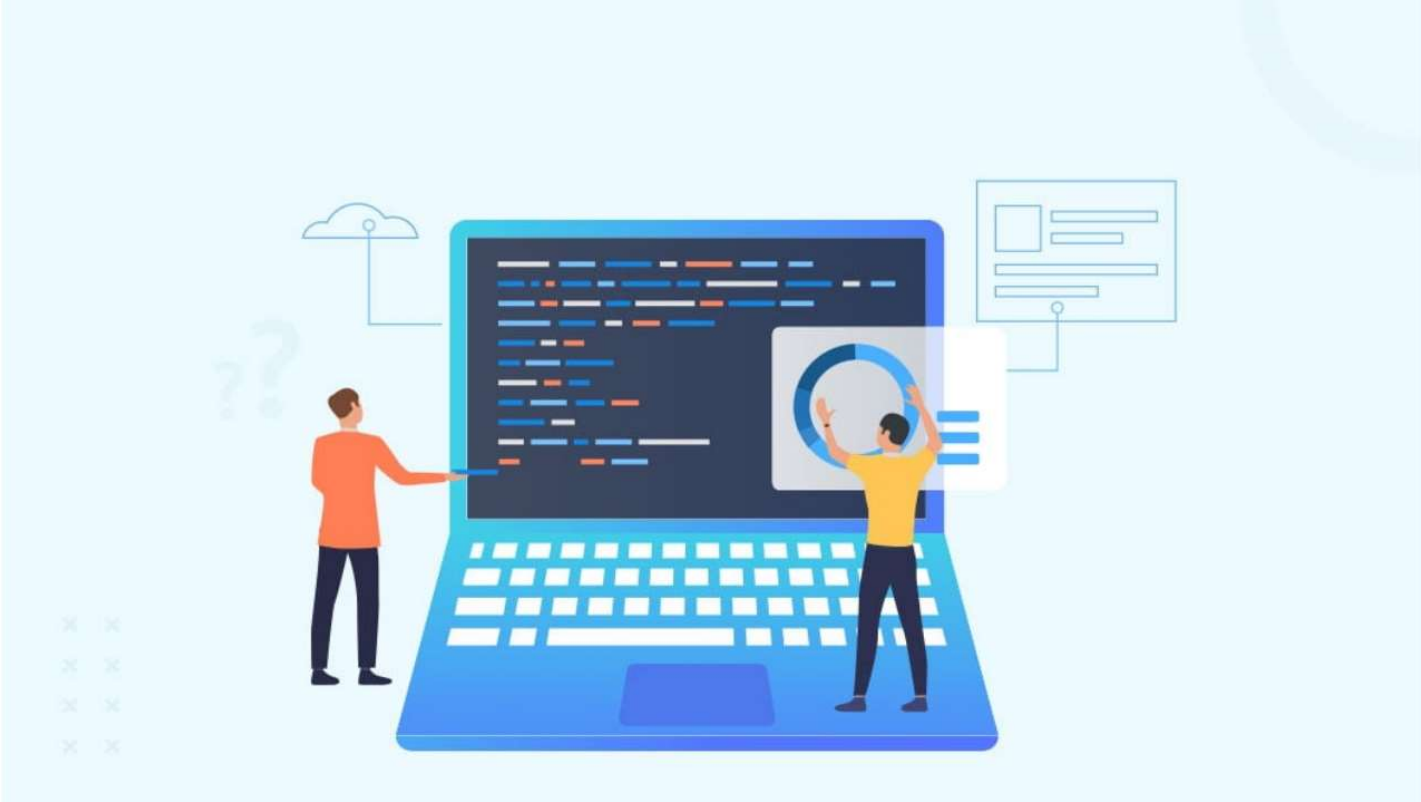
مقدمه ای برمعماری نرم افزار



معماری نرم افزار : مقدمه

- همه ما بارها در طول زندگی خود کلمه معماری را شنیده ایم ولی شاید با شنیدن این کلمه بیش از اینکه فکرمان به سمت سیستم های کامپیوتری برود به سمت ساختمان و یا ساخت و ساز و زیبایی و دکوراسیون می رود.
- حتی معنی این کلمه در فرهنگ لغت نیز، عمل یا فرایند ساخت و کار وابسته به معماری ، ساخت و ساز، هنر یا علم ساخت هر نوع عمارت که مورد استفاده بشر است بیان شده است.

معماری نرم افزار : مقدمه

- شاید اولین جایی که انسان ها از معماری استفاده کردند به زمان های ساخت ساختمان های بزرگ برمی‌گردد. مفهوم معماری به طور حتم از زمانهای قدیم در ذهن بشریت بوده و از آن استفاده می‌کرده است.
- به طورمثال، ساختن بناهای عظیم، بدون تفکر معماری امکان پذیر نبوده است. اگر بناهایی مثل اهرام مصر یا تخت جمشید را بررسی کنیم، به این نتیجه می‌رسیم که ساخت چنین بناهایی بدون نقشه اولیه و تدبیرات قبل از ساخت، امکان پذیر نبوده است و در واقع همان مفهوم معماری است.

معماری نرم افزار : مقدمه

- این بدان معنی نیست که ساختمان های کوچک به معماری نیاز ندارند بلکه می توان این گونه برداشت کرد که هر چه سیستم بزرگ تر و پیچیده باشد نیاز به تفکر معماری از قبل بیش تر احساس می شود.

