فصل ۲۷ مهندسی نرم افزار مبتنی بر اجزاء(CBSE)

مفاهیم کلیدی (مرتب بر حروف الفبا)
امتیازات ساختاری ، انواع اجزاء ، تطابق ، ترکیب ، توسعه مبتنی بر اجزاء ، توصیف رده بندی ، فرآیند مهندسی نرم افزار مبتنی بر اجزاء ، فعالیتهای مهندسی نرم افزار مبتنی بر اجزاء ، کارکردهای مشخصات ، ملاحظات اقتصادی استفاده مجدد ، مهندسی دامنه

KEY CONCEPTS
Adaptation , CBSE activities , CBSE process , characterization functions , classification , component-based development , component types , composition , domain engineering , economics of reuse , qualification , structure points

نگاه اجمالی

مهندسی نرم‌افزار مبتنی بر اجزاء چیست؟ شما یک سیستم استریو را خریداری می کنید. و به منزل می برید، هر جزء آن به گونه ای طراحی شده که با یک سبک خاص استاندارد، به راحتی به دیگر اجزاء متصل شود. پروتکل های ارتباطی پیشتر برقرار گردیده است. سوار کردن آسان است زیرا شما مجبور نیستید صدها قطعه سیستم را بسازید. مهندسی نرم افزار مبتنی بر اجزاء نیز مشابه این داستان است. برای حوزه کاری معینی تعداد اجزاء استاندارد از قبل طراحی شده ای در دسترس خواهند بود. ساخت یک برنامه کاربردی به جای آنکه از بخشهای مجزا و گسسته مانند برنامه ساری سنتی و متعارف بهره برد، از اتصال این اجزاء به یکدیگر بوجود خواهد آمد.

چه کسی عهده دار این امر می باشد؟ فرآیند CBSE توسط یک مهندس نرم‌افزار متخصص انجام خواهد شد.

چرا مهندسی نرم افزار مبتنی بر اجزاء از اهمیت برخوردار است؟ از آنرو که اجزاء یک استریو به گونه ای مجتمع و یکپارچه طراحی شده اند، بنابرین سرهم کردن آن بسیار ساده خواهد بود. ساخت نرم افزارهای پیچیده نیز اگر از این شیوه بهره برد، از هزینه کمتری برخوردار خواهد شد. CBSE مشوق استفاده از الگوهای معماری پیش ساخته و استفاده از زیرساخت های استاندارد می باشد، و از این طریق به سمت نتایجی با کیفیت بالا رهنمون خواهد بود.

مراحل کار چیست؟ CBSE مشتمل بر دو فعالیت موازی مهندسی نرم افزار است : مهندسی حوزه و توسعه مبتنی بر اجزاء. مهندسی حوزه به معنای کشف اجزای کارکردی، رفتاری و داده‌ای خاصی است که در حوزه کاربردی مشخص بتوانند مورد استفاده مجدد قرار گیرند. این اجزاء در کتابخانه های استفاده مجدد قرار داده می شوند. توسعه مبتنی بر اجزاء، نیازمندیها را از مشتری استخراج نموده سبک معماری مناسبی را انتخاب می کند که اهداف اصلی سیستم مورد ساخت را پاسخگو باشد، آنگاه ۱) اجزاء بالقوه را انتخاب می کند، ۲) بررسی می کند که اطمینان حاصل نماید اجزاء یاد شده با معماری همخوانی دارد ۳) اگر برای تجمیع اجزاء و اتصال آنها نیاز به اعمال تغییراتی وجود داشته باشد، این تطابق را به انجام می رساند، ۴) اجزاء را چنان گردآورده و تجمیع می کند که زیرسیستمها به صورت یک کل و یک سیستم جامع درآیند. به علاوه، اجزاء سفارشی ای که لازم است مهندسی شوند (جنبه هایی از سیستم که با اجزاء حاضر امکان پیاده سازی آنها وجود ندارد.) معلوم و ساخته خواهند شد.

محصول کار چیست؟ یک نرم افزار عملیاتی، که با استفاده از اجزاء از قبل آماده و جدیدا" طراحی شده، ساخته می شود.

چگونه مطمئن شوم که کارم را درست انجام داده‌ام؟ فعالیتهای تضمین کیفیتی که در دیگر فرآیندهای مهندسی نرم افزار انجام می گیرد، به طور مشابه اینجا هم مورد استفاده واقع خواهد شد _ بازبینی های فنی رسمی که مدل های تحلیل و طراحی را ارزیابی می کنند،، بازبینی های خاصی که صحت و درستی و دقت اجزاء را معلوم خواهند کرد، آزمونهایی نیز برای کشف خطاهای مربوط به اجزاء جدید و تلفیق و تجمیع اجزاء و اتصال آنها به یکدیگر تحت یک معماری خاص، انجام می گیرد.

در زمینه مهندسی نرم‌افزار، استفاده مجدد را می‌توان یک ایده جدید و در عین‌حال قدیمی پنداشت. برنامه‌سازان، ایده‌ها، مجردات و فرآیندها را از روزهای اول محاسبه، مورد استفاده مجدد قرار داده‌اند، ولی رهیافت اولیه برای استفاده مجدد امری کاملاً ویژه بود. امروزه، سیستم‌های پیچیده و دارای کیفیت بالا باید در دوره‌های زمانی بسیار کوتاه ساخته شوند. همین مسأله یک رهیافت سازمان یافته‌تر برای استفاده مجدد را تعدیل می‌نماید.

مهندسی نرم‌افزار قطعه محور یا مبتنی بر اجزاء[1] (CBSE) فرآیندی است که بر طراحی و ساخت سیستم‌های کامپیوتری با استفاده از ((اجزاء)) یا قطعات نرم‌افزاری قابل استفاده مجدد تأکید دارد. کلمنتر [CLE95][2] به طریق ذیل CBSE را توصیف کرده است:

[CBSE] روش توسعه سیستم‌های عظیم نرم‌افزاری را تغییر می‌دهد. [CBSE] فلسفه ((بخرید، خودتان آن را نسازید)) که به‌وسیله فرد- براکز و سایر دانشمندان مطرح گردیده است، دارا می‌باشد. به همان طریقی که زیر روال‌های اولیه به برنامه‌سازان امکان می‌دادند از فکر کردن درباره جزئیات فارغ باشد،

1.Component-Based Software Engineering (CBSE)
2.Clements,P.C.

CBSE تأکید را از برنامه نویسی نرم‌افزار به سیستم‌های نرم‌افزار ترکیبی برمی‌گرداند. به کارگیری آن بیوستگی را هم در برمی‌گیرد. در اصل این فرض وجود دارد که توده‌ای از سیستم‌های نرم‌افزاری عظیم وجود دارند که قطعات قابل استفاده مجدد و در حال پیشرفت را تأکید می‌کنند تا آن سیستم‌ها را مورد بهره‌برداری قرار دهند.

ولی چندین سؤال مطرح می‌گردد. آیا امکان ساختن سیستم‌های مجتمع از طریق مونتاژ آنها از روی یک کاتالوگ قطعات نرم‌افزاری قابل استفاده وجود دارد؟ آیا می‌توان به روشی مقرون به‌صرفه از لحاظ هزینه و زمان به آن هدف است یافت؟ آیا می‌توان انگیزه‌های صحیحی را برای تشویق مهندسان نرم‌افزار به‌منظور استفاده مجدد از آنها به‌جای اختراع به‌وجود آورد؟ آیا علم مدیریت قصد دارد هزینه افزوده همراه با ایجاد قطعات نرم‌افزاری قابل استفاده مجدد را در نظر گیرد؟ آیا کتابخانه قطعات که برای رسیدن به هدف ما، یعنی استفاده مجدد لازم است به‌نحوی ایجاد گردد که قابل دسترسی توسط افراد نیازمند باشد؟ آیا قطعات موجود را افرادی که به آنها نیاز دارند پیدا می‌کنند؟

این سؤال و بسیاری از پرسش‌هایی نظیر آن همواره ذهن پژوهشگران و افراد صنعتی را که سعی دارند استفاده مجدد از قطعات نرم‌افزاری را به‌صورت یک رهیافت جریان اصلی برای مهندسی نرم‌افزار درآورند، به خود مشغول کرده است. در این فصل چندین پاسخ را بررسی می‌کنیم.

۲۷-۱ مهندسی سیستم های مبتنی بر اجزاء

علی‌الظاهر، به‌نظر می‌رسد CBSE کاملاً مشابه مهندسی نرم‌افزار شئ گرا یا قراردادی باشد. این فرآیند زمانی آغاز می‌شود که یک تیم نرم‌افزاری نیازمندیهایی را برای سیستمی که باید با استفاده از فنون قراردادی ساخته شود فراهم می‌آورد (فصول ۱۰ و ۱۱) یک طراحی معماری ایجاد می‌گردد (فصل ۱۴)، ولی به‌جای حرکت به‌سوی وظائف طراحی تفصیلی، این تیم نیازمندیهایی را مورد آزمون قرار می‌دهد تا تعیین کند کدام زیرمجموعه به‌صورت مستقیم قابل کاربرد برای آن ترکیب، به‌جای ساخت است. بدین‌معنا که، آن تیم برای هر نیازمندی سیستم، سؤالات زیر را مطرح می‌سازد:

- آیا قطعات تجاری موجود[1] (COTS) برای پیاده‌سازی آن نیازمندیها وجود دارند؟
- آیا قطعاتی که در داخل توسعه یافته است برای پیاده‌سازی آن نیازمندیها وجود دارند؟

در اولین قسمت از این بخش واژه ((قطعه یا جزء)) را به‌طور مکرر به‌کار بردیم. با این حال توصیف دقیقی از این واژه ارائه نشده است. والناو و براون، احتمالات (امکانات) زیر را پیشنهاد می‌کنند:

جزء[2]. یک قطعه با اهمیت، تقریباً مستقل و قابل تعویض از یک سیستم که دارای کارکردی واضح در زمینه یک معماری خوب تعریف شده باشد.

پرسش

فعالیتهای چارچوبی مهندسی نرم افزار مبتنی بر اجزاء (CBSE) کدام هاست؟

1 Commercial Off – The – Sheff
2 component

جزء نرم‌افزاری زمان اجرا[1]. یک بسته قابل نسبت‌دهی پویا از یک یا چند برنامه‌ای که به‌عنوان یک واحد، مدیریت می‌شوند و از طریق رابط‌هایی مستند به آنها دسترسی می‌یابیم که می‌توانند در زمان احرا کشف شوند.

جزء[2]. یک واحد از ترکیب، که فقط دارای وابستگی‌های زمینه‌ای دقیق و قراردادی می‌باشد.

اجزاء تجاری[3]. کاربرد نرم‌افزار یک فرآیند تجاری یا مفهوم تجاری ((خود مختار))

علاوه بر توصیف‌های بالا، اجزاء نرم‌افزاری را می‌توان براساس کاربردشان در فرآیند CBSE مشخص نمود. علاوه بر اجزاء COTS، فرآیند CBSE موارد ذیل را به ارمغان می‌آورد:

اجزاء دارای شرایط لازم[4]. که توسط مهندسین نرم‌افزار و جهت تضمین این امر ارزیابی می‌شود که نه‌تنها از نظر کارکرد بلکه از نظر عملکرد، قابلیت اطمینان و دیگر عوامل مربوط به کیفیت بر نیازمندی‌های سیستم/ محصولی که باید ساخته شود، منطبق است.

اجزاء تطبیق داده‌شده[5]. خصوصیات نا مطلوب یا نا خواسته، اصلاح یا تطبیق و یا تعدیل شده است.

اجزاء به هم پیوسته[6]. ادغام شده در یک سبک معماری و دارای ارتباط درونی با یک زیرساخت مناسبی که امکان هماهنگی و اداره کردن اجزاء به شکل مؤثر را فراهم می‌کند.

اجزاء به هنگام شده[7] جایگزینی نرم‌افزار موجود به‌عنوان نسخه جدید اجزاء در دسترس.

چون CBSE یک نظام درحال شکل‌گیری و تحولی است این احتمال وجود ندارد که درآینده نزدیک تعریف یگانه و واحدی عرضه شود.

