فصل هفتم

اصول و قواعد طراحی نرم افزار

راهبردهای مدل سازی نیازمندی ها

- طراحی نرم افزار و مهندسی نرم افزار
- در آغاز و پس از تحلیل و تعیین نیازهای نرم افزار، طراحی نرم افزار با انجام سه فعالیت یعنی طراحی، تولید برنامه و آزمون انجام می شود.
- هر یک از عناصر مدل تحلیل عنوان شده در بخش قبل، اطلاعات لازم را در ایجاد چهار مدل طراحی فراهم می آورند.

راهبردهای مدل سازی نیازمندی ها

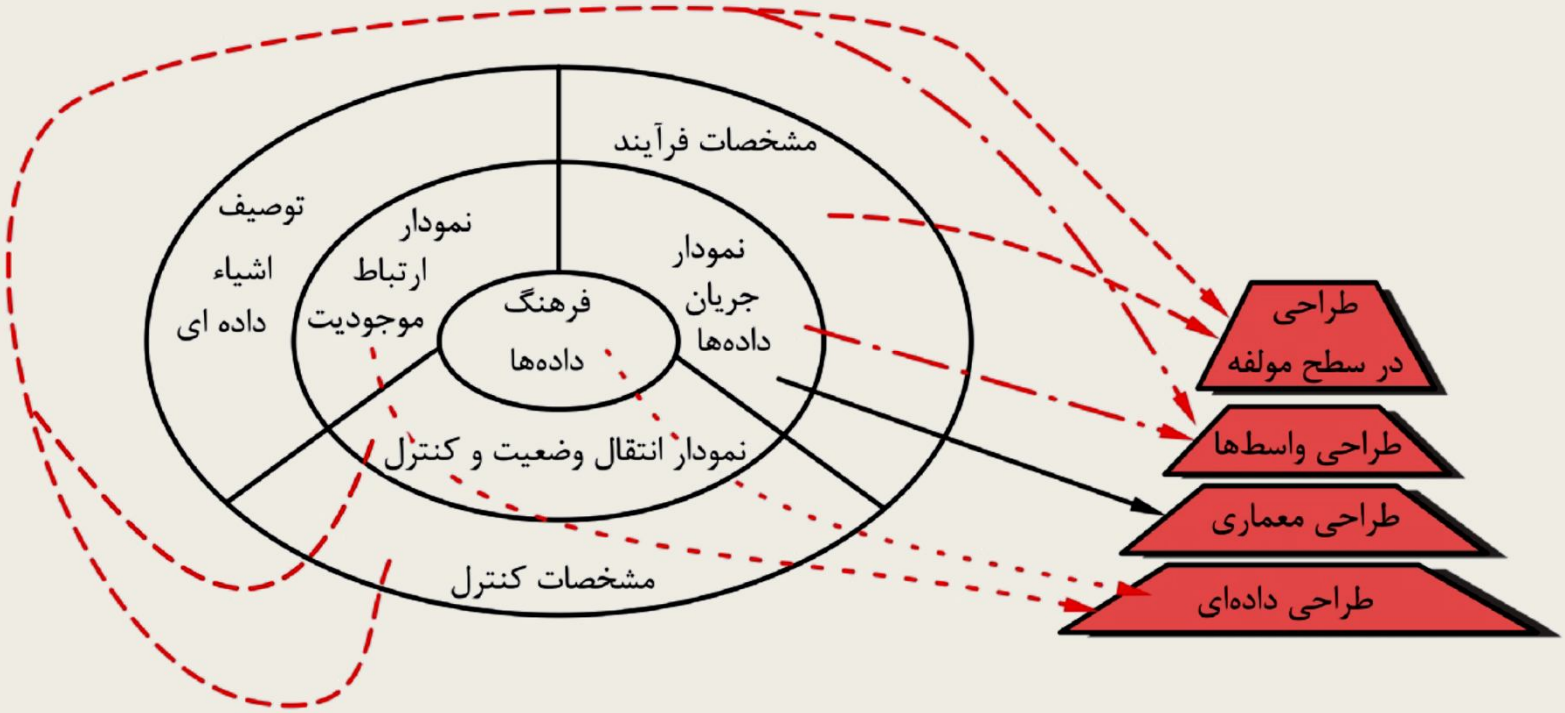
- طراحی نرم افزار و مهندسی نرم افزار
- **طراحی داده ها**، مدل داده ای تولیدشده در مدل تحلیل را به ساختارهای داده ای لازم در اجرای نرم افزار تبدیل می کند. اشیا و روابط تعیین شدهٔ داده ها در نمودار ارتباط موجودیت ها و محتوای مشروح داده ای در فرهنگ داده ها، مبنای فعالیت طراحی داده ها را فراهم می کنند. بخشی از طراحی داده ها ممکن است با طراحی معماری نرم افزار همراه شود. البته طراحی جزئی تر داده ها همراه با طراحی هر یک از مؤلفه های نرم افزار صورت می گیرد.

راهبردهای مدل سازی نیازمندی ها

- طراحی نرم افزار و مهندسی نرم افزار
- طراحی معماری، رابطهٔ بین عناصر اصلی مؤلفه های نرم افزاری را تعیین می کند، یعنی رابطهٔ "الگوهای طراحی" به کاررفته در تحقق نیاز های سیستم و محدودیت های مؤثر بر شیوهٔ اجرای الگوهای طراحی معماری تدوین می شود.
- طراحی واسط، توصیف کنندهٔ نحوهٔ ارتباط نر مافزار با سیست مها و افرادی است که با آن تعامل داشته و آن را به کار می برند.
- طراحی مؤلفه (پیمانه)، عناصر ساختاری معماری نرم افزار را به توصیف رویه ای مؤلفهٔ نرم افزاری تبدیل می کند. اطلاعات به دست آمده از STD، CSPEC و PSPEC، پایه و اساس طراحی مولفه به شمار می روند.