معماری نرم افزار : مقدمه

- مهمترین مسئله در توسعه سیستم های نرم‌افزاری مقیاس بزرگ، مبحث معماری آن می‌باشد.
- **معماری، ساختارهای مولفه ها و زیرسیستم های یک سیستم مقیاس بزرگ و ارتباط بین آنها می‌باشد.**
- معماری نرم‌افزار، یکی از مهمترین حوزه‌ها در مهندسی نرم‌افزار است و دلیل اهمیت آن تاثیر حیاتی معماری در موفقیتِ توسعه سیستم های نرم‌افزاری است.
- توسعه یک سیستم نرم‌افزاری مقیاس بزرگ با ویژگی‌های مذکور، نیازمند ارائه یک معماری مناسب و کامل برای سیستم نرم‌افزاری مورد نظر می‌باشد.

اگر می گوییم معماری یک خانه بهتراز یک خانه دیگر هست، یعنی چه ؟ منظور چی هست؟

نحوه چیدمان اجزای اساسی تشکیل دهنده خانه
یعنی جایگیری اتاق ها ، آشپزخانه و ...
واینکه این ها چطور به هم اتصال دارند و ارتباط دارند و ... نور

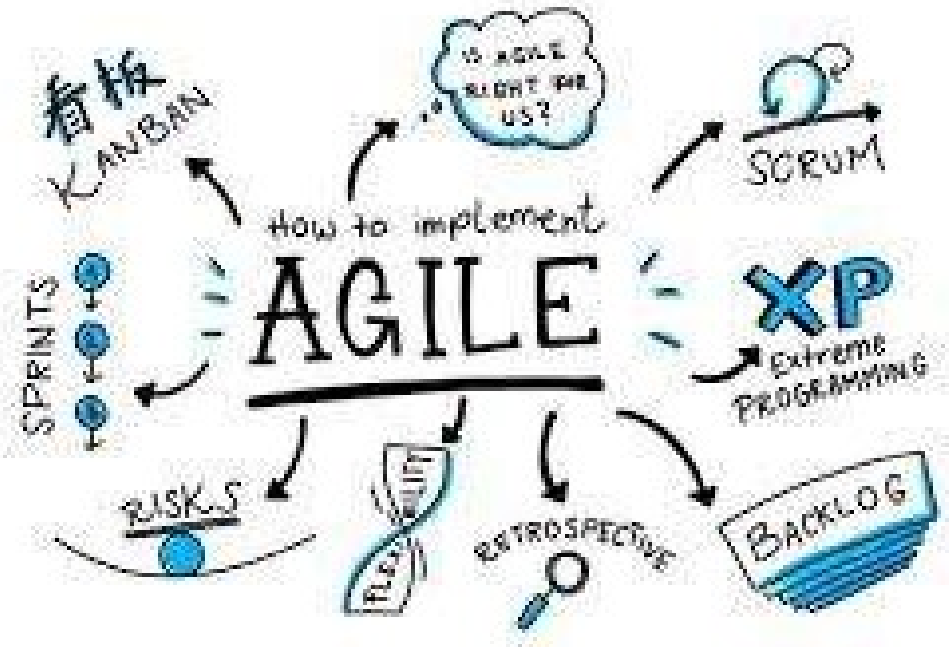
معماری در نرم افزار:

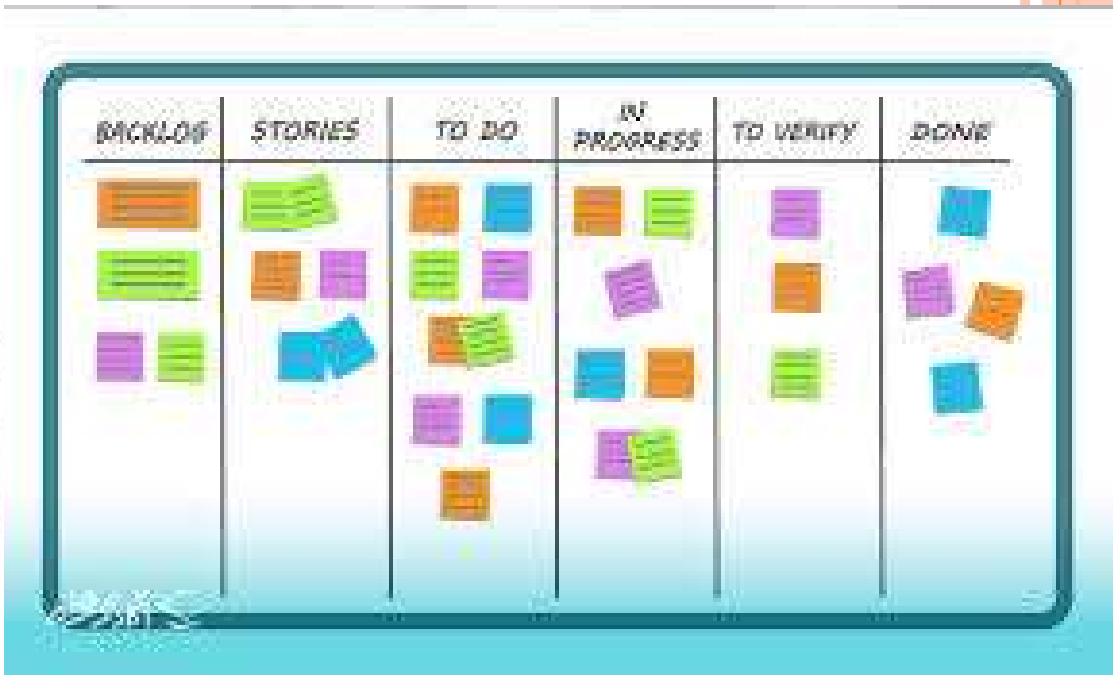
نحوه قرارگیری و ارتباط میان اجزای سازنده نرم افزار
آیا کلاسها باهم ارتباط و اتصال دارند
کدوم کلاس به توابع کلاس X دسترسی دارد و می تواند آنها را فراخوانی کند.

