فصل ۱۳: اصول و مفاهیم و طراحی نرم‌افزار

طراحی چیست؟ طراحی بازنمایی مهندسی و هدفمند چیزی است که قرار است ساخته شود. طراحی را می‌توان مطابق با خواسته‌های مشتری پیش برد و در عین حال براساس مجموعه معیارهای از پیش تعریف شدهٔ طراحی "خوب"، کیفیت آن را ارزیابی کرد. در زمینهٔ مهندسی نرم‌افزار، طراحی بر چهار کانون اصلی متمرکز است: **داده‌ها، معماری، واسطها، پیمانه‌ها.**

روش‌های طراحی نرم‌افزار از عمق، انعطاف‌پذیری و ماهیت، از ارتباط اندکی با شیوه‌های کلاسیک طراحی مهندسی برخوردارند. با این وجود شیوه‌هایی برای طراحی نرم‌افزار وجود دارند؛ معیارهایی برای کیفیت طراحی در دسترس می‌باشند و می توان از نشان‌گذاری طراحی استفاده کرد.

تعریفی دیگر از طراحی

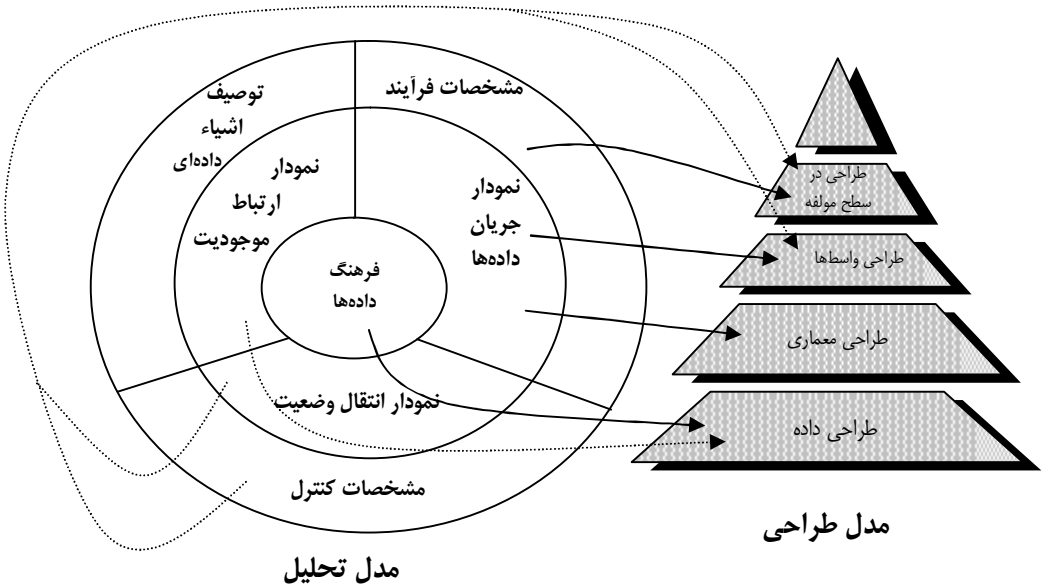
فرایند بکار گیری تکنیک‌ها اصول و قواعد مورد نیاز برای تعریف یک وسیله، یک فرایند، یک پردازش یا یک سیستم تا حد تفصیل است که با واقعیت فیزیکی آن تطبیق داشته باشد. مدل طراحی مبنایی برای پیاده سازی خواهد بود.

هر تغییری در زمان پیاده سازی بایستی روی مدل طراحی و متعاقب آن روی مدل تحلیل نیاز اعمال گردد.

طراحی نرم‌افزار و مهندسی نرم‌افزار

طراحی نرم‌افزار جزء بخش اصلی و فنی مهندسی نرم‌افزار بوده و بدون توجه به مدل به کار رفتهٔ فرآیند نرم‌افزار، اعمال می‌گردد. در آغاز و پس از تحلیل و تعیین نیازمندی‌های نرم‌افزار، طراحی نرم‌افزار با انجام سه فعالیت فنی یعنی طراحی، تولید برنامه، و آزمون، که در ساخت و تأیید نرم‌افزار ضرورت دارند انجام می‌شود. هر یک از این فعالیت‌ها، اطلاعات را به گونه‌ای تغییر می‌دهد که در نهایت به نرم‌افزار معتبر کامپیوتری منجر می‌گردد.

هر یک از عناصر مدل تحلیلی (فصل ۱۱)، اطلاعات لازم را در ایجاد چهار مدل طراحی به منظور تعیین کامل طراحی فراهم می‌آورند.



شکل ۱: برگردان مدل تحلیلی به طراحی نرم‌افزار

طراحی داده‌ها ، مدل اطلاعاتی تولید شده در طول تحلیل را بـه سـاختارهای داده‌ای لازم در اجـرای نـرم‌افـزار تبدیل می‌کند. اشیاء و روابط تعیین شدهٔ داده‌ها در نمودار ارتباط موجودیت‌ها و محتوای مشروح داده‌ای در فرهنـگ داده‌ها، مبنای فعالیت طراحی داده‌ها را فراهم می‌کنند. بخشی از طراحی داده‌ها ممکن اسـت بـا طراحـی معمـاری نرم‌افزار همراه شود. طراحی جزیی‌تر داده‌ها همراه با طراحی هر یک از مولفه‌های نرم‌افزار صورت می‌گیرد.

طراحی معماری ، رابطهٔ بین عناصر اصلی ساختاری نرم‌افزاری را تعیین می‌کند، یعنی رابطهٔ " الگوهای طراحـی" به کار رفته در تحقق نیازمندی‌های تعیین شدهٔ سیستم و محدودیت‌های مؤثر بر شـیوهٔ اجـرای الگوهـای طراحـی معماری نمایش طراحی معماری یا – چارچوب یک سیستم کامپیوتری – حاصل تعیین سیسـتم، مـدل تحلیلـی و ارتباط سیستم‌های فرعی تعیین شده در مدل تحلیلی است.

طراحی واسط ، توصیف کنندهٔ نحوهٔ ارتباط نرم‌افزار در محدودهٔ خود، با سیستم‌هایی کـه بـا آن تعامـل دارنـد و افرادی است که آن را به کار می‌برند. یک واسط بر گردش اطلاعات (مثل داده‌ها و یا کنتـرل) و نـوعی خاصـی از رفتار دلالت دارد. بنابراین نمودارهای جریان داده‌ها و کنتـرل، اکثـر اطلاعـات لازم بـرای طراحـی واسـط را ارایـه می‌کنند.

طراحی مؤلفه ، عناصر ساختاری معماری نرم‌افزار را بـه توصـیف رویـه‌ای مولفـه نـرم‌افـزاری تبـدیل مـی‌کنـد. اطلاعات به دست آمده از STD, CSPEC, PSPEC پایه و اساس طراحی مؤلفه به شمار می‌رونـد. اهمیـت طراحی نرم‌افزار را تنها با یک کلمه یعنی " کیفیت " می‌توان بیان کرد. طراحی روندی است که طـی آن کیفیـت در مهندسی نرم‌افزار، بهبود می‌یابد. طراحی، نمونه‌هایی از نرم‌افزار را که از لحاظ کیفی قابل ارزیابی هستند، در اختیـار ما قرار می‌دهد. طراحی تنها راهی است که به واسطهٔ آن می‌تـوانیم بـه طـور صـحیح نیازمنـدیها و خواسـته‌هـای مشتری را به یک محصول نرم‌افزاری یا سیستم تکمیل شده تبدیل کنیم.