راهبردهای مدل سازی نیازمندی ها

- **برگرداندن مدل تحلیل به مدل طراحی نرم افزار**



راهبردهای مدل سازی نیازمندی ها

- اهمیت طراحی نرم افزار را تنها با یک کلمه یعنی "کیفیت" می توان بیان کرد. طراحی روندی است که طی آن کیفیت فرایند مهندسی نرم افزار، بهبود می یابد. طراحی، نمونه هایی از نرم افزار را که از لحاظ کیفی قابل ارزیابی هستند، در اختیار ما قرار می دهد. این تنها راهی است که به واسطه آن می توانیم شکل درست نیازها و خواسته های مشتری را به یک محصول نرم افزاری یا سیستم تبدیل کنیم.

راهبردهای مدل سازی نیازمندی ها

- طراحی نرم افزار فرایندی تکراری است که به موجب آن نیازها و ضرورت ها برای ساخت نرم افزار، تبدیل به یک "طرح یا نقشه" می شوند. در ابتدا، طراحی یک دید کلی از نرم افزار را در سطح بالایی از انتزاع نشان می دهد. با تکرار و پالایش در طراحی، نمایش طرحی در سطوح بسیار پایین تر از انتزاع حاصل می شود. این سطوح برای دستیابی به نیازهای قابل ردیابی، ملاک کار خواهند بود.

راهبردهای مدل سازی نیازمندی ها

- **معیارهای ارزیابی کیفیت یک طراحی خوب**
- **طراحی باید پیمانه ای باشد؛ یعنی نرم افزار باید به طور منطقی به اجزایی تقسیم شود که اعمال اصلی و فرعی خاصی را انجام دهند.**
- **طراحی باید نمایش های مجزایی از داده ها، معماری، واسط ها و پیمانه ها را دربرداشته باشد.**
- **طراحی باید منجر به ساختارهای داده ای شود که برای پیاده سازی اشیا مناسب بوده و از الگوهای قابل تشخیص داده ها ناشی شوند.**
- **طراحی باید به اجزایی منتهی شود که خصوصیات مستقل کاربردی را نمایش دهند.**
- **طراحی باید به واسط هایی ختم شود که از پیچیدگی روابط بین پیمانه ها و محیط خارجی می کاهند.**
- **طراحی باید حاصل کاربرد یک شیوهٔ قابل تکرار با استفاده از اطلاعات به دست آمده در اثنای تحلیل نیازهای نرم افزاری باشد.**

قواعد طراحی ۹ گانه

انتزاع

- وقتی برای هر مسئله به دنبال راه حل پیمانه ای باشیم، سطوح انتزاع زیادی نیز مطرح می شوند. در بالاترین سطح انتزاع، راه حل با به کارگیری زبان رایج در محیط مسئله و به صورت کلی بیان می شود. در سطوح پایین تر انتزاع، جهت گیری و گرایش بیشتر رویه ای است.
- حین پیش روی در سطوح مختلف انتزاع، تلاش می کنیم تا انتزاع های رویه ای، داده ای و کنترلی ایجاد شوند.
  - انتزاع رویه ای توالی مشخصی از دستورالعمل هاست که عملکرد خاص و محدودی دارد.
  - انتزاع داده ای مجموعه ای از داده های با نام است که یک شیء داده ای را توصیف می کند.
  - انتزاع کنترلی سومین شکل انتزاعی است که در طراحی نرم افزار به کار می رود. همانند انتزاع رویه ای و داده ای، انتزاع کنترلی بر مکانیزم کنترل برنامه بدون مشخص کردن جزئیات داخلی اشاره دارد.

قواعد طراحی ۹ گانه

- **پالایش**

پالایش گام به گام یک راهبرد طراحی بالا به پایین است. برنامه با سطوح پالایشی متوالی جزئیات رویه ای، توسعه می یابد. تا زمان دستیابی به دستورات زبان برنامه نویسی، توسعهٔ سلسله مراتب با تجزیهٔ عملکردها (انتزاع رویه ای) طی شیوه ای گام به گام صورت می گیرد.

