نام و نام خانوادگی دانشجو: محمدرضا هادی و محمدامین احمدی

رشته :کامپیوتر

نام استاد : دکتر سخائی نیا

موضوع : نظام تولید شراکتی مرحله ی 2 (شامل فاز 1و2)


بسم الله الرحمن الرحيم

فهرست مطالب

مقدمه

در دنیای پیچیده و متغیر نرم‌افزار، آگاهی از ساختار و رفتار سیستم‌ها امری حیاتی و ضروری است. در این مسیر، استفاده از نمودارهای کلاس به عنوان یک ابزار قدرتمند در تحلیل و آنالیز سیستم‌های نرم‌افزاری، یک اقدام ضروری و حیاتی تلقی می‌شود. این نمودارها، نه تنها به ما کمک می‌کنند تا ساختار درونی سیستم را درک کنیم بلکه به بهترین شکل ممکن از نیازها و اهداف پروژه خود مطلع شویم.

در این گزارش، ما به استخراج نمودار کلاس و آنالیز آن در یک پروژه مرتبط با مدیریت یک سیستم تولیدی پرداخته‌ایم. از اهمیت مدیریت اطلاعات شرکا تا نظارت بر فرایندها و تقسیم وظایف، تمامی جنبه‌های پروژه ما در این نمودارها به تصویر کشیده شده‌اند

همراه با این گزارش، به یک سفر جذاب در دنیای ساختارها و تعاملات سیستمی خواهیم پرداخت. ما به تفصیل باز کرده‌ایم چگونه هر کلاس، ویژگی‌ها و عملکردهای خود را دارد و چطور با یکدیگر در ارتباط هستند تا یک سیستم کارآمد و هماهنگ ایجاد کنند.

Communication Diagram

استخراج Communication Diagram از Use Case Diagram :

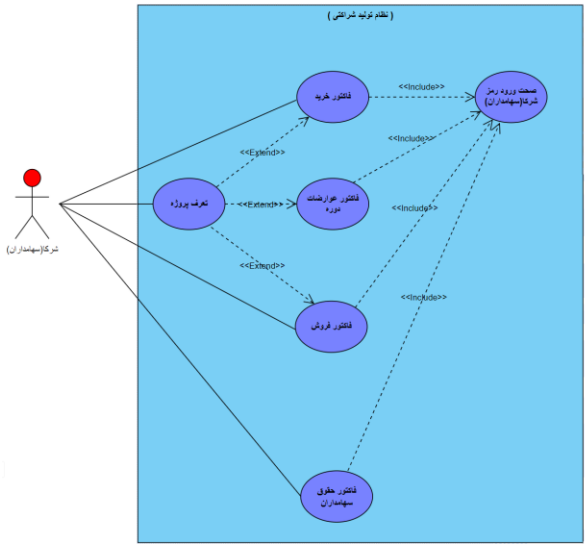
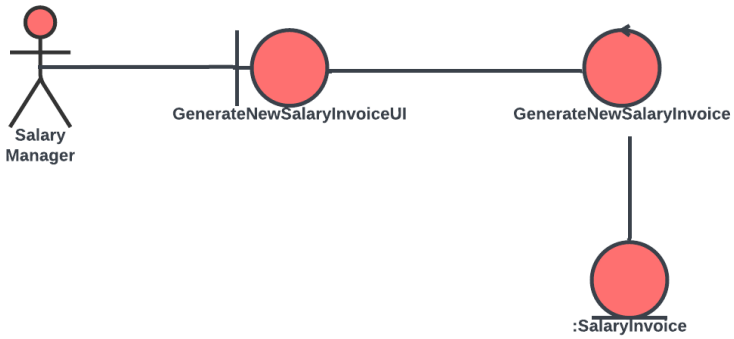
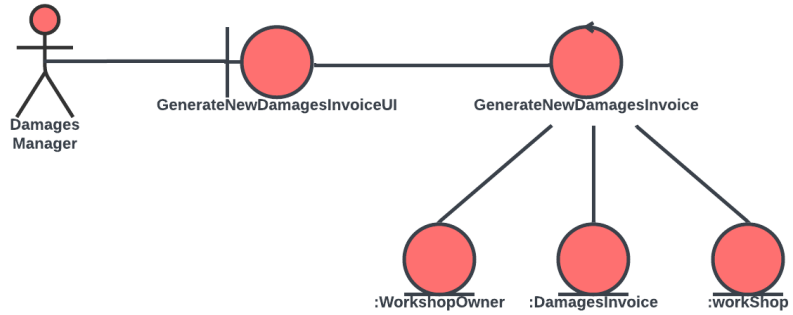


