نام اعضای تیم و شماره دانشجویی‌ها

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

سوال ۱

یک قانون سرانگشتی در فاز تحلیل این است که «افراد تیم ایجاد در فاز تحلیل باید بر نیازمندی‌هایی تمرکز کنند که در حوزه‌ی مسئله و کسب و کار قرار دارد».

الف) چه نوع نیازمندی‌هایی در این حوزه‌ها نیستند؟

ب) مثال بزنید.

جواب سوال ۱

مقدمه

در فاز تحلیل مهندسی نرم‌افزار، تمرکز اصلی بر شناسایی و تعریف نیازمندی‌های کاربردی است که مستقیماً به حوزه‌ی مسئله و کسب و کار مرتبط هستند. با این حال، برخی نیازمندی‌ها وجود دارند که معمولاً در این فاز در نظر گرفته نمی‌شوند.

نیازمندی‌های غیرمرتبط

الف) **نیازمندی‌های غیرعملکردی:** این نیازمندی‌ها شامل مواردی مانند امنیت، پایداری، کارایی و استانداردهای کیفی می‌شوند. به عنوان مثال، الزامات امنیتی یا زمان پاسخ سیستم. این نیازمندی‌ها بیشتر به چگونگی ارائه سرویس توسط سیستم مربوط می‌شود تا خود سرویس

ب) **نیازمندی‌های فنی:** این‌ها شامل انتخاب‌های فناورانه مانند پلتفرم‌های سخت‌افزاری و نرم‌افزاری، زبان‌های برنامه‌نویسی و ابزارهای توسعه می‌شوند. این نیازمندی‌ها بیشتر به راه‌حل فنی برای تحقق نیازمندی‌های کاربردی مربوط می‌شوند.

ج) **نیازمندی‌های مدیریتی یا سازمانی:** این نیازمندی‌ها به فرایندهای داخلی سازمانی، رویه‌های مدیریت پروژه و سیاست‌های کلان سازمانی مربوط می‌شوند. به عنوان مثال، نیازمندی‌هایی مانند رعایت استانداردهای خاص یا روش‌های گزارش‌دهی.

مثال‌ها

- نیازمندی غیرعملکردی: در نظر گرفتن استانداردهای امنیتی بالا برای یک سیستم بانکی آنلاین که باید تراکنش‌ها را به شکل امن انجام دهد.
- نیازمندی فنی: استفاده از یک پایگاه داده خاص مانند MySQL به دلیل تجربه قبلی تیم توسعه در استفاده از این فناوری.
- نیازمندی مدیریتی: توسعه نرم‌افزار با استفاده از روش Agile به دلیل نیاز سازمان به انعطاف‌پذیری بالا و بازخورد سریع از کاربران.

سوال ۲

۱. معماری یک خانه یا ساختمان را در نظر بگیرید و با معماری نرم‌افزار مقایسه کنید.
۲. رشته‌های معماری ساختمان و معماری نرم‌افزار چه شباهت‌هایی دارند؟ چه تفاوت‌هایی دارند؟

جواب سوال ۲

مقایسه معماری خانه/ساختمان با معماری نرم‌افزار

معماری ساختمان و معماری نرم‌افزار، هر دو فرایندهای برنامه‌ریزی، طراحی و سازماندهی هستند که برای ایجاد یک محصول نهایی پیچیده و کاربردی استفاده می‌شوند. در هر دو حوزه، معمار باید مجموعه‌ای از الزامات و نیازمندی‌ها را در نظر بگیرد، راه‌حل‌های مختلف را بررسی کند، و ساختاری منطقی و کارآمد را تعریف کند.

شباهت‌ها:

- برنامه‌ریزی و طراحی: هر دو نیازمند فرایندی برای تعیین نیازمندی‌ها، محدودیت‌ها، و هدف‌های پروژه هستند.
- اصول اساسی: در هر دو حوزه، اصول اساسی مانند کارایی، پایداری، و کاربرپسندی حائز اهمیت هستند.
- توجه به جزئیات: جزئیات در هر دو حوزه نقش کلیدی در موفقیت نهایی پروژه دارند.

تفاوت‌ها:

- ماهیت محصول: محصول نهایی در معماری ساختمان فیزیکی و در معماری نرم‌افزار مجازی است.
- روند توسعه: معماری نرم‌افزار اغلب شامل فرایندهای تکراری و انعطاف‌پذیر است، در حالی که ساختمان‌ها معمولاً بر اساس طرح‌های نهایی و دقیق ساخته می‌شوند.
- تغییر و نگهداری: نرم‌افزارها معمولاً برای تغییر و به‌روزرسانی طراحی می‌شوند، در حالی که ساختمان‌ها به ندرت برای تغییرات عمده طراحی می‌شوند.

شباهت‌ها و تفاوت‌های رشته‌های معماری ساختمان و معماری نرم‌افزار

شباهت‌ها:

- تفکر سیستماتیک: در هر دو رشته، لازم است که معمار تفکر سیستماتیک داشته باشد و بتواند اجزای مختلف را به صورت یک کل هماهنگ در نظر بگیرد.
- حل مسئله: هر دو رشته به شدت بر حل مسئله و ارائه راه‌حل‌های خلاقانه تمرکز دارند.
- نیاز به همکاری و ارتباطات: در هر دو رشته، معماران نیاز به همکاری نزدیک با سایر اعضای تیم و ذینفعان دارند.

تفاوت‌ها:

- مهارت‌های تخصصی: مهارت‌های مورد نیاز در هر رشته متفاوت است؛ مهندسی نرم‌افزار به دانش برنامه‌نویسی و فناوری اطلاعات نیاز دارد، در حالی که معماری ساختمان به دانش مهندسی ساختمان و طراحی نیاز دارد.
- محیط کاری: محیط کاری و ابزارهای مورد استفاده در هر رشته متفاوت است.
- طبیعت پروژه‌ها: نوع و ماهیت پروژه‌ها در هر دو رشته به طور قابل توجهی متفاوت است.

