MODUL REKAYASA PERANGKAT LUNAK (RPL)

(CCC-110)

MODUL 07

STRUCTURED DESIGN

DISUSUN OLEH

MALABAY,S.KOM,M.KOM

UNIVERSITAS ESA UNGGUL

2020

STRUCTURED DESIGN

A. Kemampuan Akhir Yang Diharapkan

Setelah mempelajari modul ini, diharapkan mahasiswa mampu : Mahasiswa mampu memahami pengertian *Structured Design***.**

B. Uraian dan Contoh

Structured Design merupakan metodologi yang sistematis untuk menentukan spesifikasi desain perangkat lunak. Prinsip dasar, alat, dan teknik metodologi terstruktur dibahas dalam bab ini. Ini mencakup empat komponen desain perangkat lunak, yaitu, desain arsitektur, desain detail, desain data dan desain antarmuka. Bab ini menjelaskan konsep, alat, dan teknik desain terstruktur berikut:

- *Coupling and cohesion*
- *Structure chart*
- *Transaction analysis and transform analysis*
- *Program flowchart*
- *Structured flowchart*
- *HIPO documentation*

Coupling adalah ukuran tingkat interdependensi antar modul. Perangkat lunak yang baik akan memiliki kopling rendah.

Jenis *Coupling*:

Data Coupling: Jika ketergantungan antar modul didasarkan pada fakta bahwa berkomunikasi hanya dengan melewatkan data, maka modul tersebut dikatakan sebagai data coupled. Dalam penggandengan data, komponen tidak bergantung satu sama lain dan berkomunikasi melalui data. Komunikasi modul tidak berisi data gelandangan. Contoh-sistem penagihan pelanggan.

Kopling Stempel Dalam kopling stempel, struktur data lengkap diteruskan dari satu modul ke modul lainnya. Oleh karena itu, ini melibatkan data gelandangan. Ini mungkin diperlukan karena faktor efisiensi - pilihan ini dibuat oleh desainer yang berwawasan, bukan programmer yang malas.

Kopling Kontrol: Jika modul berkomunikasi dengan melewatkan informasi kontrol, maka modul tersebut disebut sebagai gabungan kontrol. Ini bisa menjadi buruk jika parameter menunjukkan perilaku yang sama sekali berbeda dan bagus jika parameter memungkinkan pemfaktoran dan penggunaan kembali fungsionalitas. Contoh- fungsi sortir yang mengambil fungsi perbandingan sebagai argumen.

Kopling Eksternal: Dalam kopling eksternal, modul bergantung pada modul lain, di luar perangkat lunak yang dikembangkan atau jenis perangkat keras tertentu. Mantan protokol, file eksternal, format perangkat, dll.

Common Coupling: Modul memiliki data bersama seperti struktur data global Perubahan dalam data global berarti menelusuri kembali ke semua modul yang mengakses data tersebut untuk mengevaluasi efek perubahan. Jadi ada kekurangan seperti kesulitan dalam menggunakan kembali modul, berkurangnya kemampuan untuk mengontrol akses data dan berkurangnya pemeliharaan.

Content Coupling: Dalam content coupling, satu modul dapat memodifikasi data modul lain atau aliran kontrol diteruskan dari satu modul ke modul lainnya. Ini adalah bentuk kopling terburuk dan harus dihindari.

C*ohesion* adalah ukuran sejauh mana elemen-elemen modul terkait secara fungsional. Ini adalah sejauh mana semua elemen yang diarahkan untuk melakukan satu tugas terkandung dalam komponen. Pada dasarnya, kohesi adalah perekat internal yang menyatukan modul. Desain perangkat lunak yang baik akan memiliki kohesi yang tinggi.

Jenis *Cohesion*:

Kohesi Fungsional: Setiap elemen penting untuk satu komputasi terkandung dalam komponen. Sebuah kohesi fungsional melakukan tugas dan fungsi. Ini adalah situasi yang ideal.

Sequential Cohesion: Sebuah elemen mengeluarkan beberapa data yang menjadi masukan untuk elemen lain, yaitu, aliran data antar bagian. Itu terjadi secara alami dalam bahasa pemrograman fungsional.

Kohesi Komunikasional: Dua elemen beroperasi pada data masukan yang sama atau berkontribusi terhadap data keluaran yang sama. Contoh- perbarui catatan ke dalam database dan kirimkan ke printer.

