International Conference on
Engineering & Applied Sciences
UAE - DUBAI
10 March 2016
کنفرانس بین المللی
مهندسی و علوم کاربردی
امارات - دبی ۲۰ اسفند ۹۴

بررسی و مقایسه چهار روش سبک های معماری نرم افزار

سعید شیخی[۱]

۱- دانشگاه آزاد اسلامی واحد گرگان دپارتمان فنی مهندسی- گروه کامپیوتر

آدرس رایانامه : S.Sheikhi@outlook.com

۱- چکیده

با گسترش روزافزون دامنه تقاضاهای کاربران کامپیوتر و به دنبال آن اندازه سیستم های نرم افزاری، دیگر روش ها و سبک های کلاسیک تولید نرم افزار، پاسخ گوی این نیازمندی ها نبوده اند. در دنیای امروز، طراحی یک نرم افزار موفق تنها انتخاب و یا ایجاد ساختمان داده های مناسب و الگوریتم های کارآمد نیست. حجم نرم افزارهای تجاری سالهای اخیر، مهندسان نرم افزار را بر آن داشته که برای غلبه بر پیچیدگی های حاصل از این حجم بالا، به دنبال تکنیک های استفاده مجدد از نرم افزار و روش های Component-based بروند.

علی رغم تمام نکات مثبتی که در استفاده از این روش ها وجود دارد، باید به این نکته هم اشاره کرد که مشکلات و مسائل جدیدی نیز به همراه این دید نوین، پا به دنیای نرم افزار گذاشته اند: در یک سیتم نرم افزاری بزرگ که متشکل از اجزای گوناگون خواهد بود، نحوه سازماندهی و ارتباطات این اجزا با یکدیگر چگونه باشد تا نیازمندی های تعیین شده برای نرم افزار برآورده شوند ؟ پاسخ به این پرسش، وظیفه اصلی معماران نرم افزار است.

با توجه به اهمیتی که معماری نرم افزار در مهندسی نرم افزار روز دنیا پیدا کرده است، در این مقاله سعی خواهیم کرد چهار روش از روش ها و سبک های موجود و متداول معماری نرم افزار را بررسی کرده، مزایا و معایب هر کدام را بیان کنیم و به این مسئله بپردازیم که هر سبک، مناسب کدام کلاس از نرم افزارهایی است که امروزه تولید و روانه بازار می شوند.

کلمات کلیدی: معماری نرم افزار , سبک معماری , سبک های معماری نرم افزار

[۱] دانشگاه آزاد اسلامی واحد گرگان دپارتمان فنی مهندسی- گروه کامپیوتر

۱. مقدمه

امروزه تحقیق و بررسی در زمینه معماری نرم افزار به عنوان یک عنصر کلیدی در فرآیند تولید سیستم های نرم افزاری، به یک موضوع مهم تحقیقاتی تبدیل شده است. اهمیت و نیاز به ارتقاء سیستم ها، نگه داشتشان، آموزش افراد جدید و مستندات مستدل باعث پرنگ تر شدن نقش معماری نرم افزار گردیده است. امروزه بسیاری از سیستم های بزرگ تجاری موجود که در حال کار هستند و فاقد معماری نرم افزار و یا معماری مستند و قابل استفاده ای هستند، با گذشت زمان، نگهداشت آنها پر هزینه و در بسیاری از مواقع به علت تغییرات داخلی و خارجی غیر ممکن گشته است. این سیستم ها نیاز به ارتقاء و تطبیق و تغییر در ساختار خود هستند تا همچنان بتوانند به کارخود ادامه دهند. اما برای ارتقاء آنها می بایست متحمل هزینه زیادی شویم تا آنجا که در بسیاری از موارد تعویض سیستم قدیمی با یک سیستم جدید مقرون به صرفه تر از تغییرات کلی و حتی جزئی است که در سیستم قدیمی می بایست انجام دهیم. وجود معماری به عنوان راهنمایی برای نگهداشت و ارتقاء سیستم های می تواند نقشی اساسی در پایین آوردن هزینه های تغییر و ارتقاء سیستم های امروزی بازی کند. همچنین معماری نرم افزار با توجه به درک صحیحی که از مفاهیم طراحی برای ما فراهم می کند امکان تشخیص خطاهای احتمالی را به ما می دهد تا با شناخت و رفع آنها سیستم موجود را بهینه سازی کنیم.(Clements and Northrop, 2002)

۲. معماری:

معماری مجموعه ای از نقشه های فنی است که هر نقشه شامل توصیف جنبه خاصی از سیستم است .برای توصیف جنبه های مختلف معماری از یکسری مدل استفاده می شود و هرمدل از علائم، قواعد نحوی و معنایی خاصی پیروی می کند که استاندارد و شناخته شده است .

