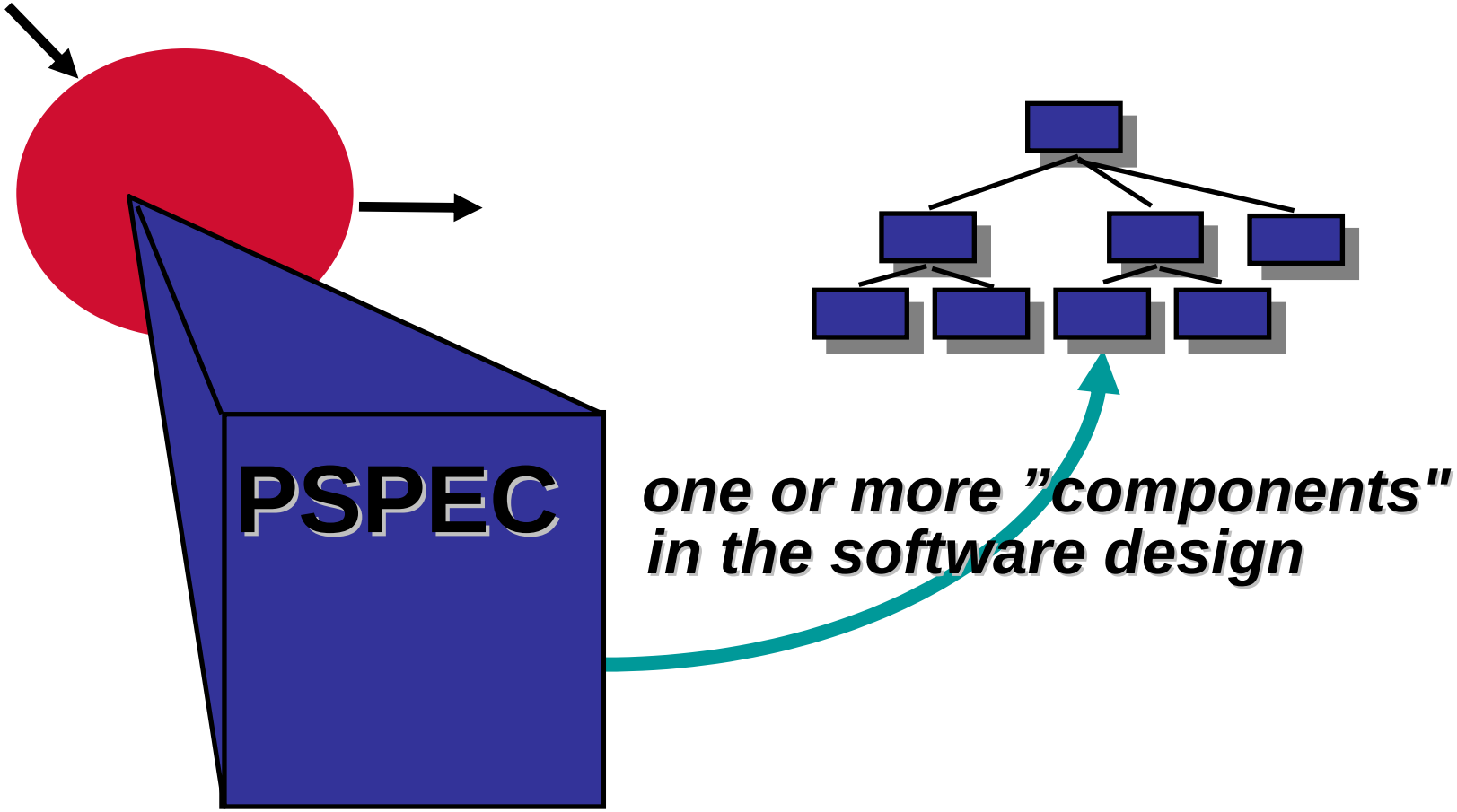
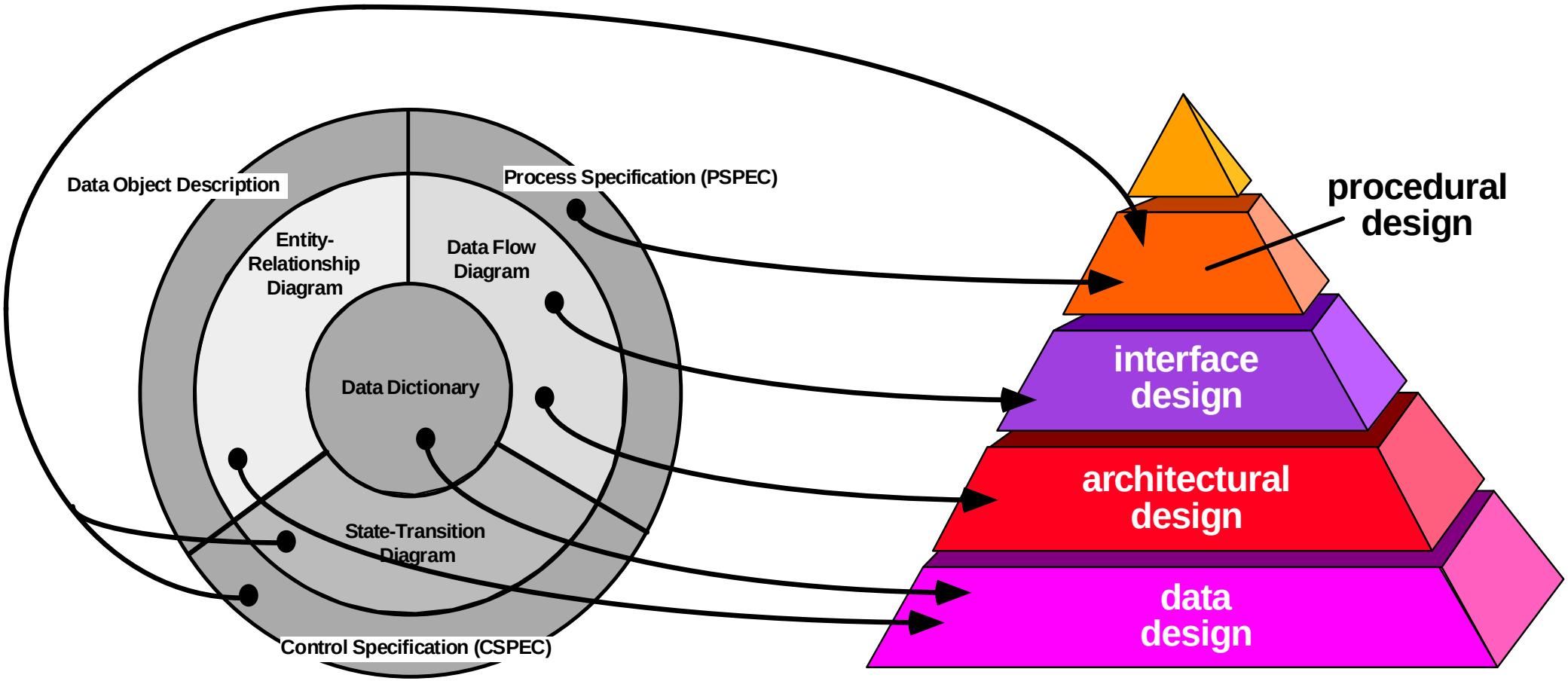
Software Design