Figure1:Usecase Diagram

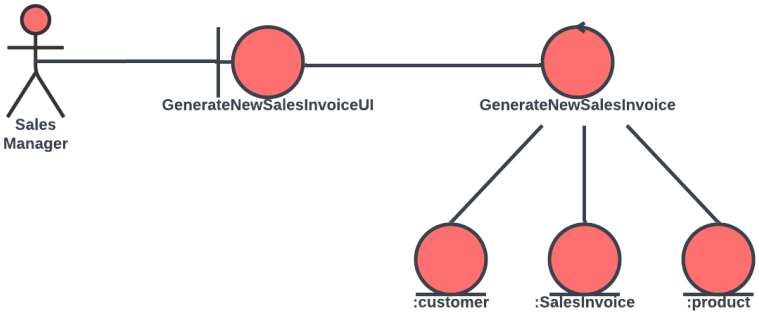
در مرحله استخراج نمودار ارتباط (Communication Diagram) از نمودار use case ، ابتدا از هر use case به عنوان یک شیء (object) در نظر گرفته می‌شود. سپس ارتباطات بین این اشیاء به شکل پیغام‌ها (messages) نمایش داده می‌شوند. این پیغام‌ها نشان‌دهنده فراخوانی‌ها و تعاملات میان اشیاء در طول اجرای یک سناریو یا use case خاص هستند.که البته اینجا نمایش نداده ایم و فقط ارتباط را با اتصال کامپوننت ها با یک خط به هم به تصویر کشیده ایم.



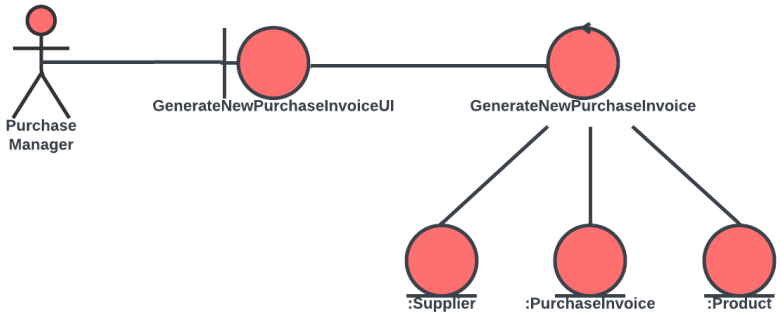
Comunication Diagram1 : فاکتور حقوق سهامداران



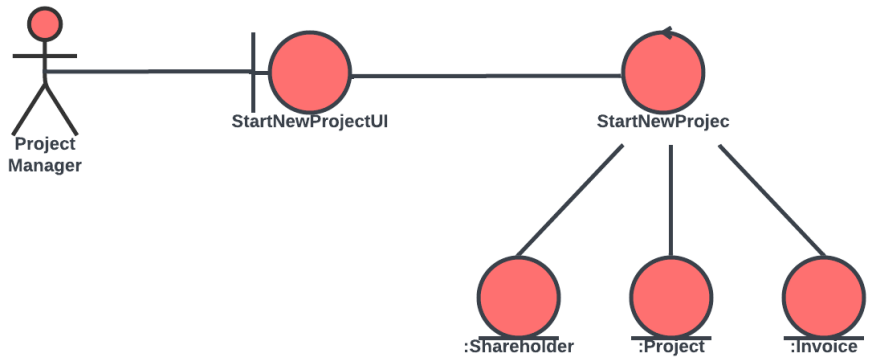
Comunication Diagram 2 : فاکتور عوارضات



Comunication Diagram 3:فاکتور فروش



Comunication Diagram 4:فاکتور خرید



تعریف پروژه : Comunication Diagram5

تأثیر استخراج Communication Diagram در ایجاد کلاس دیاگرام:

1. **شناخت ارتباطات :**با داشتن یک نمودار ارتباط، ما می‌توانیم ارتباطات بین اشیاء را به وضوح شناسایی کنیم. این اطلاعات کمک می‌کند تا ما بتوانیم تعاملات مهم در سیستم را بهبود بخشیده و بهینه‌سازی نماییم.
2. **شناسایی موارد کاربردی :**از آنجا که ارتباطات بین اشیاء براساس موارد کاربردی (use case) صورت گرفته است، شما قادر خواهید بود که موارد کاربردی کلیدی را تشخیص دهید و اطمینان حاصل کنید که این موارد به درستی پوشش داده شده‌اند.
3. **تدوین کلاس دیاگرام :**با داشتن نمودار ارتباط، می‌توانید به راحتی کلاس‌های مرتبط با هر اشیاء را تشخیص داده و سپس کلاس دیاگرام را با توجه به این اطلاعات به‌روز کنید. این کمک می‌کند تا ساختار دقیق‌تری از سیستم برقرار شود.