نقل قول

واضح و روشن است که در آینده ای نه چندان دور، اکثر نرم افزارهای کاربردی بجای برنامه نویسی از ابتدا، با به هم پیوستن اجزاء (از قبل آماده) ساخته خواهند شد. پاول

ارجاع به فصلها

تضمین و تصدیق اجزاء نرم افزاری، با استفاده از شیوه های اطاق تمیز امکان پذیر است. برای جزئیات بیشتر به فصل ۲۶ مراجعه نمایید.

۲-۲۷ فرآیند CBSE

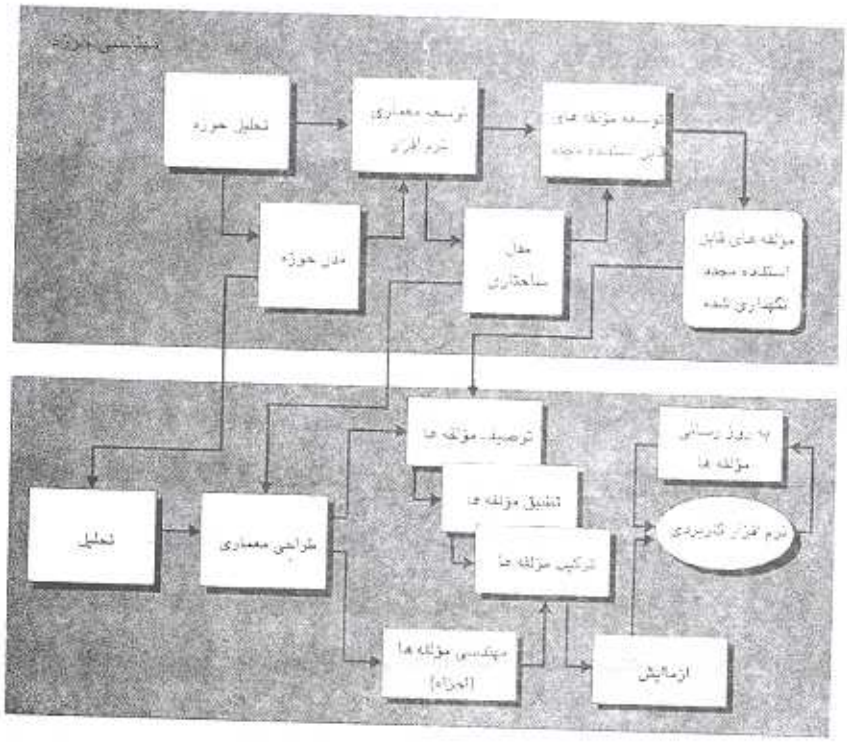
در فصل ۲ از مدل توسعه مبتنی بر جزء (شکل ۲-۱۱) برای توصیف نحوهٔ ادغام کتابخانهٔ دارای اجزاء کاندید استفاده مجدد در مدل روند تکاملی، استفاده کردیم. البته رویه CBSE باید به‌نحوی مشخص شود که نه‌تنها اجزاء کاندید را تعیین کند بلکه باید ارتباط هر جزء را نیز ارزیابی کرده و اجزاء را برای حذف عدم انطباق‌های معماری تعدیل کرده، اجزاء را در سبک معماری انتخاب شده سوار کرده و آنها را بر طبق نیازمندی‌های تغییر سیستم ارتقاء دهد.**[BRO96]** [8] مدل فرآیند مهندسی نرم‌افزار مبتنی بر جزء بر

ارجاع به وب

سایت وب فن آوری اجزاء، اطلاعات مفیدی در خصوص CBSE فراهم آورده است: www.odateam.com/cop/

1.run-time software component
2.software component
3.business component
4.qualified components
5.adapted components
6.assembled components
7.updated components
8.Brown,A.W.

مسیرهای موازی تأکید دارد که در آن مهندسی قلمرو[1] به‌صورت هم‌زمان و ملازم با توسعه مبتنی برجزء بروز می‌کند. مهندسی قلمرو کار لازم برای ایجاد یک‌سری اجزاء نرم‌افزاری را انجام می‌دهد که توسط مهندس نرم‌افزار قابل استفاده مجدد هستند. سپس این اجزاء در طول مرزی که مهندسی قلمرو را از توسعه مبتنی بر جزء جدا می‌کند انتقال داده می‌شوند.



شکل ۲۷-۱ مدل فرایند که از CBSE پشتیبانی می کند

شکل ۲۷-۱ مدل فرایند شاخصی را نشان می‌دهد که با CBSE هم‌ساز است.[CHR95] [2] مهندسی قلمرو ، مدلی خلق می‌کند که به‌عنوان مبنایی برای تجزیه و تحلیل نیازمندی‌های کاربر در جریان مهندسی نرم‌افزار به‌کار می‌رود. یک معماری عمومی نرم‌افزار (با توجه به نکات ساختاری بخش ۲۷-۳-۳) ورودی برای طراحی برنامه کاربردی ایجاد می‌کند. درنهایت بعد از خریداری اجزاء قابل استفاده مجدد که از میان کتابخانه موجود انتخاب شده یا ساخته شده‌اند آنها درطول فعالیت توسعه مبتنی بر جزء، در اختیار مهندسین نرم‌افزار قرار می‌گیرند.

نقل قول

مهندسی حوزه (حیطه) در رابطه با یافتن مشترکات سیستم ها با استفاده از اجزایی است که در بسیاری از سیستم ها کاربرد دارند. این امر به همراه تعیین خانواده ای از برنامه ها ضورت می پذیرد که کاملا" از امتیاز این اجزاء برخوردار باشند.
پاول کلمنتز

1.domain engineering
2.Christensen,S.R.

۲۷-۳ مهندسی حوزه

هدف مهندسی قلمرو (حوزه) مشخص کردن، فهرست کردن و انتشار یک سری اجزاء نرم‌افزاری است که در نرم‌افزارهای فعلی و آتی در قلمرو کاربردی مشخص، قابلیت استفاده دارند. هدف کلی ایجاد مکانیزم‌هایی است که به مهندسین نرم‌افزار امکان می‌دهند تا درحین کار در سیستم‌های جدید موجود، این اجزاء را به اشتراک گذارده و از آنها دوباره استفاده کنند.

مهندسی قلمرو یا حوزه سه فعالیت اصلی تحلیل، ساخت و انتشار را در بر می‌گیرد. بررسی تحلیل قلمرو در فصل ۲۱ مطرح شد. البته این عنوان در بخش بعدی نیز آورده می‌شود. ساخت و انتشار قلمرو در بخش‌های بعدی این فصل مورد توجه واقع خواند شد. می‌توان گفت که استفاده مجدد به واسطه ادغام و نه حذف - در عملکرد اصلی مهندسی نرم‌افزار - از بین می‌رود.[TRA95] [1] چون تأکید بیشتری بر استفاده مجدد اعمال می‌شود برخی معتقدند که مهندسی قلمرو دردهه بعدی به اندازه مهندسی نرم‌افزار اهمیت پیاده خواهد کرد.

۲۷-۳-۱ فرآیند تحلیل حوزه (قلمرو یا میدان)

در فصل ۲۱ روش کلی تحلیل قلمرو را در محتوای مهندسی نرم‌افزار شیء مدار بررسی کردیم. مراحل این روند به شرح زیر تشریح شدند.

۱- تشریح قلمروبی که باید بررسی شود.

۲- طبقه‌بندی اقلامی که از قلمرو گرفته شده اند.

۳- جمع‌آوری نمونه‌هایی از عملیات و کاربردها در قلمرو.

۴- تحلیل هر کاربرد در نمونه.

۵- عرضه مدل تحلیل برای اشیا.

پرسش

چگونه می توانیم اجزاء با قابلیت استفاده مجدد را شناسایی و طبقه بندی نماییم ؟

توجه به این امر مهم است که تحلیل در هر الگوی مهندسی نرم‌افزار قابل اجرا است و می‌توان آن را در توسعه متعارف و شئ گرا بکار برد.

پریتو دیاز[PRI87] [2] مرحله تحلیل قلمرو فوق‌الذکر را، توسعه داده و رهیافت هشت مرحله‌ای را برای مشخص کردن و طبقه‌بندی اجزاء قابل استفاده مجدد ذکر می‌کند:

۱- انتخاب کارکردهای اشیا مشخص .

۲- تجرید کارکردها یا اشیاء.

۳- تشریح یک طبقه‌بندی.

۴- مشخص کردن ویژگی‌های مشترک.

---

1. Tracz, W.
2. Prieto-Diaz, R.

۵- مشخص کردن روابط ویژه

۶- تجزیه روابط

۷- تشریح مدل کارکردی

۸- تشریح یک زبان قلمرو

زبان قلمرو[1] امکان مشخص کردن و سپس ساخت برنامه های کاربردی را در قلمرو، فراهم می‌کند.

البته مراحل فوق‌الذکر مدل مفیدی برای تحلیل قلمرو ارائه می‌کنند اما هیچ راهنمایی برای تصمیم‌گیری جهت انتخاب اجزاء نرم‌افزار برای استفاده مجدد ارائه نمی‌کنند

هاچینسون و هایندلی، مجموعه پرسش‌های عملی زیر را به‌عنوان راهنمایی برای مشخص کردن اجزاء نرم‌افزار قابل استفاده مجدد ذکر می‌کنند:

آیا در پیاده‌سازی آتی به کارکردپذیری جزء نیاز داریم؟

کارکردجزء در قلمرو تا چه حد متداول و مشترک است؟

آیا در قلمرو تکرار کارکرد جزء موجود دارد؟

آیا جزء به سخت‌افزار وابسته است؟

آیا سخت‌افزار در حین پیاده‌سازی بدون تغییر می‌ماند؟

آیا می‌توان مشخصات سخت‌افزار را به جزء دیگری انتقال داد؟

آیا طراحی جهت پیاده‌سازی بعدی به اندازه کافی بهینه شده است؟

آیا می‌توان جزء فاقد قابلیت استفاده مجدد را به‌نحوی پارامتری کرد که قابل استفاده مجدد باشد؟

آیا اجزاء با تغییر اندک در پیاده‌سازی های مختلف قابل استفاده مجدد خواهند بود؟

آیا استفاده مجدد از طریق تغییر و اصلاح عملی است؟

آیا جزء فاقد قابلیت استفاده مجدد را می‌توان برای حصول به جزء دارای قابلیت استفاده مجدد تبدیل کرد؟

تجربه جزء برای استفاده مجدد تا چه حد معتبر است؟

بحث عمیق در مورد روش‌های تحلیل قلمرو خارج از حیطه این کتاب است. برای کسب اطلاعات بیشتر در مورد تحلیل قلمرو، به **[PRI93]** رجوع کنید.

۲-۳-۲۷ خصوصیات کارکردها (توابع)

گاهی تعیین این مطلب مشکل است که آیا جزء دارای قابلیت استفاده مجدد در واقع در یک موقعیت مشخص قابل استفاده است یا خیر. برای تعیین این مطلب باید یک سری خصوصیات قلمرو را مشخص کنید که در تمای نرم‌افزارهای یک قلمرو مشترک است. مشخصه قلمرو، برخی ویژگی‌های ذاتی تمامی

1.domain language

محصولات موجود در قلمرو را مشخص می‌کند. به‌عنوان نمونه ویژگی‌های اصلی باید موارد زیر را شامل شوند: اهمیت ایمنی/ قابلیت اطمینان، زبان برنامه سازی، هم‌زمانی (همروندی) در پردازش و بسیاری موارد دیگر.

می‌توان مجموعه خصوصیات قلمرو برای جزء قابل استفاده مجدد را به‌صورت $\{D_p\}$ نشان داد که در آن هر قلم، D_{pi} ، در مجموعه خصوصیت قلمرو ویژه‌ای را نشان می‌دهد. ارزش اختصاص داده شده به Dpi معرف مقیاس ترتیبی و وضعی است که نشانه ارتباط خصوصیت با جزء p است. مقیاس شاخص باید:[BAS94] [1]

۱ ـ به مناسب بودن یا نبودن استفاده مجدد ارتباط نداشته باشد.