- Konsep dan Prinsip Desain
- Struktur Desain

Catatan dari Sebuah Desain



Analisis Ke Desain



THE ANALYSIS MODEL

Mira/Rpl/Design

THE DESIGN MODEL

Proses Desain

- Desain P/L adalah suatu proses Interaktif yang melalui persyaratan diterjemahkan ke dalam "*blue print*" untuk membangun suatu perangkat lunak.
- Sepanjang proses desain, kualitas desain yang melengkapi dinilai dengan serangkaian kajian teknis formal atau desain awal.
- Pedoman bagi evaluasi suatu desain yang baik :
  - Desain harus mengimplementasi keseluruhan persyaratan eksplisit yang dibebankan dalam model analisis, dan harus mengakomodasi semua persyaratan implisit yang diinginkan pelanggan.
  - Desain harus menjadi panduan yang dapat dibaca.
  - Desain harus memberikan suatu gambaran lengkap mengenai P/L.

Evolusi dari Desain P/L

- Pengembangan program Modular
- Pemrograman Terstruktur
  - Aspek-aspek prosedural dari definisi desain
- Menterjemahkan aliran data atau struktur data kedalam definisi desain
- OO design

Prinsip Desain

- Proses Desain tidak boleh menderita karena “tunnel vision” → Harus memperhatikan pendekatan-pendekatan alternatif.
- Desain harus dapat ditelusuri sampai model analisis
- Desain tidak boleh berulang → gunakan pola desain
- Design harus “meminimalkan kesenjangan intelektual” antara P/L dan masalah yang ada di dunia nyata.
- Desain harus mengungkapkan keseragaman dan integrasi
- Desain harus terstruktur untuk mengakomodasi perubahan
- Desain harus terstruktur untuk berdegradasi dengan baik, bahkan pada saat data dan *event-event* menyimpang atau menghadapi kondisi operasi
- Desain bukanlah pengkodean dan pengkodean bukanlah desain.
- Desain harus dinilai kualitasnya pada saat desain dibuat, bukanlah setelah jadi.
- Desain haruslah dinilai kualitasnya pada saat desai dibuat, bukan setelah jadi.
- Desain harus dikaji untuk meminimalkan kesalahan-kesalahan konseptual (semantik)

