Arsitektur Perangkat Lunak

Team Teaching Mata Kuliah Rekayasa Perangkat Lunak
Jurusan Teknologi Informasi

Politeknik Negeri Malang

Dosen Pengampu: Wilda Imama Sabilla, S.Kom., M.Kom.

Tujuan

- Mengetahui pentingnya arsitektur pada perangkat lunak.
- Mengetahui pola arsitektur pada perangkat lunak.

Desain Arsitektur Perangkat Lunak

- Desain arsitektur berkaitan dengan pemahaman bagaimana suatu sistem harus diatur dan merancang keseluruhan struktur sistem tersebut.
- Desain arsitektur menghubungkan antara proses desain dan spesifikasi, dengan mengidentifikasi komponen struktural utama dalam suatu sistem dan hubungan komponen-komponen tersebut.

Arsitektur Perangkat Lunak

- Arsitektur perangkat lunak adalah struktur atau kerangka kerja yang mendefinisikan bagaimana komponen dalam sebuah sistem perangkat lunak berinteraksi satu sama lain.
- Arsitektur perangkat lunak berperan sebagai peta untuk membangun, memelihara, dan mengembangkan perangkat lunak.
- Arsitektur perangkat lunak memiliki tiga elemen utama:
  1. **Komponen**: Unit yang melakukan tugas-tugas spesifik (contoh: modul, kelas, atau layanan).
  2. **Konektor**: Alur komunikasi antar komponen (contoh: API, protokol jaringan).
  3. **Konfigurasi**: Struktur yang mengatur hubungan antara komponen dan konektor.

Mengapa Arsitektur PL Penting?

- Rancangan arsitektur dapat digunakan untuk mengkomunikasikan perangkat lunak kepada pihak-pihak yang terkait yaitu anggota tim pengembang / developer.
- Arsitektur sistem menentukan keberhasilan sistem maupun menentukan pekerjaan Rekayasa Perangkat Lunak selanjutnya.
- Arsitektur menggambarkan ilustrasi model intelektual suatu sistem yang distrukturkan untuk mudah dipahami

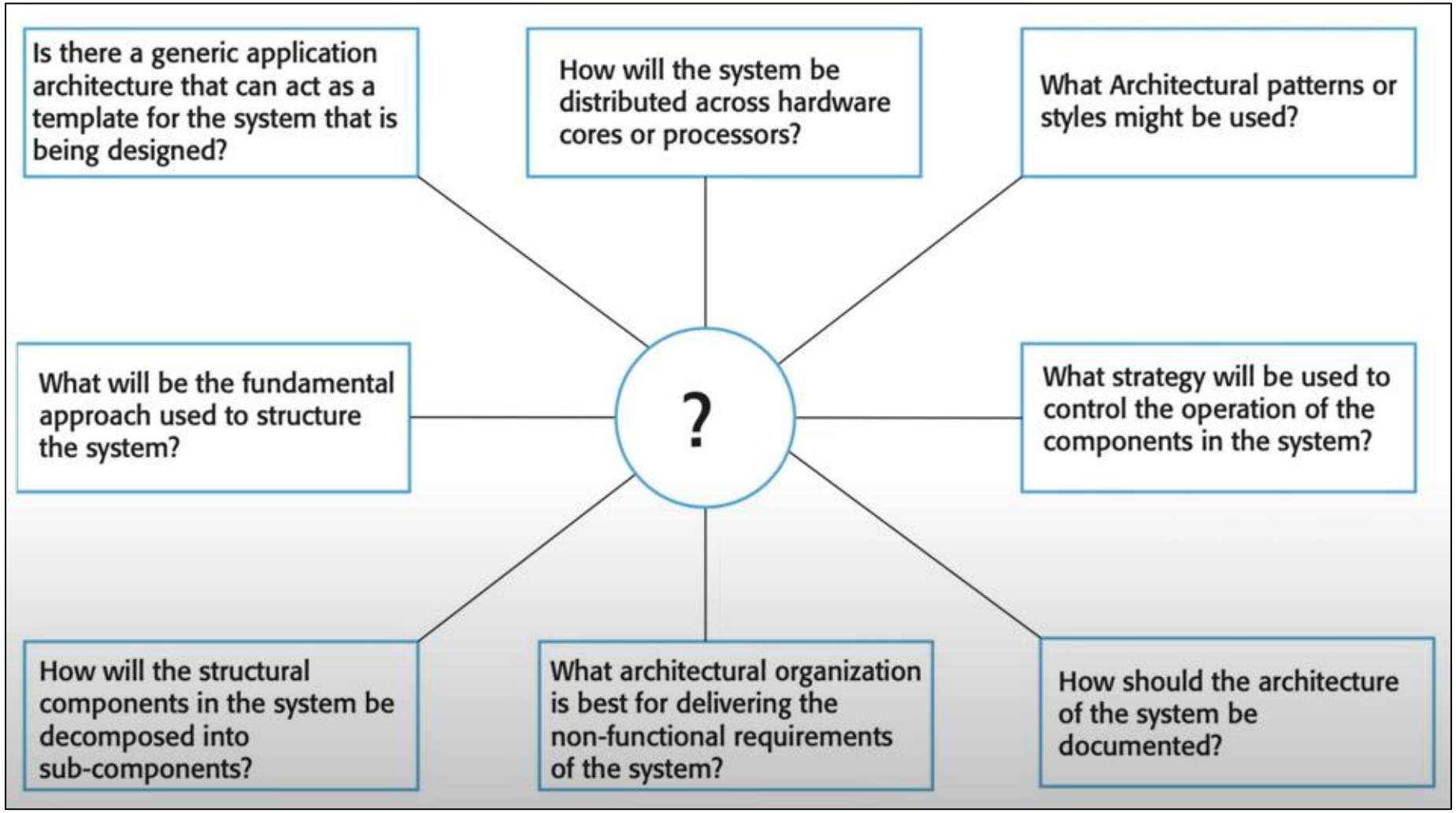
Pemilihan Arsitektur PL

Pemilihan jenis arsitektur bergantung pada:

- Ukuran dan kompleksitas proyek.
- Persyaratan non-fungsional (misalnya kinerja, skalabilitas, keandalan).
- Anggaran dan waktu pengembangan.
- Ketersediaan teknologi dan keahlian tim.