۲ ـ تنها در شرایط غیرمعمول صادق باشد.

۳ ـ علی‌رغم اختلافات می‌توان جزء را به‌نحوی تغییر داد که قابل استفاده باشد.

۴ ـ دارای ارتباط روشن، و درصورتی که نرم‌افزار جدید این ویژگی را نداشته باشد استفاده مجدد ناکافی خواهد بود اما هنوز امکان‌پذیر است.

۵ ـ دارای ارتباط روشن، و درصورتی‌که نرم‌افزار جدید این ویژگی را نداشته باشد استفاده مجدد ناکافی خواهد بود و استفاده مجدد بدون این خصوصیات پیشنهاد نمی‌شود.

پرسش

کدامیک از اجزاء شناسایی شده طی تحلیل حوزه ای، برای استفاده مجدد کاندید خواهند بود؟

وقتی قرار است نرم‌افزار جدید w را در قلمرو کاربرد بسازیم یک‌سری خصوصیات قلمرو برای آن بدست می‌آوریم. سپس مقایسه‌ای بین D_{pi} و D_{wi} انجام می‌شود تا مشخص شود که آیا جزء p موجود به شکل مؤثر در کاربرد w قابل استفاده مجدد است یاخیر.

1.Basili, V.R.

افراد	فرآیند	محصول
انگیزه	مدل فرآیند	پایداری نیازمندیها
تخصیلات	محیط پروژه	نرم افزار همروند
تجربه / آموزش	محدودیتهای زمان بندی	محدودیتهای حافظه
• حوزه کاربرد	محدودیتهای بودجه بندی	اندازه برنامه های کاربردی
• فرآیند	بهره وری	پیچیدگی رابط کاربر
• سکو		(زبان های برنامه سازی)
• زبان		امنیت / قابلیت اطمینان
تیم توسعه		نیازمندیهای دوره فعالیت
بهره وری		کیفیت محصول
		قابلیت اطمینان محصول

جدول ۲۷-۱ ویژگیهای حوزه ای مؤثر بر استفاده مجدد[BAS94]

ارجاع به وب

گزارشی ارزنده از فن آوری، معماری، و تحلیل حوزه ای شئ گرا در آدرس زیر وجود دارد: www.sei.cmu.edu/mbse/wisr_reports.html

جدول ۲۷-۱ خصوصیات قلمرو شاخص را نشان می‌دهد که می‌توانند بر استفاده مجدد از نرم‌افزار تأثیر گذارند این خصوصیات قلمرو باید برای استفاده مجدد از جزء درنظر گرفته شوند.

حتی وقتی نرم‌افزاری که باید مهندسی شود در قلمرو کاربرد به روشنی وجود دارد، اجرا استفاده مجدد در آن قلمرو باید برای تعیین کاربردپذیری تحلیل شوند. در برخی موارد ابداع مجدد، مؤثرترین و اقتصادی‌ترین گزینه است.

۲۷-۳-۳ مدل سازی ساختاری و نکات ساختار

وقتی تحلیل قلمرو اعمال می‌شود، تحلیل‌گر به‌دنبال الگوهای تکراری در کاربردهایی است که در قلمرو ایجاد می‌شوند. مدل‌سازی ساختاری[1] روش مهندسی قلمرو مبتنی بر الگویی است که با این فرض کار می‌کند که هر قلمرو کاربرد دارای الگوهای تکراری است که از پتانسیل استفاده مجدد برخوردارند.

پولاک و ریسمن[POL94][2] مدل‌های ساختاری را به شکل زیر تشریح می‌کنند:

مدل‌های ساختاری از تعداد کمی عناصر ساختاری تشکیل می‌شوند که الگوهای تأثیر متقابل روشنی را نشان می‌دهند.

1 structural modeling
2 Pollak, W.

معمارهای سیستم‌ها که از مدل‌های ساختاری استفاده می‌کنند با اثر کلی مضاعف ایجاد شده از این عناصر مدل مشخص می‌شوند. سپس تأثیرات متقابل پیچیده در میان سیستم‌ها با بسیاری از واحدهای معماری از الگوهای ساده تأثیر متقابل میان این تعداد اندک عناصر پدید می‌آیند.

هر قلمرو کاربرد را می‌توان با مدل ساختاری مشخص کرد پس مدل ساختاری سبک معماری است که می‌توان و باید از آن در کاربردهای درون قلمرو دوباره استفاده کرد.

مک ماهون[MCM95][1] نقطه ساختاری را یک ساختار تمایز در مدل ساختاری تشریح می‌کند. نقاط ساختاری سه ویژگی متمایز دارند:

۱ـ نقطه ساختاری[2] تجریدی است که باید تعداد محدودی مورد و نمونه داشته باشد. با بیان مجدد این امر در شیء گرایی مداری اندازه سلسله مراتب گروه کوچک می‌شود. به‌علاوه تجرید باید در کل کاربردهای قلمرو ایجاد شود. در غیر این‌صورت تلاش لازم برای بازبینی، مستندکردن و انتشار نقطه ساختاری از نظر هزینه موجه نخواهد بود.

پرسش

یک "نقطه ساختار" چیست و خصوصیات آن کدام است؟

۲ـ قوانین حاکم بر کاربرد نقطه ساختاری باید به‌راحتی قابل درک باشند. به علاوه ارتباط با نقطه ساختاری باید ساده باشد.

۳ـ نقطه ساختاری باید پنهان‌سازی اطلاعات را با ایزوله کردن پیچیدگی‌های موجود در خود نقطه ساختاری، ایجاد کند. این امر پیچیدگی کل سیستم را کاهش می‌دهد.

یک نمونه از نقاط ساختاری به‌عنوان الگوهای ساختاری برای یک سیستم، قلمرو نرم‌افزار برای سیستم‌های اخطاردهنده است. این قلمرو باید سیستم‌های ساده‌ای چون خانه امن (توضیح داده شده در فصلهای پیشین) یا سیستم‌های پیچیده چون فرآیند صنعتی را در برگیرد. البته در هرصورت با یکسری الگوهای ساختاری قابل پیش‌بینی مواجه می‌شویم:

- **یک رابط**[3] که به کاربر امکان می‌دهد تا با سیستم ارتباط برقرار کند.
- **مکانیزم تعیین حد و مرز**[4] که به کاربر امکان می‌دهد تا حدود پارامتری را که باید اندازه‌گیری شود مشخص می‌کند.
- **مکانیزم مدیریت سنسور**[5] که با تمامی سنسورهای کنترل‌کننده ارتباط بر قرار می‌کند.
- **مکانیزم واکنش**[6] که با ورودی ارائه شده توسط سیستم مدیریت سنسور واکنش انجام می‌دهد.
- **مکانیزم کنترل**[7] که به کاربر امکان می‌دهد تا نحوه انجام کنترل را نظارت نماید.

نکته تعریف

یک نقطه ساختار می تواند به عنوان یک الگوی طراحی قلمداد شود که به طور مکرر در حوزه ای مشخص از نرم افزارهای کاربردی یافت می گردد.

1.McMahon,P.E.
2.structure point
3.interface
4.Bounds Setting mechanism
5.sensore management mechanism
6.response mechanism
7.control mechanism

هر کدام از این نقاط ساختاری در معماری قلمرو ادغام می‌شوند.

می‌توان نقاط ساختاری اصلی را تشریح کرد که از برخی قلمروهای کاربرد مختلف، برتری پیدا می‌کنند. [STA94][1]

- **جبهه کاربرد**[2] - GUI تمامی فهرست‌ها، بانل‌ها و تسهیلات ورودی و ویرایش فرمان را شامل می‌شود.
- **پایگاه داده**[3] - انبار و مخزن تمامی اشیا مربوط به قلمرو کاربرد.
- **موتور محاسباتی**[4] - مدل‌های عددی و غیرعددی که داده‌ها را تغییر می‌دهند.
- **تسهیلات گزارش‌دهی**[5] - کارکردی(تابعی) که همه نوع خروجی تولید می‌کند.
- **ویرایش‌گر کاربرد**[6] - مکانیزم انطباق برنامه کاربردی با نیازهای کاربران ویژه

نقاط ساختاری یک جایگزین برای امتیازات کارکردی و تعداد خطوط برنامه، جهت تخمین هزینه نرم‌افزار خوانده می‌شوند. [MCM95][7] بحث مختصر مربوط به هزینه‌یابی با استفاده از نقاط ساختاری در بخش ۲۷-۶-۲ آورده می‌شود.

۴-۲۷ توسعه مبتنی بر اجزاء

توسعه اجزاء یکی از فعالیت‌های CBSE است که بر موارات مهندسی قلمرو انجام می‌گیرد. تیم نرم‌افزاری با استفاده از تحلیل و شیوه‌های طراحی معماری که قبلاً در این کتاب مورد بحث قرارگرفت سبک معماری مناسب برای مدل تحلیلی ایجاد شدهٔ برنامه کاربردی تحت ساخت را، پالایش می‌کند.[8]

پس از تعیین معماری، بایستی اجزائی وارد آن شود که (۱) از طریق کتابخانه‌های قابل استفاده مجدد در دسترس بوده و یا (۲) جهت تأمین نیازهای سفارشی، مهندسی می‌شوند. از این‌رو، جریان کارها برای ایجاد اجزاء دو مسیر موازی است (شکل ۲۷-۱) زمانی‌که اجزاء قابل استفاده مجدد برای یکپارچه‌سازی احتمالی در معماری، در دسترس قرار می‌گیرند، باید واجد شرایط بوده و انطباق یابند. وقتی اجزاء جدید ضرورت یابند، باید مهندسی شوند. سپس اجزاء حاصل داخل الگوی معماری ترکیب یا (یکپارچه) شده و به‌طور کامل مورد آزمون قرار می‌گیرند.

---

1 Staringer, W.
2 application front end
3 data base
4 computational engine
5 reporting facility
6 application editor
7 McMahon, P.E.

۸. قابل ذکر است که سبک های معماری از مدل ساختاری عمومی ساخته شده طی مهندسی حوزه، تبعیت می کند. (شکل ۲۷-۱ را ببینید.)

۲۷-۴-۱ توصیف، تطبیق و ترکیب اجزاء

همان‌طورکه قبلاً دیدیم، مهندسی قلمرو، کتابخانه اجزاء قابل استفاده مجدد را فراهم می‌کند که برای مهندسی نرم‌افزار مبتنی بر اجزاء ضرورت دارند. برخی از این اجزاء قابل استفاده مجدد به‌صورت داخلی ایجاد شده و سایر اجزاء می‌توان از برنامه‌های کاربردی موجود انتخاب کرده و برخی دیگر را از طریق اشخاص ثالث بدست آورد. متأسفانه، وجود اجزاء قابل استفاده مجدد متضمن آن نیست که بتوان آنها را به‌راحتی و به‌گونه‌ای مؤثر در معماری منتخب برای یک کاربرد جدید، یکپارچه و مجتمع کرد. به‌همین دلیل است که وقتی کاربرد اجزاء پیشنهاد می‌شود، توالی فعالیت‌های ایجاد اجزاء به‌کار می‌روند.

شرایط لازم یک جزء (صلاحیت جزء)

صلاحیت جزء[1] تضمین می‌کند که یک جزء وظیفه لازم را انجام خواهد داد، به طرزی مناسب با سبک معماری تعیین شده برای سیستم سازگاری دارد، وخصوصیات کیفی لازم برای برنامهٔ کاربردی (مثل عملکرد، قابلیت اطمینان و قابلیت استفاده) را نمایش خوهد داد.

