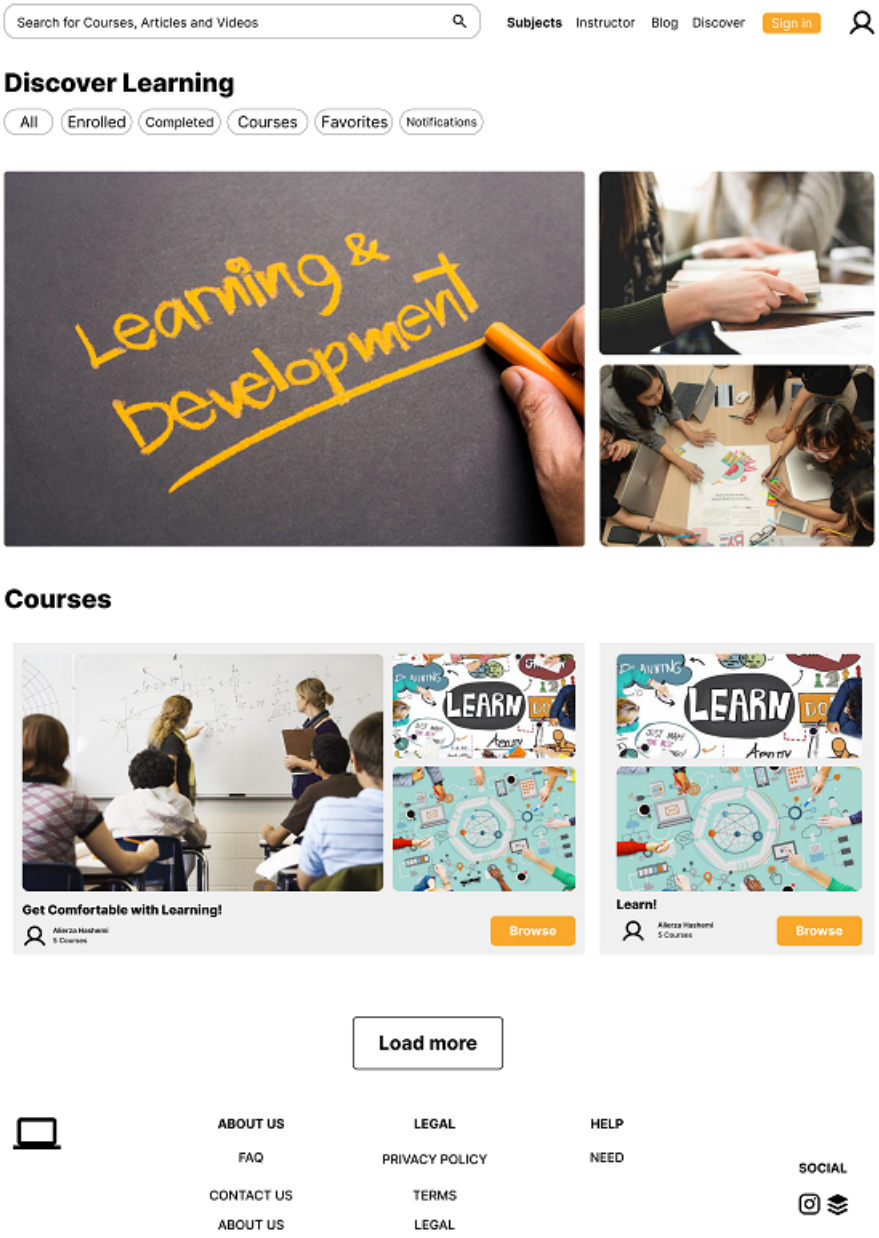
نام اعضای تیم و شماره دانشجویی‌ها

سید ابوالحسن رضوی (۴۰۲۲۱۲۶۵۵)
ایمان محمدی (۹۹۱۰۲۲۰۷)
علی اسلامی نژاد (۴۰۲۲۱۱۷۸۹)
شماره گروه: ۲۰

جواب سوال ۱

برای این سوال، طراحی در فیگما صورت گرفته است. لطفا به این لینک مراجعه کنید.



Search for Courses, Articles and Videos

Popular Top

Discover Learning

All | Enrolled | Completed | Courses | Favorites | Notifications

Template Text

Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text
Template Text Template Text

Button | Button | Button

Alireza Hashemi
5 Courses

Alireza Hashemi
5 Courses

Buy | Use

جواب سوال ۲

جواب این سوال می‌تواند خیلی طولانی باشد و با بررسی نکات مختلف هر کدام از امکان‌سنجی‌ها، اما تلاش می‌کنم به شکل خلاصه جواب دهم به هر کدام از سوالات.

۱ بررسی امکان‌سنجی

۱.۱ امکان‌سنجی فنی

امکان‌سنجی فنی به ارزیابی توانایی فنی سازمان برای توسعه پروژه می‌پردازد. این بررسی شامل تحلیل زیرساخت‌های موجود، دانش فنی تیم، و تکنولوژی‌های لازم است. در واقع سوال آیا می‌توانیم بسازیم را در این‌جا جواب می‌دهیم.

ریسک‌ها:

- *کمبود دانش فنی:* اگر تیم توسعه دانش کافی راجع به تکنولوژی‌های جدید نداشته باشد، پروژه ممکن است با شکست مواجه شود.
- *محدودیت‌های زیرساختی:* زیرساخت‌های ناکافی می‌تواند مانعی برای پیاده‌سازی موفق پروژه باشد.

