به نام خدا

تمرین دو

نیم‌سال ۱۴۰۲۱

توضیحات

- لطفاً پاسخ‌ها را به صورت تایپ شده در قالب فایل PDF، حداکثر تا ساعت ۲۳:۵۹ تاریخ تعیین‌شده در صفحه‌ی درس‌افزار درس بارگذاری نمایید.
- ذکر **نام و نام خانوادگی** به همراه **شماره دانشجویی** همه‌ی اعضای گروه، هم‌چنین **شماره‌ی تیم** در فایل PDF پاسخ‌ها ضروری است. در صورتی که نام هر یک از اعضای گروه در فایل پاسخ‌ها نباشد، به منزله عدم همکاری آن عضو در گروه و نارضایتی سایر هم‌گروهی‌ها محسوب شده و نمره تمرین برای آن فرد لحاظ نخواهد شد.
- در صورت ارسال پاسخ‌ها به صورت دست‌نویس تضمینی در تصحیح آن وجود نخواهد داشت.
- هدف درس مهندسی نرم‌افزار آشنایی شما با دنیای نرم‌افزار و افزایش مهارت تحلیل شماست. استفاده از ربات‌های هوشمند مانند ChatGPT برای پاسخ‌دهی به سوالات، مغایر با اهداف گفته شده است؛ از این رو توصیه می‌کنیم که برای پاسخ‌دهی به تمرین‌ها از این ربات‌ها استفاده نکنید.
- تمرین از **۱۱۰ نمره** است و **۱۰ نمره** امتیازی دارد. نمرات امتیازی هر تمرین فقط می‌تواند برای جبران نمرات ازدست‌رفته‌ی سایر تمرین‌ها استفاده شود و به بخش‌های دیگر درس مانند آزمون‌ها منتقل نمی‌شود.
- سیاست ارسال با تاخیر برای این تمرین به صورت زیر است:
  - تا ۲۴ ساعت نمره‌ای کسر نمی‌شود.
  - پس از ۲۴ ساعت، به ازای هر ساعت تاخیر ۱ درصد نمره کسر می‌شود.
  - این سیاست برای هر یک از تمارین درس برقرار است.
- چنانچه یک نفر از اعضای هر گروه پاسخ تمرین را در درس‌افزار درس بارگذاری کند، کافی است.
- پاسخ‌ها را به زبان **فارسی** بنویسید. در صورتی که ترجمه‌ی کلمه‌ای ناملموس می‌شد، واژه‌ی اصلی را به‌صورت پانویس اضافه کنید.
- **توجه کنید که پوشایی و دقت پاسخ‌های شما، ملاک ارزیابی است.**

موفق باشید

تیم آموزش مهندسی نرم‌افزار

sharif.software.engineering@gmail.com

سوال ۱ (۵ نمره)

یک قانون سرانگشتی در فاز تحلیل این است که «افراد تیم ایجاد در فاز تحلیل[1] باید بر نیازمندی‌هایی تمرکز کنند که در حوزه‌ی مسئله[2] و کسب‌وکار[3] قرار دارد».

۱. چه نوع نیازمندی‌هایی در این حوزه‌ها نیستند؟

۲. مثال بزنید.

پاسخ:

نیازمندی‌های فنی و technical؛ مثلاً موارد مرتبط با پایگاه داده یا تکنولوژی‌هایی که قرار است در فاز طراحی و پیاده‌سازی[4] استفاده کنیم.

هم‌چنین در فاز تحلیل در مورد نیازمندی‌های غیروظیفه‌ای نیز صحبت نمی‌کنیم. نیازمندی‌هایی مانند سرعت پاسخ به درخواست‌ها[5]، دسترس‌پذیری[6] نرم‌افزار، رابط کاربری[7] و تجربه‌ی کاربری[8] ...

---

[1] Analysis
[2] Problem Domain
[3] Business Domain
[4] Implementation
[5] Response Time
[6] Accessibility
[7] User Interface (UI)
[8] User Experience (UX)

سوال ۲ (۱۰ نمره)

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؟

پاسخ بخش ۱:

شباهت‌ها:

- طراحی معماری هر دو، در ابتدای فرآیند ایجاد آن‌ها انجام می‌شود.
- با گذشته زمان، هزینه‌ی تغییر معماری در هر دو زیاد و زیادتر می‌شود.
- نسبت آدم‌هایی که می‌توانند یک ساختمان/نرم‌افزار را ایجاد/پیاده‌سازی کنند، از آدم‌هایی که می‌توانند معماری یک ساختمان/نرم‌افزار را طراحی کنند، بیشتر است (در صنعت نرم‌افزار، تعداد کدنویسان بیشتر از معماران است).
- برای طراحی معماری ساختمان/نرم‌افزار، الگوهای از پیش آماده‌ای وجود دارد (برای مثال در صنعت ساختمان، الگوهایی برای طراحی ساختمان‌های آموزشی (مانند مدرسه‌ها یا دانشگاه‌ها) یا ساختمان‌های درمانی (مانند بیمارستان‌ها) یا ساختمان‌های مسکونی وجود دارد).
- در طراحی معماری ساختمان/نرم‌افزار، به جنبه‌های غیروظیفه‌ای[9] نیز توجه می‌شود (برای مثال در طراحی معماری ساختمان، به دسترس‌پذیری[10] بخش‌های مختلف ساختمان از جمله پارکینگ، رسیدن یا نرسیدن نور خورشید به یک واحد، منظره‌ی جلوی پنجره‌های یک واحد یا ... اهمیت داده می‌شود).
- هر چه ساختمان/نرم‌افزار بزرگ‌تر باشد، طراحی معماری آن نیز سخت‌تر می‌شود.
- «خلاقیت» در طراحی معماری ساختمان/نرم‌افزار نقش پررنگی دارد.

تفاوت‌ها:

- پس از ایجاد ساختمان/نرم‌افزار، تغییر معماری ساختمان تقریباً غیرممکن است، اما تغییر معماری نرم‌افزار امکان‌پذیرتر است؛ چرا که به‌صورت کلی می‌توان گفت نرم‌افزار نرم‌تر از ساختمان است.

پاسخ بخش ۲:

شباهت‌ها:

- هر دو دانشجو دارند :/ (قبول داریم که سوال بخش ۲، سوال خوبی نبود)

تفاوت‌ها:

- دانشجویان رشته‌ی معماری ساختمان، پیشینه‌ی هنری و انسانی بیشتری تا دانشجویان رشته‌ی معماری کامپیوتر دارند. دانشجویان رشته‌ی معماری کامپیوتر جنبه‌ی مهندسی و ریاضی بیشتری

[9] Non-functional
[10] Accessibility

دارند.

- تعداد دانشجویان رشته‌ی معماری ساختمان بیشتر از دانشجویان رشته‌ی معماری نرم‌افزار است؛ چرا که معماری ساختمان حوزه‌ی مشهودتر، کاربردی‌تر و واضح‌تری از علم است (بشر به قدمت بودنش به ساختمان برای زندگی‌کردن نیاز داشته است، اما کمتر از ۱۰۰ سال است که نرم‌افزار پا به زندگی بشر گذاشته

توجه کنید که جواب‌های صحیح دیگر نیز قابل قبول است.

سوال ۳ (۲۰ نمره)

تفاوت فعالیت‌های تحلیل[11] و طراحی[12] سیستم‌های نرم‌افزاری را توضیح دهید. اطمینان حاصل کنید که در توضیحات خود به موارد زیر بپردازید:

- ارتباط آن دو با یک مساله و راه‌حل آن
- اهداف و تمرکز هر یک
- سطح انتزاع[13] هر کدام
- تقدم و تاخر هر یک از این دو فعالیت
- تفاوت مدل‌سازی ذیل هر فعالیت

پاسخ:

ارتباط آن دو با یک مساله و راه‌حل آن

در جریان‌کاری تحلیل به سوال چیستیِ سیستم پاسخ خواهیم داد در حالی که در جریان کاری طراحی به چگونگی عملکرد آن پاسخ خواهیم داد. تحلیل بر قلمروی مسئله احاطه دارد در حالی که طراحی بر قلمرو راه‌حل.

