**فصل سوم: طراحی معماری**

**3-1 فرایند طراحی معماری**

طراحی معماری یک سیستم نرم‌افزاری یک فرایند شناختی تصمیم‌گیری به‌منظور تبیین ساختار کلی سیستم، زیرسیستم‌ها و ارتباط میان آنها است و عوامل متعددی در این امر دخیل است. از این عوامل می‌توان به نوع سیستم تحت توسعه و اهداف دنبال شده جهت طراحی معماری سیستم اشاره کرد. باتوجه‌به اینکه طراحی معماری یک فرایند بازگشتی است، هر سیستم متشکل از تعدادی زیرسیستم است و هرکدام از این زیرسیستم‌ها نیز از زیرسیستم‌ها نیز از سطوح پایین‌تری تشکیل شده‌اند و تکرار فرایند بازگشتی طراحی برای هر سطح و تا پایین‌ترین سطح لازم است. پایان فرایند به عوامل گوناگونی نظیر اندازه و پیچیدگی سیستم، تجربه تیم توسعه و اهداف طراحی بستگی دارد.

**3-1-1 تبیین اهداف طراحی**

ابتدا نیاز است که ملزومات اساسی و محدودیت‌های سیستم بنا بر شاخص‌های قابل‌توجه بررسی شوند:

1- سادگی تغییر و نگهداری: سبتا سیستمی بر پایه نیازمندی‌ها و قیود از پیش تعیین شده است که نشانگر ثبات آن در طول زمان و عدم نیاز به تغییرات احتمالی و مداوم است.

2- کاربرد قطعات تجاری: استفاده از قطعات تجاری در سبتا مجاز و بلامانع است.

3- کارایی سیستم: نیازمندی‌های سیستم و به طور خاص‌تر نیازمندی‌های کارایی ایجاب می‌کند که سبتا تعداد زیادی از درخواست‌های کاربران را درآن‌واحد پردازش کند.

4- قابلیت اطمینان: سبتا نیاز دارد که قابلیت پوشش نیازمندی‌ها و قیود طراحی شده را داشته باشد تا از درجه اطمینان بالایی برخوردار باشد.

5- امنیت: سبتا باید به‌منظور حفظ امنیت سیستم و کاربران از رمزگذاری اطلاعات پشتیبانی کند و با استفاده از امنیت چندسطحی امنیت حریم خصوصی کاربران را حفظ کند.

6- تحمل‌پذیری خطا: سیستم سبتا می‌بایست گزارش خطای خودکار در زیرسیستم‌های مختلف را در کنار قابلیت گزارش دستی خطا را داشته باشد.

7- ترمیم: سیستم سبتا نیاز دارد که اطلاعات کاربران را از طریق پایگاه‌داده قوه قضائیه بازیابی کند.

**3-1-2 تعیین نوع سیستم**

نوع یک سیستم، مدل‌سازی، تحلیل، طراحی، پیاده‌سازی، و آزمون سیستم را به‌شدت تحت‌تأثیر خود قرار می‌دهد. به همین دلیل در زمان طراحی معماری نرم‌افزار انتخاب نوع سیستم بسیار اهمیت دارد.

باتوجه‌به اهمیت تعامل بین سیستم و کنشگر برای انجام یک فرایند در سبتا و اهداف طراحی معماری ذکر شده و علاوه‌برآن:

1. تعامل بین سیستم و کنشگر برای انجام یک فرایند در سبتا، شامل دنباله ثابتی از درخواست‌های کنش‌گر مثل ورود، مشاهده ابلاغیه و پرونده است و سیستم باید به آنها پاسخ دهد.

2. در بیشتر اوقات سیستم در هر فرایند، تنها با یک کنشگر تعامل می‌کند.

3. کنشگرهای سبتا فقط شامل انسان‌ها می‌شود.

4. در همه فرایندها تعامل از کنشگر شروع شده و به او ختم می‌شود.

5. کنشگر از سیستم خدماتی را درخواست می‌کند و سیستم به آنها پاسخ می‌دهد، به‌نوعی بین کنشگر و سیستم رابطهٔ مشتری - خادم برقرار است.

در نتیجه سبتا یک سیستم تعاملی است و باید معماری نرم‌افزار را متناسب با آن انتخاب کنیم.

**3-1-3 استفاده از سبک‌های معماری**

انواع مختلف سیستم‌ها، به معماری‌های متفاوت نرم‌افزار نیازمندند، بنابراین باید باتوجه‌به سیستم درحال‌توسعه سبک معماری مناسب انتخاب شود.

در سیستم‌های تعاملی سبک معماری N مناسب است، این سبک معماری، اجزای سیستم را به لایه‌های نسبتاً مستقل با اتصال ضعیف، مرتب می‌نماید. هر لایه یک وظیفه و عملکرد خوش تعریف دارد و تأثیرات بر لایه‌های دیگر را کاهش می‌دهد.