معماری با چابکی در تضاد است؟

توجه به معماری جلوی چابکی را میگیرد؟

نرم افزار باید ساختار داشته باشد ...





ماژولار بودن در معماری نرم افزار چیست ؟

ماژولار بودن نرم افزار، تجزیه نرم افزار به بخش‌های کوچک‌تر با رابط‌های استاندارد است. ما می‌خواهیم محصولاتی با قطعه کدهای قابل استفاده مجدد ایجاد کنیم، بنابراین فقط یک‌بار یک قابلیت عملکردی (فیچر | فانکشن) را پیاده‌سازی کرده و سپس از آن مکرراً استفاده می‌کنیم. ماژولار بودن برای ساده‌سازی برنامه‌های کاربردی و کدهای پایه‌ای بزرگ انجام می‌شود و دارای مزایایی است که در ادامه به برخی از آن‌ها اشاره شده است.

- پیمانه‌بندی کمک می‌کند تا قابلیت‌های عملکردی تقسیم‌بندی شوند و در نتیجه بصری‌سازی، درک و سازماندهی پروژه تسهیل خواهد شد.
- هنگامی که پروژه به‌صورت شفاف سازمان‌دهی و تقسیم‌بندی‌شده باشد، نگهداریآن آسان‌تر است و خطاها و باگ‌های کمتری تجربه خواهد شد.
- اگر پروژه شما به بخش‌های مختلفی تقسیم شده باشد، می‌توان هر بخش را به طور مستقل توسعه داد و اصلاحات لازم را پیاده‌سازی کرد که اغلب بسیار مفید است.

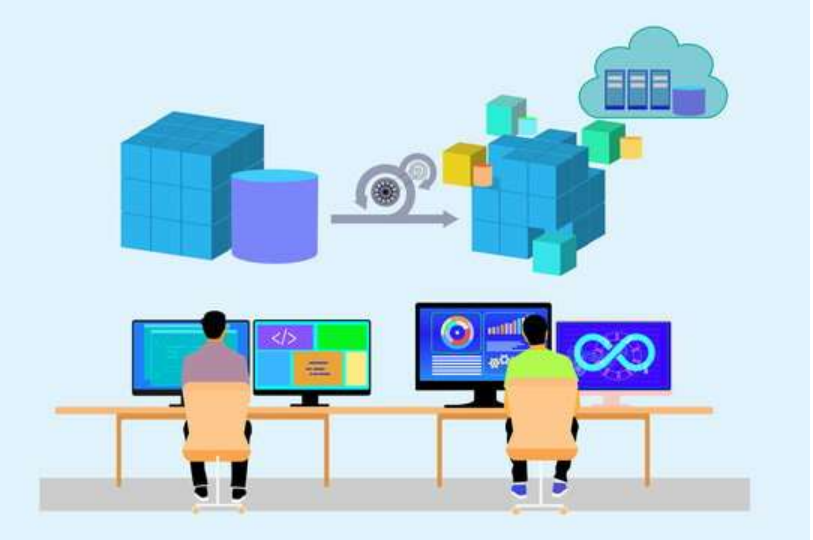
مدلهای معماری

ARCHITECTURAL STYLES

Each style describes a system category that encompasses: (1) a **set of components** (e.g., a database, computational modules) that perform a function required by a system, (2) a **set of connectors** that enable "communication, coordination and cooperation" among components, (3) **constraints** that define how components can be integrated to form the system, and (4) **semantic models** that enable a designer to understand the overall properties of a system by analyzing the known properties of its constituent parts.

انواع معماری:

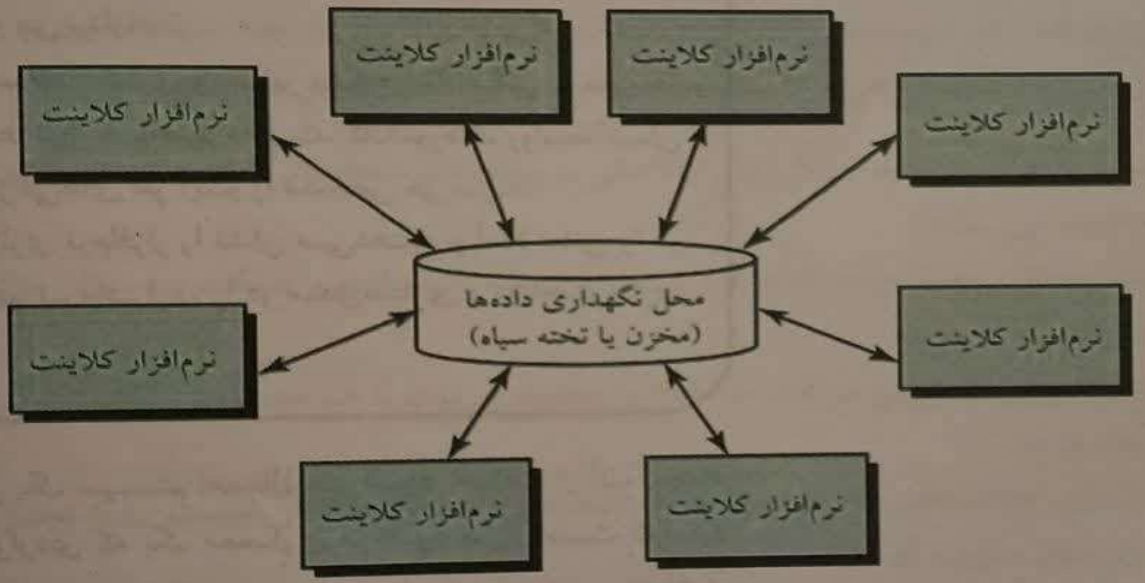
- Data-centered architectures
- Data flow architectures
- Call and return architectures
- Object-oriented architectures
- Layered architectures