از مدلها می توان برای توصیف اجزاء یک سیستم، ارتباط بین آنها و غیره استفاده کرد. (هاشمیان ، ۱۳۸۵)

۲-۱ معماری نرم افزار:

یک معماری نرم افزار در حقیقت ساختار یک مجموعه راه حل برای فضای مسئله را توصیف می کند و در نتیجهٔ تجزیه شدن و شکستن فضای مسئله به بخشهای کوچکتر، تعیین مشخصه های عمومی و کلی هر بخش و یافتن راه حل برای مجموعه ای از بخشهایی که خصوصیات مشترک دارند و در قالب یک مؤلفه قرار می گیرند، حاصل می شود.(هاشمیان ، ۱۳۸۵)

۲-۲ تعریف معماری نرم افزار :

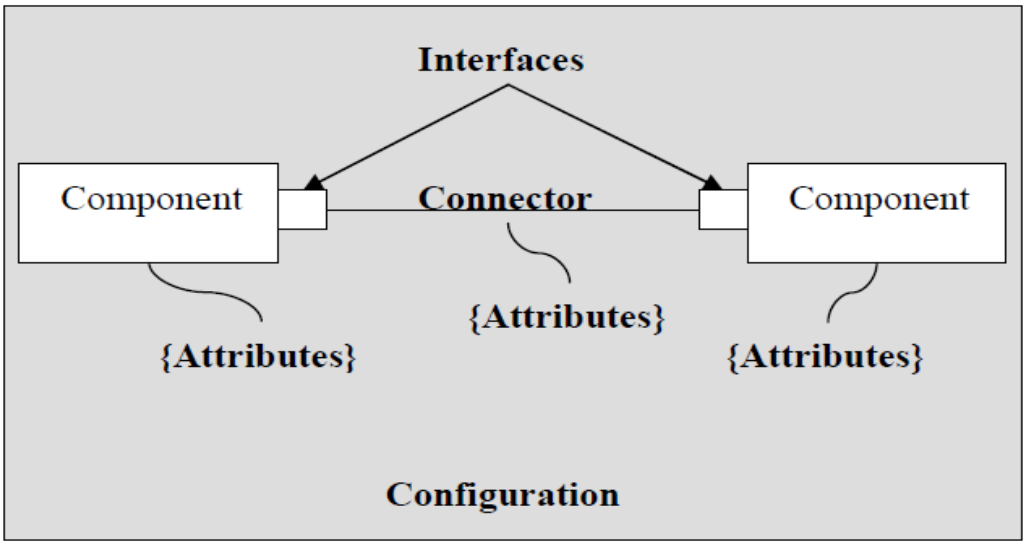
Perry and Wolf در سال ۱۹۹۲ معماری نرم افزاری را به این صورت تعریف کرده اند :

معماری نرم افزار مجموعه ای از اجزاء معماری طراحی است که شکل خاصی دارند .این اجزاء معماری ۳ دسته اند :پردازشی، داده ای و اتصالی.

Garlan and Shaw در سال ۱۹۹۴ تعریف زیر را پیشنهاد کردند :(Garlan, 1994)

معماری نرم افزار مجموعه ای است از مؤلفه ها و اتصال دهنده ها به همراه توصیف تعاملات بین آنها.

۳-۲ اجزا معماری نرم افزار :



شکل ۱. اجزای معماری نرم افزار(Kaisler, 2005)

مؤلفه :مؤلفه ها به عنوان بلوکهای اولیه و موجودیتهایی محاسباتی در ساخت سیستم شرکت کرده و از طریق محاسبات داخلی و ارتباطات خارجی خود کارها را انجام می دهند.

اتصال دهنده :اتصال دهنده ها تعاملات بین مؤلفه ها را تعریف کرده و قواعد حاکم بر آنها را توصیف می نمایند .یک اتصال دهنده درگاه های دو یا چند مؤلفه را بهم متصل می نماید.

واسط :زمانی که اتصال دهنده بین دو مؤلفه ارتباط برقرار می کند، مؤلفه یک واسط تعریف می نماید و هر مؤلفه می تواند چند واسط داشته باشد .یک واسط تنها به یک مؤلفه مربوط می شود و هر واسط یک مؤلفه، می تواند به چندین واسط در مؤلفه های دیگر وصل شود.

پیکربندی :پیکربندی که گاهی تحت عنوان توپولوژی نیز از آن یاد می شود گراف همبندی است که از مؤلفه ها و اتصال دهنده ها تشکیل شده است و ساختار معماری را توصیف می نماید.