راه حل‌ها:

- برای کمبود دانش فنی، برنامه‌ریزی دوره‌های آموزشی و کارگاه‌های تخصصی پیشنهاد می‌شود. خیلی وقت‌ها این مورد رو با استخدام کارشناسان به‌خصوص و متخصص در زمینه‌های مختلف نیز بهبود می‌دهند و این مشکل را حل می‌کنند.
- برای محدودیت‌های زیرساختی، به‌روزرسانی و ارتقاء زیرساخت‌ها قبل از شروع پروژه ضروری است. در این بخش نیاز به بودجه‌ی مالی دقیق و همچنین برنامه‌ریزی درست برای هزینه‌های زیرساختی خواهد بود.

۲.۱ امکان‌سنجی اقتصادی

امکان‌سنجی اقتصادی به بررسی صرفه اقتصادی پروژه از جنبه‌های هزینه و سودآوری می‌پردازد. در واقع، در این بخش به سوال آیا باید این پروژه را انجام دهیم می‌پردازیم و جواب این سوال را می‌دهیم. گام‌های این سنجش بدین شکل هستند:

۱- شناخت سودها و زیان‌های سیستم

۲- اعطای مقدار به آنان

۳- مشخص ساختن جریان مالی

۴- مشخص ساختن برآیند ارزش فعلی

۵- مشخص ساختن بازگشت روی سرمایه

۶- مشخص ساختن نقطه سرشکن

۷- نمایش نموداری نقطه سرشکن

ریسک‌ها:

- *هزینه‌های پیش‌بینی نشده:* هزینه‌های غیرمنتظره می‌تواند برآوردهای اولیه را نامعتبر کند.
- *تخمین نادرست سود:* بیش‌برآورد کردن سود می‌تواند منجر به تصمیم‌گیری‌های نادرست شود.

راه حل‌ها:

- برای مقابله با هزینه‌های پیش‌بینی نشده، ایجاد یک بودجه احتیاطی توص یه می‌شود.
- برای جلوگیری از تخمین نادرست سود، انجام تحلیل‌های بازار دقیق‌تر و استفاده از داده‌های تاریخی مشابه برای پیش‌بینی‌های واقع‌بینانه توصیه می‌شود.

۳.۱ امکان‌سنجی سازمانی

امکان‌سنجی سازمانی به ارزیابی تطابق پروژه با اهداف و استراتژی‌های کلی سازمان می‌پردازد. در واقع به سوال اگر بسازیم آیا سرمایه‌گذاران می‌آیند این‌جا پاس می‌دهیم. به‌خصوص باید بررسی کنیم هم‌سو هستش اهداف پروژه با اهداف کسب و کار روزمره یا خیر.

ریسک‌ها:

- *مقاومت در برابر تغییر:* کارکنان ممکن است نسبت به تغییرات ناشی از پروژه مقاومت نشان دهند.
- *ناهماهنگی با اهداف سازمانی:* پروژه ممکن است با اهداف بلندمدت سازمان هماهنگ نباشد.

راه حل‌ها:

- برای مقابله با مقاومت در برابر تغییر، برگزاری جلسات توجیهی و ایجاد ارتباطات موثر با کارکنان پیشنهاد می‌شود.
- برای اطمینان از هماهنگی با اهداف سازمانی، باید از مراحل اولیه برنامه‌ریزی، اهداف پروژه با اهداف کلی سازمان مطابقت داده شود.

۲ محاسبه سود و زیان پروژه "رندان"

۱.۲ تحلیل سود

Calculation	Definition	Formula
Present Value (PV)	The amount of an investment today compared to that same amount in the future, taking into account inflation and time.	$\frac{\text{Amount}}{(1 + \text{interest rate})^n}$ n = number of years in future
Net Present Value (NPV)	The present value of benefit less the present value of costs.	PV Benefits − PV Costs
Return on Investment (ROI)	The amount of revenues or cost savings results from a given investment.	$\frac{\text{Total benefits} - \text{Total costs}}{\text{Total costs}}$
Break-Even Point	The point in time at which the costs of the project equal the value it has delivered.	$\frac{\text{Yearly NPV*} - \text{Cumulative NPV}}{\text{Yearly NPV*}}$ *Use the Yearly NPV amount from the first year in which the project has a positive cash flow. Add the above amount to the year in which the project has a positive cash flow.

در این عکس، تعاریف هر کدام از موارد مختلف این بخش را می‌بینیم.

برای تحلیل سود پروژه "رندان"، ابتدا با استفاده از داده‌های فروش پیش‌بینی شده توسط مالک محصول و رشد پیش‌بینی شده توسط مدیر مالی، سود خالص هر سال محاسبه می‌شود. سپس، با احتساب سود مازاد حاصل از فروش افزونه‌ها، جمع کل سودهای هر سال به دست می‌آید.

۲.۲ تحلیل زیان

در بخش زیان، هزینه‌های مرتبط با نگهداری سرورها و حقوق کارکنان محاسبه شده و به عنوان هزینه‌های عملیاتی در نظر گرفته می‌شود. همچنین، با توجه به افزایش حقوق سالانه کارکنان، هزینه‌های مرتبط با نیروی انسانی برای هر سال پیش‌بینی می‌شود.

۳.۲ محاسبه بازگشت سرمایه و نقطه سر به سر

بازگشت سرمایه (ROI) با مقایسه کل سود حاصل از پروژه با کل هزینه‌های پروژه محاسبه می‌شود. نقطه سربه سر (BEP)، زمانی را نشان می‌دهد که در آن کل درآمدها با کل هزینه‌ها برابر می‌شوند و پروژه شروع به سوددهی می‌کند. برای محاسبه این دو شاخص، ابتدا باید ارزش فعلی خالص (NPV) و سپس ROI و BEP بر اساس آن محاسبه شود.