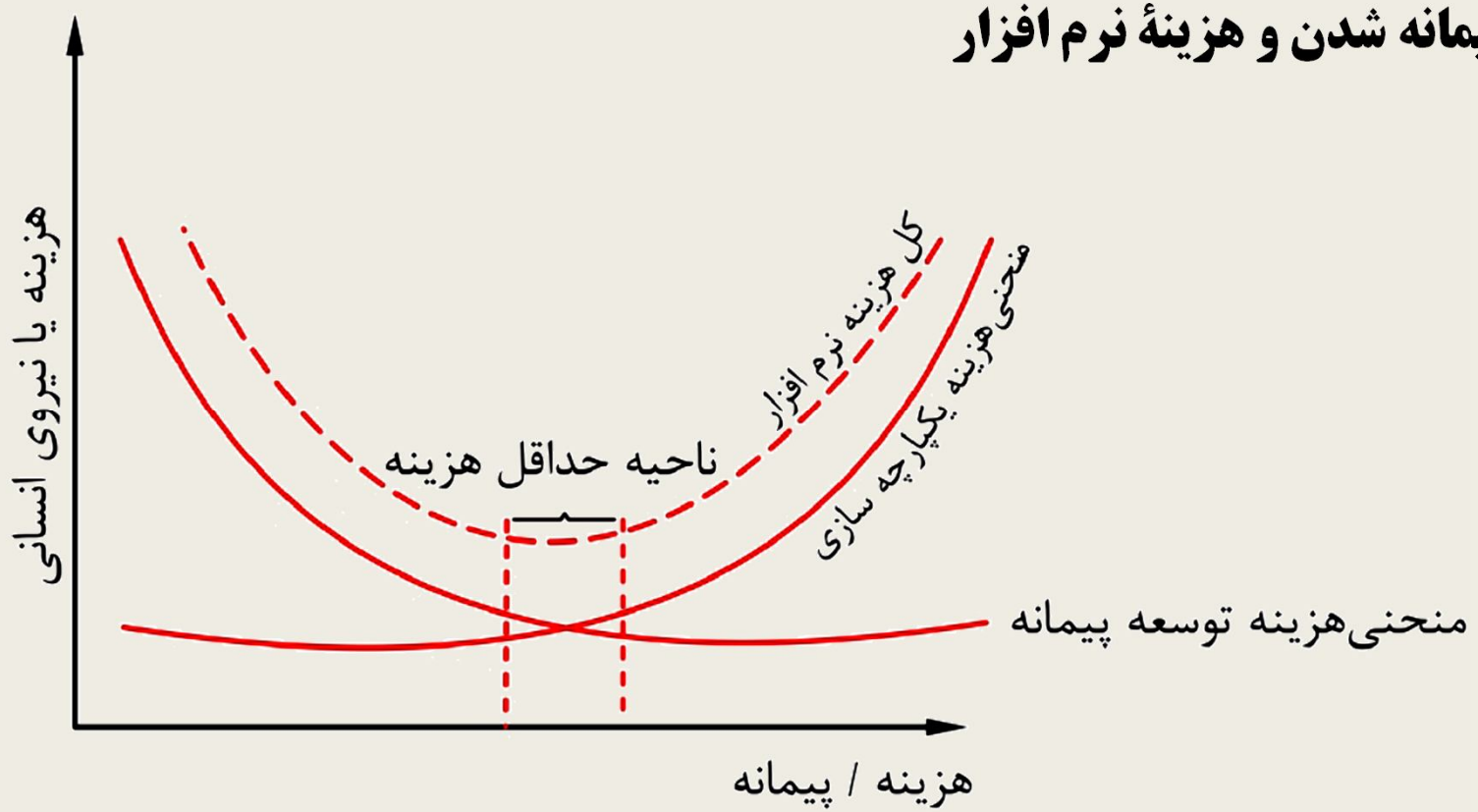
- **پیمانه ای**

معماری یک نرم افزار، وقتی پیمانه ای است که در آن نرم افزار به اجزای نشانی پذیر با اسامی جداگانه به نام "پیمانه ها" تقسیم شده باشد و سپس برای رفع نیازهای مسئله، یکپارچه و مجتمع شوند. با افزایش پیمانه ها هزینه یکپارچگی و درنهایت کل هزینه نرم افزار افزایش می یابد. بنابراین تعیین تعداد پیمانه های مطلوب در طراحی نرم افزار مهم و عموما به صورت شهودی و بر اساس تجربه صورت می گیرد.

قواعد طراحی ۹ گانه

- **پیمانه ای**

پیمانه شدن و هزینهٔ نرم افزار



قواعد طراحی ۹ گانه

- **پیمانه ای**

Meyer پنج معیار را در ارزیابی یک شیوهٔ طراحی و بر اساس توانایی آن در تعریف یک سیستم مؤثر پیمانه ای معرفی می کند:

- **تجزیه پذیری پیمانه ای (کاهش پیچیدگی کل)**
- **قابلیت ترکیب پیمانه ای (استفاده از پیمانه هایی با قابلیت استفاده مجدد)**
- **قابلیت درک پیمانه ای (پیمانه مستقل و قابل تغییر)**
- **پایداری پیمانه ای (قابلیت توسعه)**
- **حفاظت پیمانه ای (قابلیت عدم انتشار خطا)**

قواعد طراحی ۹ گانه

- **معماری نرم افزار**

معماری نرم افزار به "ساختار کلی نرم افزار و راه های ایجاد یکپارچگی ذهنی سیستم از طریق این ساختار" اشاره می کند.

معماری در ساده ترین شکل خود عبارت است از: ساختار سلسله مراتبی اجزای برنامه (پیمانه ها)، شیوهٔ ارتباط این اجزا و ساختار داده هایی که توسط این اجزا مورد استفاده قرار می گیرند.

معماری نرم افزار باید از مدل نیاز استخراج شود.

قواعد طراحی ۹ گانه

سلسله مراتب کنترل در برنامه ها

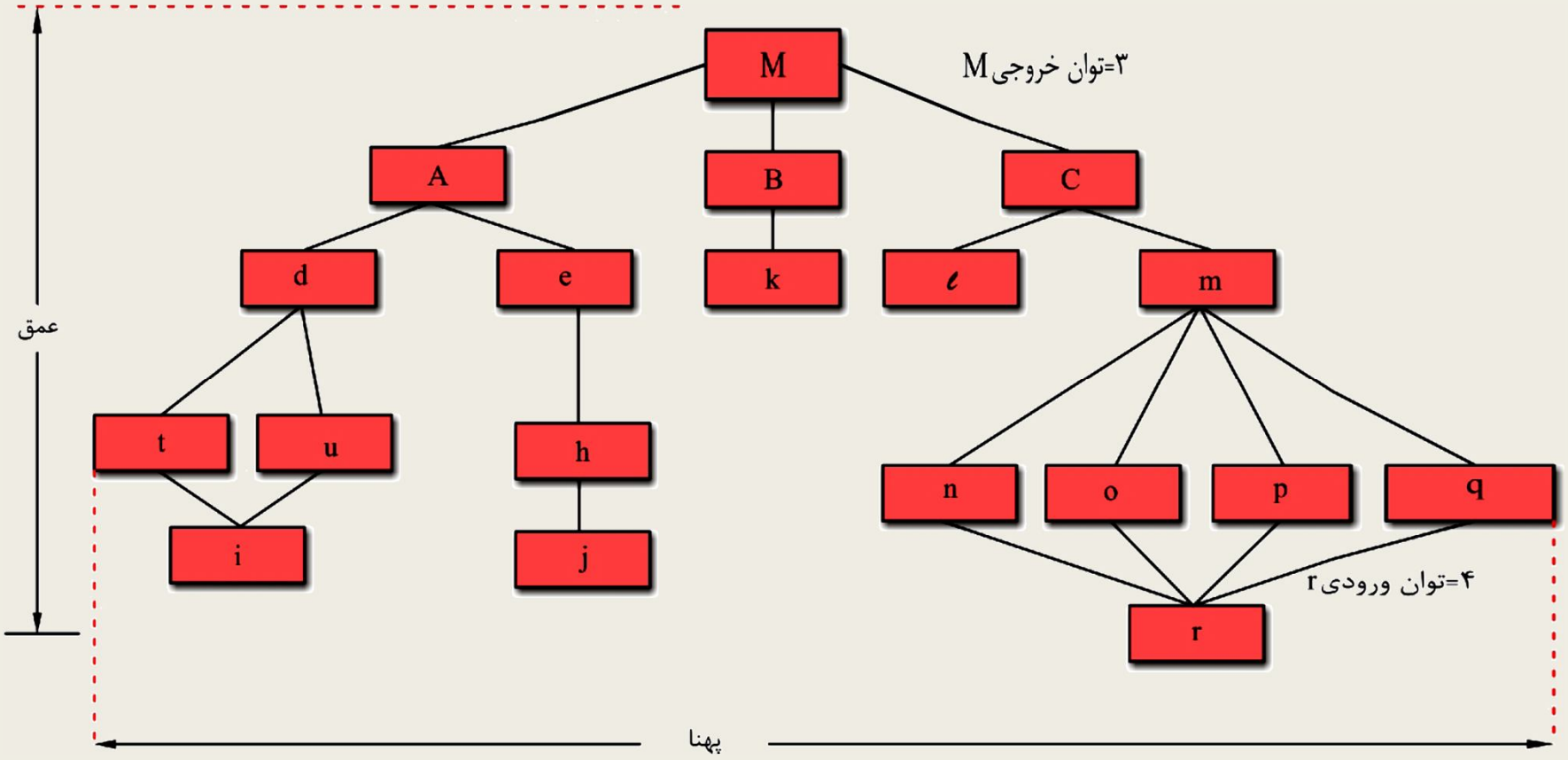
"سلسله مراتب کنترل" که "ساختار برنامه" نیز نام دارد، بیانگر سازماندهی اجزای برنامه (پیمانه ها) بوده و بر سلسله مراتب کنترل نیز دلالت دارد. دو مفهوم زیر در این ارتباط تعریف می شوند:

- **توان خروجی مقیاس تعداد پیمانه هایی است که مستقیماً توسط پیمان های دیگر کنترل می شوند.**
- **توان ورودی بیانگر تعداد پیمانه هایی است که یک پیمانهٔ خاص را به طور مستقیم کنترل می کنند.**
- **رابطهٔ کنترلی میان پیمانه ها به شیوهٔ زیر بیان می شود:**

پیمانه ای که پیمانهٔ دیگری را کنترل می کند، پیمانهٔ حاکم نام دارد و برعکس پیمانهٔ تحت کنترل پیمانهٔ دیگر، پیمانهٔ کنترل شده نامیده می شود.

قواعد طراحی ۹ گانه

- **سلسله مراتب کنترل در برنامه ها**



قواعد طراحی ۹ گانه

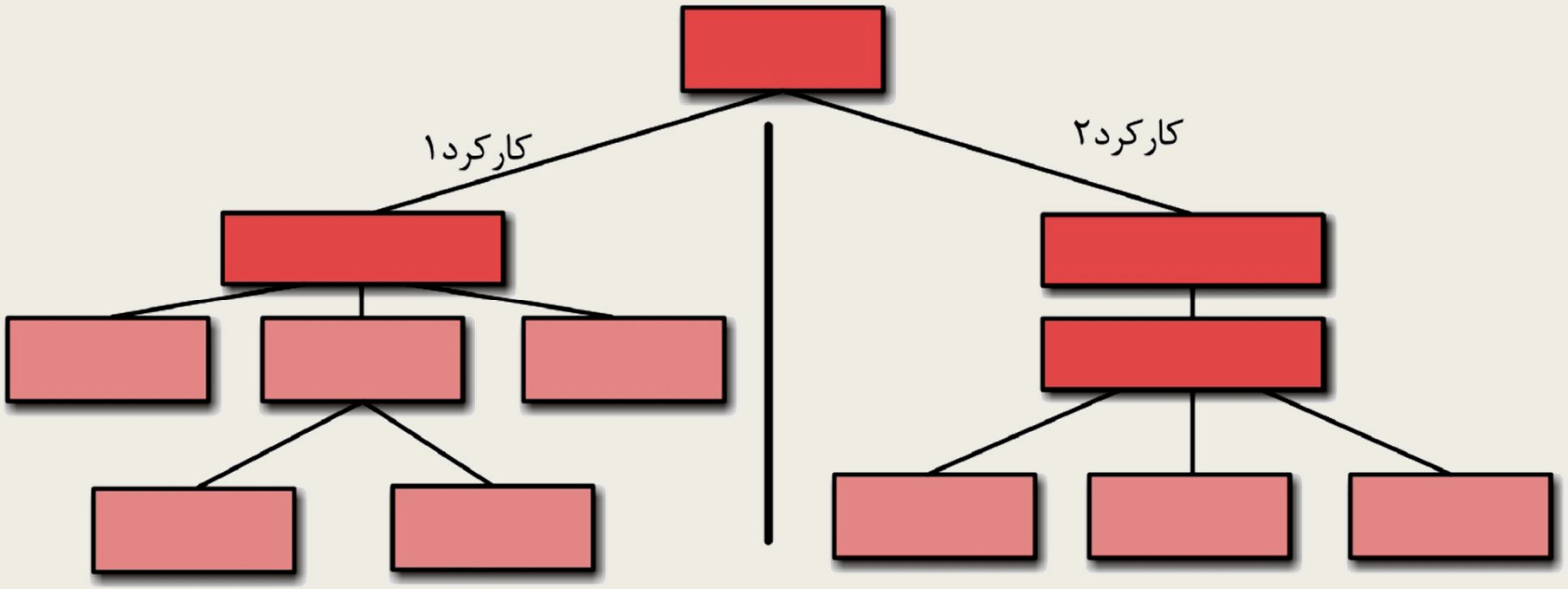
- **تجزیهٔ ساختاری**

اگر سبک معماری یک سیستم، سلسله مراتبی باشد، می توان ساختار برنامه را، هم به صورت افقی و هم به طور عمودی تجزیه و افراز کرد. افراز افقی، شاخه های جداگانهٔ سلسله مراتب پیمانه ای را برای هر یک از وظایف اصلی برنامه تعیین می کند. پیمانه های کنترلی که با سایهٔ تیره تر نشان داده شده اند، برای هماهنگی ارتباط بین وظایف و اجرای آن ها به کار می روند. ساده ترین شیوهٔ افراز افقی، سه بخش ورودی، تغییر و تبدیل داده ها (که اغلب فرایند نام دارد) و خروجی را تعیین می کند.