طراحی نرم‌افزار یک فرآیند تکراری است که به موجب آن نیازمندی‌ها و ضرورت‌ها برای ساخت نـرم‌افـزار، تبـدیل به یک " طرح یا نقشه " می‌شوند. در ابتدا، طرح یک دید کلی از نرم‌افزار را نشان می‌دهـد. یعنـی آنکـه طراحـی در سطح بالایی از انتزاع ارائه می‌شود. با انجام تکرار در طراحی، پالایش بعدی، به نمـایش طراحـی در سـطوح بسـیار پایین‌تر انتزاع منجر می‌گردد. این سطوح نیز برای دستیابی به نیازمندی‌ها قابل ردیابی است، امـا ایـن نـوع ارتبـاط ظریف‌تر می‌باشد.

راهنمای ارزیابی یک طراحی خوب به شرح زیر ارایه شده‌اند:

- طراحی باید همهٔ ضرورت‌های آشکار موجود در مدل تحلیل را تحقق بخشیده و با تمامی خواسـته‌هـا و نیازهای ضمنی و مطلوب مشتری سازگار باشد.
- طراحی باید برای کسانی که برنامه‌نویسی می‌کنند و نیز اشخاصی که نرم‌افزار را آزمون نموده و بعـداً آن را پشتیبانی می‌کنند، راهنمایی خوانا و قابل درک باشد.
- طراحی باید تصویر کاملی از نرم‌افزار را ارائه کرده و حوزه‌های داده‌ای، کـاربردی و رفتـاری را از دیـد پیاده‌سازی مورد توجه قرار دهد.

برای ارزیابی کیفیت یک طراحی خوب باید:

- طراحی باید یک ساختار معماری ارایه کند که (۱) با استفاده از الگوهای قابل تشخیص طراحـی ایجـاد شده باشد. (۲) متشکل از اجزاء و عناصری باشد که خصوصیات طراحی خوب را نشان می‌دهنـد (بعـداً در این فصل مورد بحث قرار می‌گیرند) و (۳) به شیوه‌ای تکـاملی اجـرا شـده و بـدین ترتیـب اجـرا و آزمون را تهسیل کند.
- طراحی باید پیمانه‌ای (ماژولار) باشد یعنی نرم‌افزار باید به طور منطقـی بـه اجزایـی تقسـیم شـود کـه اعمال اصلی و فرعی خاصی را انجام دهند.
- طراحی بایستی نمایش‌های مجزایی از داده‌ها، معماری، واسطها و اجزاء (پیمانه‌ها) را در برداشته باشد.

- طراحی باید به ساختارهای داده‌ای منجر شود که برای پیاده‌سازی اشیاء مناسب بوده و از الگوهای قابل تشخیص داده‌ها ناشی می‌شوند.
- طراحی باید به اجزایی منتهی گردد که خصوصیات مستقل کاربردی را نمایش دهند.
- طراحی باید به واسطهایی ختم شود که از پیچیدگی روابط بین پیمانه‌ها و محیط خارجی می‌کاهند.
- طراحی باید حاصل کاربرد یک شیوهٔ قابل تکرار با استفاده اطلاعات به دست آمده در طول تحلیل نیازمندیهای نرم‌افزاری باشد.

اصول طراحی

طراحی نرم‌افزار، هم یک فرآیند است و هم یک مدل. داویس [DAV95] مجموعه اصولی را برای طراحی نرم‌افزار پیشنهاد می‌کند که این اصول به شرح زیر ارایه می‌شوند:

- **باریک بینی نباید در فرآیند طراحی وجود داشته باشد.** یک طراح خوب بایستی رهیافت‌های دیگر را در نظر داشته باشد و هر یک را براساس ضرورت‌های مساله، منابع موجود برای انجام کار و مفاهیم طراحی مورد قضاوت قرار دهد.
- **طراحی باید قابل ردیابی به مدل تحلیل باشد.** از آن جا که هر یک از عناصر مدل طراحی اغلب قابل ردیابی به ضروریات چندگانه است، بنابراین وجود روشی برای پیگیری چگونگی رفع نیازمندی‌ها در مدل طراحی لازم است.
- **طراحی نباید دوباره‌کاری باشد.** سیستم‌ها با استفاده از مجموعه الگوهای طراحی ساخته می‌شوند که احتمالاً با خیلی از آنها قبلاً هم برخورد داشته‌اند. این الگوها باید همواره به عنوان گزینه دیگر دوباره‌کاری انتخاب گردند. منابع محدودند! لذا زمان طراحی باید صرف ارایه نظرات واقعاً جدید و یکپارچه کردن الگوهای از قبل موجود شود.
- **طراحی باید فاصلهٔ منطقی بین نرم‌افزار و مساله موجود در جهان واقعی را به حداقل برساند.** یعنی، ساختار طراحی نرم‌افزار باید (در صورت امکان) ساختار میدان مساله را تقلید نماید.
- **طراحی باید از یکنواختی و یکپارچگی برخوردار باشد.** اگر این طور به نظر برسد که توسعهٔ کل کار برعهدهٔ یک شخص بوده، در این صورت طراحی یکسان و یکنواخت است. قبل از شروع کار طراحی، قوانین، سبک و قالب‌ها باید برای تیم طراحی تعریف و تعیین گردد. توجه و دقت در تعیین واسطهای بین اجزای طراحی، یکپارچگی طراحی را به دنبال خواهد داشت.
- **ساختار طراحی باید پذیرای تغییر باشد.** مفاهیم مورد بحث طراحی در بخش بعدی، باعث تطابق آن با این اصل می‌باشند.
- **ساختار طراحی حتی در صورت مواجهه با داده‌های غیرعادی، رویدادها یا شرایط کاری، باید به آرامی از کار بایستد.** یک نرم‌افزار خوب طراحی شده نباید هرگز ناگهان متوقف گردد. بلکه باید طراحی آن به گونه‌ای باشد که با شرایط غیرعادی سازگار بوده و اگر قرار شد پردازشی را پایان دهد این کار به طور ملایم انجام دهد.
- **طراحی به معنای برنامه‌نویسی نیست و برنامه‌نویسی نیز معادل طراحی نمی‌باشد.** حتی پس از ایجاد طراحی‌های جزیی رویه‌ای برای اجزای برنامه، سطح انتزاع مدل طراحی بالاتر از برنامهٔ منبع است. تنها تصمیمات طراحی اتخاذ شده در سطح برنامه‌نویسی، جزییات ظریف پیاده‌سازی را که موجب برنامه‌نویسی طراحی رویه‌ای می‌شوند را مطرح می‌کند.
- **طراحی باید ضمن شکل‌گیری، از نظر کیفی مورد ارزیابی قرار گیرد نه بعد از اتمام.** مجموعه‌ای از مفاهیم طراحی (فصل قبل) و اقدامات طراحی برای کمک به طراح به منظور ارزیابی کیفی، در دسترس می‌باشند.

❖ **طراحی باید به منظور به حداقل رساندن خطاهای مفهومی (معنایی) مرور و بررسی شود.**
گاه هنگام بررسی و مرور طراحی، تمایل به تأکید روی جزییات وجود دارد. یعنی در جنگل فقط به دنبال درخت بودن. تیم طراحی بایستی قبل از نگرانی دربارهٔ مدل طراحی، تضمین کند که عناصر اصلی مفهومی در طراحی (حذف و از قلم افتادگی، ابهام و ناسازگاری) مورد توجه قرارگرفته و رفع و رجوع گشته‌اند.