در معماری N لایه درخواست‌ها در هر فرایند از یک‌لایه به لایهٔ دیگر فرستاده می‌شود و ارسال درخواست از لایهٔ پایین‌تر به لایهٔ بالاتر مجاز نیست.

لایه‌های این سبک معماری شامل:

1. لایهٔ واسط کاربر گرافیکی

2. لایهٔ اشیای کسب‌وکار

3. لایهٔ پایگاه‌داده

4 . لایهٔ ارتباط شبکه

**3-1-4 زیرسیستم‌ها و واسط‌های سیستم**

در این گام نیازمندی‌های نرم‌افزار و اهداف طراحی آن، به زیرسیستم‌ها و مؤلفه‌های معماری تخصیص داده می‌شود.

1. GUI Layer : لایهٔ واسط کاربر گرافیکی یک گروه از اشیا است که مسئول نمایش اطلاعات، منوها، و دکمه‌های عملیاتی به کاربر هستند. به‌طورکلی در این لایه همه صفحه‌هایی که کاربر با آنها در ارتباط استقرار دارند مانند:

* صفحه ثبت‌نام
* صفحه ورود به سامانه
* صفحه احراز هویت
* صفحه پروفایل کاربر

1. Business Layer : این لایه مسئول پردازش تراکنش‌های کسب‌وکار است که با موارد کاربرد نشان‌داده‌شده‌اند. این لایه شامل دو بخش می‌شود که در ادامه آنها را بررسی می‌کنیم.

* Controller : این لایه شامل اشیاء کنترل‌گر می‌باشد. هر کنترل‌گر، مسئول برخورد با رویدادهای مربوط به یک مورد کاربرد مشخص است.

در بیشتر موارد یک تناظر یک‌به‌یک بین موردهای کاربرد و اشیای کنترل‌گر برقرار است.

هر شیء در زمان ارسال یک خدمت از سوی کاربر، مسئول برخورد با رویدادهای مربوط به آن هستند.

* Business Logic : اشیای کسب‌وکار در این زیرسیستم وجود دارند.

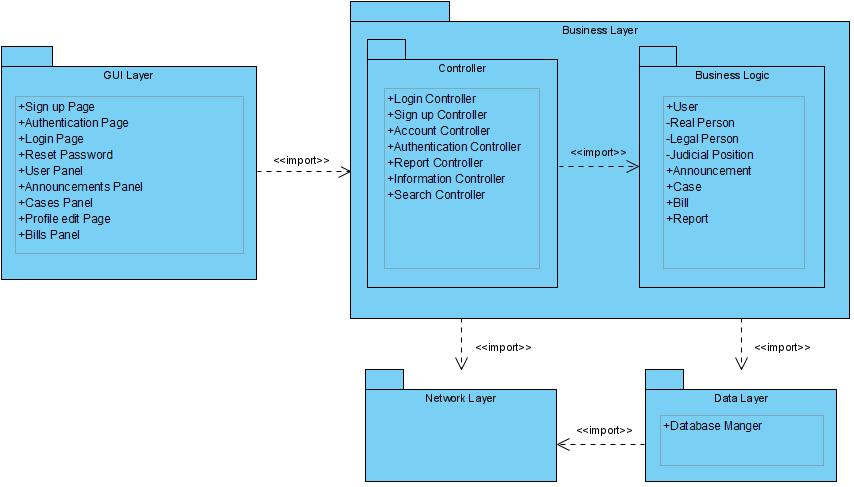
این بخش شامل مهم‌ترین زیرسیستم‌های سامانه می‌باشد و منطق سامانه در این بخش پیاده‌سازی می‌شود.

1. Data Layer : این لایه از اشیایی تشکیل می‌شود که عملیات مربوط به پایگاه‌داده، مانند ذخیره و بازیابی اشیا، را فراهم می‌نماید.
2. Network Layer : این لایه، عملیات مربوط به ارتباطات شبکه را فراهم می‌سازد.

**بازبینی طراحی معماری**

در این بخش، طراحی معماری انجام شده بازبینی می‌شود تا از پیاده‌سازی اهداف موردنظر سیستم، اطمینان حاصل شود.

**3-2 سبک معماری و نمودار بسته**



**3-3 قوانین طراحی نرم‌افزار**

بسیاری از مشکلات طراحی بر بهره‌وری و کیفیت نرم‌افزار تأثیر منفی گذاشته و هزینه‌های نگهداری نرم‌افزار را به‌شدت افزایش می‌دهند. یکی از راه‌حل‌های پیشنهادشده برای حل این‌گونه مسائل، قوانین طراحی نرم‌افزار است استفاده صحیح آنها در طراحی نرم‌افزار، می‌تواند کیفیت نرم‌افزار را به‌شدت افزایش دهد. سیستم سبتا با درنظرگرفتن این قوانین که در ادامه با جزئیات بیان شده است، سعی کرده است که کیفیت نرم‌افزاری خود را بهبود بدهد.

