فصل ۱۴ طراحی معماری

مفاهیم کلیدی (مرتب بر حروف الفبا)

الگوها ، پالایش ساختاری ، سبک ها ، سبک های ارزیابی ، طراحی داده ها ، عامل سازی ها (factoring) ، مخزن داده ها ، معماری ، نگاشت تبدیلات ، نگاشت تراکنش‌ها

KEY CONCEPTS

architectural refinement , architecture , data design , data warehouse , evaluating styles , factoring , patterns , styles , transaction , mapping , Transform mapping

نگاه اجمالی

طراحی معماری چیست؟ طراحی معماری نمایان‌گر ساختار اجزای داده‌ای و برنامه‌ای می‌باشد که برای ساختن یک سیستم مبتنی بر کامپیوتر لازمند. این طرح سبک معماری را که سیستم نیاز دارد، ساختار و خواص اجزای تشکیل‌دهنده سیستم و روابط متقابل میان همه اجزای معماری سیستم را در نظر می‌گیرد.

این طراحی را چه کسی انجام می‌دهد؟ با این‌که مهندس نرم‌افزار می‌تواند داده‌ها و معماری سیستم را طراحی کند، اما اغلب این کار برعهده متخصصانی می‌باشد که در هنگام ساخت سیستم‌های بزرگ و پیچیده فرا خوانده می‌شوند. مهندس پایگاه داده‌ای یا طراح مخزن اطلاعات، معماری داده‌ای را برای سیستم ایجاد می‌کند. آرشیتکت سیستم (معمار سیستم) سبک معماری مناسب را برای نیازمندیهای سیستم در طول مهندسی سیستم و تحلیل نیازمندیهای نرم‌افزاری، انجام می‌دهد.

چرا طراحی معماری مهم است؟ در قسمت "بررسی اجمالی" در فصل قبل پرسیدیم: "شما بدون داشتن نقشه شروع به ساختن خانه نمی‌کنید، این‌طور نیست؟" هم‌چنین بدون کشیدن طرح‌بندی لوله‌کشی ساختمان، کار کشیدن طرح و نقشه خانه را آغاز نمی‌کنید. قبل از پرداختن به جزییات، هیچ نظری نسبت به خود خانه ندارید. این همان کاری است که در طراحی معماری صورت می‌گیرد. یعنی تصویر بزرگی از کار در اختیارتان گذاشته و شما را مطمئن می‌سازد که آن را به‌درستی انجام می‌دهید.

مراحل کار چیست؟ طراحی معماری با طراحی داده‌ها آغاز شده و سپس به یک یا چند شاخه از ساختمان معماری سیستم مشتق می‌شود. سبک‌ها یا الگوهای معماری جایگزین در تلاش برای بدست آوردن بهترین ساختمان و کیفیت موجود، تحلیل می‌گردند. وقتی طرح موردنظر انتخاب شد، طرح معماری با استفاده از روش طراحی معماری بسط می‌یابد.

نتیجهٔ این کار چیست؟ در طول کار طراحی، یک مدل که در برگیرنده ساختمان برنامه و معماری داده‌هاست، ایجاد می‌شود. علاوه بر آن، مشخصه‌های اجزا و ارتباطشان توصیف می‌شود.

چگونه از درستی انجام کار اطمینان حاصل کنیم؟ در هر مرحله، نتایج کار طراحی نرم‌افزار از نظر روشن بودن، درستی و سازگاری با نیازمندیها و با یکدیگر بررسی می‌شوند.

طراحی به‌عنوان یک فرآیند چند مرحله‌ای توصیف شده که در آن ارائه اطلاعات و ساختار برنامه، مشخصه‌های رابط و جزییات روش کار از روی نیازمندیهای اطلاعاتی تلفیق می‌شوند. این توصیف توسط فریمن بسط داده شده است:[FRE80][1]

طراحی فعالیتی مربوط به اتخاذ تصمیمات بزرگ است که اغلب دارای ماهیت ساختاری هستند. این کار از نظر بازنمایی انتزاعی اطلاعات و سلسله پردازش‌ها، با برنامه‌سازی دارای فصول مشترکی است، امّا در جزئیات اختلافاتی وجود دارند، طراحی انسجام می‌آفریند، ارائه درست و طراحی شدهٔ برنامه‌هایی که روی روابط درونی بخش‌ها در سطح بالاتری کار می‌کنند و عملیات منطقی در سطوح پایین ...

همان‌گونه که در فصل قبلی متوجه شدیم، طراحی مبتنی بر اطلاعات است. روش‌های طراحی نرم‌افزار هر یک از سه حوزهٔ مدل تحلیل می‌باشند. این سه حوزه یعنی داده‌ای، کارکردی و رفتاری رهنمودی برای خلق طراحی نرم‌افزاری هستند.

روش‌های لازم برای ایجاد انسجام و هماهنگی یعنی ارائه برنامه‌ریزی شدهٔ داده‌ها و لایه‌های معماری مدل طراحی در این فصل ارائه شده‌اند. هدف ما عبارتست از تأمین یک نگرش نظام‌مند برای اشتقاق طراحی معماری یعنی طرح اولیه‌ای که نرم‌افزار از روی آن ساخته می‌شود.

۱۴-۱ معماری نرم افزار

در کتاب معروف شاو وگارلن[SHA96][2] درخصوص این موضوع، آنها به شکل زیر در مورد این موضوع یعنی معماری نرم‌افزار به بحث می‌پردازد:

از زمانی‌که اولین برنامه به پیمانه‌هایی تقسیم شد، سیستم‌های نرم‌افزاری دارای طرح معماری بوده‌اند و برنامه‌سازان مسئول ارتباط بین پیمانه‌ها و خواص جهانی مونتاژ آنها شده‌اند. از نظر تاریخی، این ساختارها اتفاقات تصادفی پیاده‌سازی یا سیستم‌های بازمانده گذشته‌اند. تولیدکنندگان نرم‌افزارهای خوب اغلب یک یا چند الگوی معماری را به‌عنوان راهبردهایی برای سازمان‌دهی سیستم به‌کار گرفته‌اند. اما از این الگوها رسماً استفاده کرده و هیچ وسیله‌ای در اختیار ندارند تا آنها را در سیستم بدست آمده واضح و مشخص کنند.

1.Freeman,P.

2.Shaw,M. and D.

امروزه، ساختار مؤثر نرم‌افزار و طراحی و ارائه صریح آن موضوعات گسترده‌ای در مهندسی نرم‌افزار هستند.

۱۴-۱-۱ معماری چیست ؟

وقتی در مورد معماری یک ساختمان یا بنا صحبت می‌کنیم، نگرش‌های متفاوتی به ذهنمان می‌رسد. در ساده‌ترین سطح، شکل کلی ساختمان فیزیکی را در نظر می‌گیریم. اما در واقع معماری آن، بسیار پیچیده‌تر از اینهاست. این حالتی است که در آن اجزای مختلف ساختمانی به‌صورتی با هم ترکیب می‌شوند که یک کل سازمانمند و منسجم را تشکیل می‌دهند. این شیوه‌ای است که با آن ساختمان، همراه با ساختمان‌های مجاور خود در محیط سازگاری می‌یابد و هماهنگ می‌شود. در این‌جا این ساختار سیستم تا حدی هدف مشخص شده خود را نشان داده و نیازهای صاحبش را برطرف می‌کند. در این‌جا حس زیباشناسی ساختمان یعنی تأثیر بصری ساختمان، و شیوه ترکیب بافت‌ها، رنگ‌ها و مواد برای خلق نما و محیط زنده درونی مطرح است. در این‌جا جزییات کوچک مثل طراحی تسهیلات نور، نوع کف‌پوش و جایگزینی آویزه‌های دیوار فهرستی بلند بالا را به‌وجود می‌آورند. و نهایتاً، موضوع اصلی هنر است.

در مورد ساختار نرم‌افزاری چه می‌توان گفت؟ باس و همکارانش[BAS98][1] این عبارت دشوار و دیرفهم را به شکل زیر تعریف کرده‌اند:

معماری نرم‌افزار در یک برنامه یا سیستم محاسباتی عبارتست از ساختار یا ساختارهای سیستم که شامل اجزای نرم‌افزاری، مشخصه‌های مشهود برونی این اجزاء و ارتباطات میان آنها می‌باشد.

معماری، یک نرم‌افزار عملیاتی نیست. بلکه نمودی است که مهندس نرم‌افزار را قادر می‌سازد:

۱) میزان تأثیر طرح را در مرتفع نمودن نیازمندیهای بیان شده، تحلیل کند.

۲) معماری‌های جایگزین دیگر را در مرحله‌ای که تغییر طرح هنوز نسبتاً آسان است، بررسی کند.

۳) خطرات مربوط به ساخت نرم‌افزار را کاهش دهد.

تعریف اشاره شدهٔ فوق بر نقش "اجزای نرم‌افزاری" در هرگونه نمایش معماری و ساختاری، تأکید دارد. در متن طراحی معماری، هر جزء نرم‌افزاری می‌تواند چیزی به سادگی یک پیمانه برنامه بوده، اما می‌توان آن را بسط داد تا پایگاه‌های داده‌ای را نیز در بربگیرد و هم‌چنین می‌تواند شامل میان‌افزار یا Middleware شود که پیکربندی شبکه خادم‌ها و مخدوم‌ها را ممکن می‌سازد. خواص این اجزاء مشخصه‌هایی هستند که برای درک چگونگی تأثیر متقابل اجزاء با دیگر قسمت‌ها لازم و ضروری هستند. در سطح معماری، خواص درونی مثل جزییات یک الگوریتم، مشخص نیست. ارتباط بین اجزاء می‌تواند به

ارجاع به وب

لیست مفیدی از منابع معماری نرم افزار در آدرس زیر وجود دارد:
www.massd.edu/secenter/saresources.html

نقل قول

معماری یک سیستم، یک چهارچوب جامعی است که فرمها، ساختار و اجزاء آن را تشریح می کند و نشان می دهد که آنها چگونه به یکدیگر مربوط و متصل می باشند.
جرولد گروچف

---

1 Bass,L.

سادگی فراخوانی زیر روال یک پیمانه توسط پیمانه دیگر، یا به پیچیدگی پروتکل دسترسی به پایگاه داده‌ای باشد.

در این کتاب، طراحی معماری نرم‌افزار دو سطح هرم طراحی را در نظر می‌گیرد، یعنی طراحی داده‌ها و طراحی معماری. در لابه‌لای مبحث ارائه شده فوق، طراحی داده‌ها ما را قادر می‌سازد تا جزء داده‌ای را در معماری ارائه دهیم. طراحی معماری روی ارائه ساختار اجزاء نرم‌افزاری، خواصشان و روابط متقابلشان معطوف می‌شود.

۱۴-۱-۲ چرا معماری از اهمیت برخوردار است؟

مدل معماری، یک دید ترکیبی از یک سیستم را مهیا می سازد، و اجازه می دهد که مهندس نرم افزار آن را به صورت یک کل مورد آزمایش قرار دهد.

باس و همکارانش در کتابی که به معماری نرم‌افزاری تخصیص یافته، سه دلیل مهم را برای اهمیت این مطلب ارائه می‌کنند:

- ارائه معماری نرم‌افزاری کار برقراری ارتباط را میان همه طرفهای علاقه‌مند به توسعه سیستم مبتنی بر کامپیوتر، میسر می‌سازد.
- این معماری تصمیمات طراحی اولیه را مشخص می‌سازد که دارای تأثیر زیادی بر همه کارهای مهندسی نرم‌افزار است و روی موفقیت نهایی سیستم به‌عنوان یک ماهیت عملیاتی نیز بسیار مؤثر است.
- این معماری یک مدل نسبتاً کوچک و قابل درک یا چگونگی ساخت سیستم و چگونگی کار کردن اجزای آن با یکدیگر را ارائه می‌دهد.[BAS98]

مدل طراحی معماری و الگوهای معماری موجود در آن قابل انتقالند. یعنی سبکها و الگوهای معماری را می‌توان در طراحی دیگر سیستمها به‌کار گرفت و مجموعه‌ای از تصاویر انتزاعی را به‌وجود آورد که مهندسین نرم‌افزار را قادر می‌سازند سبک معماری را به شیوه‌های قابل پیش‌بینی توصیف کنند.

۱۴-۲ طراحی داده ها

طراحی داده‌ها[1] هم‌چون دیگر فعالیتهای مهندسی نرم افزار (کا گاهی به آن معماری داده‌ها[2] نیز می‌گویند) مدلی از داده‌ها و/ یا اطلاعات را در سطح بالایی از حالت انتزاعی، ایجاد می‌کند. سپس مدل داده‌ای به‌صورت بازنمایی خاص پیاده‌سازی درمی‌آید که می‌توان آن را به‌وسیله سیستم مبتنی بر کامپیوتر پردازش کرد. در بسیاری از برنامه‌های کاربردی نرم‌افزاری، معماری داده‌ها تأثیر شگرفی بر معماری نرم‌افزاری دارد که باید آن را پردازش کند.

1.data design

2.data architecting

ساختار داده همیشه بخش مهمی از طراحی نرم‌افزار بوده است. در سطح مؤلفهٔ برنامه، طراحی ساختارهای داده‌ها و الگوریتم‌های مربوطه که نیازمند تغییراتی می‌باشند، برای ارائه برنامه‌های کاربردی با کیفیت، ضروری هستند. در سطح برنامه کاربردی، تغییر مدل داده‌ای به یک پایگاه داده‌ای برای دستیابی به اهداف تجاری سیستم، در مرکز توجه قرار دارد. از نظر تجاری، مجموع اطلاعات ذخیره شده در پایگاه‌های داده‌ای مجزا و اطلاعاتی که در مخازن اطلاعاتی سازمان‌دهی شده‌اند، استخراج داده‌ها یا کشف آن را امکان‌پذیر می‌سازند که این امر می‌تواند بر موفقیت خود تجارت اثر داشته باشد. در هر مورد، طراحی داده‌ها نقش مهمی را ایفا می‌کند.

۱۴-۲-۱مدل سازی داده، ساختار داده، پایگاه داده، و انبار داده ها

اشیای داده‌ای که در طول تحلیل نیازمندیهای نرم‌افزاری تعریف شده‌اند، با استفاده از دیاگرام‌های موجودیت/ رابطه و فرهنگ داده‌ها، مدل‌سازی می‌شوند. کار طراحی داده‌ها این عناصر مورد نیاز مدل نیازمندیها را در سطح جزء نرم‌افزاری به ساختمان داده‌ها و وقتی لازم باشد معماری پایگاه داده‌ای را در سطح برنامه کاربردی، تبدیل می‌کند.

در سال‌های گذشته، معماری داده‌ها معمولاً به ساختمان داده‌ها در سطح برنامه و پایگاه داده‌ای در سطح برنامه کاربردی محدود بود. اما امروزه، تجارت‌های بزرگ و کوچک لبریز از اطلاعات هستند. حتی برای یک کار معمولی، داشتن دهها پایگاه داده‌ای که در خدمت بسیاری از برنامه‌های کامپیوتری هستند و در برگیرنده صدها گیگا بایت اطلاعات هستند، نیز چندان غیرمعمول نیست. چالش اصلی در مورد یک تجارت بدست آوردن اطلاعات مفید از این محیط داده‌ای است، به‌خصوص وقتی اطلاعات مورد نیاز چند کاربردی است (مثل اطلاعاتی که تنها در صورت تقابل اطلاعات بازاریابی خاصی، همراه با داده‌های مهندسی محصول، به می‌توان بدست آورد).

برای حل این مسئله، انجمن تجارت IT فنون استخراج داده‌ها[1] یا کشف اطلاعات در پایگاه داده‌ای[2] (KDD) را ارائه داده است که به‌منظور بدست آوردن اطلاعات در سطح تجاری مناسب، پایگاه‌های داده‌ای را کاملاً جستجو می‌کند. وجود پایگاه‌های داده‌ای چندگانه، ساختارهای متفاوت آنها، میزان جزییات موجود در پایگاه‌ها و بسیاری از عوامل دیگر، استخراج داده‌ها را در محیط پایگاه داده‌ای موجود، مشکل می‌سازند. راه‌حل دیگر به نام انبار اطلاعاتی[3]، یک لایه اضافه به معماری داده‌ها می‌افزاید.

نقل قول

کیفیت داده،تفاوت میان ذخیره سازی داده ها مفید و انبار کردن داده های به درد نخور است. جارت روزنبرگ

ارجاع به وب

طلاعات به هنگام در خصوص فنون نگهداری داده ها را می توانید در آدرس زیر بیابید :
www.datawarehouse.com

1.data mining techniques
2.knowledge discovery in databases
3.data warehouse

یک انبار اطلاعاتی، یک محیط داده‌ای جداگانه است که مستقیماً با برنامه‌های کاربردی روزبه‌روز تلفیق نمی‌شود، [MAT96] [1] بلکه تمام اطلاعات مورد استفاده در تجارت را در برمی‌گیرد. از یک نظر، انبار اطلاعات یک پایگاه داده‌ای بزرگ و مستقل است که بعضی از داده‌ها، نه همه آنها، را که در پایگاه داده‌ای ذخیره شده و در خدمت مجموعه‌ای از برنامه‌های کاربردی لازم برای تجارت هستند، در برمی‌گیرد. اما مشخصه‌های بسیاری هستند که بین یک انبار اطلاعاتی و یک پایگاه داده‌ای معمولی تمایز ایجاد می‌کنند : [INM95] [2]

جهت‌گیری موضوع. انبار اطلاعاتی توسط موضوعات عمده تجاری سازمان‌دهی می‌شود نه فرآیند یا عملکرد تجاری، این کار منجر به اجرای داده‌ها می‌شود که ممکن است برای یک کار تجاری به‌خصوص لازم باشد. اما این کار معمولاً برای استخراج داده‌ها لازم نیست.