ازآنجاکه کارکردهای اصلی از یکدیگر جدا می شوند، اعمال تغییرات، پیچیدگی کمتری دارد و بسط و توسعهٔ سیستم بدون تأثیرات جانبی، راحت تر انجام می شود. از جنبهٔ منفی، افراز افقی باعث می شود داده های بیشتری از واسط های پیمانه عبور کرده و کنترل کلی گردش برنامه دشوار و پیچیده شود.

قواعد طراحی ۹ گانه

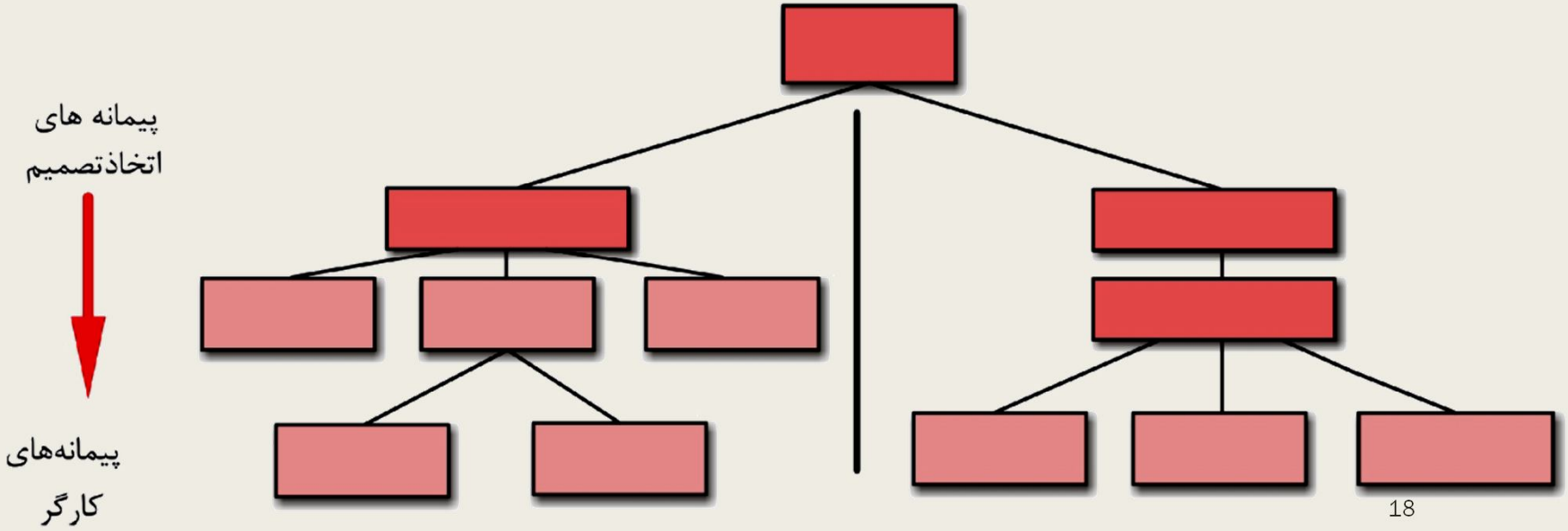
- **تجزیهٔ ساختاری – افراز افقی**



قواعد طراحی ۹ گانه

- **تجزیهٔ ساختاری**

افراز عمودی که فاکتورگیری نیز نامیده می شود، پیشنهاد توزیع کنترل (تصمیم گیری) و کارها را به صورت از بالا به پایین می کند. پیمانه های سطح بالا باید توابع کنترل را اجرا کنند و کار پردازش واقعی کمتری را انجام دهند. پیمانه های سطح پایین متولی اجرای پردازش ها هستند و پیمانه های کارگر نامیده می شوند



قواعد طراحی ۹ گانه

- **ساختمان داده ها**

ساختار داده ها، نمایش رابطهٔ منطقی میان عناصر جداگانهٔ داده هاست.

ازآنجا که ساختار اطلاعات بر طراحی نهایی رویه ای تأثیر خواهد داشت، ساختار داده ها به اندازهٔ ساختار برنامه در نمایش معماری نرم افزار اهمیت دارد.

ساختار داده ها تعیین کنندهٔ نحوه سازماندهی، روش های دستیابی، میزان اتصال و راه های مختلف پردازش اطلاعات است.

قواعد طراحی ۹ گانه

- **پردازش های نرم افزار**

ساختار برنامه، سلسله مراتب کنترل را بدون توجه به توالی پردازش ها و تصمیمات تعیین می کند.

رویهٔ نرم افزار بر جزئیات پردازشی هر پیمانه به طور جداگانه تأکید دارد.

رویه باید مشخصه دقیق پردازش از جمله توالی رویدادها، نقاط دقیق تصمیم گیری، اعمال تکراری و حتی سازماندهی/ ساختار داده ای را ارائه دهد.

قواعد طراحی ۹ گانه

- **پنهان کردن اطلاعات**

مفهوم پیمانه ای بودن، یک سؤال اساسی را برای هر طراح نرم افزار مطرح می کند:

« برای دستیابی به بهترین مجموعهٔ پیمانه ها، چگونه یک راه حل نرم افزاری را تجزیه کنیم؟»

اصل پنهان سازی اطلاعات بیانگر آن است که "وجه مشخصهٔ پیمانه ها، تصمیمات طراحی است که هرپیمانه را از پیمانه های دیگر مخفی می سازد".

