

## نام اعضای تیم و شماره دانشجویی ها

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 فناوریانه 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 نرم افزار مقایسه کنید.
۲. رشته های معماری ساختمان و معماری نرم افزار چه شباهت هایی دارند؟ چه تفاوت هایی دارند؟

## جواب سوال ۲

### مقایسه معماری خانه/ساختمان با معماری نرم افزار

معماری ساختمان و معماری نرم افزار، هر دو فرایندهای برنامه ریزی، طراحی و سازماندهی هستند که برای ایجاد یک محصول نهایی پیچیده و کاربردی استفاده می شوند. در هر دو حوزه، معمار باید مجموعه ای از الزامات و نیازمندی ها را در نظر بگیرد، راه حل های مختلف را بررسی کند، و ساختاری منطقی و کارآمد را تعریف کند.

#### شباهت ها:

- برنامه ریزی و طراحی: هر دو نیازمند فرایندی برای تعیین نیازمندی ها، محدودیت ها، و هدف های 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.

#### تفاوت ها:

- ماهیت محصول: محصول نهایی در معماری ساختمان فیزیکی و در معماری نرم افزار مجازی است.
- روند توسعه: معماری نرم افزار اغلب شامل فرایندهای تکراری و انعطاف پذیر است، در حالی که ساختمان ها معمولاً بر اساس طرح های نهایی و دقیق ساخته می شوند.
- تغییر و نگهداری: نرم افزارها معمولاً برای تغییر و به روزرسانی طراحی می شوند، در حالی که ساختمان ها به ندرت برای تغییرات عمده طراحی می شوند.

### شباهت ها و تفاوت های رشته های معماری ساختمان و معماری نرم افزار

#### شباهت ها:

- 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 راه حل های 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 نرم افزاری 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 مدل سازی ذیل هر فعالیت

## جواب سوال ۳

### ارتباط با مسئله و راه حل

**تحلیل:** در تحلیل نرم افزار، مسئله مورد بررسی قرار می گیرد. هدف این است که دقیقاً تعریف کنیم مسئله چیست و چه نیازهایی باید توسط نرم افزار برآورده شود.

**طراحی:** در مرحله طراحی، راه حل های ممکن برای مسائل تحلیل شده مطرح می شوند. این مرحله شامل تعیین چگونگی عملکرد نرم افزار برای برآورده کردن نیازهای شناسایی شده است.

### اهداف و تمرکز

**تحلیل:** تمرکز در تحلیل بر روی شناسایی و فهم نیازمندی های کاربر و مشخص کردن آنچه سیستم باید انجام دهد، است.

**طراحی:** هدف از طراحی ایجاد یک معماری قابل اجرا برای نرم افزار است که نیازمندی های 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 پایین تر قرار دارد و بیشتر به جزئیات "چگونه" می پردازد.

### تقدم و تاخر

**تحلیل:** معمولاً قبل از طراحی انجام می شود. ابتدا باید مسائل و نیازمندی ها را درک کرد.

**طراحی:** پس از تحلیل انجام می شود و بر اساس نتایج به دست آمده از تحلیل، راه حل ها طراحی می شوند.

## تفاوت در مدل سازی

**تحلیل:** مدل سازی در تحلیل بر روی نمایش نیازمندی ها و فرایندهای کسب و کار تمرکز دارد.

**طراحی:** در طراحی، مدل سازی به توصیف معماری سیستم، کلاس ها، اشیاء، و روابط بین آنها می پردازد.

## 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 مدل سازی عمده در مهندسی نرم افزار - CRC Card ، User Story ، UML ، DFD و BPMN - مورد بررسی و مقایسه قرار دهید از جنبه های مختلف.

- چه چیزهایی را مدل می کنند
- آن ها را چگونه مدل می کنند
- کجا/در چه زمانی استفاده می شوند
- تفاوت سطح انتزاع در مدل سازی