تلفیق. بدون توجه به منبع داده‌ها، این کار قراردادهای نام‌گذاری منسجم، واحدها و اندازه‌ها ساختارهای کدگذاری و نگرش‌های فیزیکی را نشان می‌دهد. حتی وقتی عدم انسجام در برنامه‌های مختلف در پایگاه‌های داده‌ای وجود دارد.

تفاوت زمانی. در مورد یک محیط برنامه کاربردی مبتنی بر تراکنش، داده‌ها در لحظه دستیابی و برای مدت کوتاهی (معمولاً ۶۰ تا ۹۰ روز) قبل از دسترسی، دقیق هستند. در مورد انبار اطلاعاتی، می‌توان در یک لحظهٔ به‌خصوص زمانی به داده‌ها دسترسی پیدا کرد (مثلاً مشتریان در تاریخی مطلع می‌شوند که محصول جدیدی به مطبوعات تجاری معرفی شده است). حد زمانی معمول برای انبار اطلاعات ۵ تا ۱۰ سال است.

ثبات. بر خلاف پایگاه‌های برنامه‌های کاربردی تجاری که تحت تغییرات مستمر قرار می‌گیرند (وارد کردن، حذف کردن، به هنگام‌سازی)، اطلاعات در انبار ذخیره می‌شوند، اما بعد از انتقال اصلی، تغییر دیگری نمی‌کنند.

این مشخصه‌ها نمایان‌گر مجموعه منحصربه‌فردی از چالش‌های طراحی برای معماری داده‌هاست. مانند ([PRE96] [3]و [DAT95] [4]و [KIM95] [5])

یک انبار داده ها مشتمل بر تمام داده هایی است که مورد استفاده یک اداره یا سازمان قرار می گیرد. هدف، دسترسی به کل معلوماتی است که نباید به گونه ای دیگر و در جای دیگر قرار داشته باشد.

1 Mattison ,R.
2.Inmon,W.H.
3.Preiss,B.R.
4.Date,C.J.
5.Kimball,R.

۱۴-۲-۲ طراحی تفصیلی داده ها (در سطح اجزاء)

برای طراحی داده ها کدام اصول کاربردی وجود دارد؟

طراحی داده‌ها در این سطح روی ارائه و نمایش ساختمان داده‌هایی متمرکز می‌شود که مستقیماً توسط یک یا چند جزء نرم‌افزاری قابل دسترسی هستند. ویسرمن[WAS80][1] مجموعه اصولی را پیشنهاد کرد که ممکن است برای مشخص کردن و طراحی چنین ساختارهایی به‌کار روند. در واقع، طراحی داده‌ها در طول خلق مدل تحلیل شروع می‌شود. فراخوانی تحلیل نیازها و طراحی اغلب هم‌پوشانی می‌شود. مجموعه اصول زیر را برای مشخصه‌های داده‌ای، در نظر بگیرید:

۱- **اصول تحلیل نظام مند که در مورد عملکرد و رفتار به‌کار می‌رود باید در مورد داده‌ها نیز به‌کار رود.** ما وقت و تلاش زیادی را صرف اشتقاق، مرور و مشخص کردن نیازمندیهای کارکردی و طراحی اولیه می‌کنیم. نمایش جریان و محتوای داده‌ها نیز باید توسعه یافته و بازبینی شود. اشیای داده‌ای باید شناسایی شده، سازمان‌های داده‌ای جایگزین در نظر گرفته شده و تأثیر مدل‌سازی داده‌ها روی طراحی نرم‌افزار باید ارزیابی گردد. به‌طور مثال، مشخص کردن یک فهرست چند زنجیره‌ای ممکن است نیازمندیهای اطلاعاتی را مرتفع سازد، اما ممکن است منجر به یک طراحی نرم‌افزاری فاقد کارآیی شود. یک سازمان‌دهی داده‌ای ممکن است نتایج بهتری داشته باشد.

۲- **تمام عملیات و ساختمان داده‌ها باید شناسایی شوند، در طراحی یک ساختار داده‌ای کارآمد باید عملیاتی که روی ساختار داده‌ها صورت می‌گیرد را در نظر گرفت.** مثلاً یک ساختار داده‌ای را که از مجموعه عناصر داده‌ای مختلف تشکیل شده، در نظر بگیرید. از نظر چند کارکرد نرم‌افزاری عمده باید در ساختار داده‌ها تغییراتی صورت گیرد. بر اساس ارزیابی صورت گرفته روی ساختمان داده، یک نوع داده انتزاعی برای استفاده در طراحی بعدی نرم افزار تعریف می‌شود. مشخصه این نوع داده انتزاعی ممکن است تا حد زیادی طراحی نرم‌افزاری را ساده سازد. [AHO83][2]

۳- **یک فرهنگ داده‌ای به‌وجود آورده و از آن برای تعریف طراحی داده‌ها و برنامه استفاده کنید.** مفهوم این فرهنگ لغت داده ای در فصل ۱۲ تشریح شده است. این فرهنگ مشخصاً نمایان‌گر روابط میان اشیای داده‌ای و محدودیت‌های عناصر ساختار داده‌ای است. الگوریتم‌هایی که باید از روابط خاصی استفاده کنند را می توان به‌راحتی تعریف کرد اگر که چنین فرهنگ داده‌ای وجود داشته باشد.

۴- **تصمیمات مربوط به سطوح پایین طراحی داده‌ها را باید تا اواخر فرآیند طراحی به تعویق انداخت.** ممکن است از یک پالایش مرحله‌ای برای طراحی داده‌ها استفاده کرد. یعنی در طول تحلیل نیازمندیها یا در طول کار طراحی داده‌ها، سازمان کلی داده‌ها را تعریف کرد و در طول طراحی در سطح اجزاء آن را به‌طور دقیق مشخص نمود. روش بالا به پایین در طراحی داده‌ها مزایایی دارد که با روش

---

1. Wasserman,A.
2. Aho,A.V.

بالا به پایین در طراحی نرم‌افزار شبیه هم هستند. صفات خاصه عمده ساختاری ابتدا طراحی و ارزیابی شده‌اند. به‌طوری‌که معماری داده‌ها ایجاد می‌شود.

۵- نمود ساختاری داده‌ها تنها باید برای پیمانه‌هایی شناخته شده باشد که مستقیماً از داده‌های موجود در ساختار استفاده می‌کنند. مفهوم پنهان‌سازی اطلاعات و مفهوم چسبندگی و اتصال که مربوط به آن است دیدگاه مهمی در مورد کیفیت طراحی نرم‌افزار ارائه می‌دهد. این اصل نشان‌گر اهمیت این مفاهیم و هم‌چنین اهمیت جداسازی دیدگاه منطقی یک شی داده‌ای از دیدگاه فیزیکی آن است.

۶- کتابخانه‌ای از ساختارهای داده‌ای مفید و عملیاتی که ممکن است در مورد آنها به‌کار رود، باید ایجاد گردد. عملیات و ساختارهای داده‌ای را باید به‌عنوان منبعی برای طراحی نرم‌افزار موردنظر قرار داد. ساختارهای داده‌ها را می‌توان برای قابلیت کاربرد مجدد طراحی نمود. کتابخانه‌ای از صفحات ساختمان داده‌ها (انواع داده‌های انتزاعی) می‌تواند میزان کار شناسایی و طراحی برای داده‌ها را کاهش دهد. **[WAS80][1]**

۷- طراحی نرم‌افزار و زبان برنامه‌نویسی باید خصوصیات و شناسایی انواع داده‌های انتزاعی را پشتیبانی کند. پیاده‌سازی یک ساختمان پیچیده داده‌ای در صورتی می‌تواند بسیار مشکل شود که هیچ وسیله‌ای برای شناسایی مستقیم ساختمان در زبان برنامه‌نویسی انتخاب شده برای اجرا وجود نداشته باشد.

اصول توصیف شده فوق مبنایی برای روش طراحی داده‌ها در سطح اجزاء تشکیل می‌دهند که می‌توان آن را در فعالیتهای تحلیل و طراحی تلفیق کرد.

۱۴-۳ سبک های معماری

وقتی یکسازنده از عبارت "Centre Hall Colonial" برای توصیف خانه استفاده می‌کند. اکثر افرادی که با خانه‌های آمریکایی آشنایی دارند می‌توانند تصویر کلی از خانه و طرح طبقه همکف را در ذهن مجسم کنند. معمار از یک سبک معماری[2] به‌عنوان یک مکانیسم توصیفی برای ایجاد تمایز بین سبک‌های مختلف استفاده کرده است. اما مهم‌تر این‌که سبک معماری الگویی برای ساخت نیز می‌باشد. باید جزییات بیشتر خانه تعریف شده، ابعاد نهایی آن مشخص گردد، مشخصه‌های سفارش شدهٔ آن افزوده گردیده و مواد ساختمانی تعیین گردند، اما این الگوی سبک Centre Hall Colonial است که یک سازنده را در کارش هدایت می‌کند.

---

1. Wasserman,A.
2. architectural style

نرم‌افزاری که برای سیستم‌های مبتنی بر کامپیوتر ساخته می‌شود نیز یکی از سبک‌های بی‌شمار معماری را نشان می‌دهد.[۱] هر سبک یک طبقه از سیستم را توصیف می‌کند که در برگیرندهٔ ۱) مجموعه‌ای از اجزایی[۲] است که (مثل پایگاه داده‌ای، پیمانه‌های محاسباتی)عمل مورد نیاز سیستم را انجام می‌دهند. ۲) مجموعه‌ای از پیوند دهندگان[۳] و رابطین است که ارتباط، هماهنگی‌ها و همکاری را در میان اجزاء امکان‌پذیر می‌سازند ۳) محدودیت‌هایی[۴] که مشخص می‌کنند چگونه اجزا را می‌توان تلفیق کرد تا یک سیستم تشکیل دهند و ۴) مدل‌هایی معنایی[۵] است که طراح را قادر به درک خواص کلی سیستم از طریق تحلیل خواص شناخته شده اجزای تشکیل‌دهنده می‌کند. در بخش بعدی الگوهای رایج به‌کار گرفته شده برای نرم‌افزار را در معماری بررسی می‌کنیم.[BAS98][۶]

ارجاع به وب

پروژه ABLE در CMU . مقالات و مثالهای مفیدی از سبک های معماری تهیه نموده است : tom.cscmu.edu/able

یک سبک معماری چیست؟

۱۴-۳-۱ گروه بندی مختصر سبک ها و الگوها

گرچه در طول ۵۰ سال گذشته صدها هزار سیستم مبتنی بر کامپیوتر به‌وجود آمده اما تعداد نسبتاً زیادی از آنها را می‌توان در یک تعداد محدودی از سبک‌های معماری قرار داد: ([BAS98][۷] و [SHA97][۸] و [BUS96][۹])

معماری‌های متمرکز بر داده‌ها ـ یک مخزن داده‌ای در مرکز این معماری قرار دارد و اغلب توسط دیگر اجزایی که به روزسازی، افزودن، حذف یا کارهای دیگر اصلاحی را در مورد مخزن انجام می‌دهند، قابل دسترسی است. شکل ۱۴-۱ سبک متمرکز بر داده‌ها را نشان می‌دهد. نرم‌افزار مخدوم به مخزن مرکزی دسترسی دارد. در بعضی موارد، مخزن داده‌ای منفعل[۱۰] است. یعنی نرم‌افزار مخدوم جدا از هرگونه تغییری در داده‌ها یا اعمال دیگر نرم‌افزار مخدوم ، به داده‌ها دسترسی دارد. گونه دیگری از این روش مخزن را به یک تخته سیاه تبدیل می‌کند که عباراتی را برای نرم‌افزار مخدوم در زمانی می‌فرستد که داده موردنظر مشتری تغییر می‌کند.

---

۱. واژگان سبک و الگو در این توضیح معادل واقع شده اند.

2.components

3.connectors

4.constraints

5.semantic models

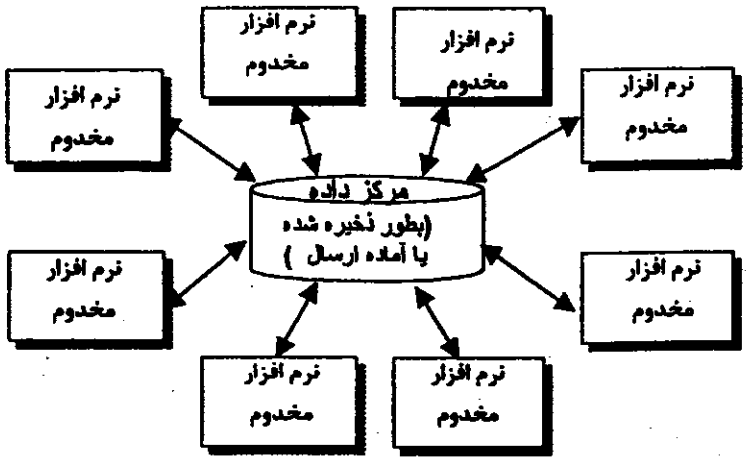
6.Bass,L.

7.Bass,L.

8.Shaw,M.and P.

9.Buschmann,F.

10.passive



شکل ۱۴-۱ معماری متمرکز داده ها

معماری‌های متمرکز برداده‌ها. این معماری قابلیت تلفیق‌پذیری[1] را افزایش می دهد. یعنی اجزای موجود را می‌توان تغییر داد یا اجزاء مخدوم جدیدی را به معماری بدون نگرانی در مورد مخدوم‌ها افزود. علاوه بر این، داده‌ها می‌توانند با استفاده از مکانیزم تخته سیاه از میان مخدوم‌ها عبور کنند (یعنی جزء تخته سیاه در خدمت هماهنگی انتقال اطلاعات بین مخدوم‌هاست). اجزاء سرویس‌گیرنده به‌طور مستقل فرآیندها را اجرا می‌کنند.

معماری‌های جریان داده‌ها. این معماری وقتی به‌کار گرفته می‌شود که داده‌های ورودی قرار است از طریق یک‌سری اجزاء محاسباتی یا تغییراتی به داده‌های خروجی تبدیل شوند. الگوی لوله و فیلتر[2](شکل ۱۴-۲ الف) دارای مجموعه‌ای از اجزاء است به نام فیلتر که توسط لوله‌هایی که داده‌ها را از یک جزء به دیگری انتقال می‌دهند، با هم مرتبطند. هر فیلتر به‌طور مستقل از اجزاء بالایی و پایینی کار می‌کند. فیلتر طوی طراحی شده که منتظر ورودی داده‌هایی با شکل معین بوده و داده‌های خروجی با فرم مشخص نیز تولید شود. در هر حال، فیلتر نیازمند شناخت کار فیلترهای مجاورش نیست.

اگر جریان داده‌ها به‌صورت یک خط از تغییرات تنزل یابد (شکل ۱۴-۲ ب)، به آن ترتیبی، دسته‌ای[3] یا سری ساخت می‌گویند. این الگو یک‌گروه اطلاعات را پذیرفته و سپس با استفاده از یک‌سری اجزاء پیاپی و متوالی آن را تغییر می‌دهد.

معماری فراخوانی و بازگشت ـ این سبک از معماری طراح نرم‌افزار (معمار سیستم) را قادر می‌سازد تا به ساختار برنامه‌ای دست یابد که از نظر اصلاح و ارزیابی نسبتاً ساده است.

نقل قول

استفاده از الگوها و سبک طراحی در دیسیپلین و انضباط مهندسی ناقذ و تاثیرگذار است. ماری شاو و دیوید گارلن

نقل قول

یک معمار خوب، کسی است که نظرات اساسی در خصوص دید استفاده کننده از محصول نهایی را در نظر می گیرد. نورمن سایمنسون

1.intergrability

2.pipe and filter pattern

3.batch sequential

تعدادی از سبک‌های فرعی موجود در این گروه عبارتند از:

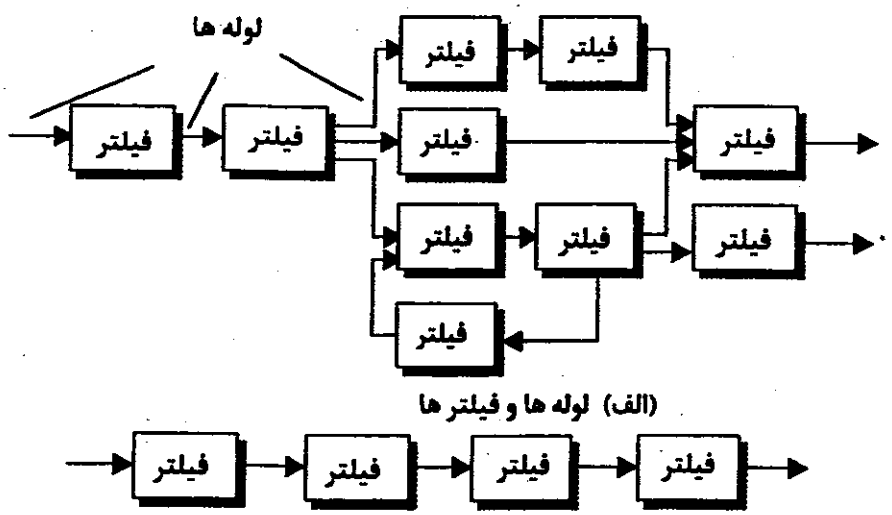
- *معماری برنامه اصلی/ برنامه فرعی*[1] . این ساختمان برنامه کلاسیک کارکرد را به سلسله مراتب کنترلی تجزیه می‌کند که در آن برنامه اصلی تعدادی از اجزاء را تحریک می‌سازد که در عوض آنها نیز ممکن است بقیه اجزاء را تحریک کنند. شکل ۱۴-۳ این نوع معمای را تشریح می‌کند.
- *معماری فراخوانی روال از راه دور*[2] . اجزاء معماری برنامه اصلی/ برنامه فرعی در کامپیوترهای مختلف یک شبکه توزیع می‌شوند.

