تمرین های درس معماری نرم افزار

استاد : دکتر علی رضایی

محمد فرزانفر- زمستان 95

1	موسسه SEI (Software Engineering Institute) چیست ؟

سازمانهایی مانند وزارت دفاع آمریکا، دولتی غیرنظامی و صنایع برای رسیدن به اهداف و ماموریتهایشان نیاز به نرم افزارهای مطمئن که منطبق بر وظیفه تعریف شده و بدون آسیب پذیری کار کنند هستند.

سیستمهای نرم افزاری برای توسعه خود و افزایش اثرگذاری و ایجاد شرایط رقابتی با یکدیگر ارتباط برقرار می کنند، پیچیدگی و ارتباطات داخلی این سیستمها، خطر حملات سایبری و افزایش ریسک خرابی ها را زیاد کرده است. بیش از سه دهه، موسسه مهندسی نرم افزار به دولتها و شرکتهای صنعتی در ایجاد، توسعه، کاربری و نگهداری از سیستمهای نرم افزاری خلاق، مورد اعتماد و قدرتمند کمک نموده است.

مطالعات و تحقیق در زمینه مشکلات حاد نرم افزاری و امنیت سایبری و متعاقباً ایجاد و آزمایش تکنولوژی های خلاقانه و تبدیل آن به راهکارهایی برای استفاده وسیع از آن از دستاوردهای این موسسه است.

2	تفاوت نمونه سازی (prototyping) اکتشافی (Exploratory) و تکاملی(Evolutionary)

نمونه سازی مجموعه فعالیت هایی است که برای ایجاد یک یا چند نمونه از برنامه نرم افزاری انجام می شود. یک نمونه معمولاً یک یا چند مورد یا انتظارات خاص از محصول نهایی را شبیه سازی می کند. لیست اولیه ای از نیازهای کاربران مشخص می شوند و نمونه ای از سیستم ساخته می شود و سپس میتوان آن را در چندین مرحله بر اساس بازخوردهایی که از کاربران گرفته می شود تکمیل کرد.

نمونه سازی اکتشافی (Exploratory) : برای انجام آزمایشات کوچک بر روی فرضیات کلیدی پروژه در زمینه وظیفه مندی ها، تکنولوژی یا هردو ایجاد می شوند. ممکن است به کوچکی قطعه کد چند صد خطی برای تست عملکرد یک از اجزاء کلیدی نرم افزاری یا سخت افزاری سیستم باشد. یا برای شفاف سازی برداشت توسعه دهندگان و استفاده کنندگان از رفتار یا الزامات فنی ساخته می شود. طراحی این نوع نمونه ها بسیار سریع و غیررسمی بوده و بعد از استفاده کاربردی ندارند.

نمونه سازی تکاملی : (Evolutionary) این نمونه سازی از یک تکرار به تکرار بعدی کاملتر می شود. برخلاف کیفیت اولیه محصول، با تکمیل شدن محصول بر روی کدها کارم مجدد صورت می گیرد. برای مدیریت این کارها نیاز است که طراحی ها و تست ها حتی در مراحل اولیه بصورت رسمی انجام شود و هر چه سیستم تکمیل تر می شود طراحی ها و تست ها نیز شکل رسمی تری به خود می گیرند.

3	دیدگاه 4+1

مدل 4+1 توسط فیلیپ کروچن برای توصیف معماری سیستمهای نرم افزاری معرفی شد. این مدل مبتنی بر استفاده از چند view است. View ها برای توصیف سیستم از دید مصرف کنندگان و ذینفعان نرم افزار است. مانند کاربران نهایی، برنامه نویسان و مدیران پروژه. در این مدل 4 View زیر و نیز Use Case برای تشریح معماری استفاده می شود.

دید Logical : دید منطقی برای نمایش ویژگیهای وظیفه مندی و قابلیت هایی است که سیستم برای کاربران نهایی فراهم می سازد. برای این دید از Class Diagram و Package Diagramاستفاده می شود.

دید Process : دید فرآیند، درگیر صفات کیفی و پویای سیستم است. فرایندهای سیستم و نحوه تعامل آنها با یکدیگر را توصیف می کند. بر رفتار سیستم در زمان اجرا تمرکز دارد. شامل نمودارهای Sequence, Communication و Activity و Timing است.