سوال ۳

تفاوت فعالیت های تحلیل و طراحی سیستم های نرم‌افزاری را توضیح دهید. اطمینان حاصل کنید که در توضیحات خود به موارد زیر بپردازید:

- ارتباط آن دو با یک مساله و راه حل آن
- اهداف و تمرکز هر یک
- سطح انتزاع هر کدام
- تقدم و تاخر هر یک از این دو فعالیت
- تفاوت مدل‌سازی ذیل هر فعالیت

جواب سوال ۳

ارتباط با مسئله و راه‌حل

تحلیل: در تحلیل نرم‌افزار، مسئله مورد بررسی قرار می‌گیرد. هدف این است که دقیقاً تعریف کنیم مسئله چیست و چه نیازهایی باید توسط نرم‌افزار برآورده شود.
طراحی: در مرحله طراحی، راه‌حل‌های ممکن برای مسائل تحلیل شده مطرح می‌شوند. این مرحله شامل تعیین چگونگی عملکرد نرم‌افزار برای برآورده کردن نیازهای شناسایی‌شده است.

اهداف و تمرکز

تحلیل: تمرکز در تحلیل بر روی شناسایی و فهم نیازمندی‌های کاربر و مشخص کردن آنچه سیستم باید انجام دهد، است.
طراحی: هدف از طراحی ایجاد یک معماری قابل اجرا برای نرم‌افزار است که نیازمندی‌های تحلیل شده را پوشش دهد.

سطح انتزاع

تحلیل: در تحلیل، سطح انتزاع بالاتر است. این مرحله بیشتر بر روی "چه" تمرکز دارد تا "چگونه".
طراحی: طراحی در سطح انتزاع پایین‌تر قرار دارد و بیشتر به جزئیات "چگونه" می‌پردازد.

تقدم و تاخر

تحلیل: معمولاً قبل از طراحی انجام می‌شود. ابتدا باید مسائل و نیازمندی‌ها را درک کرد.
طراحی: پس از تحلیل انجام می‌شود و بر اساس نتایج به دست آمده از تحلیل، راه‌حل‌ها طراحی می‌شوند.

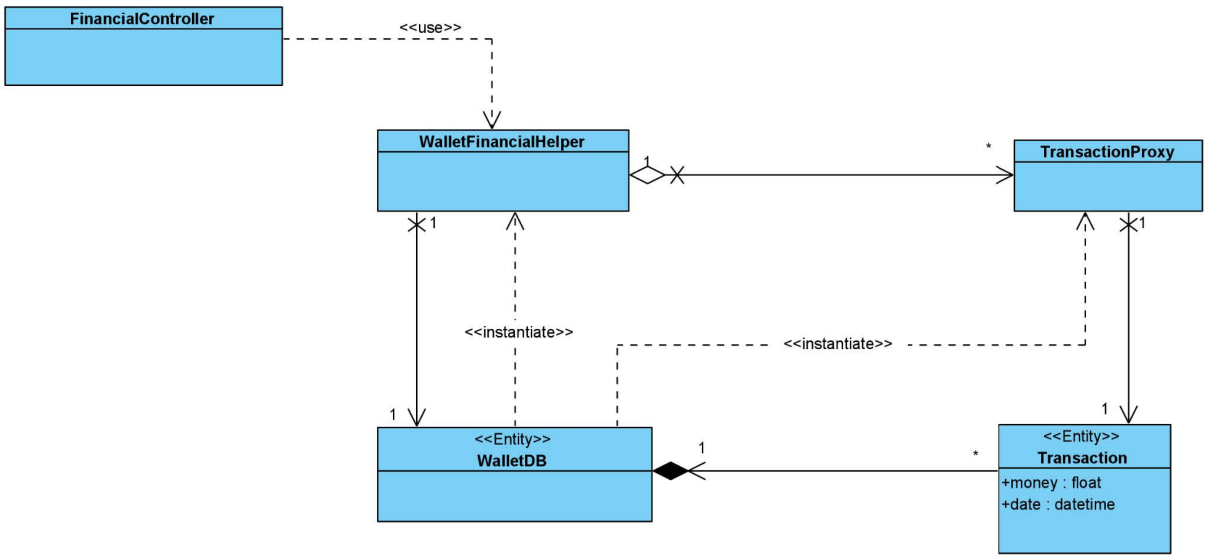
تفاوت در مدل‌سازی

تحلیل: مدل‌سازی در تحلیل بر روی نمایش نیازمندی‌ها و فرایندهای کسب‌وکار تمرکز دارد.
طراحی: در طراحی، مدل‌سازی به توصیف معماری سیستم، کلاس‌ها، اشیاء، و روابط بین آن‌ها می‌پردازد.

سوال ۴

در یک فروشگاه تحت وب، کسب و کار مربوطه وظیفه واسطه‌گری را بر عهده دارد و تعداد زیادی مشتری را به تعداد زیادی انباردار متصل می‌کند. هم مشتریان و هم انبارداران در این سیستم دارای حساب و کیف پول هستند، و هر خرید به طور مستقیم پول را از کیف پول مشتری به کیف پول فروشنده منتقل می‌کند (بدون هیچ هزینه‌ای).

مدل زیرساختی که برای این فروشگاه تعریف شده است به عنوان مدل backend شناخته می‌شود و برای بخشی از کد زیرسیستم طراحی شده است. برای سهولت در توضیح سوال، بسیاری از جزئیات (داده‌ها و عملیات کلاس‌ها) حذف شده‌اند و تمرکز بر روی کلاس‌ها و روابط بین آن‌ها است.