Arsitektur perangkat lunak yang tepat akan memastikan keberhasilan proyek perangkat lunak baik dalam jangka pendek maupun jangka panjang.

Contoh diskusi dalam pencapaian Desain Arsitektur



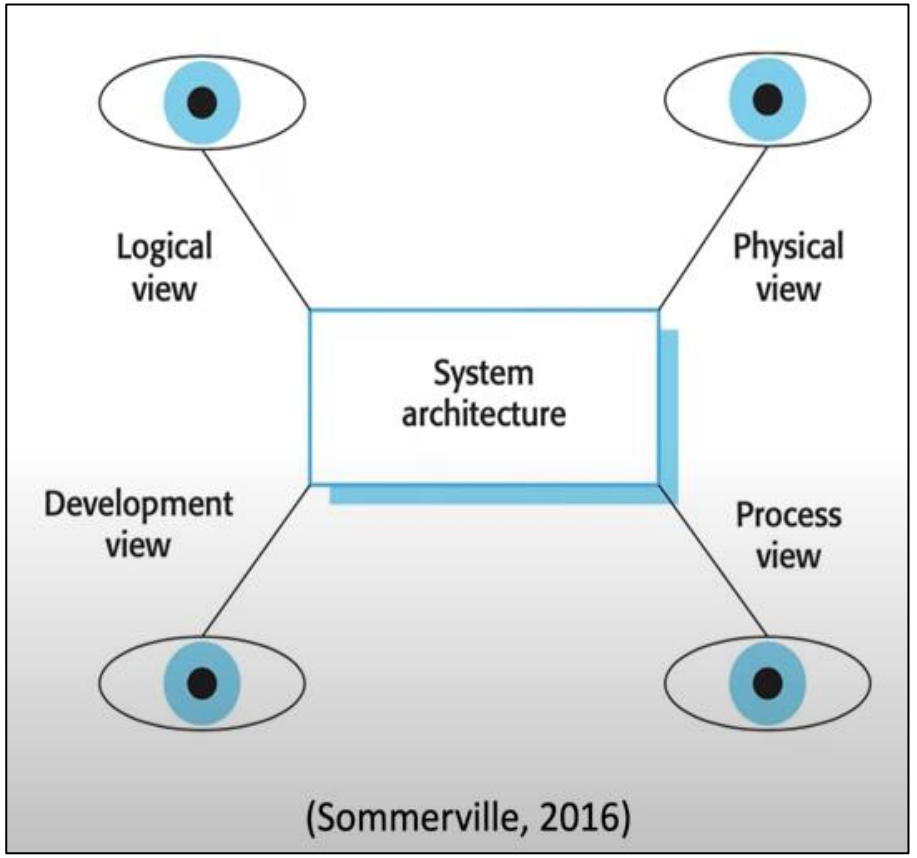
Architectural View

- Mustahil untuk menyajikan semua informasi yang relevan tentang arsitektur sistem dalam satu model arsitektur, karena setiap model hanya menunjukkan satu tampilan atau perspektif sistem.
- Model tersebut mungkin menunjukkan bagaimana sistem diurai menjadi modul, bagaimana proses run-time berinteraksi, atau berbagai cara komponen sistem didistribusikan melalui jaringan.
- Sehingga untuk desain maupun dokumentasi, biasanya perlu menyajikan beberapa tampilan arsitektur perangkat lunak.
- Setiap **view** berfokus pada aspek tertentu dari sistem perangkat lunak, seperti struktur, alur data, atau hubungan antar komponen. Dengan menggunakan architectural view, pengembang dapat memahami, mengomunikasikan, dan memvalidasi desain perangkat lunak secara lebih efektif.

Architectural View

1. Logical View
Menunjukkan abstraksi kunci dari sebuah sistem dalam bentuk objek atau kelas objek. Berfokus pada fungsi atau kemampuan perangkat lunak dari perspektif pengguna akhir.
Kegunaan: Menunjukkan bagaimana sistem memenuhi kebutuhan fungsional.
Representasi: Diagram kelas, diagram objek.
Stakeholder Utama: Pengembang dan analis sistem.

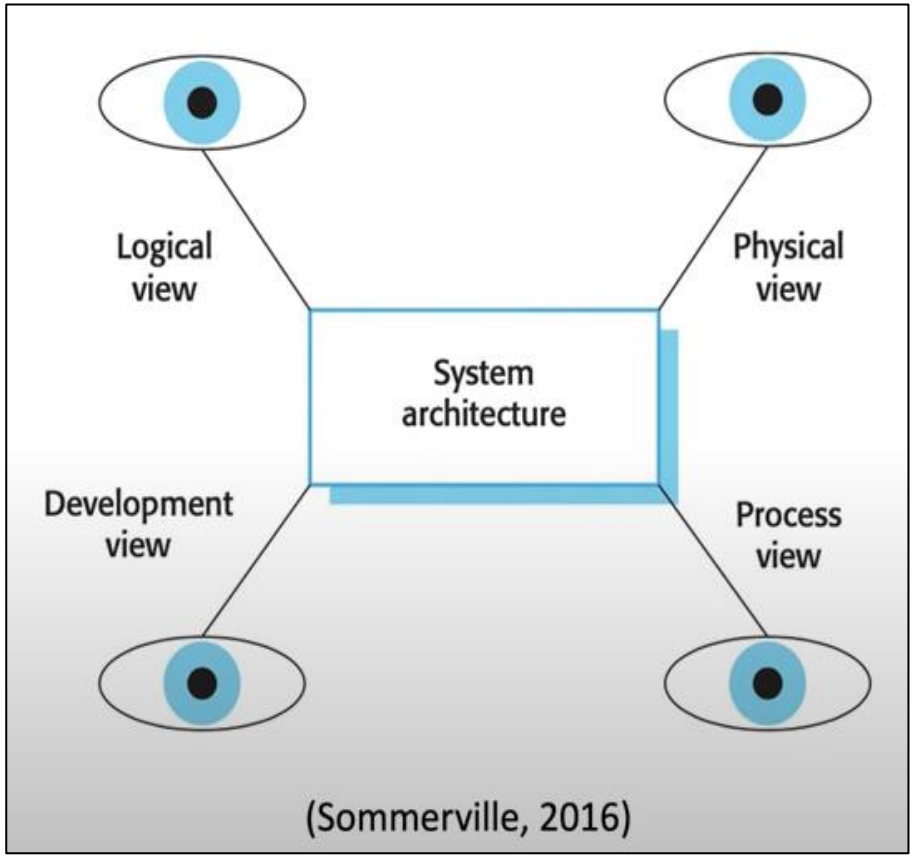
2. Process View
Menunjukkan bagaimana sistem menjalankan suatu proses tertentu. Berfokus pada aspek dinamis dari sistem, seperti proses, thread, dan komunikasi antar proses.
Kegunaan: Menunjukkan bagaimana sistem menangani kebutuhan kinerja, skalabilitas, dan paralelisme.
Representasi: Diagram alur kerja, diagram aktivitas.
Stakeholder Utama: Tim pengembang dan penguji.