ارجاع به فصلها

توضیح مفصلی از معماری شئ گرا در بخش چهارم ارائه شده است.

معماری‌های شی‌گرا . اجزای یک سیستم در برگیرنده داده‌ها و عملیاتی هستند که باید برای تغییر داده‌ها به‌کار روند. ارتباط و هماهنگی بین اجزا از طریق عبور پیام‌ها حاصل می‌شود.

معماری‌های لایه‌ای. ساختار اصلی این معماری در شکل ۱۴-۳ آمده است. تعدادی لایه‌های مختلف تعریف شده‌اند که هر کدام به عملیاتی دست می‌یابند که به‌طور گسترده‌ای به مجموعه دستورات ماشین نزدیک‌تر می‌شوند. در لایه خارجی‌تر، اجزاء در خدمت عملیات رابط کاربر هستند. در لایه داخلی‌تر، اجزاء کار ارتباط سیستم عامل را انجام می‌دهند. لایه‌های میانی خدمات استفاده و بهره‌برداری و عملیات کارکردی نرم‌افزار را مهیا می‌کنند.



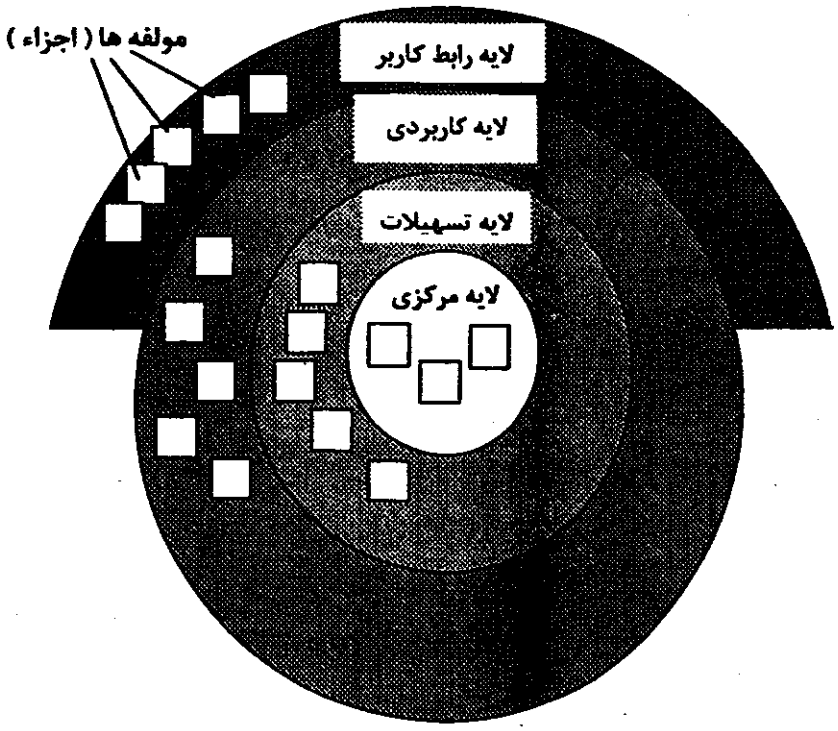
(الف) لوله ها و فیلتر ها

(ب) ترتیب دسته ای

شکل ۱۴ـ۲ معماری جریان

1.Main program/subprogram architectures

2.Remote procedure call architectures



شکل ۱۴ـ۳ معماری لایه لایه شده

سبک‌های معماری فوق‌الذکر تنها زیر مجموعه کوچکی از سبک‌های موجود برای طراح نرم‌افزار هستند.[1] زمانی که نیازمندیهای مهندسی مشخصه‌ها و محدودیت‌های سیستم را در راه ساخت، برملا ساخت، الگوی معماری یا سبک یا ترکیبی از الگوها که به بهترین نحو با مشخصه‌ها و محدودیت‌ها هم‌خوانی دارد، انتخاب می‌گردد. در بسیاری از موارد، ممکن است بیش از یک الگو مناسب شناخته شده و سبک‌های جایگزین دیگری طراحی و ارزیابی شوند.

۱۴-۳-۲ سازمان دهی و پالایش

پرسش

چگونه می توان یک سبک معماری به کار رفته را ارزیابی نمود؟

از آن‌جا که فرآیند طراحی اغلب چندین شق مختلف از نظر معماری پیش‌روی مهندس نرم‌افزار می‌گذارد، نکته مهم ایجاد مجموعه‌ای از معیارهای طراحی است که بتوان از آنها برای دسترسی به طراحی معماری مشتق شده استفاده نمود. سؤالات زیر، دیدگاهی در مورد سبک اشتقاقی ارائه می‌دهند:

کنترل. چگونه کنترل در معماری سامان‌دهی و مدیریت می‌شود؟ آیا سلسله مراتب کنترلی متمایزی وجود دارد و اگر دارد نقش اجزای مختلف این سلسله مراتب کنترلی چیست؟ چگونه این اجزاء کنترل را در سیستم انتقال می دهند؟ چگونه کنترل در میان اجزاء تقسیم می‌شود؟ توپولوژی کنترل چیست؟

(یعنی فرم هندسی[۲] که کنترل به‌خود می‌گیرد). آیا کنترل به‌صورت همگام صورت می‌گیرد یا غیر همگام عمل می‌کند؟

داده‌ها . چگونه داده‌ها بین اجزا ارتباط برقرار می‌کنند؟ آیا جریان داده‌ها ممتد است یا اشیاء داده‌ای به‌صورت پراکنده به درون سیستم وارد می‌شوند؟ حالت انتقال داده‌ها چیست؟ (یعنی آیا داده‌ها از یک جزء به جزء دیگر می‌روند یا این‌که در همه جا حضور دارند تا در میان اجزاء تقسیم شوند؟) آیا اجزاء داده‌ای (مثل تخته سیاه یا مخزن) وجود دارند و اگر وجود دارند نقششان چیست؟ چگونه اجزاء کارکردی با جزءهای داده ارتباط متقابل برقرار می‌کنند؟ آیا اجزاء داده‌ای منفعلند[۳] یا فعال[۴] (یعنی آیا به‌طور فعالانه با دیگر اجزاء سیستم ارتباط برقرار می‌کنند؟) چگونه داده‌ها و کنترل با هم در یک سیستم ارتباط دارند؟

این سؤالات، ارزیابی اولیه را از سیستم در مورد کیفیت طراحی در اختیار طراح قرار می‌دهند و پایه‌گذار تحلیل دقیق‌تری از معمای می‌شوند.

۱۴-۴ تحلیل طراحی ها ی انواع معماری های جایگزین

سؤالات مطرح شده در بخش قبلی، ارزیابی اولیه را از سبک معماری انتخابی برای سیستم فرضی مهیا می‌کنند. در هر حال، روش کامل‌تری برای ارزیابی کیفیت کار معماری در صورتی ضروری است که طراحی به‌صورت مؤثری صورت گیرد. در بخش‌های بعدی، دو نگرش مختلف را در مورد تحلیل طرح‌های معماری جایگزین در نظر می‌گیریم. اولین روش از روش تکرار برای ارزیابی توازن طراحی استفاده می‌کند. دومین رهیافت یک تکنیک شبه کمّی برای ارزیابی کیفیت طراحی فراهم می‌سازد.

نقل قول

ممکن است آن در طبقه زیرین باشد. به من اجازه دهید از پله ها بالا بروم و چک کنم. ام. سی. راشر

۱۴-۴-۱ یک شیوه تحلیل توازن معماری

مؤسسه مهندسی نرم‌افزار (SEI) روش ATAM یا روش تحلیل توازن معماری[۵] را ارائه کرده و فرآیند ارزیابی مکرر را برای معماری‌های نرم‌افزاری به‌وجود آورده است. فعالیت‌های تحلیل طراحی که در زیر آمده‌اند، به‌صورت تکرارشونده انجام می‌گیرند: [KAZ98][۱]

۱- جمع‌آوری (سناریوها) طرح‌ها[۲]. مجموعه‌ای از موارد استعمال ارائه شده‌اند تا نمایانگر سیستم از دیدگاه کاربر باشند.

ارجاع به وب

توضیح مفصلی در خصوص تحلیل و بررسی معماری در آدرس زیر وجود دارد : www.sei.cmu.edu/ata/ata-method.html

---

۱. برای توضیحات بیشتر درخصوص سبک ها و الگو های معماری به [SHA96] و [SHA97] و [BUS96] مراجعه نمایید.

۲. یک سلسله مراتب به فرم لایه لایه خواهد بود، اما دیگر مکانیسم های کنترل متمرکز و دود ، در یک سیستم خادم / مخدوم نیز قابل ملاحظه خواهند بود

3.passive

4.active

5.architecture trade-off analysis method

۲- **بدست آوردن نیازمندیها، محدودیت‌ها و توصیف محیط**[2]. این اطلاعات به‌عنوان بخشی از مهندسی مربوط نیازمندیها، لازمند و برای اطمینان از این امر استفاده می‌شوند که تمام نگرانی‌های مشتری، کاربر و سهام‌داران مورد توجه قرار گرفته‌اند.

۳- **توصیف سبک‌ها/ الگوهای معماری که برای بررسی طرح‌ها و نیازها انتخاب شده‌اند.**[4] این سبک‌ها را باید با استفاده از دیدگاه‌های معماری مورد توصیف قرار داد مثل:

- **دیدگاه پیمانه.**[5] برای تحلیل کارهای تخصیصی مربوط به جزء و درجه‌ای که نسبت به آن اطلاعات مخفی شده بازیابی می‌شوند.
- **دیدگاه پردازشی.**[6] برای تحلیل عملکرد سیستم.
- **دیدگاه جریان داده‌ای.**[7] برای تحلیل مقداری که نسبت به آن کار معماری، نیازمندیهای کارکردی را مرتفع می‌سازد.

ارجاع به فصلها
توضیح مفصلی از صفات خاصه کیفیت در فصلهای ۸ و ۱۹ ارائه شده است.

۴- **ارزیابی صفات خاصه کیفی با در نظر گرفتن هر صفت خاصه به‌طور جداگانه.**[8] تعداد صفات خاصه انتخابی برای تحلیل تابعی از زمان موجود برای بازبینی و مقداری است که نسبت به آن این نگرش‌ها با سیستم موردنظر مرتبط می‌شوند. نگرش‌های کیفی برای برآورد طراحی معماری شامل قابلیت اطمینان، عملکرد، امنیت، قابلیت نگهداری، انعطاف‌پذیری، آزمون پذیری، قابلیت حمل، استفاده مجدد و قابلیت عمل در همکاری می‌باشد.

۵- **شناسایی حساسیت صفات خاصه به صفات خاصه مختلف معماری برای سبک به‌خصوصی.**[9] این کار با ایجاد تغییرات کوچکی در ساختار و تعیین چگونگی حساسیت یک صفت خاصه کیفی مثلاً عملکرد نسبت به تغییر، انجام می‌گیرد. صفت خاصه‌ای که تا حد زیادی تحت تأثیر تنوع معماری قرار دارد، "نقاط حساسیت"[10] نامیده می‌شود.

1.Kazman,R. et al.

2.Collect scenarios

3.Elicit requirements,constraints,and environment description

4.Describe the architectural styles/patterns that have been chosen to chosen to address the scenarios and requirements

5.Modele view

6.Process view

7.Data flow view

8.Evaluate quality attributes by considering each attribute in isolation

9.Identify the sensitivity of quality attributes to various architectural attributes for a specific architectural style

10.sensitivity points

۶- متخصصان با استفاده از تحلیل حساسیت که در مرحله ۵ صورت گرفته، معماری‌های کاندیدا را پیشنهاد می‌کنند.[1] SEI این روش را به‌صورت زیر توصیف می‌کند [KAZ98][2]:

وقتی نقاط حساس معماری تعیین شد، پیدا کردن نقاط تعادل صرفاً عبارتست از شناسایی عناصر معماری که نسبت به آن صفات خاصه چندگانه حساس هستند. مثلاً، عملکرد معماری خادم ـ مخدوم ممکن است نسبت به تعداد خادم‌ها حساس باشد. (افزایش عملکرد در یک دامنه با افزایش تعداد خادم‌ها) ممکن است قابلیت دسترسی به آن معماری نیز مستقیماً با تعداد خادم‌ها تفاوت یابد.

امنیت سیستم نیز ممکن است با تعداد خادم‌ها تفاوت یابد (زیرا سیستم نقاط پتانسیل حمله بیشتری است.) تعداد خادم‌ها با توجه به این معماری نقطه تعادل است. این یک عنصر است که به‌طور بالقوه یکی از میان چندین می‌باشد و جایی است که در آن تعادل‌های معماری به‌طور آگاهانه یا ناآگاهانه ایجاد می‌شود.

شش مرحله فوق‌الذکر نمایان‌گر اولین تکرار ATAM است. بر اساس نتایج مراحل ۵ و ۶ بعضی از گزینه‌های معماری حذف می‌شوند، یک یا چند معماری باقی‌مانده نیز تغییر کرده و به‌طور دقیق‌تر ارائه می‌شوند و سپس مراحل ATAM مجدداً به‌کار گرفته می‌شوند.

۱۴-۴-۲ رهنمود کمی برای طراحی معماری

• یکی از مشکلات بی‌شماری که در پیش روی مهندسین نرم‌افزار در طول فرآیند طراحی قرار دارد کمبود کلی روش‌های کمی برای ارزیابی کیفیت طرح‌های پیشنهادی است. روش ATAM که در بخش ۱۴-۴-۱ مورد بحث قرار گرفت، نمایان‌گر یک رهیافت مفید و البته کیفی در تحلیل طراحی است.

آسادا[ASA96][3] یک‌سری مدل‌های ساده پیشنهاد کرد که به طراح در تعیین مقداری که نسبت به آن معماری خاصی معیارهای خوب از پیش تعریف شده‌ای را مرتفع می‌سازد، کمک می‌کند. این معیارها که گاهی به آن ابعاد طراحی[4] می‌گویند، اغلب در برگیرنده روش‌های کیفی تعریف شده در بخش آخر هستند یعنی قابلیت اطمینان‌عملکرد، ایمنی، قابلیت نگهداری، انعطاف‌پذیری، آزمون پذیری، قابلیت حمل، قابلیت کاربرد مجدد و عمل در همکاری با دیگر قسمت‌ها.

اولین مدل به نام تحلیل طیف[5] یک طراحی معماری را از نظر خوب بودن طیف، از بهترین تا بدترین طراحی‌های ممکن، ارزیابی می‌کند. وقتی‌که معماری نرم‌افزاری ارائه شد. این طرح با اختصاص "امتیازی"

1.Critique candidate architectures(developed in step 3)using the sensitivity analysis conducted in step 5

2.Kazman,R. et al.

3.Asada,T.

4.design dimensions

5.spectrum analysis

به هر یک از ابعاد طراحیش، ارزیابی گردید. این امتیازات ابعاد جمع زده می‌شوند تا امتیاز کل یعنی S طراحی را در کل تعیین کنند.

سپس بدترین امتیازات موردی[1] به طراحی مفروض اختصاص می‌یابند و مجموع S_w برای بدترین مورد معماری محاسبه می‌شود. بهترین امتیاز مورد یعنی S_b برای طراحی بهینه محاسبه می‌شود.[2] سپس شاخص طیف[3] یا I_s را با استفاده از معادله زیر محاسبه می‌کنیم:

$$I_s = [(S - S_w) / (S_b - S_w)] \times 100$$

شاخص طیف نشانگر مقداری است که مشخص کننده طراحی رهیافتهای پیشنهادی معماری نسبت به یک سیستم بهینه در یکسری انتخابهای منطقی برای طراحی می‌باشد.

اگر تغییراتی در طراحی پیشنهادی صورت گیرد یا اگر یک طراحی کاملاً جدید پیشنهاد شود، شاخص‌های طیف برای هر دو مورد مقایسه شده و یک شاخص پیشرفت[4] یا I_{mp} محاسبه می‌گردد یعنی:

$$I_{mp} = I_{s1} - I_{s2}$$

این کار، یک شاخص نسبی از پیشرفت صورت گرفته بر اساس تغییرات معماری یا معماری جدید پیشنهادی در اختیار طراح قرار می‌دهد. اگر I_{mp} مقداری مثبت باشد پس می‌توان نتیجه‌گیری کرد که سیستم ۱ نسبت به ۲ پیشرفت داشته است.

تحلیل انتخاب طراحی[5] مدل دیگری است که نیازمند مجموعه‌ای از ابعاد طراحی است که باید تعریف شوند. سپس معماری پیشنهادی ارزیابی می‌گردد تا تعداد ابعاد طراحی که در مقایسه با سیستم ایده‌آل بدست می‌آیند، تعیین گردد. مثلاً اگر یک معماری پیشنهادی به جزء عالی قابل استفاده مجدد برسد و این بعد برای یک سیستم ایده‌آل لازم باشد بعد قابلیت استفاده مجدد حاصل گردیده است. اگر معماری پیشنهادی ایمنی ضعیفی داشته باشد و ایمنی قوی لازم باشد، این بعد طرح بدست نمی‌آید.