اهداف و تمرکز هر یک

اهداف و تمرکز فعالیت‌های تحلیل:

مدلسازی سیستم با یک رویکردِ «همانطور که هست»[14]. در بررسی‌های این جریان کاری ما شهودی کاملی به هر آنچه در سیستم می‌گذرد، منتزع از مسائل نرم‌افزاری، خواهیم رسید.

تکمیل و تدقیق نیازمندی‌ها. برای درک و مدلسازی کامل‌تر از ما وقعِ سیستم، غالبا نیازمندی‌های موجود کم می‌آورند. به این منظور نیازمندی‌های کنونی سیستم را تدقیق کرده و نیازمندی‌های جدیدتری نیز استخراج می‌کنیم که به ما در مدل کردن و فهم کامل سیستم کمک کند.

غالبا مدل‌ها و مستندات تهیه شده در این جریان کاری به حدی منتزع از جزییات هستند که می‌توان به عنوان یک پروپوزال از سیستم با ذی‌نفعان به اشتراک گذاشته شوند و برای «تصمیم رو به جلو»[15] مورد استفاده قرار گیرند.

اهداف و تمرکز فعالیت‌های طراحی:

مدلسازی سیستم با یک رویکردِ «آن طور که باید باشد»[16]. در بررسی‌های این جریان کاری ما شهود کاملی نسبت به سیستم **نرم‌افزاری** که در حال ایجاد آن هستیم پیدا خواهیم کرد.

---

[11] Analysis
[12] Design
[13] Abstraction
[14] as-is
[15] Go Forward Decision
[16] to-be

تکمیل و تدقیق مدل‌های تحلیل. به مدل‌های تحلیلی که پیشتر ساختیم (با هر زبان مدل‌سازی، بر فرض UML یا حتی مستندات متنی) جزییات نرم‌افزاری می‌افزاییم. برای مثال ملاحظات مربوط به پایگاه داده، رابط کاربری[17]، کارگزارها[18] و ... همگی به مدل‌های پیشین اضافه می‌شوند.

تهیه نقشه‌راهی برای پیاده‌سازی. فعالیت‌های طراحی به حدی راه‌حل نرم‌افزاری را شفاف می‌کنند که مرحله بعد یعنی پیاده‌سازی به سادگی با استفاده از مستندات طراحی، یا حتی با استفاده از ابزارهای خودکارِ تبدیلِ مدل به کد، انجام خواهد شد.

سطح انتزاع هر کدام و تقدم و تاخر هر یک از این دو فعالیت

از لحاظ سطح انتزاع و تقدم-تاخر می‌توان ترتیب زیر را برای هر تکرارِ ایجاد نرم‌افزار در نظر گرفت:

نیازمندی←تحلیل←طراحی←پیاده‌سازی و تست

همانطور که در اهداف و تمرکزهای جریان‌های کاری تحلیل و طراحی ذکر شد، طراحی در واقع تدقیقِ[19] تحلیل و بالطبع تحلیل انتزاعی[20] از طراحی است. یک چنین ترتیبی از حل مسئله در حوزه نرم‌افزار از این بابت مورد پسند مهندسین است که مانند خود روش‌های برنامه نویسی، از سطوح انتزاعی استفاده می‌کند و شبیه به تعریف چند کلاس که از همدیگر ارث‌بری می‌کنند و یک دیگر را کامل می‌کنند می‌باشد. در واقع مهندس می‌تواند مرحله مرحله نیازمندی را محقق کند و در هر مرحله پاسخ پیشین خود را کامل‌تر سازد (به جای اینکه پرش بلندی از نیازمندی به پیاده‌سازی بزند).

تفاوت مدل‌سازی ذیل هر فعالیت

از منظر مدل‌سازی نیز مدل‌های تحلیل - همانطور که پیش‌تر مطرح شد - از جزییات نرم‌افزاری کاملا مبرا هستند. این موضوع اما در مورد مدل‌های طراحی کاملا بر عکس است. مدل‌های طراحی می‌توانند به حدی دارای جزییات باشند که فعالیت پیاده‌سازی تقریبا کار بدیعی نداشته باشد جز کد زدن مابه ازای مدل‌های طراحی. مدل‌سازی با استفاده از نمودار کلاسی UML به عنوان مثال در نظر بگیرید. کلاس‌های تحلیل معمولا کلاس‌های ماوقع سیستم هستند؛ مفاهیمی خواهند بود که حتی یک مالک محصول[21] غیر نرم‌افزاری نیست با شنیدن اسم آن‌ها حسی از وجودشان خواهند داشت. بسیاری از این کلاس‌ها دقیقا از خود دامنه‌ی کسب‌وکار استخراج شده‌اند و در مرحله بعد - یعنی طراحی - معادلا برای بسیاری از آن‌ها یک کلاسِ ORM نیز قرار داده می‌شود تا نمونه‌هایشان[22] در پایگاه داده ذخیره شوند.

جزییات زیادی به این مدل‌ها در فعالیت‌های طراحی اضافه خواهد شد و به عبارتی تدقیق می‌شوند. تعداد کلاس‌های موجود در نمودار کلاسی چند برابر می‌شود چرا که تعداد زیادی کلاس جعلی[23] تحت اثر الگوهای طراحی شکل می‌گیرند. به علاوه وظایف حیطه نرم‌افزار (مانند کار با UI یا پایگاه داده) به کلاس‌های پیشین اضافه می‌شود که به دلیل حفظ تک مسئولیت در سیستم خود این وظایف به سلسه‌مراتب[24] هایی موازی با

---

[17] User Interface (UI)
[18] Servers
[19] Refinement
[20] Abstraction
[21] Product Owner - PO
[22] Instance
[23] Fabricated
[24] Hierarchy

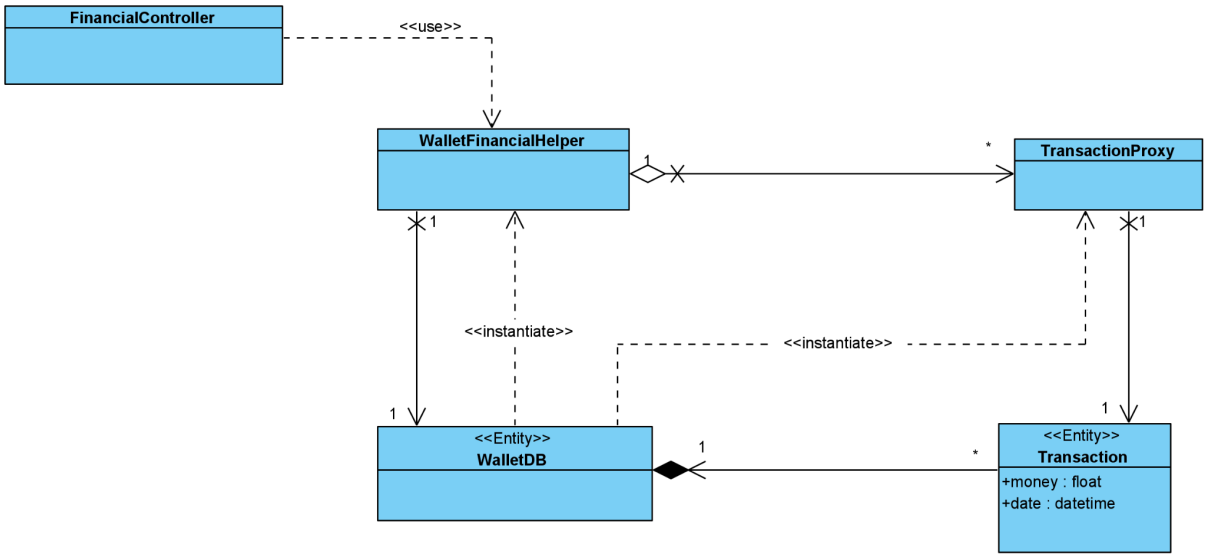
کلاس‌های پیشین واگذار[25] خواهند شد. تعدادی عملیات و صفت جدید به نمودار اضافه می‌شود و عملیات‌ها و صفاتی که از مرحله تحلیل وجود داشتن نیز جزییاتی از قبیل سطح دسترسی (private, protected, public) و نوع[26] دریافت خواهند کرد.