Fundamental Concepts

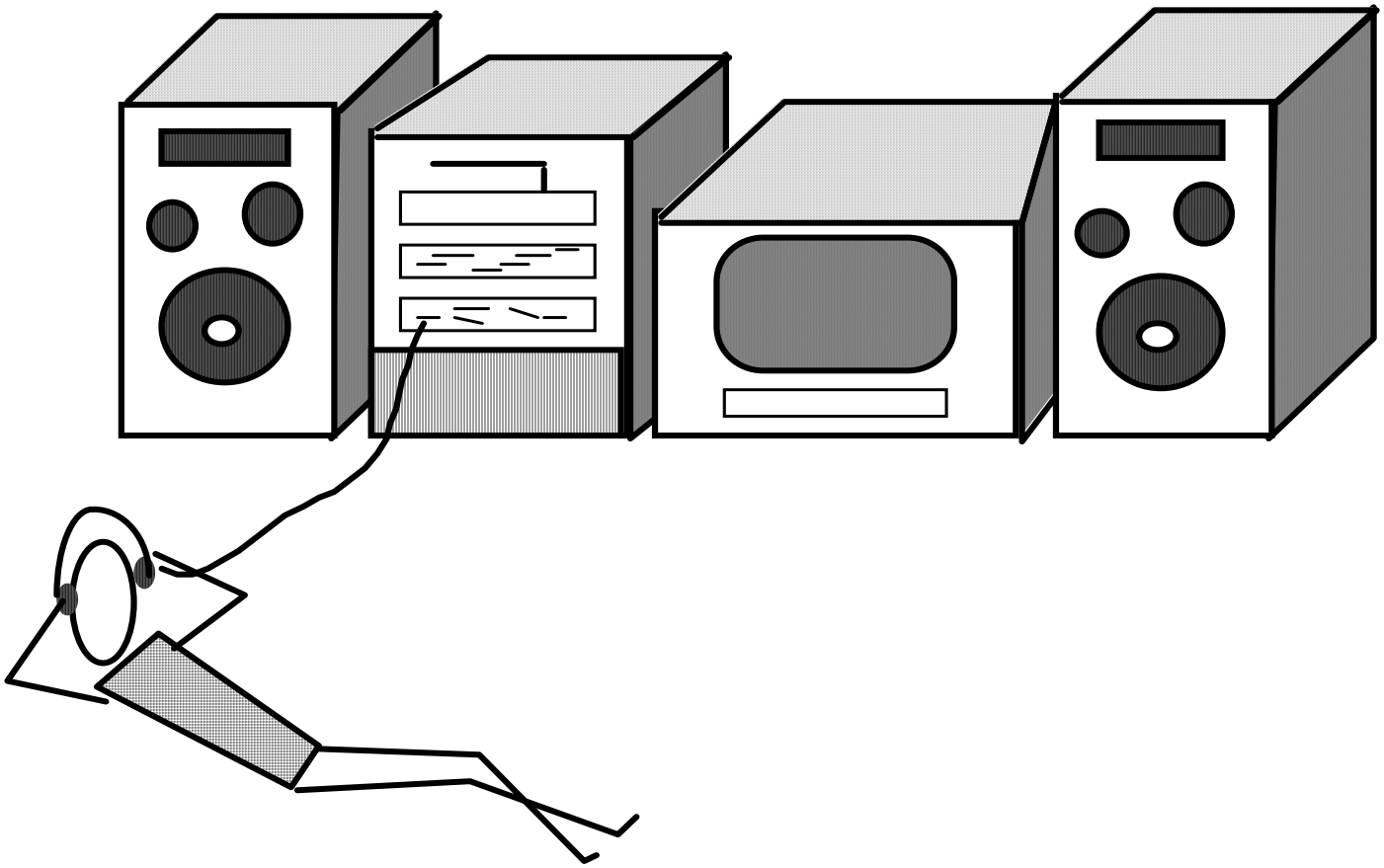
- **Abstraksi/Abstraction** - memungkinkan designers untuk berkonsentrasi pada suatu permasalahan pada beberapa tingkat generalisasi tanpa memperhatikan detail tingkat rendah yang tidak relevan (Abstraksi prosedural – urutan instruksi yg diberi nama yg mempunyai fungsi tertentu dan terbatas, abstraksi data – kumpulan data yg bernama yg menggambarkan objek data)
- **Penyaringan/Refinement** – proses elaborasi, dimana " the designer" berhasil membuat detail dari setiap komponen yang didesain.
- **Modularitas/Modularity** – di mana perangkat lunak dibagi ke dalam komponen-komponen bernama dan dapat dipanggil terpisah.
- **Arsitektur/P/LSoftware architecture** – struktur keseluruhan perangkat lunak dan cara di mana struktur memberikan integrasi konseptual bagi sistem.

Fundamental Concepts (2)

- **Hirarki Kontrol/Control hierarchy or program structure** – merepresentasikan organisasi komponen program (modul) serta mengimplikasikan suatu hirarki kontrol, tidak mengimplikasikan aspek prosedural dari P/L, seperti urutan proses, kejadian dari keputusan, atau pengulangan operasi.
- **Partisi Struktural/Structural partitioning** – partisi horisontal menentukan cabang-cabang terpisah dari hirarki modular untuk setiap fungsi program mayor(input, transformasi data, dan output); partisi vertikal (Pemfaktoran) menyatakan bahwa kontrol dan kerja harus didistribusikan secara top-down dalam arsitektur program
- **Struktur Data/Data structure** – representasi dari hubungan logis antara elemen-elemen data individual
- **Prosedur P/L/Software procedure** - precise specification of processing (event sequences, decision points, repetitive operations, data organization/structure)
- **Penyembunyian Informasi/Information hiding** - informasi (data and procedure) yang diisikan pada sebuah modul tidak dapat diakses ke modul alin yang tidak memiliki kepentingan terhadap informasi tersebut.

Modular Design

easier to build, easier to change, easier to fix ...



Independensi Fungsional

COHESION /KOHESI

Ukuran kekuatan fungsional relatif dari sebuah modul

COUPLING/PERANGKAIAN

Ukuran kesalingtergantungan relatif di antara modul-modul

Cohesion and Coupling Spectrum

Rendah

- Koinsidental
- Logikal
- Temporal
- Prosedural
- Komunikasional
- Sekuensial
- Fungsional

Tinggi

Rendah

- Tidak ada perangkaian langsung
- Perangkaian Data
- Perangkaian Melekat
- Perangkaian Kontrol
- Eksternal
- Perangkaian Umum
- Perangkaian Isi

Tinggi

Why Information Hiding?

- Menyatakan bahwa modul ditandai dengan keputusan desain (masing-masing) tersebunyi dari desain lain
- Modul seharusnya ditentukan dan didesain sehingga informasi informasi yang diisikan pada sebuah modul tidak dapat diakses ke modul lain yang tidak memiliki kepentingan terhadap informasi tertentu
- Menetapkan dan menyelenggarakan batasan-batasan akses ke detail prosedural pada suatu modul dan struktur data lokal yang digunakan oleh modul tersebut
- Suatu kriteria desain untuk sistem modular memberikan keuntungan terbesarnya pada saat dibutuhkan modifikasi selama pengujian dan sesudahnya, selama pemeliharaan P/L
- Kesalahan kecil yang terjadi selama modifikasi punya kemungkinan lebih kecil untuk menyebar ke lokasi lain dalam P/L

Why Architecture?