## جواب سوال ۵

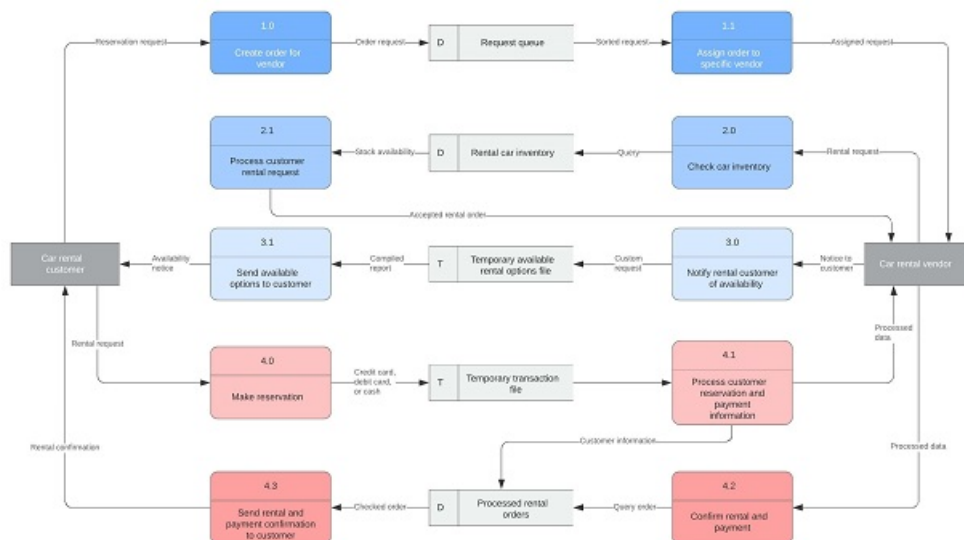
در مهندسی نرم افزار، مدل سازی فرایندی است که به منظور ایجاد یک نمایش گرافیکی یا نمادین از یک سیستم، فرایند یا مفهوم انجام می شود. مدل سازی می تواند برای اهداف مختلفی استفاده شود، از جمله:

- 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 مدل سازی در مهندسی نرم افزار را تحلیل و مقایسه می کنیم.

## DFD (Data Flow Diagram)

DFD ، که مخفف Data Flow Diagram است، یک ابزار مهم در مهندسی نرم افزار برای نمایش جریان داده ها درون یک سیستم است. این نمودار به تحلیل گران و طراحان کمک می کند تا درک عمیق تری از سیستم ها داشته باشند.

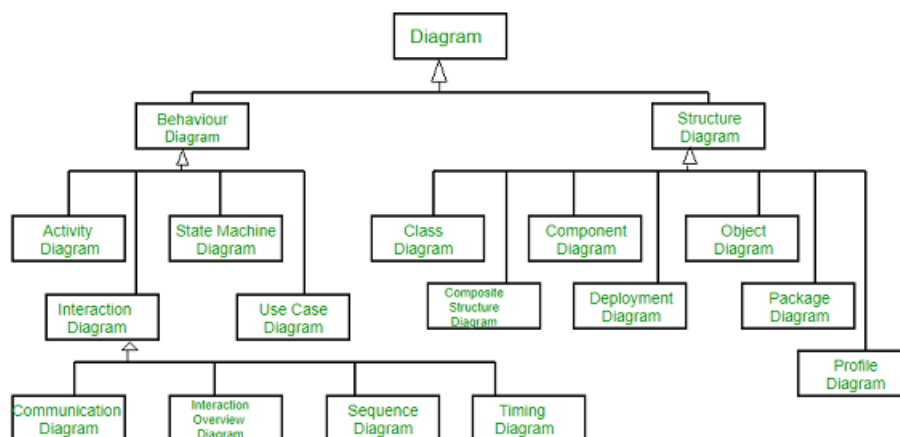
- کاربرد: DFD برای نمایش جریان داده ها بین فرآیندها، داده ها، سازمان ها و ذخیره سازی ها در سیستم های نرم افزاری استفاده می شود.
- ساختار: DFD شامل گره ها و خطوط است. گره ها نشان دهنده فرآیندها، مخازن داده، منابع داده و مقاصد داده هستند، و خطوط جریان داده ها را نشان می دهند.
- سطح انتزاع: DFD در سطوح مختلف انتزاع مورد استفاده قرار می گیرد.
- مزایا: این ابزار به شناسایی مسیرهای داده و تحلیل چگونگی انجام کار توسط سیستم کمک می کند.



## UML (Unified Modeling Language)

UML ، که مخفف Unified Modeling Language است، یک زبان استاندارد برای مدل سازی و توصیف ساختار و رفتار سیستم های نرم افزاری است. این زبان از طریق استفاده از مجموعه ای متنوع از نمودارها کاربرد دارد.

- **انتزاع در UML :** یکی از ویژگی های برجسته UML توانایی آن در نمایش سطوح مختلف انتزاع است.
- **کاربرد UML :** UML در تمام مراحل توسعه نرم افزار مورد استفاده قرار می گیرد.



## User Story

User Story در مهندسی نرم افزار به عنوان یک تکنیک مدل سازی برای تعریف نیازمندی ها و رفتار کاربران در سیستم استفاده می شود. این روش متمرکز بر بیان خواسته ها و نیازهای کاربران از طریق سناریوهای کوتاه و روشن است.

