# DESAIN SISTEM SECARA UMUM



## PENDAHULUAN

Tujuan dari desain sistem secara umum adalah untuk memberikan gambaran secara umum kepada user tentang sistem yang baru.

Dalam SDLC, fase ini adalah yang ketiga. Fase ini melibatkan sebuah proses yang dilaksanakan untuk membuat desain sistem konseptual yang lain (alternatif) yang memenuhi permintaan user.
Hasil2 dari fase desain secara umum banyak yang dapat diterima yang dituangkan dalam Laporan Desain Sistem secara Umum. Alternatif ini siap untuk dievaluasi dan diseleksi untuk dicari desain sistem umum yang optimal.

Analisis sistem dan desain sistem umum bergantung satu sama lain. Studi menunjukkan bahwa apa yang dikumpulkan, dianalisis dan dimodelkan selama fase analisis menyediakan dasar bagi desain sistem umum untuk dibuat. Fase analisis sistem merupakan investigasi dan berorientasi ke temuan.

Pada fase ini, professional sistem harus sering membuat fitur yang baru atau berbeda dari model dasar yang dibuat selama analisis sistem.

Kuncinya adalah dapatkan atau tuliskan semua ke dalam kertas tanpa mencoba untuk memperbaiki desain sistem lebih awal. Aturannya adalah : berinteraksi dengan user, periksa dengan anggota tim, periksa dengan teknisi (pemrogram); desain ulang, periksa, periksa, dan periksa kembali tetapi jangan coba-coba untuk membangun detail yang lebih rendah atau spec kecil selama fase ini. Semua ini akan dilakukan jika salah satu dari desain sistem umum sudah dipilih untuk implementasi.

TIGA KATEGORI DESAIN SISTEM (lihat figure 6.2 buku Burch)

- Sistem Global-based
- Sistem group-based
- Sistem local-based

### Sistem global-based (berbasis global)

Untuk mendesain sistem yang berbasis global (global-based) membutuhkan pemeriksaan secara seksama dan lengkap atau penggantian dari seluruh komponen desain umum. Beberapa tipe perubahan yang umum adalah :

- Output yang lama : dari laporan berbentuk tabel setiap bulannya menjadi layar grafik berwarna 2 atau 3 dimensi
- Proses baru dibuat
- Input diambil dari peralatan scan daripada dengan pensil dan kertas
- Database hirarki lama diubah ke database relasional baru dengan standar bahasa query
- Kontrol yang bervariasi diinstal, termasuk UPS, DRP, peralatan enkripsi dan peralatan kontrol akses biometri
- Platform teknologi baru yang menggabungkan seluruh topologi jaringan organisasi (komputer dan peralatannya) yang mendukung

Membutuhkan beberapa tim proyek yang langsung ditunjuk dari CIO.

Lembar kerjanya berisi semua komponen desain umum berikut deskripsi masing2 secara umum. Beberapa alternatif diberikan ke user untuk direview dan diketahui. Setelah direview, alternatif beberapa aspek dapat digabungkan untuk dibuat gabungannya. Beberapa di antaranya dapat diterima atau dapat ditolak.

Lihat figure 6.3 Burch

Sekali sudah direview oleh user dan para professional sistem, desain ini siap untuk masuk ke tahap selanjutnya yaitu fase evaluasi dan seleksi (minggu 7).

**Sistem group-based (berbasis kelompok)**
Sistem ini melayani cabang2 atau group user khusus dalam organisasi. Kelompok ini memiliki kebutuhan khusus untuk menyelesaikan pekerjaan dan membuat keputusan yang tepat. Perancang sistem yang bekerja pada group ini perlu memiliki pengetahuan tentang bekerja pada sistem group-based. Perancang tidak perlu memusatkan perhatian ke perancangan desain sistem tertentu seperti database dan platform teknologi tetapi pada output, input, proses, kontrol dan untuk platform teknologi, khusus untuk group local (LAN).