Architectural View

3. Development View
Menunjukkan bagaimana sistem diuraikan untuk keperluan pengembangannya.
Biasanya tergambarkan komponen-komponen sistem. Berfokus pada pengorganisasian modul perangkat lunak selama pengembangan.
Kegunaan: Menunjukkan struktur kode dan pembagian tugas.
Representasi: Diagram paket, diagram komponen.
Stakeholder Utama: Pengembang dan arsitek perangkat lunak.

4. Physical View
Menunjukkan perangkat keras yang berkaitan dengan sistem.
Berfokus pada bagaimana perangkat lunak akan di-deploy ke perangkat keras.
Kegunaan: Menunjukkan hubungan antara perangkat keras dan perangkat lunak.
Representasi: Diagram deployment, diagram jaringan.
Stakeholder Utama: Administrator jaringan, tim infrastruktur.

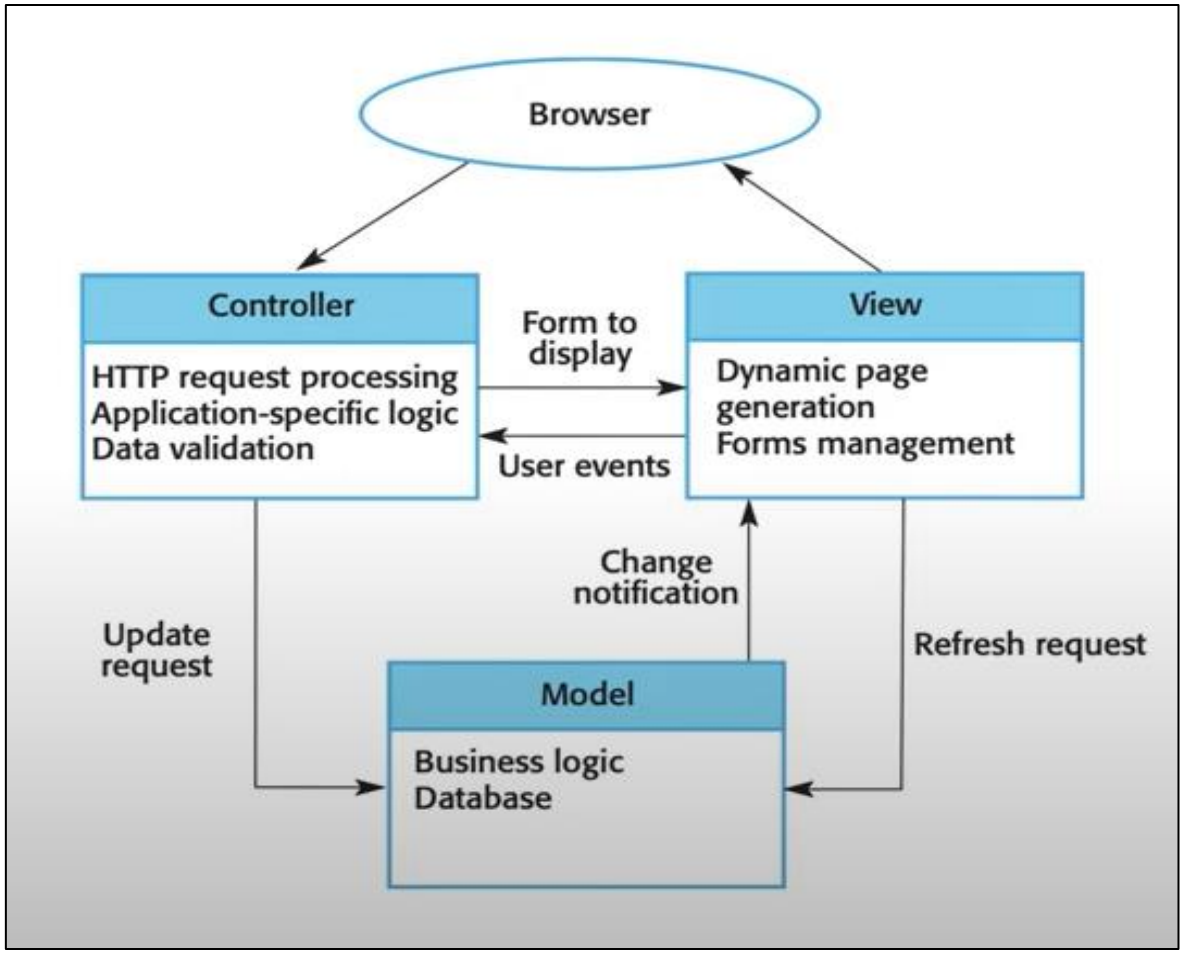


Pola Arsitektur (Architectural Pattern)

- **Architectural pattern** adalah solusi desain generik dan teruji yang digunakan untuk mengatasi masalah umum dalam pengembangan arsitektur perangkat lunak. Pola ini memberikan kerangka kerja konseptual yang dapat digunakan kembali untuk membangun struktur sistem perangkat lunak.
- Pola arsitektur tidak spesifik terhadap teknologi tertentu tetapi lebih berfokus pada prinsip dan pendekatan yang dapat diadaptasi pada berbagai proyek. Mereka membantu merancang sistem dengan efisiensi, skalabilitas, dan modularitas yang lebih baik.

