to-bottom atau bottom-to-top
- ü Repetisi (perulangan) atau loop
- ü Kontrol standard untuk urutan, pilihan dan repetisi.

Lihat fotokopi gambar 16.3.

Perancangan Terbagi dalam Modul2

Modularitas merupakan pembagian perancangan SW ke komponen2 individu, disebut modul, objek untuk mengurangi kompleksitas. Tujuan pendekomposisian ke dalam modul :

- ü Modul design, code dan tes terpisah
- ü Revise dan maintain modul mudah setelah dikonversikan ke operasi.

Tujuan tekniknya :

- ü Pass as few data as possible between modules
- ü Minimize the use of control data
- ü Effect module independence
- ü Focus modules on single functions.

Coupling

Coupling mengukur derajat kebebasan dan interaksi antara modul. Jika modul bebas, tidak perlu ada modifikasi oleh modul lain. Interaksi berarti perubahan dan modifikasi variabel antar modul.

Jika interaksi kecil terjadi di antara 2 modul, modul berderajat kebebasan tinggi dan dianggap sebagai *loosely coupled*.

Jika interaksi yang terjadi cukup besar di antara modul2, modul berderajat bebas dan disebut *tightly coupled*. Jika tightly coupled, 1 modul tidak mungkin dimaintain tanpa merubah modul lainnya.

Beberapa koneksi antar modul harus ada dalam sebuah program. Perancangan berkualitas tinggi berarti modul adalah loosely coupled dan mudah dimaintain dan reusable oleh aplikasi lain.

Tipe2 dari cooupling :

- ü Data coupling
  Jika data yang diperlukan dikomunikasikan antar modul
- ü Stamp coupling
  Jika komunikasi grup berhubungan dengan item data seperti record yang berisi bermacam2 field
- ü Control coupling
  Jika satu dari modul mengkomunikasikan data yang mengontrol logika internal dari modul lain
- ü Common coupling
  Jika modul2 berbagi data yang disimpan pada common area
- ü Content coupling
  Jika satu modul menunjuk atau mengubah isi dari modul lain.

Lihat fotokopi gambar 16.5 & gambar 16.6.

Cohesion

Pengukuran kebebasan modul yang lain, sering disebut binding. Kohesi mengukur kekuatan dari relasi antar elemen dari kode dalam modul.
Sistem dengan modul berkohesi tinggi memiliki loose coupling atau yang rendah, keduanya merupakan tujuan perancangan dari perancangan SW terstruktur. Tipe2nya :

- ü Functional cohesion
  Semua prosedur berkontribusi terhadap eksekusi dari satu dan hanya satu tugas yang well-defined
- ü Sequential cohesion
  Prosedurnya berisi aktivitas yang terurut seperti output dari satu kegiatan dari merupakan input untuk kegiatan berikutnya dari modul
- ü Communicational cohesion
  Prosedurnya berkontribusi ke aktivitas2 yang menggunakan data yang sama untuk tujuan yang berbeda
- ü Procedural cohesion
  Prosedurnya berisi aktivitas yang tidak berhubungan dan berbeda dimana kontrol mengalir dari setiap aktivitas ke berikutnya
- ü Temporal cohesion
  Semakin kuat relasi antar prosedut berarti modul2 dapat dieksekusi pada waktu yang sama
- ü Logical cohesion
  Jika elemennya tidak terhubung dengan aliran data atau aliran kontrol tetapi berhubungan hanya dengan tugas2 dari fungsi yang sama
- ü Coincidental cohesion
  Relasi yang tidak berarti antar prosedur modul terjadi.

Guidelines

Beberapa guideline untuk merancang modul2 yang well-designed :

- ü Factoring (leveling) and module size

ü Decision splitting
ü Fan-in/fan-out
ü Number of modules.

Top-to-Bottom Design & Top-to-Bottom or Bottom-to-Top Coding
Hasil pendekatan ini adalah struktur hirarki dari program modul. Dalam beberapa situasi, campuran dari koding dan testing top-down atau bottom-up dipakai.

Using standar control construct to build structured programs
Setiap logika program dapat dibentuk dari kombinasi 3 kontrol konstruksi :
ü Sequence
ü Selection
ü Repetition

Lihat fotokopi gambar 16.9.