توصیف رابط اطلاعات مفید را دربارهٔ عملکرد و کاربرد یک جزء نرم‌افزاری بدست می‌دهد اما همهٔ اطلاعات لازم در تعیین این موضوع را دراختیار نمی‌گذارد که آیا یک جزء پیشنهادی می‌تواند

به‌طور مؤثر در یک کاربرد جدید، مورد استفاده مجدد قرار گیرد یا خیر. از میان بسیاری که طی صلاحیت اجزاء مدنظر قرار می‌گیرند. می‌توان به موارد زیر اشاره کرد:[BRO96] [2]

پرسش

کدام فاکتورها طی توصیف جزء باید ملاحظه گردند؟

- رابط برنامه‌نویسی کاربردی (ADI)
- ابزارهای ایجاد و یکپارچه‌سازی لازم توسط اجزاء
- نیازمندیهای زمان اجراء از جمله استفاده منبع (مثل حافظه یا ذخیره)، سرعت یا زمان‌بندی و پروتوکل شبکه
- نیازمندیهای خدمات از قبل رابطهای سیستم عامل و پشتیبانی از سوی اجزاء دیگر
- خصوصیات امنیتی مانند کنترل دستیابی و پروتوکل تأیید
- فرضیات جاساری شده طراحی نظیر استفاده از الگوریتم‌های خاص عددی و غیرعددی
- استثناء گردانی

برای اجزاء قابل استفاده مجدد که به‌طور داخلی ایجاد شده‌اند، ارزیابی هر یک از عوامل فوق، نسبتاً آسان است. به شرط اعمال شیوه‌های مناسب مهندسی نرم‌افزار درحین ایجاد آنها، پاسخ سؤالات ناشی از لیست فوق بدست می‌آید.

1.component qualification
2.Brown,A.W.

با این وجود، تعیین کارآیی‌های داخلی COTS یا اجزاء شخص ثالث بسیار دشوارتر است، زیرا تنها اطلاعات موجود ممکن است خود مشخصه رابط باشد.

تطابق اجزاء

در یک وضعیت ایده‌آل، مهندسی قلمرو کتابخانه اجزائی را ایجاد می‌کند که بتواند به سادگی در معماری یا ساختار برنامه کاربردی ادغام و یکپارچه شوند. بوجودآوردن "یکپارچه‌سازی آسان" یعنی اینکه: ۱) روش‌های هماهنگ مدیریت منبع برای تمامی اجزاء کتابخانه اجراء شده باشند. ۲) فعالیت‌های معمول نظیر مدیریت داده‌ها برای تمامی اجزاء وجود داشته باشد و ۳) رابط‌های داخل معماری و محیط خارج به شیوه‌ای هماهنگ انجام شده باشند.

نقل قول

آنان که حامعیت اجزاء را بر عهده دارند، نیاز به کشف عملیات و شکل اجزاء نرم افزاری دارند. آلن براون و کورث ولناو

در واقع، حتی پس از تعیین صلاحیت کاربرد یک جزء در ساختار برنامه، ممکن است آن جزء تضادهایی را در یک یا چند حوزه فوق‌الذکر نشان دهد. جهت کاهش این تضادها، یک تکنیک انطباق به‌نام "پوشش اجزاء"[1] [BRO96] غالباً به‌کار می‌رود. وقتی تیم نرم‌افزاری بر طراحی داخلی و کد برای یک جزء دستیابی کامل دارد (این حالت اغلب در کاربرد اجزاء COTS وجود ندارد) "پوشش جعبه سفید"[2] به‌کار می‌رود. همانند همتایی خود برای آزمون نرم‌افزاری (فصل ۷) برای رفع تضادها جزئیات پردازش داخلی اجزاء را بررسی کرده و اصلاحات سطح رمز را انجام می‌دهد.

"پوشش جعبه خاکستری"[3]، زمانی مورد استفاده قرار می‌گیرد که کتابخانه، یک زبان بسط اجزاء یا API را ایجاد می‌کند که رفع یا پوشش تضادها را امکان‌پذیر می‌سازد. "پوشش جعبه سیاه"[4] مستلزم معرفی پیش‌پردازی و پس‌پردازی در رابط اجزاء جهت رفع یا پوشش تضادها می‌باشد. تیم نرم‌افزاری بایستی مشخص نماید که آیا تلاش لازم جهت پوشش کافی یک جزء موجه است یا این‌که یک جزء سفارشی (طراحی شده برای رفع تضادها) باید بجای آن، مهندسی شود.

ترکیب اجزاء

عمل ترکیب اجزاء[5]، اجزاء واحدالشرایط، سازگار یافته و مهندسی شده را برای تکمیل معماری تعیین شده جهت برنامه کاربردی، مجتمع می‌سازد. در انجام این کار، برای اتصال اجزاء در داخل یک سیستم اجرائی یک زیر ساخت بایستی تعیین شود. زیرساخت (که معمولاً کتابخانه اجزای تخصصی است) مدلی را برای هماهنگی اجزاء و خدمات خاص ایجاد می‌کند که موجب می‌شود اجزاء با یکدیگر هماهنگ شده و وظایف معمول را انجام دهد.

1.component wrapping
2.white-box wrapping
3.gray-box wrapping
4.black-box wrapping
5.component composition

در میان مکانیزم‌های فراوان ایجاد یک زیرساخت مؤثر و مجموعه چهارتایی "ملزومات معماری" وجود دارد که حضور آن در تحقیق ترکیب اجزاء الزامی است.[ADL95] [1]

پرسش

چه ملزوماتی برای ترکیب اجزا هورد نیازند؟

مدل تبادل‌سازی داده‌ها. مکانیزم‌هایی که کاربران و برنامه‌های کاربردی را قادر به ارتباط و انتقال داده‌ها می‌سازند (مثل کشیدن و انداختن، بریدن و چسباندن) بایستی برای تمامی اجزاء قابل استفاده مجدد تعریف شوند. مکانیزم‌های تبادل اطلاعات نه‌تنها انتقال داده‌ها را به‌صورت انسان با نرم‌افزار و جزء با جزء امکان‌پذیر می‌سازند. بلکه هم‌چنین انتقال مابین منابع سیستم را نیز فراهم می‌آورند.

(مثل کشیدن یک فایل به نماد چاپگر برای خروج)

خودکارسازی. جهت تسهیل ارتباط بین اجزاء قابل استفاده مجدد، مجموعه‌ای از ابزارها، دستورات درشت و دستورالعمل‌ها بایستی به‌کار گرفته شوند.

حافظه ساختیافته. اطلاعات ناهمگن (مثل اطلاعات گرافیکی، متن، صوتی/ تصویری و داده‌های عددی) که داخل یک "سند مرکب" قرار دارند، باید به‌نحوی سازمان‌دهی شوند که به‌صورت یک ساختار داده‌ای واحد قابل دسترسی باشند تا مجموعه‌ای از فایل‌های جداگانه.

"داده‌های ساختیافته، شاخص توصیف‌گر ساختارهای متداخلند. چنانچه برنامه‌های کاربردی می‌توانند محتویات داده‌ای جداگانه را که تحت هدایت کاربرنهایی هستند، آزادانه مکان یابی، ایجاد یا ویرایش کنند." [ADL95]

مدل شیء گرای زیر بنایی. مدل شیءگرا تضمین می‌کند که اجزاء ایجاد شده در زبان‌های مختلف برنامه‌نویسی که در سیستم عامل‌های مختلف قرار دارند، می‌توانند عمل‌پذیری درونی داشته باشند. یعنی این‌که، اشیا باید در سراسر یک شبکه قادر به برقراری ارتباط باشند. برای دستیابی بر این هدف، مدل شیءگرا استانداردی را جهت عمل‌پذیری درونی اجزاء تعریف می‌کند.

ارجاع به وب

آخرین اطلاعات مربوط به CORBA در آدرس زیر قرار دارد: www.omg.org

از آن‌جا که تأثیر احتمالی استفاده مجدد و CBSE در صنعت نرم‌افزار بسیار قابل‌توجه است، تعدادی از کمپانی‌ها و اتحادیه‌های مهم صنعتی[2]، استانداردهایی را برای نرم‌افزار اجزاء پیشنهاد داده‌اند.

CORBA / OMG. گروه مدیریت شیءای یک "معماری واسطه درخواست شیء"[3] را منتشر کرده است. واسطه درخواست شیء (ORB) خدماتی را عرضه می‌کند که ارتباط اجزاء قابل استفاده مجدد (اشیاء) با سایر اجزاء را بدون درنظر گرفتن محل آنها در داخل سیستم، امکان‌پذیرمی‌سازد. وقتی اجزاء با به‌کارگیری استاندارد OMG / CORBA ساخته می‌شوند، درصورتی‌که "زبان تعریف رابط"[4] (IDL) برای هر یک از اجزا ایجاد شود، یکپارچگی آن اجزاء

(بدون اصلاح) در داخل سیستم، تضمین می‌گردد.

Adler,R.M.

۲. توصیفی عالی از استانداردهای مربوط به "اشیاء توزیع شده" در [ORF96] و [YOU98] ارائه شده است.

3. common object request broker architecture(OMG/CORBA)

4. interface definition language

با به‌کارگیری استعارهٔ مخدوم/ خادم، اشیا داخل برنامه کاربردی مخدوم یک خدمت یا خدماتی چند را از خادم ORB تقاضا می‌کند. در خواست‌ها از طریق یک IDL یا به‌طور پویا در زمان اجرا، انجام می‌گیرند.

محرن رابط، تمامی اطلاعات لازم دربارهٔ تقاضای خدمات و قالب‌های پاسخ‌گویی را، در بردارد. CORBA در فصل ۲۸ بیشتر مورد بحث قرار می‌گیرد.

MicrosoftCOM. مایکروسافت یک مدل شیءگرا اجزاء COM را توسعه داده است که برای کاربرد اجزا تولید شده توسط عرضه‌کنندگان مختلف در داخل یک برنامهٔ منفرد که تحت سیستم عامل ویندوز اجرا می‌شود، مشخصه‌ای را تعیین می‌کند. COM دو عنصر را در بردارد: رابط‌های COM (که به‌صورت اشیاء COM اجزاء می‌شوند) و مجموعه مکانیزم‌هایی برای ثبت و عبور پیغام‌ها بین رابط‌های COM) از نظر نقطه‌نظر کاربرد، "تأکید بر چگونگی پیاده‌سازی اشیاء COM نیست، بلکه تأکید تنها بر این واقعیت است که شیء دارای رابطی است که با سیستم به ثبت رسانده و این‌که سیستم اجزاء را برای برقراری ارتباط با سایر اشیا COM به‌کار می‌برد."[HAR98]

آخرین اطلاعات مربوط به COM در آدرس زیر قرار دارد:
www.microsoft.com/COM

اجزاء Sun Javabean. سیستم اجزاء JavaBean یک زیرساخت CBSE قابل انتقال و مستقل از سیستم عامل است که با استفاده از زبان برنامه‌نویسی Java ایجاد می‌گردد. سیستم JavaBean اپلت‌های جاوا[1] برای تطبیق با اجزاء پیچیده‌تر نرم‌افزاری که در توسعه اجزا ضروری هستند را، ارائه می‌دارد. سیستم JavaBean ، مجموعه ابزارهایی با نام *لوازم توسعه Bean ، (BDK)*[2] را در بردارد که بر توسعه‌دهندگان امکان می‌دهد تا:

آخرین اطلاعات مربوط به javabean ها در آدرس زیر قرار دارد:
www.java.sun.com/beans

(۱) چگونگی عملکرد اجزا موجود Beans را تحلیل کنند، (۲) رفتار و ظاهر آنها را سفارشی نمایند، (۳) مکانیزم‌هایی را برای هماهنگی و ارتباط تعیین کنند،(۴) Beans سفارشی را برای کاربرد در یک برنامه خاص توسعه دهند، (۵) رفتار Bean را آزمون و ارزیابی کنند.

کدام یک از این استانداردها در صنعت رواج خواهند یافت؟ درحال حاضر نمی‌توان به‌راحتی به این سؤال پاسخ داد. گرچه بسیاری از توسعه‌دهندگان یکی از استانداردهای فوق‌الذکر را مدنظر قرار داده‌اند، احتمال می‌رود که سازمان‌های بزرگ نرم‌افزاری بسته به مقولات کاربردی و سیستم عامل‌های انتخابی، هر سه استاندارد را به‌کار برند.