[25] Delegate
[26] Type

سوال ۴ (۲۵ نمره)

در یک فروشگاه تحت وب، کسب و کار مربوطه تنها عمل واسط را بر عهده داشته و تعداد زیادی مشتری را به تعداد زیادی انباردار متصل می‌کند. در واقع هم مشتری و هم انباردار در این سیستم دارای حساب و کیف پول می‌باشند، و هر خرید پول را مستقیم از کیف پول مشتری به کیف پول خریدار منتقل می‌کند (برای سادگی انتقال پول بدون هیچ هزینه‌ای اتفاق می‌افتد).

فرض کنید مدل زیر مابه‌ازای بخشی از کد زیرسیستم backend این فروشگاه می‌باشد؛ به چنین مدلی، مدل طراحی می‌گویند. به منظور تسهیل سوال، بسیاری از جزئیات (داده و عملیات کلاس‌ها) حذف شده‌اند و تمرکز مدل بر کلاس‌ها و روابط بین آن‌ها است.

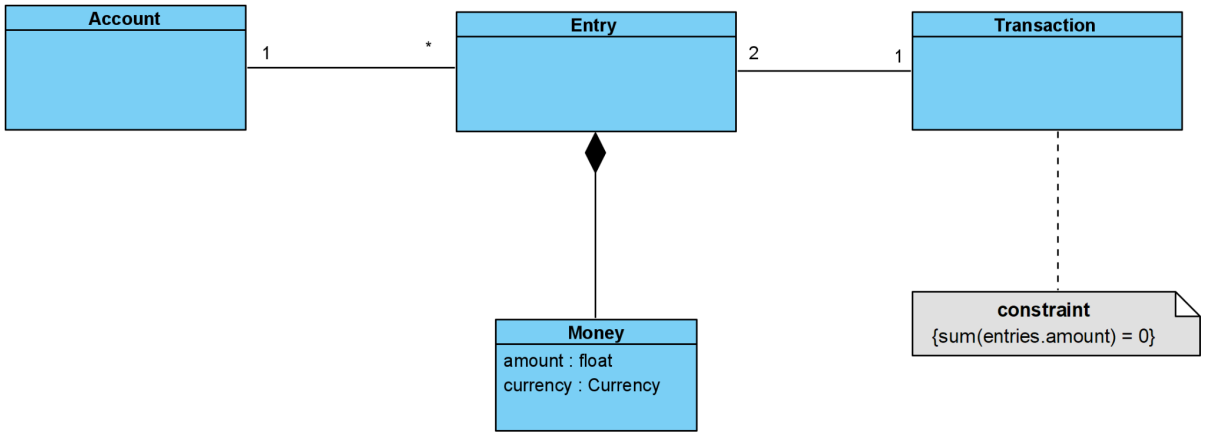


۱. مدل تحلیل متناظر با مدل طراحی فوق از دست رفته است. آن را شما تعبیه کنید. (**راهنمایی:** مدل تعبیه شده توسط شما باید بسیار ساده‌تر و کوچک‌تر از مدل فوق باشد)

۲. در فصل‌های ۸ تا ۱۱ کتاب پرسمن، بارها به الگوهای تحلیل -بالاخص الگوهای تحلیل فاولر[27]- اشاره شده است. می‌توانید کتاب فاولر را از اینجا دریافت کنید.

الگوی زیر «موجودی و حسابداری - تراکنش»[28] نام دارد (فصل ۶ کتاب الگوهای تحلیل فاولر).

[27] Fowler's Analysis Patterns
[28] Inventory and Accounting - Transaction



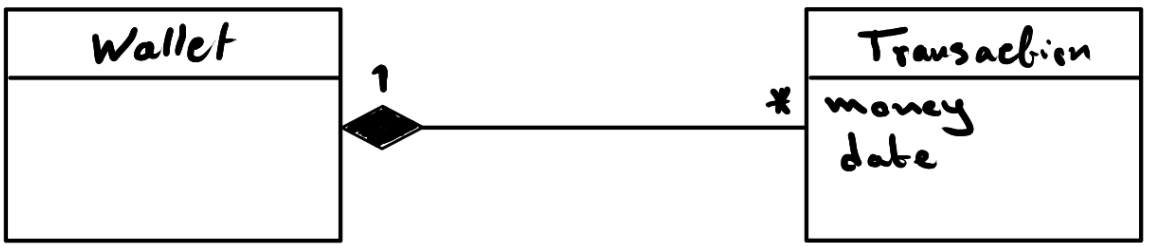
از این الگو برای غنی کردن مدل تحلیلی که در بخش ۱ فراهم آوردید، استفاده کنید. در واقع باید به نحوی مدل بخش ۲ را تغییر دهید که این الگو در آن نهادینه شده باشد (مانند استفاده از الگوهای طراحی در هنگام کد زدن).

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

پاسخ:

۱. (۱۰ نمره: ۷ نمره مدلسازی، ۳ نمره توضیحات)

پیش‌تر در سوال ۳ از این تمرین تفاوت مدلسازی ذیل جریان‌کاری تحلیل و طراحی را تشریح کردیم. مدلسازی در فضای تحلیل از جزییات نرم‌افزاری مبری است و تمرکز آن بر بصری‌سازی دامنه‌ی مسئله است. به همین ترتیب انتظار می‌رود مدل تحلیلی که از روی مدل طراحی فوق می‌کشیم تنها دارای کلاس‌هایی باشد که در قلمروی مسئله دارای موضوعیت هستند. کلاس‌هایی که وقتی از آن‌ها اسم می‌بریم، برای فردی با دانش نرم‌افزاری اندک اما دارای درک کافی از دامنه‌ی کسب‌وکار معنادار هستند.



کلاس TransactionProxy در مدل طراحی، یک بسته‌بند[29] دور کلاس دیتابیسیِ Transaction بود. تمام بسته‌بندها (پروکسی، آداپتور، دکوراتور و نما[30]) کلاس‌های جعلی هستند که طراحی را راحت‌تر و خواناتر

[29] Wrapper
[30] Facade

می‌کنند و وجودشان در کلاس‌های تحلیل فاقد موضوعیت است. چنین کلاس‌هایی در کسب‌وکار وجود ندارند و فقط در پیاده‌سازی نرم‌افزاری با توجیحات مهندسی وارد می‌شوند.

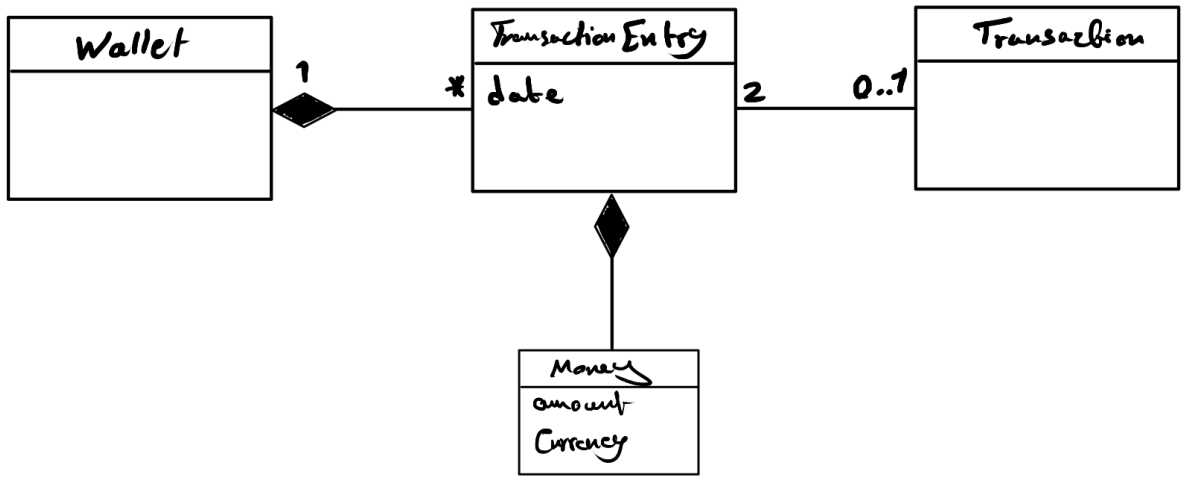
کلاس WalletFinancialHelper یک واگذاری[31] از بخشی از رفتارِ کلاس WalletDB است. در واقع طراح به دلیل حفظ اصل تک‌مسئولیتی[32] ترجیح داد کلاس WalletDB را منسجم[33] نگه دارد و برای بخشی از رفتار ریزدانه‌ی مالی آن یک کلاس Helper قرار دهد. وجود این کلاس نیز در فضای مسئله فاقد موضوعیت است و انتظار می‌رود مسئولیت[34]های آن در خود ماهیتِ کیف‌پول مشهود باشد.