۱. بازسازی مدل تحلیل

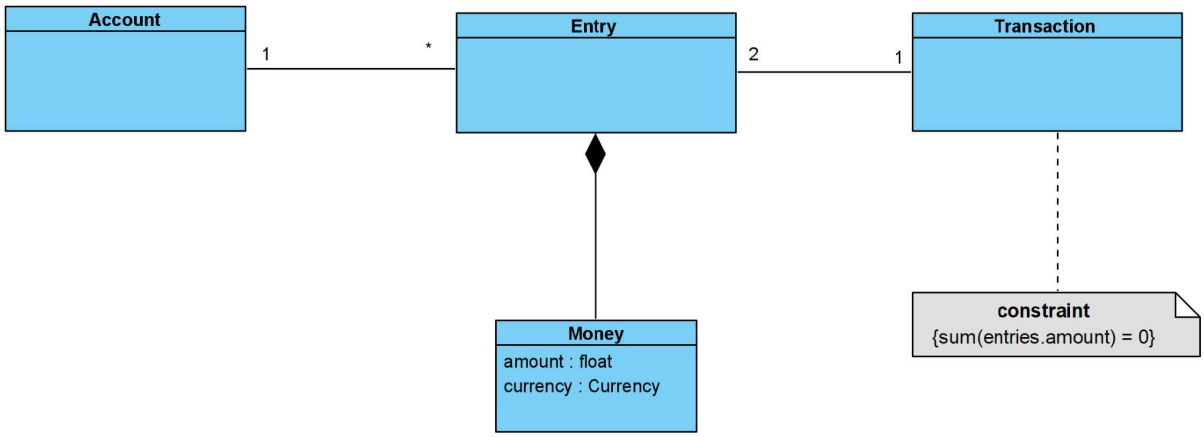
مدل تحلیل متناظر با مدل طراحی فوق از دست رفته است. آن را بازسازی کنید. (نکته: مدل تعبیه شده باید ساده‌تر و کوچک‌تر از مدل طراحی باشد.)

۲. استفاده از الگوهای تحلیل فاولر

در فصل‌های ۸ تا ۱۱ کتاب «پرسمن»، به الگوهای تحلیل - به خصوص الگوهای تحلیل فاولر - اشاره شده است. کتاب فاولر را می‌توانید از لینک مذکور دریافت کنید. از الگوی «موجودی و حسابداری - تراکنش» (فصل ۶ کتاب الگوهای تحلیل فاولر) استفاده کنید و مدل تحلیلی را که در بخش ۱ ایجاد کردید، با استفاده از این الگو غنی‌سازی کنید.

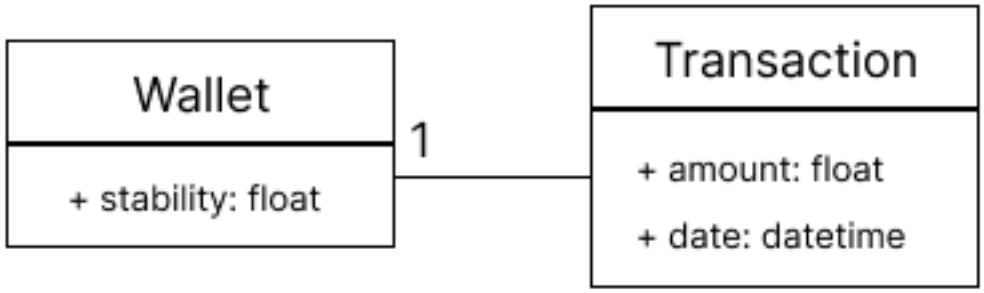
۳. توجیه بهبودها

دو مورد از بهبودهایی که این الگو به ارمغان می‌آورد را توجیه کنید.

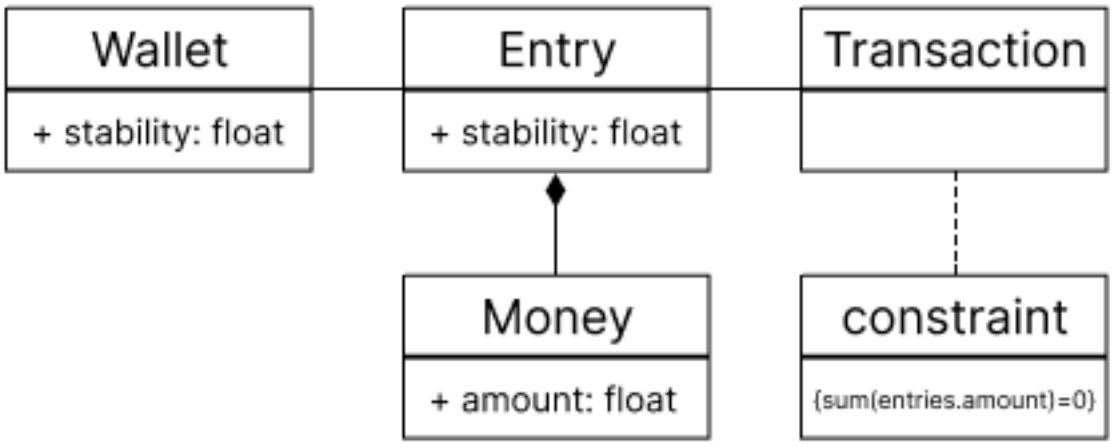


جواب سوال ۴

جواب بخش اول



جواب بخش دوم



جواب بخش سوم

بهبود اطمینان از صحت تراکنش‌ها

در مدل قبلی، هر تراکنش به صورت جداگانه ایجاد می‌شد، بدون اطمینانی از اینکه عملیات مرتبط با آن قرینه و صحیح باشد. این امر خطر ایجاد تراکنش‌های نامعتبر را افزایش می‌داد، که می‌توانست منجر به خلق یا حذف نادرست مقداری پول شود. با اعمال الگوی فاولر، هر تراکنش به وضوح با یک عملیات متقابل مرتبط می‌شود. این امر به وسیله تعریف دو transaction برای هر entry و تضمین اینکه جمع مبالغ در هر entry صفر باشد، اطمینان از صحت و تعادل مالی را فراهم می‌کند.

افزایش شفافیت و ردیابی تراکنش‌ها

در مدل قبلی، برای هر طرف در تراکنش، یک transaction جداگانه ایجاد می‌شد و رابطه میان این transaction ها مشخص نبود. در نتیجه، دسترسی به اطلاعات کامل تراکنش برای هر دو طرف دشوار بود. با به کارگیری الگوی جدید، هر transaction به یک entry مرتبط است و هر entry شامل تمام transaction های مربوط به یک مبادله مالی است. این ساختار جدید امکان ردیابی و شفافیت بیشتری را در مورد جریان‌های مالی فراهم می‌کند، زیرا تمام مبادلات مالی مرتبط با یک فرد به راحتی قابل شناسایی است.