محاسبه NPV : NPV با استفاده از نرخ بهره مورد انتظار و جریان‌های نقدی پیش‌بینی شده برای پروژه محاسبه می‌شود. این شاخص به ما می‌گوید که ارزش کل پروژه با توجه به زمان و ارزش زمانی پول چقدر است.

محاسبه ROI : ROI با تقسیم کل سود خالص به دست آمده از پروژه (پس از کسر هزینه‌ها) بر کل سرمایه‌گذاری اولیه (هزینه‌های پروژه) محاسبه می‌شود. این شاخص نشان‌دهنده درصد بازگشت سرمایه از پروژه است.

محاسبه BEP : BEP زمانی را نشان می‌دهد که در آن درآمد حاصل از پروژه برای اولین بار هزینه‌های انجام پروژه را پوشش می‌دهد. این نقطه از نظر زمانی نشان‌دهنده شروع سوددهی پروژه است.

۴.۲ نتیجه‌گیری

محصول	سال سوم	سال دوم	سال اول	فروش محصول
۷۲۷۵	۳۱۵۰	۲۶۲۵	۱۵۰۰	فروش محصول
۲۹۱	۱۲۶	۱۰۵	۶۰	فروش افزونه‌ها
۷۵۶۶	۳۲۷۶	۲۷۳۰	۱۵۶۰	جمع سودها
۶۱۳۵/۴	۲۴۶۱/۳	۲۲۵۶/۱	۱۴۱۸	ارزش لحظه‌ای سودها (Present Value)

جدول ۱: جدول سودها؛ مقادیر عددی به واحد میلیون تومان هستند.

محصول	سال سوم	سال دوم	سال اول	حقوق افراد
۷۶۲۵	۳۱۲۵	۲۵۰۰	۲۰۰۰	فروش محصول
۲۵۰	۰	۰	۲۵۰	نگهداری سرورها
۷۸۷۵	۳۱۲۵	۲۵۰۰	۲۲۵۰	جمع زیان‌ها
۶۴۵۹/۳	۲۳۴۷/۸	۲۰۶۶/۱	۲۰۴۵/۴	ارزش لحظه‌ای زیان‌ها (Present Value)

جدول ۲: جدول زیان‌ها؛ مقادیر عددی به واحد میلیون تومان هستند.

در جدول بالا تمامی مفاهیم و فرمول‌های لازم برای این سوال آمده است. اما به جهت شفاف‌سازی به ارائه بررسی کوتاهی از مفاهیم موجود می‌پردازیم.

منظور از **Present Value** یا ارزش فعلی، دخیل کردن فاکتور زمان و مواردی اقتصادی چون تورم، سود و ...می‌باشد چرا که با در نظر گرفتن این موارد، ارزش دلار امروز با ارزش دلار فردا برابر نمی‌باشد.

منظور از **Return on Investment (ROI)** یا بازگشت سرمایه، به معنای محاسبه سود کلی سازمان و دخیل کردن ضررها و سرمایه‌گذاری‌ها در کنار سودها می‌باشد و در انتها نیز مقدار این تفریق دوباره تقسیم بر کلیه هزینه‌ها معمولا به صورت سالانه و یا پروژه محور صورت می‌گیرد. مشکل این معیار عدم در نظر گرفتن وضعیت مالی در سازمان در بازه‌های کوتاه مدت می‌باشد و نمی‌تواند به تنهایی نشان‌گر وضعیت مالی پروژه باشد.

بازگشت سرمایه در این پروژه برابر مقدار زیر می‌باشد:

$$\frac{(۷۵۶۶ - ۷۸۷۵)}{۷۸۷۵} = -۰/۰۳۹$$

مفهوم **Break Even Point (BEP)** نیز به معنای نقطه‌ای است که میزان سود سازمان با ضررها و سرمایه‌گذاری‌های صورت گرفته در آن برابر می‌شود. با توجه به اهمیت عنصر زمان در این معیار لازم است که از **Present Value (PV)** استفاده کنیم. نهایی حاصل تفریق **Net Present Value (NPV)** با مجموع **NPV** سالانه (در اولین سالی که روند مالی شرکت مثبت شده است) می‌باشد.

با توجه به مقادیر موجود در نمودارهای ارائه شده (و عدم برخورد نمودارهای سود و هزینه)، پروژه مذکور در سوال در حدفاصل سه ساله به نقطه **BEP** نمی‌رسد.

جواب سوال ۳

جواب این سوال خیلی گسترده هستش، من سعی می‌کنم خیلی خلاصه و با بیان خلاصه‌ترین حالات، خواسته‌ی سوال را پیاده‌سازی کنم.

الگوهای تحلیل فاولر

الگوهای تحلیل فاولر شامل هفت الگوی اصلی است:

- **الگوی تک وظیفه (Single Responsibility Principle):** هر کلاس باید تنها یک مسئولیت مشخص را پوشش دهد.
- **الگوی بازتابیدگی (Open Closed Principle):** واحدهای نرم‌افزاری باید به گونه‌ای طراحی شوند که تغییرپذیر باشند ولی قابلیت اضافه شدن عملکرد جدید را داشته باشند.
- **الگوی جایگزینی لیسبرگ (Liskov Substitution Principle):** زیرکلاس‌ها باید در محل استفاده از کلاس‌های بالادستی خود جایگزین شوند بدون اینکه قابلیت‌های برنامه را تغییر دهند.
- **الگوی وابستگی خارجی (Dependency Inversion Principle):** واحدهای پایین‌تر باید وابسته به واحدهای سطح بالاتر باشند نه برعکس.
- **الگوی تقلیل تعداد وابستگی‌ها (Principle of Least Knowledge):** یک عنصر نباید از جزئیات پیچیده‌تر از آنچه ضروری است، آگاه باشد.
- **الگوی تقسیم مسئولیت (Principle of Seperation of Concerns):** جدا کردن مولفه‌هایی که تغییر می‌کنند از مولفه‌هایی که ثابت هستند.
- **الگوی تسطیح (Principle of Stable Dependencies):** وابستگی‌ها باید جهت مولفه‌های پایین‌تر باشند نه جهت مولفه‌های بالاتر.