- **سطح انتزاع:** User Story نسبت به سایر روش های مدل سازی، 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.

# User Story

**As a <role>  
I want <goal>  
so that <benefit>**

**Acceptance criteria:**  
(Conditions of Satisfaction)

...  
...

**As an** Account Manager  
**I want** a sales report of my account  
to be sent to my inbox daily  
**So that** I can monitor the sales  
progress of my customer portfolio

**Acceptance criteria:**

1. The report is sent daily to my inbox
2. The report contains the following sales details: ...
3. The report is in csv format.

## 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.

# CRC Cards

CRC cards are a notational device to record information about a class, what it must do and with whom it must collaborate

Class Name:	Superclass:	Subclasses:
Responsibilities		Collaborations

## BPMN

BPMN مخفف Business Process Model and Notation به معنای مدل و نشانه گذاری فرآیند کسب و کار است. این یک زبان مدل سازی بصری برای برنامه های تجزیه و تحلیل کسب و کار و مشخص کردن گردش کار فرایندهای سازمانی است. BPMN توسط ابتکار مدیریت فرآیند کسب و کار (BPMP) توسعه یافت و از زمان ادغام دو سازمان در سال ۲۰۰۵ توسط گروه مدیریت اشیاء (OMG) حفظ شده است.

### عناصر اصلی

BPMN از مجموعه ای از عناصر بصری برای مدل سازی فرایندهای کسب و کار استفاده می کند. این عناصر عبارتند از:

- فعالیت ها (Activities): فعالیت ها اقداماتی هستند که در یک فرآیند انجام می شوند. فعالیت ها می توانند شامل کارهای فیزیکی، پردازش اطلاعات یا تصمیم گیری باشند.
- جریان ها (Flows): جریان ها نحوه ارتباط فعالیت ها را نشان می دهند. جریان ها می توانند به صورت خط مستقیم، خط نقطه چین یا خط مورب نشان داده شوند.
- شروع و پایان (Start and End): شروع و پایان نشان دهنده نقاط شروع و پایان یک فرآیند هستند.
- کنترل ها (Controls): کنترل ها شرایطی را که بر جریان فرآیند تأثیر می گذارند، نشان می دهند.
- پیوندها (Links): پیوندها فعالیت ها یا جریان های مختلف را به هم متصل می کنند.

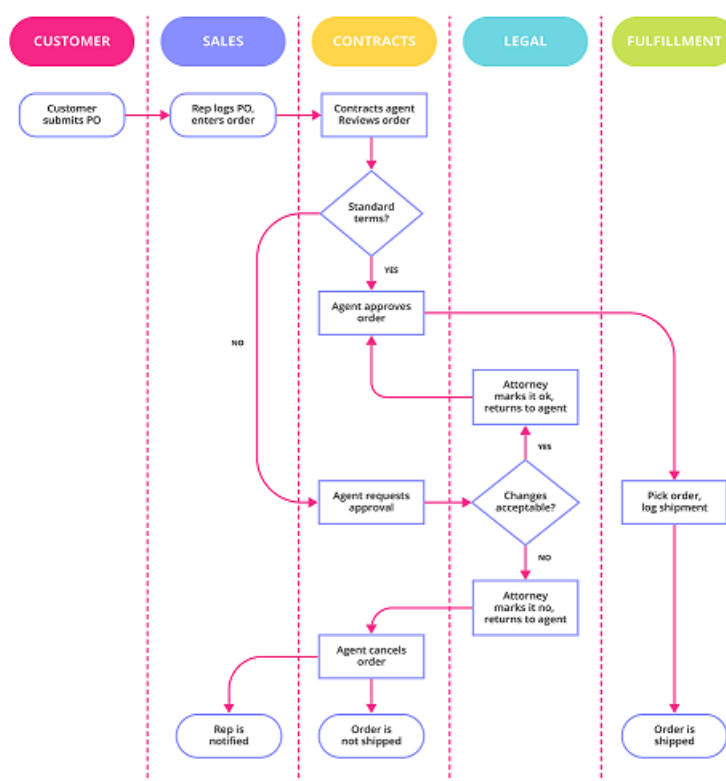
### کاربردهای BPMN

BPMN در طیف گسترده ای از کاربردها استفاده می شود، 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. (2010). '9 best business process modeling 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 ویژگی رانه (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.