۳. سبک های معماری

۳-۱ تعریف سبک معماری

در مورد سبک معماری تعاریف متعددی ارائه شده است که ما در ادامه به دو مورد از آنها اشاره می کنیم.

سبک معماری نرم افزار، مجموعه واژگانی متشکل از اجزای نرم افزار و نوع اتصالات میان این اجزا را به همراه مجموعه قوانینی برای ترکیب اجزا و اتصالات با یکدیگر تعریف می کند

Shaw در سال ۱۹۹۶ چنین تعریفی ارائه نمود : (Garlan, 1994)

یک سبک معماری توصیفی است از انواع مؤلفه ها و نحوه چیدمان آنها ، که شامل توصیف الگوی تعامل داده و کنترل بین مؤلفه ها است . یک سبک همچنین یک توصیف غیر رسمی از مزایا و معایب خود را در صورت استفاده از آن ارائه می نماید.

Clements در سال ۲۰۰۲ تعریف زیر را ارائه کرده است: (Clements et al , 2002)

یک سبک معماری یک تخصیص از انواع اجزاء و ارتباطات میان آنها با یکدیگر به همراه یک مجموعه قواعد و محدودیتها در مورد نحوه استفاده از آنها می باشد.

۳-۲ بررسی روش ها

۳-۲-۱ سبک جریان داده

هدف در این سبک، رسیدن به ویژگی استفاده مجدد و اصلاح پذیری است.

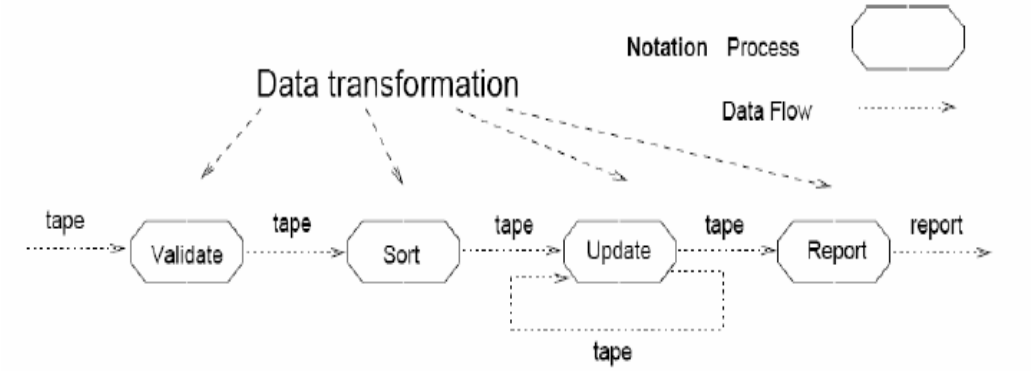
در این سبک، به سیستم به این دید نگاه می شود که یک سری از تبدیلات روی قطعاتی متوالی از داده های ورودی انجام

می شود .داده، به سیستم وارد می شود و در میان مؤلفه ها جریان می یابد . تا زمانی که به مقصد نهایی برسد و این مقصد نهایی می تواند ذخیره ای از اطلاعات، یا یک خروجی باشد.

این سبک شامل دو زیر سبک است : Batch Sequential و Pipe&Filter

در سبک Batch Sequential گام های پردازش یا همان مؤلفه ها ، برنامه های مستقلی هستند و فرض بر این است که هر گام قبل از شروع گام بعد، اجرایش به تکامل می رسد .هر دسته ای از داده ها، بین گام ها انتقال داده می شوند .فرم کلی برنامه در این سبک، فرم کلاسیک پردازش داده است. (هاشمیان ، ۱۳۸۵)

شکل ۲ این سبک را نشان می دهد.



شکل ۲. سبک Batch Sequential (Clements and Kazman, 2003)

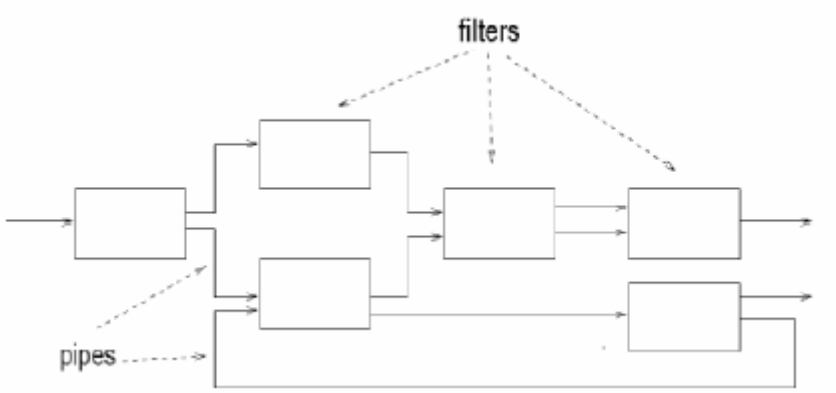
سبک pipe&filter تأکید روی تبدیل و پردازش تدریجی روی داده بوسیله مؤلفه های متوالی دارد .این سبک یک سبک متداول در خانواده سیستم های عامل UNIX است.