این الگوها برای طراحی معماری سیستم‌های نرم‌افزاری قابل استفاده، تعمیرپذیر و تغییرپذیر مورد استفاده قرار می‌گیرند.

الگوهای طراحی غنی‌سازی یا Creational Patterns

در الگوهای طراحی گانگ اوف فاولر شامل:

- **الگوی تکیه‌گاه سازنده (Factory Method):** تعریف یک واسط برای ساخت شیء، اما انتخاب نوع شیء را به زیرکلاسها واگذار می‌کند.
- **الگوی سازنده انتزاعی (Abstract Factory):** ارائه یک و اسط برای ساخت گروهی از شیءهای مرتبط بدون مشخص کردن کلاسهای خاص.
- **الگوی سازنده تک (Singleton):** تضمین ایجاد تنها یک نمونه از کلاس و دسترسی مرکزی به آن.
- **الگوی ساختارساز ساده (Builder):** جداسازی ساخت یک شیء پیچیده از تعریف آن.

- **الگوی نمونه‌سازی اولیه‌سازی (Prototype):** استفاده از یک نمونه برای ایجاد شیء جدید با همان مشخصات.

این الگوها مکانیسم‌های غنی‌سازی را فراهم می‌کنند تا بتوان شیءها را بدون توجه مستقیم به کلاس‌های آن‌ها خلق کرد. مزایایی همچون انعطاف‌پذیری، استفاده‌پذیری و تغییرپذیری را به ارمغان می‌آورند.

مقدمه

در این سند، ما به بررسی نیازمندی‌های کلی سیستم مدیریت وظیفه می‌پردازیم و سپس چارچوب کلی برای تحلیل و طراحی را بر اساس الگوهای فاولر و GoF تعیین می‌کنیم.

تعریف برنامه‌های سیستم مدیریت وظیفه

برنامه‌های مدیریت وظیفه (Plan)

- هر برنامه شامل مجموعه‌ای از کنش‌ها یا وظایف است که باید انجام شوند.
- برنامه‌ها می‌توانند به صورت مشترک از کنش‌های مشابه استفاده کنند.
- برنامه‌ها باید بتوانند کنش‌های پیش‌نیاز را تعریف کنند تا ترتیب انجام کارها را مشخص کنند.

کنش‌ها (Action)

- کنش‌ها می‌توانند در مراحل مختلفی باشند: پیشنهادی، تایید شده، در حال اجرا، یا انجام شده.
- کنش‌ها دارای وضعیت‌های مختلفی هستند که باید در رابط کاربری نمایش داده شوند.

کاربران (Users)

- کاربران می‌توانند به کنش‌ها تخصیص داده شوند.
- کاربران باید از تغییرات مربوط به کنش‌های تخصیص داده شده مطلع شوند.

تحلیل

در مرحله تحلیل، از الگوهای برنامه‌ریزی فاولر برای تعریف روابط بین برنامه‌ها و کنش‌ها استفاده می‌کنیم.

الگوی سمت کنش‌ها

- این الگو توضیح می‌دهد که چگونه کنش‌ها می‌توانند به یک یا چند برنامه تخصیص داده شوند.
- باید برای هر کنش، وابستگی‌هایی به کنش‌های دیگر ایجاد کرد که به صورت پیش‌نیاز عمل می‌کنند، بنابراین ایجاد وضعیت‌های وابسته به یکدیگر ممکن است.

الگوی سمت برنامه‌ها

- این الگو تعیین می‌کند که چگونه برنامه‌ها می‌توانند چندین کنش را در خود جای دهند و چگونه این کنش‌ها می‌توانند به اهداف کلی برنامه کمک کنند.
- باید ساختاری طراحی کنیم که اجازه دهد برنامه‌ها به صورت موثر و کارآمد وظایف و کنش‌های مورد نیاز خود را مدیریت کنند.

طراحی با استفاده از الگوی پل و الگوی ناظر

الگوی پل (Bridge)

- بخش‌هایی از رفتار کلاس کاربر که به تخصیص کنش‌ها مربوط می‌شوند، باید به کلاس‌های مجزا تقسیم شوند تا انعطاف‌پذیری و توسعه‌پذیری بیشتری داشته باشیم.
- این کار به ما اجازه می‌دهد که تغییرات را به راحتی و بدون نیاز به تغییر در کلاس‌های اصلی کاربر اعمال کنیم.

الگوی ناظر (Observer)

- از طریق الگوی ناظر، کلاس‌هایی که از الگوی پل پیروی می‌کنند باید بتوانند تغییرات در کنش‌ها را به کاربرانی که به آن کنش‌ها تخصیص داده شده‌اند، اطلاع‌رسانی کنند.
- این سیستم اطلاع‌رسانی باید به گونه‌ای باشد که کاربران بتوانند از تغییرات وضعیت و محتوای کنش‌ها به طور فوری مطلع شوند.

الگوی حالت (State)

- اگر رفتار کنش‌ها بسته به حالت‌های مختلفی که دارند تغییر کند، باید از الگوی حالت استفاده کنیم.
- این کمک می‌کند تا رفتار مرتبط با هر کنش را بر اساس وضعیت آن مدیریت کنیم.