در ادامه:

با استفاده از نمودار ارتباطی که از use case ها ایجاد کردیم، ما موفق به شناسایی ارتباطات بین اشیاء و موارد کاربردی اصلی پروژه شدیم. این اطلاعات برای تدوین نمودارهای کلاس به ما کمک

کرد تا کلاس‌های مهم سیستم را با دقت بیشتری شناسایی کنیم و ارتباطات میان آن‌ها را توضیح دهیم. این اقدامات در نهایت بهبود و بهینه‌سازی ساختار نرم‌افزار را ایجاد خواهد کرد

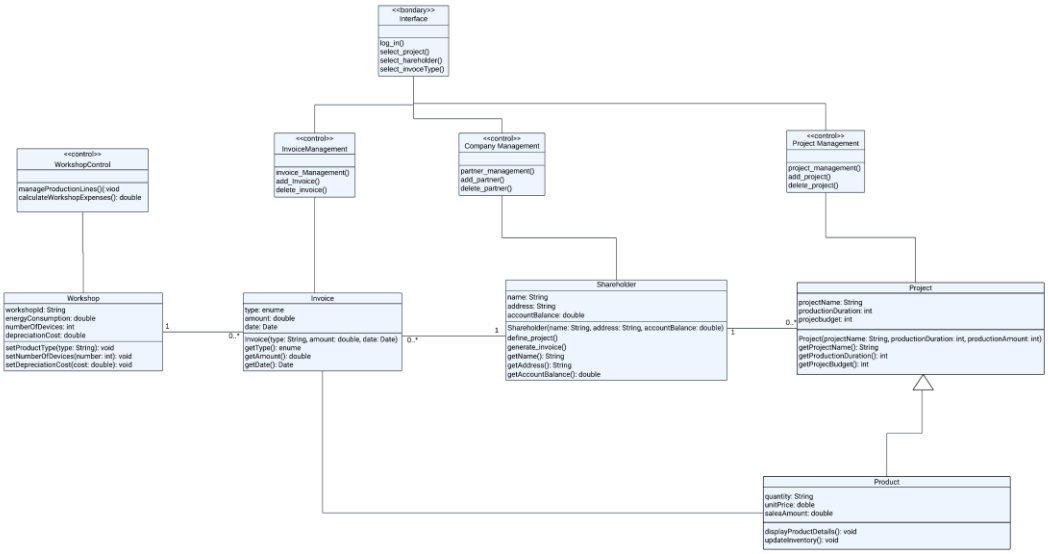
Class Diagram

فاز 1) استخراج Class Diagram از Communication Diagram :

استخراج نمودار کلاس از نمودار ارتباطی یک Communication Diagram به ما این امکان را می‌دهد که ساختار دقیق‌تری از کلاس‌ها و ارتباطات بین آن‌ها را در سیستم درک کنیم. در این مرحله، هر شیء موجود در نمودار ارتباطی به یک کلاس تبدیل می‌شود و ارتباطات میان آن‌ها با توجه به پیغام‌ها نمایش داده می‌شود.

گام‌های استخراج Class Diagram:

1. **شناسایی کلاس‌ها :**در این مرحله، هر شیء در نمودار ارتباطی به یک کلاس تبدیل می‌شود. هر پیغام از یک شیء به دیگری نمایانگر یک تعامل میان دو کلاس است.
2. **تعیین ویژگی‌ها و عملکردها :**بر اساس پیغام‌ها و تعاملات میان اشیاء در نمودار ارتباطی، ویژگی‌ها و عملکردهای هر کلاس مشخص می‌شوند. این شامل ویژگی‌ها (attribute) و متدها (method) مرتبط با هر کلاس است.
3. **تعیین روابط :**ارتباطات بین کلاس‌ها بر اساس ارتباطات در نمودار ارتباطی مشخص می‌شوند. ارتباطات می‌توانند شامل ارث‌بری، انجمن، تعاملات وابسته، و غیره باشند.