Arsitektur Perangkat lunak bukan menggambarkan P/L Operasional melainkan merepresentasikan struktur keseluruhan P/L cara di mana struktur memberikan integrasi konseptual bagi suatu sistem . Kenapa harus dibuat :

(1) Mendapatkan gambaran arsitektural sebuah sistem.

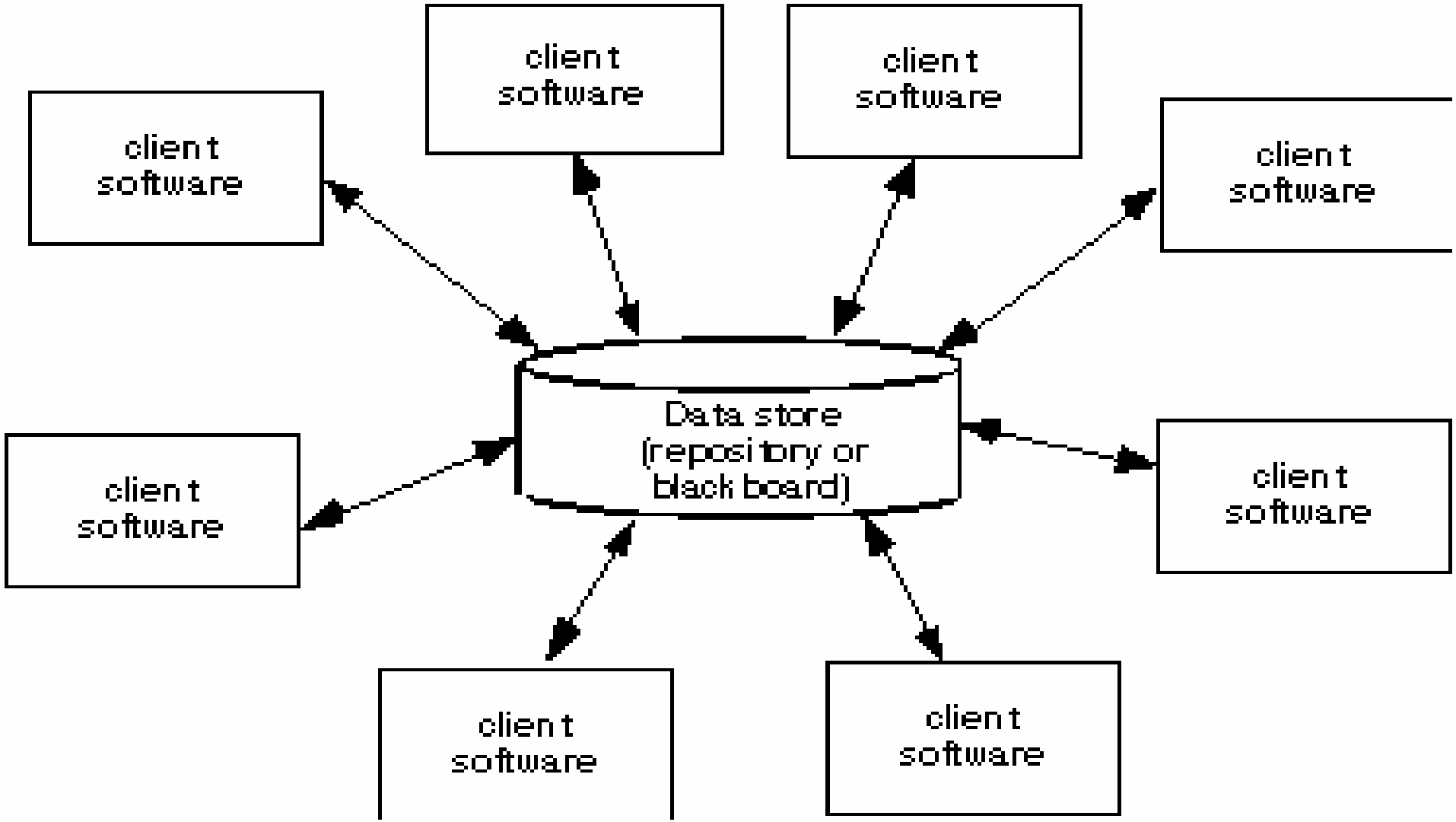
(2) Gambaran tersebut berfungsi sebagai kerangka kerja dari aktivitas desain yang lebih detail ang dilakukan.