در هر مرحله از تحلیل و طراحی، باید توجه داشته باشیم که تمام جزئیات بر اساس نیازمندی‌های تعریف شده و درک ما از سیستم مورد نظر باشد. باید توضیحات کافی برای توجیه تصمیمات طراحی ارائه دهیم و نشان دهیم که چگونه از الگوها برای رسیدن به راه حل‌های مسئله استفاده کرده‌ایم.

جواب سوال ۴

الگوی Pipes and Filters :

هدف: این الگو به تقسیم وظایف به مدولار مستقل کمک می کند تا خطوط ارتباطی (pipes) بین آنها ایجاد کند و داده ها را از یک فیلتر به دیگری منتقل کند. این امر باعث میشود که تغییر و تعمیر هر فیلتر بدون تاثیر بر دیگران انجام شود.

مزایا:

- خوانایی و تعمیرپذیری بهتر نرم افزار
- امکان تغییر و تعمیر هر فیلتر به طور مستقل
- امکان اعمال تغییرات بدون مرور کل سیستم

معایب:

- افزایش پیچیدگی برنامه به دلیل تعداد زیاد مرزها
- نیاز به مدیریت ترافیک داده ها بین فیلترها

حوزه کاربرد پیشنهادی:

- صنعت نرم افزار، پردازش تصاویر و فیلم ها
- پردازش متون و مستندات
- پردازش داده های حساس مانند مالی و پزشکی

با توجه به الگوی Filters and Pipes می‌توان گفت:

در این الگو سیستم به قطعات کوچک موسوم به فیلترها تقسیم می‌شود که هر فیلتر وظیفه‌ای خاص را انجام می‌دهد.
فیلترها به وسیله خطوط ارتباطی یا pipes به هم متصل می‌شوند تا داده‌ها را از یک فیلتر به فیلتر دیگر منتقل کنند.
فیلترها می‌توانند عملیاتی مانند فیلتر کردن، ترکیب، تجزیه و تحلیل و ... انجام دهند.
سیستم به شکل خطی قرار می‌گیرد و داده ها از یک سو شروع می‌شود و به ترتیب از فیلتری به فیلتر دیگر می‌گذرد.
هر فیلتر مستقل است و تغییر در آن تاثیری بر دیگر فیلترها ندارد.
امکان تغییر و تعمیر هر قطعه به صورت مستقل وجود دارد.
بنابراین این الگو ساختار مدولار و جدا از هم دارد که باعث انعطاف‌پذیری و تعمیرپذیری ساده‌تر می‌شود.

الگوی Broker :

هدف: ارتباط مستقیم بین ماژولها را از طریق واسطه ای به نام بروکر کنترل و مدیریت میکند تا کارایی بالاتر و قابلیت استفاده مجدد بیشتری داشته باشند.

مزایا:

- استقلال ماژولها و جلوگیری از وابستگی متقابل
- امکان تغییر و تعمیر ماژولها بدون تاثیر بر دیگران
- انعطاف پذیری و قابلیت استفاده مجدد بیشتر

معایب:

- افزایش پیچیدگی و تاخیر در ارتباطات به دلیل وجود واسطه
- نیاز به مدیریت واسطه

حوزه کاربرد پیشنهادی:

- سیستم های توزیع شده و موازی
- سیستم هایی با ارتباطات پیچیده
- سیستم های اشتراک منابع

الگوی Broker به شرح زیر است:

در این الگو ارتباط مستقیم میان ماژول‌های مختلف جلوگیری می‌شود.

به جای آن ارتباطات از طریق یک المان مرکزی به نام Broker برقرار می‌گردد.

بروکر نقش واسطه و کنترل‌کننده تردد پیام‌ها را بین ماژول‌ها برعهده دارد.

ماژول‌ها دیگر مستقیماً با هم ارتباط ندارند بلکه پیام‌ها را عبوری از بروکر می‌فرستند.

این کار باعث کاهش وابستگی‌ها و افزایش قابلیت تغییرپذیری می‌شود.

بروکر همچنین مسئولیت‌هایی مانند اعتبارسنجی، تعیین ترتیب پیام‌ها و تنظیم منابع را برعهده دارد.

در عوض افزایش پیچیدگی و تاخیر در ارتباطات ناشی از واسطه رخ می‌دهد.

بنابراین Broker الگویی مرکزی و واسطه‌ای برای ارتباط ماژول‌ها فراهم می‌کند.

الگوی Microkernel :

هدف: جداسازی سرویس های اصلی سیستم عامل از هسته بسیار کوچک با هدف افزایش امنیت و قابلیت اطمینان.

مزایا:

- امنیت و پایداری بهتر سیستم عامل
- قابلیت تغییرپذیری و تعمیرپذیری آسان‌تر
- جلوگیری از خطاهای هسته

معایب: - کاهش عملکرد به دلیل افزایش بارگذاری بین هسته و سرویس‌ها
- پیچیدگی بیشتر مدیریت منابع

حوزه کاربرد پیشنهادی:

- سیستم عامل های توکار و صنعتی با نیاز به امنیت بالا
- سیستم عامل های سرور بزرگ
- سیستم عامل های توزیع شده

الگوی Microkernel شامل موارد زیر است:

در این الگو هسته سیستم عامل بسیار ساده می‌شود و تنها وظایف بسیار اساسی مانند مدیریت پردازنده، حافظه، رابط با سخت‌افزار را برعهده دارد.

سایر سرویس‌های سیستم عامل مانند سیستم پرونده‌ها، شبکه، چاپگر و .. در قالب سرویس‌های کاربردی به هسته خارج می‌شوند.

ارتباط بین هسته و سرویس‌های کاربردی به صورت پیام‌رسانی (Message Passing) برقرار می‌شود که باعث افزایش امنیت و جداسازی می‌گردد.