Filter ها در این سبک همان مؤلفه ها هستند و داده ها را به طور تدریجی تبدیل می کنند. Pipe ها همان اتصال دهنده ها هستند و حالتی به خود نمی گیرند و فقط برای حرکت دادن، بین Filter ها قرار می گیرند.

قواعدی که روی این سبک حاکم اند، نحوه وچگونگی بستن pipe ها و فیلترها به هم را مشخص می کنند.

یک pipe یک source End دارد که به درگاه خروجی یک Filter متصل است و یک Sink End که به درگاه ورودی یک Filter متصل است.(هاشمیان ، ۱۳۸۵)

شکل ۳ یک شمای کلی از یک نمونه از این سبک را نشان می دهد.



شکل ۳. سبک pipe&filter (Land, 2001)

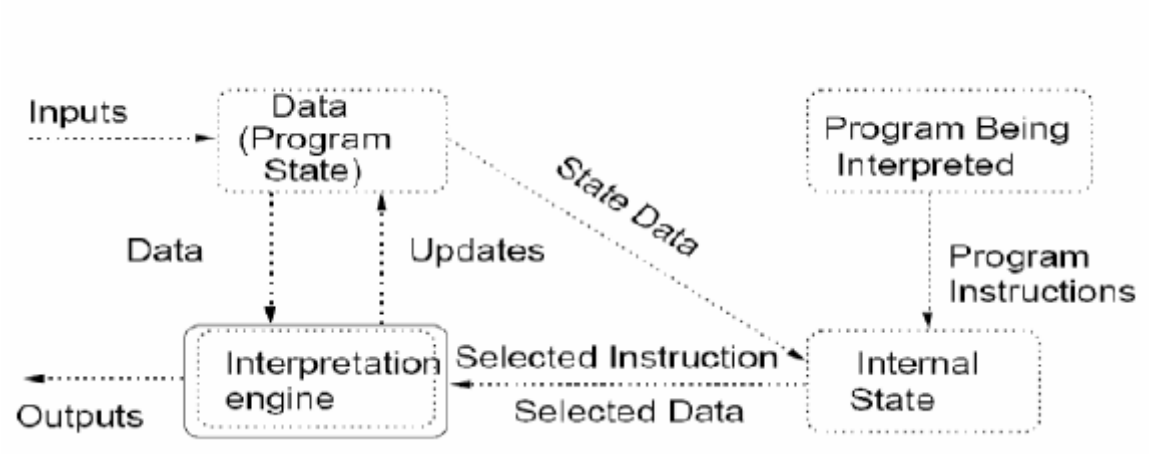
۳-۲-۲ سبک ماشین مجازی

در این سبک معماری، هدف اصلی رسیدن به قابلیت حمل بالا است .این سبک، عملکردی را که برای سخت افزار یا نرم افزاری که برنامه روی آن اجرا می شود، غیر بومی است، شبیه سازی می کند .اینکار به روشهای مختلفی انجام می شود.

این سبک می تواند platform هایی که هنوز ساخته نشده اند را شبیه سازی کند، (مثل سخت افزاری جدید) و می تواند حالت های بدی که ممکن است در عمل بسیار پیچیده، پر هزینه و خطرناک باشند شبیه سازی نماید .(مثل سیستم های بحرانی / امنیت یا سیستم های پرواز)

مثال های متداول از این سبک شامل :مفسرها، سیستم های قاعده مند و پردازشگرهای زبان های فرمانی می باشند .مثلا زبان جاوا، روی ماشین مجازی جاوا اجرا می شود که اجازه می دهد که زبان جاوا مستقل از platform باشد.(هاشمیان ، ۱۳۸۵)

ماشین مجازی ساختاری به شکل ۴ دارد.



شکل ۴. سبک ماشین مجازی(Clements and Kazman, 2003)

در این شکل سه نوع داده موجود می باشند: برنامه در حال تفسیر شدن، داده های برنامه (مثل مقادیری که در حین اجرای برنامه به متغیرها داده شده است) و حالات داخلی مفسر (مثل مقادیر ثبات ها و یا دستورات در حال اجرا). موتور تفسیر، دستوری را از برنامه در حال تفسیر انتخاب می کند، حالات داخلی خود را بهنگام می کند و بنابر دستور، قاعدتًا داده های برنامه را بهنگام می سازد.