سوال ۵

در این بخش، پنج مفهوم مدل‌سازی عمده در مهندسی نرم‌افزار – DFD ، UML ، User Story ، CRC Card و BPMN – مورد بررسی و مقایسه قرار دهید از جنبه‌های مختلف.

- چه چیزهایی را مدل می‌کنند
- آن‌ها را چگونه مدل می‌کنند
- کجا/در چه زمانی استفاده می‌شوند
- تفاوت سطح انتزاع در مدل‌سازی

جواب سوال ۵

در مهندسی نرم‌افزار، مدل‌سازی فرایندی است که به منظور ایجاد یک نمایش گرافیکی یا نمادین از یک سیستم، فرایند یا مفهوم انجام می‌شود. مدل‌سازی می‌تواند برای اهداف مختلفی استفاده شود، از جمله:

تجسم سیستم یا فرآیند

درک بهتر سیستم یا فرآیند

ارتباط موثرتر با سایرین در مورد سیستم یا فرآیند

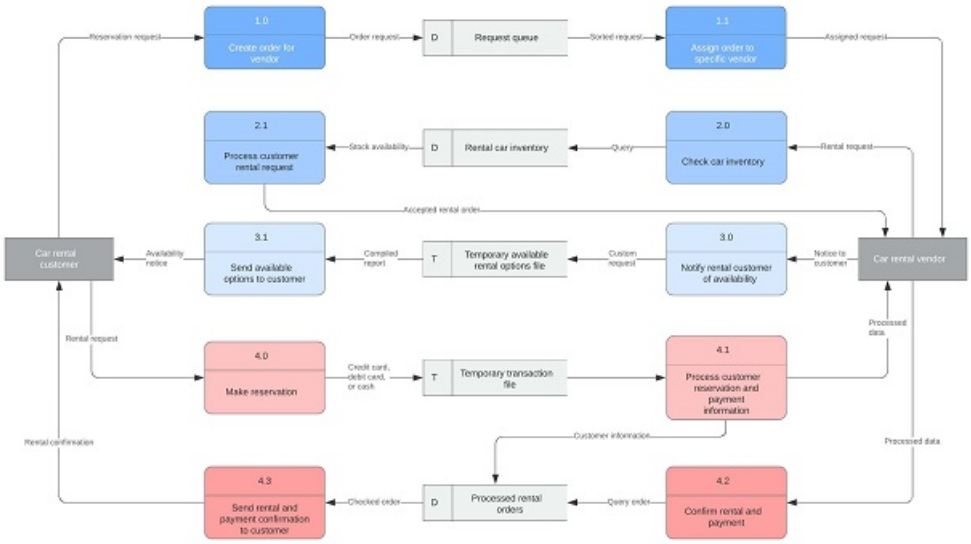
تجزیه و تحلیل و بهبود سیستم یا فرآیند

در این مقاله، مفاهیم مختلف مدل‌سازی در مهندسی نرم‌افزار را تحلیل و مقایسه می‌کنیم.

DFD (Data Flow Diagram)

DFD ، که مخفف Data Flow Diagram است، یک ابزار مهم در مهندسی نرم‌افزار برای نمایش جریان داده‌ها درون یک سیستم است. این نمودار به تحلیل‌گران و طراحان کمک می‌کند تا درک عمیق‌تری از سیستم‌ها داشته باشند.

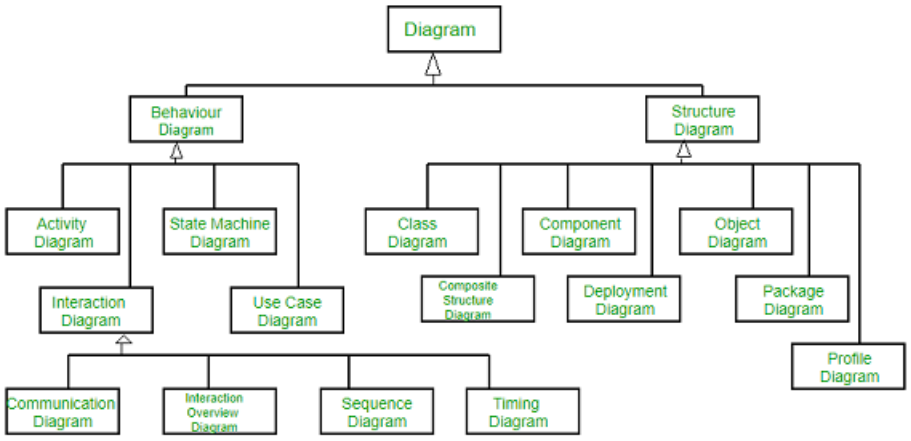
- **کاربرد:** DFD برای نمایش جریان داده‌ها بین فرآیندها، داده‌ها، سازمان‌ها و ذخیره‌سازی‌ها در سیستم‌های نرم‌افزاری استفاده می‌شود.
- **ساختار:** DFD شامل گره‌ها و خطوط است. گره‌ها نشان‌دهنده فرآیندها، مخازن داده، منابع داده و مقاصد داده هستند، و خطوط جریان داده‌ها را نشان می‌دهند.
- **سطح انتزاع:** DFD در سطوح مختلف انتزاع مورد استفاده قرار می‌گیرد.
- **مزایا:** این ابزار به شناسایی مسیرهای داده و تحلیل چگونگی انجام کار توسط سیستم کمک می‌کند.



UML (Unified Modeling Language)

UML ، که مخفف Unified Modeling Language است، یک زبان استاندارد برای مدل‌سازی و توصیف ساختار و رفتار سیستم‌های نرم‌افزاری است. این زبان از طریق استفاده از مجموعه‌ای متنوع از نمودارها کاربرد دارد.

- **انتزاع در UML :** یکی از ویژگی‌های برجسته UML توانایی آن در نمایش سطوح مختلف انتزاع است.
- **کاربرد UML :** UML در تمام مراحل توسعه نرم‌افزار مورد استفاده قرار می‌گیرد.