نمونه ی اولیه نمودار کلاس : Class Diagram 1

توضیح فرآیند:

پس از استخراج نمودار ارتباطی از use case ها، ما به تحلیل دقیق‌تر کلاس‌ها پرداختیم. هر شیء از نمودار ارتباطی به یک کلاس تبدیل شد و ارتباطات میان آن‌ها در نمودار ارتباطی نمایش داده شد. ویژگی‌ها و عملکردها بر اساس تعاملات میان اشیاء مشخص شدند و روابط بین کلاس‌ها نیز با توجه به ارتباطات در نمودار ارتباطی تعیین گردیدند.

سپس class diagram ناشی از هر کدام از این comuniacation diagram ها با هم ادغام شدند و چکیده و نتیجه ی آن ها تبدیل شد به تصویر بالا.

نکته: از یک Enum برای نوع فاکتورها استفاده شده است تا امکان افزودن نوع‌های جدید بدون تغییر در ساختار کلاس Invoice فراهم شود.

فاز 2) اعمال Reuse و Object Interaction روی Class Diagram و بهینه سازی:

- **اعمال Reuse:**

– در متدهایی که با ویژگی‌ها تعامل دارند، نام متد باید بازتابی از تعامل باشد.

– اعمال ارث بری ها.

– در کلاس‌های Entity، متدها و ویژگی‌ها مرتبط با داده‌ها و مدل دامنه وجود دارد. این کلاس‌ها به عنوان مدل دامنه عمل می‌کنند و از آنجا که از کلاس‌های Entity می‌توان به عنوان ابسترکشن از منطق سیستم استفاده کرد، اینجا REUSE مشاهده می‌شود.

– ارتباط بین Boundary و Control به نظر مناسب و معقول است. از مزیت‌های استفاده از این اتصالات این است که Boundary کنترل‌های کاربری را فراهم می‌کند و Control اقدامات مربوط به منطق سیستم را انجام می‌دهد. در کلاس‌های Control، متدها ویژگی‌های مرتبط با موجودیت‌ها (Entity) را مدیریت می‌کنند. این ایجاد یک لایه میانی بین کلاس‌های Boundary و Entity است که باعث جدا بودن کامپوننت‌های گوناگون و امکان REUSE آن‌ها می‌شود.

- **ایجاد اصلاحاتی در طرح اولیه:**

– در مورد ارث‌بری کلاس `Product` از کلاس `Project`، این موضوع نادرست به نظر می‌رسد. چرا که یک محصول (Product) و یک پروژه (Project) از لحاظ مفهومی با یکدیگر تفاوت زیادی دارند و نمی‌توان گفت "هر محصول یک پروژه است". ارتباط بین این دو کلاس باید از طریق یک رابطه دیگر مثل Composition یا Aggregation باشد.

- **Object Interaction(Composition ,Aggregation ,Association):**

- Association: متدهای Boundary (مانند log_in و select_project) با متدهای Control تعامل دارند. می‌توانید این تعاملات را با استفاده از روابط Association یا Dependency نشان دهید. به عنوان مثال، متد log_in در کلاس Boundary باعث ایجاد ارتباط با متدهای Control مربوط به مدیریت شرکا یا فاکتورها می‌شود.