کلاس FinancialController تحت اثرِ الگوهای طراحیِ GRASP[35] قرار داده شده است و وظیفه آن این است که مرز مشخصی بین View (درگاه ارتباط با UI) و Model (محل قرارگیری منطق کسب‌وکار) وجود داشته باشد و در واقع این دو مهم در هم بافته[36] نشوند - و تحت تاثیر این پیچش، تغییر و خوانایی کد آسیب ببیند. چنین ملاحظاتی برای بار دیگر تنها در طراحی معنادار است و در فضای مسئله که UI فاقد موضوعیت است، حضور چنین کلاس‌هایی را نمی‌بینیم.

لازم به ذکر است در یک سناریوی واقعی کلاس‌های درون نمودار کلاسیِ طراحی، دارای تعداد زیادی عملیات[37] (و صفت)[38] خواهند بود که در نمودار تحلیل معادل آن، به ازای هر یک یا چند عملیات (و صفت) تنها یک عملیات (صفت) به صورت ادغام شده قرار خواهد داشت.

در مورد مدل تحلیل رسم شده در تصویر بالا، ما حتی جهت رابطه بین Transaction و Wallet هم رسم نکردیم. معمولا مقوله «جهت» در مواردی که اشاره‌گرهای سطح زبان مورد توجه ما است حائز اهمیت می‌شود که باز مربوط به فضای طراحی می‌باشد.

۲. (۵ نمره: مدلسازی)



[31] Delegation
[32] Single Responsibility Principle
[33] Cohesive
[34] Responsibility
[35] **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns - GRASP
[36] Intertwine
[37] Operation (AKA Method)
[38] Attribute (AKA Field)

تحت اثر الگوی تحلیل مزبور زین پس TransactionEntry (در مدل Entry نام دارد) بر موجودی کیف پول ما تاثیر خواهد داشت. Transaction در مدل ما تنها نگهدارنده‌ی دو TransactionEntry کنار یکدیگر است؛ ورودی اول مقداری پول از یک کیف‌پول (نوعا خریدار) کم کرده و ورودی دوم آن مقدار پول را به کیف‌پول مقصد (نوعا انباردار) منتقل می‌کند. در مورد اینکه تاریخ تراکنش در TransactionEntry قرار گیرد یا Transaction اگرچه می‌توان همچنان بحث نمود و تحت تاثیر شرایط مسئله تصمیم نهایی را گرفت؛ فعلا در این مدل، آن را در کلاس TransactionEntry قرار داده‌ایم.

یک مورد گوشه‌ای در شرایط مسئله وجود دارد که آن واریز پول به کیف‌پول است - در این حالت کاربر از درگاه بانکی مقدار پول از کارت بانکی خود که کسب‌وکار ما نسبت به آن دیدی ندارد خارج می‌کند و وارد کیف پول درون سیستم می‌کند. در این موارد ورودی به کیف‌پول یک طرفه است (پول از کیف‌پولی به کیف‌پول دیگر نمی‌رود). به همین منظور صلاح است شمارش[39] در سمت Transaction صفر یا یک باشد (که طبق قواعد UML نمایش آن اینطور خواهد بود: 0..1). **لازم به ذکر است اگر این مورد در پاسخ شما نادیده گرفته شد، بابت آن نمره‌ای از شما کسر نشده است.**

۳. (۱۰ نمره: ۸ نمره بهبود اول، ۲ نمره بهبود دوم)

بهبود اول مفهوم انتقال پول از یک حساب به حساب دیگر است که آن به خوبی با استفاده از این الگو مدل شده است. در مدل پیشین تحلیلمان کسر و افزایش پول از کیف‌پول‌های مختلف مشهود بود اما به خوبی معلوم نمی‌شد که وقتی پولی از حسابی کسر شده به کدام حساب افزوده شده است؟ ما ترجیح می‌دهیم مدل‌های تحلیلمان (که یک نمونه مهم از مدلسازی برای مهندسین نمودارهای کلاسی است) تا جای ممکن به دامنه کسب‌وکار نزدیک باشد. این نزدیکی، احتمال تاثیرگذاری کسب‌وکار در طراحی و پیاده‌سازی نهایی را بالا می‌برد. در ادامه می‌توان مانند خود دامنه کسب‌وکار یک سری صحت‌سنجی‌ها را در مورد وقایع انجام داد برای مثال اطمینان حاصل کرد به میزانی که پول از یک کیف پول خارج می‌شود حتما وارد حساب دیگر شود.

بهبود دوم استفاده همزمان از الگوی پول[40] در مدل‌هایمان است - این الگو به صورت عام‌تر تعداد[41] نام دارد. کلاس Money قرار داده شده ازین پس قابلیت پشتیبانی از واحدهای پول دیگر را به ما می‌دهد همچنین رفتارهای مرتبط با مسائل ساده‌ی مالی (مانند تبدیل واحدهای پول) ماژول تعریف شده‌ای برای خود دارد - یعنی همین کلاس Money.

---

[39] Cardinality
[40] Money
[41] Quantity

سوال ۵ (۲۰ نمره)

پنج مفهوم BPMN و CRC card و User Story و UML و DFD را از جنبه‌های زیر با یکدیگر مقایسه کنید:

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

پاسخ:

BPMN

- چه چیزی را مدل می‌کند؟
  - فرآیندهای موجود در دامنه‌ی کسب و کار[42] را مدل می‌کند (جنبه‌ی رفتاری)، اما در مورد ساختار اجزای موجود در سیستم (جنبه‌ی ساختاری) یا وظایفی که هر جز در سیستم انجام می‌دهد (جنبه‌ی وظیفه‌ای) صحبتی نمی‌کند.
- آن را چگونه مدل می‌کند؟
  - خود BPMN نشانه‌گذاری[43] خاصی را تعریف می‌کند که به flowchart و بعضی از نمودارهای UML شباهت دارد. با این نشانه‌گذاری فرآیندهای کسب و کار را مدل می‌کنند.
- کجا/در چه زمانی استفاده می‌شود؟
  - تحلیل‌گرها[44] از BPMN و برای مدل‌سازی فرآیندهای موجود در دامنه‌ی کسب و کار استفاده می‌کنند. این مدل‌سازی اصلاً جنبه‌ی فنی و technical ندارد و در فاز تحلیل (قبل از ورود به دامنه‌ی راه‌حل و دامنه‌ی پیاده‌سازی) انجام می‌شود.
- سطح انتزاع مدل‌سازی؟
  - BPMN می‌تواند هم به‌صورت ریزدانه نقش[45]های دخیل در هر فرآیند ریزدانه‌ی کسب و کاری و جریان[46] اجرای آن را مشخص کند، و هم به صورت درشت‌دانه کلیت یک فرآیند درشت‌دانه را نمایش دهد.

CRC card

- چه چیزی را مدل می‌کند؟
  - جنبه‌ی وظیفه‌ای[47] ارتباط کلاس‌های مختلف موجود در یک سیستم را مدل می‌کند؛ به این صورت که مشخص می‌کند هر کلاس **چه** وظایفی را انجام می‌دهد و برای این وظایف به چه

---

[42] Business
[43] Notation
[44] Analysts
[45] Actor
[46] Flow
[47] Functional

کلاس‌های دیگری تعامل دارد. توجه کنید که کارت‌های CRC در مورد ساختار یک کلاس (جنبه‌ی ساختاری[48]) صحبتی نمی‌کند. هم‌چنین در مورد **چگونگی** انجام وظایف (جنبه‌ی رفتاری[49]) نیز صحبتی نمی‌کند.

- آن را چگونه مدل می‌کند؟
  - به صورت نوشتاری، با مشخص‌کردن یک کلاس، وظایف آن و کلاس‌های دیگری که برای انجام این وظایف با آن‌ها تعامل می‌کند
- کجا/در چه زمانی استفاده می‌شود؟
  - در فاز طراحی (دامنه‌ی راه حل[50]) و برای مدل‌کردن چیستی ارتباط بین کلاس‌های مختلف با هم.
- سطح انتزاع مدل‌سازی؟
  - کارت‌های CRC در سطح کلاس (ریزدانه) مدل‌سازی می‌کنند.