قواعد طراحی

قواعد طراحی که در ادامه بحث ارایه شده است، در پاسخ سوالات زیر کمک می نماید:

- چه معیاری برای نرم افزار به مولفه های مجزا چیست ؟
- چگونه کار کردها یا ساختار داده ها از نمایش مفهومی نرم افزار مجزا می شود ؟
- آیا معیار یکنواختی وجود دارد که کیفیت طراحی نرم افزار را تعریف کند ؟
- هدف از نوشتن برنامه کار کردن آن و یا درست کار کردن آن است؟

GETTING A PROGRAM TO WORK OR GETTING IT RIGHT?

- هدف از نوشتن برنامه درست کار کردن است و برای اینکار :

خوب طراحی شود ⟵ تحلیل نیاز درست ⟵ تحلیل سیستم درست

انتزاع

وقتی برای هر مساله به دنبال راه‌حل پیمانه‌ای باشیم، بسیاری از سطوح انتزاع نیز مطرح می‌گردند. در بالاترین سطح انتزاع، راه‌حل با بکارگیری زبان محیط مسئله و به صورت کلی بیان می‌شود. در سطوح پایین‌تر انتزاع، جهت‌گیری و گرایش بیشتر رویه‌ای است. اصطلاحات مساله‌گرا در تلاش برای بیان یک راه‌حل با اصطلاحات اجرایی تلفیق می‌شوند. و نهایت این که در پایین‌ترین سطح انتزاع، راه‌حل به گونه‌ای بیان می‌شود که مستقیماً قابل اجرا باشد.

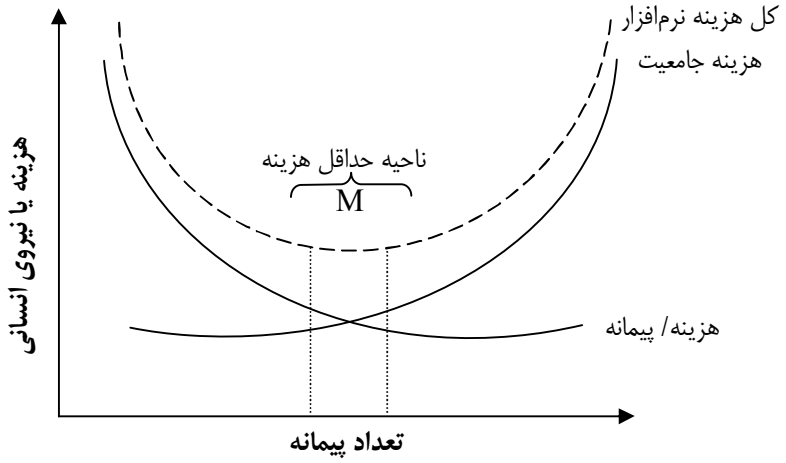
حین پیشروی در سطوح مختلف انتزاع، تلاش می‌کنیم تا انتزاع‌های رویه‌ای و داده‌ها ایجاد گردد. انتزاع رویه‌ای توالی مشخصی از دستورالعمل‌ها است که عملکرد خاص و محدودی دارد. نمونه‌ای از یک انتزاع رویه‌ای، وجود کلمهٔ " Open " برای یک " Door " می‌باشد. Open بر زنجیرهٔ طولانی مراحل رویه‌ای دلالت دارد: (مثلاً رفتن به سمت در، یا بردن دست و گرفتن دستگیره، چرخاندن دستگیره و کشیدن در، فاصله گرفتن از در و غیره)

انتزاع کل سومین شکل انتزاعی است که در طراحی نرم‌افزار به کار می‌رود. همانند انتزاع رویه‌ای و داده‌ای، انتزاع کل بر مکانیزم کنترل برنامه بدون مشخص کردن جزییات داخلی اشاره دارد. نمونهٔ انتزاع کنترل "سمافور همزمانی" است که برای هماهنگ کردن فعالیت‌ها در سیستم عامل به کار می‌رود.

پالایش

پالایش گام به گام یک راهبرد طراحی بالا به پایین است. برنامه با سطوح پالایشی متوالی جزییات رویه‌ای، توسعه می‌یابد. تا زمان دستیابی به دستورات زبان برنامه‌نویسی، ایجاد توسعهٔ سلسله مراتب با تجزیهٔ دستور ماکروسکوپی عمل (انتزاع رویه‌ای) بر شیوه‌ای گام به گام صورت می‌گیرد. دید کلی این مفهوم توسط Wirth ارایه می‌شود:

مفهوم پیمانه‌ای در نرم‌افزار کامپیوتر تقریباً پنج دهه، مورد حمایت واقع شده است. معماری نرم‌افزاری پیمانه‌ای است؛ نرم‌افزار بر اجزای نشانی پذیر با اسامی جداگانه به نام " پیمانه‌ها " تقسیم می‌شود که برای رفع نیازهای مساله، یکپارچه و مجتمع می‌باشند.



شکل۲: پیمانه شدن و هزینه نرم‌افزار

Meyers پنج معیار را در ارزیابی یک شیوهٔ طراحی و براساس توانایی آن در تعریف یک سیستم مؤثر پیمانه‌ای معرفی می‌کند:

- **تجزیه‌پذیری پیمانه‌ای:** اگر شیوهٔ طراحی مکانیزم منظمی را برای تجزیهٔ مساله به مسایل فرعی ارایه دهد، در آن صورت پیچیدگی مساله کلی کاهش یافته و بدین ترتیب تحقق یک راه‌حل مؤثر پیمانه‌ای میسر می‌گردد.
- **قابلیت ترکیب پیمانه‌ای:** اگر یک شیوهٔ طراحی امکان هم‌گذاری اجزای موجود (قابل استفاده مجدد) طراحی را در یک سیستم جدید به وجود بیاورد، آن‌گاه یک راه‌حل پیمانه‌ای به دست خواهد آمد که دوباره‌کاری نخواهد داشت.
- **قابلیت درک پیمانه‌ای:** اگر پیمانه‌ای به عنوان یک پیمانه مستقل قابل درک باشد (بدون ارجاع به پیمانه‌های دیگر) ساختن و تغییر آن آسان‌تر خواهد بود.
- **استمرار پیمانه‌ای:** اگر تغییرات کوچک در نیازمندی‌های سیستم، بیش از تغییر سیستم، منجر به تغییرات در پیمانه‌های جداگانه شود، تأثیر اثرات جانبی حاصل از تغییر به حداقل خواهد رسید.
- **محافظت پیمانه‌ای:** اگر در یک پیمانه شرایط غیرعادی پیش بیاید و تأثیرات آن به همان پیمانه محدود شود، تأثیر اثرات جانبی ناشی از خطا، به حداقل خواهد رسید.

معماری نرم‌افزار

آیا معماری نرم افزار بایستی از مدل نیاز استخراج گردد؟

معماری نرم‌افزار به " ساختار کلی نرم‌افزار و راه‌های ایجاد یکپارچگی ذهنی سیستم از طریق این ساختار " اشاره می‌کند. **معماری در ساده‌ترین شکل خود عبارت است از ساختار سلسله مراتبی اجزاء برنامه (پیمانه‌ها)، شیوهٔ ارتباط این اجزاء و ساختار داده‌هایی که توسط اجزاء مورد استفاده قرار می‌گیرند.**

یکی از اهداف طراحی نرم‌افزار، نقشهٔ معماری سیستم است. این نقشه چارچوب و مبنایی است که فعالیت‌های جزیی‌تر طراحی براساس آن انجام می‌گیرند. مجموعه الگوهای معماری، مهندس نرم‌افزار را قادر می‌سازد تا مفاهیم سطح طراحی را مجدداً قابل استفاده نماید.