شاخص انتخاب طراحی[6] یا d را بر اساس رابطه زیر محاسبه می‌کنیم:

$$d = (N_s / N_a) \times 100$$

نقل قول

یک پزشک می تواند اشتباهات خویش را دفن نماید، اما یک معمار تنها می تواند با مشتریان خود توصیه کند که درخت مو بکارند. فرانک لیود

1.Worst – Case Scores

ط: اند قابل پیاده سازی باشد، تصور نشودکه هزینه بهترین مشکل گشا خواهد بود.

۲. طراحی باید بهینه باشد، اما قیود و محدودیتها هزینه ها یا دیگر عوامل وقوع این امر را اجازه نمی دهند.

3.spectrum index

4.improvement index

5.Design selection analysis

6.design selection index

که در آن N_s تعداد ابعاد طراحی بدست آمده توسط معماری پیشنهادی و N_a مجموع کل ابعاد در فضای طراحی است. هر چه شاخص انتخاب طراحی بیشتر باشد، رهیافتهای معماری پیشنهادی به سیستم ایده‌آل نزدیک می‌شوند.

تحلیل همکاری[1]، دلایلی را شناسایی می‌کند که به‌واسطه آنها مجموعه‌ای از طرحهای انتخابی امتیاز کمتری را نسبت به طرحهای دیگر می‌گیرند.[ASA96] [2] با یادآوری بحث ارائه کارکرد کیفی در فصل ۱۱، تحلیل مقدار برای تعیین اولویت نسبی نیازمندیهای تعیین شده در طول نمایش تابع، اطلاعات و وظیفه، صورت می‌گیرد. مجموعه‌ای از مکانیزمهای تجسم، شناسایی می‌گردند. تمام نیازمندیهای مشتری (که با استفاده از QFD تعیین شده) فهرستبندی گردیده و یک ماتریس مرجع متقاطع ایجاد می‌گردد. سلول‌های این ماتریس یا شبکه نشانگر نیروی نسبی ارتباط میان یک مکانیزم تجسم و نیاز به هر معماری جایگزین می‌باشد. این مورد را گاهی "فضای اندازه‌گیری شدهٔ طراحی(QDS)"[3] می‌نامند ، QDS به‌عنوان یک مدل مسطح از نظر پیاده‌سازی ساده است و می‌توان برای فهمیدن این موضوع که چرا مجموعه‌ای از انتخابهای طراحی امتیاز کمتری نسبت به بقیه می‌گیرند، آن را مورد استفاده قرار داد.

۱۴-۴-۳ پیچیدگی معماری

یک روش مفید برای ارزیابی پیچیدگی کلی معماری پیشنهادی، عبارتست از در نظر گرفتن وابستگی میان اجزاء درون معماری. این وابستگی‌ها ناشی از جریان کنترل/ اطلاعات درون سیستم می‌باشند.

ژائو سه نوع وابستگی ارائه می‌دهد:[ZHA98] [4]

وابستگی‌های مشترک[5] نمایانگر ارتباطات وابسته‌ای در میان مصرف‌کنندگانی هستند که از منبعی یکسان استفاده کرده یا تولیدکنندگانی که برای مصرف‌کنندگانی یکسان تولید می‌کنند. مثلاً، در مورد دو جزء v,u ، اگر v,u یکسانی را مورد ارجاع قرار دهند، یک رابطه وابستگی مشترک بین v,u وجود دارد.

وابستگی‌های جریان[6] نمایانگر ارتباطات وابستگی میان تولیدکننده و مصرف‌کننده‌های منبع است. مثلاً در مورد دو جزء v,u، اگر باید u قبل از جریان یافتن کنترل در v تکمیل شود یا اگر u به‌وسیله پارامترهایی با v ارتباط برقرار می‌کند، در این‌صورت یک جریان وابستگی میان این دو وجود دارد.

1.Contribution analysis

2.Asada,T.

3.quantified design space

4.Zhao, J.

5 sharing dependencies

6.Flow dependencies

وابستگی‌های مقیدشده[1] نمایانگر محدودیت‌ها و قیودی در جریان نسبی کنترل میان مجموعه‌ای از فعالیت‌ها هستند. مثلاً در مورد دو جزء v,u، آنها نمی‌توانند در یک زمان اجرا شوند پس یک رابطه وابستگی محدود و مقید شده بین آنها وجود دارد.

وابستگی‌های مشترک و جریان که مورد توجه ژائو قرار گرفت از بعضی جهات شبیه مفهوم جفت شدن (پیوستگی و اتصال) هستند که در فصل ۱۳ بررسی شد. در فصل ۱۹ معیارهای ساده‌ای برای ارزیابی این وابستگی‌ها مورد بحث قرار می‌گیرد.

۱۴-۵ نگاشت نیازمندیها در یک معماری نرم افزار

در فصل ۱۳ توجه کردیم که نیازمندیهای نرم‌افزاری را می‌توان به طرق مختلف در مدل طراحی مطرح نمود. سبکهای معماری مورد بحث در بخش ۱۴-۳-۱ نمایانگر معماری‌های کاملاً متفاوتی می‌باشند، پس جای تعجبی نیست که یک نگاشت که انتقال از مدل نیازمندیها را به مدل طراحی با موفقیت انجام داده باشد، وجود ندارد. در واقع، نگاشتهای مختلف سبکهای معماری هنوز وجود ندارد و طراح باید به تفسیر نیازمندیها در طراحی از نظر این سبکها، به شیوه‌ای مخصوص دست یابد.

به‌منظور تشریح یک روش در کشیدن نقشه معماری (نگاشت معماری)، فراخوانی و بازگشت معماری را در نظر می‌گیریم، یعنی یک ساختمان بسیار معمول برای انواع سیستمها.[2] تکنیک نگاشت موردنظر، طراح را قادر می‌سازد تا فراخوانی و بازگشت پیچیدهٔ منطقی معماری‌ها را از روی دیاگرام جریان داده‌ها در مدل مقتضیات ارائه دهد. این تکنیک که گاهی طراحی ساخت‌یافته[3] نامیده می‌شود، ریشه در مفاهیم اولیه طراحی دارد که بر پیمانه سازی[DEN73][4] و برنامه‌نویسی ساخت‌یافته تأکید دارند.([DAH72][5] و [LIN79][6]) استیونس، میر و کنستانتین[STE74][7] از طرفداران اولیه طراحی نرم‌افزار بر اساس جریان داده‌ای طی یک سیستم بودند. کار اولیه بازنگری شد و در کتابهای نوشته میر و یوردون و کنستانتین به چاپ رسید.[MYE78][8] و [YOU79][9]

1.Constrained dependencies

۲. این امر نیز مهم است که توجه نمائیم، معماری فراخوانی و بازگشت می تواند در کنار بسیاری از معماری های دیگر توضیح داده شده در ابتدای این فصل قرار گیرد. برای مثال، معماری یک جزء یا بیشتر از یک جزء معماری خادم / مخدوم می تواند به صورت فراخوانی و بازگشت باشد.

3.structured design

4.Dennis,J.B.

5.Dah.O.

6.Linger,R.C.

7.Stevens,W.

8.Myers,G.

9.Yourdon,E. and L.

در یک معماری رفت و برگشت کدام گامها را برای ترسیم DFD باید برداریم؟

طراحی ساختیافته شده اغلب به‌عنوان یک روش طراحی مبتنی بر جریان داده‌ای معرفی می‌گردد زیرا انتقال راحت از دیاگرام جریان داده‌ای (DFD) را به معماری نرم‌افزار فراهم می‌سازد.[1] کار انتقال از جریان اطلاعات به ساختار برنامه به‌عنوان بخشی از فرآیند مرحله ۶ با موفقیت به انجام می‌رسد: ۱) نوع جریان اطلاعاتی تعیین می‌گردد. ۲) سرحدات جریان مشخص می‌شوند. ۳) DFD در ساختمان برنامه نگاشت می‌شود. ۴) سلسله مراتب کنترلی تعیین می‌گردد. ۵) ساختمان بدست آمده با استفاده از معیارهای طراحی و علوم تجربی بازنگری می‌شود و ۶) توصیف معماری بازنگری و تعیین می‌گردد.

نوع جریان اطلاعات، محرک رهیافت نگاشت مورد نیاز در مرحله ۳ است. در بخش‌های بعدی دو نوع جریان را بررسی می‌کنیم.

نمودارهای جریان داده ها، به تفصیل در فصل ۱۲ توضیح داده شده اند.

۱۴-۵-۱جریان تبدیلات

اگر مدل سیستم بنیادی را به‌خاطر بیاورید (دیاگرام جریان داده‌ای سطح صفر)، اطلاعات باید به شکل دنیای بیرونی وارد و خارج از نرم‌افزار شوند. مثلاً اطلاعات تایپ شده، تن‌های خط تلفن و تصاویر ویدیویی در یک برنامه کاربردی پند رسانه‌ای همگی از اشکال اطلاعات دنیای بیرونی هستند. چنین اطلاعات برونی باید به‌صورت درونی برای پردازش، درآیند. اطلاعات در طول مسیرهایی که اطلاعات برونی را به درونی تبدیل می‌کنند، وارد سیستم می‌شوند. این مسیرها به‌عنوان جریان در حال ورود[2]، شناخته می‌شوند. در قسمت مرکزی نرم‌افزار، انتقال رخ می‌دهد. اطلاعات ورودی از مرکز تغییردهنده گذشته و در طول مسیری به حرکت درمی‌آیند که اکنون به سمت خروجی نرم‌افزار می‌رود. اطلاعاتی که در طول این مسیرها حرکت می‌کنند جریان خروجی[3] نامیده می‌شوند. جریان کلی داده‌ها به‌صورت سریالی رخ داده و یک یا تنها چند مسیر مستقیم را دنبال می‌کنند.[4] وقتی بخشی از دیاگرام جریان داده‌ای این مشخصه‌ها را نشان می‌دهد، جریان تبدیل[5] حضور دارد.

---

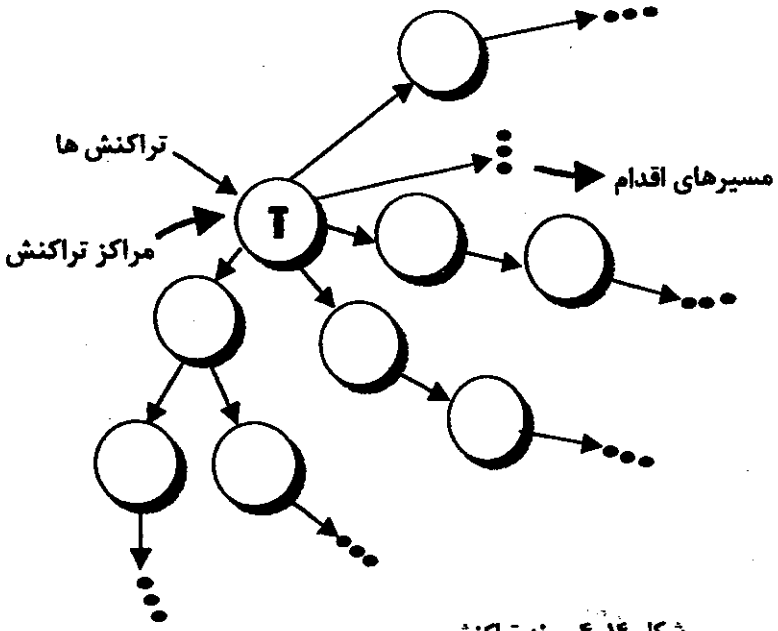
۱. باید توجه نمود که دیگر عناصر مدل تحلیلی نیز (مانند فرهنگ داده ها، مشخصه های فرآیندها، مشخصه های کنترلی) طی شیوه نگاشت به کار می روند.

2.incoming flow

3.outgoing flow

۴. یک نگاشت صریح برای این نوع جریان اطلاعات، معماری جریان داده ها می باشد که در بخش ۱۴-۳-۱ توضیح داده شد. با این وجود در بسیاری از موارد، معماری جریان داده ها بهترین انتخاب برای سیستم های پیچیده نخواهد بود. برای مثال سیستم هایی که تغییرات متعدد و پردازش های غیر ترتیبی دارن، با نمودار جریان داده ها قابل نمایش نیستند.

5.transform flow



شکل ۱۴-۴ روند تراکنش

۱۴-۵-۲ جریان تراکنش

مدل سیستم بنیادی به جریان تبدیل اشاره دارد. بنابراین ممکن است همه جریان داده‌ای در این طبقه قرار گیرد. جریان اطلاعات اغلب به‌وسیله یک قلم داده‌ای به نام مبادله[1] یا قرارداد توصیف می‌شود(تراکنش) که در طول یکی از چندین مسیر دیگر جریان اطلاعاتی را اصلاح می‌کند. وقتی DFD به شکل نشان داده شده در شکل ۱۴-۴ می‌رسد، جریان تراکنش[2] حضور یافته است.

جریان تراکنش با حرکت داده‌ها در طول مسیر توصیف می‌شود که اطلاعات جهان بیرونی را به یک قرارداد تبدیل می‌کند. این تراکنش ارزیابی شده و بر اساس ارزشش، جریان در طول یکی از مسیرهای متعدد[3] اقدام، آغاز می‌گردد.

باید توجه داشت که در یک DFD برای یک سیستم بزرگ جریان تبدیل و تراکنش ممکن است هر دو وجود داشته باشند. مثلاً در یک جریان مبتنی بر تراکنش، جریان اطلاعات ممکن است در طول یک مسیر تبدیل دارای مشخصه‌های جریان تبدیل باشند.

۱۴-۶ نگاشت تبدیلات

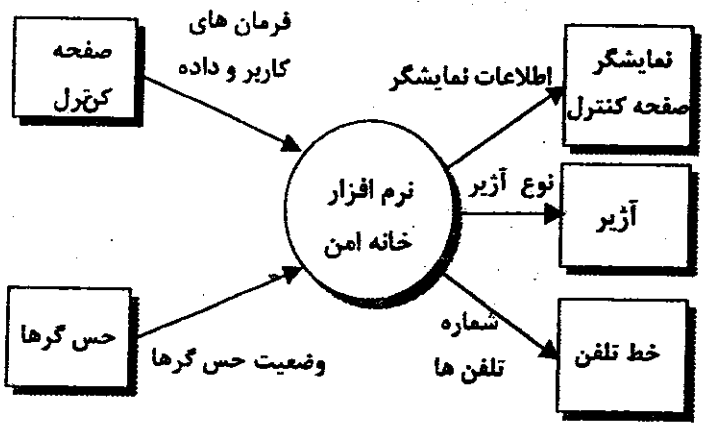
نگاشت تبدیلات[4]، تبدیل مجموعه‌ای از مراحل طراحی است که به DFD دارای مشخصات جریان تبدیل، امکان می‌دهد به سبک به‌خصوصی در معماری طراحی شود. در این بخش طراحی تغییر با

1.transaction
2.transaction flow
3.action paths
4.Transform mapping

به‌کارگیری مراحل طراحی در یک سیستم نمونه توصیف شده یعنی بخشی از نرم‌افزار امنیتی خانهٔ امن[1] که در فصول پیش‌تر معرفی شده بود.

۱۴-۶-۱ یک مثال

سیستم حفاظتی خانهٔ امن که پیش‌تر در این کتاب معرفی شد نماینده بسیاری از محصولات و سیستم‌های مبتنی بر کامپیوتر امروزی است. محصول، جهان واقعی را مشاهده کرده و در مقابل تغییراتی که با آنها برخورد دارد، واکنش نشان می‌دهد. هم‌چنین از طریق یک‌سری داده‌های ورودی تایپی و نمایش‌های حروف و ارقام با کاربر رابطه متقابل برقرار می‌سازد. دیاگرام جریان داده‌ای در سطح صفر برای خانهٔ امن که از فصل ۱۲ برگرفته شده، در شکل ۱۴-۵ آمده است.



شکل ۱۴-۵ سطح متن برای DFD (نمودار جریان داده) خانه امن

در طول تحلیل نیازمندیها مدل‌های دقیق‌تری از جریان برای خانهٔ امن به‌وجود می‌آید. علاوه بر آن، مشخصه‌های کنترل و فرآیند، واژه‌نامه داده‌ای و مدل‌های مختلف رفتاری نیز به‌وجود می‌آیند.