اجرای برنامه توسط مفسر، انعطا ف پذیری یا وفق پذیری را همراه با قابلیت داشتن وقفه و پرس وجو از برنامه و معرفی تغییرات در زمان اجرا به همراه می آورد ولی بخاطر داشتن محاسبات اضافی در حین اجرا ، کارایی کاهش میبابد.(هاشمیان ، ۱۳۸۵)

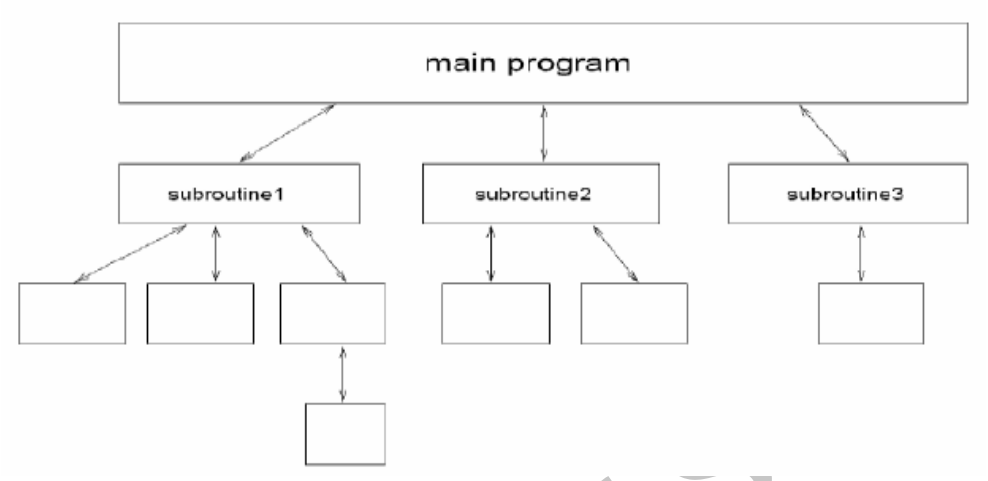
۳-۲-۳ سبک فراخوانی و بازگشت :

هدفی که در این سبک دنبال می شود، رسیدن به اصلاح پذیری و مقیاس پذیری است .این سبک، معماری غالب سیستم های نرم افزاری بزرگ در ۳۰ سال گذشته بوده، ولی از این سبک، زیر سبک هایی با خصوصیت های ارزشمند، پدید آمدند که در ادامه تعدادی از آنها بررسی می شوند .(هاشمیان ، ۱۳۸۵)

۱-۳-۲-۳ سبک برنامه اصلی و زیر روال :

این سبک همان نمونه برنامه سازی به روش کلاسیک را نشان می دهد .هدف، تکه کردن برنامه به قطعات کوچک تر، برای رسیدن به اصلاح پذیری بالاتر است .یک برنامه به صورت سلسله مراتبی تقسیم می شود و یک رشته واحد از کنترل است و هر مؤلفه در این سلسله مراتب، این کنترل را از پدر خود بدست می آورد و به فرزندان خود می دهد .جریان کنترل می تواند همراه با جریان داده هم باشد . (هاشمیان ، ۱۳۸۵)

شکل ۵ شمای کلی این سبک را نشان می دهد



شکل ۵. سبک برنامه اصلی و زیر روال (Clements and Kazman, 2003)

۳-۲-۳-۱-۱ سبک سیستمهای فراخوانی روالهای خارجی

سیستم های مبتنی بر این سبک، سیستم های با سبک" برنامه اصلی و زیر روال "هستند که به تکه هایی تقسیم می شوند که آن تکه ها روی کامپیوتر هایی که از طریق شبکه ای به هم متصل شده اند، قرار می گیرند .هدف در این سبک، بالا بردن کارایی، بوسیله توزیع محاسبات و استفاده از چندین پردازشگر است

.در این سیستم ها، انتساب واقعی بخشها به پردازشگرها تا زمان اجرا به تأخیر می افتد و این به آن معنی است که انتساب ها در جهت بالا بردن کارایی براحتی می توانند تغییراتی داشته باشند.