سرویس‌های کاربردی به صورت اشیاء مستقل در می‌آیند که می‌توان آن‌ها را بدون تأثیر بر سایر مؤلفه‌ها تغییر داد.

این روش سبب افزایش قابلیت اعتماد، امنیت و تعمیرپذیری می‌شود.

هرچند به دلیل افزایش پیام، عملکرد سیستم کاهش می‌یابد.

بنابراین Microkernel ساختار باز و مدولار داشته که قابلیت تغییر و تعمیر را تسهیل کرده و امنیت را افزایش می‌دهد.

جواب سوال ۵

۳ مقدمه

تکنیک‌های معماری نرم‌افزار نقش حیاتی در تعیین ویژگی‌های کیفی سیستم‌های نرم‌افزاری دارند. در این مقاله، به بررسی سه تکنیک معماری اصلی: Discovery Service و Redundant Spare و Timestamp پرداخته و تاثیر آن‌ها بر ویژگی‌های کیفی مورد بررسی قرار می‌گیرد.

۴ Discovery Service

Discovery Service یا خدمت کشف، به سیستم‌های نرم‌افزاری امکان می‌دهد تا به طور خودکار سرویس‌ها و منابع موجود را در شبکه کشف کنند. این تکنیک به ویژه در معماری‌های میکروسرویسی که در محیط‌های ابری اجرا می‌شوند، اهمیت پیدا می‌کند.

کشف سرویس و تاثیر آن بر ویژگی‌های کیفی

- **قابلیت اطمینان:** Discovery Service با ارائه مکانیزمی برای کشف خودکار سرویس‌ها و منابع، اطمینان می‌دهد که سیستم‌ها و برنامه‌های کاربردی همواره قادر به یافتن و استفاده از منابع مورد نیاز خود هستند، حتی در صورت تغییر موقعیت یا دسترسی سرویس‌ها. این امر باعث می‌شود که سیستم نرم‌افزاری در مقابل اختلالات و تغییرات زیرساختی انعطاف‌پذیر و مقاوم باشد.
- **انعطاف‌پذیری:** Discovery Service به سیستم‌ها اجازه می‌دهد تا به سرعت به تغییرات پاسخ دهند و منابع جدید را بدون نیاز به تنظیمات دستی کشف و مدیریت کنند. این تکنیک به ویژه برای سیستم‌های مبتنی بر میکروسرویس‌ها و زیرساخت‌های ابری که مدام در حال تحول هستند، ایده‌آل است و به آن‌ها امکان می‌دهد تا با کمترین وقفه به رشد و توسعه پایدار ادامه دهند..

۵ Spare Redundant

Spare Redundant به سه شکل عمده اجرا می‌شود:

active redundancy (hot spare), passive redundancy (warm spare)، spare (cold spare). این تکنیک به سیستم اجازه می‌دهد تا در صورت بروز خطا در یکی از اجزاء، به سرعت به جزء یدکی سوئیچ کند و به این ترتیب قابلیت اطمینان و دسترس‌پذیری سیستم را افزایش دهد.

Spare Redundant و تاثیر آن بر ویژگی‌های کیفی

- **قابلیت اطمینان و دسترس‌پذیری:** با استفاده از Redundant Spare ، سیستم‌ها می‌توانند در صورت بروز خطا در یک جزء، به طور خودکار به یک جزء یدکی سوئیچ کنند، بدون اینکه کاربران اختلال قابل توجهی را تجربه کنند. این تکنیک با افزایش دسترس‌پذیری و کاهش زمان توقف، به ویژه برای سیستم‌های بحرانی که نیاز به دسترسی مداوم دارند، حیاتی است.

- **انعطاف‌پذیری در برابر خطاها:**
  Redundant Spare با فراهم آوردن سطوح مختلفی از پشتیبان‌گیری (فعال، غیرفعال و سرد) به سیستم‌ها امکان می‌دهد تا با انواع مختلف خطاها به شکل موثرتری مقابله کنند. این تکنیک به سیستم اجازه می‌دهد تا حتی در شرایط نامطلوب نیز به فعالیت خود ادامه دهد و از اختلالات گسترده جلوگیری کند.

۶ Timestamp

تکنیک Timestamp برای تشخیص ترتیب غلط رخدادها، به ویژه در سیستم‌های توزیع‌شده که از پیام‌رسانی استفاده می‌کنند، به کار می‌رود. این تکنیک با اختصاص یک مهر زمانی به هر رویداد، پس از وقوع آن، به تعیین ترتیب صحیح رخدادها کمک می‌کند.

Timestamp و تاثیر آن بر ویژگی‌های کیفی

- **دقت و یکپارچگی داده‌ها:** استفاده از Timestamp به همزمان‌سازی دقیق عملیات‌ها در سیستم‌های توزیع‌شده کمک می‌کند، به طوری که هر عملیات بر اساس زمان وقوع خود به ترتیب اجرا می‌شود. این کنترل دقیق بر ترتیب رویدادها از تداخل داده‌ها و ناسازگاری‌ها جلوگیری می‌کند و به حفظ دقت و هماهنگی در سیستم‌های پیچیده کمک می‌کند.
- **همزمان‌سازی:** تکنیک Timestamp در همزمان‌سازی عملیات‌ها در سیستم‌های توزیع‌شده نقش مهمی دارد و از اجرای نادرست عملیات‌ها بر اساس ترتیب زمانی غلط جلوگیری می‌کند.

۷ نتیجه‌گیری

تکنیک‌های معماری نرم‌افزار مانند Discovery Service و Redundant Spare و Timestamp نقش مهمی در بهبود ویژگی‌های کیفی سیستم‌های نرم‌افزاری دارند. هر یک از این تکنیک‌ها به شیوه‌ای منحصر به فرد به افزایش دسترس‌پذیری، قابلیت اطمینان، دقت داده‌ها، و همزمان‌سازی کمک می‌کنند. انتخاب درست این تکنیک‌ها بر اساس نیازهای خاص هر پروژه، می‌تواند تاثیر به سزایی در موفقیت نهایی سیستم داشته باشد.