User Story

- چه چیزی را مدل می‌کند؟
  - می‌توان **نیازمندی‌های مشتریان** را در قالب داستان کاربر یا همان User Story بیان کرد؛ در اصل User Story چیزی جز جملاتی قالب‌مند برای بیان نیازمندی‌ها نیست.

    داستان‌های کاربر جنبه‌ی وظیفه‌ای نیازمندی‌ها را مدل می‌کنند؛ به این صورت که چیستی و چرایی نیازمندی‌ها را مشخص می‌کنند (جنبه‌ی وظیفه‌ای)، اما در مورد ساختار یا چگونگی پیاده‌سازی آن نیازمندی صحبتی نمی‌کنند (جنبه‌های ساختاری و رفتاری).
- آن را چگونه مدل می‌کند؟
  - هر نیازمندی را در قالب یک جمله به صورت زیر مدل می‌کند:
    - به عنوان ..(۱).. می‌خواهم ..(۲).. تا ..(۳)..
      - مورد (۱) نقشی را که این نیازمندی به آن نیاز دارد، بیان می‌کند
      - مورد (۲) چیستی نیازمندی را مشخص می‌کند
      - مورد (۳) چرایی و هدف پشت نیازمندی را مشخص می‌کند
- کجا/در چه زمانی استفاده می‌شود؟
  - داستان‌های کاربر اغلب در فاز تحلیل، برای جمع‌آوری نیازمندی‌های مشتریان استفاده می‌شوند، و در طول فرآیند ایجاد نرم‌افزار نیز از آن‌ها برای مدل‌سازی و پیاده‌سازی به کار گرفته می‌شوند.
- سطح انتزاع مدل‌سازی؟

---

[48] Structural
[49] Behavioral
[50] Solution Domain

- داستان‌های کاربر می‌توانند در سطوح انتزاع مختلف قرار بگیرند؛ از نیازمندی‌های در سطح درشت‌دانه‌ی معماری گرفته تا نیازمندی‌های ریزدانه‌ی در سطح یک خصوصیت/قابلیت[51] نرم‌افزار

UML

- چه چیزی را مدل می‌کند؟
  - UML زبان مدل‌سازی قدرتمندی برای مدل‌سازی جنبه‌های مختلف ساختاری و رفتاری یک سیستم است. UML در مدل‌سازی جنبه‌ی وظیفه‌ای ضعف دارد.
- آن را چگونه مدل می‌کند؟
  - UML با نمودارهای مختلفی که دارد، مانند نمودار فعالیت، نمودار بسته[52]، نمودار حالت[53]، نمودار مورد کاربرد[54]، ...، جنبه‌های ساختاری و رفتاری مختلف سیستم را مدل می‌کند.
- کجا/در چه زمانی استفاده می‌شود؟
  - در فاز تحلیل و طراحی، از UML برای مدل‌سازی استفاده می‌شود و از مدل‌های ساخته شده به زبان UML در تمام طول فرآیند ایجاد نرم‌افزار، برای مواردی از جمله پیاده‌سازی و کدنویسی استفاده می‌شود.
- سطح انتزاع مدل‌سازی؟
  - UML با داشتن انواع نمودارهای مختلف، می‌تواند سطوح انتزاع مختلف را مدل‌سازی کند؛ برای مثال با استفاده از نمودار بسته (package diagram) می‌توان در سطح درشت‌دانه و با استفاده از نمودار کلاس (class diagram) می‌توان در سطح کلاس و ریزدانه مدل‌سازی را انجام داد.

DFD

- چه چیزی را مدل می‌کند؟
  - نمودار DFD جریان داده را در فرآیندهای یک سیستم مدل‌سازی می‌کند (جنبه‌ی وظیفه‌ای). توجه کنید که DFD در مورد ساختار داده‌ها در سیستم صحبتی نمی‌کند (جنبه‌ی سیستمی)، هم‌چنین در مورد چگونگی انتقال داده‌ها یا چگونگی ارتباط و ترتیب عملیات‌ها (جنبه‌ی رفتاری) نیز صحبتی نمی‌کند.
- آن را چگونه مدل می‌کند؟
  - این دسته از نمودارها با کمک علائم و نشانه‌گذاری‌های خاصی که تعریف کرده‌اند، جریان انتقال داده در فرآیندهای سیستم را مدل‌سازی می‌کنند.
- کجا/در چه زمانی استفاده می‌شود؟

---

[51] Feature
[52] Package Diagram
[53] State Diagram
[54] Use Case Diagram

- از DFD در فاز تحلیل و طراحی، برای فهم بهتر نیازمندی‌های مشتری استفاده می‌شود. از آن‌جایی که DFD به جریان داده‌ها می‌پردازد و نمودار UML در آن ضعف دارد، می‌توان گفت DFD و UML تکمیل‌کننده‌ی یکدیگر در مدل‌سازی جنبه‌های مختلف یک سیستم هستند.

- سطح انتزاع مدل‌سازی؟
  - نمودارهای DFD را می‌توان هم در سطوح ریزدانه برای مدل‌سازی جریان انتقال داده بین کلاس‌ها استفاده کرد، و هم می‌توان در سطح درشت‌دانه برای مدل‌سازی جریان انتقال داده بین مولفه‌های مختلف سیستم به‌کار برد.

سوال ۶ (۴۰ نمره)

روش طراحی ویژگی‌رانه[55](نسخه سوم) را به دقت مطالعه کنید و تحلیل خود را از این روش بر اساس موارد زیر بیان کنید.

- **مستندات معماری:** شامل تصمیمات، عقلانیت[56]، دیدهای معماری[57]، راه‌حل‌های جایگزین، بازنمایی[58] 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**

توجه کنید که پوشایی و دقت پاسخ شما، ملاک مهم ارزیابی در این سوال است.

پاسخ:

پاسخ در این لینک نوشته شده است.

[55] Attribute-driven Design Method
[56] Rationality
[57] Architectural Views
[58] Representation

پاسخ سوال ششم

«تمرکز این سوال بر روی تحلیل مبتنی بر معیار است. در این سوال، شما می‌بایست بر اساس معیارهای سطح بالای داده شده در صورت سوال، به تحلیل روش ADD پیرامون معیارهای داده شده می‌پرداختید. در روند تحلیل، شما با زیرمعیارهای زیادی مواجه می‌شوید که لازم است تا جایی که در محدوده کتاب مرجع این درس است، در روش مورد نظر عمیق شوید. هدف اصلی، پوشایی و دقت تحلیل شماست. از منظر پوشایی، جوانب مختلف هر کدام از معیارها(تشخیص زیرمعیارهایی که همه جوانب را پوشش می‌دهند) بایستی در پاسخ‌های شما وجود داشته باشد. از منظر دقت، انتظار می‌رود که هر یک از این جوانب و معیارها و زیرمعیارها، به درستی تحلیل شده باشند. بنابراین پاسخی که در ادامه برای این سوال در نظر گرفته شده است، نمونه‌ای برای تحلیل روش ADD از این دو منظر و پیرامون معیارهای خواسته شده در صورت سوال است.»

«دقت کنید که در این سوال تنها معیارهای داده شده مهم است. برای هر روش معیارهای زیادی وجود دارد که در این تمرین و متناسب با مرجع درس، همان معیارهای درشت‌دانه داده شده، جزو معیارهای قابل قبول است و مواردی مانند بررسی فرایند این روش مورد توجه سوال نیست. ***هدف این سوال یادگیری ADD نیست. بلکه افزایش توانایی شما در تحلیل روش‌ها پیرامون معیارهای داده شده و رسیدن به درک درستی از معیارهای مطرح شده است.»***

روش طراحی ویژگی‌رانه با تمرکز بر تحقق نیازمندی‌های مهم معماری و به ویژه ویژگی‌های کیفی، فعالیت دوم چرخه عمر ایجاد معماری نرم‌افزار[1] را در یک فرایند تکراری با گام‌های مشخص و متوالی در هر تکرار، محقق می‌کند. دغدغه اصلی این روش، برقراری ویژگی‌های کیفی سیستم است و واژه Attribute در نام این روش، در واقع به Quality Attribute اشاره دارد. به طور خلاصه در این روش، طراحی معماری در دورهای طراحی[2] صورت می‌گیرد و در هر دور، یک یا چند تکرار رخ می‌دهد. فازهای مربوطه در هر تکرار انجام می‌شوند تا هدف طراحی را برآورده سازند. از آنجا که بررسی فرایند این روش مورد توجه سوال نیست، از بحث درباره آن پرهیز می‌کنیم و در ادامه، به بررسی این روش در مواجهه با معیارهای خواسته شده در صورت سوال می‌پردازیم.