۱۴-۶-۲ گامهای طراحی

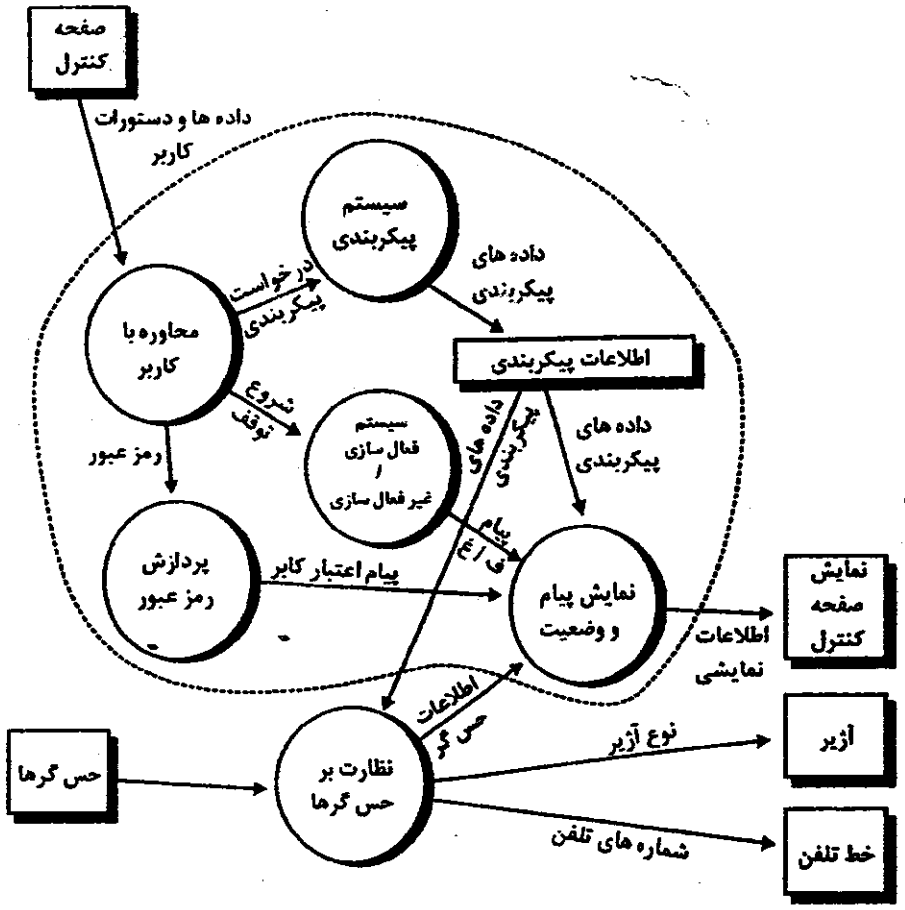
مثال فوق برای تشریح هر مرحله از نگاشت تبدیل استفاده می‌شود. این مراحل با یک بررسی مجدد از کار انجام شده در طول تحلیل نیازمندیها، شروع شده و سپس به‌سوی طراحی معماری نرم‌افزار حرکت می‌کند.

مرحله اول- بازنگری مدل سیستم بنیادی

مدل سیستم بنیادی در برگیرندهٔ سطح صفر DFD و اطلاعات پشتیبان است. در عمل مرحله طراحی با ارزیابی مشخصه‌های سیستم[1] و مشخصه‌های نیازمندیهای نرم‌افزار[2] شروع می‌گردد. هر دو سند،

1.SafeHome

ساختار و جریان اطلاعات را در رابط نرم‌افزاری توصیف می‌کنند. شکل ۱۴-۵ و ۱۴-۶ سطح صفر و یک را در جریان داده برای نرم‌افزار خانهٔ امن نشان می‌دهد.



ف : فعال سازی
غ : غیر فعال سازی

شکل ۱۴ - ۶ DFD سطح ۱ برای خانه امن

مرحله دوم. بازنگری و پالایش نمودارهای جریان داده‌ای در مورد نرم‌افزار

اطلاعات بدست آمده از مدل‌های تحلیل که در مشخصات نیازمندیهای نرم افزاری است برای جزییات بیشتر مورد اصلاح قرار می‌گیرند. مثلاً، سطح ۲ DFD در مورد سنسورهای مونیتور[3](شکل ۱۴-۷) بررسی می‌گردند. و نمودار جریان داده‌ای سطح ۳ همان‌گونه که در شکل ۱۴-۸ آمده بدست می‌آید. در سطح ۳، هر تغییر در نمودار جریان داده‌ها نشانگر انسجام نسبتاً بیشتری است. یعنی فرآیند مورد اشاره هر تبدیل

توصیه

اگر DFD را در سطوح بیشتری (در این زمان) طبقه بندی کنیم، در ساخت حبابهایی با پیوستگی زیاد کوشیده ایم.

1.System Specification

2.Software Requirements Specification

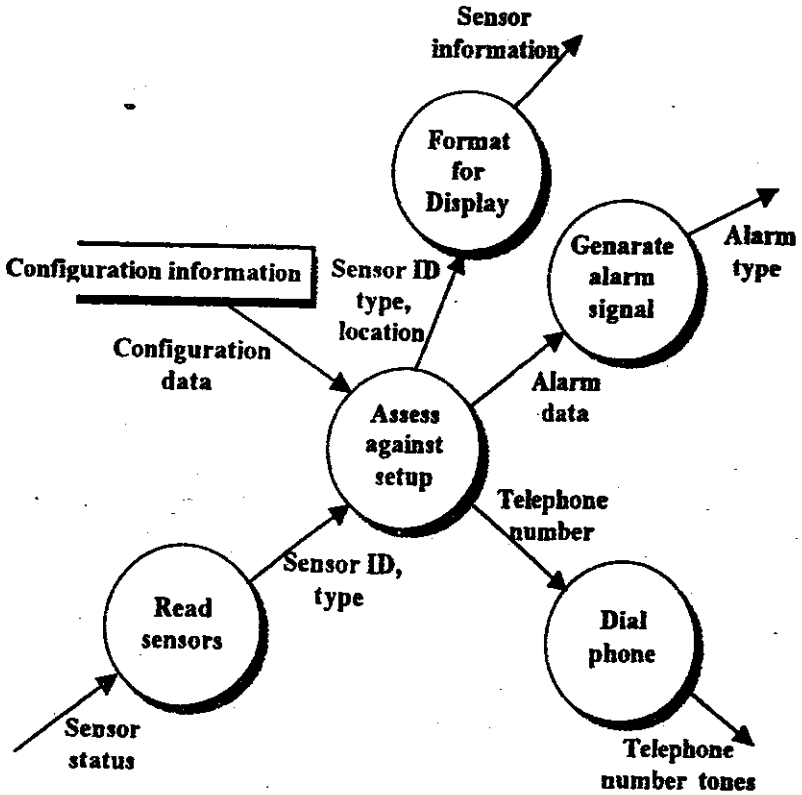
3.monitor sensors

یک عمل مشخص و تنها انجام می‌دهد که می‌توان آن را یک پیمانه[1] در نرم‌افزار خانهٔ امن نامید. بنابراین، DFD در شکل ۱۴-۸ دارای جزییات کافی برای اولین پالایش در طراحی معماری در مورد سیستم فرعی سنسورهای مونیتور است و ما بدون پالایش بیشتر به آن می‌پردازیم.

مرحله سوم. تعیین این‌که آیا DFD دارای مشخصات جریان تغییر یا مبادله‌ای است یا خیر.

ما اغلب طی مدل تحلیلی با هر دو نوع جریان داده ها مواجهیم. جریان ها قسمت بندی شده، ساختار داده ای یا به کارگیری نگاشت مناسب به دست می آید.

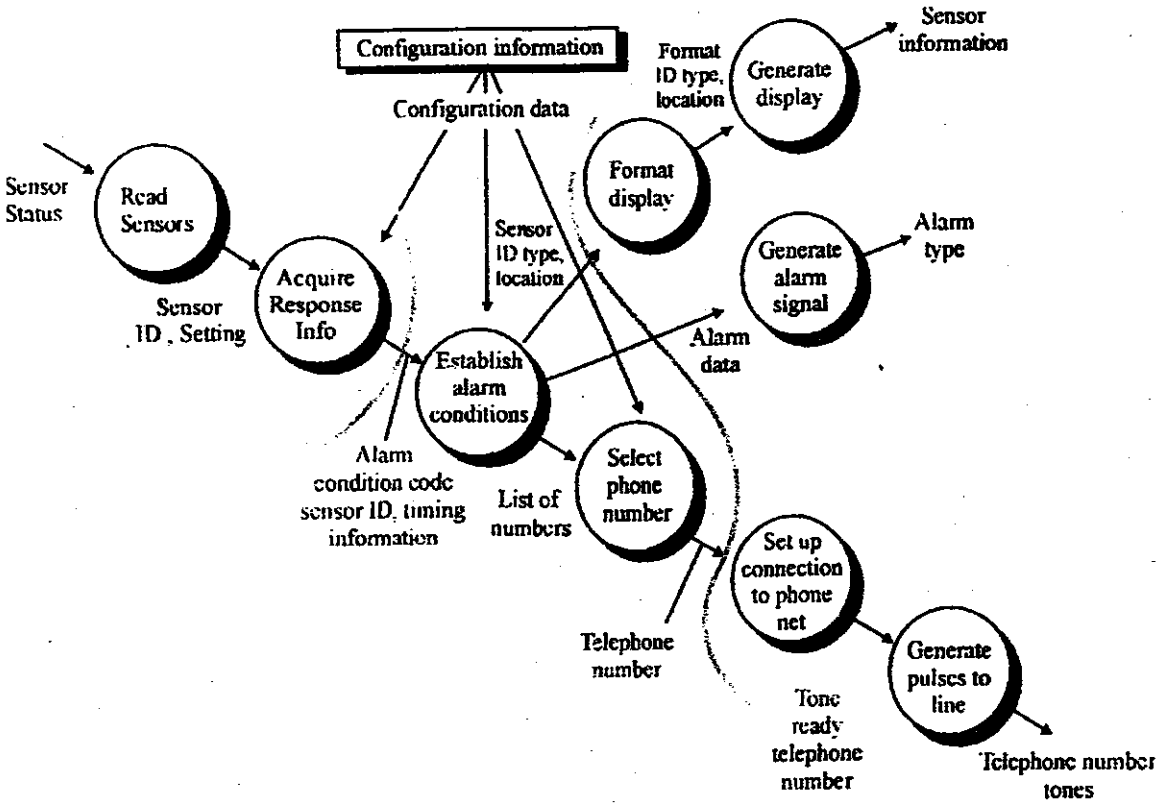
به‌طور کلی، جریان اطلاعات در یک سیستم را همیشه می‌توان به‌عنوان پالایش ارائه داد. وقتی به یک مشخصه معین تراکنش (شکل ۱۴-۴) برمی‌خوریم، یک نگاشت متفاوت در طراحی توصیه می‌شود. در این مرحله، طراح همهٔ مشخصه‌های جریان را بر اساس پی‌گیری موجودیت DFD انتخاب می‌کند. علاوه بر این، مناطق موضعی جریان تبدیل یا تراکنش مجزا می‌شوند. این جریانات فرعی[2] را می‌توان برای پالایش معماری برنامه مشتق از مشخصه‌های همه‌گیر فوق، استفاده نمود. اکنون ما توجه خود را تنها بر جریان داده‌های سیستم فرعی سنسورهای مونیتور که در شکل ۱۴-۸ آمده‌اند، متمرکز می‌کنیم.



شکل ۱۴-۷ سطح دوم DFD (نمودار جریان داده‌ها) برای فرآیند حسگرهای نمایشگر

۱. استفاده از واژه پیمانه در این فصل معادل جزء و مولفه در توضیحات اولیه معماری نرم افزار است.

2. subflows



شکل ۱۴ - ۸ سطح سوم DFD (نمودار جریان داده) برای خانه امن با مرزهای روند

با ارزیابی DFD (شکل ۱۴-۸) می‌بینیم که داده‌ها از یک مسیر ورودی وارد نرم‌افزار شده و از سه مسیر از آن خارج می‌شوند. هیچ مرکز مبادله مشخصی مورد اشاره قرار نگرفته (گرچه شرایط هشداردهنده این تبدیل را می‌توان درک نمود.) بنابراین، یک مشخصه کلی تبدیل برای جریان اطلاعات، فرض می‌شود.

مرحله چهارم. با مشخص کردن سرحدات جریان ورودی و خروجی مرکز تبدیل جدا می‌شود.

توصیه

مرزهای جریان را هنگام تلاش برای یافتن ساختارهای جایگزین برنامه، جابجا کنید. این امر زمان کوتاهی می برد، اما نتایجی شگرف دارد.

در بخش قبلی جریان ورودی به‌عنوان مسیری توصیف شد که در آن اطلاعات از فرم بیرونی به درونی تغییر می‌یابند و جریان خروجی از درونی به بیرونی تبدیل می‌شود. سرحدات جریان ورودی و خروجی در معرض تفسیر قرار دارند. یعنی طراحان مختلف ممکن است نقاط کاملاً متفاوت را در جریان، به‌عنوان محل سرحدات انتخاب کنند. در واقع، راه‌حل‌های مختلف طراحان با جابه‌جایی مختلف سرحدات جریان، بدست می‌آیند. گرچه باید در هنگام انتخاب سرحداد دقت نمود، اما معمولاً واریانس یک حباب در طول مسیر جریان تأثیر کمی روی ساختار برنامهٔ نهایی دارد.

به‌طور مثال سرحدات جریان به‌صورت منحنی‌های سیاهی به‌صورت عمودی در طول جریان در شکل ۱۴-۸ نشان داده شده‌اند. تبدیلاتی که مرکز تبدیل را تشکیل می‌دهند در دو سرحد هاشور خورده، قرار دارند که در شکل از بالا به پایین قرار گرفته‌اند. در مورد تنظیم مجدد هر سرحد می‌توان بحث نمود (مثلاً

یک سرحد جداکننده جریان ورودی سنسورهای خواندن[1]، از اطلاعات واکنش اکتسابی[2] را می‌توان پیشنهاد کرد.) تأکید در این مرحله از طراحی باید روی انتخاب سرحدات منطقی باشد تا تکرار طولانی در مورد محل تقسیم‌بندی‌ها.

مرحله پنجم. انجام "فاکتورگیری سطح نخست"

ساختار برنامه نمایان‌گر توزیع کنترل از بالا به پایین است. عمل ساخت منجر به ساختار برنامه‌ای می‌شود که درآن پیمانه‌های سطح فوقانی کار تصمیم‌گیری را انجام داده و پیمانه‌های سطح تحتانی اکثراً کار محاسبات ورودی و خروجی را انجام می‌دهند. پیمانه‌های سطح میانی، بعضی کنترل‌ها و مقداری از کار را، انجام می‌دهند.

وقتی با جریان تبدیل روبه‌روی می‌شویم DFD در ساختار به‌خصوصی نگاشت می‌شود که برای ورودی، تغییر و پردازش اطلاعات خروجی، کنترل ایجاد می‌کند. این عمل فاکتورگیری اولین سطح در مورد سیستم فرعی سنسورهای مونیتور در شکل ۱۴-۹ تشریح شده است. کنترل‌کننده اصلی (به نام اجراکننده سنسورهای مونیتور)[3] در بالای ساختار برنامه قرار گرفته و در کار هماهنگی توابع کنترلی فرعی زیر شرکت دارند:

- کنترل‌کننده پردازشگر اطلاعات ورودی به نام کنترل‌کننده ورودی سنسور که دریافت داده‌های ورودی را کنترل می‌کند.
- کنترل‌کننده جریان تبدیل، به نام کنترل‌کننده شرایط هشداردهنده که بر تمام عملیات انجام گرفته روی داده‌ها در فرم درونی، نظارت دارد (مثلاً پیمانه‌ای که رویه‌های مختلف تبدیل داده‌ای را موجب می‌شود.)
- کنترل‌کننده پردازش اطلاعات خروجی به نام کنترل‌کننده خروجی هشداردهنده که تولید اطلاعات خروجی را هماهنگ می‌سازد.

گرچه یک ساختار سه بخشی در شکل ۱۴-۹ مورد اشاره قرار گرفته، اما جریانات پیچیده در سیستم‌های بزرگ ممکن است، دو یا چند پیمانه کنترلی به هر تابع کنترلی کلی که در بالا توصیف شده است، اختصاص دهند. تعداد پیمانه‌ها در اولین سطح، محدود به حداقل مقداری است که می‌تواند به توابع کنترلی دست یافته و هنوز هم مشخصه‌های انسجام و پیوند (چسبیدگی و پیوستگی) را به‌خوبی حفظ کند.

توصیه

در این مرحله از خود تعصب به خرج ندهید. ممکن است بسته به میزان پیچیدگی سیستم مورد ساخت، دو کنترل یا بیشتر را برای پردازش ورودی ها یا محاسبات به کار برید. اگر عقل سلیم، چنین رهیافتی را حکم کرد، آن را انجام دهید.

1.read sensors

2.acquire response

3.monitor sensors executive

مرحله ۶ انجام "دومین سطح فاکتورگیری"

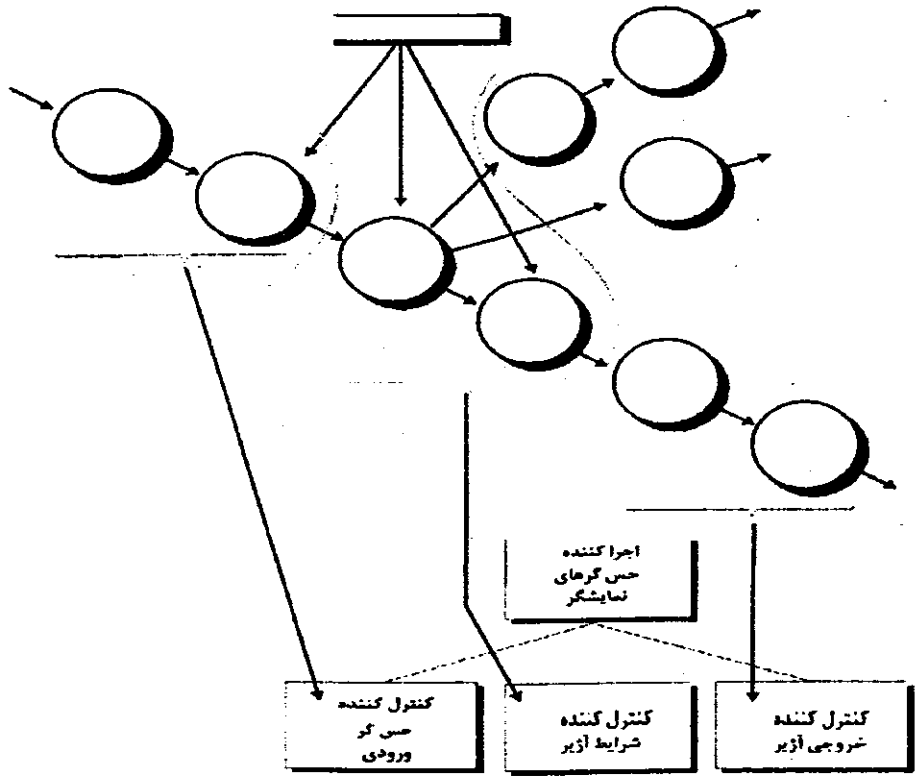
توصیه

" پیمانه های کارگر " در ساختار برنامه کمتر استفاده کنید. این امر منجر به معماری ای می شود که راحت تر تغییر خواهد یافت.