منابع:

- SEIBlog. (2021). Tactics and Patterns for Software Robustness.
- Oroumand. (2021). Intro to Service Discovery.

جواب سوال ۶

۸ مقدمه

متدولوژی Extreme Programming (XP) با تاکید بر بهبود فرآیند توسعه نرم‌افزار و ارتقاء کیفیت محصول نهایی، استراتژی‌های مختلفی را برای مرور و بازبینی کد پیشنهاد می‌دهد. این متدولوژی چابک، که بر تعاملات تیمی، رضایت مشتری و تحویل مداوم محصولات با کیفیت تاکید دارد، فعالیت‌های مروری را به عنوان بخش مهمی از فرآیند توسعه در نظر می‌گیرد.

۹ فعالیت‌های مروری در XP

فعالیت‌های مروری در XP شامل کدنویسی جفتی، توسعه محور تست (TDD) ، و بازبینی کد توسط همتیمی‌ها است. هر یک از این فعالیت‌ها به شیوه‌ای منحصر به فرد به ارتقاء کیفیت کمک می‌کند.

۱.۹ کدنویسی جفتی

کدنویسی جفتی، که در آن دو برنامه‌نویس به طور همزمان بر روی یک مسئله کار می‌کنند، از ایجاد اشتباهات جلوگیری کرده و به بهبود کیفیت کد کمک می‌کند. این فرآیند مروری، ارتباط و همفکری را در بین اعضای تیم تقویت می‌کند و به تسریع فرآیند شناسایی و رفع خطاها منجر می‌شود.

۲.۹ توسعه محور تست (TDD)

TDD یک رویکرد سیستماتیک است که در آن توسعه‌دهندگان ابتدا تست‌هایی برای ویژگی‌های نرم‌افزار می‌نویسند و سپس کدی را پیاده‌سازی می‌کنند که این تست‌ها را پاس کند. این روش موجب می‌شود که توسعه‌دهندگان تمرکز بیشتری بر روی نیازمندی‌ها و کیفیت کد داشته باشند.

۳.۹ بازبینی کد توسط همتیمی‌ها

بازبینی‌های کد توسط همتیمی‌ها فرصتی برای ارزیابی و بهبود کد از دیدگاه‌های مختلف فراهم می‌آورد. این فرآیند به اشتراک‌گذاری دانش و بهترین شیوه‌ها کمک کرده و اطمینان حاصل می‌کند که کد نوشته شده با استانداردهای تیم سازگار است.

۱۰ اثرگذاری فعالیت‌های مروری بر کیفیت

این فعالیت‌های مروری به طور مستقیم بر افزایش کیفیت نرم‌افزار تاثیر می‌گذارند. کدنویسی جفتی و TDD با کاهش خطاها و افزایش پوشش تست، اطمینان از ایجاد کد با کیفیت بالا را فراهم می‌آورند. بازبینی کد توسط همتیمی‌ها به حفظ یکپارچگی کد و اطمینان از رعایت استانداردهای تیم کمک می‌کند.

۱۱ چالش‌ها

علی‌رغم مزایای بی‌شمار، پیاده‌سازی این فعالیت‌های مروری ممکن است با چالش‌هایی همراه باشد، از جمله مقاومت در برابر تغییر، نیاز به زمان بیشتر برای اجرای فعالیت‌ها، و نیاز به آموزش برای افزایش مهارت‌های تیم.

۱۲ نتیجه‌گیری

فعالیت‌های مروری در متدولوژی XP یک عنصر حیاتی برای تضمین کیفیت نرم‌افزار هستند. با اجرای موثر این استراتژی‌ها، تیم‌های توسعه می‌توانند محصولاتی با کیفیت بالاتر و با اطمینان بیشتری تولید کنند. در نهایت، تعهد به این فعالیت‌های مروری و حل چالش‌های مربوطه، می‌تواند به ساخت نرم‌افزارهایی دوام‌پذیر، قابل نگهداری و کارآمد منجر شود.

جواب سوال ۷

مقایسه وظایف مهارت مرور در سطح دوم و سوم

سطح سوم (Entry Level)

در سطح سوم، فرد در حال یادگیری و اکتشاف ابتدایی فعالیت‌های مرتبط با مرور نرم‌افزار است. وظایف در این سطح عبارتند از:

الف) کمک به لجستیک‌های مورد نیاز برای بررسی‌ها و بازرسی‌ها، شامل:

- مدیریت لجستیک جلسه
- انجام داده‌کاوی مناسب
- تولید گزارش‌های مرتبط با جلسه

ب) استفاده از چک‌لیست‌های مناسب که توسط برگزارکننده بررسی تعیین شده‌اند.

ج) جمع‌آوری داده‌های مناسب و دقیق که توسط برگزارکننده بررسی درخواست شده است.

سطح دوم (Practitioner)

در سطح دوم، فرد مسئولیت‌های بیشتری در فرآیند مرور نرم‌افزار دارد و به طور فعال در بهبود کیفیت محصول مشارکت می‌کند. وظایف در این سطح شامل:

الف) شرکت فعال به عنوان عضو تیم بررسی برای دستیابی به اهداف فعالیت.

ب) شناسایی فرایندهای بررسی مناسب برای دستیابی به اهداف کیفیت محصول.

ج) شناسایی افراد مناسب برای مشارکت در فعالیت‌های بررسی.