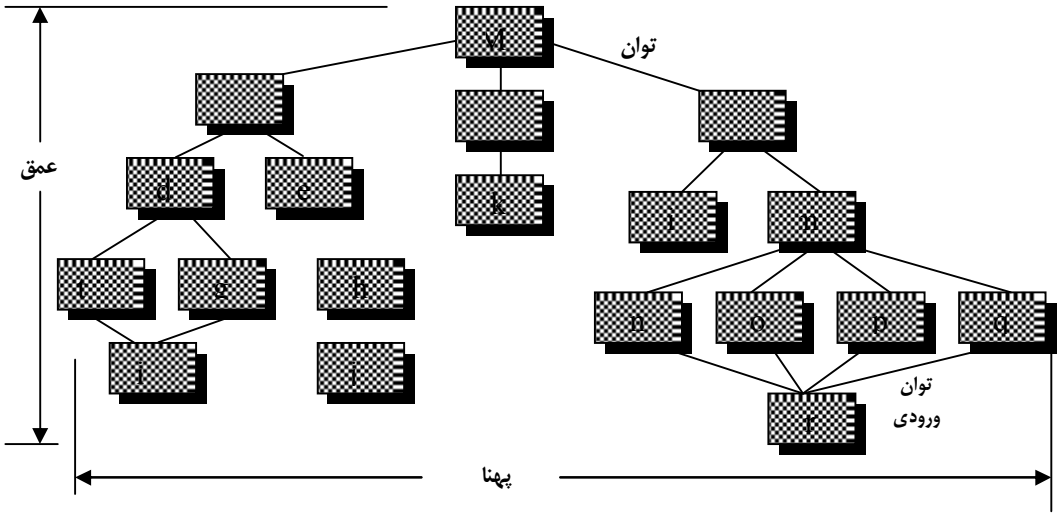
سلسله مراتب کنترل در برنامه ها

"**سلسله مراتب کنترل**" که "**ساختار برنامه**" نیز نام دارد، بیانگر سازمان‌دهی اجزاء برنامه (پیمانه‌ها) بوده و بر سلسله مراتب کنترل دلالت دارد. سلسله مراتب کنترل نشانهٔ جنبه‌های رویه‌ای نرم‌افزار مثل توالی فرآیندها، وقوع/ ترتیب تصمیمات یا تکرار عملکردها نبوده و لزوماً در تمامی سبک‌های معماری قابل کاربرد نمی‌باشد.

در آن دسته از سبک‌های معماری که قابل نمایش هستند، نشان‌گذاری‌های مختلفی برای نمایش سلسله مراتب کنترل به کار می‌روند.

توان خروجی مقیاس تعداد پیمانه‌هایی است که مستقیماً توسط پیمانه‌ای دیگر کنترل می‌شوند. **توان ورودی** نیز بیانگر تعداد پیمانه‌هایی است که یک پیمانه خاص را به طور مستقیم کنترل می‌کنند.

رابطهٔ کنترلی میان پیمانه‌ها به شیوهٔ زیر بیان می‌شود. پیمانه‌ای که پیمانه دیگری را کنترل می‌کند، **پیمانه حاکم** نام دارد و برعکس پیمانه تحت کنترل پیمانه دیگر، تابع **پیمانه کنترل کننده** می‌باشد.



شکل۳: ساختار سلسله مراتبی پیمانه‌ها در برنامه

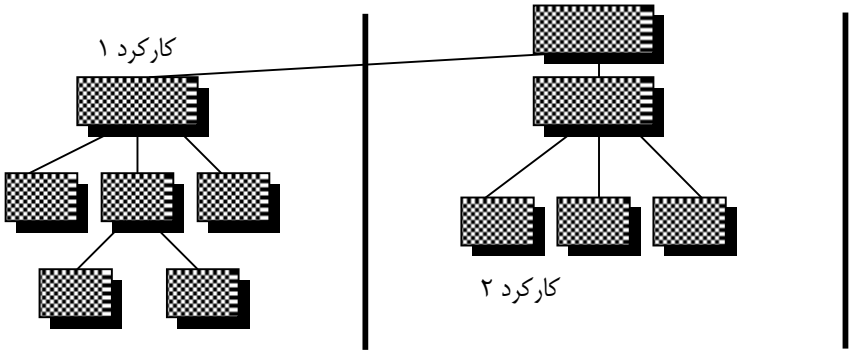
تجزیه ساختاری

اگر سبک معماری یک سیستم، سلسله مراتبی باشد، می‌توان ساختار برنامه را هم به صورت افقی و هم به طور عمودی تقسیم‌بندی کرد. با توجه به شکل۴ الف، تقسیم‌بندی افقی، شاخه‌های جداگانهٔ سلسله مراتب پیمانه‌ای را برای هر یک از وظایف اصلی برنامه تعیین می‌کند. پیمانه‌های کنترلی که با سایهٔ تیره‌تر نشان داده شده‌اند، برای هماهنگی ارتباط بین وظایف و اجرای آنها به کار می‌رود. ساده‌ترین شیوهٔ تقسیم‌بندی افقی، سه بخش را تعیین می‌کند که عبارتند از: ورودی، تغییر و تبدیل داده‌ها (که اغلب فرآیند نام دارد) و خروجی.

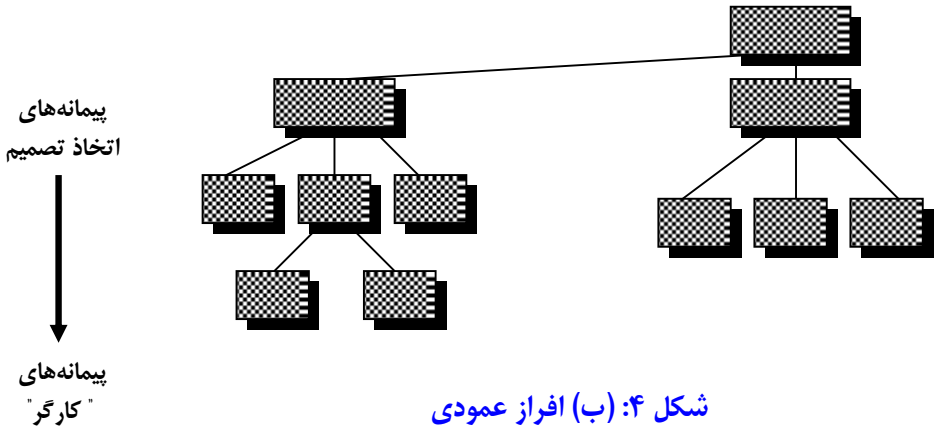
تقسیم‌بندی معماری به صورت افقی، مزایای روشنی را به همراه دارد.

- منجر به نرم‌افزاری می‌شود که آزمون آن آسان‌تر است.
- منجر به نرم‌افزاری می‌شود که نگهداری و حفظ آن آسان‌تر است.
- منجر به نرم‌افزاری می‌شود که انتشار آن از پیامدهای جنبی کمتری برخوردار است.
- نرم‌افزاری را ایجاد می‌کند که بسط و توسعهٔ آن ساده‌تر است.

از آن جا که کارکرد اصلی از یکدیگر جدا می‌گردند، تغییرات پیچیدگی کمتری دارند و بسط و توسعهٔ سیستم (که امری رایج است) بدون تأثیرات جانبی، راحت‌تر انجام می‌شود. از جنبهٔ منفی، تقسیم‌بندی افقی باعث می‌شود داده‌های بیشتری از واسطهای پیمانه عبور کرده و کنترل کلی گردش برنامه را دشوار و پیچیده می‌سازد(در صورتی که فرآیند مستلزم جابه‌جایی سریع از یک کارکرد به کارکردی دیگر باشد).