۲۰-۴-۲ مهندسی

همان‌طور که قبلاً در این فصل یادآور شدیم، فرایند CBSE استفاده از اجزاء موجود نرم‌افزاری را توصیه می‌کند، هرچند که گاهی اوقات لازم است اجزاء مهندسی شوند. یعنی، اجزاء جدید نرم‌افزاری باید

---

۱. در این متن، یک اپلت به عنوان موادء ای ساده دیده شده است.

2 Bean Development Kit (BDK)

ایجاد شده و با اجزاء موجود COTS، داخلی ادغام گردند. از آن‌جا که این اجزاء جدید، به عضویت کتابخانه داخلی استفاده مجدد، در می‌آیند، باید برای استفاده مجدد نیز، مهندسی شوند.

در ارتباط با ایجاد اجزاء نرم‌افزاری که می‌توانند مجدداً مورد استفاده قرار بگیرند، هیچ چیز خارق‌العاده‌ای وجود ندارد. مفاهیم طراحی نظیر انتزاع، اختفاء، استقلال کارکردی، پالایش و برنامه‌نویسی ساخت یافته همراه با روش‌های شیءگرا، آزمون، SQA و شیوه‌های تأیید درستی، همگی در ایجاد اجزاء نرم‌افزاری با استفاده مجدد، دخیل و تأثیر گذارند.[1] در این بخش ما مجدداً بر این موضوعات نمی‌پردازیم. بلکه در عوض مسایل خاص استفاده مجدد را، بررسی خواهیم کرد که مکمل روش‌های یکپارچه مهندسی نرم‌افزار هستند.

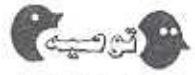
۳-۴-۲۷ تحلیل و طراحی برای استفاده مجدد

اجزاء مدل تحلیلی به‌طور مفصل در بخش‌های سه و چهار این کتاب مورد بحث و بررسی قرار گرفتند. جهت توصیف کار اجرائی یک برنامه کاربردی خاص، می‌توان مدل‌های داده‌ای، کارکردی و رفتاری (که در مجموعه‌ای از نشان‌گذاری‌های مختلف به‌نمایش درمی‌آیند) را ایجاد کرد. آن‌گاه مشخصات نوشتاری برای توصیف این مدل‌ها به‌کار می‌روند. نتیجه حاصل توصیف کامل نیازمندیها می‌باشد.

درحالت ایده‌آل، مدل تحلیلی برای تعیین آن عناصری از مدل که به اجزاء موجود و قابل استفاده مجدد اشاره دارند، مورد تجزیه و تحلیل قرار می‌گیرد. مشکل، استخراج اطلاعات از مدل نیازمندیها به‌نحوی است که بتواند به "تطابق مشخصات" منجر گردد. بلین زونا و همکارانش [BEL95][2] یکی از روش‌های سیستم‌های شیءگرا را توصیف می‌کنند:

اجزاء به‌صورت طبقات مشخصه‌ای، طراحی و پیاده‌سازی در سطوح مختلف انتزاع، تعریف و ذخیره می‌شوند. به‌طوری‌که هر طبقه، توصیف مهندسی شده محصولی از برنامه‌های قبلی است. دانش مشخصات ـ دانش توسعه ـ به شکل طبقات پیشنهادی استفاده مجدد، ذخیره می‌شود که دستورالی را برای بازیابی اجزاء قابل کاربرد مجدد براساس توصیفشان و نیز ساخت و مناسب نمودن آنها پس از بازیابی را در بردارد.

در تلاش برای تطبیق ضروریات اشاره شده در توصیف فعلی با موارد تشریح شده برای اجزاء (طبقات) موجود و قابل استفاده مجدد، ابزارهای خودکار جهت جستجوی یک مخزن به‌کار می‌روند. توابع مشخصات (قسمت ۲۷ ـ ۳ ـ ۲) و واژه‌های کلیدی برای کمک در یافتن اجزائی که بالقوه قابل کاربرد هستند، مورد استفاده قرار می‌گیرند.



حتی اگر سازمان شما، مهندسی حوزه را به کار نمی بندد، شما به شخصه آن را در کار خود به طور غیر رسمی بکار گیرید. هنگام که مدل تحلیلی را می سازید از خود بپرسید : " آیا به نظر نمی رسد، این شئ یا عملیات در دیگر برنامه ها کاربردی مشابه، وجود داشته باشد؟ " اگر پاسخ مثبت بود، جزء احتمالاً موجود می باشد.

---

[1] ـ برای آموختن بیشتر این عناوین به فصلهای ۱۳-۱۶ و ۲۰-۲۲ مراجعه کنید.

2. Bellinzona R.

اگر تطبیق مشخصات اجزایی متناسب با نیازهای برنامه کاربردی فعلی را بدست دهد، طراح می‌تواند این اجزاء را از کتابخانه (گنجینه) قابل استفاده مجدد استخراج کرده و آنها را در طراحی سیستم‌های جدید به‌کار برد.

اگر اجزاء طراحی یافت نشوند، مهندس نرم‌افزار باید در ایجاد آنها از روش‌های طراحی قراردادی یا OO استفاده کند. در همین زمان یعنی وقتی طراح ایجاد یک جزء جدید را آغاز می‌کند ـ "طراحی برای استفاده مجدد: DFR "[1] باید مدنظر قرار گیرد.

همان‌طورکه قبلاً اشاره کردیم، DFR مستلزم آن است که مهندس نرم‌افزار، مفاهیم و اصول یکپارچه طراحی نرم‌افزار را به‌کار برد. (فصل ۱۳). اما خصوصیات حوزه کاربردی نیز باید مورد توجه قرار گیرد. بایندر **[BIN93]** برخی مسائل مهم و اساسی در طراحی برای استفاده مجدد[2] را پیشنهاد می‌کند:

داده‌های استاندارد. حوزه کاربردی باید مورد بررسی قرار گرفته و ساختارهای داده‌ای استاندارد و سراسری (مثل ساخت‌های فایل یا یک پایگاه داده‌های کامل) باید تعیین و شناسائی شوند. آن‌گاه می‌توان تمامی اجزاء طراحی را برای استفاده از این ساختارهای داده‌ای استاندارد، تعریف و توصیف کرد.

پروتوکل‌های رابط استاندارد. سه سطح پروتوکل رابط بایستی تعیین شود: طبیعت رابط‌های داخل پیمانه ها ، طراحی رابط‌های فنی خارجی (غیرانسانی) و رابط انسان / ماشین .

الگوهای برنامه. مدل ساختاری (قسمت ۲۷-۳-۳) می‌تواند در طراحی معماری یک برنامه جدید مؤثر واقع شود.

پس از تعیین داده‌های استاندارد، رابط‌ها و الگوهای برنامه، طراح چارچوبی را برای ایجاد طراحی دراختیار دارد. اجزای جدیدی که با این چارچوب مطابقت می‌کنند، احتمال بیشتری برای کاربرد مجدد بعدی برخوردارند.

ساخت اجزا قابل استفاده مجدد نیز مانند طراحی وابسته به روش‌های مهندسی نرم‌افزار است که در سایر قسمت‌های این کتاب مورد بحث قرار گرفته‌اند.

ساخت و به‌کارگیری برنامه‌نویسی قراردادی نسل سوم، زبان‌های نسل چهارم و تولیدکننده‌های کد فنون برنامه‌نویسی ویژوال با ابزارهای پیشرفته‌تر قابل انجام است.

۲۷-۵ رده بندی و بازیابی اجزاء

کتابخانه یک دانشگاه بزرگ را درنظر بگیرید. ده‌ها هزار کتاب، مجلات و سایر منابع اطلاعاتی برای استفاده موجودند. اما برای دستیابی به این منابع، یک طرح طبقه‌بندی بایستی ایجاد شود. به‌منظور این حجم وسیع اطلاعاتی، کتابداران یک طرح طبقه‌بندی را مشتمل بر کد طبقه‌بندی کتابخانه کنگره(آمریکا-

توصیه

هرچند اگر استفاده مجدد یک هدف اصلی باشد، موضوعات خاص دیگر نیز باید مد نظرباشند، بنابرین بر اصول طراحی خوب تکیه کنید. اگر اینگونه عمل کنید، بخت شما برای استفاده مجدد اجزاء به طور قابل ملاحظه ای افزایش می یابد.

نقل قول

مهمترین چیزی که در کنار چیزهای دیگر باید بدانید، دانستن آن است که آنرا کجا بیابید. ساموئل جانسون

1.design for reuse (DFR)

۲. به طور کلی، طراحی استفاده مجدد، بخشی از مهندسی حوزه خواهد بود.

م)، واژه‌های کلیدی ، نام نویسنده در سایر مداخل راهنما تعریف و مشخص کرده‌اند. تمامی اینها کاربر را قادر می‌سازند تا سریع و راحت منبع مورد نیاز خود را پیدا کنند.

حال مخزن یا گنجینه بزرگ اجزاء را مدنظر قرار دهید. ده‌ها هزار اجزاء قابل استفاده مجدد نرم‌افزاری در آن وجود دارند. اما یک مهندس نرم‌افزار چگونه جزء مورد نیاز خود را پیدا می‌کند؟ پاسخ به این سؤال، پرسش دیگری را به‌دنبال دارد. چگونه می‌توانیم اجزا نرم‌افزاری را به زبانی غیرمبهم و قابل طبقه‌بندی توصیف کنیم؟ اینها سؤالات دشواری هستند و تاکنون هیچ پاسخ قطعی بدان‌ها داده نشده است. در این قسمت ما دست دستورالعمل‌های فعلی را که امکان جستجوی کتابخانه‌های کاربرد مجدد را برای مهندسان نرم‌افزاری آتی فراهم می‌سازند، مورد مطالعه و بررسی قرار می‌دهیم.

برای تعریف یک جزء یا خاصیت استفاده مجدد، به ترتیب : مفهوم آن، مندرجات آن، و محتوی آن باید تشریح گردند.

۲۷-۵-۱ تعریف اجزاء قابل استفاده مجدد

یک جزء نرم‌افزاری قابل استفاده مجدد را می‌توان به صورت‌های مختلف توصیف کرد، اما یک شرح یا توصیف مطلوب در بردارنده آن چیزی است که تراز [TRA90][1] آن را مدل 3C[2] نامیده است و شامل مفهوم، محتوا و بافت می‌باشد.

مفهوم[3] یک جزء نرم‌افزاری عبارت است از "توصیف آن‌چه که جزء انجام می‌دهد". رابط جزء به‌طور کامل تشریح می‌گردد و معناشناسی در محدوده بافت پیش‌شرط‌ها و پس‌شرط‌ها، تعیین و شناسایی می‌شود.

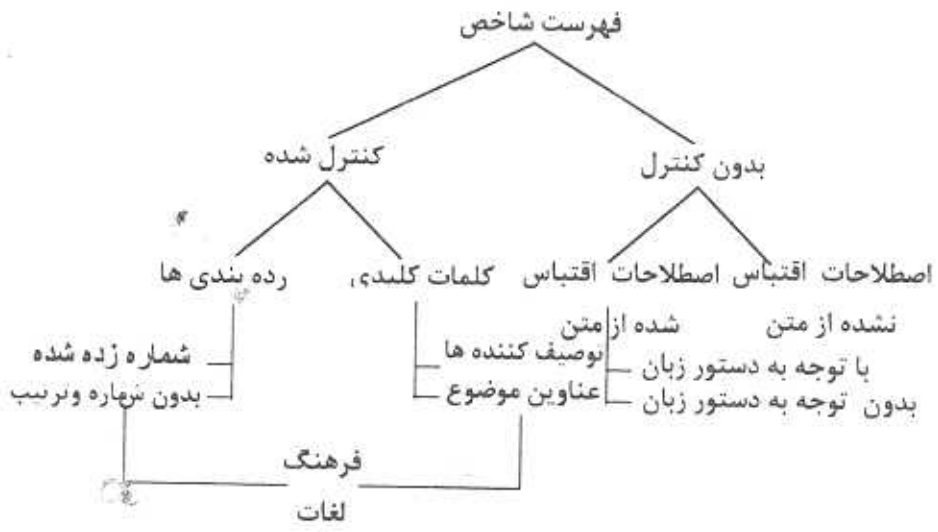
محتوای[4] یک جزء چگونگی تحقق مفهوم را توصیف می‌کند. اساساً، محتوا اطلاعاتی است که از کاربران موقت مخفی می‌شود و فقط در اختیار کسانی قرار می‌گیرند که قصد اصلاح آن جزء را دارند.