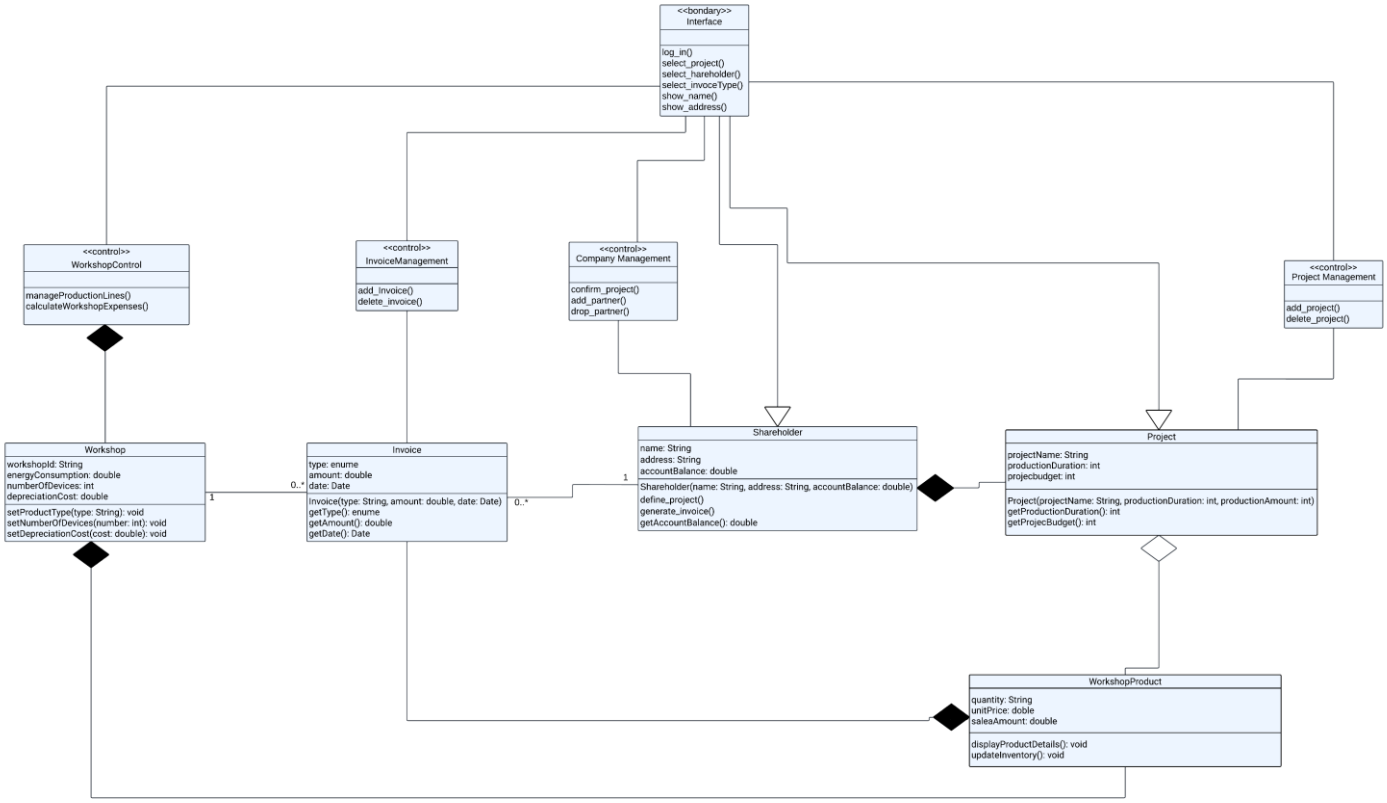
- Composition: از روابط Composition برای ارتباط بین `Workshop` و `Product` استفاده شده است تا نشان دهد که هر تغییر در `Workshop` می‌تواند تأثیری بر `Product` داشته باشد. همچنین از روابط Composition بین کلاس‌های Invoice و Product استفاده شده است تا تأثیر تولید محصول بر فاکتورها نشان داده شود.

- Aggregation: در اینجا، `Project` به عنوان Aggregate عمل می‌کند که یک یا چند شیء از نوع `Product` را در خود نگه می‌دارد.

سایر تعاملات:

- تعامل میان `Boundary` و `Control`:
  - `Boundary` ها وظیفه گزارش کارهای کاربر به `Control` ها را دارند.
  - `Control` ها با استفاده از متدها و ویژگی‌ها، عملکرد سیستم را مدیریت می‌کنند.
- تعامل میان `Control` و `Entity`:
  - `Control` ها با `Entity` ها ارتباط دارند تا اطلاعات را مدیریت کنند.
  - `Company Management` با `Shareholder` برای افزودن و حذف شرکا تعامل دارد.
  - `InvoiceManagement` با `Invoice` برای افزودن و حذف فاکتورها تعامل دارد.
  - `Project Management` با `Project` برای افزودن و حذف پروژه‌ها تعامل دارد.
  - `WorkshopControl` با `Workshop` برای مدیریت خطوط تولید و هزینه‌های کارگاه تعامل دارد.
- تعامل میان `Entity` ها:
  - `Shareholder` با `Project` برای تعریف پروژه تعامل دارد.
  - `Shareholder` با `Invoice` برای تولید فاکتور تعامل دارد.

***نمودار کلاس نهایی پس از انجام اصلاحات به شکل زیر شد:**



کلاس دیاگرام نهایی : Class Diagram2