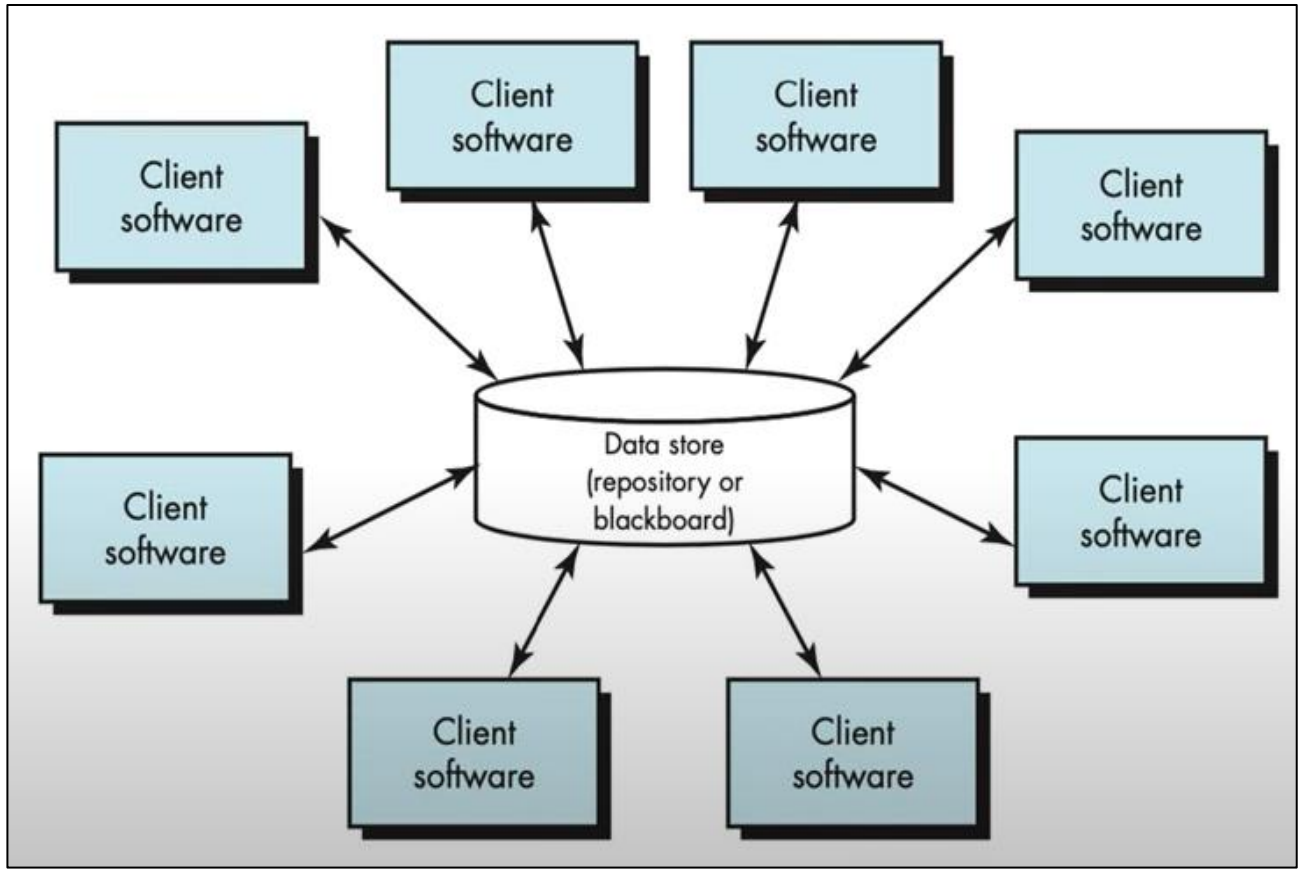
Pola MVC – Model View Controller

- Arsitektur MVC (Model View Controller) dapat digunakan sebagai pola untuk mengatasi permasalahan: “Bagaimana membuat struktur kode program untuk sistem berbasis web”.
- MVC memberikan kerangka kerja untuk memisahkan logika aplikasi (Model), antarmuka pengguna (View), dan kontrol alur data (Controller) dalam sebuah sistem.