Kohesi Prosedural: Elemen kohesi prosedural memastikan urutan eksekusi. Tindakan masih terhubung dengan lemah dan sepertinya tidak dapat digunakan kembali. Misalnya menghitung IPK siswa, mencetak catatan siswa, menghitung IPK kumulatif, mencetak IPK kumulatif.

Kohesi Temporal: Unsur-unsur terkait dengan waktu yang terlibat. Sebuah modul yang terhubung dengan kohesi temporal, semua tugas harus dijalankan dalam rentang waktu yang sama. Kohesi ini berisi kode untuk menginisialisasi semua bagian sistem. Banyak aktivitas berbeda terjadi, semuanya pada waktu yang sama.

Kohesi Logis: Unsur-unsur terkait secara logis dan tidak fungsional. Ex- Komponen membaca input dari tape, disk, dan jaringan. Semua kode untuk fungsi ini ada di komponen yang sama. Operasi terkait, tetapi fungsinya berbeda secara signifikan.

Coincidental Cohesion: Elemen tidak berhubungan (tidak berhubungan). Elemen-elemen tersebut tidak memiliki hubungan konseptual selain lokasi dalam kode sumber. Itu tidak disengaja dan merupakan bentuk keterpaduan yang paling buruk. Ex- cetak baris berikutnya dan balikkan karakter string dalam satu komponen.

Desain Terstruktur adalah metode untuk mengubah `` spesifikasi informal " termasuk sekumpulan diagram aliran data dan spesifikasi proses menjadi desain yang dapat diimplementasikan menggunakan bahasa seperti Pascal atau C. Ini

dikembangkan oleh Edward Yourdon dan Larry Constantine, dilaporkan dengan memeriksa sejumlah sistem yang telah dikembangkan pada tahun 1960-an yang dianggap sebagai contoh desain yang baik, dan mengidentifikasi fitur dan pola umum yang terkandung di dalamnya.

Desain Terstruktur tidak menghasilkan desain yang memiliki sejumlah properti yang diinginkan:

Itu tidak memanfaatkan fitur-fitur baru yang disediakan oleh bahasa pemrograman berorientasi objek. Sementara desain terstruktur menyediakan beberapa dukungan terbatas untuk `` enkapsulasi data " (penyembunyian informasi), itu tidak benar-benar mendukung pewarisan atau templat, jadi tidak membantu memanfaatkan peluang untuk menggunakan kembali perangkat lunak.

Itu tidak menghasilkan desain yang cocok untuk sistem `` sangat interaktif " - yaitu, sistem di mana pengguna berinteraksi dengan sistem menggunakan mouse serta terminal, sehingga pengguna memiliki kontrol lebih besar atas urutan di mana hal-hal selesai, daripada yang biasanya disediakan oleh sistem yang menggunakan antarmuka `` baris perintah ", atau yang mengandalkan menu sederhana yang dapat digunakan dengan terminal, daripada pada workstation atau PC yang menjalankan sistem jendela. Antarmuka yang lebih sederhana ini (baris perintah, dan yang berbasis menu sederhana) adalah satu-satunya yang digunakan secara luas saat metode desain ini dikembangkan.

Itu tidak menghasilkan desain yang berguna untuk sistem waktu nyata, atau sistem lain di mana persyaratan kinerja memainkan peran yang lebih besar daripada yang dilakukan dalam sistem bisnis yang lebih tradisional. Berasumsi, seperti yang biasanya masih terjadi, bahwa itu cukup (dan karenanya sesuai) untuk merancang dan menerapkan tanpa banyak pertimbangan persyaratan kinerja - dan, setelah itu, akan melakukan tes yang mencakup `` tes kinerja, " 'untuk mencari bukti bahwa kebutuhan sumber daya belum terpenuhi, temukan kemacetan, dan kemudian kodekan ulang (dan mungkin juga desain ulang) jika diperlukan.

Itu tidak mengarah pada desain yang inovatif atau elegan. Teknik seperti ini dapat digunakan untuk menangani 90% (atau lebih) dari sistem yang rutin - seperti kebanyakan sistem yang sama yang memecahkan jenis masalah ini - dan yang

dapat ditangani dengan cukup baik menggunakan metode konvensional yang lugas. . Penggunaan metode semacam ini (bisa dibilang) akan membebaskan lebih banyak waktu untuk bagian-bagian masalah yang memang membutuhkan inovasi atau keahlian.