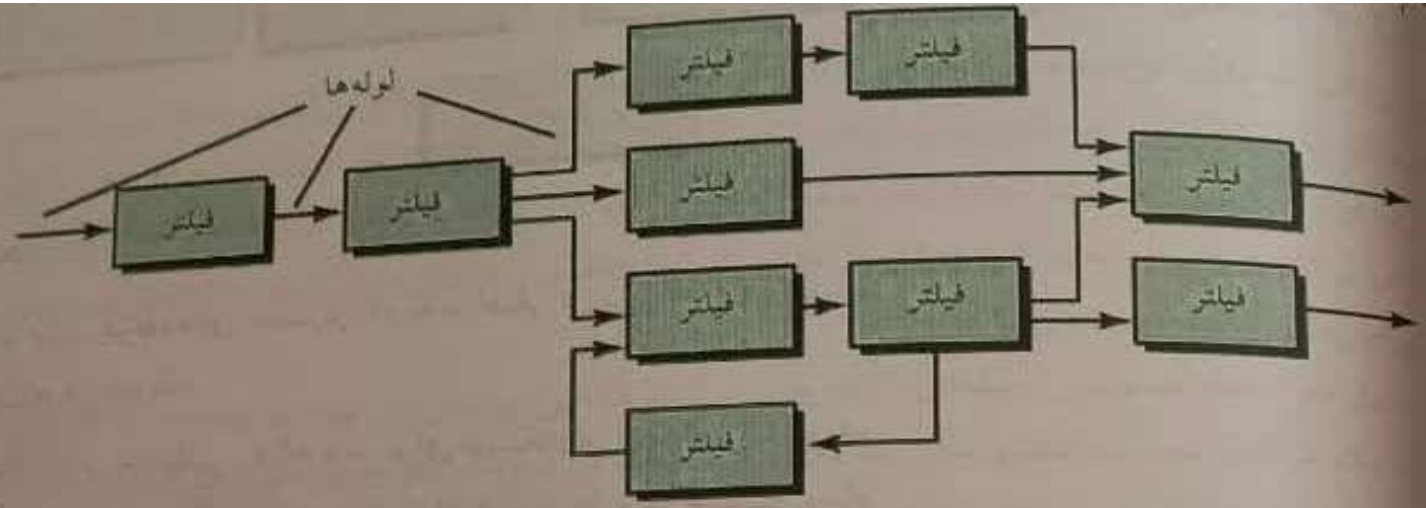
۹-۳-۱ طبقه‌بندی مختصر سبک‌های معماری

گرچه طی شصت سال اخیر، میلیون‌ها سیستم کامپیوتری ایجاد شده است، اکثر آن‌ها را می‌توان در یکی از چند سبک معماری زیر خلاصه کرد:

معماری‌های داده محور (Data-centered). محور این نوع معماری را یک انبار داده‌ها (فایل یا بانک اطلاعاتی) تشکیل می‌دهد که دستیابی به آن غالباً توسط مؤلفه‌های دیگری صورت می‌پذیرد که داده‌های موجود در این انبار را به هنگام، اضافه، حذف یا به طریقی دیگر، اصلاح می‌کنند. در شکل ۹-۱ نمونه‌ای از یک سبک معماری داده محور نشان داده شده است. نرم‌افزار کلاینت به یک مخزن مرکزی دستیابی دارد. در برخی موارد، این مخزن داده‌ها منفعل است به این معنی که نرم‌افزار کلاینت مستقل از هرگونه تغییراتی در داده‌ها یا کنش‌های سایر نرم‌افزارهای کلاینت، به این داده‌ها دستیابی دارد. با تغییر این رویکرد، مخزن به یک «تخته سیاه» تغییر ماهیت می‌دهد که هرگاه داده‌های مورد نظر کلاینت تغییر کند، کلاینت را از آن آگاه می‌سازد.



شکل ۹-۱ معماری داده محور.