می توان گفت، این سبک هیچ فرقی با سیستم با سبک برنامه اصلی و زیر روال استاندارد ندارد مگر اینکه فراخوانی زیر روا ل ها و انجام عملیات آنها در صورتی که آن تابع روی یک ماشین خارجی باشد زمان بیشتری صرف خواهد کرد.(هاشمیان ، ۱۳۸۵)

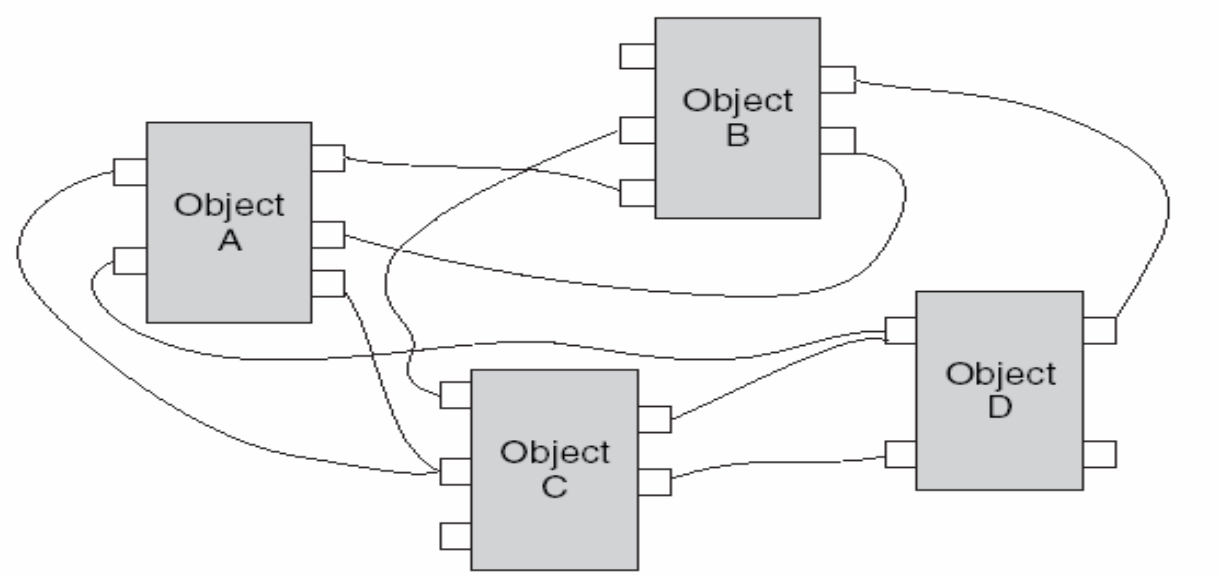
۳-۲-۳-۲ سبک سیستم های شی گرا

این سبک، نسخه مدرنی از سبک های فراخوانی و بازگشت می باشد .نمونه شی گرا، تکیه بر بسته بندی داده و دانشی در مورد چگونگی انجام عملیات و دستیابی بر داده ها دارد. بسته بندی شامل محصورسازی داده ها و پنهان سازی رموز داخلی داده ها از محیط است.

دستیابی به شیء، فقط از طریق عملیاتی خاص، موسوم به متد، مقدور است .محصورسازی قابلیت استفاده مجدد و اصلاح پذیری را بالا می برد و این بخاطر این است که طی این مسیر مقوله ها از هم جداسازی می شوند .مثلا کاربر یک سرویس نیاز به دانستن اینکه سرویس چگونه کار می کند، ندارد .خاصیت اصلی که در واقع وجه تمایز نمونه شی گرا و انواع داده های مجرد است، وراثت و چند ریختی است.

هنگامی که تجریدهایی ازشیء، یک مؤلفه می سازند که سرویس های Black box را تامین کند و باقی مؤلفه ها از آن دسته اول استفاده کنند، سبک Call-based Client-server بوجود می آید.(هاشمیان ، ۱۳۸۵)

شکل ۶ شمای کلی این سبک را نشان می دهد.



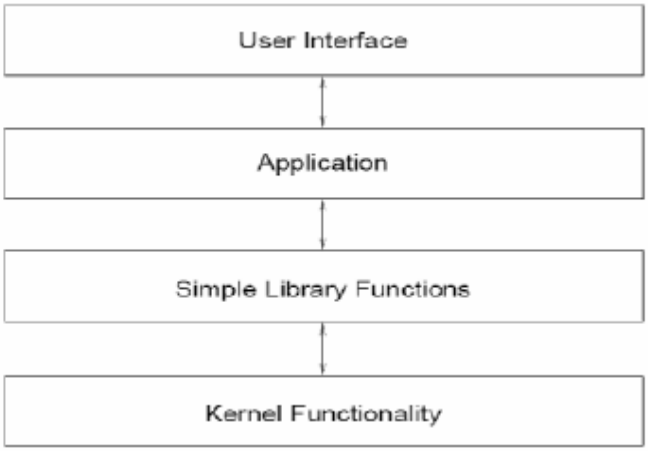
شکل ۶. سبک سیستم های شی گرا(Kaisler, 2005)