Namun, Desain Terstruktur memang menyediakan desain yang dapat digunakan dan dipelihara untuk jenis program (dan bahasa pemrograman) yang didukungnya. Selain itu, sistem yang sangat interaktif dan real time sering kali menyertakan subsistem yang dapat dikembangkan menggunakan metode lama ini, jadi mungkin masih berguna.

Akhirnya, beberapa ide yang akan dilihat dengan mempelajari metode ini telah dibawa ke dalam beberapa teknik desain yang lebih modern. Metode yang lebih baru lebih rumit dan belum (belum) dideskripsikan dengan baik sebagai Desain Terstruktur, jadi (orang berharap) ini adalah `` contoh metode desain " yang berguna untuk dipelajari sebelum mempertimbangkan metode yang lebih modern.

Bagan Struktur

Desain Terstruktur dapat digunakan untuk mengubah sekumpulan diagram aliran data dan spesifikasi proses menjadi spesifikasi desain yang disertakan diagram struktur, yang menunjukkan modul sistem dan area data, struktur kontrolnya, dan I / O di antara keduanya; spesifikasi modul untuk setiap modul dalam bagan struktur (serta untuk antarmuka untuk setiap area data).

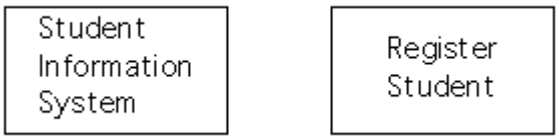
Lebih lanjut akan dikatakan tentang spesifikasi modul nanti; akan dimulai dengan mempertimbangkan komponen bagan struktur.

Modul

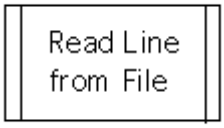
Sebuah modul berhubungan dengan satu fungsi atau prosedur dalam bahasa seperti FORTRAN atau Pascal, atau ke satu fungsi di C atau C ++ (tetapi, tidak secara umum untuk sebuah `‘object " atau" `` class " dalam C ++).

Sebuah modul digambar sebagai persegi panjang, diberi label dengan nama modul - yang juga harus menjadi nama yang wajar untuk fungsi atau prosedur (atau,

kadang-kadang, subsistem) dalam program yang sedang dikembangkan. Berikut dua contoh.

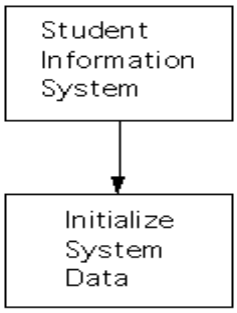


Notasi di atas digunakan untuk merepresentasikan modul untuk prosedur yang perlu dikembangkan sendiri. Ini juga berguna (atau perlu) untuk menampilkan `` modul perpustakaan " yang merupakan bagian dari sistem operasi (atau sistem jendela, atau, lebih umum, perpustakaan perangkat lunak atau perpustakaan) yang mendukung sistem yang sedang kembangkan - yaitu, modul bahwa yang tidak perlu dikembangkan sendiri. Batang vertikal ekstra di setiap sisi persegi panjang digunakan untuk mengidentifikasi ini:



Sebuah Hierarki Kontrol

Bagan struktur mencakup satu modul utama - modul di bagian atas bagan struktur, yang sesuai dengan bagian `` utama " dari program yang sedang diproduksi. Setiap modul terhubung ke semua modul yang dipanggil secara langsung. Koneksi ini ditunjukkan dengan panah yang menunjuk dari modul pemanggil ke modul yang dapat dipanggilnya.

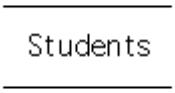


Teknik desain ini tampaknya tidak memperkenalkan fungsi rekursif (dengan sendirinya). Namun, unutk dapat menambahkan ini saat memperbaiki bagan

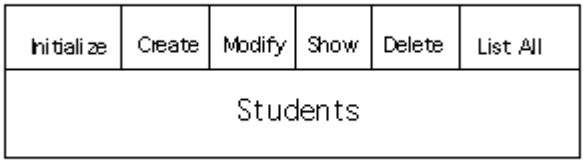
struktur di akhir `` desain arsitektur ", jadi memungkinkan akan menggambar panah yang menghubungkan modul ke modul itu sendiri, dll.