(3) Mengurangi resiko dalam pembangunan P/L

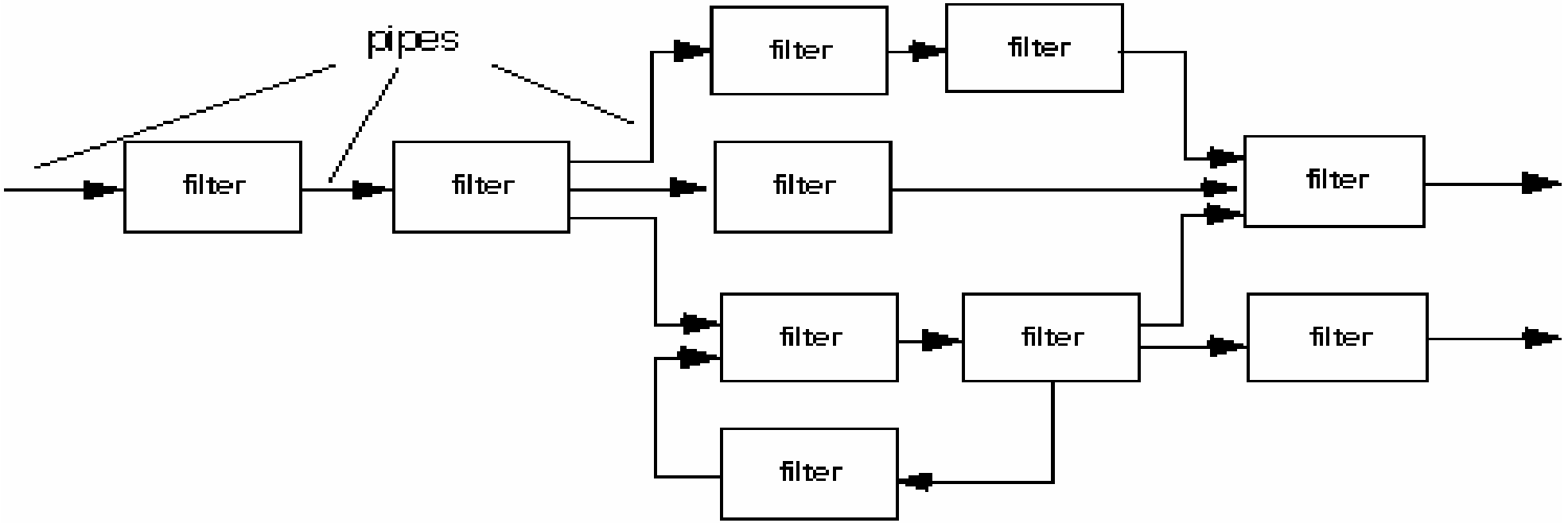
Architectural Styles

- Data-centered architectures
- Data flow architectures
- Call and return architectures
- Object-oriented architectures
- Layered architectures