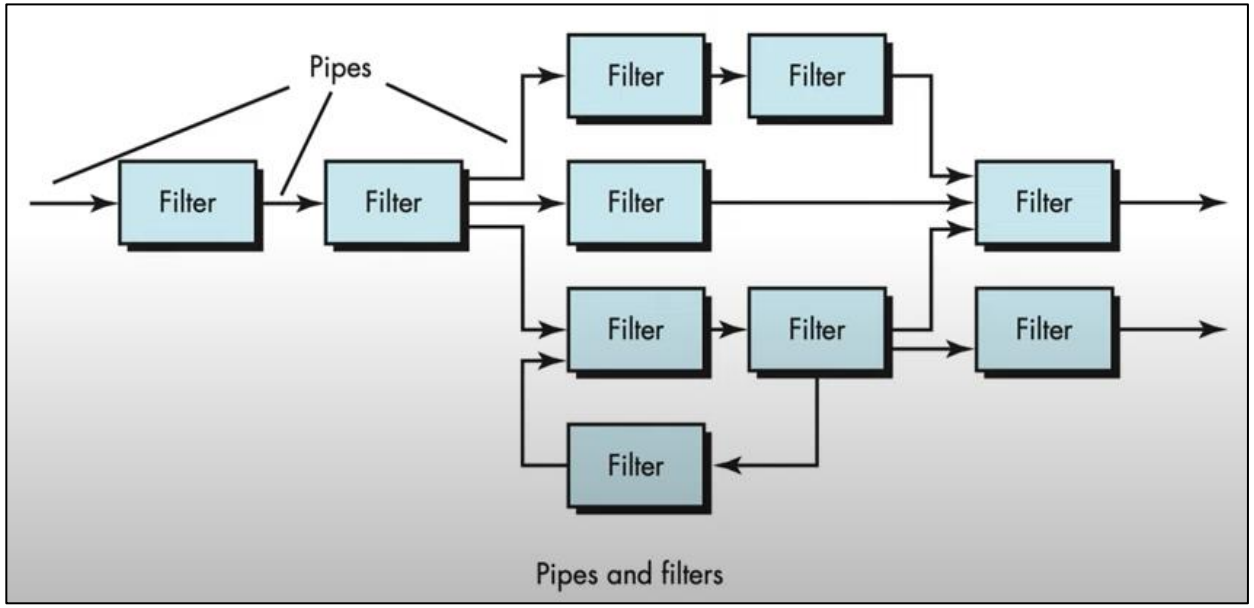
Pola *Data Center Architecture*

- Sebuah arsitektur yang berpusat pada data.
- Sebuah tempat penyimpanan data berada pada pusat arsitektur, komponen-komponen yang menggunakan atau mengelola data tersebut berada di sekelilingnya



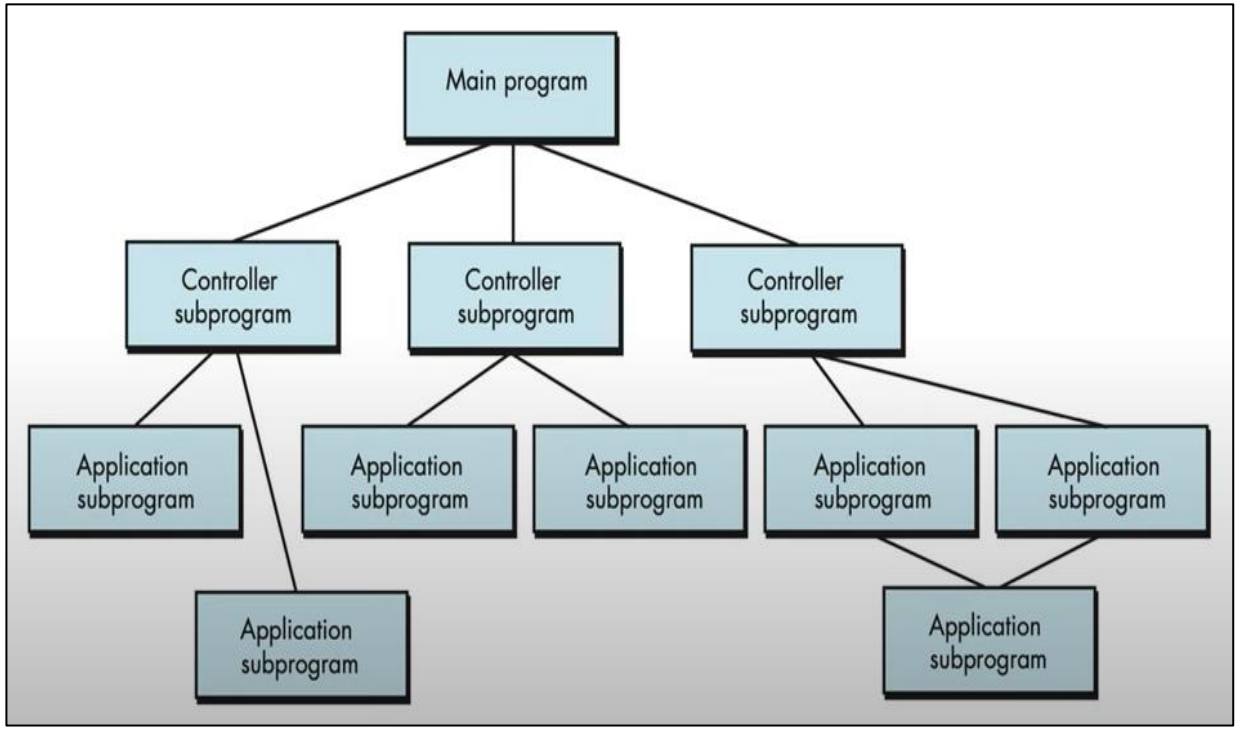
Pola Data Flow Architecture (Pipe-and-Filter Pattern)

- Sebuah arsitektur yang menggambarkan aliran data, biasanya diilustrasikan dengan Pipe and Filter.
- Filter berguna untuk memproses data, sedangkan Pipe menunjukkan kemana data yang telah diproses diteruskan.
- Penggunaan: Aplikasi pemrosesan data, compiler.