Area Data

Bagan struktur juga menyertakan representasi dari `` area data. " Dalam bagan struktur `` draf pertama " yang didapatkan di awal desain terstruktur, akan menggambar area data dengan mencantumkan nama area data di antara dua horizontal garis:



Namun, saat bagan struktur telah selesai, akan mengidentifikasi sekumpulan fungsi antarmuka yang digunakan untuk semua komunikasi antara area data dan sistem lainnya - yaitu, harus menyelesaikan setidaknya langkah awal desain data . Pada titik itu, area data (juga sekarang disebut cluster informasional) akan digambar sebagai berikut:



Kotak di bagian atas diberi label dengan nama masing-masing fungsi antarmuka yang dapat digunakan oleh seluruh sistem untuk mengakses (atau mengubah) area data yang sekarang ditampilkan di bawahnya.

Dalam bagan struktur `` draf " atau `` potongan pertama ", akan menggambar panah dari masing-masing modul (di seluruh sistem) yang dapat mengakses area data, turun ke garis horizontal atas di representasi dari area data. Dalam bagan struktur `` selesai ", masing-masing modul ini (dari sistem lainnya) harus memanggil satu atau lebih fungsi `` antarmuka " - jadi akan menggambar panah ke

bawah ke kotak (atau boks) untuk fungsi antarmuka (atau fungsi antarmuka) yang dipanggil oleh modul.

Tidaklah masuk akal untuk menunjukkan panah yang naik dari area data atau klaster informasi ke modul lain dalam sistem, karena (biasanya terjadi) bahwa `` area data " bersifat pasif, dan tidak dapat memanggil (atau, oleh karena itu, `` mengontrol ") hal lain.

Menggunakan notasi `` cluster informasi " ini untuk mengelompokkan sekumpulan `` fungsi antarmuka " yang secara kolektif menyediakan akses ke perangkat I / O, sehingga bagian dari sistem yang berkomunikasi langsung dengan perangkat itu - dan perlu dimodifikasi jika perangkat itu diganti - dikelompokkan bersama, dan mudah ditemukan.

Aliran data

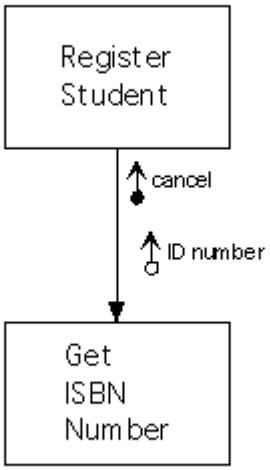
Bagan struktur juga menunjukkan data yang dapat dikirim antar modul (atau antara modul dan fungsi antarmuka untuk area data atau perangkat I / O). Dua simbol digunakan untuk item data - satu untuk item data `` biasa ", dan satu lagi untuk `` sinyal kontrol " - yang dapat anggap sesuai dengan seseorang yang menekan tombol (dan yang akan digambar pada aliran data tambahan diagram menggunakan panah putus-putus, seperti yang dijelaskan sebelumnya.

Item `` biasa " digambar menggunakan panah pendek, dengan lingkaran berongga di salah satu ujung panah (dan kepala panah di ujung lainnya). Ini diberi label dengan nama item data.

Sebuah `` sinyal kontrol " digambar menggunakan simbol yang hampir sama: Lingkaran di ujung panah diisi, bukan berlubang. Itu diberi label dengan nama sinyal kontrol.

Aliran data ditarik dekat dengan ("mengalir bersama ") koneksi antar modul, atau antara modul dan fungsi antarmuka, dan setiap titik dari modul atau fungsi

mengirimkan data ke modul atau fungsi yang menerimanya. Masing-masing koneksi ini antara modul sesuai dengan panggilan prosedur (mungkin) - jadi biasanya akan memiliki data mengalir ke modul yang dipanggil yang sesuai dengan setiap input yang disediakan oleh modul yang memanggilnya, dan akan memiliki aliran data kembali dari modul yang dipanggil sesuai dengan setiap keluaran yang mungkin dikembalikan oleh modul yang dipanggil.



Kekurangan) Terminator

Bagan struktur umumnya tidak mencakup apa pun yang berhubungan langsung dengan `` Terminator " pada diagram aliran data - yaitu, dengan orang atau sistem lain yang berinteraksi dengan sistem yang dikembangkan. Ini juga umumnya tidak termasuk aliran data apa pun antara sistem dan terminator ini.