۳-۳-۲-۳ سبک سیستم های لایه ای

در این سبک، مؤلفه ها هر یک به لایه های انتساب داده می شوند تا محاورات بین مؤلفه ها کنترل شود .در مدل محض از این زیر سبک، هر لایه فقط با لایه های مجاور خود ارتباط دارد .هدف در این سبک رسیدن به اصلاح پذیری و قابلیت حمل است .پایین ترین لایه، عملیات هسته ای و مرکزی (مثل سخت افزار یا هسته سیستم عامل) را فراهم می کند .هر لایه بالاتر روی لایه قبلی ساخته می شود و لایه پایینی اش را پنهان می سازد و سرویس هایی را که توسط لایه های بالاتر مورد استفاده اند، فراهم می کند.

در عمل، اغلب سیستم های لایه ای، محض نیستند و عملیات در یک لایه امکان ایجاد ارتباط با عملیات هر لایه ای را دارند .این امکان "Layer Bridging" نامیده می شود و هنگامی مورد استفاده قرار می گیرد که مسأله کارایی در زمان اجرا مورد نظر باشد.(هاشمیان ، ۱۳۸۵)

شمای کلی سبک سیستم های لایه ای در شکل ۷ آورده شده است.



شکل ۷. سبک سیستم های لایه ای(Clements and Kazman, 2003)

۳-۲-۴ سبک مؤلفه های مستقل

این سبک، برای رسیدن به اصلاح پذیری، بوسیله جداسازی بخش های مختلف محاسباتی است .این سبک از فرایندهای مختلف که با ارسال پیام به هم، کارمی کنند تشکیل شده است .این فرایندها به هم داده می فرستند ولی یکدیگر را مستقیماً کنترل نمی کنند .پیام ها می توانند به یک مشترک مشخص فرستاده شوند یا در حالت استفاده از سیستم بر پایه رویداد که از نمونه publish/subscribe استفاده می کند، می تواند بین مشترک های غیر مشخص مبادله شود.

سیستم های بر پایه رخداد، زیر سبک هایی هستند که در آنها، کنترل بخشی از مدل است. هر مؤلفه ای ، داده هایی را که می خواهد با محیطش تسهیم کند، اعلام می دارد (عمل Publish) و بقیه مؤلفه ها می توانند در این کلاس داده ، مقداری را ثبت کنند.(subscribe)

اگراین کار را بکنند هر زمان که داده ای پدیدار شد، فراخوانی می شوند و داده را دریافت می کنند. در این روش مدیریت پیام ممکن است مؤلفه هایی که به آنها پیام ارسال می کند، کنترل نکند. مؤلفه ها مقادیری را ثبت می کنند که می خواهند دریافتشان کنند و سپس پیام را به مدیریت پیام می دهند تا پیام و یا مرجع شیء را به بقیه مشترکین متقاضی بدهد.

این زیر سبک برای این مهم است که هر مؤلفه را از شناسایی مؤلفه های دیگر بی نیاز می کند .همچنین مؤلفه ها می توانند بطور موازی کار کنند و فقط در زمان نیاز با هم مقادیری از داده را رد و بدل کنند .این تجزیه ، یکپارچگی مؤلفه ها را تسهیل می کند. فرایندهای مرتبط در سیستم های چند پردازند ه ای کلاسیک (مثل مشتری/خدمتگذار) زیر سبک دیگری از سبک اصلی مطرح شده است که در آن هدف اصلی رسیدن به مقیاس پذیری بالا است. .(هاشمیان ، ۱۳۸۵)

خدمتگذار می تواند داده را به یک یا بیشتر مشتری بدهد، مشتری به خدمتگذار تقاضا می دهد، اگر خدمتگذار سنکرون عمل کند، در همان زمان که داده را می دهد، کنترل را به مشتری برمی گرداند و اگر آسنکرون عمل کند فقط داده را به مشتری می دهد.(هاشمیان ، ۱۳۸۵)

۴. مقایسه چهار روش

اکنون بعد از معرفی این چهار سبک معماری نرم افزار می توانیم به مقایسه انها بپردازیم و مزایا و معایب هر کدام را بیان کنیم.