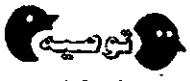
دومین نگاشت تبدیلهای (حبابها) مستقل یک DFD در پیمانه‌های مناسب معماری، حاصل می‌گردد. تغییرات که در سرحد مرکز تبدیل آغاز شده و به سمت خروجی در طول ورودی حرکت کرده و بعد به مسیرهای خروجی می‌روند، در سطوح فرعی ساختار نرم‌افزار نگاشت می‌شوند. روش کلی فاکتورگیزی سطح دوم، در مورد جریان داده‌های خانهٔ امن در شکل ۱۴-۱۰ تشریح شده است.

گره شکل ۱۴-۱۰ نگاشت یک به یک بین پیمانه‌های نرم‌افزار و تبدیلات را نشان می‌دهد اما اغلب نگاشته‌های مختلفی بوجود می‌آیند. می‌توان دو یا حتی سه حباب را بهم متصل نموده و به‌عنوان یک پیمانه نمایش داد (یا بازآوری مشکلات بالقوه مربوط به انسجام و یکپارچگی) یا این‌که ممکن است یک حباب به دو یا چند پیمانه بسط یابد. اقدامات و ملاحظات عملی در کیفیت طراحی نتیجه فاکتورگیری دومین سطح را تعیین می‌کنند. بازنگری و پالایش ممکن است منجر به تغییراتی در این ساختار شود اما می‌توان به‌عنوان اولین تکرار طراحی، عمل نماید.

فاکتورگیری دومین سطح در مورد جریان ورودی به همین شکل است. فاکتورگیری با حرکت از سرحد مرکز تبدیل به سمت خارج در طرف جریان ورودی، حاصل می‌گردد. مرکز تبدیل نرم‌افزار سیستم فرعی سنسورهای مونیتور کمی متفاوت‌تر نگاشت می‌شود. هر تبدیل داده‌ای یا تغییرات محاسبه‌ای در بخش تبدیل DFD در قسمت فرعی پیمانه در کنترل‌کننده تبدیل نگاشت می‌شود. معماری کامل اولین تکرار در شکل ۱۴-۱۱ نشان داده شده است.



شکل ۱۴-۹ سطح نخست حس‌گرهای نمایشگر



پیمانه های تکراری کنترل را حذف نمایید. این بدان معناست که اگر یک پیمانه کاری جز کنترل پیمانه دیگر ندارد، این کنترل می تواند به سطح بالاتری منتقل شود.

پیمانه ای که به شیوهٔ فوق نگاشت شده و در شکل ۱۴-۱۱ نشان داده شده است، نمایان‌گر طراحی اولیه معماری نرم‌افزاری است. گرچه پیمانه‌ها به شیوه‌ای نام‌گذاری می‌شوند که نشان‌گر کارکرد باشد، اما یک خلاصه توضیحی برای فرآیند برای هر کدام باید نوشته شود. این توضیحات شامل موارد زیر می‌باشد:

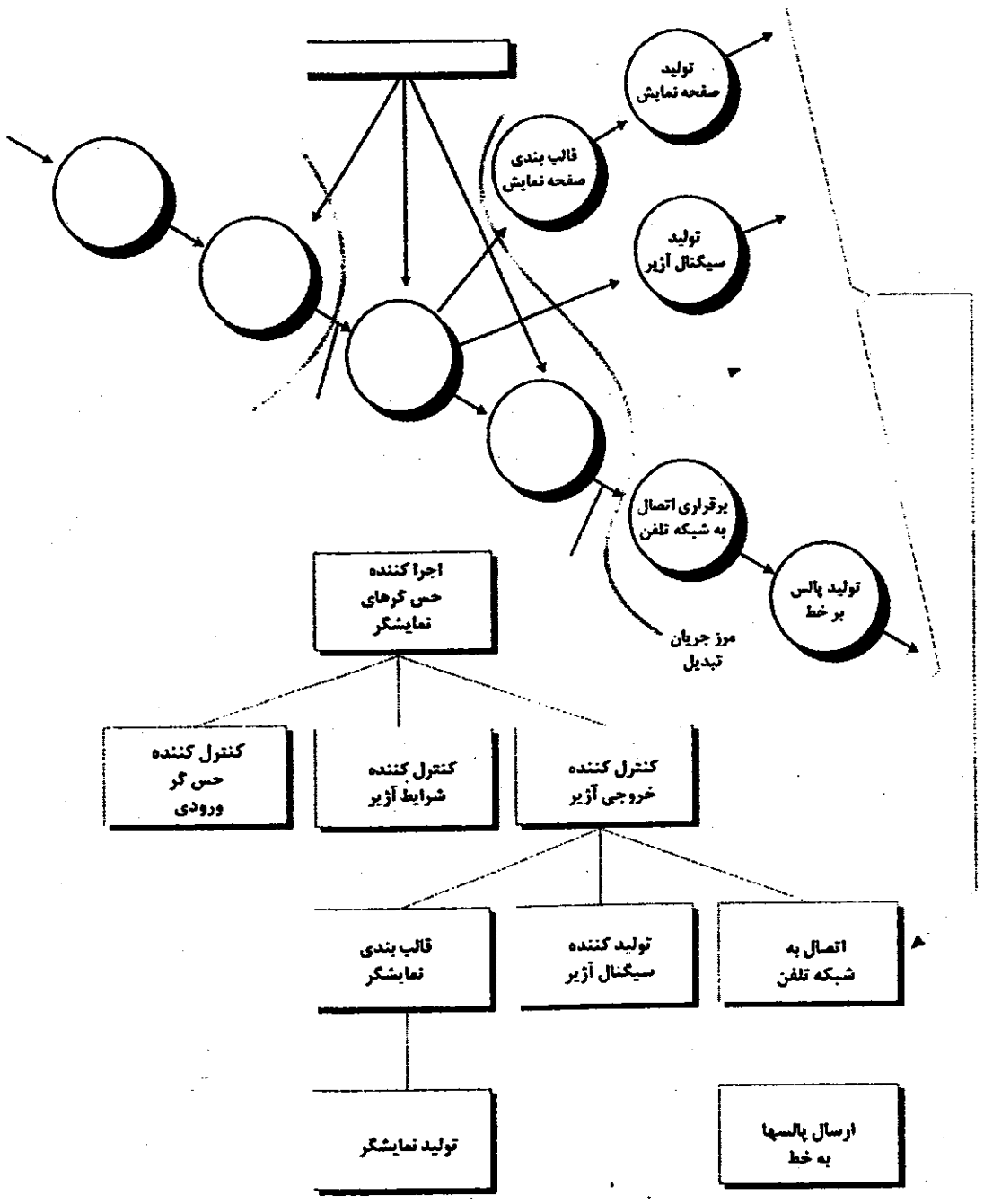
- اطلاعاتی که به‌پیمانه وارد و از آن خارج می‌شوند. (توصیف رابط)
- اطلاعاتی که توسط پیمانه حفظ می‌شود مثل داده‌های ذخیره شده در سطح داده‌های محلی.
- یک توضیح رویه‌ای که نشان‌گر نقاط تصمیم‌گیری عمده و وظائف اصلی باشد.
- مبحث خلاصه‌ای از محدودیت‌ها و مشخصه‌های ویژه (مثل فایل I/O مشخصه‌های وابستگی سخت‌افزار و نیازمندیهای خاص زمانی).

این گزارش به‌عنوان اولین نسل مشخصه‌های طراحی[1] عمل می‌کند. پالایش و اضافه‌سازی‌های بیشتر مرتباً در طول این دوره از طراحی رخ می‌دهد.

مـرحله ۷. اصـلاح معماری اولین تکرار با استفاده از روش اکتشافی در طراحی برای بهبود کیفیت نرم‌افزار

معماری اولین تکرار، همیشه می‌تواند با به‌کارگیری مفاهیم استقلال پیمانه‌ها، مورد اصلاح قرار گیرد. پیمانه‌ها را می‌توان به‌صورت درونی یا برونی باز نمود تا تولید درست، انسجامی خوب، پیوند حداقل بیانجامد و مهم‌تر از همه ساختاری ایجاد شود که بتواند بدون مشکل عمل کرده، بدون اشتباه آزمون شده و بدون ناراحتی نگهداری گردد.

1.Design Specification



شکل ۱۴ - ۱۰ فاکتور بندی سطح دوم برای حس گرهای نظارتی

اصلاحات توسط تحلیل و روشهای ارزیابی تعیین می شوند که به طور خلاصه در بخش ۴-۱۴ آمده همچنین در این قسمت ملاحظات کلی و راههای درست عملی ذکر شده اند. مواردی وجود دارد که در آن مثلاً کنترل کننده برای جریان ورودی داده کاملاً غیرضروری است یا پردازش ورودی در یک پیمانه ای لازم می آیدکه نسبت به کنترل کننده تغییر, در حالت فرعی قرار دارد. یا نمی توان به خاطر داده های فراگیر از

پیوند و اتصال قوی پرهیز نمود. اما چیزی که حرف آخر را می‌زند شرایط نرم‌افزار همراه با صلاحدید انسان است.

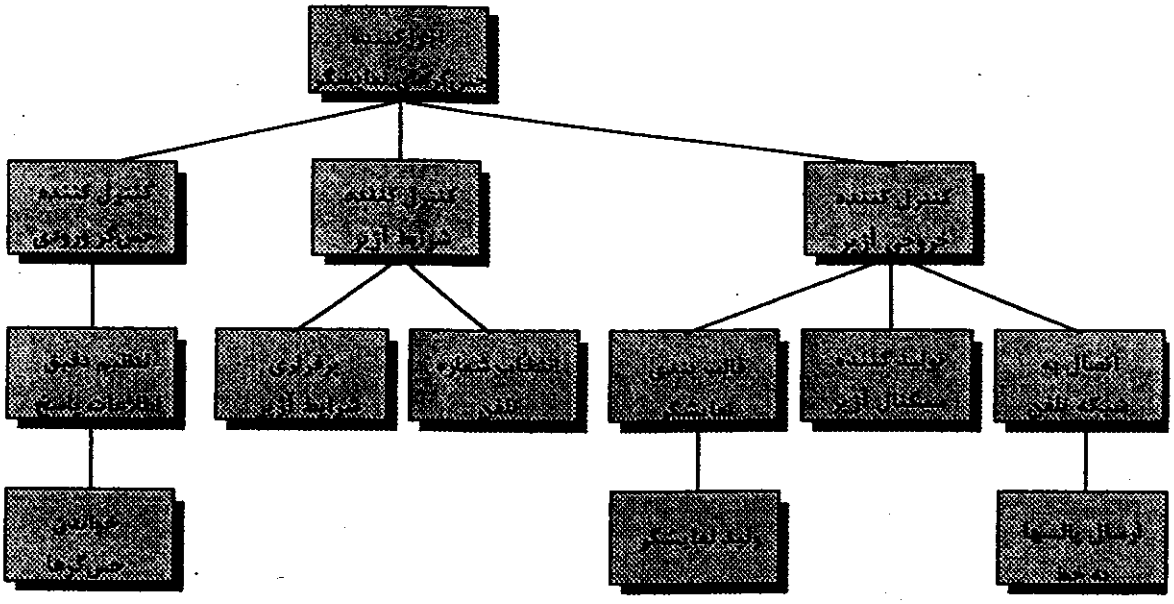
در سیستم فرعی *سنسورهای ناظر خانهٔ امن* می‌توان اصلاحات زیادی را در معماری اولین تکرار به‌وجود آورد. از میان این احتمالات می‌توان به موارد زیر اشاره نمود:

۱- می‌توان کنترل‌کننده ورودی را برداشت زیرا وقتی یک مسیر جریان ورودی قرار است درست شود این کنترل‌کننده ضروری نیست.

۲- می‌توان ساختمان فرعی تولید شده از جریان تبدیل را در پیمانه ای حل نمود که *شرایط هشداردهنده* را به‌وجود می‌آورد (که اکنون شامل فرآیند مورد اشاره با انتخاب شماره تلفن است). این کنترل‌کننده تبدیل ضروری نیست و افت کوچکی در میزان یکپارچگی اشکال ندارد.

۳- پیمانه‌های *نمایش قالب* و *نمایش ایجاد* را می‌توان برداشت و در یک پیمانه تازه به نام پیمانه *نمایش تولید* (Produce display) قرار داد.

ساختار اصلاح شده نرم افزار در مورد سیستم فرعی سنسورهای مونیتور در شکل ۱۴-۱۲ آمده است. هدف از این هفت مرحله ذکر شده، توسعه یک بازنمایی معماری نرم‌افزار می‌باشد. یعنی وقتی ساختار مشخص شد می‌توانیم با بازبینی کلی آن معماری نرم‌افزار را ارزیابی و اصلاح کنیم. تغییرات صورت گرفته در این مرحله نیازمند کار بیشتری است و این می‌تواند تأثیر زیادی روی کیفیت نرم‌افزار داشته باشد.



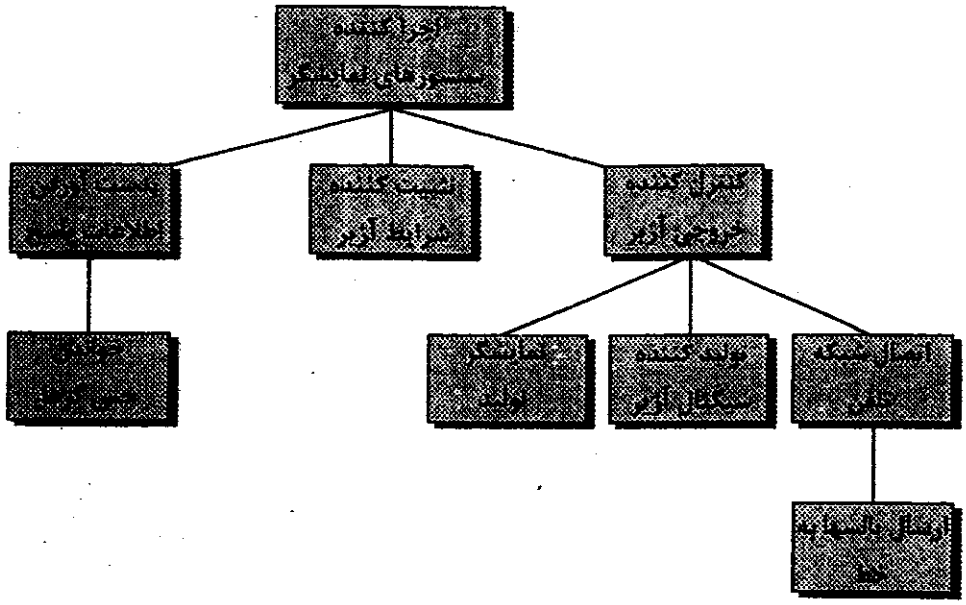
شکل ۱۴ـ ۱۱ " نمودار نخست " ساختار برنامه برای حس گرهای نمایش دهنده

خواننده باید لحظه‌ای تأمل نموده و تفاوت بین روش طراحی ذکر شده و فرآیند نوشتن برنامه را از نظر بگیرد. اگر کد تنها نشان‌گر نرم‌افزار باشد، تولیدکننده در ارزیابی و اصلاح نرم‌افزار در سطح کلی یا اکتشافی دچار مشکل خواهد شد.

۱۴-۷ نگاشت تراکنش ها

در بسیاری از برنامه‌های کاربردی نرم افزاری یک قلم داده‌ای به تنهایی یک یا چند جریان اطلاعاتی را که به کارکرد موردنظر راه‌اندازی داده‌ها اثر می‌گذارد، آغاز می‌کند. قلم داده را که تراکنش[1] نام دارد و مشخصه های جریان مربوطه در بخش ۱۴-۵-۲ مورد بحث قرار گرفتند. در این بخش مراحل طراحی به‌کار رفته برای جریان تراکنش را بررسی می‌کنیم.

1.transaction



شکل ۱۴ـ۱۲ ساختار پالایش شده برنامه برای حس گرها ی نمایش دهنده ده

۱۴-۷-۱ یک مثال

نگاشت تراکنش با در نظر گرفتن سیستم فرعی رابط کاربر[1] در نرم‌افزار خانهٔ امن تشریح می‌گردد. جریان داده‌ای سطح ۱ برای این سیستم به‌عنوان بخشی از شکل۱۴-۶ نشان داده شده است. اصلاح جریان، یعنی نمودار جریان داده‌ای سطح ۲ (فرهنگنامه داده‌ای مربوطه، PSPEC, CSPECها نیز ایجاد شده‌اند) ارائه و در شکل ۱۴-۱۳ نشان شده است.

همان‌گونه که در شکل آمده فرمان‌های کاربر به سیستم وارد می‌شود و نتایج اطلاعاتی اضافی، در طول یکی از سه مسیر اقدام، جریان می‌یابد. یک قلم داده‌ای منفرد از نوع فرمان باعث می‌شود که جریان داده‌ای از یک مرکز به طرف خارج جریان یابد. بنابراین، مشخصه کلی جریان داده‌ای بر پایه تراکنش است.