Konektor Off Page

Ada satu simbol lagi yang bukan `` standar " - mungkin tidak menemukannya di buku Page-Jones - tetapi itu tampaknya perlu (dan itu, pada kenyataannya, simbol diagram alir standar) - Off Konektor Halaman. "

Jika telah kehabisan ruang di bagian bawah halaman, alih-alih memiliki panah ke bawah ke modul lain yang akan dipanggil, gambar panah ke bawah ke bagian atas salah satu konektor ini. Beri label konektor `` To (nama modul panggil) " dengan menuliskan pesan ini di dalamnya, dan cantumkan nomor halaman yang akan berisi modul yang dipanggil, di samping konektor. Menampilkan data yang mengalir ke dan dari modul yang dipanggil di samping panah ke konektor, seperti yang dilakukan jika data tersebut menunjuk langsung ke modul.

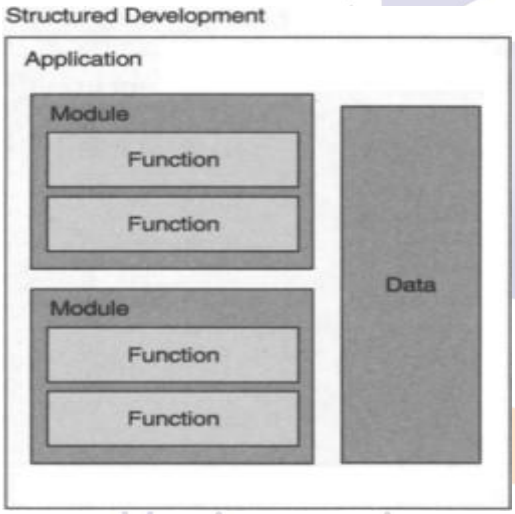
Pada halaman yang berisi modul yang dipanggil, tampilkan konektor off page yang cocok di bagian atas. Beri label ini dengan `` Dari (modul yang melakukan panggilan) " dan cantumkan nomor halaman dari modul yang memanggil modul ini di samping konektor. Gambar panah dari titik bawah konektor ke modul yang dipanggil dan reproduksi data yang mengalir dari dan kembali ke modul pemanggil di sepanjang koneksi ini.

Dengan pengecualian yang mungkin dari penambahan `` konektor off page " ini, notasi yang dijelaskan di sini sama persis dengan notasi yang digunakan oleh Page-Jones, dalam bukunya tentang Structured Design. Menemukan beberapa contoh bagan struktur dalam buku itu, jika ingin melihat yang lengkap sekarang. Beberapa di antaranya akan muncul di halaman catatan online berikutnya.

Desain Terstruktur terdiri dari dua langkah utama:
Menghasilkan Bagan Struktur `` Potongan Pertama "
Evaluasi dan perbaiki bagan struktur hingga diperoleh bagan struktur yang dapat digunakan
Langkah pertama dimulai dengan diagram aliran data dan terdiri dari banyak keputusan dan tes kecil. Langkah kedua menyediakan cara untuk meningkatkan hasil dari langkah pertama (yang biasanya tidak diharapkan sangat baik). Karena langkah kedua memang ada, tidak perlu terlalu khawatir tentang cara membuat beberapa keputusan pada langkah pertama jika tidak jelas - akan memiliki kesempatan untuk memperbaiki bagan struktur yang dibuat dengan membuat keputusan ini nanti

Desain Terstruktur

Desain dan pengembangan terstruktur menurut Yourdon dan Constantine, digambarkan pada Gambar dibawah ini, melibatkan penguraian proses yang lebih besar menjadi proses yang lebih kecil. Desainer memecah proses yang lebih besar menjadi proses yang lebih kecil untuk mengurangi kompleksitas dan meningkatkan penggunaan kembali. Desain terstruktur membahas porsi perilaku sistem perangkat lunak secara terpisah dari porsi data. Memecah struktur program membantu mengembangkan aplikasi yang lebih kompleks, tetapi mengelola data dalam aplikasi itu sulit, karena fungsi yang berbeda bekerja pada sebagian besar data yang sama.



Menurut Parnas bahwa Pengembangan terstruktur membantu menyembunyikan informasi tentang struktur dan proses program, tetapi tidak menyembunyikan detail data di dalam program. Prinsip desain standar yang dikenal sebagai penyembunyian informasi melibatkan pembatasan pengetahuan yang dimiliki satu bagian program tentang bagian lain. Ini termasuk data, format data, struktur internal, dan proses internal. Pengembangan berorientasi objek memungkinkan pengembang untuk menyembunyikan perilaku program dan data di dalam objek.