**Sistem local-based (berbasis local)**
Sistem ini khusus didesain untuk beberapa orang, sering satu atau dua, untuk aplikasi khusus tambahan. User memiliki PC dan ia direncanakan untuk memiliki sistemnya. Professional sistem umumnya dipakai untuk bekerja sama dengan user menganalisis mendesain, mengevaluasi sistem yang berbeda, memilih satu dan mengimplementasikan dengan menggunakan jaringan dan pendukungnya.



EMPAT KUNCI ELEMEN DARI RAPID APPLICATION DESIGN (RAD) UNTUK MENDESAIN SISTEM
RAD dipopulerkan oleh James Martin.
Sinergismenya adalah bahwa RAD menggabungkan elemen2 yang bekerja sama sehingga dampak keseluruhannya lebih besar dibandingkan dengan jumlah dampak per individu/ masing2.

- Joint application development (JAD)
- Specialist with advanced tools (SWAT) teams
- Computer-aided System and Software Engineering (CASE) tools
- Prototyping

*JAD*
Efektif untuk digunakan di sistem global-based.
JAD dapat juga dipakai di sistem group-based mau pun local-based.
Kunci utamanya adalah joint; user dan professional sistem bekerja sama untuk menganalisis dan mendesain sistem.

Lihat figure 6.4 buku Burch

Model perancangan mental desainer
Model ini diformulasikan dari pengalaman, pengetahuan, studi lapangan dan input dari interaksi yang dilakukan dengan user.

Model desain mental user
Idealnya model ini dan model desain sistem konseptual adalah sama. Interaksi joint dan proses desain diulang hingga model desain sistem konseptual sama dengan model desain mental user.

*SWAT team*
Terdiri dari 3 atau 4 profesional sistem yang memiliki kemampuan dan motivasi. Tim proyek yang kecil lebih produktif dibandingkan dengan tim proyek untuk sistem yang lebih besar.

*Tool CASE*
Digunakan oleh tim SWAT untuk menambah produktifitas dan kualitas kerja dari membangun sistem.
§ Menambah disiplin
§ Mengurangi kesalahan dan kekosongan desain
§ Mengurangi kerja sistem yang berulang.

*Prototyping*
Bekerja dengan JAD dimana user ditunjukkan dengan apa yang akan mereka dapatkan dan meresponnya. CASE memfasilitasi prototyping untuk membuat desain layar, model2 yang bervariasi dan dialog yang cepat serta untuk memodifikasinya saat berinteraksi dengan user.

Dengan RAD, penyusunan prototyping tidak dibuang, tetapi menjadi bagian dari desain sistem akhir. Pendekatannya mencapai aturan 80:20, 80% permintaan user dapat dipenuhi dengan 20% desain sistem. Tim SWAT bekerja di akhir dari sistem. Pengalaman user membantu tim SWAT dalam mendefinisikan perubahan2 yang tidak terbayangkan.

Macam dari aturan 80:20 ini untuk membangun sistem adalah teknik kotak waktu DuPont (time box technique) dimana proyek sistem harus diselesaikan tidak lebih dari 90 hari. Pendekatan ini lebih ke teknik manajemen proyek. Jika melebihi 90 hari berarti kehilangan kesempatan bisnis dan akan melebihi estimasi waktu dan uang.

PENDEKATAN DESAIN BERORIENTASI STRUKTUR (STRUCTURED-ORIENTED)
Pendekatan ini berbasis pada metodologi, tool pemodelan, dan teknik dari pendekatan terstruktur. Dibagi 2 :
§ Pendekatan berorientasi proses
§ Pendekatan berorientasi data

Tujuan kedua pendekatan ini adalah untuk mengidentifikasikan semua atribut data yang dibutuhkan oleh sistem yang dibangun. Yang berorientasi proses dikerjakan

dengan memeriksa semua input, output dan proses untuk sistem. Sedangkan berorientasi data memeriksa keputusan2 yang dibuat sistem dan kemudian bekerja ke belakang untuk mengidentifikasikan data yang dibutuhkan untuk mendukung keputusan tersebut.