User Story

User Story در مهندسی نرم‌افزار به عنوان یک تکنیک مدل‌سازی برای تعریف نیازمندی‌ها و رفتار کاربران در سیستم استفاده می‌شود. این روش متمرکز بر بیان خواسته‌ها و نیازهای کاربران از طریق سناریوهای کوتاه و روشن است.

- **سطح انتزاع:** User Story نسبت به سایر روش‌های مدل‌سازی، از سطح انتزاع بالاتری برخوردار است.

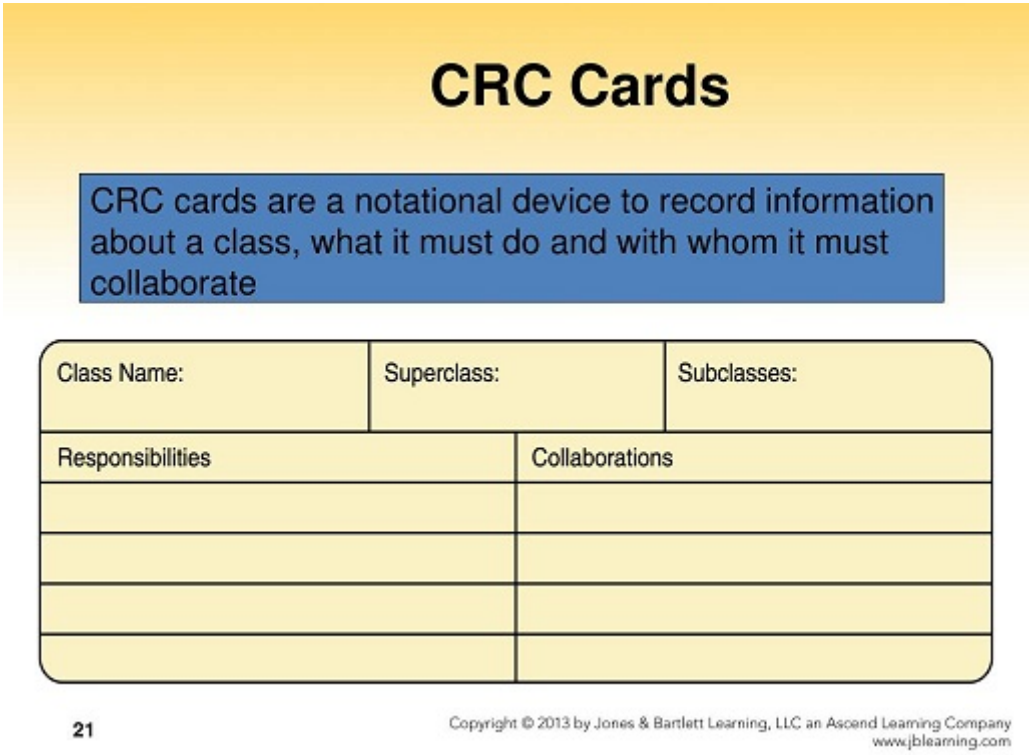
- **بررسی نیازمندی‌ها:** این روش نیازمندی‌ها را از دیدگاه ذی‌نفعان بررسی می‌کند.
- **مثال‌ها:** در سامانه دانشگاهی، User Story ها می‌توانند شامل مواردی مانند «من به عنوان یک دانشجو می‌خواهم لیست تمرین‌ها و ددلاین‌های آن‌ها را داشته باشم...» باشند.

CRC Card

CRC Card ، مخفف Class-Responsibility-Collaboration ، یک ابزار مدل‌سازی در مهندسی نرم‌افزار است که برای تحلیل و طراحی سیستم‌های شیگرا استفاده می‌شود. هر کارت CRC سه جزء اصلی دارد: کلاس، مسئولیت‌ها و همکاری‌ها.

- **کلاس:** نام کلاس در بالای کارت قرار می‌گیرد و نشان‌دهنده یک مفهوم، شیء یا موجودیت در سیستم است.
- **مسئولیت‌ها:** این بخش شامل لیستی از وظایف یا مسئولیت‌هایی است که کلاس باید انجام دهد. این مسئولیت‌ها معمولاً عملیات یا رفتارهایی هستند که کلاس بر عهده دارد.
- **همکاری‌ها:** این بخش شامل کلاس‌های دیگری است که کلاس برای انجام مسئولیت‌های خود با آن‌ها همکاری می‌کند.

کارت‌های CRC به تیم‌های توسعه کمک می‌کنند تا ساختار کلاس‌های سیستم را درک کنند و نحوه تعامل آن‌ها با یکدیگر را تجزیه و تحلیل نمایند. این رویکرد تمرکز بر روی همکاری و مسئولیت‌های متقابل را ترویج می‌کند و به شناسایی و حذف وابستگی‌های غیرضروری کمک می‌کند.



BPMN

BPMN مخفف Business Process Model and Notation به معنای مدل و نشانه‌گذاری فرآیند کسب‌وکار است. این یک زبان مدل‌سازی بصری برای برنامه‌های تجزیه و تحلیل کسب‌وکار و مشخص کردن گردش کار فرایندهای سازمانی است. BPMN توسط ابتکار مدیریت فرآیند کسب‌وکار (BPMI) توسعه یافت و از زمان ادغام دو سازمان در سال ۲۰۰۵ توسط گروه مدیریت اشیاء (OMG) حفظ شده است.

عناصر اصلی

BPMN از مجموعه‌ای از عناصر بصری برای مدل‌سازی فرآیندهای کسب‌وکار استفاده می‌کند. این عناصر عبارتند از:

- فعالیت‌ها (Activities) : فعالیت‌ها اقداماتی هستند که در یک فرآیند انجام می‌شوند. فعالیت‌ها می‌توانند شامل کارهای فیزیکی، پردازش اطلاعات یا تصمیم‌گیری باشند.
- جریان‌ها (Flows) : جریان‌ها نحوه ارتباط فعالیت‌ها را نشان می‌دهند. جریان‌ها می‌توانند به صورت خط مستقیم، خط نقطه‌چین یا خط مورب نشان داده شوند.
- شروع و پایان (Start and End) : شروع و پایان نشان‌دهنده نقاط شروع و پایان یک فرآیند هستند.
- کنترل‌ها (Controls) : کنترل‌ها شرایطی را که بر جریان فرآیند تأثیر می‌گذارند، نشان می‌دهند.
- پیوندها (Links) : پیوندها فعالیت‌ها یا جریان‌های مختلف را به هم متصل می‌کنند.