Ada 2 tool pemodelan yang populer digunakan untuk merancang SW terstruktur, yaitu :
ü Structured charts
  Diagram hirarki yang mendefinisikan semua arsitektur perancangan SW dengan menampilkan modul2 dan keterhubungannya
  Lihat fotokopi gambar 16.10.
ü Structured english

```
ORDER_ENTRY :
        FOR each customer PURCH_ORD
                GET CUSTOMER record
                IF CUS_NUM is valid
                        SET INVOICE_HEADER record
                        ENTER CUS_NAME, CUS_ADDR, DATE_ORD,
                                and PO_NUM in INVOICE_RECORD
                        WRITE INVOICE_HEADER record
                        GET DISCOUNT from table
                ELSE
                        Display “Invalid Customer Number”
                        QUIT ORDER_ENTRY
                ENDIF
                FOR each line item
                        COPY ITEM_NUM and QTY_ORD on
                                INVOICE record
                        GET ITEM_PRICE from price table
                        SET ITEM_SUBTOTAL to ITEM_PRICE x QTY_ORD x
                                                        (100-DISCOUNT)
                        SET INVOICE_TOTAL to sum of ITEM_SUBTOTAL
                        Write INVOICE_RECORD
                ENDFOR
                Prepare bill of loading
        ENDFOR
EXIT ORDER_ENTRY.
```

Biasanya menggunakan DFD dan STD.

Transformasi DFD ke Bagan Terstruktur
Analisa transform meliputi :
Melaksanakan input dan pengeditan, seperti validasi
Melaksanakan pemrosesan, seperti pemeriksaan inventori
Mengenerate output, seperti menyiapkan dokumen.

Gambar 16.13 & 16.14.

Perancangan SW Berorientasi Objek (OO SW Design)

Object
Objek dianggap orang atau sesuatu dengan identitas. Objek merepresentasikan entitas kongkrit dari domain aplikasi yang dirancang.
Contoh : student, customer, analis sistem, no rekening, inventori, dll.

Objek mengenkapsulasi atribut data (struktur data/ atribut) dan operasi (prosedur). Operasi berisi metode (koding program) yang mengoperasikan atribut. Beberapa otoritas mendefinisikan objek sebagai data, kode yang terenkapsulasi atau data dan metode.

Sekali perancangan SW dibuat, format data harus dideskripsikan rinci dan kode program (metode) harus ditulis untuk implementasinya.

Classes
Kelas objek menggambarkan kumpulan objek dengan atribut yang mirip, perilaku bersama dan relasi bersama ke objek lainnya.

Macam kelas :
- ü Sub kelas
- ü Super kelas

Semua objek dari sub kelas merupakan bagian dari super kelas juga.

Inheritance : kemampuan untuk mendefinisikan objek sub kelas dari sebuah kelas objek. Dilihat dari terminologi genetic, inheritance merupakan kemampuan keturunan (descendant) untuk menerima karakteristik dari nenek moyangnya (ancestor).

Lihat fotokopi gambar 16.15 & 16.16.

Relationships
Relasi menggambarkan keterhubungan antara kelas dengan objek. Relasi kelas memungkinkan inheritance.

Dalam OO, sering disebut dengan Multiplicity, yaitu berapa banyak instance dari sebuah kelas berelasi dengan sebuah instance tunggal dari kelas yang berasosiasi.

Menggunakan Pendekatan Perancangan OO ke Model SW Perancangan
Identifikasi kelas dan objek merupakan dasar dari perancangan SW OO. Perancangan SW OO mengidentifikasikan kelas dan objek yang diimplementasikan pada domain. Secara umum, cara paling baik untuk membuatnya adalah membangun sebuah kelas hirarki dimana sub kelas inherit atribut dan operasi dari kelas dan super kelas yang lebih umum.

Pemodelan OO
Untukb beberapa modifikasi, menggunakan ERD. Seperti perancangan SW terstruktur, perancangan OO sering diambil dari DFD. Setiap sink, proses atau data store pada DFD terdekomposisi penuh mengindikasikan objek kandidat atau beberapa objek. Kandidat kelas berasal dari aliran data DFD. Kamus data dan model data logik disiapkan selama perancangan sistem terinci menjadi atribut data dalam objek atau beberapa objek.