Pemeriksaan I, O dan P untuk menentukan Kebutuhan Data
Yang dilihat adalah semua laporan, tampilan layar, dan keputusan yang dibutuhkan untuk sebuah proses. DFD sering dipakai untuk pendekatan ini.

Contoh : lihat figure 6.6 – 6.7 buku Burch

Pendekatan berorientasi proses ini bekerja sangat baik jika professional sistem mengetahui lebih lanjut tentang input, proses dan output yang dihasilkan dari sistem. Juga akan baik jika data dari setiap kumpulan aplikasi dipisahkan.
Contoh : aplikasi berbasis transaksi, misal utang, piutang, gaji, kontrol inventori.

Hasil dari sistem ?
Output berubah secara berkala untuk memenuhi kebutuhan baru ?
Identifikasi data yang dibutuhkan untuk menyelesaikan proses yang tergantung dari variable proses ?

⟶ Gunakan pendekatan berorientasi data

Konsultasikan dengan user untuk menentukan kebutuhan data
Pendekatan ini digunakan jika proses sistem, seperti input dan output belum ditentukan. Pada kasus ini, professional sistem harus berusaha dengan user menentukan bagaimana sistem nantinya. Focusnya adalah untuk menentukan kebutuhan data untuk keputusan2 yang akan berbasis pada data.

Tahapannya
1. Diskusikan keputusan2 yang potensial yang akan dibuat dari sistem dengan user dari sistem. Misalnya :
   - § Which vendor can provide the best price ?
   - § Which vendor can deliver the best quality item ?
   - § Which vendor has the best financing options ?
   - § Which vendor has a track record of on-time deliveries ?

2. Modelkan sistem pendukung keputusan melalui tool pemodelan yang fleksibel misalnya ERD.

3. Bagi setiap criteria ke atribut2nya. Desainer sistem perlu bertanya ke pembeli tentang setiap criteria dari sistem.

4. Break down atribut yang diidentifikasikan di atas ke dalam komponen data yang paling dasar.

Membuat kamus data untuk sistem
Data yang diidentifikasikan untuk sistem perlu dibuatkan kamus datanya sebagai informasi. Kamus data merinci setiap atribut dan sub atribut sistem dan menentukan karakteristik yang sesuai. Kamus datanya meliputi :

- Ukuran
- Tipe
- Deksripsi
- Batasan dan pengecualian
- Range
- Level keamanan
- Hak akses

Ada 3 aspek dari kamus data yang perlu diperhatikan :

1. Perubahan
2. Deksripsi
3. Urutan

PENDEKATAN BERORIENTASI OBJEK
Tujuan utamanya adalah mendesain dan membangun sistem dengan mengumpulkan objek software yang dapat digunakan, bukan dengan menulis modul software dari awal.

Kunci reusability adalah untuk mendaftar (dalam library atau database) objek dimana beberapa diantaranya ada yang sesuai dengan permintaan user. Tentunya proses pencarian membutuhkan cara yang kuat dan efisien.

Elemen2 kunci
Objek : sesuatu yang dapat berhubungan dengan lingkungan, misal mobil, komputer, roti, dll. Sebuah objek menunjukkan perilaku tertentu. Orientasi objek menyediakan konsep yang sama dalam sistem.
Di sistem informasi, atribut (data atau struktur data) dan operasi dienkapsulasi (ditarik bersama = pulled together) dalam membuat objek yang berperilaku tertentu.

Operasi digunakan untuk menggambarkan bagaimana atribut diproses di level desain. Metode (prosedur, fungsi, kode) digunakan untuk menggambarkan program sebenarnya yang ditulis menggunakan bahasa pemrograman berorientasi objek (OOP). Desain sistem memfokuskan pada identifikasi objek daripada spesifikasi atribut dan kode program untuk memanipulasinya.

Ke user, objek berperilaku dan cara tertentu yaitu merespon pesan. Semua akses ke objek melalui pesan yaitu apa yang sudah dilakukan, bukan bagaimana, objek dibiarkan untuk memilih operasi yang dipakai untuk menjawab pesan.