به عبارتی دیگر، پیمانه ها باید طوری طراحی و مشخص شوند که اطلاعات (رویه و داده ها) موجود در هر پیمانه برای پیمانه های دیگری که به چنین اطلاعاتی نیاز ندارند، قابل دسترسی نباشند.

فصل هشتم

طراحی معماری نرم افزار

چرا معماری؟

- معماری یک نرم افزار نمودی است که مهندس نرم افزار را قادر می کند:
  - میزان تأثیر طرح را در مرتفع کردن نیاز های بیان شده، تحلیل کند.
  - معماری های جایگزین را در مرحله ای که تغییر طرح هنوز نسبتاً آسان است، بررسی کند.
  - خطرات مربوط به ساخت نرم افزار را کاهش دهد.
- طراحی داده باعث نمایش مؤلفه داده ای معماری و طراحی معماری بر تمرکز نمایش ساختار مؤلفه های نرم افزار، خواص آن ها و ارتباط بین آن ها قرار دارد

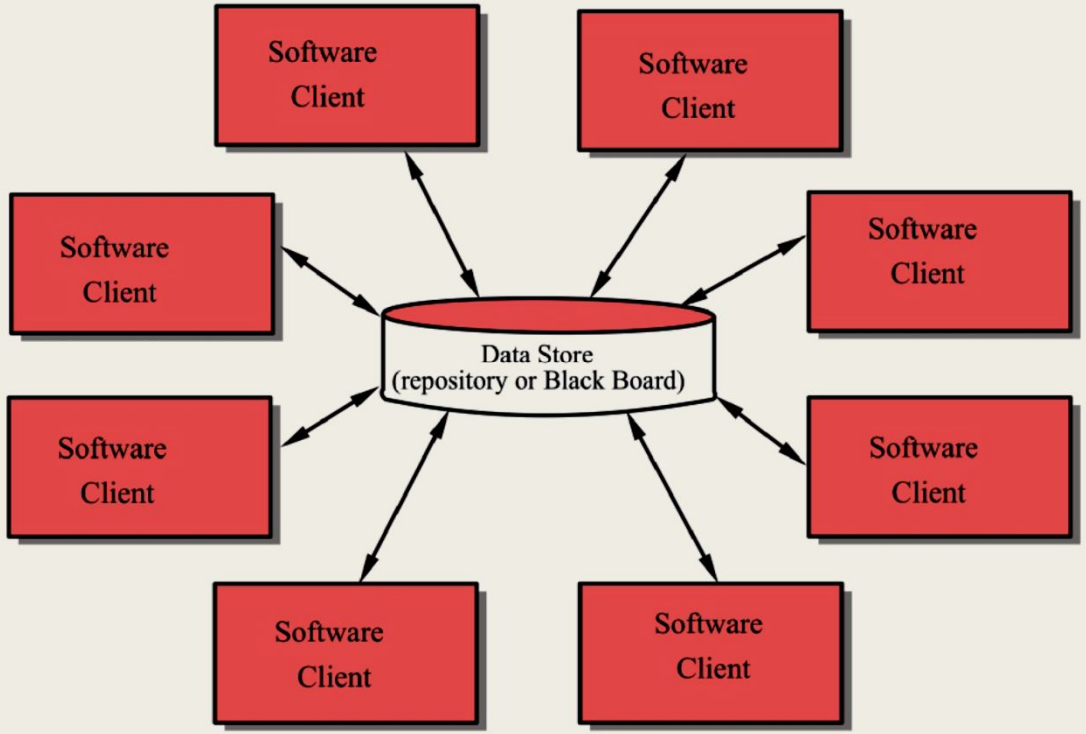
طراحی داده ها

- در بسیاری از برنامه های کاربردی، معماری داده ها تأثیر شگرفی بر معماری نرم افزار دارد که باید آن را پردازش کند؛ بنابراین مدل داده ای:
  - اشیای داده ای را پالایش کرده و مجموع های از انتزاعات داده ای را توسعه می دهد.
  - صفات اشیای داده ای را به عنوان یک یا چند ساختمان داده ای پیاده سازی می کند.
  - ساختمان های داده ای برای اطمینان از ارتباط های مناسب بازنگری می شوند.
  - ساختمان داده ها تا حد ممکن ساده سازی می شود.

سبک های معماری نرم افزار

- **معماری مبتنی بر مخزن داده ها(Data Store Architecture)**

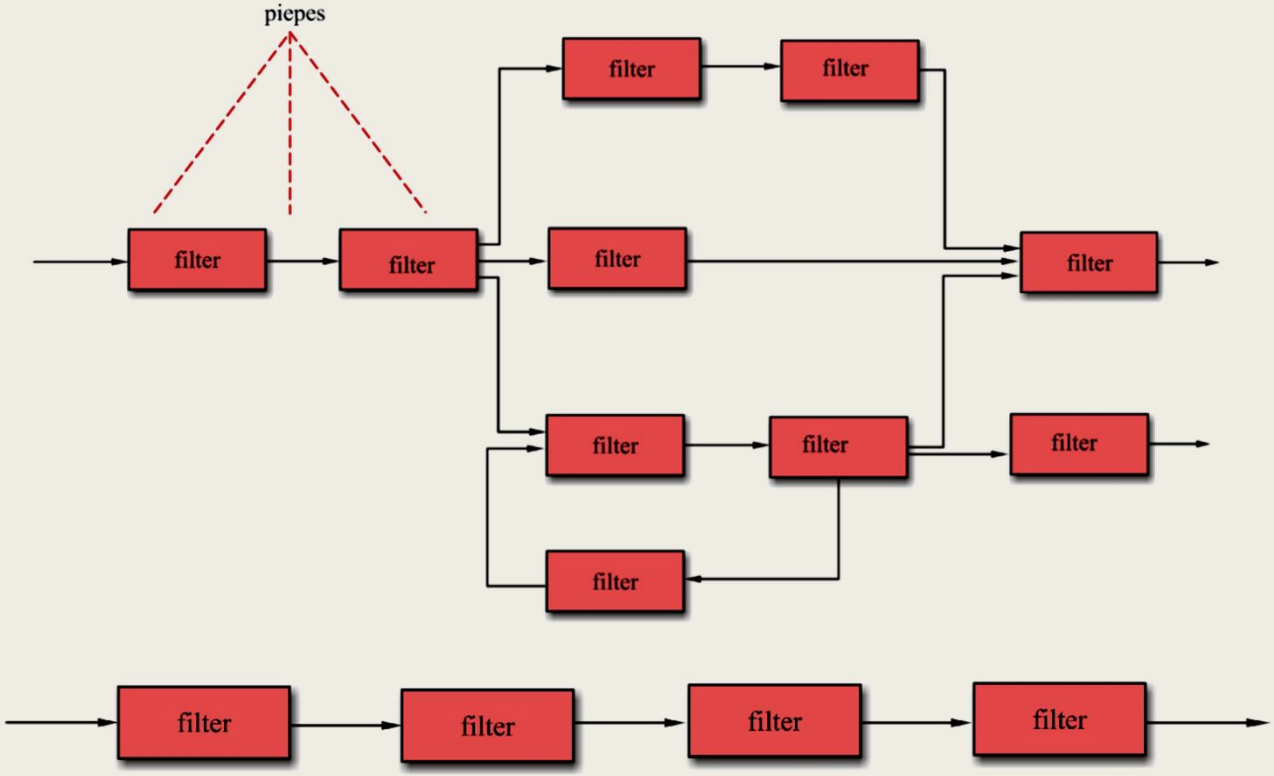
یک مخزن داده ای در مرکز این معماری قرار دارد و اغلب توسط دیگر اجزایی که به هنگام سازی، افزودن، حذف یا کارهای دیگر اصلاحی را در مورد مخزن انجام می دهند، قابل دسترسی است.