Structured design :

Analisis transaksi, membagi sistem menjadi unit-unit yang dapat diatur.

Transaksi memiliki lima komponen dasar:

1. peristiwa di lingkungan sistem yang menyebabkan

transaksi terjadi

2. stimulus yang diterapkan pada sistem untuk menginformasikannya tentang acara tersebut

3. aktivitas yang dilakukan oleh sistem sebagai hasil dari stimulus

4. respon yang dihasilkan dalam bentuk output dari sistem

5. efek ini terhadap lingkungan sistem

Analisis transformasi, mengubah unit menjadi bagan struktur.

Analisis transformasi

• Tujuannya adalah untuk menemukan fungsi inti modul:

- ambil model DFD dari masalah dan ubahlah ke dalam struktur hierarki

1. Pertama, identifikasi transformasi pusat di DFD:

• Transformasi pusat terletak di tengah input dan arus data keluaran

• terkadang transformasi sentral harus dibuat, dengan menambahkan gelembung ekstra

2. DFD 'diambil' oleh transformasi pusat sehingga fungsi lain tergantung darinya

• Transformasi pusat membentuk badan utama modul

3. Bagan Struktur potongan pertama dibentuk

• Gelembung diganti dengan balok

• Panah digambar ulang untuk menunjukkan permintaan:

- menjelaskan aliran data di DFD, tetapi menunjukkannya aliran kontrol (panggilan prosedur) dalam Bagan Struktur

• Arus data ditambahkan sebagai panah samping (pasangan data)

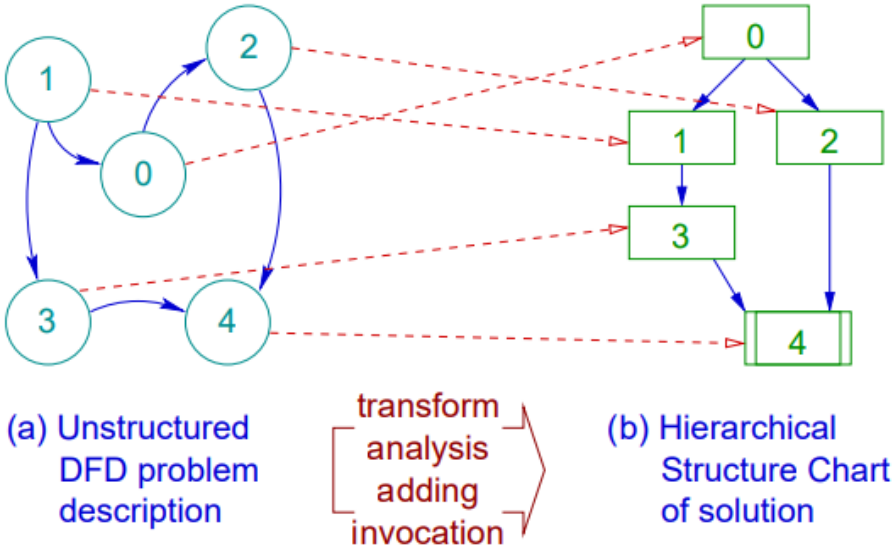


Illustration of transformation from a simple DFD to a hierarchical structure chart describing the solution.