Classes
Sebuah kelas merupakan kumpulan objek yang berbagi struktur dan berperilaku umum. Sebuah kelas adalah sebuah tipe; objek tunggal adalah instant dari kelas.

Objek dapat menurunkan atribut dan operasi dari objek lainnya dan dapat menambah atribut dan operasinya sendiri. Inheritance memungkinkan property bersama (atribut dan operasi) di antara kelas meski pun berbeda. Dengan inheritance, desainer dan programmer dapat mengurangi kerangkapan atribut dan operasi.

Kemampuan dari objek apa pun merespon pesan yang sama dan dari setiap objek untuk mengimplementasikannya disebut polymorphism.
Misal :

Fortran ISUM = IA + IB

+ = polimorfisme

SUM = A + B

Selama fase desain sistem, desainer tidak terlalu memperhatikan rincian dari atribut, operasi dan kode pemrograman, inheritance dan polimorfisme.

Tahapan dalam Membuat Desain Berorientasi Objek

1. Identifikasi kelas objek

Kandidat kelas objek didapat dari interview dengan user. Kata benda atau frase kata benda merupakan kelas objek (noun, noun phrase). Kemudian memilih hanya beberapa kelas objek saja yang relevan dengan domain aplikasi sistem. Jika 2 atau lebih kelas objek menunjukkan informasi yang sama, yang paling deskriptif yang dipilih. Kelas objek juga harus spesifik.

2. Identifikasi relasi antara kelas objek

Sebuah relasi berhubungan dengan kata kerja (verb).

3. Identifikasi atribut utama, bukan rincian spesifikasi

Setelah relasi dibuat di antara kelas objek, atribut kelas objek utama perlu diidentifikasi, dengan kata kata sifat (adjective). Di fase desain sistem ini, atribut dan nilainya perlu dibuat lengkap. Gambarkan atribut yang paling penting dahulu, rincian selanjutnya dapat ditambah kemudian.

4. Tentukan relasi inheritance dan buat hirarki kelasnya.

Inheritance digunakan untuk mengeneralisasi aspek umum dari kelas yang ada ke dalam super kelas (bottom up) atau dari kelas perbaikan yang ada ke sub kelas (top down) khusus. Atau dengan membangun suatu hirarki kelas dimana sub kelas inherit (mewarisi) property dari super kelas.

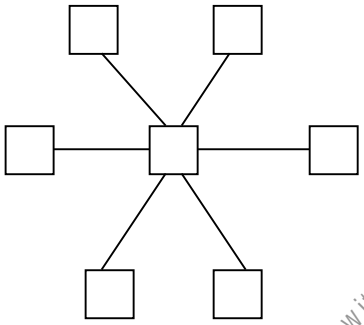
PERBEDAAN DESAIN BERORIENTASI TERSTRUKTUR DENGAN BERORIENTASI OBJEK

Banyak yang mengatakan bahwa desain berorientasi terstruktur merupakan kependekan dari desain berorientasi objek.

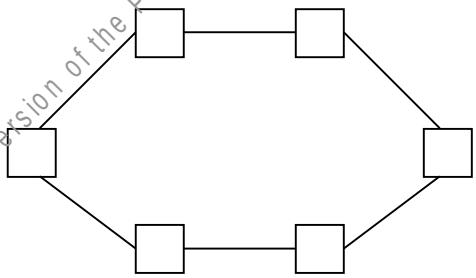
| Berorientasi terstruktur | Berorientasi objek |
|---|---|
| Modul merupakan unit dari kode software yang menjalankan fungsi. | Modul merupakan objek yang mengenkapsulasi atribut dan kode program untuk berjalan. |

Modularitas merupakan prinsip dasar dari desain yang baik, karena memfasilitasi maintenance, reusability, reliability dan extendability sistem. (MURRE)
Modularitas merupakan tingkat dimana modul dibuat standar dan bebas dan menunjukkan keragaman dalam penggunaannya. Jika desain sistem terdiri dari n buah modul, jumlah keterhubungan antar modul (disebut binding dan coupling) mendekati minimum n-1 (a) dan maksimum n(n-1)/2 (d).
Jika terlalu banyak hubungan di antara modul, efek perubahan atau kesalahan jadi meluas ke jumlah modul yang semakin besar.