شکل ۴ : (الف) افراز عمودی



شکل ۴: (ب) افراز عمودی

تعریف ساختمان داده ها

ساختار داده‌ها، نمایش رابطهٔ منطقی میان عناصر جداگانهٔ داده‌ها است. از آن جا که ساختار اطلاعات بر طراحی نهایی رویه‌ای تأثیر خواهد داشت، ساختار داده‌ها به اندازهٔ ساختار برنامه در نمایش معماری نرم‌افزار اهمیت دارد.
ساختار داده‌ها تعیین کنندهٔ سازمان‌دهی، روش‌های دستیابی، میزان شرکت‌پذیری و جایگزین‌های فرآیند اطلاعات می‌باشد.

تعریف پردازش‌های نرم‌افزار

ساختار برنامه، سلسله مراتب کنترل را بدون توجه به توالی فرآیند و تصمیمات تعیین می‌کند. رویهٔ نرم‌افزار بر جزییات پردازشی هر پیمانه به طور جداگانه تأکید دارد. رویه باید تعیین دقیق پردازش از جمله توالی رویدادها، نقاط دقیق تصمیم‌گیری، اعمال تکراری و حتی سازمان‌دهی/ ساختار داده‌ای را ارایه دهد.
البته، بین ساختار و رویه ارتباطی وجود دارد. فرآیند تعیین شده برای هر پیمانه، باید ارجاع به تمامی پیمانه‌های تابع آن را در بر داشته باشد.

عدم نمایش اطلاعات (پنهان کردن کردن)(Information Hiding)

مفهوم پیمانه‌ای بودن یک سؤال اساسی را برای هر طراح نرم‌افزاری مطرح می‌کند. «برای دستیابی به بهترین مجموعهٔ پیمانه‌ها، چگونه یک راه‌حل نرم‌افزاری را تجزیه کنیم؟» اصل پنهان‌سازی اطلاعات(Information Hiding) بیانگر آن است که "وجه مشخصهٔ پیمانه‌ها، تصمیمات طراحی است که هر پیمانه از پیمانه‌های دیگر مخفی می‌سازد". به عبارتی دیگر، پیمانه‌ها باید طوری طراحی و مشخص شوند که اطلاعات (رویه و داده‌ها) موجود در هر پیمانه برای پیمانه‌های دیگری که به چنین اطلاعاتی نیاز ندارند، غیر قابل دسترسی باشد. بنابراین:

همه اطلاعات را به همه کاربران نباید نشان داد.

طراحی پیمانه های کارآمد

- **ویژگی پیمانه ای(modular) یک روش پذیرفته شده در کلیه رشته های مهندسی است.**
- **تاثیرات طراحی پیمانه‌ای:**
  - کاهش پیچیدگی
  - تسهیل تغییرات
  - امکان توسعه موازی مولفه های مختلف و در نتیجه آن تسهیل پیاده سازی
- **سه مفهوم قابل توجه در طراحی پیمانه‌ها:**
  - استقلال عملیاتی (Functional Independence)
  - انسجام (Cohesion)
  - پیوستگی (Coupling)

استقلال عملیاتی

مفهوم " استقلال عملیاتی " پیامد مستقیم پیمانه ساختن و مفاهیم انتزاع و پنهان‌سازی اطلاعات است. استقلال عملیاتی از طریق ایجاد پیمانه‌هایی با عملکرد یک منظوره و عدم ارتباط بیش از حد با پیمانه‌های دیگر، تحقق می‌یابد. به بیان دیگر، ما قصد داریم نرم‌افزاری را طراحی کنیم که هر پیمانه یک وظیفهٔ خاص فرعی از ضرورت‌ها را انجام داده و از دید سایر بخش‌های ساختار برنامه، واسط ساده‌ای داشته باشد.

توسعهٔ نرم‌افزاری با پیمانه‌ای شدن کارآمد یعنی پیمانه‌های مستقل، آسان‌تر است زیرا کار تقسیم‌بندی شده و واسطها ساده شده‌اند. (انشعابات را هنگام انجام توسعه توسط یک تیم، در نظر بگیرید.) نگهداری و آزمون پیمانه‌های مستقل ساده‌تر است زیرا تأثیرات ثانویه ایجاد شده به واسطهٔ تغییر طراحی/ برنامه، محدود می‌شوند. انتشار خطا کاهش می‌یابد و پیمانه‌هایی با قابلیت استفادهٔ مجدد امکان‌پذیر می‌گردند. به طور خلاصه، استقلال عملیاتی رمز طراحی خوب و طراحی، رمز کیفیت نرم‌افزار است. استقلال با دو معیار کیفی ارزیابی می‌گردد: **انسجام و اتصال**

انسجام" قدرت عملیاتی نسبی یک پیمانه است و اتصال مقیاس وابستگی نسبی پیمانه‌ها به هم می‌باشد.

چرا استقلال دارای اهمیت است؟

- **توسعه آسانتر**
- **تسهیل درنگهداری و آزمایش**
- **کاهش انتشار خطا**
- **قابلیت استفاده مجدد**

انسجام(Cohesion)

انسجام یا چسبندگی، بسط طبیعی مفهوم پنهان‌سازی اطلاعات است. یک پیمانه یکپارچه، یک وظیفه منفرد را در یک رویه نرم‌افزاری و با برقراری ارتباط محدود با رویه‌های در حال اجرای سایر بخش‌های برنامه، انجام می‌دهد. به بیان ساده‌تر، یک پیمانه منسجم (به طور ایده‌آل) باید تنها یک کار را انجام دهد.

انسجام را می‌توان به صورت یک " طیف " نشان داد. ما همیشه تلاش می‌کنیم تا به **انسجام و پکپارچگی زیاد** دست یابیم. هر چند که دامنهٔ متوسط طیف نیز اغلب قابل قبول است. مقیاس انسجام، غیرخطی است. یعنی آن که، انسجام انتهای پایانی به مراتب بدتر از دامنهٔ متوسط (اواسط طیف) است که تقریباً به اندازهٔ انسجام انتهای بالایی، قابل قبول می‌باشد. عملاً لزومی ندارد که طراح به طبقه‌بندی انسجام در یک پیمانه خاص بپردازد، بلکه باید مفهوم کلی را درک نموده و هنگام طراحی پیمانه‌ها بایستی از سطوح پایین انسجام اجتناب کرد.

انواع پیمانه‌ها از نظر انسجام

- تصادفا منسجم (Coincidentally Cohesive)

در انتهای پایینی (نامطلوب) طیف, با پیمانه‌های سرو کار داریم که مجموعه ای از وظایف را انجام می‌دهد که ارتباط محکمی با هم ندارند.

- منطقا منسجم (Logically Cohesive)

اگر پیمانه ای وظایفی انجام دهد که با هم ارتباط منطقی دارند (مثل پیمانه هایی که خروجی ها را بدون در نظر گرفتن نوع آنها تولید می‌کنند), آن پیمانه را منطقا منسجم گویند.

- انسجام موقتی (Temporal Cohesive)

هنگامی که پیمانه ای حاوی وظایفی است که باید در یک گستره زمانی مشترک اجرا شوند, آن پیمانه دارای انسجام موقتی است. مثال

- انسجام رویه ای (Procedural Cohesive)

سطوح میانه ای از انسجام از لحاظ استقلال پیمانه ها نسبتا به هم نزدیکند. هنگامی که عناصر پردازشی یک پیمانه با هم ارتباط داشته باشند و باید به ترتیب مشخصی انجام شوند , انسجام رویه ای وجود دارد.

- انسجام ارتباطی (Communicational Cohesive)

همه عناصر پردازشی به یک ناحیه از ساختمان داده ها متمرکز شوند.