جدول۱. جدول مقایسه جهار سبک معماری نرم افزار

نام روش	مزایا	معایب
سبک جریان داده	- رفتار کل سیستم به صوت ترکیب ساده ای از رفتار فیلترها مدل می شود. - به دلیل استقلال فیلترها از هم، توسع سیستم با سهولت انجام می شود. - استفاده مجدد از نرم افزار توسط این سبک پشتیبانی می شود. - سبک لوله ها و فیلترها به طور طبیعی از اجرای موازی کارها پشتیبانی می کنند - اصلاح پذیری	- به دلیل آنکه نرم افزار هایی که با این سبک ساخته می شوند ذاتاً به تبدیلات متنوع و مکرر داده ها بستگی دارند، معمولاً این سبک برای نرم افزارهای محاوره ای انتخاب مناسبی نیست. - افت کارایی
سبک ماشین مجازی	- انعطاف پذیری - قابلیت حمل بالا	- پیچیدگی: انعطاف پذیری بالای این گونه سیستم ها هزینه هایی را هم در هنگام تولید و تست به شرکت تولید کننده تحمیل می کند. کاهش کارایی به دلیل داشتن محاسبات اضافی در حین اجرا
سبک سیستم های شی گرا	- به بهترین شکل ممکن استفاده مجدد از نرم افزار را پشتیبانی می کنند. - پنهان سازی اطلاعات سبب می شود ارتباط میان اجزای نرم افزار ساده تر شود. - از مدل فرآیند گرای کلاسیک که کل سیستم را در قالب تعدادی تابع پیاده سازی می کرد فاصله گرفته و به داده ها اهمیت داده شده است. - دنیای واقعی با این سبک بهتر مدل می شود. - اصلاح پذیری بالا بوسیله محصورسازی	- بر خلاف مدل هایی نظیر لوله ها و فیلترها، در این سبک اشیاء برای آنکه بتوانند با یکدیگر تعامل داشته باشند، می باید از وجود هم و همچنین واسط هایی که ارائه می کنند، باخبر باشند. به این ترتیب اشیاء به یکدیگر وابستگی پیدا می کنند.
سبک سیستم های لایه ای	- استفاده مجدد از نرم افزار در این سبک پشتیبانی می شود. - ساده بودن فرایند طراحی - اصلاح پذیری - قابلیت حمل - به دلیل modularity بالا بهبود نرم افزار، در این روش به سادگی انجام می شود.	- تمام سیستم های نرم افزاری را نمی توان به سهولت به لایه هایی تقسیم کرد و این امر می تواند کاربرد این سبک را تا محدود به موارد خاص کند. - افت کارایی در برخی مواقع - منتقدین این سبک معتقدند لایه بندی یک سبک نیست و تنها مشخصه ای طبیعی از سیستم هایo با ساختار سلسله مراتبی است.
سبک مولفه های مستقل	- استفاده مجدد از نرم افزار - قابلیت گسترش - مقیاس پذیری بالا - اصلاح پذیری بوسیله جداسازی بخش های مختلف محاسباتی	- کارایی: به دلیل ارتباط میان پردازه ای هزینه ای از کارایی سیستم باید پرداخت کنیم که این تا حد زیادی به نوع ارتباطات اجزا با هم بستگی دارد. - پیچیدگی: استفاده از این سبک باعث پیچیده تر شدن نرم افزار هم می شود.

۵. نتیجه گیری

در این مقاله ، ما به بررسی چهار سبک از سبک های معماری نرم افزار پرداختیم و سعی کردیم توصیف تقریبا کاملی از هر یک از این روش ها ارائه بدهیم .

از آنجایی که امروزه از سبک های مختلفی استفاده می شود ، ما در این مقاله در صدد آن بر آمدیم که اطلاعات مناسبی از این چهار سبک انتخابی را ارائه دهیم و به بیان هدف ها و اصول آنها بپردازیم .

و در آخر برای کامل کردن این بررسی مزایا و معایب هر کدام از این سبک ها را بیان کردیم تا تصویر کامل تری از هر یک از سبک ها مورد بحث ارائه داده باشیم .

مراجع

۱. سمینار کارشناسی ارشد، دانشگاه آزاد اسلامی، سبک های معماری نرم افزار، حسین هاشمیان ۱۳۸۵
2. Clements, P. & Northrop, L. Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002.
3. M. S. David Garlan, An Introduction to Software Architecture, (January 1994).
4. S. H.Kaisler, Software Paradigms, John Wiley & Sons, 2005.
5. F. B. Paul Clements, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford, Documenting Software Architectures: Views and Beyond, Addison Wesley, September 27, 2002.
6. L. B. Paul Clements, Rick Kazman, Software Architecture in Practice, Addison Wesely, April 11,2003.
7. R. Land, Architecture Solutions in PAM, 2001.