Pola Call and Return Architecture

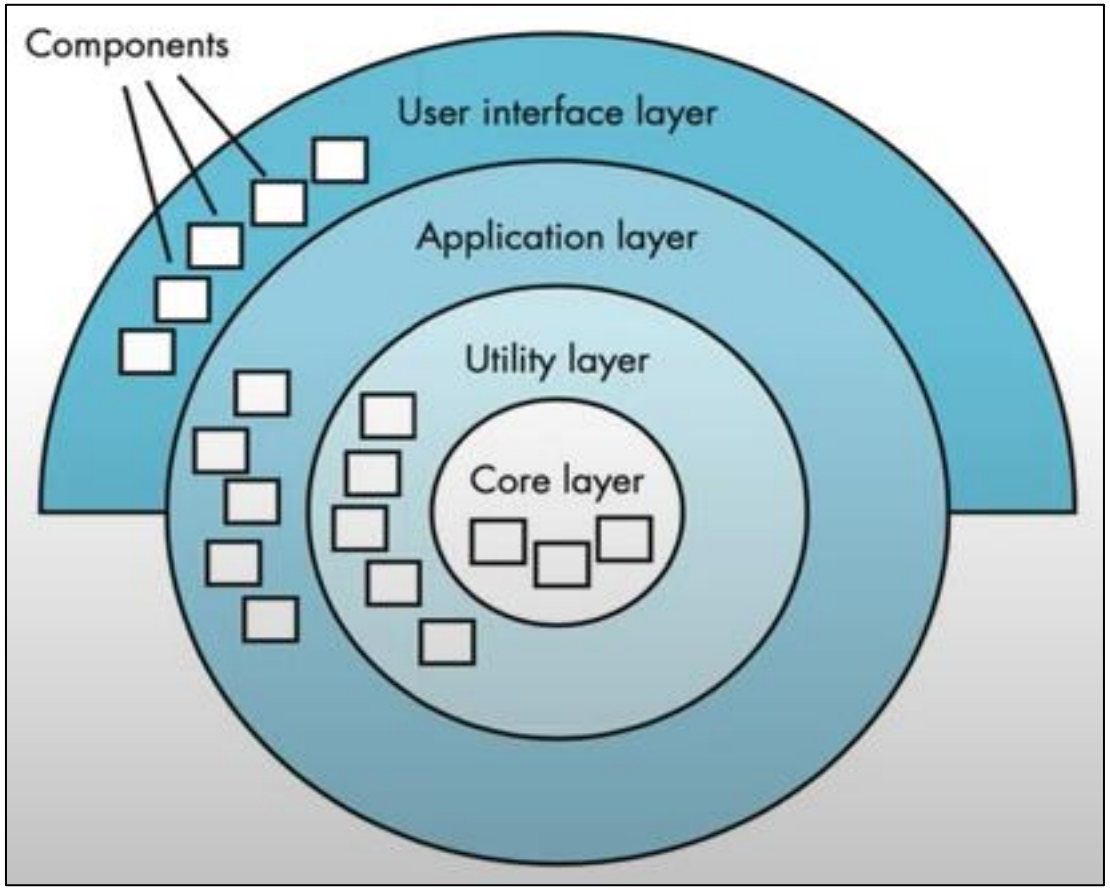
- Dengan menggambarkan struktur program yang relatif mudah untuk dimodifikasi dan diubah ukurannya.
- Dilakukan dengan memecah fungsi ke dalam beberapa komponen atau sub komponen.



Pola Layered Architecture

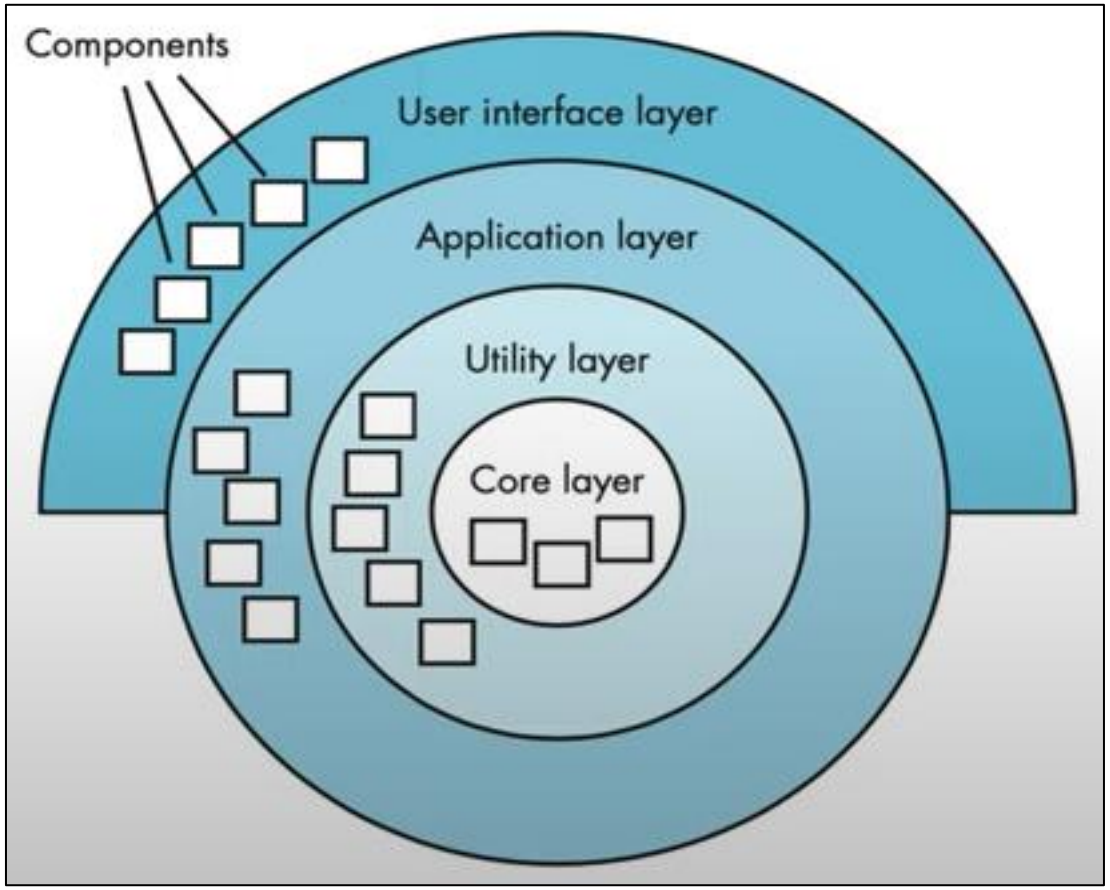
Sebuah arsitektur yang membagi komponen ke dalam lapisan-lapisan. Biasanya setiap lapisan atau layer memiliki fungsi-fungsi tertentu.