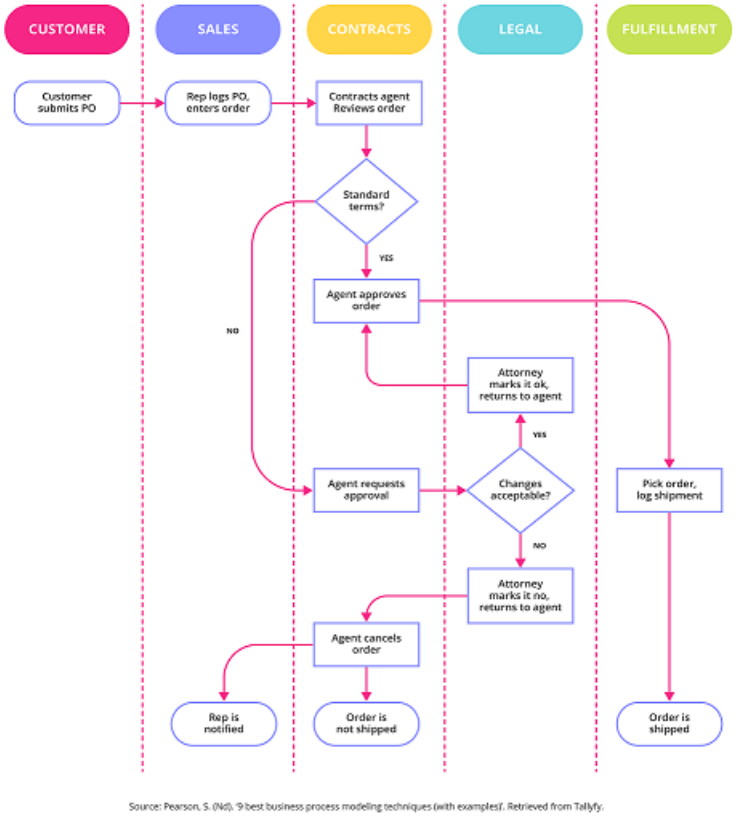
کاربردهای BPMN

BPMN در طیف گسترده‌ای از کاربردها استفاده می‌شود، از جمله:

- تجزیه و تحلیل فرآیندهای کسب‌وکار: BPMN می‌تواند برای تجزیه و تحلیل فرآیندهای کسب‌وکار موجود و شناسایی فرصت‌های بهبود استفاده شود.
- طراحی فرآیندهای کسب‌وکار جدید: BPMN می‌تواند برای طراحی فرآیندهای کسب‌وکار جدید استفاده شود.
- پیاده‌سازی فرآیندهای کسب‌وکار: BPMN می‌تواند برای پیاده‌سازی فرآیندهای کسب‌وکار در سیستم‌های فناوری اطلاعات استفاده شود.
- آموزش فرآیندهای کسب‌وکار: BPMN می‌تواند برای آموزش کارکنان در مورد فرآیندهای کسب‌وکار استفاده شود.

آینده BPMN

BPMN یک زبان قدرتمند و انعطاف‌پذیر است که به سازمان‌ها کمک می‌کند تا فرآیندهای کسب‌وکار خود را به‌طور موثرتر مدیریت کنند. انتظار می‌رود که BPMN در سال‌های آینده به محبوبیت خود ادامه دهد، زیرا سازمان‌ها به دنبال راه‌هایی برای بهبود کارایی و بهره‌وری خود هستند.



Source: Pearson, S. (Nd). '9 best business process modelling techniques (with examples)'. Retrieved from Tallyfy.

DFD (Data Flow Diagram)

- **چه چیزهایی را مدل می‌کند:** جریان داده‌ها و ارتباطات بین فرایندها، داده‌ها و ذخیره‌سازی‌ها.
- **چگونگی مدل‌سازی:** با استفاده از نمودارهای گرافیکی برای نشان دادن جریان داده‌ها.
- **زمان استفاده:** در مراحل اولیه تحلیل سیستم برای درک بهتر جریان اطلاعات.
- **سطح انتزاع:** سطح بالا در ارتباط با جریان داده‌ها.

UML (Unified Modeling Language)

- **چه چیزهایی را مدل می‌کند:** ساختار و رفتار سیستم‌های نرم‌افزاری.
- **چگونگی مدل‌سازی:** با استفاده از مجموعه‌ای متنوع از نمودارها (مانند نمودار کلاس، نمودار توالی).
- **زمان استفاده:** در تمام مراحل توسعه نرم‌افزار.
- **سطح انتزاع:** متغیر، بسته به نوع نمودار.

User Story

- **چه چیزهایی را مدل می‌کند:** نیازمندی‌ها و ویژگی‌های کاربران از دیدگاه آن‌ها.
- **چگونگی مدل‌سازی:** به صورت جملات ساده و قابل فهم برای توصیف داستان‌های کاربری.
- **زمان استفاده:** بیشتر در رویکردهای توسعه چابک.
- **سطح انتزاع:** بسیار بالا و کاربر محور.

CRC Card (Class-Responsibility-Collaboration)

- **چه چیزهایی را مدل می‌کند:** وظایف، مسئولیت‌ها و همکاری‌های کلاس‌ها.
- **چگونگی مدل‌سازی:** با استفاده از کارت‌هایی که کلاس‌ها و وظایف آن‌ها را نمایش می‌دهند.
- **زمان استفاده:** در مرحله طراحی سیستم و تعریف مسئولیت‌های کلاس‌ها.
- **سطح انتزاع:** متوسط تا بالا در ارتباط با ساختار کلاس‌ها.