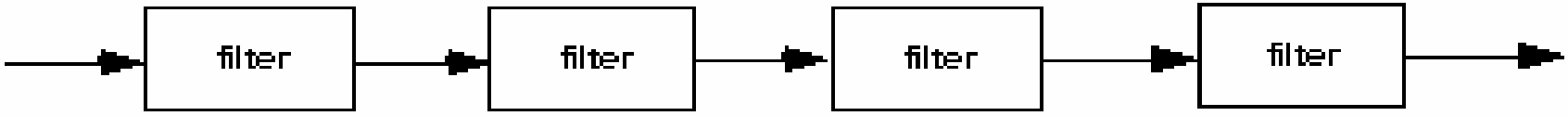
Data-Centered Architecture



Data Flow Architecture

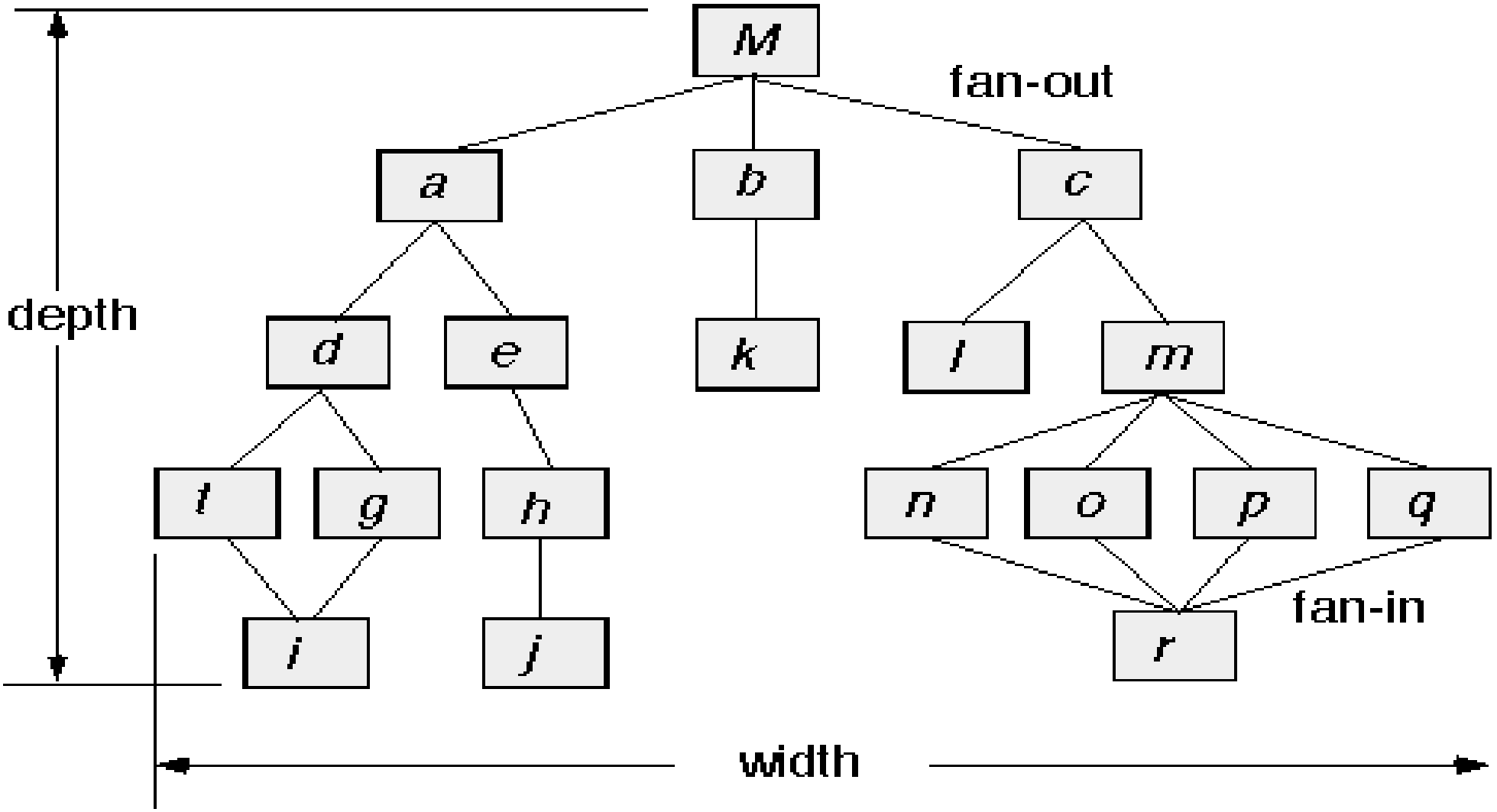


(a) pipes and filters

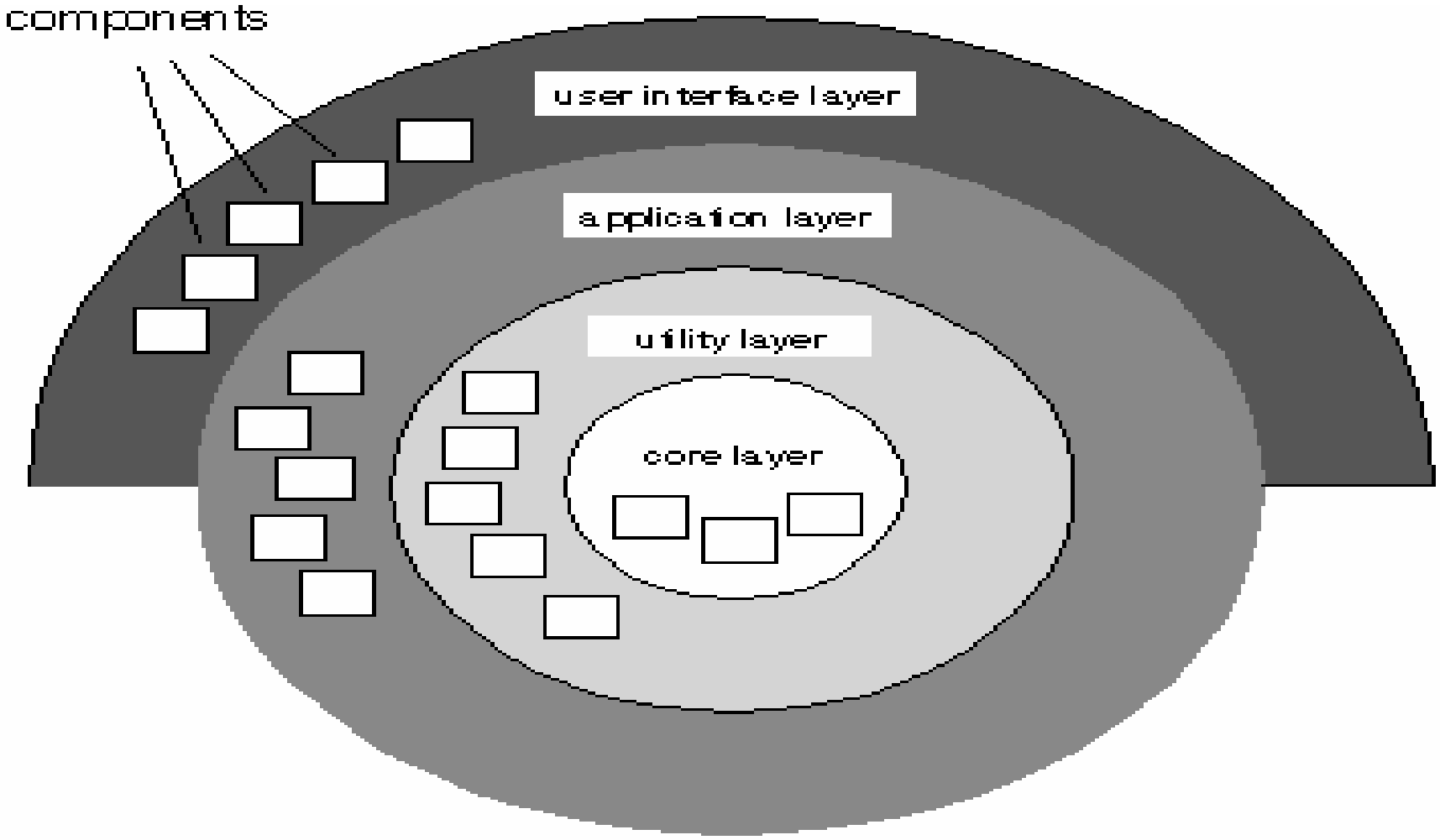


(b) batch sequential

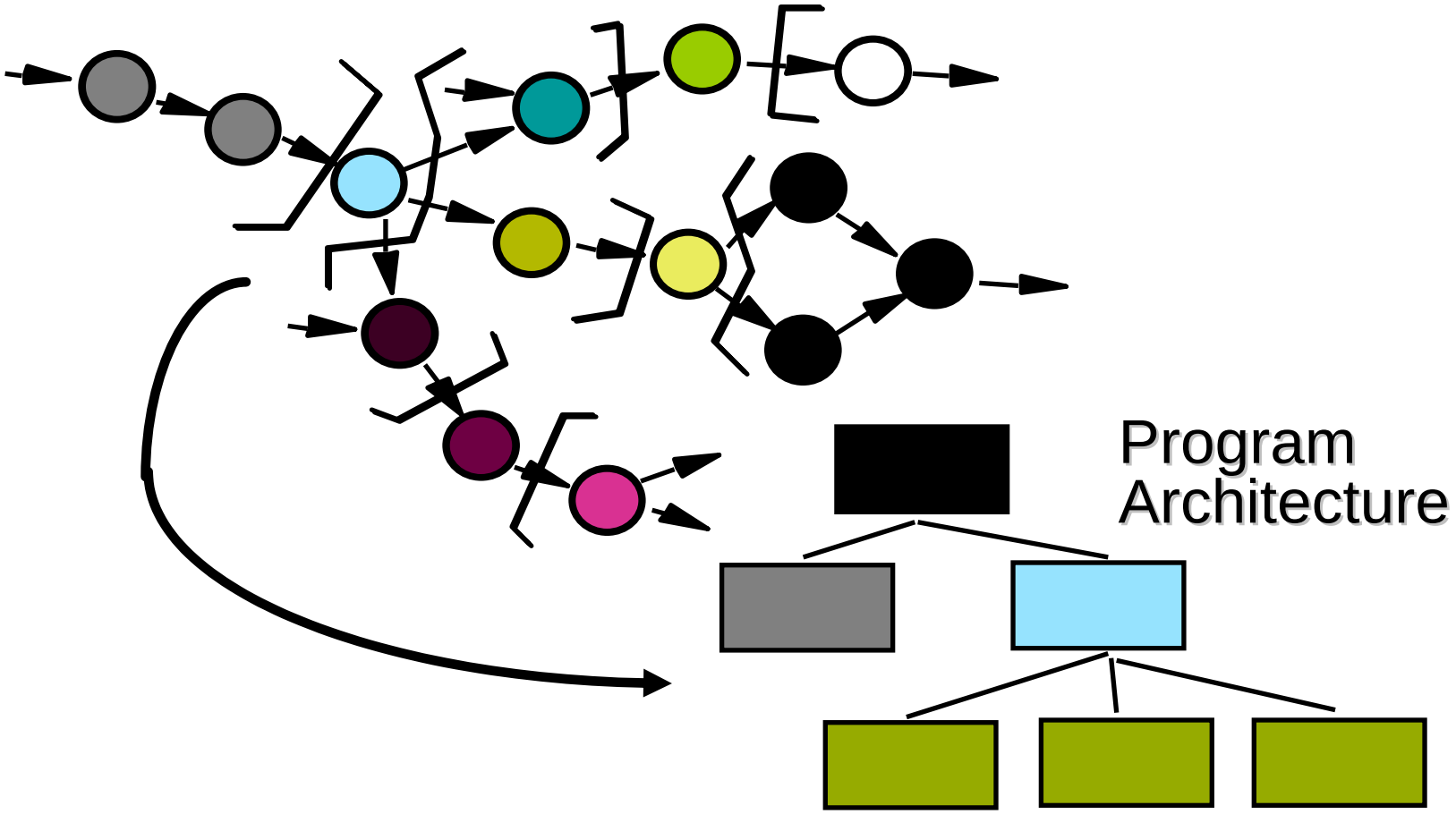
Call and Return Architecture



Layered Architecture

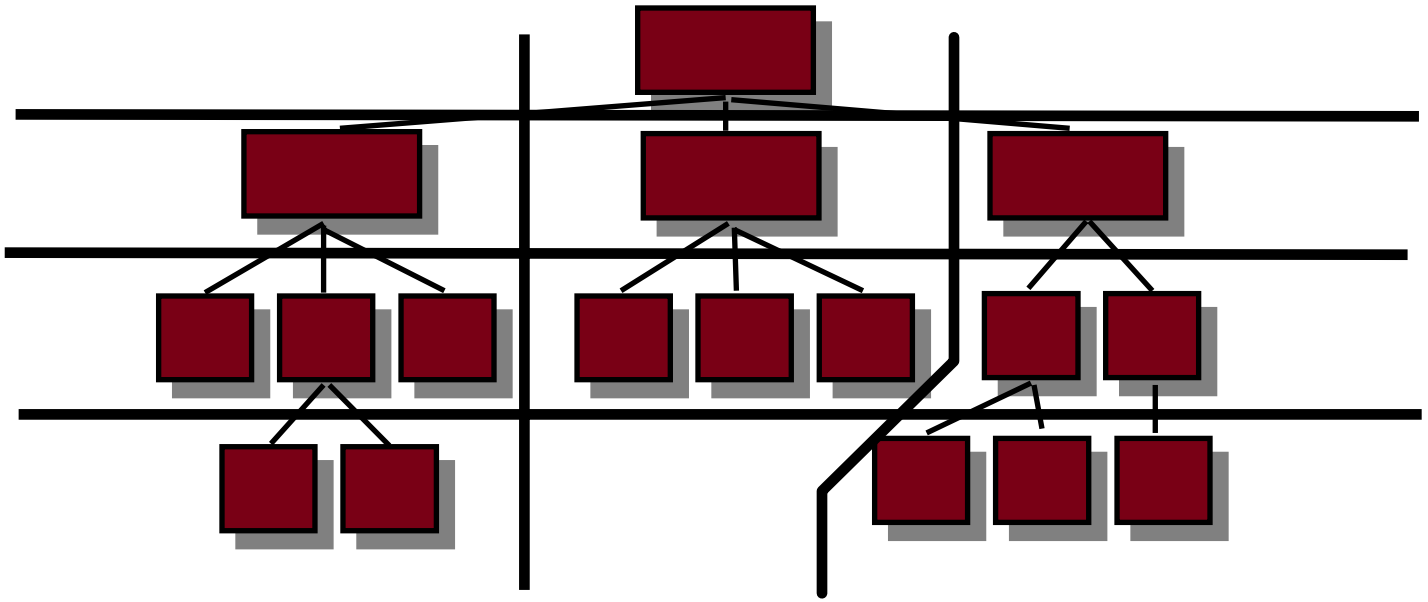


Deriving Program Architecture



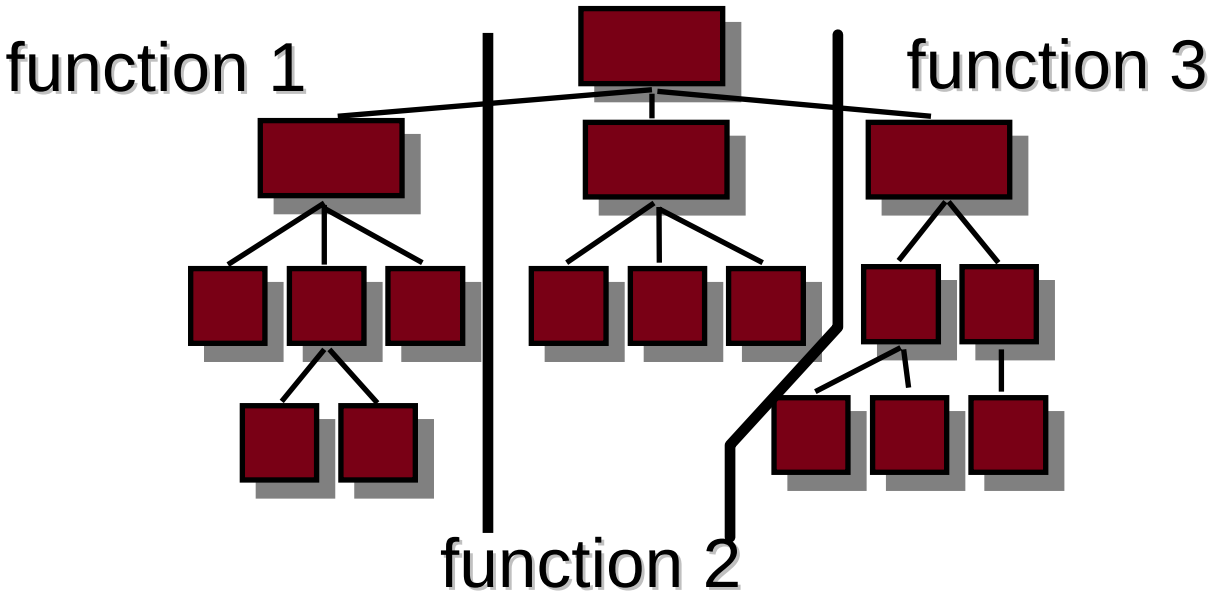
Partitioning the Architecture

- “horizontal” and “vertical” partitioning are required



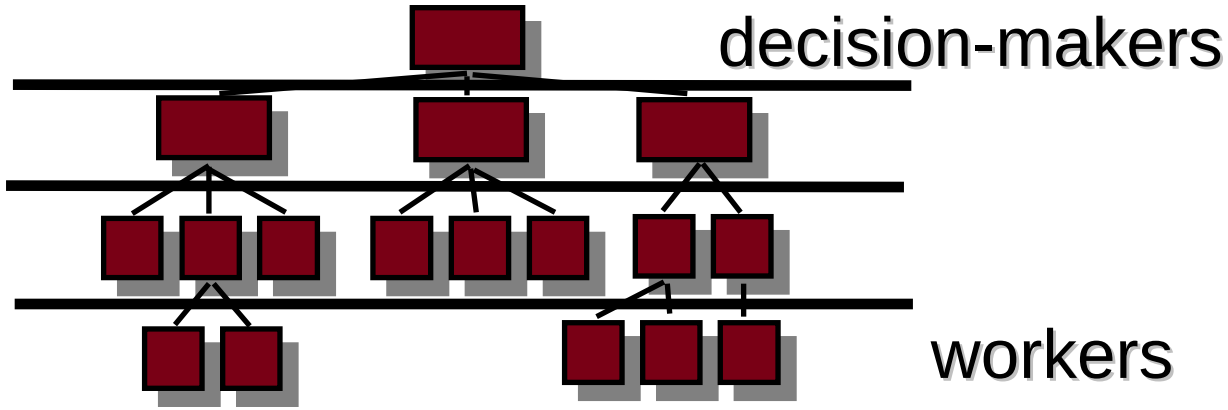
Horizontal Partitioning

- Menentukan cabang-cabang terpisah dari hirarki modular untuk setiap fungsi program mayor.