لوله‌ها و فیلترها

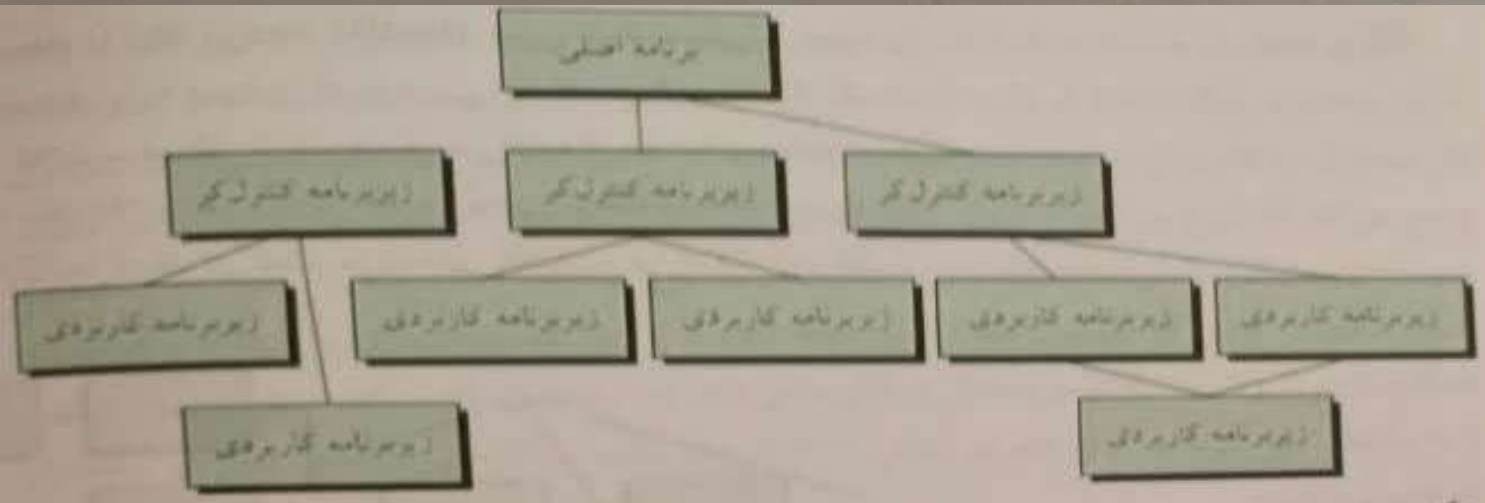
شکل ۲-۹ معماری جریان داده‌ها.

معماری‌های جریان داده‌ها (data-flow). این معماری هنگامی به‌کار برده می‌شود که قرار باشد داده‌های ورودی از طریق یک سری مؤلفه‌های محاسباتی و دستکاری، به داده‌های خروجی تبدیل شوند. الگوی لوله (pipe) و فیلتر (شکل ۲-۹) شامل مجموعه‌ای از مؤلفه‌ها، موسوم به فیلتر، می‌شود که توسط یک سری لوله به هم متصل می‌شوند؛ این لوله‌ها داده‌ها را از مؤلفه‌ای به مؤلفه‌ی بعدی ارسال می‌کنند. هر فیلتر مستقل از مؤلفه‌های فوقانی و زیرین خود عمل می‌کند، طوری طراحی می‌شود که داده‌های ورودی را در شکلی معین پذیرا باشد و داده‌های خروجی را با شکلی خاص تولید می‌کند (تا در اختیار فیلتر بعدی قرار گیرد). به هر حال، فیلتر نیازی ندارد که از عملکرد فیلترهای مجاور خود اطلاع داشته باشد.

اگر جریان داده‌ها تنها به یک خط از تبدیلات منجر شود، به آن، *ترتیب دسته‌ای* (batch sequential) گفته می‌شود. این ساختار، دسته‌ای از داده‌ها را می‌پذیرد سپس یک سری مؤلفه‌های ترتیبی (فیلترها) را به‌کار می‌گیرد تا آن دسته از داده‌ها را تبدیل کند.

معماری‌های فراخوانی و بازگشت. به کمک این سبک معماری می‌توانید به ساختاری برای برنامه دست پیدا کنید که اصلاح و تغییر دادن ابعاد آن نسبتاً آهسته باشد. در این گروه چند سبک فرعی نیز وجود دارد [Bas03]:

- *معماری‌های برنامه اصلی/ زیر برنامه.* در این ساختار کلاسیک برنامه، تابع به یک سلسله مراتب کنترلی تجزیه می‌شود که در آن یک برنامه «اصلی» چند مؤلفه از برنامه را فرا می‌خواند که هر یک به نوبه خود ممکن است مؤلفه‌های دیگری را فراخوانی کند. در شکل ۳-۹، یک معماری از این نوع نشان داده شده است.

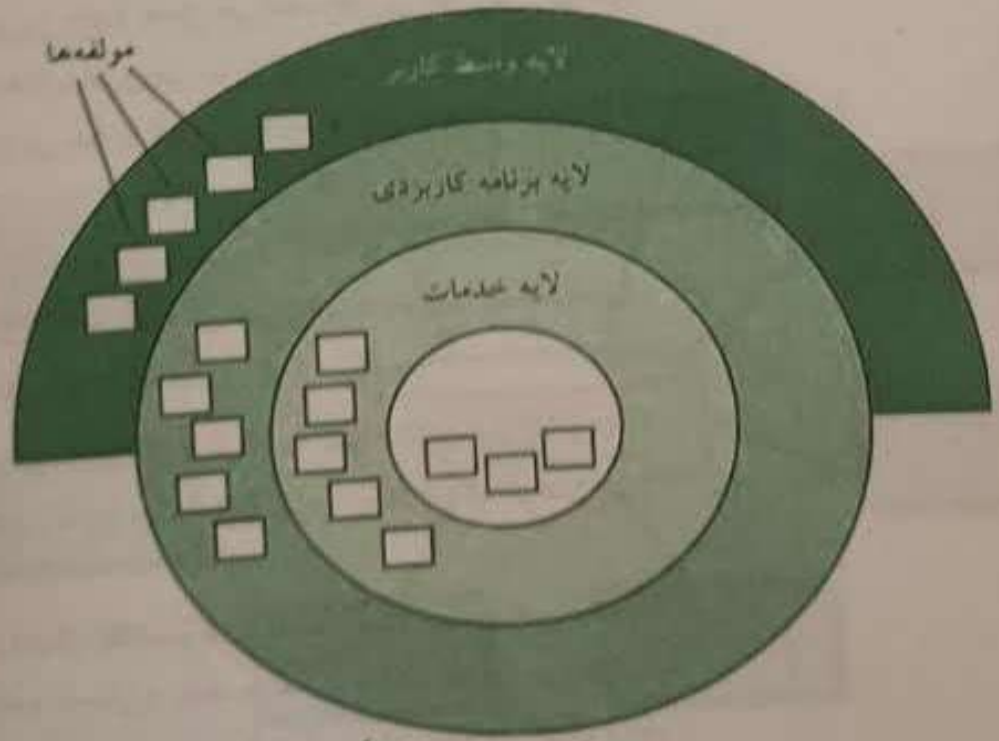


شکل ۳-۹ معماری برنامه اصلی / زیربرنامه.