فهرست مطالب

[1] Software Architecture Development Lifecycle (SADLC)

[2] Design Cycle

بررسی مستندات معماری در روش طراحی ویژگی‌رانه

دسته اول از معیارهای مورد توجه سوال بر مستندات معماری است. بنابراین ابعاد مختلف آن را یافته و تحلیل می‌کنیم.

زیرمعیارهای مستندات معماری

ابتدا به مواردی که کتاب پرسمن در جهت مستندات معماری اشاره کرده می‌پردازیم تا براساس آنها بتوانیم میزان پوشایی مستندات معماری و معیارهای مربوطه را در روش ویژگی‌رانه بیابیم. در دسته مستندات معماری مجموعه معیارهای زیر مورد توجه‌اند:

- **زبان توصیف معماری**
- **تصمیمات معماری**
- **راه‌حل‌های در نظر گرفته شده برای معماری در طول عمر سیستم**
- **عقلانیت تصمیمات**
- **دیدهای معماری**
- **محدودیت‌ها**
- **ریسک‌ها**

تحلیل زیرمعیارها

زیرمعیارهایی را که در بخش قبلی یافتیم، در این بخش به ترتیب مورد بررسی قرار می‌دهیم.

زبان توصیف معماری

در ابتدا باید گفت که روش ویژگی‌رانه به طور مفصل به فعالیت مستندسازی معماری نمی‌پردازد و خود این روش نیز در مستندات آن به این اذعان دارد. همانطور که گفته شد، این روش، فعالیت دوم چرخه عمر ایجاد معماری را پوشش می‌دهد. فعالیت اول نیازمندی‌های معماری، فعالیت دوم طراحی معماری و فعالیت سوم مستندسازی معماری است. این روش بخشی از فعالیت‌های اول و سوم را دربردارد اما تمرکز آن بر روی طراحی معماری است؛ در نتیجه به مستندسازی معماری به صورت رسمی و مفصل نپرداخته است. اولین نکته‌ای که در این روش به چشم می‌خورد، این است که برای مستندسازی قالب و فرمت مشخصی ارائه نمی‌دهد و حتی زبان مدل‌سازی و نمادگذاری را نیز تعیین نکرده است. هر چند استفاده از UML در Case-Study ها مورد توجه است، اما خود روش بر آن تاکیدی ندارد. از آنجا که این روش، قالب و فرمت به خصوصی ندارد، در نتیجه از زبان توصیف معماری خاصی نیز استفاده نمی‌کند. عدم مشخص بودن فرمت یا حتی نمادگذاری بدین معناست که خط و مستطیل‌های ساده یا قواعد UML هر دو می‌توانند در این روش به‌کارگرفته شوند. برای مثال می‌توان به این نکته توجه کرد:

“No UML class diagram will help you reason about schedulability, nor will a sequence diagram tell you very much about the system’s likelihood of being delivered on time. You should choose your notations and representation languages while keeping in mind the important issues you need to capture and reason about.”

بنابراین هر چند این روش، خود نوع مشخص از قالب و فرمت را ارائه نداده است، اما بر اتخاذ آن تاکید دارد و تیم‌ها می‌بایست متناسب با نیاز خود، نوع زبان خود را انتخاب کنند.

تصمیمات معماری، عقلانیت تصمیمات و راه‌حل‌های جایگزین

در فاز ششم این روش که به ثبت مستند از موارد متعدد پرداخته شده است، ثبت تصمیمات معماری را مورد توجه قرار داده است. همانطور که در توضیحات این روش آمده است، در جریان طراحی، معمار برای رفع یک یا چند ویژگی کیفی، باید از تکنیک‌ها و الگوها بهره برده و باید آن‌ها را از جهت پوشش نیازمندی‌های معماری بررسی کند. در جریان این فرایند، معمار باید مزایا و معایب هر کدام از گزینه‌ها را بررسی کند و تصمیم بگیرد که از کدام راه‌حل بهره ببرد. در جریان انتخاب راه‌حل، اولویت‌های سازمان، ریسک‌ها و محدودیت‌های موجود نیز در تصمیم‌گیری معمار تاثیرگذارند. بنابراین معمار با بررسی دقیق این موارد، یک گزینه را انتخاب می‌کند. مجموعه دلایل و گزینه‌های پیش‌روی معمار و مزایا و معایب و بررسی نهایی وی، عقلانیت تصمیمات او را تشکیل می‌دهد. روش ADD تاکید می‌کند که باید این موارد ثبت و ضبط شوند تا در آینده و برای تصمیمات بعدی یا تغییرات در ساختارهای انتخاب شده به کار گرفته شوند. ثبت راه‌حل‌های جایگزین به تیم کمک می‌کند تا در ارزیابی و یا تغییرات آینده، گزینه‌ها را بررسی کند و در صورت تصمیم به تغییر برخی از تصمیمات، تاثیرات و عواقب آن را به سایر ذی‌نفعان اطلاع دهد. بنابراین ثبت این موارد بسیار مهم است.

دیدهای معماری

در ابتدا بهتر است تعریف دقیقی از دید معماری ارائه کنیم. دید، بازنمایی از نوع خاصی از مجموعه ساختارهای سیستم و روابط میان آنهاست. در واقع مستندسازی معماری، شامل دیدهای معماری و مجموعه مستنداتی است که برای بیش از یک دید به کار می‌روند. این روش 3 دید معماری اصلی و دیدهای معماری متعدد از نوع جنبه‌های کیفی را مطابق نیاز سیستم، مورد توجه قرار داده است. دیدهای آن عبارتند از:

1. دید پیمانه[3]: یک پیمانه از سیستم بیانگر یک واحد پیاده‌سازی از سیستم است که مجموعه منسجم از مسئولیت‌ها را بر عهده دارد و دید پیمانه، شامل مجموعه‌ای از ویژگی‌هایی است که به این پیمانه نسبت داده شده‌اند. بنابراین مطابق این روش، در یک سیستم به تعداد پیمانه‌های موجود در سیستم، دید پیمانه وجود دارد. ویژگی‌های موجود در دید پیمانه، دربرگیرنده اطلاعات مهمی است که به خود پیمانه، روابط آن با دیگر پیمانه‌ها و محدودیت‌های اعمال شده بر پیمانه بستگی دارد. سطح دسترسی، مسئولیت‌ها، تاریخچه بازبینی، روابط وراثت و وابستگی نمونه‌هایی از این ویژگی‌ها هستند.

[3] Module View

2. دید مولفه و رابط[4]: عناصری از سیستم که به زمان اجرای سیستم مربوط می‌شوند، در این دید جای می‌گیرند، مانند پردازه‌ها، سرویس‌ها، اشیا و ... . این موارد مولفه نامیده می‌شوند. عناصری که مسیرهای ارتباطی بین مولفه‌ها را برقرار می‌سازند، مانند پروتکل‌ها، جریان‌های اطلاعاتی و ... رابط نامیده می‌شوند و در این دید جای می‌گیرند. هر مولفه در دید مولفه و رابط، دربرگیرنده یک زیرسیستم پیچیده است که به صورت تکرارشونده شامل زیرسیستم‌های دیگر می‌تواند باشد که هر کدام، می‌توانند زیرمعماری متفاوتی داشته باشند.
3. دید تخصیص[5]: در دید تخصیص، تناسب بین واحدهای نرم‌افزاری با عناصر محیطی که نرم‌افزار قرار است در آن عملیاتی شود، مشخص می‌گردد. این محیط می‌تواند یک سخت‌افزار، سیستم‌عامل، فایل‌های سیستمی و یا سازمان مربوطه باشد.
4. دیدهای کیفی: دیدهای قبلی، همگی از نوع ساختاری هستند. این دیدها ساختار سیستم را برای تحقق اهداف معماری پشتیبانی می‌کنند. اما در سیستمی که ویژگی‌های کیفی اهمیت بالایی دارند، نیاز ذی‌نفعان با چنین دیدهایی برطرف نمی‌شود. در نتیجه دیدهایی کیفی که ویژگی‌های کیفی مورد توجه را پشتیبانی می‌کنند، مورد نیاز است. مانند دیدهای امنیت، ارتباطات، مدیریت خطا، اتکاپذیری و ... .