مثالی از سطوح پایین انسجام

به عنوان مثالی از انسجام پایین , پیمانه ای را در نظر می‌گیریم که پردازش خطا را برای یک برنامه پکیج نرم‌افزاری تحلیل مهندسی انجام می‌دهد.این پیمانه زمانی فراخوانی می‌شود که داده های محاسبه شده، از مرزهای از پیش تعیین شده تجاوز نمایند. وظایف ان عبارتند از:

۱. محاسبه داده های مکمل بر اساس داده های محاسبه شده اولیه
۲. تولید گزارش خطا (با محتویات گرافیکی) روی ایستگاه کاری کاربر
۳. اجرای محاسبات بعدی که مورد درخواست کاربر است
۴. بهنگام سازی یک بانک اطلاعاتی
۵. فراهم آوردن امکان انتخاب از منو برای پردازش‌های بعدی

گرچه وظایف قبلی ارتباط محکمی با هم ندارند، هر کدام یک نهاد عملیاتی مستقل هستند که ممکن است به بهترین نحو به صورت پیمانه‌ای جداگانه اجرا شوند. ترکیب عملکردها در یک پیمانه واحد، تنها می‌تواند به افزایش احتمال انتشار خطا به هنگام اصلاح یکی از وظایف پردازشی فوق کمک کند.

اتصال(Coupling)

اتصال، مقیاس ارتباط بین پیمانه‌ها در ساختار نرم‌افزار است. اتصال به پیچیدگی واسط بین پیمانه‌ها، نقطه ورود (دخول) و ارجاع به یک پیمانه و نوع داده‌های عبوری از واسط، بستگی دارد.

در طراحی نرم‌افزار، تلاش ما در جهت **پایین‌ترین سطح ممکن اتصال** است. اتصال سادهٔ بین پیمانه‌ها باعث ایجاد نرم‌افزاری می‌شود که فهم آن ساده‌تر بوده و هنگام وقوع خطاها در یک محل و پخش آنها در سیستم، کمتر در معرض «اثر فزونگری» ایجاد شده قرار دارد.

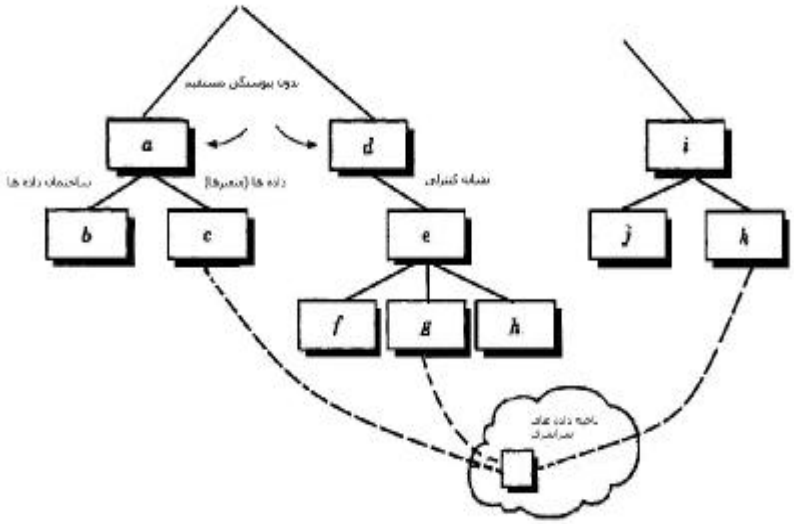
در شکل زیر پیمانه c تابع پیمانه a بوده و از طریق لیست قراردادی ارگومان که داده‌ها از طریق آن منتقل می‌شوند، قابل دسترسی است. تا مادامی که لیست آرگومان ساده وجود دارد(یعنی داده‌های ساده انتقال یافته و ارتباطی یک به یک بین اقلام برقرار است)، اتصال ضعیف یا(**اتصال داده‌ها**) در این بخش از ساختار به نمایش گذاشته می‌شود. نوعی اتصال داده‌ها به نام " **اتصال برچسبی**" زمانی ظاهر می‌شود که بخشی از ساختار داده‌ای(به جای آرگومان‌های ساده) از طریق واسط پیمانه، منتقل می‌گردد. این اتصال بین پیمانه‌های a وb بوجود می‌آید.

در سطوح متوسط " اتصال با انتقال کنترل بین پیمانه‌ها " توصیف می‌گردد. " **اتصال کنترل**" در اکثر طراحی‌های نرم‌افزاری بسیار رایج است و در شکل زیر با عبور " نشانه نمای کنترل " (متغیری که تصمیمات را در یک پیمانه تابع یا حاکم کنترل می‌کند) بین پیمانه‌های d و e نشان داده شده است.

در صورت **ارتباط پیمانه‌ها با محیط خارج از نرم‌افزار، سطوح نسبتاً بالای اتصال**، به وجود می‌آیند. بعنوان مثال I/O، پیمانه‌ها را به دستگاه‌های خاص، قالب‌ها و پروتوکل‌های ارتباطاتی، متصل می‌کند.

" اتصال خارجی " ضروری است اما باید به تعداد کمی از پیمانه‌ها محدود شود. اتصال سطح بالا نیز زمانی پدید می‌آید که تعدادی از پیمانه‌ها به ناحیهٔ سراسری داده‌ها ارجاع می‌کنند " **اتصال مشترک**" که نام این حالت می‌باشد.

بالاترین میزان اتصال با عنوان " **اتصال محتوا** " زمانی به وجود می‌آید که یک پیمانه، از داده‌ها یا اطلاعات کنترلی موجود در حد و مرز پیمانه‌ای دیگر، استفاده می‌کند. ثانیاً اتصال محتوایی به هنگام ایجاد شاخه‌هایی در میان یک پیمانه نیز اتفاق می‌افتد. این حالت اتصال، قابل پیشگیری بوده و باید از آن اجتناب کرد.



شکل ۵: " نمودار نخست " ساختار برنامه برای حس‌گرهای نمایش دهنده

انواع اتصال (Coupling)

- **اتصال داده‌ای** (Structural Coupling)

مادمی که لیستی از آرگومانهای ساده وجود داسته باشند (یعنی داده های ساده عبور داده شوند، و بین عناصر موجود، تناظر یک به یک برقرار باشد)، در این بخش از برنامه اتصال اندکی موسوم به اتصال داده‌ای به چشم می‌خورد.(درشکل بین a و c)

- **اتصال برچسبی** (Stamp Coupling)

هنگامی یافت می‌شود که بخشی از ساختمان داده‌ها (به جای آرگومانهای ساده) از طریق یک واسط پیمانه عبور داده شوند. (درشکل بین a و b)

- **اتصال کنترلی** (Control Coupling)

در سطوح میانی، اتصال با تبادل کنترل بین پیمانه‌ها مشخص می‌شود. اتصال کنترلی در اکثر طراحی‌های نرم افزاری بسیار متداول است و در شکل بین c-d موجود است.

- **اتصال خارجی** (External Coupling)

سطوح نسبتا بالای اتصال هنگامی رخ می‌دهد که پیمانه‌ها با محیط خروجی نرم‌افزار ارتباطی تنگاتنگ داشته باشند. اتصال خارجی ضروری است، ولی باید به تعداد اندکی از پیمانه‌ها با یک ساختار محدود شود.

- **اتصال مشترک** (Common Coupling)

زمانی که پیمانه یک دسته از داده‌های سرتا سری را آدرس‌دهی می‌کند , اتصال بالایی رخ می‌دهد .