- **User interface atau Antarmuka**: Bertanggung jawab atas interaksi dengan pengguna, Menerima input dari pengguna dan mengirimkannya ke lapisan Application untuk diproses. Contoh: Menampilkan halaman web, formulir, atau tampilan aplikasi mobile.
- **Application Layer** : Mengelola logika aplikasi, aturan bisnis, dan alur kerja sistem. Contoh: Menjalankan proses alur bisnis, seperti memverifikasi login atau memproses transaksi.



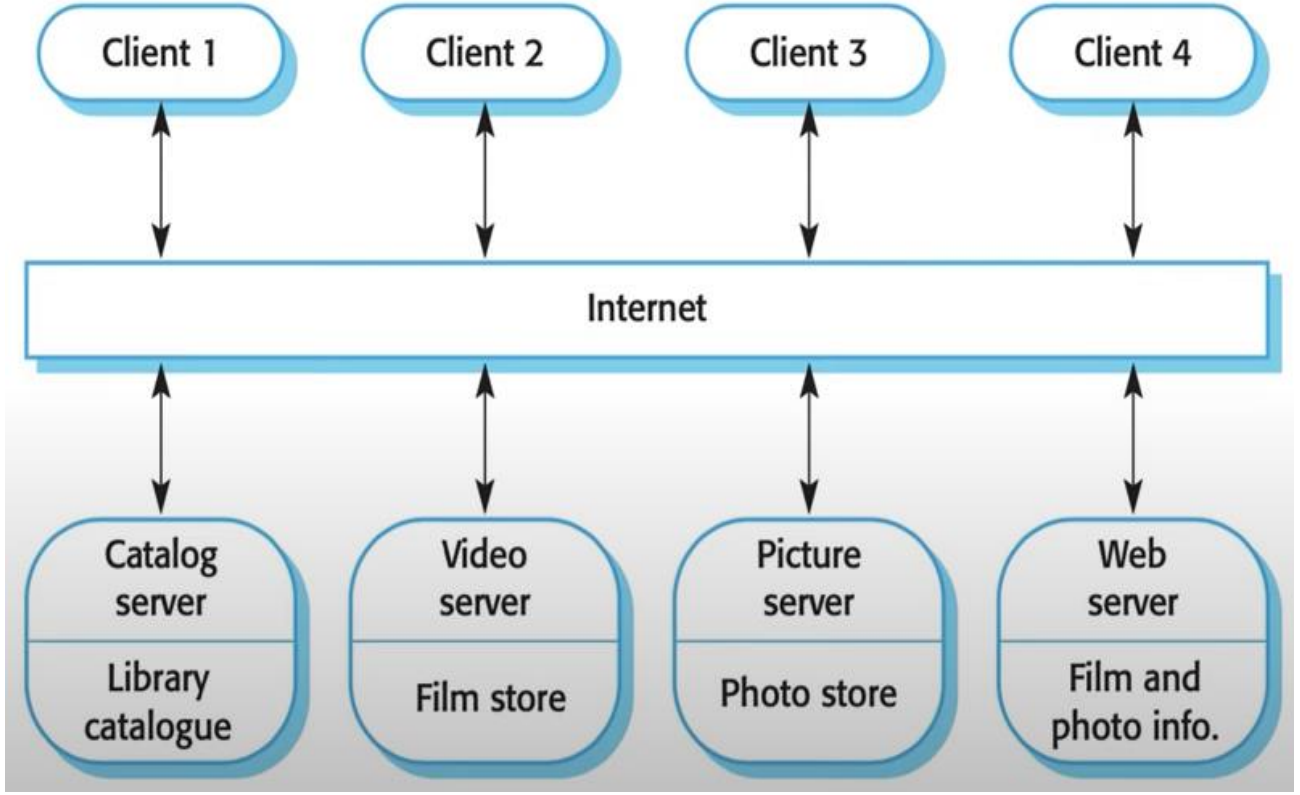
Pola Layered Architecture

- **Utility Layer** : Menyediakan fungsi-fungsi umum atau utilitas yang dapat digunakan oleh lapisan lain. Biasanya mencakup fungsi non-spesifik seperti logging, pengelolaan file, enkripsi, atau koneksi jaringan.
- **Core Layer** : Menyimpan data inti dan logika bisnis yang berkaitan langsung dengan domain aplikasi. Berinteraksi dengan database atau sistem penyimpanan data lainnya.



Pola Client-Server Architectures

- Sistem digambarkan sebagai satu set servis yang diberikan oleh server terpisah.
- Sistem terdiri dari klien (pengguna) dan server (penyedia layanan).
- Penggunaan: Aplikasi berbasis jaringan.