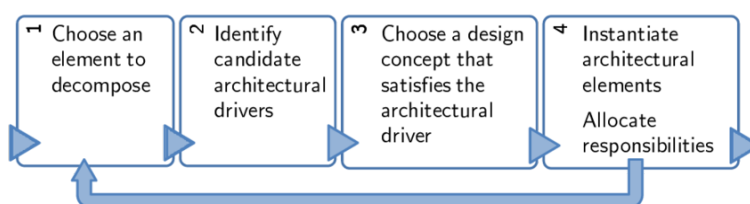
### نگرانی‌های همه انواع ذی‌نفعان

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 ویژگی رانه (ADD) یک رویکرد چابک برای توسعه معماری نرم‌افزار است که براساس 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 نرم افزار را توسعه می دهند. نگرانی های این افراد شامل موارد زیر است: پیچیدگی نرم افزار  
قابلیت نگهداری نرم افزار  
قابلیت توسعه پذیری نرم افزار
- مدیر پروژه (Project Manager) : افرادی که پروژه توسعه نرم افزار را مدیریت می کنند. نگرانی های این افراد شامل موارد زیر است:  
ریسک های پروژه  
زمان بندی پروژه  
بودجه پروژه

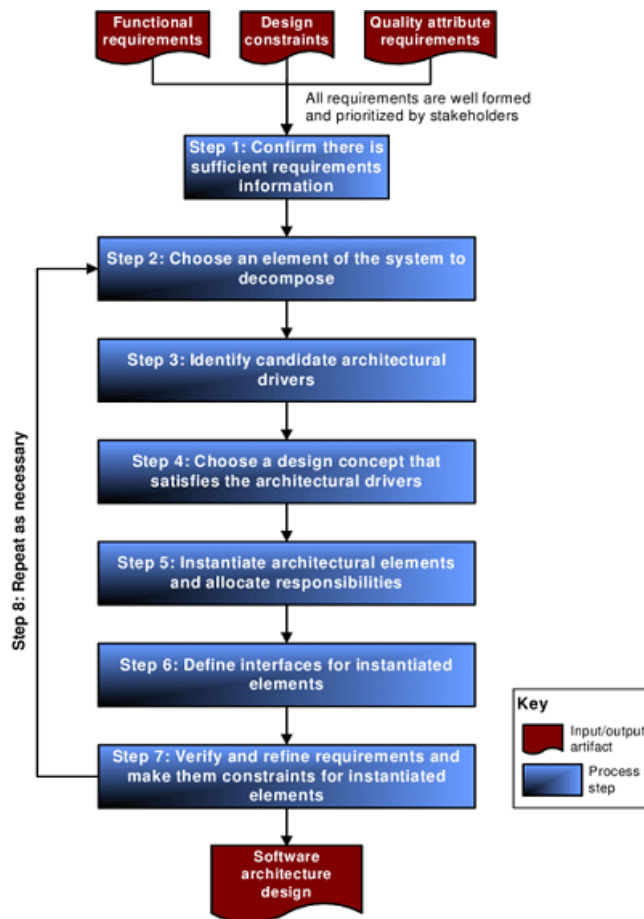
روش طراحی ویژگی رانه به ذی نفعان کمک می کند تا در فرآیند طراحی معماری مشارکت داشته باشند و نگرانی های خود را بیان کنند.

## چگونگی کاربرد مفاهیم، اصول، الگوها و سبک های معماری

روش طراحی ویژگی رانه از مفاهیم، اصول، الگوها و سبک های معماری برای ایجاد یک معماری نرم افزار کارآمد و پایدار استفاده می کند. این مفاهیم، اصول، الگوها و سبک ها عبارتند از:

- مفاهیم معماری: مانند مفاهیم سیستم های باز، سیستم های توزیع شده و سیستم های چند لایه.
- اصول معماری: مانند اصل انتزاع، اصل تفکیک وظایف و اصل وابستگی ضعیف.
- الگوهای معماری: مانند الگوهای معمارانه مشترک مانند سه لایه، MVC و MVVM.
- سبک های معماری: اینها روش هایی هستند که بیانگر یک رویکرد کلی در طراحی معماری هستند. سبک های معماری مانند معماری میکروسرویس ها، معماری مبتنی بر رویداد و معماری سرویس گرا (SOA) تاکید بر اصولی مانند تجزیه و تحلیل، توزیع وزنی کارکردها و تعامل میان سرویس های مستقل دارند. این سبک ها به معماران کمک می کنند تا سیستم هایی پیچیده و قابل توسعه را طراحی کنند که بتوانند به خوبی در محیط های پیچیده و متغیر پاسخگو باشند.

در نهایت، این مفاهیم، اصول، الگوها و سبک های معماری به معماران نرم افزار این امکان را می دهند تا سیستم هایی ایجاد کنند که نه تنها به خوبی عمل می کنند 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.