بافت[5] **یا زمینه**، جزء نرم‌افزاری قابل استفاده مجدد را در قلمرو کاربرد آن قرار می‌دهد. یعنی، بافت با تعیین ویژگی‌های مفهومی، عملیاتی و پیاده‌سازی، به مهندس نرم‌افزار امکان می‌دهد تا جزء مناسب در تأمین نیازمندیهای برنامه کاربردی را پیدا کند.

یک راهنمای متصل برای استفاده مجدد اجزاء از آدرس زیر قابل پیاده سازی می باشد :
Web1.ssg.gunter.af.mil/sep/SEPver40/main.html#GD

برای قابلیت کاربرد در یک محیط کاربردی، مفهوم، محتوا و بافت بایستی در یک طرح مشخصه‌ای ملموس تبدیل و ترجمه شوند. مقالات و تألیفات بسیار زیادی در بارهٔ طرح‌های طبقه‌بندی اجزاء قابل کاربرد مجدد نرم‌افزاری نوشته شده است. (به عنوان مثال [WHI 95] کتابشناسی جامعی را شامل است). شیوه‌های پیشنهادی را می‌توان به سه حوزه اصلی تقسیم کرد: روش‌های علم اطلاعات و بایگانی، شیوه‌های هوش مصنوعی و سیستم‌های فرامتنی. اکثریت وسیعی از کارهایی که تاکنون انجام شده، کاربرد شیوه‌های علم بایگانی را برای رده‌بندی اجزاء پیشنهاد میدهد.

1.Tracz , W.
2.3C model
3.concept
4.content
5.context



شکل ۲۷-۲ سلسله مراتبی از متدهای شاخص بندی

شکل ۲۷-۲ طبقه‌بندی روش‌های فهرست‌سازی علم بایگانی را ارائه می‌کند. واژگان کنترل شده شاخص‌گذاری[1] الفاظ یا نحوی را که در طبقه‌بندی یک شیء یا (جزء) می‌تواند به‌کار رود را محدود می‌سازند. واژگان کنترل نشده فهرست‌سازی[2] هیچ‌گونه محدودیتی را در توصیف ایجاد نمی‌کند. اغلب طرح‌های طبقه‌بندی اجزاء نرم‌افزاری به سه دسته تقسیم می‌شوند:

طبقه‌بندی لیست شده.

اجزاء به‌واسطه یک ساختار سلسله مراتبی مشخص توصیف می‌شوند که در آن طبقات و سطوح مختلف طبقات فرعی نرم‌افزار قرار می‌گیرند. اجزاء واقعی در پایین‌ترین سطح هر مسیر سلسله مراتب در شده فهرست می‌شوند. به‌عنوان مثال، سلسله مراتب ذکرشده، برای عملیات پنجره[3] ممکن است به‌صورت زیر باشد:

پرسش

تمام گزینه ها برای رده بندی اجزاء در اختیار می باشد؟

```
Window operations
    Dis play
        Open
          Menu – based
            Open window
    System – based
        Sys window
Close
```

1.controlled indexing vocabularies
2.uncontrolled indexing vocabularies

۳. تنها مجموعه ای کوچک از تمام عملیات ممکن ارائه گردیده است.

```
  Via pointer
  ...
resize
  via command
      Set window size , S + d  Resize , Skrink window
  Via dray
      Pull window , stretch window
Up / down   shuffle
    ...
move
    ...
close
    ...
```

ساختار سلسله مراتبی یک طرح طبقه‌بندی لیست شده، درک و کاربرد ان را آسان می‌نماید. هرچند که قبل از ایجاد یک سلسله مراتب، مهندسی قلمرو باید انجام شود تا اطلاعات کافی در باره مراحل مناسب سلسله مراتب در دسترس قرار گیرد.

طبقه‌بندی شکلی. یک حوزه قلمرو تحلیل شده و مجموعه‌ای از خصوصیات اصلی توصیفی، تعیین می‌گردند. این مشخصات که منظر یا شکل‌ها[1] نام دارند، از لحاظ اهمیت در اولویت قرار گرفته و به یک جزء ملحق می‌شوند. یک شکل یا منظر می‌تواند وظیفه‌ای را که جزء انجام می‌دهد، داده‌هایی که دستکاری می‌شوند، بافتی که در آن به‌کار می‌روند یا هر مشخصه دیگری را توصیف‌کنند. مجموعه طرحهایی که یک جزء را توصیف می‌کنند، "توصیف‌گر شکلی "[2] نام دارد.به‌طورکلی، توصیف شکلی به ۷ یا ۸ طرح محدود می‌شود.

به‌عنوان توضیح ساده‌ای در کاربرد منظرها در طبقه‌بندی اجزاء، طرحی را درنظر بگیرید که از توصیف‌گر شکلی زیر استفاده می‌کند [LIA93][3].

[function , object , type , system type]

هر یک از منظرهای توصیف‌گر طرحدار دارای یک یا چند ارزش است که عموماً واژه‌های کلیدی توصیفی هستند. مثلاً اگر function، طرحی از یک جزء باشد، ارزش‌های معمول منتسب بدان ممکن است به‌صورت زیر باشند:

function = (copy , from) or (copy , replace , all)

کاربرد ارزش‌های چندگانه یک شکل باعث می‌شود تا عمل اولیه copy به‌طور کامل‌تر اصلاح شود.

1.facets
2.facet descriptor
3.Liao,H.

واژه‌های کلیدی[1] (ارزش‌ها) برای مجموعه طرح‌های هر یک از اجزاء در کتابخانه قابل استفاده مجدد، تعیین می‌گردند. زمانی‌که مهندس نرم‌افزار قصد دارد کتابخانه را برای اجزاء احتمالی طراحی جستجو کند، مجموعه‌ای از ارزش‌ها مشخص شده و کتابخانه برای نظیریابی بررسی و جستجو می‌گردد.

شمای طبقه‌بندی طرح‌دار(شکلی) انعطاف‌پذیری بیشتری را برای تعیین توصیف‌گرهای پیچیده اجزاء، در اختیار مهندس قلمرو قرار می‌دهد.در آن‌جا که ارزش‌های جدید هر طرح به‌راحتی قابل افزایش است، توسعه و تطبیق شمای طبقه‌بندی شکلی نسبت به رهیافت قبلی شمارشی(لیست‌شده)، آسان‌تر است.

طبقه‌بندی ارزش ـ ویژگی. برای تمامی اجزاء یک قلمرو، خصوصیاتی تعریف و تعیین می‌شوند. سپس همانند طبقه‌بندی طرح‌دار، ارزش‌هایی بر این ویژگی‌هایی منتسب می‌گردد. در واقع، طبقه‌بندی ارزش ویژگی به جزء استثناء های زیر، به طبقه‌بندی طرح‌دار شباهت دارد:

۱) در مقدار ویژگی‌هایی مورد استفاده هیچگونه محدودیتی وجود ندارد.

۲) ویژگی‌ها با خصوصیات ارجحیت و اولویتی نسبت به یکدیگر ندارند.

۳) فانکشن مترادف نامه به‌کار نمی‌روند.

براساس تحقیق تجربی هر یک از فنون طبقه‌بندی فوق، Pole , Frankes [FRA96] معتقدند که هنوز "بهترین" تکنیک به‌وجود نیامده و هر یک از روش‌های فوق‌الذکر تنها کارآیی متوسطی در جستجو داشته اند.

۲-۵-۲۷ محیط استفاده مجدد

استفادهٔ مجدد جزء نرم‌افزاری باید توسط محیطی با عناصر و عوامل زیر، حمایت و پشتیبانی گردد:

- یک پایگاه داده‌ای اجزاء که قادر به ذخیره اجزاء نرم‌افزاری و اطلاعات طبقه‌بندی لازم برای بازیابی آنها باشد.
- یک سیستم مدیریت کتابخانه‌ای که دستیابی به پایگاه داده‌ای را ممکن می‌سازد.
- سیستم بازیابی جزء نرم‌افزاری (مثل واسطه درخواست شیء) که به برنامه مخدوم امکان می‌دهد تا اجزاء و خدمات را از خادم کتابخانه، بازیابی کند.
- ابزارهای CBSE که یکپارچگی اجزاء مجدد استفاده شده را در یک طراحی یا پیاده‌سازی جدید، پشتیبانی می‌کنند.

هر یک از این کارکردها با محدوده یک کتابخانه کاربرد مجدد ارتباط داشته و یا داخل آنها قرار دارند. کتابخانه کاربرد مجدد یکی از عناصر یک مخزن بزرگتر است (فصل ۳۱) و تسهیلاتی را برای ذخیره اجزاء نرم‌افزاری و مجموعه وسیعی از اشیا قابل استفاده مجدد (مثل مشخصات، طرح‌ها، اجزای برنامه، موارد آزمون، راهنمای کاربر) را فراهم می‌کند.

1.keywords

جستجوها اغلب با استفاده از عنصر بافت یا زمینه در مدل 3C که قبلاً در این قسمت توصیف شد، مشخص می‌گردند. اگر جستجوی آغازین به لیست زیادی از اجزاء کاندید منجر گردد، پرس‌وجو برای محدودسازی لیست، بالایش می‌شود.

توصیف مفصل و جامع ساختار کتابخانه‌ای استفاده مجدد ابزارهای مدیریتی آنان، از حوزه این کتاب خارج است. خواننده علاقه‌مند برای اطلاعات بیشتر باید به [HOO91][1] و [LIN95] مراجعه کند.

۲۷-۶ جنبه اقتصادی CBSE

مهندسی نرم‌افزار مبتنی بر اجزاء از یک جاذبه شهودی برخوردار است. از لحاظ نظری، باید یک سازمان‌دهی نرم‌افزاری با مزایای کیفیت و مناسب را تأمین نماید. و اینها بایستی به صرفه‌جویی‌های هزینه‌ای تبدیل شوند. اما آیا اطلاعات موثقی در تقویت این دیدگاه وجود دارد؟

در پاسخ به این سؤال ابتدا باید دانست که چه ... در بافت مهندسی نرم‌افزار قابل استفاده مجدد است و این‌که واقعاً هزینه‌های مرتبط با کاربرد مجدد چه هستند. که در نتیجه، ایجاد یک تحلیل مقرون به‌صرفه برای کاربرد مجدد اجزاء امکان‌پذیر می‌گردد.

CBSE هنگامی که اجزاء مورد نیاز دقیقاً موجود باشند، بسیار اقتصادی خواهد بود ولی هنگامی که نیاز به تغییر و سفارشی نمودن باشد، با احتیاط عمل نماید.

۲۷-۶-۱ تاثیر بر کیفیت، بهره وری و هزینه

شواهد قابل توجه به دست آمده از مطالعات موردی صنعت (مثلاً [HEN95][2]، [MCM95][3] و [LIM94][4]) نشان می‌دهد که مزایای تجاری چشمگیری از کاربرد مجدد نرم‌افزاری بدست می‌آید کیفیت محصول، بهره‌وری توسعه و هزینه کلی همگی بهبود می‌یابند.

کیفیت. در یک وضعیت ایده‌آل، صحت و درستی جزیی نرم‌افزاری که برای کابرد مجدد توسعه می‌یابد، تأیید شده (فصل ۲۶) و ثابت می‌شود که هیچ عیب و نقصی ندارد. در واقع، تأیید رسمی به‌طور عادی انجام نمی‌شود و احتمال دارد نقایص وجود داشته باشد. هر چند که در هر کاربرد مجدد نقایص شناسایی برطرف می‌شوند و در نتیجه کیفیت جزء ارتقاء می‌یابد. به مرور زمان ، جزء عملاً فاقد عیب و نقص می‌گردد.

هرچند داده های تجربی متنوع می باشند، اما مدارک صنعتی گواهی می دهند که استفاده مجدد، هزینه و سود واقعی را مهیا می سازد.