محدودیت‌ها و ریسک‌ها

با دقت در مطالعه مستندات این روش، می‌توان توجه روش ویژگی‌رانه را به این دو مورد به خوبی درک کرد. این روش، ریسک‌ها و محدودیت‌ها را به عنوان ورودی مورد توجه قرار داده است و اهمیت دستیابی به آن را به خوبی با قرار دادن پیش‌نیاز شروع فاز طراحی مشخص کرده است. با اینکه ریسک به طور مشخص به عنوان به ورودی مشخص نشده است، اما تصمیمات فرایندی این روش، انتخاب ساختارها و برگزیدن راه‌حل‌ها و حتی زمان پایان فرایند طراحی، به ریسک وابسته شده است. در انتخاب معماری مرجع و ساختارها، تکنیک‌ها و الگوهای مرتبط با نیازمندی‌های هر ساختار و کل سیستم، این موارد مهم شمرده شده‌اند. حتی در انتخاب نیازمندی‌های کارکردی مهم، این ریسک‌ها هستند که تاثیرگذارند و به طور غیرمستقیم عمل می‌کنند. بنابراین لازم است راهکارهای لازم برای مقابله با ریسک‌ها و محدودیت‌های پروژه ثبت شوند تا اطمینان از قابلیت اجرایی سیستم بالا رود. همانطور که در مستند این روش آمده است:

“What are the criteria for evaluating if more design iterations are necessary? We let ***risk*** be our guide.”

[4] Component and Connector View
[5] Allocation View

بررسی نگرانی‌های انواع ذی‌نفعان در روش طراحی ویژگی‌رانه

دسته دوم از معیارهای مورد توجه سوال بر نگرانی‌های انواع ذی‌نفعان است. بنابراین ابعاد مختلف آن را یافته و تحلیل می‌کنیم.

زیرمعیارهای نگرانی‌های انواع ذی‌نفعان

زیرمعیارهای مورد توجه کتاب در فصل‌های 9 و 10 عبارتند از:

- **تیم ایجاد**
- **مدیران و متخصصان مارکتینگ**
- **مدیر پروژه**
- **طراحان معماری**
- **تیم نگهداری**
- **کاربران نهایی**

تحلیل زیرمعیارها

زیرمعیارهایی را که در بخش قبلی یافتیم، در این بخش به ترتیب مورد بررسی قرار می‌دهیم. یکی از اهداف معماری، امکان برقراری ارتباط با ذی‌نفعان و حل نگرانی‌ها و اهدافی است که ذی‌نفعان مورد توجه قرار داده‌اند. اولین نقطه‌ای که این روش به نگرانی‌های مرتبط با ذی‌نفعان می‌پردازد، در تعیین اهداف اصلی سیستم و شناخت نیازمندی‌های کیفی مورد توجه آنان است که به طور غیر مستقیم این روش در آن دخالت دارد. این روش ویژگی‌های کیفی و نیازمندی‌های وظیفه‌ای اصلی را به عنوان ورودی قرار داده است که در آنها نگرانی‌های مختلف ذی‌نفعان در نظر گرفته شده است. در طول فرایند طراحی نیز، بازبینی‌های متعدد قرار گرفته است تا تغییرات در این نیازمندی‌ها را بر معماری اعمال شوند. در نتیجه به طور کلی به نیازهای ذی‌نفعان مختلفی که در فرآیند پیاده‌سازی دخیل نیستند، مانند کاربران نهایی، مدیران پروژه و متخصصان مارکتینگ، پاسخ داده است. از آنجا که پیاده‌سازی معماری توسط تیم ایجاد انجام می‌شود، مستنداتی که در طول طراحی آماده می‌شوند با هدف شناخت و درک آن‌ها از معماری است. اما این روش به طور کلی، درباره پیاده‌سازی معماری صحبتی نمی‌کند و در نتیجه مشکلاتی که در طول پیاده‌سازی ممکن است به وجود آید، مورد توجه این روش نیست. اما طراحان و معماران سیستم، نگرانی‌هایی از جمله چگونگی فرایند ساخت و ادغام و استقرار را در طول طراحی در نظر گرفته‌اند. نیازمندی‌های کیفی تنها به موارد مربوط به سیستم مانند عملکرد، مدیریت خطا، مقیاس‌پذیری محدود نیست؛ بلکه شامل چگونگی ساختار تیم‌ها، ارتباط افراد درون تیم با یکدیگر و سرعت تیم ایجاد نیز هست. بنابراین معماران به این جنبه‌ها نیز در طول طراحی می‌پردازند و نیازمندی‌های کیفی که به عنوان ورودی در نظر گرفته شده است، شامل این موارد نیز هست. رویکرد این تیم در مواجهه به تیم نگهداری نیز، به مانند تیم ایجاد است و همه‌ی اشکالات و مزایای گفته شده برای تیم ایجاد، برای تیم نگهداری نیز کاربرد دارد.

پاسخ مهم‌تری که در راستای رفع نگرانی‌های ذی‌نفعان وجود دارد، ارائه مستندات به آنهاست و باید بررسی کرد که مستندات تهیه شده در طول این روش، چگونه به دغدغه‌های افراد پاسخ می‌دهد. این روش برای هرکدام از دسته ذی‌نفعان زیر، نحوه پاسخدهی به دغدغه‌های آنها را مشخص کرده است. مشخصا چون متخصصان مارکتینگ با معماری ارتباط غیرمستقیمی دارند، بررسی چگونگی تحقق اهداف آنها، کافی است. بنابراین سایر موارد را مورد بررسی قرار می‌دهیم.

تیم ایجاد

نگرانی‌ها و نیازمندی‌های تیم ایجاد شامل ایده کلی، عناصری که هر عضو تیم مسئول پیاده‌سازی آن است و جزئیات مربوط به آن مانند کدها و محدودیت‌ها است. دیدهای معماری سه گانه در رفع این نیازمندی‌ها و نگرانی‌ها کمک‌کننده هستند.

مدیران پروژه

مدیران پروژه، نگرانی‌هایی از قبیل زمان‌بندی، تخصیص منابع و برنامه انتشار بخش‌هایی از سیستم دارند. آنها به ساختار همه عناصر نیازی ندارند، اما به معماری کلی سیستم و زیرسیستم‌ها، ارتباطات آنها با یکدیگر و با سیستم‌های بیرونی و محیط‌های عملیاتی سیستم برای تحقق اهداف و رفع نگرانی‌ها نیاز دارند. بنابراین دیدهای پیمانه و تخصیص می‌تواند به آنها در رفع این نگرانی‌ها کمک‌کننده باشد.

طراحان معماری

طراحان معماری ذی‌نفع سیستم، بیشتر به دو دسته معماران آینده و معماران سایر سیستم‌هایی که با سیستم مورد نظر کار می‌کنند، تقسیم می‌شوند. در مورد دسته اول، معماران آینده به همه داده‌های موجود در سیستم مانند تصمیمات معماری، عقلانیت تصمیمات و محدودیت‌ها و ریسک‌ها نیاز دارند تا بتوانند در مورد علت آنها و تغییرات مورد نظر خود ارزیابی انجام دهند و مشکلات احتمالی را شناسایی کنند. در مورد دسته دوم نیز دیدهای موفه و رابط که نحوه ارتباطات را مشخص کرده‌اند، کمک زیادی می‌کند. اگر مستندات دیگری درباره مدل داده‌ها و یا نماهای سیستم وجود دارد، خوب است که در اختیار آنها قرار بگیرد. چرا که نماها در روش ویژگی‌رانه مورد توجه هستند و طراحی آنها بخشی از فعالیت‌های اصلی در فرایند پیاده‌سازی است.