- *معماری‌های فراخوانی روال‌های راه دور.* مؤلفه‌های معماری «برنامه اصلی / زیربرنامه» در میان چندین کامپیوتر روی یک شبکه توزیع می‌شوند.

معماری شیءگرا. مؤلفه‌های این سیستم، داده‌ها و عملیاتی را که باید برای دستکاری آنها اجرا شوند، کپسوله (encapsulate) می‌کنند. برقراری ارتباط و هماهنگ‌سازی میان مؤلفه‌ها از طریق مبادله پیام انجام می‌شود.

معماری لایه‌ای. ساختار اصلی یک معماری لایه‌ای در شکل ۴-۹ نشان داده شده است. تعدادی لایه‌های متفاوت تعریف می‌شود که هر یک عملیاتی را انجام می‌دهند و به‌طور تدریجی به دستورات ماشین نزدیک‌تر می‌شود. در لایه خارجی، مؤلفه‌ها به عملیات واسط کاربر سرویس می‌دهند. در لایه داخلی، مؤلفه‌ها، ارتباط با سیستم‌عامل را برقرار می‌کنند. لایه‌های میانی خدمات و عملکردهای اصلی نرم‌افزار را فراهم می‌آورند.



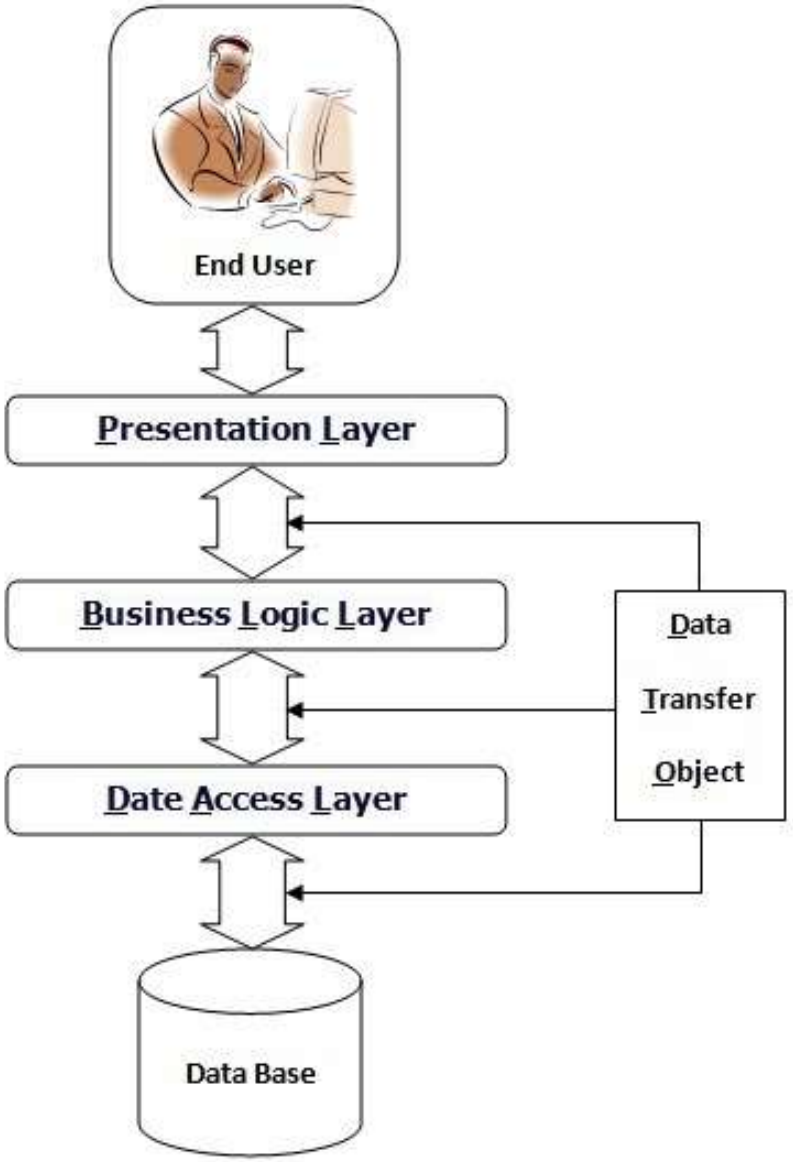
۴-۹ معماری لایه‌ای.

معماری چند لایه

- در مهندسی نرم افزار، سیستم‌های نرم افزاری را به دلیل کاهش پیچیدگی و ساده‌تر شدن آن‌ها و همچنین به خاطر تسهیل در امر نگـــهداری و اعمال تغییرات در آنها، به چـند زیر سیستم تقسیم کـرده و قسمت‌های مستقل سیستم را به صورت لایه‌های جداگانه و مستقل از هم طراحی می‌کنند.

- هر کدام از این لایه‌ها ضمن اینکه وظیفه خاص خود را دارند، با هم در ارتباط بوده به‌طوری‌که هر لایه به لایه‌های بالایی و پایینی خود سرویس داده و از آن‌ها سرویس می‌گیرد.