دید Implementation or Development : برای تشریح سیستم از دید برنامه نویس است و در گیر مدیریت و تنظیمات نرم افزار است. برای توصیف این دید از نمودارهای Component استفاده می شود.

دید Deployment or Physical : اید دید در لایه فیزیکی ، نودها و ارتباطات فیزیکی آنها با کامپوننت ها را بررسی می کند،

ارتباط کامپوننت ها با لایه فیزیکی ، اینکه چه کامپوننتی روی چه بستر نرم افزاری یا سخت افزاری قرار گیرد از درگیری های این دید است. نمودار Deployment برای این دید استفاده می شود.

دید : UseCase دید سناریو، توصیف معماری با استفاده از مجموعه ای از usecase ها ، این سناریوها شامل موارد کاربردی است که رفتار سیستم را از نگاه کاربران و ذینفعان آن بیان می کند. این دید چهار دید دیگر را در ارتباط با هم نگه میدارد و دلیلی است برای نتایج و شکل گیری هر یک از دیدهای دیگر. با کمک این دید هست که نیازمندی های اصلی و مهم سیستم در قالب سناریوها تعریف می شود.

4	تعریف مفاهیمSequence Diagram – Deployment Diagram

نمودار Class Diagram : نمودار ساختاری ایستا است که ساختار سیستم را با نمایش کلاسهای سیستم، خصوصیات و روابط بین آنها توصیف می کند. هم برای مدلسازی مفهومی و هم برای مدلسازی طراحی جزئیات برای ترجمه به کد برنامه نویسی بکار میرود. یک کلاس نمایشگر یک موجودیت از یک سیستم معین است، کلاس دارای ویژگیهایی است که خصایص یکتای آن را بیان می کنند از سه بخش نام، خصوصیات و متدها تشکیل شده است.

نمودار UseCase : این نمودار نشان دهنده روابط بین کنشگر ها و موارد کاربرد است. این نمودار رفتار سیستم را از دید ناظر بیرونی نشان میدهد ناظر بیرونی میتواند یک شخص، یک سیستم اطلاعاتی یا یک وسیله سخت افزاری باشد.

نمودار Sequence Diagram : برای نشان دادن جریان عملیات در یک UseCase برحسب زمان استفاده می شود ، زمانی مفید است که بخواهیم روند منطقی یک سناریو را دید. نشان می دهد که چگونه اشیاء با یکدیگر در قالب پیامهایی متوالی ارتباط برقرار می کنند و همچنین نمایشگر طول عمر اشیاء نسبت به این پیامها است.

نمودار Component : چگونگی تقسیم سیستم به مولفه های آن و وابستگی بین مولفه را توصیف می کند.

نمودار : Deployment سخت افزار بکاررفته در پیاده سازی سیستم و همچنین محیط های اجرا و سایر اجزایی که باید بر روی این سخت افزار قرار گیرند را توصیف می کند.

5	تعریف و تفاوت های GIT وSVN

هر دو سیستمهای کنترل ورژن هستند، این امکان را می دهند که افراد مختلف بتوانند بصورت همزمان روی فایلهای مشترک کار کنند و تغییرات همزمانی روی آنها اعمال کنند.

مهمترین تفاوت های Git و SVN :

سیستم GITبصورت توزیع شده و SVN متمرکز است. Git نسخه متمرکز نیز دارد اما از مزیتهای آن این است که هر کس می تواند نسخه غیربرخط خود را داشته باشد و از همه امکانات Branching و Commit کردن تغییرات خود استفاده کند و در زمان اتصال به سرور اصلی این تغییرات را بر روی آن بروزرسانی کند و در ضمن Repository محلی خود را نیز بروزرسانی کند در حالیکه SVN این قابلیت Offline بودن را ندارد و حتما باید بصورت Online روی سرور متمرکز کارها انجام شود. همه برنامه های کنترل ورژن صرفا اطلاعات مربوط به فرادادە را در فولدرهای خاصی نگهداری می کند در حالیکه Git تمامی محتوای روی سرور را بطور کامل روی دستگاه مقصد دانلود می کند.