تیم نگهداری

تیم نگهداری به همان مواردی که تیم ایجاد نیاز داشت، نیاز دارند. اما علاوه بر آن، آنها به دید تجزیه از سیستم که به آنها اجازه اعمال تغییرات خود را در بخش‌های مختلف می‌دهد نیز نیاز دارند. آنها همچنین به عقلانیت تصمیمات طراحی و راه‌حل‌های جایگزین نیاز دارند تا بتوانند علت عدم استفاده از آنها را درک کنند و در مورد تصمیمات خود در طول تغییرات بر سیستم، دچار خطا نشوند.

کاربران نهایی

با اینکه کاربران نهایی نیازی به معماری سیستم ندارند و به طور کلی اصلا آن را نمی‌بینند، اما مشاهده معماری توسط آنها یا نماینده آنها می‌تواند به درک بهتری از عملیات‌های سیستم و امکانات آن می‌دهد و آنها می‌توانند نحوه چگونگی اعمال دستورات را مشاهده کنند و استفاده بهتری از سیستم داشته باشند. بنابراین دیدهای مولفه و رابط و تخصیص می‌تواند به آنها در درک جریان انتقال اطلاعات سیستم و نحوه تبدیل ورودی به خروجی کمک کند.

بررسی مفاهیم طراحی در روش طراحی ویژگی‌رانه

دسته سوم از معیارهای مورد توجه سوال بر مفاهیم طراحی است. بنابراین ابعاد مختلف آن را یافته و تحلیل می‌کنیم.

زیرمعیارهای مفاهیم طراحی

با دقت و تمرکز بر فصل‌های 9 و 10، به زیرمعیارهای زیر برای کاربرد و تاثیر مفاهیم طراحی در روش‌های طراحی معماری می‌رسیم:

- **الگوهای معماری**
- **سبک‌های معماری**
- **ساختارهای معماری**
- **چارچوب‌های معماری**
- **الگوهای طراحی**
- **توجه به موارد کیفی و اصول طراحی**
    - موارد 1 تا 8 بخش 9.2.1 کتاب

تحلیل زیرمعیارها

زیرمعیارهایی را که در بخش قبلی یافتیم، در این بخش مورد بررسی قرار می‌دهیم. در مورد این معیار، به بررسی تک تک زیرمعیارها نمی‌پردازیم؛ بلکه آنها را در کنار هم ارزیابی می‌کنیم و تنها زیرمعیار آخر را به طور جداگانه مورد بررسی قرار خواهیم داد.

الگوهای طراحی و چارچوب‌ها، سبک‌ها، الگوها و ساختارهای معماری

روش ویژگی‌رانه مبتنی بر ویژگی‌های کیفی و تهیه‌ی ساختارهایی برای پاسخ به تصمیماتی است که در مورد این ویژگی‌ها گرفته می‌شود که این ساختارها، شامل تکنیک‌ها و سبک‌ها و الگوها هستند. این روش برای شروع طراحی معماری، استفاده از چارچوب‌های معماری و یا معماری‌های مرجع را برای دامنه‌های بالغ ضروری می‌داند. زمانی که می‌خواهید به کمک این روش طراحی معماری را شروع کنید و اگر در دامنه بالغی هستید، بهتر است ابتدا گزینه‌های مربوط به چارچوب معماری را انتخاب کنید و از میان آنها، با توجه به سایر موارد مانند ریسک و محدودیت، یکی را برگزیده و سایر مراحل را طی کنید. در نتیجه این روش به خوبی از چارچوب‌های معماری پشتیبانی می‌کند. البته استفاده از چارچوب، مربوط به نسخه 2.5 و 3 از این روش است و در نسخه‌های قبلی وجود نداشت. سپس باید برای هر تحقق ویژگی‌های کیفی و پیاده‌سازی عناصر مختلف، تکنیک‌ها و سبک‌هایی را انتخاب کرده و برای هر کدام الگوها را مشخص کنید و ساختارهای لازم برای تحقق این موارد را برگزینید تا هدف طراحی محقق شود. بنابراین این روش به خوبی از مفاهیم طراحی مختلف مانند چارچوب تکنیک، سبک، الگو و

ساختارها پشتیبانی می‌کند. اما باید دقت کرد که الگوهای طراحی، برای طراحی سطح پایین‌تر به کار می‌روند و در معماری مورد استفاده نیستند. در نتیجه الگوهای طراحی در این روش جایی ندارند. مگر اینکه برای مدیریت ریسک، مجبور باشیم طراحی معماری را تا سطوح پایین‌تری دنبال کنیم که در آن صورت می‌توان برای حل مسائل، از الگوهای طراحی بهره برد.

توجه به موارد کیفی و اصول طراحی

مورد اول درباره استفاده از الگوها و سبک‌ها، تشکیل شدن از مولفه‌ها و تکاملی بودن طراحی است. این مورد به طور کامل در روش ویژگی‌رانه برقرار است. با توجه به توضیحاتی که تا الان داده شد، دو مورد اول مشخص است. مورد سوم در روند و فرایند این روش یافت می‌شود. فرایند روش ویژگی‌رانه تکراری-افزایشی است و معماری به صورت تکاملی و طول تکرارها کامل می‌شود.
مورد دوم درباره پیمانه‌ای بودن طراحی است. تمرکز روش ویژگی‌رانه بر این بخش نیز در بخش‌های قبل داده شده است.
مورد سوم درباره وجود بازنمایی‌هایی از سیستم درباره طراحی، داده، نماها و مولفه‌هاست. سه مورد مولفه، نما و معماری در روش ویژگی‌رانه وجود دارد. اما در مورد داده، حرف دقیقی درباره آن نزده است. جریان‌های اطلاعاتی در روش ویژگی‌رانه و در طی دیدهای کیفی قابل مشاهده است و ساختار داده‌ها نیز در طول طراحی معماری همواره مشخص می‌گردد. به همین علت نمی‌توان این مورد را نقدی بر روش ویژگی‌رانه دانست.
مورد چهارم درباره الگوهای داده است. از آنجا که روش ویژگی‌رانه از الگوها پشتیبانی می‌کند، می‌توان در موارد نیاز از آنها نیز استفاده کرد.
مورد پنجم درباره مولفه‌هاست و در روش ویژگی‌رانه برقرار است.
مورد ششم درباره نماها و کاستن از پیچیدگی ارتباطات است که در روش ویژگی‌رانه مورد توجه‌اند. گام پنجم از فرایند روش ویژگی‌رانه به تعریف نماهای ساختارها و عناصر می‌پردازد.
مورد هفتم درباره تکرارپذیر بودن روش طراحی و استفاده نیازمندی‌ها در طول طراحی است. این موضوع به خوبی درباره روش ویژگی‌رانه برقرار است. روش ویژگی‌رانه به عنوان ورودی، نیازمندی‌های وظیفه‌ای مهم و کیفی را قرار داده است تا بر اهمیت آنها بر طراحی معماری تاکید کند و همچنین ساختارها و روش‌های پیشنهادی برای آنها نیز دارد. برای مثال فرمت و قالب پیشنهادی سناریو ویژگی کیفی[6] را برای نیازمندی‌های کیفی ارائه داده است و روش کارگاه ویژگی کیفی[7] را برای استخراج نیازمندی‌های کیفی پیشنهاد کرده است. در مورد تکرارپذیر بودن روش، باید گفت که تکرارپذیری اصل اول در ایجاد یک روش است. در مقدمه معرفی روش ویژگی‌رانه و در بخش دلیل نیاز و ایجاد این روش، به این موضوع اشاره شده است که طراحی معماری، عمدتا توسط افراد باتجربه و ارشد صورت می‌گرفت. اما با وجود یک روش تکرارپذیر و اثبات شده و قابل آموزش، می‌توان طراحی معماری را در سایر سطوح ارشدیت نیز انجام داد. روش ویژگی‌رانه با هدف پیروی از این سه اصل ایجاد شد و به خوبی تکرارپذیر است.

[6] Quality Attribute Scenario (QAS)
[7] Quality Attribute Workshop (QAW)

مورد هشتم درباره بازنمایی به کمک نمادها برای برقراری ارتباط است و همانطور که در مستندات معماری بحث شد، این روش توصیه به نمادگذاری دارد و دیدهایی برای بازنمایی معماری در نظر گرفته است، اما نمادهای خاصی را پیشنهاد نمی‌کند و معمار نرم‌افزار در این مورد آزادی عمل دارد.