BPMN (Business Process Model and Notation)

- **چه چیزهایی را مدل می‌کند:** فرایندهای کسب‌وکار و وظایف مرتبط.
- **چگونگی مدل‌سازی:** با استفاده از نمودارهای فرایندی و نشانه‌گذاری‌های استاندارد.
- **زمان استفاده:** برای تحلیل و بهبود فرایندهای کسب‌وکار.
- **سطح انتزاع:** بالا در ارتباط با فرایندهای سازمانی.

سوال ۶

تحلیل روش طراحی ویژگی‌رانه (نسخه سوم) را به دقت مطالعه کنید و تحلیل خود را از این روش براساس موارد زیر بیان کنید.

مستندات معماری

شامل تصمیمات، عقلانیت، دیدگاه‌های معماری، راه‌حل‌های جایگزین، بازنمایی و سایر موارد اشاره شده در کتاب.

نگرانی‌های همه انواع ذی‌نفعان

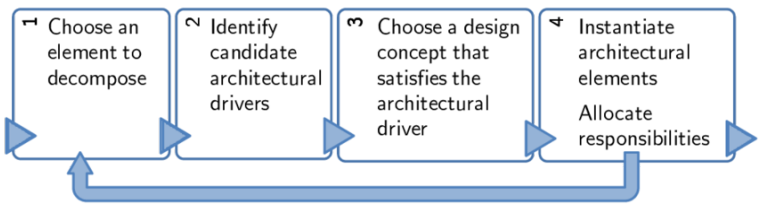
شامل کاربر نهایی، مشتری، تیم ایجاد، مدیر پروژه و ...

چگونگی کاربرد مفاهیم، اصول، الگوها و سبک‌های معماری

بررسی نحوه کاربرد این مفاهیم در روش طراحی ویژگی رانه.

توجه کنید پوشایی و دقت پاسخ شما به عنوان ملاک مهم ارزیابی در این سوال در نظر گرفته می‌شود.

جواب سوال ۶



روش طراحی ویژگی‌رانه (ADD) یک رویکرد چابک برای توسعه معماری نرم‌افزار است که بر اساس ویژگی‌ها متمرکز است. این روش در سال ۲۰۰۴ توسط دو نفر به نام‌های جف باک و ریچارد هافمن معرفی شد و در سال ۲۰۱۶ نسخه سوم آن منتشر شد.

روش ADD روشی است که ما در آن از یک رویکرد تکراری استفاده می‌کنیم. هدف اصلی ما این است که یک سیستم را به گونه‌ای طراحی، تست و تولید کنیم که بتوانیم به خوبی کنترل کنیم و به ویژگی‌های کیفی مورد نظر دست یابیم. این فرآیند شامل سه مرحله اصلی: برنامه‌ریزی، اجرا و بررسی است. یکی از مزایای کلیدی این رویکرد تولید یک معماری کاربردی است که ممکن است کاملاً بی‌نقص نباشد، اما در طول زمان با استفاده از تجربیات کسب شده بهبود می‌یابد. همچنین، این روش باعث افزایش قابلیت پیش‌بینی و قابلیت ردیابی فرآیند می‌شود.

مستندات معماری

در روش طراحی ویژگی‌رانه، مستندات معماری به صورت مداوم و در طول فرآیند توسعه ایجاد می‌شود. این مستندات شامل موارد زیر هستند:

- دستور کار (Action Plan) : این سند شامل جزئیات هر گام از فرآیند طراحی ویژگی‌رانه است. این سند باید شامل موارد زیر باشد:
  اهداف هر گام
  فعالیت‌های مورد نیاز برای رسیدن به اهداف
  زمان‌بندی هر فعالیت
  منابع مورد نیاز برای هر فعالیت
- استانداردها و الگوها (Standards and Patterns) : این سند شامل استانداردها و الگوهای مورد استفاده در طراحی معماری است. این سند باید شامل موارد زیر باشد:
  استانداردهای فنی، مانند استانداردهای زبان برنامه‌نویسی، استانداردهای پایگاه داده، و استانداردهای امنیت
  الگوهای معمارانه، مانند الگوهای سه‌لایه، MVC ، و MVVM
- توضیحات معماری (Architecture Description) :
  این سند شامل توضیحات کامل از معماری نرم‌افزار است. این سند باید شامل موارد زیر باشد:
  اجزای نرم‌افزار
  روابط بین اجزای نرم‌افزار
  ویژگی‌های نرم‌افزار
  محدودیت‌های نرم‌افزار

این مستندات باید به گونه‌ای تهیه شوند که برای همه ذی‌نفعان قابل فهم باشند.

نگرانی‌های همه انواع ذی‌نفعان

روش طراحی ویژگی‌رانه بر اهمیت توجه به نگرانی‌های همه انواع ذی‌نفعان تأکید دارد. این ذی‌نفعان شامل موارد زیر هستند:

- کاربر نهایی (End User) : افرادی که نرم‌افزار را استفاده می‌کنند. نگرانی‌های این افراد شامل موارد زیر است:
  کارایی نرم‌افزار
  سهولت استفاده از نرم‌افزار
  قابلیت اطمینان نرم‌افزار
- مشتری (Customer) : افرادی که نرم‌افزار را سفارش می‌دهند. نگرانی‌های این افراد شامل موارد زیر است:
  هزینه نرم‌افزار
  زمان تحویل نرم‌افزار
  الزامات قانونی نرم‌افزار

- تیم ایجاد (Development Team) : افرادی که نرم‌افزار را توسعه می‌دهند. نگرانی‌های این افراد شامل موارد زیر است: پیچیدگی نرم‌افزار
  قابلیت نگهداری نرم‌افزار
  قابلیت توسعه‌پذیری نرم‌افزار
- مدیر پروژه (Project Manager) : افرادی که پروژه توسعه نرم‌افزار را مدیریت می‌کنند. نگرانی‌های این افراد شامل موارد زیر است:
  ریسک‌های پروژه
  زمان‌بندی پروژه
  بودجه پروژه