6	Metamorphic Testingچیست ؟

رهیافتی است برای تولید موارد تست جدید بر اساس مجموعه تست های انجام شده . که هدف آن پیداکردن نقص های سیستم که در تست های اصلی بروز نکرده اند می باشد. اگر به یک تابع شناخته شده ورودی بدهیم خروجی آن تولید می شود. صفات تغییرپذیر یک تابع این امکان را می دهد که ورودی های جدیدی را تولید کنیم که با توجه به ویژگی های صفات می توان خروجی آن را پیش بینی کرد. اگر خروجی تولید شده با خروجی پیش بینی شده اختلاف قابل توجهی داشته باشد نقص یا عیبی در

سیستم وجود دارد. این روش تست به صورتهای مختلفی بسط داده شده است. برای نمونه در برنامه های غیرقابل تست که داده هایی برای پیش بینی نتیجه ندارند، این روش می تواند حتی بدون در نظر گرفتن درست یا غلط بودن خروجی سیستم، وجود اشکال در سیستم را شناسایی کند. این روش برای برنامه های یادگیری ماشین، داده کاوی، جستجو و شبیه سازی که در تحلیل داده های حجیم مرسوم هستند ایجاد شده است.

7	Spring Framework چیست؟

چهارچوب Spring محبوب ترین چهارچوب کاری توسعه نرم افزارهای سطح بالا به زبان جاوا است. از این چهارچوب برای تولید نرم افزارهایی با کارایی بالا، ساده جهت تست و قابل استفاده مجدد از کدهای آن، استفاده می کنند.
این چهارچوب بسیار ساده و کم حجم جهت نصب و انتقال به روی سیستم ها می باشد.
از قابلیت های اصلی چهارچوب Spring می توان جهت توسعه نرم افزارهای مختلف برروی پلتفرم جاوا استفاده نمود
اسپرینگ ماژولار است و براساس نیاز می توان ماژولهای مورد نیاز را انتخاب و کنار هم گذاشت. این چهارچوب همه چیز را از اول ایجاد نکرده است بلکه نحوه استفاده از تکنولوژی های موجود را ساده تر می کند و بهبود می بخشد. با استفاده از Dependency Injection و Inversion Of Control (What To Do/When To Do) وابستگی بین اشیاء را در کد کاهش می دهد.

8	مدل Furps چیست؟

برای سهولت در روند تحلیل نیازمندیها چک لیستی ایجاد شده که یادآوری می کند چیزی را از قلم نیاندازیم. این چک لیست را به اختصار FURPS می گوییم که به شرح ذیل است.
حرف F برای Functional Requirements : همان نیازمندیهای وظیفه مندی (کارکردی) که می تواند شامل امکانات ،قابلیتها وامنیت باشد.
حرف U برای Usability : مانند مستندات راهنما، مستندات آموزشی ، که میتواند شامل فاکتورهای انسانی، زیبایی ، توافق و سازگاری در واسط کاربر، راهنمای برخط و حساس به متن، مستندات کاربر و اصول آموزش باشد.
حرف R برای Reliability : برنامه ریزی برای آنچه که در زمان از کار افتادن سیستم باید انجام دهیم. مثلا اگر بانک اطلاعاتی از کار بیفتد چگونه سرویس دهی به کاربران قطع نشود. که میتواند شامل تکرار و شدت خطاها، قابلیت ترمیم، قابلیت پیشبینی، صحت و درستی و متوسط زمان رخداد خطا باشد.
حرف P برای Performance : سیستم به چه حجمی از کاربران باید پاسخ دهد. زمان مناسب پاسخ دهی سیستم چقدر باید باشد. شرایطی را از قبیل سرعت،کارایی، در دسترس بودن، صحت و درستی، توان عملیاتی، زمان پاسخ، زمان ترمیم و بهره وری منابع را برنیازمندیهای کیفی تحمیل میکند.

حرف S برای : Supportablity که میتواند شامل قابلیت تست، قابلیت توسعه پذیری ، قابلیت انطباق ، قابلیت نگهداری، سازگاری ، ، قابلیت نصب قابلیت تعمیرپذیری - پشتیبانی سیستم پس از اجرا به چه نحوی خواهد بود. آیا به سیستمهای جانبی دیگر برای پشتیبانی نیاز داریم، برای مثال آیا برای کمک به کاربران سیستم ، یک سیستم FAQ طراحی شده است؟

9	سرویس choreography را بررسی نمایید.