Menggambarkan Objek menggunakan Notasi ERD yang Dimodifikasi
Class box

Class :	Checking account
Attributes :	Name Address Balance
Operations :	Open (create new account) Deposit Withdraw Close (delete account)

Listing attributes
Object : CAR
Attribute : color

Instance objek yang berbeda memiliki nilai yang berbeda atau sama untuk atribut yang diberikan.

Listing Operations & Methods
Operasi berada di area ketiga dari class box. Operasi merupakan fungsi atau transformasi yang diaplikasikan ke atau oleh objek dalam kelas.
Operasi yang sama dapat diaplikasikan ke banyak kelas yang berbeda, merupakan karakteristik polymorphic dari pendekatan OO.
Sebuah metode (kode program) merupakan implementasi dari sebuah operasi kelas.

Pemodelan Relasi Antara Objek & Kelas
Relasi menggambarkan asosiasi dari kelas dan objek. Relasi antara objek berarti berarti objek dapat mengirim pesan ke yang lainnya. Pesan bersifat bi-directional. Meskipun relasi dimodelkan bi-directional, tidak perlu diimplentasikan dalam 2 arah.

Lihat fotokopi gambar 16.18.

Pemodelan Inheritance (Penurunan)
Setiap sub kelas tidak hanya menurunkan semua sifat dari super kelas tetapi dapat juga menambah atribut khusus dan operasi. Sifat inheritance ini disebut pengecualian (extension).

Lihat fotokopi gambar 16.19.

Faktor perancangan MURRE, dapat diaplikasikan ke perancangan SW OO dan juga ke perancangan SW terstruktur. Modular yang mendukung faktor perancangan MURRE dapat diaplikasikan ke perancangan OO.

Modul untuk perancangan SW terstruktur sebaiknya kecil dan berarti. Metode seharusnya standar dan mempunyai variabel2 yang berarti, menghindari singkatan, dan menggunakan kondisi standar dan tipe data standar. Metode dapat dimengerti oleh seseorang selain pemrogram dari metode itu dan setelah beberapa waktu harus dimengerti oleh pemrogram. Seperti kode terstruktur, kode OO juga perlu didokmentasikan. Dokumentasi dari metode menggambarkan tujuan, fungsi, input, dan output, seperti mendeskripsikan objek. Enkapsulasi melibatkan pembedaan dari aspek eksternal objek yang dapat diakses ke objek dari atribut internal dan operasi objek. Enkapsulasi membutuhkan loose coupling, dan inheritance membutuhkan tighter coupling.

Banyak pemrograman OOP mendeklarasikan atribut dan operasi sebagai public dan private. Atribut public dapat dibaca dan operasi public dapat dieksekusi. Atribut dan operasi dari private tidak dapat dibaca dan dieksekusi.

Perbedaan

Structured SW Design	OO SW Design
Proses atau data Menunjukkan proses (fungsionalitas) Berbasis pada cakupan sistem	Objek Menunjukkan kelas objek, relasi antar objek dan kelas Fleksibel untuk diubah dan dapat diperluas

Pemilihannya adalah dengan melihat reusability. Tujuan idealnya adalah memiliki class library semua objek yang ada untuk mendukung aplikasi perancangan. Class library ini didokumentasikan dan dipublikasikan untuk keberadaannya.

Performing A SW Design Walkthrough
Ada 2 variabel utama :
- ü Tingkat keformalan atau struktur walkthrough
- ü Timing, selama SDLC dan SWDLC dapat dilakukan design walkthrough.

SW design walkthrough dilaksanakan suatu tim yang memutuskan :
- ü Apakah menerima perancangan tanpa modifikasi selanjutnya
- ü Apakah menolak perancangan karena kesalahan besar
- ü Apakah menerima perancangan seperti apa adanya.

Keputusan diambil untuk merelease perancangan untuk pengkodingan atau mengirimkan kembali ke perancangan ulang. Setelah beberapa waktu, tim akan melihat kelanjutkan perancangan untuk memastikan semua kesalahan diperbaiki dan tidak ada yang terabaikan.

Lihat fotokopi gambar 16.20.