Analisis transformasi

• Transformasi awal membuat pemetaan proses satu-ke-satu gelembung ke subprogram

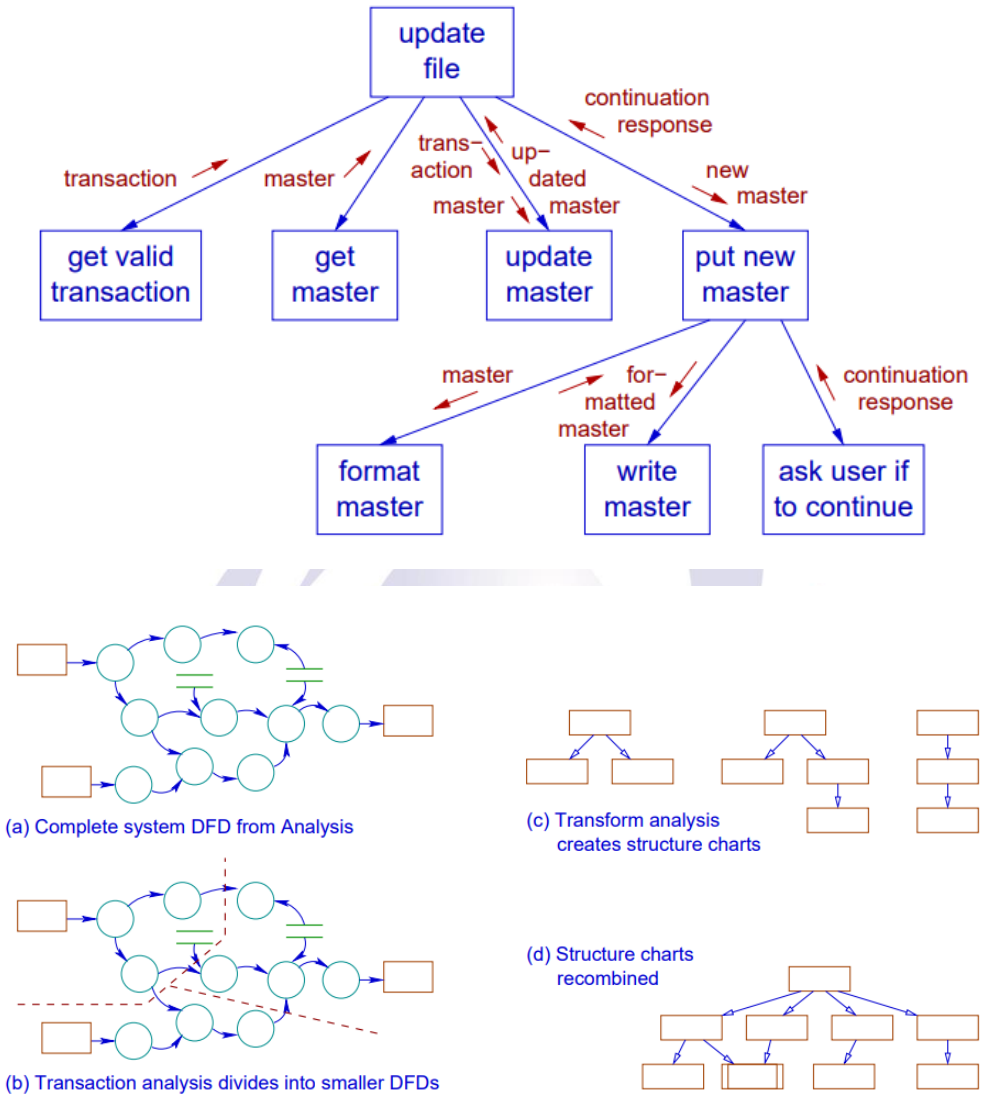
• Transformasi pusat dipilih menjadi yang paling abstrak gelembung proses

• Gelembung abstrak terkecil cenderung paling dekat dengan I / O - dipetakan ke tingkat yang lebih rendah dari Bagan Struktur

• Perlu membuat transformasi pusat untuk prosedur itu mengatur urutan panggilan ke subprocedures (yang tidak jelas dari DFD tingkat tunggal)

• Bagan Struktur potongan pertama membutuhkan penyempurnaan lebih lanjut menghasilkan desain yang bagus

Integrasi system, menggabungkan kembali grafik dengan menghubungkan unit bersama.



Transformations of the DFD from analysis into a recombined structure chart via smaller transaction DFDs & their charts.

C. Latihan

1. Metodologi yang sistematis untuk menentukan spesifikasi desain perangkat lunak.,disebut ?

2. Ukuran sejauh mana elemen-elemen modul terkait secara fungsional., disebut ?

D. Kunci Jawaban

1. Structured Design
2. *Cohesion*

E. Daftar Pustaka

1. Roger S. Pressman, Software Engineering A Practioner's Apporach, 2014
2. Ian Sommerville, Software Engineering (10th Edition), 2015
3. https://www.oreilly.com/library/view/software-engineering/9788131758694/xhtml/chapter006.xhtml
4. http://pages.cpsc.ucalgary.ca/~eberly/Courses/CPSC333/Lectures/Design/sd_intro.html
5. https://www.sciencedirect.com/topics/computer-science/structured-design
6. https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/
7. http://info.ee.surrey.ac.uk/Teaching/Courses/ee2.sad/s16sad.pdf