- Modul kontrol digunakan untuk mengkoordinasi komunikasi di antara dan eksekusi fungsi-fungsi program



Vertical Partitioning: Factoring

- Pembuatan keputusan dan kerja harus didistribusikan secara top-down dalam arsitektur program.
- Modul-modul tingkat puncak harus melakukan fungsi-fungsi kontrol dan melakukan sedikit kerja pemrosesan aktual.



Why Partitioned Architecture?

- Menghasilkan P/L yang lebih mudah diuji.
- Membawa kepada P/L yang lebih mudah dipelihara
- Menghasilkan penyebaran efek samping yang lebih sedikit
- Menghasilkan P/L yang lebih mudah untuk diperluas

Heuristik Desain Bagi Modularitas Yang Efektif

- Evaluasi iterasi pertama dari struktur program untuk mengurangi perangkaian/coupling dan meningkatkan kohesi/cohesion.
- Usahakan meminimalkan struktur dengan fan-out tinggi, usahakan untuk melakukan fan-in pada saat kedalaman bertambah
- Jagalah supaya lingkup efek dari suatu modul ada dalam lingkup kontrol dari modul itu.
- Evaluasi interface modul untuk mengurangi kompleksitas dan redundansi dan meningkatkan konsistensi.
- Tetapkan modul-modul fungsinya dapat diprediksi, tetapi hindari modul yang terlalu rekstriktif.
- Usahakan modul-modul “entri terkontrol” dengan menghindari “hubungan patologis”.
- Kemaslah P/L berdasarkan batasan desain dan persyaratan.