C. SW testing

Tujuan dari testing SW adalah reliability yang membutuhkan deteksi dan penghapusan kesalahan.
Karena manajer proyek harus mengontrol jadual dan biaya, reliability berhubungan dengan seberapa baikkah proyek sistem dimanage secara keseluruhan.
Testing SW menyediakan dasar terdokumentasi untuk memastikan bahwa program akan dilaksanakan sesuai kebutuhan.

Apa SW Testing ?
Tes SW berarti proses yang harus mengikuti pola dan perencanaan yang well-defined.

Pelaksana : kelompok QA atau tim tes.

Persiapan Kasus Tes
Lihat fotokopi gambar 18.2.

Area yang perlu dites :

Field	Data entry
Record	Controls
File	Program flow

Kesalahan SW dan Hubungannya dengan Reliability SW
Reliability SW dapat diekspresikan dari kesalahan per KLOEC yang ada. Tipe2 kesalahan yang biasa dilakukan :
- ü Fatal error
  Crash

Logika
hang
- ü Serious error
- ü Minor error.

Penyebab kesalahan program SW adalah bug. Bug merupakan cacat yang tidak diharapkan, cacat, salah atau tidak sempurna.

Debugging
Debugging adalah menghapus bug. Terjadi dari tes yang berhasil. Kasus tes (test case) mendeteksi kesalahan; debugging menghapus bug dan memperbaiki kesalahan.

Tujuan Testing : reliable SW.

Merancang Test Cases
- ü White box testing : berdasarkan pengujian struktur logika internal SW. Perintah2 dasarnya : SELECT, OPEN/ CLOSE, COPY REPLACING, IF, PERFORM UNTIL and PERFORM WHILE, CALL
- ü Black box testing : melihat apakah fungsi SW beroperasi yaitu output yang dihasilkan input benar, dan database diakses dan diupdate secara benar pula.

Ke2 tes ini memperbaiki kesalahan selama pengkodingan.

Testing equivalence classes
Bagian dari black box testing. 2 nilai input berada pada kelas ekuivalen yang sama.
Contoh : input data untuk field data adalah 1 .. 50.

2 tipe kelas ekuivalen : valid dan invalid.

Using equivalence to build test cases
Kelas ekuivalen dibuat untuk mengurangi jumlah kasus tes yang harus dirancang.
Contoh :
- ü Check to see if control totals are prepared properly
- ü Try to process a sensitive transaction without proper authorization
- ü Make numeric, alphabetic, and special character checks
- ü Input a field with a negative sign to see if it is handled as a negative value
- ü Divide an amount by zero
- ü Perform validity checks on key data fields
- ü Make range and reasonableness checks
- ü Check for proper transaction queue
- ü Include an amount number with a predetermined check digit and see if it is processed properly

- ü User units of measure different from those allowed
- ü Input several fields with incomplete or missing data
- ü Insert characters in fields to cause an overflow condition
- ü Try to read from or write to a wrong file.

Lihat fotokopi gambar 18.3.

Setelah semua kesalahan didapatkan, sebuah laporan kesalahan perlu dibuat.
Contoh :

Lihat fotokopi gambar 18.4.

Kode resolusi mengindikasikan apa yang pemrogram lakukan dengan laporan kesalahan. Meliputi :
- ü It is fixed as recommended
- ü The error cannot be reproduced
- ü The error cannot be fixed
- ü The coder disagrees with the tester's suggestion
- ü The report error has been withdrawn by the tester
- ü The code about which the error is reported works in accordance with the design specifications.

Tester dan coder harus setuju untuk resolusi akhir. Keduanya harus tanda tangan dan memberikan tanggalnya.
Setelah kesalahan program diperbaiki, tes ulang perlu dilakukan pada kesalahan2 yang tidak tercover.

Testing the test cases
Probabilitas menemukan kesalahan real *i* dalam total populasi *I* pada kesalahan yang tidak diketahui dapat dihubungkan dengan probabilitas menemukan benih kesalahan *j* (seeded error) dari *J* kesalahan dalam kode.