معماری ۳ لایه



معماری ۳ لایه حالت خاصی

از معماری چند لایه می‌باشد

که سیستم‌ها بر اساس این

معماری به ۳ لایه جداگانه

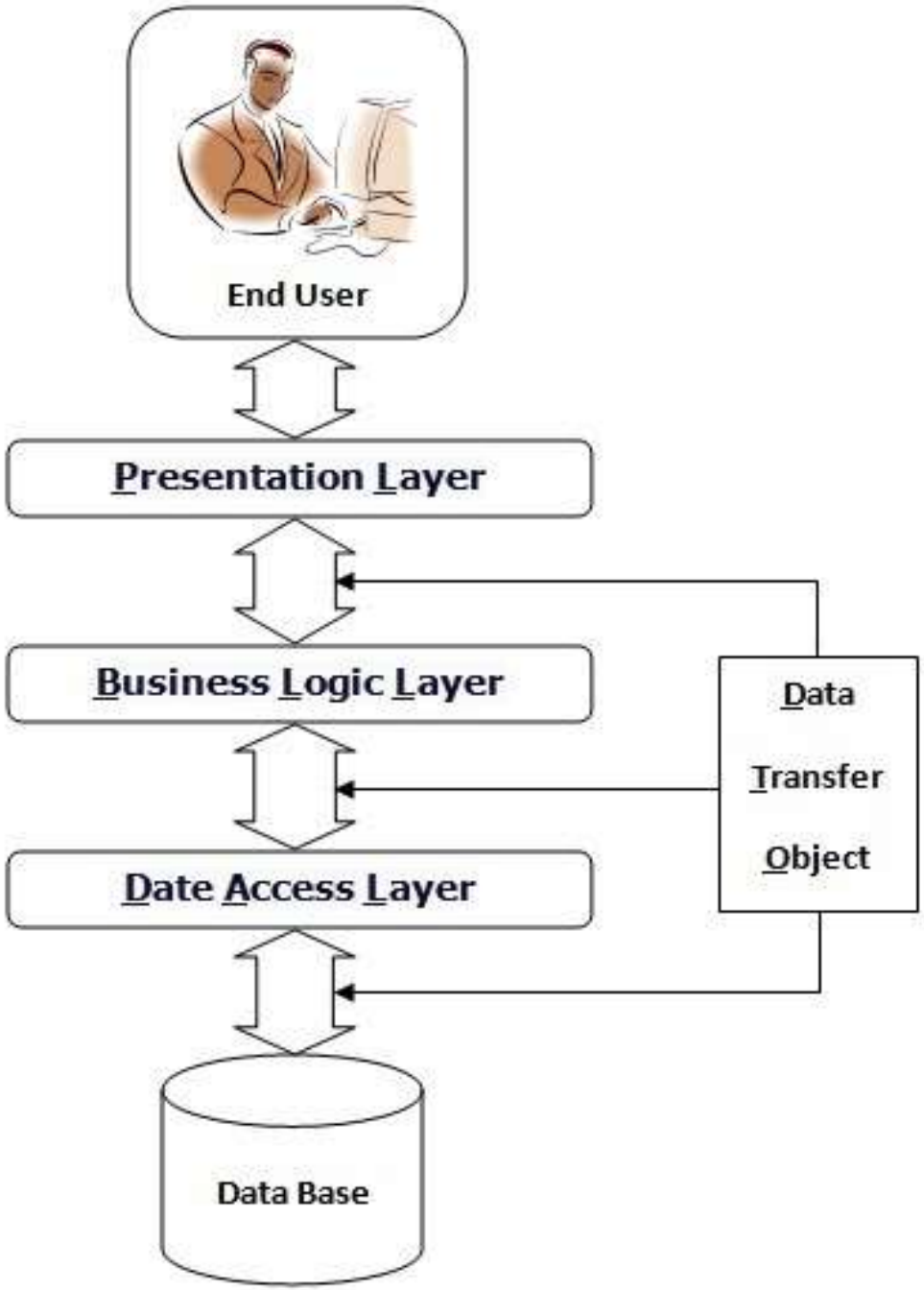
تقسیم می‌شوند.

Presentation Layer (لایه نمایش)

- این لایه که به آن لایه Interface نیز گفته می‌شود شامل تمام عناصر قابل رویت مربوط به رابط گـرافیکی کاربرمی‌باشد و در واقع هـــر آنچه را که کاربر نهایی استفاده‌کننده از سیستم مشاهـده می‌کند از قبیل فرم‌ها، کنترل‌های روی فرم‌ها، تصاویر، منوهای برنامه و... در این لایه قرار می‌گیرند.

Presentation Layer (لایه نمایش)

- کاربر سیستم فقط با این لایه در ارتباط بوده و هیچ ارتباطی با دیگر لایه‌ها ندارد و در واقع درخواست خود را از طریق لایه نمایش به لایه‌های زیرین انتقال می‌دهد. وظیفه لایه نمایش این است که اطلاعات لازم را از کاربر گرفته و در صورت لزوم برخی Validation یا اعتبار سنجی‌هایی که باید در این لایه انجام گیرد.
- مثل: کنترل طول فیلدها، کنترل اجباری بودن بعضی فیلدها و... را انجام می‌دهد و این اطلاعات را برای هر گونه پردازش لازم دیگری به لایه بعدی ارسال می‌نماید و در واقع هیچ اثری از منطق اصلی برنامه و اتصال به بانک اطلاعاتی در این لایه دیده نمی‌شود.