د) شناسایی اقدامات اصلاحی بر اساس داده‌های بررسی برای بهبود محصول در سراسر پروژه‌ها.

ه) تجزیه و تحلیل داده‌های محصول جمع‌آوری شده برای تجزیه و تحلیل علت ریشه‌ای و ارزیابی اثربخشی بررسی.

تفاوت اصلی بین سطح سوم (Entry Level) و سطح دوم (Practitioner) در مهارت مرور نرم‌افزار، در نوع و میزان مسئولیت‌ها و نحوه مشارکت فرد در فرایندهای بررسی و بازبینی نرم‌افزار نهفته است. در سطح سوم، افراد عمدتاً در حال کسب تجربه و یادگیری اصول اولیه مرور نرم‌افزار هستند و به اجرای فرآیندها کمک می‌کنند. این سطح اغلب شامل فعالیت‌های پشتیبانی مانند آماده‌سازی مستندات، مدیریت لجستیک جلسات بررسی و جمع‌آوری داده‌های مرتبط با فرآیند است. نقش افراد در این سطح بیشتر تحت نظارت و راهنمایی افراد با تجربه‌تر انجام می‌گیرد تا با استانداردها، ابزارها و روش‌های مرور آشنا شوند.

در مقابل، در سطح دوم (Practitioner) افراد انتظار می‌رود که نقشی فعال‌تر و مستقیم‌تر در فرآیندهای بررسی داشته باشند. این شامل شرکت فعال در جلسات بررسی، شناسایی فرآیندهای بررسی مناسب، انتخاب افراد مناسب برای مشارکت در فعالیت‌های بررسی، و شناسایی و پیاده‌سازی اقدامات اصلاحی بر اساس نتایج بررسی‌ها است. در این سطح، افراد به طور فعال در تجزیه و تحلیل داده‌های مرتبط با کیفیت نرم‌افزار و ارزیابی اثربخشی فرآیندهای بررسی مشارکت می‌کنند. نقش آنها از اجرای ساده فرآیندها به سمت شناسایی فرصت‌های بهبود و اجرای تغییرات

به منظور افزایش کیفیت نرم‌افزار و اثربخشی بررسی‌ها تغییر می‌کند. این سطح از مشارکت نه تنها به بهبود مستمر فرآیندها کمک می‌کند بلکه مهارت‌ها و دانش فردی افراد را در زمینه مرور و بازبینی نرم‌افزار نیز افزایش می‌دهد.

راهنمایی برای ارتقا به سطح Technical Leader در مهارت مدیریت کیفیت نرم‌افزار

برای ارتقا به سطح چهارم، آقای رسولی باید توجه ویژه‌ای به توسعه و پیاده‌سازی استراتژی‌های پیشرفته در مدیریت کیفیت نرم‌افزار داشته باشد. اقدامات کلیدی عبارتند از:

الف) **برقراری فرهنگ کیفیت:** ایجاد یک فرهنگ سازمانی که تولید محصولات با کیفیت و پیروی از فرایندهای کیفیت را در کانون توجه قرار دهد. این شامل برگزاری کارگاه‌های آموزشی، جلسات توجیهی و فراهم آوردن انگیزه‌هایی برای تیم‌ها برای رعایت استانداردهای کیفیت است.

ب) **تعیین استانداردها و فرایندهای کیفیت:** تعریف و اجرای استانداردها، مدل‌ها و فرایندهای مدیریت کیفیت به منظور اطمینان از دستیابی به اهداف کیفیتی. این شامل شناسایی و انتخاب مدل‌های کیفیت مناسب برای پروژه‌های مختلف و اطمینان از اجرای دقیق آنها توسط تیم‌ها است.

ج) **ایجاد فرایندهای جدید:** طراحی و پیاده‌سازی فرایندهای نوآورانه برای بهبود مستمر کیفیت محصولات و فرایندها. این ممکن است شامل توسعه ابزارهای جدید برای ارزیابی و تضمین کیفیت، یا روش‌های جدید برای اجرای بازرسی‌ها و تست‌ها باشد.

د) **ارزیابی و بهبود مستمر:** انجام تجزیه و تحلیل مستمر برای ارزیابی اثربخشی فرایندهای مدیریت کیفیت موجود و شناسایی فرصت‌های بهبود. این شامل بررسی بازخوردهای دریافتی از تیم‌ها، مشتریان و سایر ذینفعان است.

ه) **طراحی ابزارها و فرایندها:** توسعه و پیشنهاد ابزارها و فرایندهای جدید که به افزایش به اهداف کیفیت محصول کمک می‌کنند. این شامل طراحی راهکارهای نرم‌افزاری خودکار برای تسهیل فرآیندهای تست و بازرسی، و همچنین ابزارهای مدیریتی برای پیگیری و گزارش‌دهی پیشرفت کیفیت می‌شود.

این اقدامات به آقای رسولی کمک خواهد کرد تا در نقش خود به عنوان یک رهبر فنی در زمینه مدیریت کیفیت نرم‌افزار پیشرفت کند و به سطح بالاتری از شایستگی دست یابد. تمرکز بر رویکردهای نوآورانه و ارزیابی مستمر فرایندها، ابزارها و روش‌ها، نه تنها به بهبود مستمر کیفیت کمک می‌کند، بلکه به توسعه‌دهندگان و تیم‌های مهندسی امکان می‌دهد تا با اعتماد به نفس بیشتری بر روی پروژه‌های خود کار کنند. نقش آقای رسولی به عنوان یک Technical Leader نیازمند درک عمیقی از اصول مدیریت کیفیت، همراه با توانایی رهبری و الهام بخشی به دیگران برای پیروی از این اصول در تمامی جنبه‌های توسعه نرم‌افزار است.