Metode pembenihan kesalahan :

$$\frac{\text{Remaining number of real error}}{\text{Remaining number of seeded error}} = \frac{\text{number of real errors detected}}{\text{number of seeded errors detected}}$$

Merancang Strategi Tes SW
Langkah :
- ü Module testing
- ü Integration testing
- ü System testing
- ü Acceptance testing

Lihat fotokopi gambar 18.5.

Tes individu modul :
- ü Laksanakan setiap perintah dalam modul

ü Ikuti setiap jalur logika modul
ü Hitung ulang setiap perintah perhitungan
ü Tes modul dengan setiap kumpulan data input yang mungkin.

Lihat fotokopi gambar 18.6, 18.7, 18.8 & 18.9.

System Testing
Merupakan proses tes terhadap SW terintegrasi pad konteks sistem total. Tes in meliputi :
ü Recovery testing
ü Security testing
ü Stress testing

Acceptance Testing
Tes ini mengevaluasi sistem baru untuk menentukan apakan sudah sesuai dengan user requirement dan kondisi operasi.
Terdiri dari sampel orang2 yang bekerja dengan perancangan sistem. Ada 2 prosedur tes :
ü Tes alpha : usability labs & usability factors checklist

USABILITY FACTORS

	1 = low to 5 = high
A. ease of use	1 2 3 4 5
B. user friendliness	1 2 3 4 5
C. understandability	1 2 3 4 5
D. level of confidence	1 2 3 4 5
E. conformance to requirements	1 2 3 4 5
F. conformance to respon time	1 2 3 4 5
G. comfort level	1 2 3 4 5

Please circle the number applicable to each quality factor.

ü Tes beta : tidak dihadiri oleh para profesional sistem.

Merelease SW
Fase tes berhenti saat SWDLC juga berhenti karena SW telah melewati acceptance test. Semua manajer proyek ingin mensertifikasi SW untuk totally error and trouble-free, tetapi tidak mungkin karena manajer proyek mengetahui apa yang telah dites, tetapi tidak tahu tentang apa yang tidak dites.

D. SW implementation

Ada 2 proses :
1. Planning ; menentukan tugas implementasi sistem apa yang dilaksanakan dan kapan
2. Execution ; pelaksanaan dari tugas implementasi sistem.

Membuat perencanaan implementasi sistem
Perencanaan ini merupakan formula rinci dan presentasi grafik dari implementasi sistem yang bagaimana yang akan dicapai.
Biasanya diukur dalam bulan dan direpresentasikan dalam diagram PERT.
Timnya terdiri dari :
- ü Profesional sistem yang merancang sistem
- ü Manajer dan staf yang berbeda
- ü Wakil vendor
- ü User utama
- ü Koder
- ü teknisi

Contoh : lihat fotokopi gambar 19.2.

Jika hasil tes berhasil, sistem menuju ke proses konversi. Ada postimplementation review untuk memastikan sistem berhasil dikonversikan dan dioperasikan sesuai dengan perencanaan.

Bagian penting dari implementasi sistem :
- ü Site preparation
- ü Personnel training : in-house training, vendor-supplied training, outside training services
  Teknik training dan bantuannya :
  - þ Teleconferencing
  - þ Interactive training SW (CBT, audio-based, video-based, video-optical disk)
  - þ Instructor-based training
  - þ On-the-job training
  - þ Procedural manual
  - þ textbooks
- ü Documentation preparation
- ü Systems and file conversion
- ü Postimplementation review.

Menyiapkan Dokumentasi
Dokumentasi menunjukkan bagaimana sistem beroperasi. Digunakan untuk tujuan :
- ü Training
- ü Instructing
- ü Communicating

- ü Establishing performance standards
- ü Maintaining the system
- ü Historical reference.

4 area utama dokumentasi : user, sistem, SW dan operasi.

Konversi ke Sistem Baru
Lihat fotokopi gambar 19.6.

Mengevaluasi Sistem Baru setelah Implementasi
Ada 4 area postimplementation review :
- ü System factors
- ü Systems design components
- ü Accuracy of estimates
- ü Level of support

Lihat fotokopi gambar 19.7, 19.8, 19.9, 19.10 & 19.11.

SYSTEM MAINTENANCE

(reference Burch, chp. 21)

Pemeliharaan sistem dimulai ketika sistem yang baru dioperasionalkan untuk selamanya.