End User
Presentation Layer
Business Logic Layer
Date Access Layer
Data Base
Data
Transfer
Object

Business Logic Layer (لایه منطق تجاری)

- این لایه که به آن لایه میانی (Middle Tier) نیز گفته می‌شود حاوی منطق اصلی برنامه بوده و وظیفه ارتباط بین لایه نمایش و لایه داده را بر عهده دارد.
- در واقع کلیه درخواست‌هایی که در اثر تعامل کاربر با لایه نمایش ایجاد شده‌است به این لایه منتقل شده و تمام پردازش‌های لازم بر اساس منطق اصلی برنامه در این لایه انجام شده و نتیجه این پردازش مجدداً به لایه نمایش منتقل شده و برای کاربر به نمایش درمی آید.

Business Logic Layer (لایه منطق تجاری)

- گاهی اوقات درخواست کاربر به گونه‌ای است که لایه منطق تجاری برای انجام آن نیاز دارد که با لایه داده یعنی لایه زیرین خود ارتباط داشته باشد مثلاً کاربر ممکن است عملیات جستجو در محصولات یک شرکتی را بخواهد انجام دهد به این صورت که از طریق لایه نمایش لیست محصولات شرکت را درخواست می‌نماید، لایه نمایش درخواست کاربر را به لایه منطق تجاری ارسال می‌کند و این لایه نیز به دلیل اینکه انجام درخواست کاربر نیاز به برقراری ارتباط با لایه داده دارد، درخواست کاربر را به لایه داده ارسال می‌کند.
- لایه داده نیز درخواست کاربر را انجام داده و لیست محصولات شرکت را از بانک اطلاعاتی دریافت کرده و به لایه بالایی خود یعنی لایه منطق برنامه انتقاال داده و لایه منطق برنامه نیز این لیست را عینا با لایه بالایی خود یعنی لایه نمایش انتقال داده و در نهایت لایه نمایش این لیست را به کاربر نمایش می‌دهد.

Date Access Layer (لایه دسترسی به داده)

- این لایه که به آن لایه بانک اطلاعاتی نیز گفته می‌شود وظیفه مدیریت اطلاعات موجود در بانک اطلاعاتی یا همان Databaseرا بر عهده دارد و بر اساس درخواست‌هایی که از لایه بالایی خود دریافت می‌کند عملیاتی از قبیل: حذف، اضافه، اصلاح، خواندن اطلاعات و... را بر روی بانک اطلاعاتی انجام داده و نتیجه عمل را به لایه بالایی خود ارسال می‌کند.
- باید توجه داشت که ارتباط با بانک اطلاعاتی فقط از طریق لایه داده انجام می‌گیرد.

مزایای استفاده از معماری ۳ لایه

- هر گونه تغییر در لایه Dataیا منابع داده‌ای خارجی موجب می‌شود تا کمترین آسیب را به لایه‌های دیگر و ساختار برنامه وارد نماید.
- بخش بندی برنامه در چندین لایه که باعث کاهش پیچیدگی و حجم کد نویسی می-گردد .
- تغییر، توسعه، نگهداری و پشتیبانی از نرم افزار در آینده به آسانی صورت می‌گیرد .
- هر لایه به صورت مستقل عمل می‌کند و هیچگونه وابستگی بین آن‌ها وجود ندارد .
- امکان استفاده مجدد از کلاس‌های هر لایه در پروژه‌های دیگر .
- تجزیه یک سیستم به چند زیر سیستم و توسعه سریعتر و آسان‌تر آن .
- لایه Applicationبا عبارات و دستورات پایگاه داده آمیخته نمی‌گردد. (امنیت)

معایب استفاده از معماری ۳ لایه

- افزایش حجم کاری و طولانی‌تر شدن روند تولید و توسعه سیستم
- کاهش سرعت سیستم در صورت افزایش لایه‌ها
- امکان بروز اشتباه در تعیین مرز لایه‌ها

نتیجه گیری

- از مزایای معماری ۳ لایه یا چند لایه می‌توان به عدم وابستگی لایه‌ها به همدیگر اشاره نمود. مثلاً لایه داده مستقل از لایه‌های دیگر عمل کرده و در صورت لزوم می‌توان با سرعت بالا و هزینه پایین این لایه را تغییر داد.
- به عنوان مثال در برنامه‌ای که از بانک اطلاعاتی SQL Server استفاده می‌شود به راحتی می‌توان بانک اطلاعاتی را به اوراکل یا هر بانک اطلاعاتی دیگر تغییر داد یا مثلاً منطق اصلی برنامه را با کمترین هزینه می‌توان با تغییر لایه منطق تجاری تغییر داد.

- اما معماری ۳ لایه یا چند لایه همیشه راه حل مناسبی برای طراحی سیستم‌ها نیست به عبارت دیگر برای طراحی هر سیستمی به دلیل مقرون به صرفه نبودن نمی‌توان از این معماری استفاده کرد. مثلاً برای سیستم‌های کوچک استفاده از معماری ۳ لایه زمان بر و هزینه بر می‌باشد و در واقع هزینه Develope و نگهداری سیستم بالا می‌رود.

MVC چیست؟
الگوی معماری MVC چیست ؟

MVCالگوی معماری مرسوم برای توسعه و ساخت برنامه‌های کاربردی است که دارای رابط کاربری User Interface | UI هستند.
در ابتدا، از الگوی معماری MVCبرای رابط کاربری گرافیکی (GUIبرنامه‌های دسکتاپ) (کامپیوترهای رومیزی) استفاده می‌شد؛ اما امروزه، الگوی معماری MVC برای طراحی کاربردهای (اپلیکیشن‌های) مبتنی بر وب و برنامه‌های کاربردی موبایل به کار می‌رود.