سبک های معماری نرم افزار

- **معماری مبتنی بر جریان داده ها(Data Flow Architecture)**

این معماری وقتی به کار گرفته می شود که قرار است داده های ورودی از طریق اجرای یک سری پردازش های محاسباتی و اعمال تغییرات، به داد ههای خروجی تبدیل شوند.



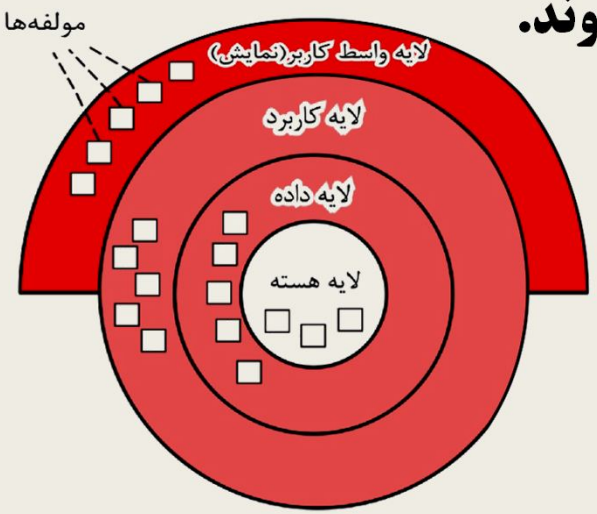
سبک های معماری نرم افزار

- **معماری های شیء گرا(Object-oriented Architecture)**

اجزای یک سیستم دربرگیرندهٔ داده ها و عملیاتی هستند که باید برای تغییر داده ها مورد استفاده قرار گیرند. ارتباط و هماهنگی بین اجزا از طریق عبور پیام ها حاصل می شود.

- **معماری های لایه ای (Layered Architecture)**

تعدادی لایهٔ مختلف تعریف شده اند که هر کدام به عملیاتی دست می یابند که به طور گسترده ای به مجموعه دستورات ماشین نزدیک تر می شوند.



نگاشت نیازها به معماری نرم افزار

- طراحی ساخت یافته اغلب به عنوان یک روش طراحی مبتنی بر جریان داده معرفی می شود؛ زیرا به کمک آن می توان به راحتی از نمودار جریان داده به معماری نرم افزار دست یافت. گذار از جریان داده ها به ساختار برنامه با انجام یک فرایند شش مرحله ای زیر صورت می گیرد:
  - تعیین نوع جریان اطلاعات
  - مشخص کردن مرز جریان
  - نگاشت نمودار جریان داده به ساختمان برنامه
  - تعیین سلسله مراتب کنترل
  - پالایش و بازنگری معماری حاصل با استفاده از معیارها، اصول و قواعد طراحی
  - توصیف و تدوین معماری نهایی

فصل نهم

طراحی واسط کاربر

انواع طراحی واسط

- طراحی واسط بین اجزای نرم افزار
- طراحی واسط ها بین نرم افزار و سایر تولیدکنندگان و مصرف کنندگان اطلاعات غیر انسانی (یعنی اشیای خارجی)
- طراحی واسط بین انسان (یعنی کاربر) و رایانه

در این فصل صرفاً بر سومین مقولهٔ طراحی واسط ها یعنی طراحی واسط کاربر توجه خواهیم داشت.

قواعد طلایی Mendal

این قوانین طلایی، عملاً به عنوان مجموعه ای از Mendal اصول طراحی واسط کاربر مطرح هستند که فعالیت مهم طراحی واسط های نرم افزاری را هدایت می کنند.