Lihat fotokopi gambar 20.2.

Tipe2 maintenance sistem :
- ü Corrective
- ü Adaptive
- ü Perfective
- ü Preventive

System Maintenance Life Cycle (SMLC), meliputi :
1. Maintenance request
2. Transform the maintenance request to a change
3. Specify the change
4. Develop the change
5. Test the change
6. Train users and run an acceptance test
7. Convert and release to operations
8. Update the documentation
9. Conduct a postimplementation review

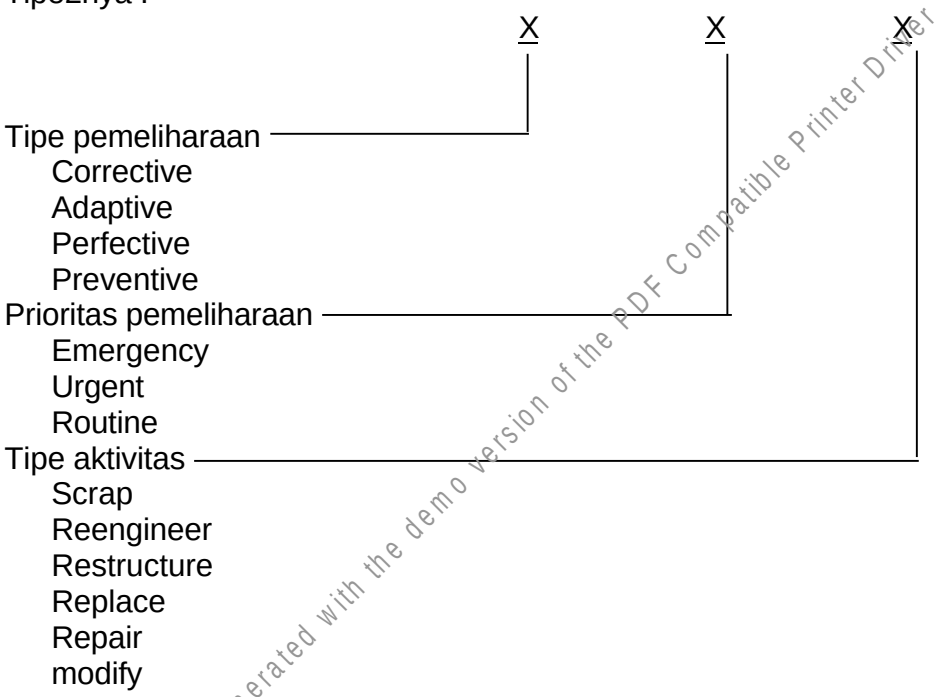
Prosedur Pemeliharaan Sistem
Ada 3 pendekatan untuk mengorganisasikan pemeliharaan sistem :
1. Separate
2. Combined
3. Functional

Menggunakan CASE tool untuk pemeliharaan sistem
Ada 5 tool CASE :
- ü Forward engineering
- ü Reverse engineering
- ü Reengineering
- ü Restructuring
- ü Maintenance expert systems.

Memantau Pemeliharaan Sistem
Dibedakan dalam 3 kategori : harian, mingguan, sewaktu2.

Inisialisasi dan recording aktivitas pemeliharaan sistem yang tidak terjadual
Dengan formulir Maintenance Work Order (WO), berisi :
- ü Work requested
- ü Work performed

ü Estimated time VS actual time
ü Maintenance code
Tipe2nya :



ü Maintenance cost

Lihat fotokopi gambar 20.5.

Mengoptimalkan Program Pemeliharaan Sistem
Lihat fotokopi gambar 20.6.

DEVELOPING A CHANGE MANAGEMENT SYSTEM (CMS)
CMS membantu mengurangi kepusingan dan kompleksitas dari perancangan sistem baru dan pemeliharaan sistem yang ada.
Merupakan bagian dari sistem CASE atau sistem yang berdiri sendiri.

Lihat fotokopi gambar 20.7.

Resiko Mengabaikan CMS
ü Lack of an accurate inventory
ü Incomplete history of program changes
ü Duplicated SW program modules
ü Unauthorized SW program changes
ü Lack of clear, comprehensive and current documentation
ü Poor SW quality and reliability