- **اتصال محتویات** (Content Coupling)

بالاترین درجه اتصال، اتصال محتویات است و هنگامی رخ می‌دهد که یک پیمانه از داده‌های اطلاعات کنترلی نگهداری شده در مرز پیمانه دیگر استفاده کند. از این نوع اتصال می‌توان و باید حذر کرد.

ابتکار در طراحی پیمانه‌های موثر و کارآ

پس از ایجاد ساختار برنامه، پیمانه‌ای کردن مؤثر و کارآمد به واسطهٔ اجرای مفاهیم طراحی که در ابتدای فصل معرفی شدند، تحقق می‌یابد. ساختار برنامه براساس مجموعه ذهنیت‌های زیر، قابل دست‌کاری است:

۱. **" اولین تکرار" ساختار برنامه را به منظور کاهش اتصال و بهبود انسجام، مورد ارزیابی قرار دهید.** پس از ساخت برنامه، می‌توان پیمانه‌ها را از جهت بهبود استقلال آنها، تفکیک یا ترکیب کرد.
یک " **پیمانه ترکیبی**" در ساختار نهایی برنامه، تبدیل به دو یا چند پیمانه می‌گردد. **پیمانه تفکیکی** اغلب زمانی حاصل می‌شود که پردازش مشترک در دو یا چند پیمانه وجود داشته و به صورت یک پیمانه منسجم و جداگانه، مجدداً قابل تعریف است. هنگامی که اتصال در سطح بالا مورد نظر می‌باشد، می‌توان گاهی پیمانه‌های را برای کاهش انتقال کنترل، ارجاع به داده‌های سراسری و پیچیدگی واسط، ترکیب کرد.

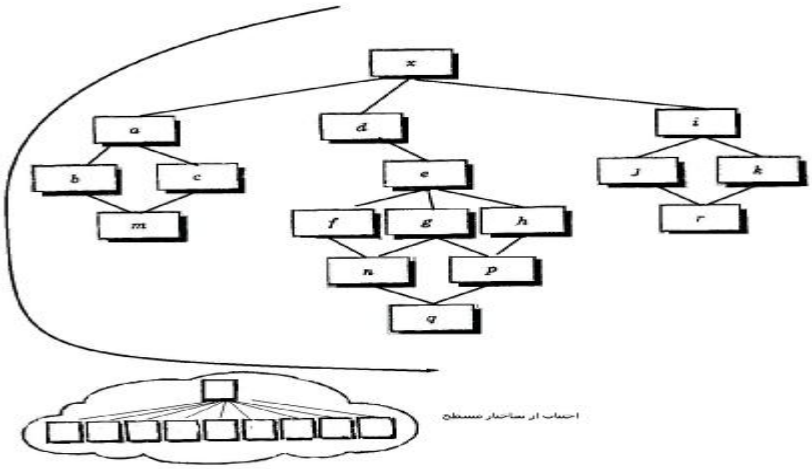
۲. **سعی کنید ساختارهایی با گنجایش خروجی زیاد را به حداقل رسانده و همان‌طور که عمق افزایش می‌یابد، برای گنجایش ورودی تلاش کنید.** ساختار داخل ابر در شکل زیر از تجزیهٔ عوامل استفادهٔ مفیدی نمی‌کند. تمامی پیمانه‌ها، زیر یک تکه پیمانه کنترل به طور یکنواخت قرار دارند. به طور کلی، توزیع معقول‌تر کنترل در ساختار سمت راست، نشان داده شده است. این ساختار بیضی شکل بوده و تعدادی لایهٔ کنترل و پیمانه‌های کاملاً سودمند را در سطوح پایین‌تر نمایش می‌دهد.

۳. **حوزه تأثیر یک پیمانه را در محدودهٔ دامنهٔ کنترل همان پیمانه حفظ کنید.** حوزهٔ تأثیر یک پیمانه e، به تمام پیمانه‌هایی اطلاق می‌شود که تحت تأثیر تصمیم اتخاذ شده در پیمانه e هستند. دامنهٔ کنترل پیمانه e، همه پیمانه‌هایی هستند که به طور مستقیم یا با واسطه تابع پیمانه e می‌باشند. اگر تصمیم اتخاذ شده در پیمانه e بر پیمانه r تأثیر بگذارد، ذهنیت ۳ نقض شده، زیرا پیمانه r در حوزهٔ کنترل پیمانه e نمی‌باشد.

۴. **واسط‌های پیمانه را به منظور کاهش پیچیدگی و بهبود سازگاری مورد ارزیابی قرار دهید.** پیچیدگی واسط، پیمانه، عامل عمدهٔ خطاهای نرم‌افزاری است. واسطها باید برای انتقال راحت اطلاعات طراحی شده و با وظیفهٔ پیمانه هماهنگ و سازگار باشند. ناسازگاری واسط (یعنی داده‌های ظاهراً نامربوط که از طریق لیست آرگومان یا تکنیک دیگری منتقل می‌شوند) نشانهٔ انسجام و یکپارچگی است. پیمانه مورد نظر بایستی مجدد مورد ارزیابی واقع شود.

۵. **پیمانه‌هایی را تعریف کنید که کارشان قابل پیش‌بینی است اما از پیمانه‌های بسیار محدودکننده اجتناب نمایید.** یک پیمانه زمانی قابل پیش‌بینی است که به صورت یک جعبهٔ سیاه تلقی شود، یعنی، داده‌های یکسان خارجی بدون در نظر داشتن جزییات داخلی پردازشی، تولید خواهند شد. پیمانه‌ای که پردازش را فقط بر یک عمل فرعی محدود می‌کند، انسجام زیادی را نشان می‌دهد و طراح نسبت به آن تمایل دارد. با این وجود آن پیمانه‌ای که به طور دلخواه اندازهٔ ساختاری داده‌ای محلی را و گزینه‌های جریان کنترل یا حالات واسط خارجی را محدود می‌کند، همواره مستلزم مراقبت و نگهداری خواهد بود تا چنین محدودیت‌هایی رفع گردند.

۶. **برای پیمانه‌هایی با " ورودی کنترل شده " تلاش نموده و از " اتصالات نامعقول" خودداری کنید.** این ذهنیت طراحی، دربارهٔ اتصال محتوایی، هشدار می‌دهد. در صورت محدودسازی و کنترل واسطهای پیمانه، درک و در نتیجه نگهداری و ترمیم نرم‌افزار ساده‌تر می‌گردد.



شکل ۵: " نمودار نخست " ساختار برنامه برای حس‌گرهای نمایش دهنده

بنابراین در طراحی پیمانه

- تقطیع وتلفیق (Exploded/ Imploded)

تقطیع پیمانه باعث تبدیل پیمانه به دو یا چند پیمانه در برنامه نهایی می‌شود و تلفیق پیمانه به ترکیب پردازش حاصل از دو یا چند پیمانه می انجامد.

- کوشش برای به حداقل رساندن ساختارهایی با توان خروجی بالا و تلاش برای افزایش دادن توان ورودی به موازات افزایش عمق شکل
- حفظ حوزه یک پیمانه در دامنه کنترل آن پیمانه: حوزه پیمانه عبارتست از کلیه پیمانه های دیگری که از تصمیم‌گیری انجام شده توسط یک پیمانه تاثیر بپذیرند. در شکل اگر r از e تاثیر بپذیرد، از این قاعده سرپیچی شده است.
- ارزیابی واسطهای پیمانه‌ها برای کاهش پیچیدگی و زواید و بهبود بخشیدن به سازگاری
- تعریف پیمانه‌هایی که عملکرد آنها قابل پیش‌بینی است و پرهیز از پیمانه هایی که همپوشانی دارند. پیمانه‌هایی که حافظه داخلی دارند می‌توانند غیرقابل پیش‌بینی باشند.
- کوشش برای دستیابی به پیمانه هایی با مدخل کنترل شده و پرهیز از اتصالات آسیب شناختی (Pathological Connection). اتصالات آسیب شناختی عبارت از ایجاد انشعاب یا رجوع به میانه یک پیمانه است.