باید توجه داشت که جریان اطلاعاتی در طول دو تا سه مسیر با جریان ورودی اضافی انطباق می‌یابد (مثلاً پارامترهای سیستم و داده‌ها در مسیر اقدام "پیکربندی" وارد می‌شوند). هر مسیر اقدام در یک تبدیل منفرد جریان می‌یابد، نمایش وضعیت و پیام‌ها[2].

۱۴-۷-۲ گامهای طراحی

مراحل طراحی در مورد نگاشت تبدیل، مشابه و در بعضی موارد عیناً مثل مراحل نگاشت تبدیل می‌باشد. (بخش ۱۴-۶) تفاوت عمده در طراحی DFD در ساختار نرم‌افزار نهفته است.

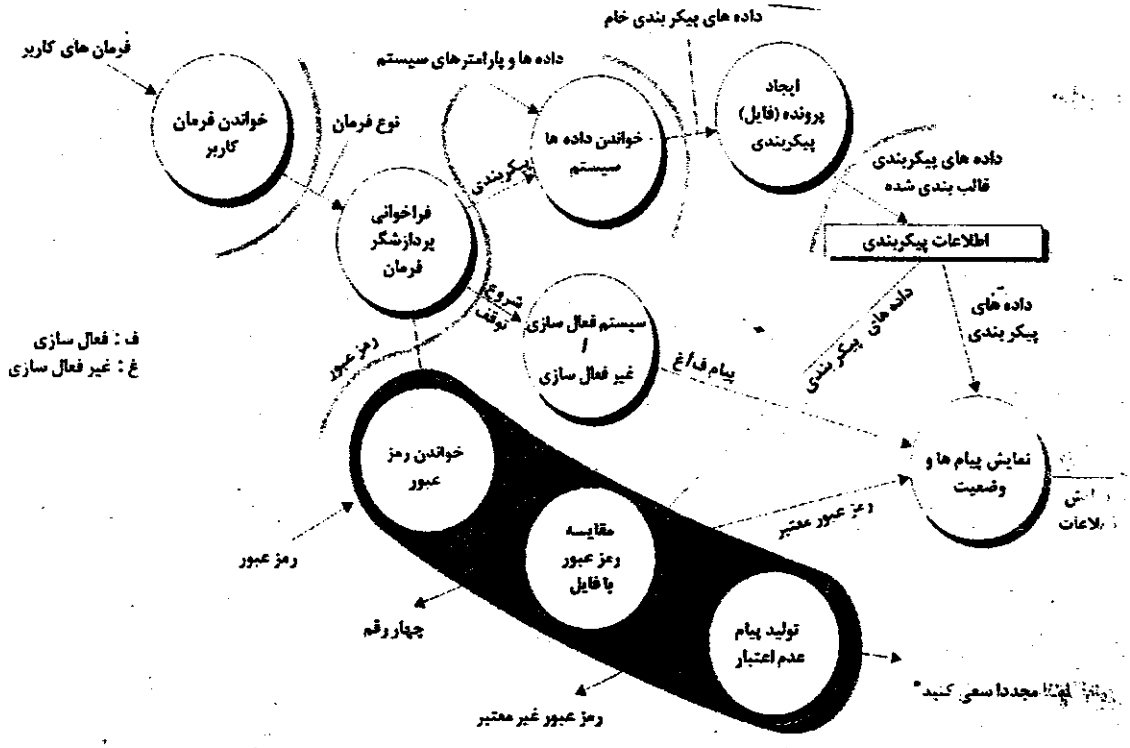
1.user interaction

2.display messages and status

مرحله ۱. بازنگری مدل سیستم بنیادی .

مرحله۲. بازنگری و اصلاح نمودارهای جریان داده‌ای نرم‌افزار .

مرحله۳. تعیین این‌که آیا DFD دارای مشخصه‌های جریان تبدیل یا تراکنش است. مراحل ۱، ۲ و ۳ مشابه مراحل مربوطه در نگاشت تبدیل، هستند. DFD نشان داده شده در (شکل۱۴-۱۳) دارای مشخصه کلاسیک جریان تراکنش است. جریانی که در طول دو مسیر از مسیرها، متوسط حباب فراخوانی پردازشگر فرمان، ظاهر می‌شود، دارای مشخصه‌های جریان تبدیل است. بنابراین، باید سرحدات جریان برای دو نوع جریان ایجاد گردد.



شکل ۱۴ - ۱۳

DFD سطح ۲ برای زیر سیستم محاوره کاربر با مرزهای جریان

مرحله ۴. شناسایی مرکز تراکنش و مشخصات جریان در طول هر یک از مسیرهای اقدام

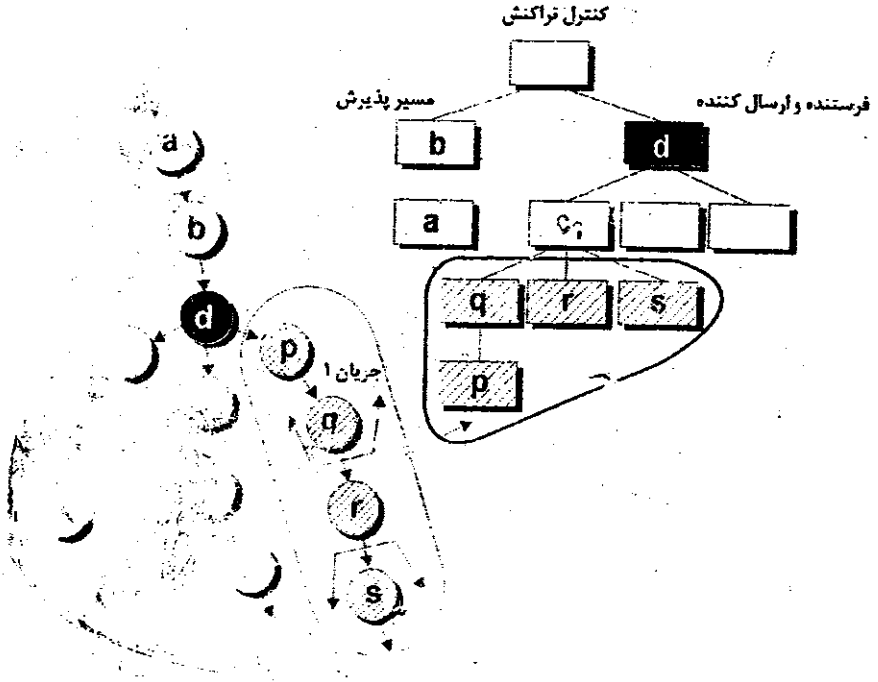
محل مرکز تراکنش را می‌توان فوراً از DFD تشخیص داد. مرکز تراکنش در محل مرکزی تعدادی از مسیرهای اقدام قرار می‌گیرد که به‌صورت شعاعی از آن جریان می‌یابند. در مورد جریان نشان داده شده در شکل ۱۴-۱۳ جباب فراخوانی پردازشگر فرمان[1]، مرکز تراکنش است.

مسیر ورودی (یعنی مسیر جریان که در طول آن یک تراکنش دریافت می‌گردد) و همه مسیرهای اقدام نیز باید جدا گردند. سرحدانی که مسیر پذیرش و مسیرهای اقدام را تعیین می‌کنند نیز در شکل نشان داده شده‌اند. هر مسیر عمل باید از نظر مشخصات جریان به تنهایی ارزیابی گردد. مثلاً، مسیر "کلمه عبور" (که با محوطه هاشور زده در شکل ۱۴-۱۳ نشان داده شده) دارای مشخصات تبدیل است. جریان ورودی، تغییر و خروجی با سرحدات آنها بیان شده است.

نتایج فاکتورگیری سطح نخست هنگام ایجاد سلسله مراتب کنترلی نرم افزار، مشهود می شود. فاکتور گیری سطح دوم، گسترش " پیمانه های کارگر " تحت کنترل های مقتضی است.

مرحله ۵. نگاشت DFD در ساختار برنامه‌ای که در پردازش تراکنش‌ها قابل بررسی است.

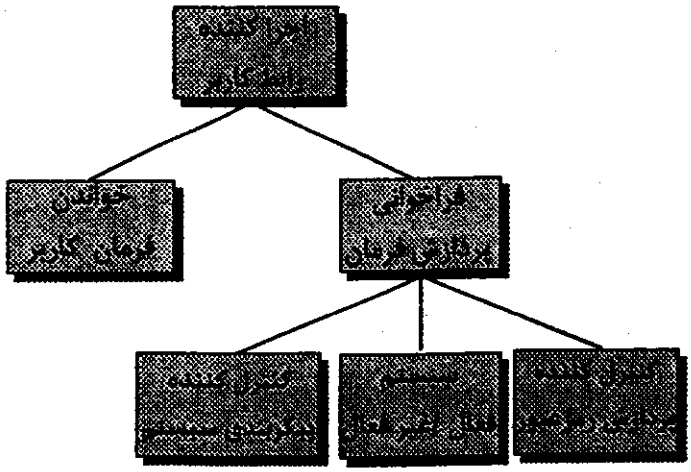
جریان تراکنش‌ها به یک معماری نگاشت شده که حاوی یک شاخه ورودی و یک شاخه خروجی یا ارسال است. ساختمان شاخه ورودی تا حد زیادی شبیه نگاشت تبدیلات، ارائه شده است. دوایری که در طول مسیر ورودی هستند و از مرکز تراکنش شروع می‌شوند و به پیمانه‌ها نگاشت شده‌اند. ساختمان شاخه خروج دارای یک پیمانه ارسال‌کننده است که تمام پیمانه‌های فرعی عمل را کنترل می‌کند. هر مسیر جریان عمل در DFD با ساختاری نگاشت می‌شود که با مشخصه‌های جریان به‌خصوص هم‌خوانی داشته باشد. این فرآیند به‌صورت شماتیک در شکل ۱۴-۱۴ آمده است.



شکل ۱۴ - ۱۴ نگاشت تراکنش

1. invoke command processing

با توجه به جریان داده‌های سیستم فرعی رابط کاربر، ساخت اولین سطح در مورد مرحله ۵ در شکل ۱۴-۱۵ آمده است. دایره‌هایی که در آن "خواندن فرمان کاربر و سیستم فعال/ غیرفعال" آمده مستقیماً در معماری بدون نیاز به پیمانه‌های کنترل فوری نسبت می‌شوند. مرکز اجرا یعنی "فراخوانی پردازشگر فرمان" مستقیماً در پیمانهٔ ارسال‌کننده هم نام، نگاشت می‌شود. کنترل‌کننده‌های پیکربندی سیستم و پردازشگر کلمه عبور، همان‌گونه که در شکل ۱۴-۱۴ آمده‌اند، ایجاد می‌شوند.

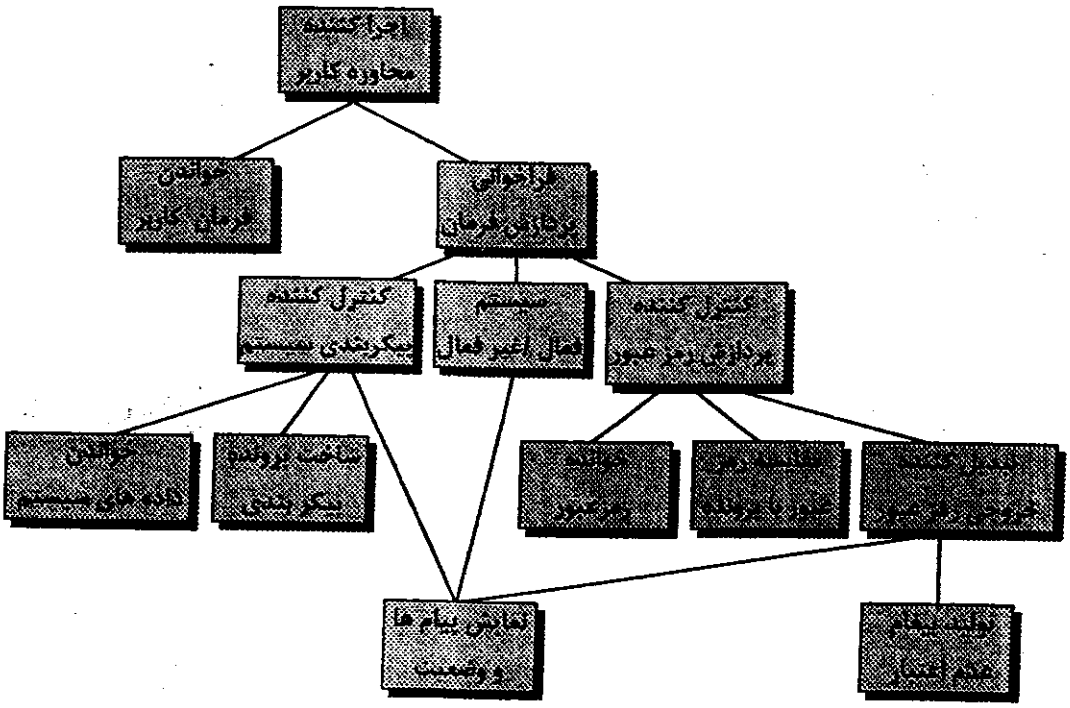


شکل ۱۴ـ۱۵ عوامل سطح نخست برای زیر سیستم رابط کاربر

مرحله ۶. فاکتور و اصلاح ساختمان تراکنش و ساختار هر مسیر اقدام.

هر مسیر اقدام در نمودار جریان داده‌ها دارای مشخصات جریان اطلاعاتی خود می‌باشد. هم‌اکنون توجه داشتید که جریان تراکنش یا تبدیل ممکن است صورت پذیرد. مسیر مربوط به ساختار فرعی با استفاده از مراحل طراحی مورد بحث در این بخش و بخش قبلی ارائه می‌شود.

به‌عنوان مثال جریان اطلاعات پردازش کلمه عبوری را در نظر بگیرید که در شکل ۱۴-۱۳ هاشورخورده است. این جریان نشان‌گر مشخصات تبدیل کلاسیک است. کلمه عبوری وارد شده (جریان ورودی) و به مرکز تبدیل می‌رود جایی‌که در آن در مقایسه با سایر کلمات عبور ذخیره شده، قرار می‌گیرد. یک پیغام هشداردهنده و اخطار (جریان خروجی) تولید می‌شود (اگر تطابقی یافت نشود). مسیر پیکربندی یکسان با استفاده از نگاشت تبدیل رسم می‌شود. معماری نرم‌افزار به‌دست آمده در شکل ۱۴-۱۶ نشان داده شده است.



شکل ۱۴ـ۱۶ معماری تکرار ـ اول برای زیر سیستم رابط کاربر

مـرحله ۷. اصـلاح معمـاری اولیـن تکرار با استفاده از روش اکتشافی طراحی برای بهبود کیفیت نرم‌افزار.

این مرحله برای نگاشت تراکنش مشابه با مرحله مربوطه برای نگاشت تبدیل است. در هر دو رهیافت طراحی، معیارهایی مثل استقلال پیمانه، قابلیت عمل (کارآیی پیاده‌سازی و آزمون) و نگهداری باید به دقت مدنظر قرار گیرند همان‌گونه که اصلاحات ساختاری اعمال می‌شوند.

۱۴-۸ پالایش طراحی معماری

پس از ایجاد معماری چه روی می دهد؟

استفاده مؤفق از نگاشت تراکنش یا تبدیل با کارهای اضافی دیگری تکمیل می‌شود که به‌عنوان بخشی از طراحی معماری ضروری هستند. بعد از ساخت ساختار برنامه و اصلاح آن، کارهای زیر باید تکمیل گردد:

- گزارش پردازشی برای هر پیمانه ارائه شود.
- یک توضیح در مورد رابط برای هر پیمانه داده شود.
- ساختارهای داده‌ای موضعی و کلی تعریف شوند.
- تمام محدودیت‌های طراحی ذکر شوند.
- مجموعه‌ای از بازنگری طراحی‌ها آماده شود.
- اصلاح در صورت لزوم درنظر گرفته شود.

گزارش پردازشی یک توصیف ساده و محدود از پردازش صورت گرفته در پیمانه است. این گزارش کارهای پردازش، تصمیم‌گیری و I/O را تشریح می‌کند. توصیف رابط طراحی رابط‌های پیمانه داخلی،

رابطهای سیستم خارجی و رابط کاربر ـ کامپیوتر را تشریح می‌کند. طراحی ساختارهای داده‌ای می‌تواند تأثیر شگرفی بر معماری و جزییات رویهٔ هر جزء نرم‌افزاری داشته باشد. محدودیت‌های هر پیمانه نیز ثبت می شوند. موضوعات معمول بحث شامل: محدودیت در ساختار داده‌ها، قالب بندی نوع داده ها، محدودیت حافظه و زمان، تخصیص مقادیر یا ساختار داده های کمی که البته موارد خاص در نظر گرفته نشده و مشخصات خاص یک پیمانه مستقل منظور نظر می‌باشد. هدف از بخش محدودیت‌ها عبارتست از کاهش تعداد خطاهای معرفی شده دلیل مشخصات کارکردی فرض شده.

مشخصات طراحی نرم افزار

وقتی مدارک و اسناد طراحی برای همه پیمانه‌ها ارائه شد، یک یا چند بازبینی در طراحی صورت می‌گیرد. این بازنگری بر قابلیت پی‌گیری نیازمندیها، کیفیت معماری نرم‌افزار، توصیف رابطها، توصیف ساختار داده‌ها، پیاده‌سازی و آزمون پذیری و قابلیت نگهداری تأکید دارد.