در Orchestration یک المان اصلی و مرکزی است که تمامی اجزاء یک فرآیند را کنترل می کند مانند یک گروه موسیقی که یک رهبر، یکپارچگی اعضا گروه را مدیریت می کند.
در Choreography به گونه ای است که هر المان بصورت خود مختار وظایف خود را کنترل می کند مانند یک گروه رقص که هر فرد نقش خود را در تیم می داند و قدم هایش را با صدای موسیقی هماهنگ می کند. ممکن است Choreography نیز از یک مرکز کنترل استفاده کند، اما در کل سازمان به نسبت موقعیت می توانند از هر دو رویکرد استفاده کنند.

در Orchestration متداولترین رویکرد برای استفاده در فرآیندهای کسب و کار و ترکیب سرویس است. در رویکرد Orchestration شما سلسله مراتبی از مراحل فرآیند، شرایط و قوانین تعریف می کنید. سپس با یک کنترل کننده ی مرکزی فرآیند را اجرایی می کنید. در معماری سرویس گرا این مراحل توسط اجرای سرویس بصورت ترتیبی مشخص صورت می گیرد. ترتیب های اجرا می توانند با تکنیک های متفاوتی انجام شوند.

در Choreography رویکردی متفاوت است که برای سناریو های شامل فرآیندهای پیچیده، سیستم های مبتنی بر رخداد و مبتنی بر عامل مناسب است. در رویکرد Choreography قوانینی وضع می شوند که رفتار هر بخش از فرآیند را بصورت جداگانه مشخص می کنند. در این رویکرد رفتار کلی فرآیند از طریق برقراری ارتباط بین زیر بخش های آن حاصل می شود، هر بخش بصورت خود مختار تنها طبق قوانین خود عمل می کند. دو رویکرد اصلی برای پیاده سازی وجود دارد: مبتنی بر پیغام(Message Component) و مبتنی بر اجزاء کاری (Work Component).

در رویکرد مبتنی بر پیغام تمرکز بر پیغام هایی است که بین بخش های فرآیند جابه جا می شود. با این رویکرد شما پیغام هایی که بصورت قراردادی بین بخش های فرآیند جابه جا می شوند را با تمام جزئیات می توانید تعریف کنید. این مکانیسم با استاندارد (Web Services Choreography Description Language(WS-CDL پشتیبانی می شود و بیشتر در برنامه های کاربردی بین دو کسب و کار (B۲B) کاربرد دارد. در برنامه های کاربردی B۲B (که چندن سازمان به یکدیگر متصل می شوند) مشخص کردن نحوه ی کارکرد یک سیستم در سازمان دیگر کاری دشوار است زیرا نمی توان جریان کاری بین سازمان ها را بصورت کلی مدل کرد (در بسیاری از موارد یک مجوز مرکزی برای انجام این کار وجود ندارد) این رویکرد بسیار جذاب است چون شما برای برقراری ارتباط با یک سازمان کافی است ساختار پیغام ها را مشخص کنید. (Syntax، Semantics و Behavior).

رویکرد دوم برای پیاده سازی، تنظیم اجزاء کاری فرآیند است. در این رویکرد شما رفتار هر جزء کاری را تعریف می کنید و زمانی که هر نمونه از فرآیند اجرا می گردد هر جزء رفتار مرتبط با آن نمونه فرآیند را پیاده می کند. به عنوان مثال شما می توانید قوانین زیر را برای اجزاء کاری فرآیند پیاده سازی کنید: چه کارهایی باید انجام شود تا یک جزء کاری به پایان برسد، چه اشخاصی می توانند به سیستم دسترسی داشته باشند و غیره.