مستند سازی طراحی

تعیین مشخصات طراحی با جنبه‌های متفاوتی از مدل طراحی سرو کار دارد و به موازات این که طراح، نمایش خود از نرم افزار را پالایش می‌کند، کامل می‌شود. ابتدا دامنه کلی کار شرح داده می‌شود. اکثر اطلاعاتی که در این مرحله ارایه می‌شود از مشخصات سیستم و مدل تحلیل(مشخصات خواسته های نرم افزار) بدست می‌آید.

سپس طراحی داده ها مشخص می‌شود. ساختار بانک اطلاعاتی، هر ساختار فایل خارجی، ساختمان داده‌های داخلی و یک ارجاع متقابل(Cross Reference) که اشیای داده‌ای را به فایل‌های مشخصی متصل می‌کند، همگی تعریف می‌شوند.

طراحی معماری نشان می‌دهد که معماری برنامه چگونه از مدل تحلیل بدست آمده است. بعلاوه از نمودارهای ساختاری برای نشان دادن سلسله مراتب پیمانه استفاده می‌شود.

طراحی واسط‌های داخلی و خارجی برنامه نمایش داده می‌شود و جزییات طراحی واسط انسان ـ ماشین شرح داده می‌شود.

مولفه‌ها، عناصر جداگانه از سیستم مثل زیربرنامه‌ها، توابع یا رویه‌ها، ابتدا با زبان مادری شرح داده می‌شوند. در این مرحله عملکرد رویه‌ای یک مولفه (پیمانه) توضیح داده می‌شود. سپس از یک ابزار طراحی رویه‌ای برای ترجمه این نسخه با توصیفی ساختیافته استفاده می‌شود.

مشخصات طراحی شامل یک ارجاع متقابل (Cross Reference) خواسته ها است . هدف از این ارجاع متقابل که توسط یک ماتریس متقابل نشان داده می‌شود:

- اثبات بر آورده شدن همه خواسته‌ها توسط طراح نرم‌افزار
- نشان دادن اینکه کدام مولفه‌ها برای پیاده سازی خواسته‌های مشخص اهمیت حیاتی دارد.

آخرین بخش از مشخصات طراحی شامل داده های مکمل است. توصیفات الگوریتم‌ها، رویه‌های دیگر، داده‌های جدولی، نیازهایی از مستندات دیگر به عنوان یادداشت خاص با پیوستی جداگانه ارایه می‌شوند. بهتر است که یک جزوه راهنمای نصب و راه اندازی مقدماتی تهیه و ضمیمه مستند طراحی شود.

چارچوب گزارش طراحی سیستم در صفحه بعد ارایه شده است.

تست‌های فصل سیزده: اصول و قواعد طراحی

۱- کدام یک از موارد زیر در مدل طراحی مورد توجه قرار نمی‌گیرند؟

الف- معماری ب) داده ج) واسطها د) محیط پروژه

۲- اهمیت طراحی نرم‌افزار را می‌توان درکدام کلمه خلاصه نمود؟

الف- صحت ب- پیچیدگی ج- کارایی د- کیفیت

۳- کدام یک از موارد زیر از ویژگی‌های مشترک تمام روش‌های طراحی نمی‌باشد؟

الف- مدیریت پیکربندی ب- نشانه مولفه عملیاتی ج- راهنماهای ارزیابی کیفیت د- پالایش ذهنی

۴- قبل از شروع به کار، باید مجموعه‌ای از قواعد طراحی وضع شوند تا انسجام و جامعیت طراحی را تضمین نماید.

الف- درست ب) نادرست

۵- کدام یک از انواع انتزاع در طراحی نرم‌افزار به کار می‌رود؟

الف- کنترلی ب) داده‌ای ج- رویه‌ای د- همه موارد

۶- هنگام استفاده از متدولوژی‌های طراحی ساختیافته، فرآیند پالایش مرحله ای غیر الزامی است.

الف- درست ب) نادرست

۷- از آنجایی که پیمانه‌ای بودن از اهداف مهم طراحی است، وجود پیمانه‌های متعدد در یک طراحی پیشنهادی ممکن نیست.

الف- درست ب) نادرست

۸- کدام یک از انواع مدل‌های زیر نمایانگر یک معماری نرم‌افزار نمی‌باشد؟

الف- داده ب- پویا ج- فرآیند (رویه) د- ساختاری

۹- سلسله مراتب کنترل نمایانگر کدام مورد زیر است؟

الف- ترتیب تصمیم‌گیری ب- سازماندهی پیمانه‌ها ج- تکرار عملیات د- ترتیب رویه‌ها

۱۰-افراز افقی برای عملیات برنامه، شاخه‌های جداگانه‌ای تعریف می‌کند. در حالی که افراز عمودی کنترل را به صورت بالا و پایین توزیع می‌کند.

الف- درست ب) نادرست

۱۱-طراحی ساختمان داده‌ها نسبت به طراحی الگوریتم زمان کمتری می‌گیرد. در نتیجه ممکن است به آخر کار موکول شود.

الف- درست ب) نادرست

۱۲-رویه نرم‌افزار بر کدام مورد زیر تاکید دارد؟

الف- سلسله مراتب کنترل در حالت انتزاعی‌تر ب- پردازش جزییات هر پیمانه به طور مجزا

ج- پردازش جزییات هر مجموعه از پیمانه‌ها به طور جمعی د- رابطه بین کنترل و رویه

۱۳-انسجام(Cohesion) نمایشگر کیفی درجه‌ای است که پیمانه در آن:

الف- میتواند به طور فشرده‌تر نوشته شود. ب- تنها بر یک مورد تاکید دارد.

ج- قادر به اتمام به موقع عملیات خود می‌باشد. د- به پیمانه‌های دیگر و جهان خارج متصل می‌شود.

۱۴-اتصال(Coupling) نمایشگر کیفی درجه‌ای است که یک پیمانه در آن:

الف- میتواند به طور فشرده‌تر نوشته شود. ب- تنها بر یک مورد تاکید دارد.

ج- قادر به اتمام به موقع عملیات خود می‌باشد. د- به پیمانه‌های دیگر و جهان خارج متصل می‌شود.

۱۵-ذهنیات طراحی معمولاً تنها مورد استفاده دانشجویان است و مهندسین نرم افزار مجرب از آن بی‌نیازند.

الف- درست ب) نادرست

۱۶- دلیل نادرستی طراحی سطح مولفه قبل از طراحی داده این است که:

الف- طراحی مولفه وابسته به زبان برنامه‌سازی است ولی طراحی داده این طور نیست.

ب- طراحی داده آسان‌تر است.

ج- طراحی داده سخت‌تر است.

د- ساختمان داده معمولاً بر روش طراحی سطح مولفه تاثیر می‌گذارد.

۱۷- پیمانه‌های c، g و k هرکدام به داده‌ای (مانند فایل یا ناحیه‌ای از حافظه که دستیابی سراسری بـه آن وجـود دارد) در ناحیه سراسری داده‌ها دستیابی دارند. پیمانه c، داده‌ای را مقدار دهی مـی‌نمایـد. سـپس پیمانـه g مقدار آن را محاسبه و بهنگام می‌کند. در این حالت کدام یک از سطوح اتصال (Coupling) وجود دارد؟

الف- Data Couplig
ب- External Coupling
ج- Common Coupling
د- Content Coupling