- **واگذاری کنترل به کاربر**
  - **تعیین شیوه های تعاملی به نحوی که کاربر را مجبور به اعمال غیر ضروری یا نامطلوب نکند**
  - **ایجاد تعامل انعطاف پذیر در ارتباط با کاربر**
  - **امکان ایجاد وقفه و خنثی سازی (بازگشت) در تعامل با کاربر**
  - **مخفی کردن موارد فنی داخلی از دید کاربران عادی**

قواعد طلایی Mendal

- **کاستن از بار حافظهٔ کاربر**
  - **کاهش بار در حافظهٔ کوتاه مدت**
  - **ایجاد پیش گزیده های معنی دار**
  - **تعیین میان برهای شهودی**
  - **آشکارسازی اطلاعات به شیوه ای تدریجی**
  - **طرح بصری واسط باید بر اساس استعارهٔ جهان واقعی استوار باشد.**

قواعد طلایی Mendal

- **سازگارسازی واسط**
  - **قرار دادن عمل فعلی در یک بافت معن یدار توسط کاربر**
  - **حفظ ثبات در خانوادهٔ برنام ههای کاربردی**
  - **اگر مدل های تعاملی پیشین انتظاراتی را در کاربر به وجود آورده اند، تا زمانی که دلیل قانع کننده ای ندارید از انجام تغییرات خودداری کنید.**

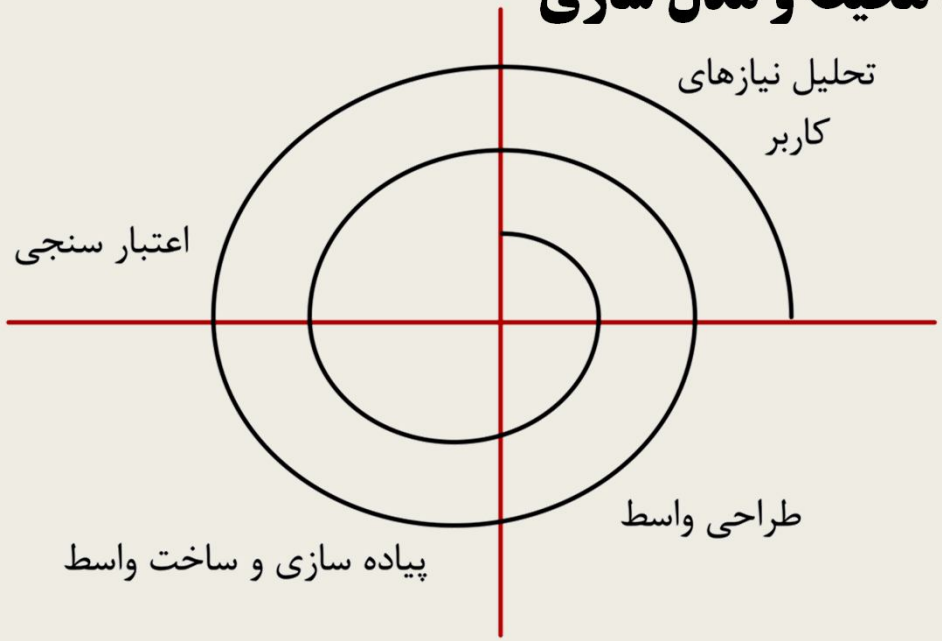
طراحی واسط کاربر

- **مدل های طراحی واسط**
  - **مهندس نرم افزار مدل طراحی را ایجاد می کند. ·**
  - **مهندس فاکتورهای انسانی (یا مهندس نرم افزار) مدل کاربر را تعیین می کند.**
  - **کاربر نهایی یک تصویر ذهنی می سازد که غالباً مدل ذهنی کاربر را به وجود می آورد.**
  - **پیاده کنندگان سیستم نیز یک تصویر سیستم ایجاد می کنند.**

ممکن است هر یک از این مدل ها، تفاوت های قابل ملاحظه ای با یکدیگر داشته باشند. نقش طراح واسط، رفع اختلافات و به دست آوردن یک نمایش منسجم و سازگار از واسط هاست.

فرایند طراحی واسط کاربر

- **فرایند طراحی واسط های کاربر، یک فرایند تکراری است و با استفاده از مدل حلزونی، قابل ارائه است. روند طراحی واسط کاربر شامل چهار فعالیت مجزای زیر است:**
  - **تحلیل نیازهای کاربر، وظایف، محیط و مدل سازی**
  - **طراحی واسط**
  - **پیاده سازی و ساخت واسط**
  - **اعتبارسنجی واسط**



مسایل طراحی واسط ها

- در حین تکمیل طراحی واسط کاربر، چهار مسئله معمول طراحی تقریباً همیشه سطحی تلقی می شوند که باید به آن ها توجه کرد:
  - زمان پاسخگویی سیستم
  - تسهیلات کمکی کاربر
  - خطاگردانی اطلاعات
  - برچسب گذاری فرمان

ابزارهای پیاده سازی واسط ها

- برای انجام روش تکراری طراحی واسط های نرم افزاری، انواع ابزارهای طراحی واسط و ساخت نمونهٔ اولیه وجود دارند. این ابزارها با عنوان سیستم توسعه واسط کاربر (UIDS) نامیده می شوند. این سیستم با استفاده از مؤلفه های نرم افزاری، راهکاری برای موارد زیر فراهم می آورد:
  - مدیریت دستگاه های ورودی (مثل ماوس یا صفحه کلید)
  - اعتبار سنجی ورودی کاربر
  - کنترل خطا و نمایش پیام های خطا
  - فراهم آوردن بازخورد (مثل بازتاب ورودی خودکار)
  - فراهم آوردن راهنما و پیغام
  - کنترل پنجره ها و فیلدها، حرکت در داخل پنجره ها
  - برقراری ارتباط میان نرم افزار کاربردی و واسط ·
  - جدا کردن برنامه کاربردی از عملکردهای مدیریت واسط ·
  - مجاز کردن کاربر به سفارشی کردن واسط

ارزیابی طراحی واسط

- برخی از معیارهای ارزیابی را هنگام بررسی های اولیهٔ طراحی، می توان اعمال کرد:
  - طول و پیچیدگی مشخصات مکتوب سیستم و واسط آن، بیانگر میزان یادگیری لازم توسط کاربران سیستم است.
  - تعداد وظایف تعیین شدهٔ کاربر و میانگین اعمال در هر کار، نشان دهندهٔ زمان محاوره و کارایی کلی سیستم است.
  - تعداد اعمال، وظایف و وضعیت های سیستم که در مدل طراحی تعیین شده، به بار حافظهٔ کاربران سیستم دلالت دارد.
  - روش ارتباط واسط، امکانات کمکی و خطاگردانی، در کل بیانگر پیچیدگی واسط و میزان پذیرش از سوی کاربر است.