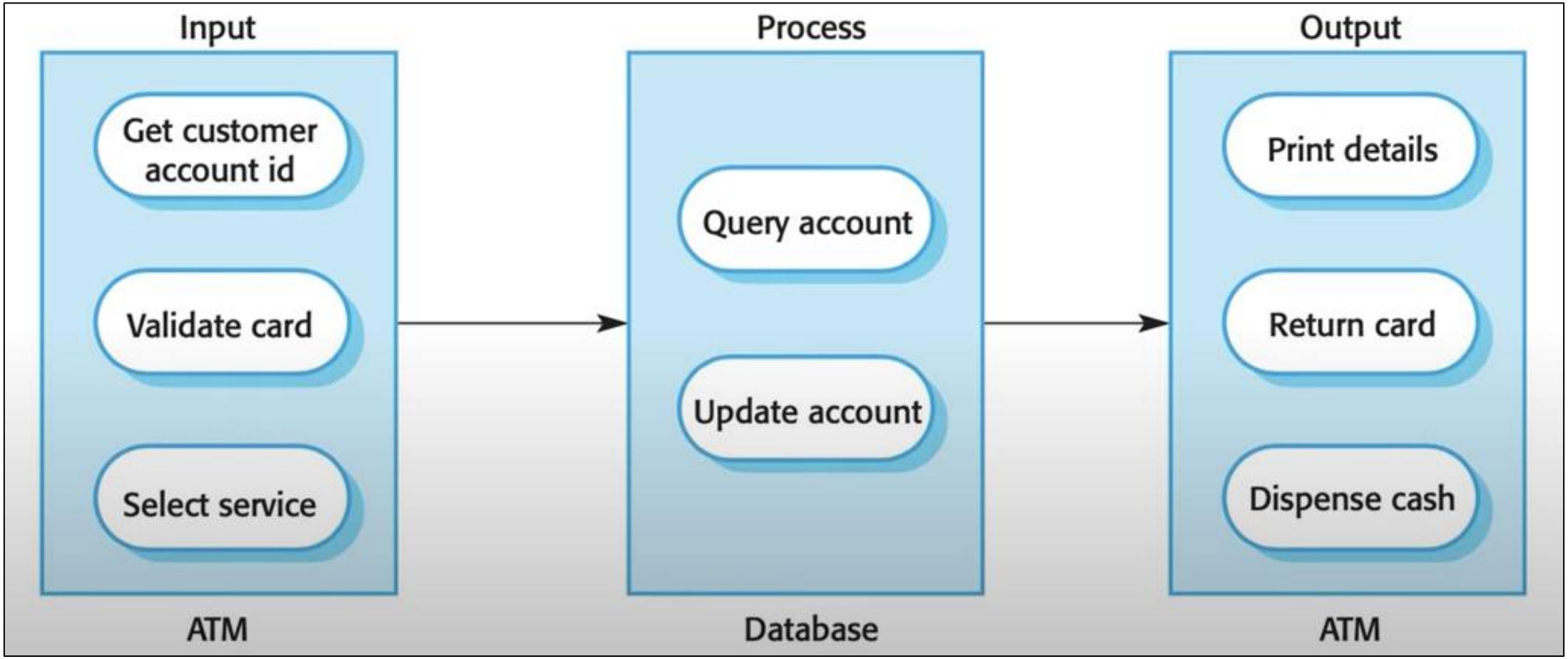


Application Architecture

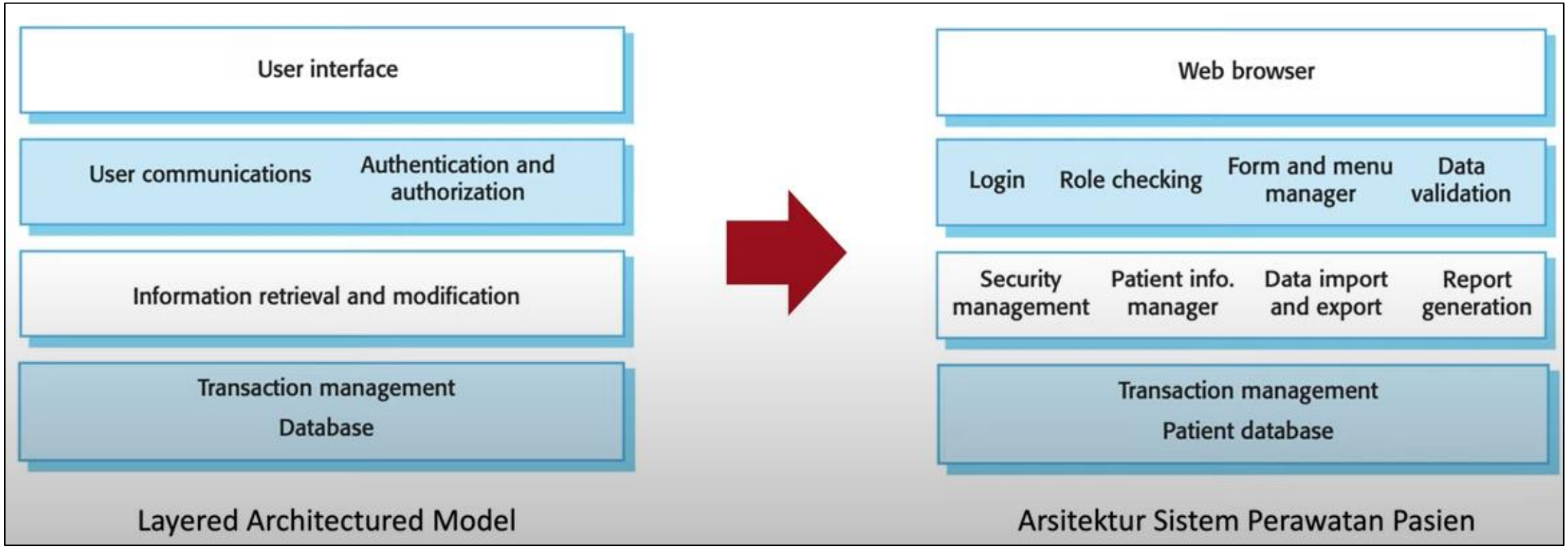
Sebagai seorang software engineer, kita dapat memanfaatkan pola atau model arsitektur dalam berbagai cara:

1. Sebagai *starting point* dalam proses perancangan arsitektur.
2. Sebagai design checklist.
3. Sebagai dasar pembagian tugas tim.
4. Sebagai bahan pertimbangan untuk menggunakan kembali atau reuse komponen yg ada pada sistem
5. Sebagai vocabulary saat membicarakan aplikasi

Contoh:
Arsitektur Mesin ATM (Automatic Teller Machine)



Contoh:
Arsitektur Sistem informasi perawatan pasien



Referensi

- Pressman, R.S. & Maxim, B.R. 2015. Softtware Enginer: A Practitioner's Approach 8th Edition. McGraw-Hill Education, New york.
- Sommerville, I. Software Engineering, 10th edition, Global Edition. Pearson Education Limited.

Any questions?