پالایش معماری نرم افزار در همان مراحل اولیه طراحی باید مورد توجه قرار گیرد. همانطور که پیشتر در این فصل توضیح دادیم، سبک های معماری جایگزین، می تواند به دست آید، پالایش شود، و به عنوان بهترین رهیافت ارزیابی شود. این رهیافت بهینه سازی یکی از منافع حقیقی بدست آمده از ایجاد بازنمایی معماری نرم افزار است.

نکته قابل توجه، سادگی ساختار است که اغلب منعکس‌کننده ظرافت و کارآیی است. اصلاح‌کننده طراحی باید تلاش کند به کمترین مقدار پیمانه که با پیمانه سازی مؤثر هماهنگی داشته و کمترین پیچیدگی ساختاری که به حد کافی در خدمت شرایط اطلاعاتی است، برسد.

۹-۱۴ خلاصه

معماری نرم‌افزار دیدگاه موشکافانه‌ای از سیستمی که قرار است ساخته شود ارائه می‌دهد. در آن ساختار و سازمان اجزاء نرم‌افزاری، خواصشان و ارتباط بین آنها مشخص می‌گردد. اجزاء نرم‌افزاری شامل پیمانه‌های برنامه و نمایش مختلف داده‌هایی است که توسط برنامه تغییر یافته‌اند. بنابراین، طراحی داده‌ها بخش سازنده اشتقاق معماری نرم افزار است. معماری تصمیمات اولیه طراحی را مشخص کرده و مکانیزمی برای بررسی مزایای ساختارهای جایگزین سیستم مهیا می‌کند.

طراحی داده‌ها، اشیای داده‌ای تعریف شده در مدل تحلیل را به ساختارهای داده‌ای، برمی‌گرداند که در داخل نرم افزار قرار دارند. نگرش‌های توصیف یک شی، ارتباط بین اشیاء داده‌ای و کاربردشان در برنامه، همگی بر انتخاب ساختارهای داده‌ای تأثیر می‌گذارد. در سطح انتزاعی بالاتر، طراحی داده‌ها ممکن است منجر به تعریف یک معماری برای پایگاه داده‌ای یا مخزن داده‌ها شود.

چند سبک مختلف معماری و الگوهای مختلف برای مهندس نرم‌افزار مهیاست. هر سبک یک طبقه از سیستم را توصیف می‌کند که در برگیرنده مجموعه‌ای از جزء هاست که یک عملی لازم برای سیستم را انجام می‌دهد، همچنین مجموعه‌ای از اتصال‌دهندگان که ارتباط را برقرار می‌سازند، هماهنگی و همکاری

میان جزء ها، محدودیتهایی که چگونگی یکپارچه‌سازی جزء ها را برای تشکیل سیستم تعریف می‌کنند و مدل‌های معنایی که طراح را قادر به درک خواص کلی سیستم می‌سازند.

وقتی یک یا چند سبک معماری برای سیستم پیشنهاد شد، ممکن است از روش تحلیل توازن معماری برای ارزیابی کارآیی معماری پیشنهادی استفاده شود. این کار با تعیین میزان حساسیت روش‌های کیفی انتخابی در مکانیزم‌های مختلف شناسایی که نمایان‌گر خواص معماری هستند، صورت می‌گیرد.

روش طراحی معماری که در این بخش استفاده شده از مشخصات جریان داده‌ای توصیف شده در مدل تحلیل برای رسیدن به یک سبک متداول معماری استفاده می‌کند. در ساختمان برنامه، یک نمودار جریان داده‌ای با استفاده از یکی از دو رهیافت نگاشت یعنی نگاشت تبدیل / نگاشت تراکنشی، ترسیم می‌شود. نگاشت تبدیل در جریان اطلاعات به‌کار گرفته می‌شود که سرحدات مشخصی را بین داده‌های ورودی و خروجی مشخص می‌کند. DFD در ساختاری طرح می‌شود که کنترل را در ورودی، پردازش و خروجی در طول سه پیمانه جداگانه ساخته شده، مطرح می‌کند. طراحی تراکنش وقتی به‌کار گرفته می‌شود که یک قلم اطلاعاتی باعث جریان در شاخه در طول یکی از چندین مسیر می‌شود. DFD به ساختاری نگاشت می‌شود که کنترل را به ساختار فرعی اختصاص می‌دهد، ساختاری که تغییر را در دست گرفته و ارزیابی می‌کند. یک ساختار فرعی دیگر تمام کارهای پردازشی بالقوه را بر اساس تغییر کنترل می‌کند. وقتی یک معماری مشتق گردید، بسط یافته و سپس در مقابل معیارهای کیفی تحلیل می‌گردد.

طراحی معماری در برگیرنده مجموعه اولیه فعالیتهای طراحی است که منجر به مدل کامل طراحی نرم‌افزار می‌شود. در فصول بعدی، طراحی بر تغییرات رابط و جزء ها متمرکز می‌شود.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۱۴-۱ معماری یک خانه یا ساختمان را به طور استعاره در نظر گیرید، قیاسی با معماری نرم افزار داشته باشید. چه نظامهای مشابهی با نرم افزار وجود دارد؟ در چه چیز متفاوتند؟

۱۴-۲ مقاله‌ای در سه تا پنج صفحه بنویسید که رهنمودهایی برای انتخاب ساختمان داده‌ها براساس طبیعت ارائه کند. کار را با توضیح ساختمان داده‌های کلاسیک در کار نرم‌افزاری آغاز کرده، آنگاه معیارهای مربوط به انتخاب برای انواع خاص از مسائل را شرح دهید.

۱۴-۳ اختلاف میان پایگاه داده هایی که به یک یا چند کاربرد تجاری متعارف خدمات می‌دهد و یک مخزن داده‌ها را شرح دهید.

۱۴-۴ مقاله‌ای در سه تا پنج صفحه بنویسید که شرح دهد روش‌های کاوش داده‌ها چگونه در بافت تجاری و وضعیت فعلی تکنیک‌های KDD به‌کار می روند؟

۱۴-۵ دو یا سه مثال از کاربردهای هر یک از سبک‌های معماری مورد اشاره در بخش ۱۴-۳-۱ بیاورید.

۱۴-۶ برخی سبک‌های معماری مورد اشاره در ۱۴-۳-۱ از طبیعتی سلسله مراتبی برخوردارند و برخی دیگر اینگونه نیستند. از هر دو نوع لیستی تهیه کنید. سبک‌های معماری که سلسله مراتبی نیستند، چگونه پیاده‌سازی خواهند شد؟

۱۴-۷ کاربردی را انتخاب کنید که برای شما آشناست. به هر یک از پرسش‌های مربوط به کنترل و داده در بخش ۱۴-۳-۲ پاسخ گوئید.

۱۴-۸ ATAM (با توجه به [KAZ98]) را تحقیق کنید. از شش مرحله ارائه شده در بخش ۱۴-۴-۱ را مفصلاً بحث کنید.

۱۴-۹ کاربردی را انتخاب کنید که برای شما آشناست. با استفاده از بهترین حدس‌های مورد نیاز، مجموعه ای از ابعاد طراحی را معرفی کنید و سپس تحلیل طیفی و تحلیل انتخاب طراحی را انجام دهید.

۱۴-۱۰ QDS (با استفاده از [ASA96]) را تحقیق کنید و یک فضای طراحی کمّی برای کاربردی که برای شما آشناست، توسعه دهید.

۱۴-۱۱ برخی طراحان مدعی هستند که تمام جریان داده ها تبدیل‌گرا می باشند. بحث کنید که این ادعا چگونه بر معماری تاثیرگذار خواهد بود مخصوصاً هنگامی که یک جریان تراکنش گرا به‌عنوان یک تبدیل منظور شود. از یک جریان نمونه برای نشان دادن نکات مهم استفاده کنید.

۱۴-۱۲ اگر مسئله ۱۲-۱۳ را حل نکرده‌اید، آن را کامل کنید. از شیوه های طراحی شرح داده شده در این فصل، جهت توسعه یک معماری نرم‌افزار برای PHTRS استفاده نمایید.

۱۴-۱۳ یک نمودار جریان داده‌ها و یک روایت پردازشی را به کار برید، سیستمی کامپیوتری را با ویژگیهای متمایز جریان تبدیلی تشریح کنید. مرزهای جریان را تعریف کنید و با استفاده از تکنیک شرح داده شده در بخش ۶-۱۴، نگاشتی از نمودار جریان داده ها بر ساختار نرم‌افزار ایجاد نمایید.

۱۴-۱۴ یک نمودار جریان داده‌ها و یک روایت پردازشی را به کار برید، سیستمی کامپیوتری را با ویژگیهای متمایز جریان تراکنشی تشریح کنید. مرزهای جریان را تعریف کنید و با استفاده از تکنیک شرح داده شده در بخش ۷-۱۴، نگاشتی از نمودار جریان داده ها بر ساختار نرم‌افزار ایجاد نمایید.

۱۴-۱۵ با استفاده از نیازمندیهایی که حاصل یک بحث کلاسی است، نمودار جریان داده ها و طراحی معماری مثال خانه امن را که در بخش‌های ۶-۱۴ و ۷-۱۴ ارائه شده کامل کنید. استقلال عملیاتی تمام پیمانه‌ها را مورد ارزیابی قرار دهید. طراحی خود را مستند کنید.

۱۴-۱۶ شایستگی ها و دشواریهای نسبی استفاده از طراحی مبتنی بر جریان داده‌ها را در حوزه های زیر تشریح کنید: الف) کاربردهای ریزپردازنده‌های تعبیه شده(درون نهاده)، ب) تحلیل مهندسی/ علمی، پ) گرافیک کامپیوتری، ت) طراحی سیستم عامل، ث) کاربردهای تجاری، ج) طراحی سیستم مدیریت پایگاه داده ها، چ) طراحی نرم‌افزارهای ارتباطاتی، ح) طراحی کامپایلر، خ) کاربردهای کنترل فرآیند، و د) کاربردهای هوش مصنوعی.

۱۴-۱۷ مجموعه ای از نیازمندیها را از استادتان دریافت دارید (یا مجموعه ای از نیازمندیها که شما در کارهای جاریتان با آن روبرو هستید) یک طراحی معماری کامل را توسعه دهید. یک مرور بر طراحی داشته باشید (فصل ۸) تا طراحی خود را ارزیابی نمایید. شاید تخصیص این مسئله به یک تیم بهتر از انجام فردی آن باشد.

فهرست منابع و مراجع

[AH083] Aho, A.V., J. Hopcroft, and J. Ullmann, *Data Structures and Algorithms*, Addison-Wesley, 1983.

[ASA96] Asada, T., et a\., "The Quantified Design Space," in *Software Architecture* (Shaw, M. and D. Garlan), Prentice-Hall, 1996, pp. 116-127.

[BAS98] Bass, L., P. Clements, and R. Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.

[BUS96] Buschmann, F., *Pattern-Oriented Software Architecture*, Wiley, 1996.

[DAH72] Dah., 0., E. Dijkstra, and C. Hoare, *Structured Programming*, Academic Press, 1972.

[DAT95] Date, C.J., *An Introduction to Database Systems*, 6th ed., Addison-Wesley, 1995.

[DEN73] Dennis, J.B., "Modularity," in *Advanced Course on Software Engineering* (F.L. Bauer, ed.), Springer-Verlag, 1973, pp. 128-182.

[FRE80] Freeman, P., "The Context of Design," in *Software Design Techniques*, 3rd ed. (P. Freeman and A. Wasserman, eds.), IEEE Computer Society Press, 1980, pp. 2-4.

[INM95] Inmon, W.H., "What Is a Data Warehouse?" Prism Solutions, 1995, presented at http://www.cait.wustl.edu/cait/papers/prism/vol1_no1.

[KAZ98] Kazman, R. et a\., *The Architectural Tradeoff Analysis Method*, Software Engineering Institute, CMU/SEI-98- TR-008, July 1998.

[KIM98] Kimball, R., L. Reeves, M. Ross, and W. Thornthwaite, *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*, Wiley, 1998.

[LIN79] Linger, R.C., H.D. Mills, and B.I. Witt, *Structured Programming*, Addison-Wesley, 1979.

[MAT96] Mattison, R., *Data Warehousing: Strategies, Technologies and Techniques*, McGraw-Hill, 1996.

[MYE78] Myers, G., *Composite Structured Design*, Van Nostrand, 1978.

[PRE98] Preiss, B.R., *Data Structures and Algorithms: With Object-Oriented Design Patterns in C++*, Wiley, 1998.

[SHA96] Shaw, M. and D. Garlan, *Software Architecture*, Prentice-Hall, 1996.

[SHA97] Shaw, M. and P. Clements, "A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems," *Proc. COMPSAC*, Washington, DC, August 1997.

[STE74] Stevens, w., G. Myers, and L. Constantine, "Structured Design," *IBM System journal*, vol.13, no. 2,1974, pp.II5-139.

[WAS80] Wasserman, A., "Principles of Systematic Data Design and Implementation," in *Software Design Techniques* (P. Freeman and A. Wasserman, ed.), 3rd ed., IEEE Computer Society Press, 1980, pp. 287-293.

[WIR7I] Wirth, N., "Program Development by Stepwise Refinement," *CACM*, vol. 14, no.4,197I,pp.221-227.

[YOU79] Yourdon, E. and L. Constantine, *Structured Design*, Prentice-Hall, 1979.

[ZHA98] Zhao,J, "On Assessing the Complexity of Software Architectures," *Proc. Intl. Software Architecture Workshop*, ACM, Orlando, FL, 1998, p. 163-167.

خواندنیهای دیگر و منابع اطلاعاتی

The literature on software architecture has exploded over the past decade. Books by Shaw and Garlan [SHA96], Bass, Clements, and Kazman [BAS98] and Buschmann et al. [BUS96] provide in-depth treatment of the subject. Earlier work by Garlan (An *Introduction to Software Architecture*, Software Engineering Institute, CMU/SEI-94-TR-021, 1994) provides an excellent introduction.

Implementation specific books on architecture address architectural design within a specific development environment or technology. Mowbray *(CORBA Design Patterns*, Wiley, 1997) and Mark et al. *(Object Management Architecture Guide*, Wiley, 1996) provide detailed design guidelines for the CORBA distributed application support framework. Shanley *(Protected Mode Software Architecture*, Addison-Wesley, 1996) provides architectural design guidance for anyone designing PC-based real-time operating systems, multi-task operating systems, or device drivers.

Current software architecture research is documented yearly in the *Proceedings of the International Workshop on Software Architecture*, sponsored by the ACM and other computing organizations, and the *Proceedings of the International Conference on Software Engineering*.

Data modeling is a prerequisite to good data design. Books by Teory *(Database Modeling and Design*, Academic Press. 1998); Schmidt *(Data Modelingfor Information Professionals*, Prentice-Hall, 1998); Bobak *(Data Modeling and Design for Today's Architectures*, Artech House, 1997); Silverston, Graziano, and Inmon *(The Data Model Resource Book*, Wiley, 1997); Date [DAT95], Reingruber and Gregory *(The Data Modeling Handbook: A Best-Practice Approach to Building Quality Data Models*, Wiley, 1994); and Hay *(Data Model Patterns: Conventions of Thought*, Dorset House, 1994) contain detailed presentations of data modeling notation, heuristics, and database design approaches. The design of data warehouses has become increasingly important in recent years. Books by Humphreys, Hawkins, and Dy *(Data Warehousing: Architecture and Implementation*, Prentice-Hall, 1999); Kimball et al. [KIM98]; and Inmon [INM95] cover the topic in considerable detail.

Dozens of current books address data design and data structures, usually in the context of a specific programming language. Typical examples are

Horowitz, E. and S. Sahni, *Fundamentals of Data Structures in Pascal*, 4th ed., W.H. Freeman and Co., 1999.

Kingston, J.H., *Algorithms and Data Structures: Design, Correctness, Analysis*, 2nd ed., Addison-Wesley, 1997.

Main, M., *Data Structures and Other Objects Using java*, Addison-Wesley, 1998.

Preiss, B.R, *Data Structures and Algorithms: With Object-Oriented Design Patterns in* C++, Wiley, 1998.

Sedgewick, R., *Algorithms in* C++: *Fundamentals, Data Structures, Sorting, Searching*, Addison-Wesley, 1999.

Standish, T.A., *Data Structures in java*, Addison-Wesley, 1997.

Standish, T.A., *Data Structures, Algorithms, and Software Principles in* C, Addison-Wesley, 1995.

General treatment of software design with discussion of architectural and data design issues can be found in most books dedicated to software engineering. Books by Pfleeger *(Software Engineering: Theory and Practice*, Prentice-Hall, 1998) and Sommerville *(Software Engineering*, 5th ed., Addison-Wesley, 1996) are representative of those that cover design issues in some detail.

More rigorous treatments of the subject can be found in Feijs *(Formalization of Design Methods*, Prentice-Hall, 1993). Witt et al. *(Software Architecture and Design Prin-*

ciples, Thomson Publishing, 1994). and Budgen *(Software Design*, Addison-Wesley, 1994).

Complete presentations of data flow-oriented design may be found in Myers [MYE78], Yourdon and Constantine [YOU79] .Buhr *(System Design with Ada*, Prentice-Hall, 1984), and Page-Jones *(The Practical Guide* to *Structured Systems Design*, 2nd ed., Prentice-Hall, 1988). These books are dedicated to design alone and provide comprehensive tutorials in the data flow approach.

A wide variety of information sources on software design and related subjects is available on the Internet. An up-to-date list of World Wide Web references that are relevant to design concepts and methods can be found at the SEPA Web *site:* http://www.mhhe.com/engcs/compsci/pressman/resources/arch-design.mhtml