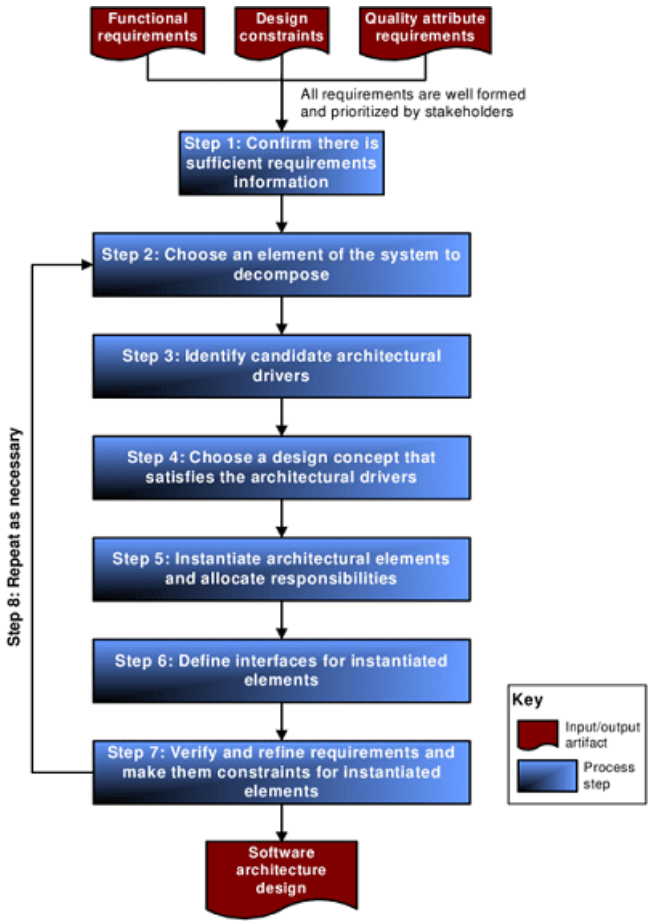
روش طراحی ویژگی‌رانه به ذی‌نفعان کمک می‌کند تا در فرآیند طراحی معماری مشارکت داشته باشند و نگرانی‌های خود را بیان کنند.

چگونگی کاربرد مفاهیم، اصول، الگوها و سبک‌های معماری

روش طراحی ویژگی‌رانه از مفاهیم، اصول، الگوها و سبک‌های معماری برای ایجاد یک معماری نرم‌افزار کارآمد و پایدار استفاده می‌کند. این مفاهیم، اصول، الگوها و سبک‌ها عبارتند از:

- مفاهیم معماری: مانند مفاهیم سیستم‌های باز، سیستم‌های توزیع‌شده و سیستم‌های چند لایه.
- اصول معماری: مانند اصل انتزاع، اصل تفکیک وظایف و اصل وابستگی ضعیف.
- الگوهای معماری: مانند الگوهای معمارانه مشترک مانند سه‌لایه، MVC و MVVM .
- سبک‌های معماری: اینها روش‌هایی هستند که بیانگر یک رویکرد کلی در طراحی معماری هستند. سبک‌های معماری مانند معماری میکروسرویس‌ها، معماری مبتنی بر رویداد و معماری سرویس گرا (SOA) تاکید بر اصولی مانند تجزیه و تحلیل، توزیع وزنی کارکردها و تعامل میان سرویس‌های مستقل دارند. این سبک‌ها به معماران کمک می‌کنند تا سیستم‌هایی پیچیده و قابل توسعه را طراحی کنند که بتوانند به خوبی در محیط‌های پیچیده و متغیر پاسخگو باشند.

در نهایت، این مفاهیم، اصول، الگوها و سبک‌های معماری به معماران نرم‌افزار این امکان را می‌دهند تا سیستم‌هایی ایجاد کنند که نه تنها به خوبی عمل می‌کنند بلکه قابلیت انعطاف و تطابق با تغییرات آینده را نیز دارند.



حالا با توجه به تصویر، به توضیح این روش و گام‌های مختلف آن نیز می‌پردازیم.

Attribute-Driven Design (ADD) نسخه ۳ یک روش طراحی معماری نرم‌افزار است که بر تحلیل و طراحی سیستم‌های نرم‌افزاری با تمرکز بر ویژگی‌های کیفی (مانند امنیت، عملکرد، قابلیت نگهداری و غیره) متمرکز است. این روش به طور خاص برای کمک به معماران نرم‌افزار در مواجهه با سیستم‌های پیچیده طراحی شده است. در ادامه هفت گام اصلی ADD نسخه ۳ را توضیح داده و کد لاتک مربوطه را ارائه می‌کنم:

الف) **شناسایی مدول‌های اصلی سیستم:** در این مرحله، وظایف اصلی سیستم شناسایی و به مدول‌های مختلف تقسیم می‌شوند. این کار با توجه به ویژگی‌های کیفی مورد نیاز و کارکردهای کلیدی سیستم انجام می‌شود.

ب) **تعریف ویژگی‌های کیفی و محدودیت‌ها:** در این گام، ویژگی‌های کیفی مورد نظر و محدودیت‌های مرتبط با هر مدول شناسایی و تعریف می‌شوند.

ج) **طراحی مدول‌ها:** در این مرحله، برای هر مدول، طراحی انجام می‌شود. این شامل تعریف رابط‌ها، خدمات و عملکردهایی است که هر مدول باید ارائه دهد.

د) **تحلیل رفتار مدول‌ها:** این گام شامل بررسی و تحلیل رفتار مدول‌ها در سناریوهای مختلف است تا اطمینان حاصل شود که ویژگی‌های کیفی مورد نظر برآورده می‌شوند.

ه) **تخصیص وظایف به مدول‌ها:** در این گام، وظایف و مسئولیت‌های مشخص به هر مدول اختصاص داده می‌شوند.

و) **تعریف رابط‌های بین مدول‌ها:** این مرحله شامل تعریف نحوه تعامل و ارتباط بین مدول‌های مختلف است.

ز) **دوباره سازی و بازبینی طراحی:** در نهایت، طراحی کلی سیستم بازبینی و در صورت نیاز، تنظیم مجدد می‌شود تا اطمینان حاصل شود که همه ویژگی‌های کیفی و محدودیت‌ها به درستی پوشش داده شده‌اند.