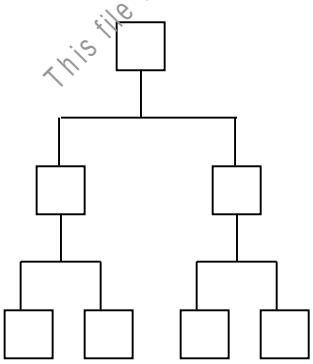




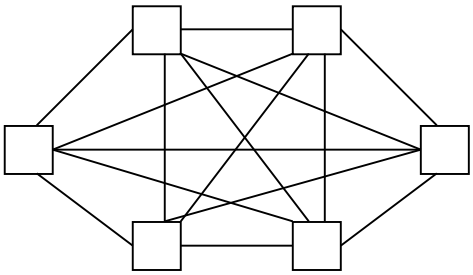
(a)



(b)



(c)



(d)

struktur modul (a) menunjukkan desain sentralisasi, yaitu modul eksekutif pusat dikomunikasikan ke setiap modul yang lain. Struktur modul (c) berlaku sama, menunjukkan struktur top-down dengan sejumlah besar komunikasi di antara modul yang teratas dengan modul di bawahnya. Modul2 ini berorientasi terstruktur. Struktur (d) memiliki keterhubungan yang paling besar dan juga berpotensi mengalami kesalahan yang besar. Desain yang memiliki keterhubungan yang rendah adalah (a) atau (c). Setiap modul berkomunikasi hanya dengan 2 tetangga terdekatnya, tetapi tidak ada otoritas pusat atau top-level. Desain (b) tidak terstruktur, bukan top-down seperti (c). Desain terstruktur ini dimana pendekatan desain berorientasi objek akan berlkau juka sesuai dengan kebutuhan user.

Pengontrolan jumlah dan bentuk komunikasi di antara modul merupakan pokok dari modularitas dan kunci tujuannya adalah modul otonomi, yang berdiri sendiri. Tetapi dalam desain berorientasi terstruktur, desain tidak dapat dibangun jika sangat tinggi derajat kebebasan modulnya yaitu very loosely tight ke modul lain.

Top-down VS Bottom-up

*Top-down*

Desain ini mengarahkan analis dan desainer untuk memulai dengan deskripsi abstrak dari sistem baru dan memperbaikinya pada tahap2 selanjutnya.

Lihat figure 6.18 buku Burch.

Desain berorientasi objek mengadopsi dekomposisi top-down ini. Desainer membuat daftar bermacam2 kebutuhan yang sesuai dengan domain aplikasi sistem, tetapi menunda spesifikasi urutan dimana kebutuhan akan diaplikasikan. Ini yang disebut dengan “shopping list”, daftar kebutuhan ditentukan selama analisis, kemudian kelas objek digabungkan untuk melihat apakah kumpulan objek ini sesuai dengan permintaan user. Jika tidak, beberapa kelas objek dikembalikan ke daftarnya dan yang lain digabungkan.

*Bottom-up*

Proses ini seperti membuat mobil baru. Objek berasal dari beberapa sumber, satu per satu dikumpulkan. Hal yang baru adalah membuat coretan. Modul yang sama (objek) digunakan kembali. Produk fisik mudah dimaintain, komponen standar digunakan dan diandalkan karena standarisasi dan ujicoba; dan produk dapat diperluas karena struktur modulnya.

Menggunakan Library kelas objek

Yang sangat menjanjikan dalam pendekatan berorientasi objek ini adalah reusability. Tujuan dari desainer berorientasi objek adalah membuat library sehingga professional sistem dapat melihat katalog kelas objek, memilih kelas objek yang sesuai, mengambilnya dari library kelas objek atau database dan menggabungkannya menjadi sistem (select, retrieve, assemble)
Misalnya adanya menu, kotak dialog dan icon.