در مطالعهٔ انجام شده در هیولت پاکارد، لیم [LIM96] گزارش می‌دهد که میزان نقص برای برنامهٔ مجدداً استفاده شده ۰/۹ خرابی در KLOC است، درحالی که میزان آن در یک نرم‌افزار تازه ایجاد شده ۴/۱ در KLOC می باشد. در یک برنامهٔ کاربردی که ۶۸٪ آن از برنامه مجدداً استفاده شده تشکیل شده بود، میزان نقص ۲/۰ در هر KLOC بود ـ یعنی ۵۱٪ بهبود از میزان مورد انتظار نسبت به وقتی

1.Hooper,J.W.
2.Henry,E.
3.McMahon ,P.E.
4.Lim ,W.C.

که کار بدون استفاده مجدد انجام شده بود. هنری و فالر [HEN95] بهبود ۳۵٪ کیفیت را گزارش می‌دهند. گرچه برخی گزارش‌ها بیان‌گر طیف گسترده‌ای از درصدهای بهبود کیفیت هستند، اما منصفانه آن است که بگوئیم کاربرد مجدد از لحاظ کیفیت و اعتبار نرم‌افزار عرضه شده، مزایای قابل تأملی را ارائه می‌کند.

بهره‌وری. به هنگام استفاده از اجزای قابل کاربرد مجدد در فرآیند نرم‌افزاری، زمان کمتری صرف ایجاد طرح‌ها، الگوها، اسناد، برنامه و داده‌ای می‌شود که در ایجاد یک سیستم قابل تحویل ضرورت دارند. از این‌رو، بهره‌وری بهبود می‌یابد. گرچه تفسیر گزارش‌های بهبود درصد بهره‌وری دشوار است،[1] به‌نظر می‌رسد که استفاده مجدد بین ۳۰ تا ۵۰ درصد، می‌تواند به بهبود و اصلاح بهره‌وری از ۲۵ تا ۴۰ درصد منجر شود.

هزینه. صرفه جویی‌های خالص هزینه برای کاربرد مجدد با تخمین هزینه پروژه اگر از صفر آغاز شود C_s و سپس کم کردن مجموع هزینه‌های مرتبط با کاربرد مجدد: C_r و هزینه‌های واقعی نرم‌افزار عرضه شده: C_d برآورد می‌شوند

C_s را می‌توان با به‌کارگیری یک یا چند تکنیک برآورد و تخمین که در فصل ۵ مورد بحث قرار گرفتند، تعیین کرد.

پرسش

چه هزینه هایی بر استفاده مجدد نرم افزار مترتب می باشد؟

هزینه‌های مرتبط با کاربرد مجدد : C_r عبارتند از: [CHR95][1]

- تحلیل و الگوسازی قلمرو
- توسعه معماری قلمرو
- افزایش مستندسازی جهت تسهیل کاربرد مجدد
- پشتیبانی و تقویت اجزای کاربرد مجدد
- حق امتیاز و مجوزها برای اجزاء فراهم شده از خارج
- ایجاد یا فراگیری و عملکرد محزن کاربرد مجدد
- آموزش پرسنل در طراحی و ساخت برای کاربرد مجدد

۲۷-۶-۲ تحلیل هزینه با استفاده از نکات ساختاری

در قسمت ۲۷-۳-۳ ما نقطه ساختاری را به‌عنوان معماری تعریف کردیم اگر در سراسر یک حوزه کاربرد خاص تکرار می‌شود. تحلیل نرم‌افزار با تعریف معماری قلمرو و سپس پر کردن آن با نقاط ساختاری می‌تواند معماری را برای یک برنامه کاربردی، سیستم یا محصول جدید توسعه دهد این نقاط ساختاری، یا اجزاء منفرد قابل استفاده مجدد هستند، یا بسته‌های اجزاء قابل کاربرد مجدد

---

[1] بسیاری از مباحث مرتبط (نظیر حوزه کاری، پیچیدگی مسئله، ساختار تیمی و اندازه و طول زمان انجام پروژه و فناوری مورد استفاده) بر بهره وری پروژه تاثیر گذار خواهند بود

گرچه نقاط ساختاری مجدداً قابل کاربرد هستند، اما هزینه‌های واجدالشرایط کردن، انطباق، یکپارچه‌سازی، نگهداری کم اهمیت نیستند. قبل از اقدام به کاربرد مجدد، مدیر پروژه بایستی از هزینه‌های مربوط به استفاده از نقاط ساختاری مطلع باشد.

از آن‌جایی که تمامی نقاط ساختاری[2] (و در کل اجزاء قابل استفاده مجدد) یک پیشینه قبلی دارند(پیشتر در پروژه‌ای ساخته شده‌اند. مترجم) می‌توان اطلاعات هزینه‌ای در ارتباط با هر یک را، جمع‌آوری کرد. سپس این اطلاعات را می‌توان برای بدست دادن هزینه‌های برآورد شده در مورد بعدی مورد تحلیل و بررسی قرار داد.

به‌عنوان مثال، درنظر بگیرید که برنامه کاربردی جدید X مستلزم ۶۰ درصد برنامه جدید و کاربرد مجدد سه نقطه ساختاری SP_3 , SP_2 , SP_1 می‌باشد. هر یک از این اجزاء قابل کاربرد مجدد در برخی کاربردهای دیگر به‌کار رفته‌اند و هزینه‌های متوسط واجد شرایط‌سازی، انطباق، یکپارچگی و نگهداری در دسترس هستند.

جهت برآورد تلاش لازم در ارائه X، بایستی مقدار زیر را تعیین کرد:

$$\text{تلاش کل} = E_{new} + E_{qual} + E_{adapt} + E_{int}$$

که در آن:

E_{new}: تلاش لازم برای مهندسی و ساخت اجزاء جدید نرم‌افزار

(که با استفاده از فنون توصیف شده در فصل ۵، تعیین می‌شود.)

E_{qual} : تلاش لازم برای واجد شرایط کردن SP_3 , SP_2 , SP_1

E_{adapt}: تلاش لازم جهت تطبیق SP_3 , SP_2 , SP_1

E_{int}: تلاش لازم برای یکپارچگی SP_3 , SP_2 , SP_1

تلاش مورد نیاز برای واجد شرایط کردن، تطابق و یکپارچگی اجزاء قابل کاربرد مجدد در سایر برنامه‌های کاربردی تعیین می‌کرد.

پرسش

محاسبه سریعی نیز دارد که ما را در سازد، برآوردی از زینه‌فایده استفاده جدد اجزاء داشته شیم؟

۲۷-۶-۳ متریک های استفاده مجدد

در تلاش برای اندازه‌گیری مزایای کاربرد مجدد در یک سیستم کامپیوتری مجموعه‌ای از متریک‌های نرم‌افزاری توسعه یافته‌اند.

مزیت مرتبط با کاربرد مجدد در داخل سیستم s را می‌توان به‌صورت زیر بیان کرد:

$$R_b(s) = [C_{noreuse} - C_{reuse}] / C_{noreuse}$$

که در آن :

$C_{noreuse}$: عبارت است از هزینه توسعهٔ s بدون کاربرد مجدد.

1.Chistensen,S.R.

2 . شاید بتوان نقطه ساختاري را امتیاز ساختاري نیز ترجمه نمود، تا خوانندگان را چه مقبول افتد. مترجم

C_{reuse} : هزینه توسعه با ایجاد s با کاربرد مجدد می‌باشد.

بنابراین می‌توان $R_b(S)$ به‌صورت یک ارزش غیربعدی در دامنه

$$0 \le R_b(S) \le 1 \qquad (۲-۲۷)$$

بیان کرد.

دوانبو و همکارانش [DEV95] پیشنهاد می‌کنند که ۱) R_b تحت‌تأثیر طراحی سیستم خواهد بود (۲) از آن‌جا که R_b تحت‌تأثیر طراحی می‌باشد، مهم است که R_b بخشی از ارزیابی مشتقات طراحی باشد و (۳) مزایای مرتبط با کاربرد مجدد همسوئی نزدیکی با منفعت هزینه‌ای هر یک از اجزاء قابل استفاده مجدد دارند.

سنجش کلی استفادهٔ مجدد در سیستم‌های شیءگرا با نام «تأثیر زیاد کاربرد مجدد» [BAN94] به‌صورت زیر تعریف می‌شود:

$$R_{lev} = OBJ_{reused} / OBJ_{built}$$

که در آن:

OBJ_{reused} تعداد اشیاء مجدداً به‌کار رفته در یک سیستم است.

OBJ_{built} تعداد اشیاء ساخته شده برای یک سیستم است.

در نتیجه، همان‌طور که R_{lev} افزایش می‌یابد، R_b نیز زیاد می‌شود.

۷-۲۷ خلاصه

مهندسی نرم‌افزار مبتنی بر اجزاء، مزایای ذاتی را در کیفیت نرم‌افزار، بهره‌وری توسعه‌دهنده و هزینهٔ کل سیستم ارائه می‌کند. و هنوز قبل از کاربرد وسیع الگوی CBSE در صنعت، موانع زیادی را باید از سر راه برداشت.

علاوه بر اجزاء نرم‌افزاری، مجموعه‌ای از اشیاء قابل کاربرد مجدد می‌تواند توسط مهندس نرم‌افزار فراهم‌گردد. و اینها عبارتند از نمایش‌های تخصصی نرم‌افزار (مثل مشخصات، مدل‌های معماری و طراحی‌ها)، اسناد داده‌های آزمون و حتی وظایف مرتبط با فرآیند (مانند فنون بازرسی).

روند CBSE شامل دو فرآیند فرعی همزمان است ـ مهندسی قلمرو و توسعهٔ مبتنی بر اجزاء.هدف از مهندسی قلمرو، شناسایی، ساخت، فهرست کردن و انتشار مجموعه‌ای از اجزاء نرم‌افزاری در یک قلمرو خاص کاربردی است. توسعهٔ مبتنی بر اجزاء نیز این اجزاء را برای کاربرد در یک سیستم جدید، واحد شرایط کرده، انطباق داده و یکپارچه می‌نماید. به‌علاوه، توسعهٔ مبتنی بر اجزاء، اجزاء جدیدی را که براساس نیازمندیهای سفارشی سیستم جدید هستند، مهندسی می‌کند.

فنون تحلیل و طراحی برای اجزاء قابل کاربرد مجدد مبتنی بر همان اصول و مفاهیم هستند که بخشی از روش‌های مناسب مهندسی نرم‌افزاری می‌باشند. اجزاء قابل استفادهٔ مجدد بایستی در محیطی

طراحی شوند که ساختارهای داده‌ای استاندارد، پروتوکل‌های رابط و معماری‌های برنامه هر حوزهٔ کاربردی را تعیین می‌کنند.

مهندسی نرم‌افزار مبتنی بر اجزاء برای ساخت برنامه‌های کاربردی، از مدل تبادل اطلاعات، ابزارها، ذخیرهٔ ساخت یافته و یک مدل شیءگرای زیربنایی استفاده می‌کند. به‌طور کلی مدل شیءگرا با یک یا چند استاندارد اجزاء (مثل OMG/ CORBA) مطابقت دارد که شیوهٔ دستیابی برنامهٔ کاربردی به اشیاء قابل استفادهٔ مجدد را تعیین می‌کند. طرح‌های طبقه‌بندی به توسعه‌دهنده امکان می دهد تا اجزاء قابل کاربرد مجدد را یافته و بازیابی کند و با مدلی که مفهوم، محتوا و بافت را شناسایی می‌کند، تطبیق یابد. طبقه‌بندی شمارشی، طبقه‌بندی طرح‌دار طبقه‌بندی ارزش ـ ویژگی نمایندهٔ بسیاری از طرح‌های طبقه‌بندی اجزاء هستند.

صرفه اقتصادی کاربرد مجدد نرم‌افزاری با یک سؤال مطرح می‌شود: آیا ساخت کمتر و کاربرد مجدد بیشتر مقرون به‌صرفه است؟ به‌طور کلی جواب مثبت است اما طراح پروژهٔ نرم‌افزار می‌بایستی هزینه‌های قابل توجه در رابطه با واجد شرایط سازی، انطباق و یکپارچگی اجزاء قابل استفاده مجدد را، مدنظر قرار دهد.