به طور خلاصه Choreography :

- رفتار کلی فرآیند از بخش های درون آن نشأت می گیرد (رویکرد پایین به بالا) و نیازی به یک دید یکپارچه بصورت کلی از فرآیندها نیست.
- فرآیندهای پیچیده به کارهای کوچک تر با دستورالعمل مجزا تقسیم می شوند و هر بخش دستورالعمل خود را کنترل می کند.
- قابل نگاشت به سیستم های مبتنی بر عامل و رخداد است.
- معمولاً راه اندازی آن ها ساده نیست ولی برای تبدیل به فرآیندهای پیچیده تر راحت تر هستند.
- می توان از فرآیند یک شمای گرافیکی تهیه کرد. به طور مثال: مدل جریان عملگرها

10	پروتکل های SOAP و REST را بررسی نمایید.

تعریف Rest: یک سبک معماری و رویکردی برای ارایه سرویس ها و API های عمومی روی بستر اینترنت با استفاده از پروتکل Http می باشد. Rest روی دسترسی به منابع آدرس دار با یک رابط ثابت متمرکز است یعنی اینکه هر منبعی دارای URI مشخص و واحدی است. مفهوم Rest روی عملیات CRUD بر روی داده استوار است. برقراری ارتباط کلاینت با Rest ساده بوده و نیاز به تنظیمات خاصی ندارد، همچنین فرمت های متنوع JSON و XML را ارایه می دهد.

تعریف SOAP : یک پروتکل انتقال پیام می باشد که به برنامه های در حال اجرا روی سیستم های مختلف اجازه تعامل و تبادل

پیام می دهد. SOAP بیشتر بر روی منطق ارایه برنامه بصورت سرویس متمرکز شده است. SOAP روی ارایه عملیات متنوع برای منطق کسب و کار بنا نهاده شده است، فرمت پیامها XML و ساختار خاص خود را دارد. وقتی کلایت ها و سرورها روی محیط وب و موبایل کار می کنند و نیازی به ارسال اطلاعات اضافی در مورد شی به سمت کلاینت نداریم REST گزینه مناسبی است. اما وقتی به یک Contract رسمی مابین سرور و کلاینت نیاز داریم و یا نیاز به تراکنش های ACID در فراخوانی سرویس ها داریم SOAP مناسب تر است.

11	فرق sporadic و stochastic چیست؟

الگوی ورود رویداد می تواند بصورت Periodic یا Sporadic در نظر گرفته شود. یک الگوی ورود رویداد دوره ای ممکن است هر 10 میلی ثانیه اتفاق بیفتد. الگوی ورود آماری به این معنی است که ورود رویداد ها با توجه به یک توزیع احتمال است. رویدادها همچنین می توانند بصورت پراکنده یا Stochastic رخ دهند(پراکنده). یعنی اینکه الگوی ورودی قابل بیانی بر اساس مشخصات الگوهای دوره ای یا آماری نیست.

12	تعریف Dependency Injection ؟

تزریق وابستگی ها یک الگوی طراحی نرم افزار است که به ما امکان می دهد وابستگی های هارد کد شده را حذف کرده و بتوانیم در زمان کامپایل و اجرا این وابستگی ها را تغییر دهیم.
در واقع ما با تزریق وابستگی ها به دنبال توسعه کدی هستیم که کمترین وابستگی را در خود داشته باشد.

چرا بایستی کد همبستگی پایینی داشته باشد؟

- توسعه پذیری Extensibility : با افزودن یک کلاس به برنامه نیازمند تغییرات در بخش های دیگر نباشیم
- قابلیت تست پذیری Testability : فرآیند تست واحد به سهولت و بدون نیاز به در نظرگرفتن بخش های دیگر در آزمون انجام گیرد
- انقیاد پویا Late Binding : بتوان در زمان اجرا و کامپایل خدمات مورداستفاده برنامه را تغییر داد
- توسعه موازی Parallel Development : توسعه یک بخش از برنامه منوط به توسعه بخش خاص دیگری نباشد
- قابلیت نگه داری Maintainability : افزودن بخش های جدید به برنامه و تغییرات پرهزینه نشوند

13	مثالی از Maintain Multiple copy of data و Maintain multiple copy of computation

برای هر دو این موارد Sqlserver راهکار مشخصی ارایه داده است برای نسخه های متعدد از محاسبات و فرایندها ، Always On Failover Cluster Instances و برای نگهداری کپی های متعدد از اطلاعات روشهای Always On Availability Groups و Database mirroring و Log shipping را پیشنهاد می کند.