مسایل و نکاتی برای تفکر و تعمق بیشتر

۲۷-۱ یکی از کلیدهای (اسنادی) اصلی برای استفاده مجدد، وادار کردن توسعه دهندگان به استفاده مجدد از اجزاء و بخشهای موجود به جای ساختن دوباره انواع جدید، است. (پس از آن ساختن چیزهای جدید جالب است!) سه یا چهار راه مختلف پیشنهاد دهید که نشان دهد سیستمهای نرم افزاری قادر به تشویق مهندسین نرم افزار به استفاده مجدد می باشند. چه فنون و شگردهایی برای پشتیبانی از تلاشهای استفاده مجدد باید به وجود آیند؟

۲۷-۲ اگرچه اجزاء نرم افزاری بیشترین اجزاء مصنوعی قابل استفاده مجدد هستند، موارد دیگرتهیه شده توسط مهندسی نرم افزار را نیز می توان دوباره استفاده نمود. پروژه ها و طرحها و قیمتهای برآورد شده را بررسی کنید. چگونه می توان از آنها استفاده مجدد نمود و فواید انجام این کار چیست ؟

۲۷-۳ تحقیقی در خصوص مهندسی میدان (حوزه) انجام داده و مدلی (بیرونی (خطوط بیرونی)) را برای شکل مثال ۲۷-۱ طراحی کنید. متنی را معرفی کنید که برای تجزیه حوزه و گسترش سبک معماری نرم افزار مورد نیاز است.

۲۷-۴ عملکرد خصوصیات و ویژگیهای عملکرد میدان و اجزاء و رده بندیهای طرح های آن در کجا یکسان و در کجا متفاوتند؟

۲۷-۵ گروهی از ویژگیهای میدان برای سیستمهای اطلاعاتی را که مناسب برای دانشجویان پردازش داده ها است، شرح (و بسط) دهید.

۲۷-۶ گروهی از ویژگیهای میدان که برای پردازش کلمه / انتشار رمینه نرم افزاری مناسب شرح دهید.

۲۷-۷ مدل ساختاری داده ای برای عملکرد میدان که توسط استادتان اختصاص یافته یا حالتی که برای خودتان آشناتر است را شرح دهید.

۲۷-۸ موضع (هدف) اصلی ساختار چیست؟

۲۷-۹ اطلاعاتی را در مورد تازه ترین استانداردهای CORBA یا COM یا JAVABEAN به دست آورده و ۳ تا ۵ صفحه در مورد مباحث اصلی تهیه نمایید. اطلاعاتی در زمینه موضوعات ابزارهای مورد نیاز واسطه ها به دست آورده و توضیح دهید که چگونه این ابزارها به حالت استاندارد می رسند؟

۲۷-۱۰ رده بندی هایی را برای عملکرد میدان که توسط استادتان اختصاص داده شده، یا نوعی را که برای خودتان آشناتر است را، شرح دهید.

۲۷-۱۱ شکلی از طرح رده بندی را برای عملکرد میدان که توسط استادتان اختصاص داده شده یا برای خودتان آشناتر است شرح دهید.

۲۷-۱۲ برای به دست آوردن کیفیتهای اخیر یا محصولات داده اخیر که استفاده از CBSE را پشتیبانی می کنند، تحقیقاتی را انجام دهید. این اطلاعات را برای کلاس خود آماده سازید.

۱۳-۲۷ (فرض کنید که) برآورد شده سیستم شئ گرا هنگامیکه تکمیل شده باشد برای ۳۲۰ بروژه و و موضوع مورد نیاز خواهد بود. همچنین تخمین زده شده که ۱۹۰ موضوع را می توان از مخازن موجود به دست آورد. شیوه به کار بردن (نیرو، قدرت نفوذ) استفاده مجدد چیست؟

فرض کنید که موضوعات جدید هرکدام ۱۰۰۰ دلار ارزش دارد و هزینه وفق دادن و درست کردن آن موضوع ۲۰۰ دلار می باشد و برای کامل کردن هرکدام ۴۰۰ دلار باید هزینه کرد. هزینه تخمین زده شده برای یک سیستم چقدر است؟ ارزش R_b چقدر است؟

فهرست منابع و مراجع

[ADL95] Adler, R.M., "Emerging Standards for Component Software, *Computer*, vol. 28, no. 3, March 1995, pp. 68-77.

[BAS94] Basili, V.R., L.C. Briand, and W.M. Thomas, "Domain Analysis for the Reuse of Software Development Experiences," *Proc. of the 19th Annual Software Engineering Workshop*, NASA/GSFC, Greenbelt, MD, December 1994.

[BEL95] Bellinzona R., M.G. Gugini, and B. Pernici, "Reusing Specifications in 00 Applications," *IEEE Software*, March 1995, pp. 65-75.

[BIN93] Binder, R., "Design for Reuse Is for Real," *American Programmer*, vol. 6, no 8, August 1993, pp. 30-37.

[BRO96] Brown, A.W. and K.C. Wallnau, "Engineering of Component Based Systems," *Component-Based Software Engineering*, IEEE Computer Society Press, 1996, pp. 7-15.

[CHR95] Christensen, S.R., "Software Reuse Initiatives at Lockheed," *CrossTalk*, vol. 8, no. 5, May 1995, pp. 26-31.

[CLE95] Clements, P.C., "From Subroutines to Subsystems: Component Based Software Development," *American Programmer*, vol. 8, No. 11, November 1995.

[DEV95] Devanbu, P., et al., *II* Analytical and Empirical Evaluation of Software Reuse Metrics," Technical Report, Computer Science Department, University of Maryland, August 1995.

[FRA94] Frakes, W.B. and T.P. Pole, "An Empirical Study of Representation Methods for Reusable Software Components," *IEEE Trans. Software Engineering*, vol. SE-20, no. 8, August 1994, pp. 617-630.

[HAR98] Harmon, P., "Navigating the Distributed Components Landscape," *Cutter IT Journal*, vol. II, no. 2., December 1 998, pp. 4-11.

[HEN95] Henry, E. and B. Faller, "Large Scale Industrial Reuse to Reduce Cost and Cycle Time," *IEEE Software*, September 1995, pp. 47-53.

[HOO91] Hooper, J.W. and R.O. Chester, *Software Reuse: Guidelines and Methods*, Plenum Press, 1991.

[HUT88] Hutchinson, J.W. and P.G. Hindley, "A Preliminary Study ofLarge Scale Software Reuse," *Software Engineering Journal*, vol. 3, no. 5, 1988, pp. 208-212.

[LIA93] Liao, H. and Wang, F., "Software Reuse Based on a Large Object-Oriented Library," *ACM Software Engineering Notes*, vol. 18, no. 1, January 1993, pp. 74-80.

[LIM94] Lim, W.C., "Effects of Reuse on Quality, Productivity, and Economics," *IEEE Software*, September 1994, pp. 23-30.

[LIN95] Linthicum, D.S., "Component Development (a Special Feature)," *Application Development Trends*, June 1995, pp. 57-78.

[MCM95] McMahon, P.E., "Pattern-Based Architecture: Bridging Software Reuse and Cost Management," *Crosstalk*, vol 8, no. 3, March 1995, pp. 10-16.

[ORF96] Orfali, R., D. Harkey, and J. Edwards, *The Essential Distributed Objects Survival Guide*, Wiley, 1 996.

[PRI87] Prieto-Diaz, R., "Domain Analysis for Reusability," *Proc. COMPSAC* '87, Tokyo, October 1987, pp. 23-29.

[PRI93] Prieto-Diaz, R., "Issues and Experiences in Software Reuse," *American Pro grammer*, vol. 6, no. 8, August 1993, pp. 10-18.

[POL94] Pollak, W. and M. Rissman, "Structural Models and Patterned Architectures," *Computer*, vol. 27, no. 8, August 1994, pp. 67-68.
[STA94] Staringer, W., "Constructing Applications from Reusable Components," *IEEE Software*, September 1994, pp. 61-68.
[TRA90] Tracz, W., "Where Does Reuse Start?" *Proc. Realities of Reuse Workshop*, Syracuse University CASE Center, January 1990.
[TRA95] Tracz, W., "Third International Conference on Software Reuse-Summary," *ACM Software Engineering Notes*, vol. 20, no. 2, April 1995, pp. 21-22.
[WHI95] Whittle, B., "Models and Languages for Component Description and Reuse," *ACM Software Engineering Notes*, vol. 20, no. 2, April 1995, pp. 76-89.
[YOU98] Yourdon, E. (ed.), "Distributed Objects," *Cutter IT journal*, vol. 11, no. 12, December 1 998.

خواندنیهای دیگر و منابع اطلاعاتی

Many books on component -based development and component reuse have been published in recent years. Allen, Frost, and Yourdon *(Component-Based Development for Enterprise Systems: Applying the Select Perspective*, Cambridge University Press, 1998) cover the entire CBSE process, using UML (Chapters 21 and 22) as the basis for their modeling approach. Books by Lim *(Managing Software Reuse: A Comprehensive Guide to Strategically Reengineering the Organization for Reusable Components*, Prentice-Hall. 1998); Coulange *(Software Reuse*, Springer-Verlag, 1998); Reifer *(Practical Software Reuse*, Wiley, 1997); and Jacobson, Griss, and Jonsson *(Software Reuse: Architecture Process and Organization for Business Success*, Addison-Wesley, 1997) address many CBSE topics. Fowler *(Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1996) considers the application of architectural patterns within the CBSE process and provides many useful examples. Tracz *(Confessions of a Used Program Salesman: Institutionalizing Software Reuse*, Addison-Wesley, 1995) presents a sometimes lighthearted, but meaningful, discussion of the issues associated with creating a reuse culture.

Leach *(Software Reuse: Methods, Models, and Costs*, McGraw-Hill, 1997) provides a detailed analysis of cost issues associated with CBSE and reuse. Poulin *(Measuring Software Reuse. Principles, Practices, and Economic Models*, Addison-Wesley, 1996) suggests a number of quantitative methods for assessing the benefits of software reuse.

Dozens of books describing the industry's component -based standards have been published in recent years. These address the inner workings of the standards themselves but also consider many important CBSE topics. A sampling for the three standards discussed in this chapter follows:

CORBA

Doss, G.M., *CORBA Networking With Java*, Wordware Publishing, 1999.
Hoque, R., *CORBA for Real Programmers*, Academic Press/Morgan Kaufmann, 1999.
Siegel, J., *CORBA 3 Fundamentals and Programming*, Wiley, 1999.
Slama, D., J. Garbis, and P. Russell, *Enterprise CORBA*, Prentice-Hall, 1999.

COM

Box, D., K. Brown, T. Ewald, and C. Sells, *Effective COM: 50 Ways to Improve Your COM- and MTS-Based Applications*, Addison-Wesley, 1999.
Kirtland, M., *Designing Component-Based Applications*, Microsoft Press, 1999.

Many organizations apply a combination of component standards. Books by Geraghty et al. (*COM-CORBA Interoperability*, Prentice-Hall, 1999), Pritchard (*COM and CORBA Side by Side: Architectures, Strategies, and Implementations*, Addison-Wesley, 1999), and Rosen et al. (*Integrating CORBA and COM Applications*, Wiley, 1999) consider the issues associated with the use of both CORBA and COM as the basis for component -based development.

JavaBeans

Asbury, S. and S.R. Weiner, *Developing Java Enterprise Applications*, Wiley, 1999.

Valesky, T.C., *Enterprise Javabeans: Developing Component-Based Distributed Applications*, Addison-Wesley, 1999.

Vogel, A. and M. Rangarao, *Programming with EnterpriseJavaBeans,JTS, and OTS*, Wiley, 1999.

A wide variety of information sources on component -based software engineering and component reuse is available on the Internet. An up-to-date list of World Wide Web references that are relevant to CBSE can be found at the SEPA Web site: **http://www.mhhe.com/eDgcs/ compsci/pressman/resources/ cbse.mhtml**