**3-3-1 طراحی برای تغییر**

سیستم سبتا به دلیل وجود یک سری رویداد ممکن است دچار تغییراتی شود که برخی از این رویدادها عبارت‌اند از:

* وقوع اختلالات سیستمی و باگ‌های منجر به تغییر نیازمندی‌های نرم‌افزاری
* تغییر در قوانین و دستورالعمل‌های محیط کسب‌وکار
* تغییرات نرم‌افزاری سیستم به دلایل مختلف مانند به‌روزرسانی و بهبود امنیت سیستم
* تغییرات سخت‌افزاری و ابزارهای موردنیاز جهت پیاده‌سازی سیستم
* ایجاد بهبودهای موردنیاز بنا بر بازخورد مشتری
* تغییر زمان تحویل پروژه و بودجه اختصاص‌داده‌شده

مزیت سبتا در چندلایه بودن معماری آن است و تا جایی که میسر بوده، لایه‌های معماری سیستم وابستگی کمی به یکدیگر دارند و هرکدام از زیرسیستم‌ها استقلال داشته باشند. به این صورت که در صورت وقوع هرگونه تغییر احتمالی در زیرسیستم موردنظر سایر زیرسیستم‌ها تاحدامکان دست‌نخورده باقی خواهند ماند و این تغییرات به‌آسانی صورت می‌گیرد.

**3-3-2 جداسازی دغدغه‌ها**

جداسازی دغدغه‌ها، ایده‌ای مطرح شده توسط ادسگر دایکسترا می‌باشد. این ایده بیان می‌کند که به‌جای تمرکز یک‌باره و هم‌زمان به همه جنبه‌های یک مسئله، هر بار بر یکی از جنبه‌ها و جدا از سایر آنها تمرکز می‌شود که از انواع نمودارها در این سند به همین سبب استفاده شده است. چسبندگی بالا در اثر پیاده‌سازی نتیجه پیاده‌سازی این کار در پروژه و تفکیک مسئولیت‌ها و دغدغه‌های گوناگون است. بنا بر تقسیم‌بندی وظایف، هر لایه دغدغه مربوط به خود را دارد به‌عنوان‌مثال لایه واسط کاربر گرافیکی وظیفه نمایش اطلاعات را بر عهده دارد و لایه پایگاه‌داده، اطلاعات مربوط به کاربران را ذخیره و بازیابی می‌کند.

**3-3-3 پنهان‌سازی اطلاعات**

قانون پنهان‌سازی اطلاعات، نخستین‌بار توسط دیوید پارناس 2 به‌عنوان یک قانون طراحی معرفی گردید. مطابق این قانون، جزئیات پیاده‌سازی یک بدنه نرم‌افزاری، برای کاهش اثرات تغییر آن بر سایر قسمت‌های سیستم نرم‌افزاری، محافظت می‌شود. N لایه بودن معماری سیستم سبتا باعث شده که اطلاعات به‌صورت کلی قابل‌دسترسی و مشاهده نباشد و هرکدام از زیرسیستم‌های مستقل به اطلاعات مربوط به خود دسترسی داشته باشند و قابلیت دستیابی به داده‌های موجود در سایر زیرسیستم‌ها وجود نداشته باشد.

**3-3-4 چسبندگی زیاد**

قانون چسبندگی زیاد توصیه می‌کند که طراحی پیمانه‌ها باید طوری باشد که توابع هر پیمانه، بیشترین درجهٔ ارتباط با مسئولیت اصلی پیمانه را داشته باشند. اعمال قانون چسبندگی زیاد در طراحی معماری به این معناست که مؤلفه‌ها و کلاس‌های هر زیرسیستم باید تا حدود زیادی به مسئولیت اصلی زیرسیستم مرتبط باشند. در سیستم سبتا هدف کلی از وظایف محول شده به هر لایه، اجرا و محقق شدن آرمان کل سیستم است و هر لایه معماری سبتا توابع و کلاس‌های مربوط به خود را داراست.

**3-3-5 جفت‌شدگی کم**

استفاده از قانون جفت‌شدگی کم در طراحی معماری، به معنای کاهش اثرات زمان اجرا و تأثیر تغییر در هر زیرسیستم بر زیرسیستم‌های دیگر است. به‌خصوص، طراحی باید از متغیرهای کنترلی دارای بیش از دو مقدار اجتناب نماید. به‌علاوه، برای کاستن تأثیر تغییر، می‌توان از قوانین طراحی برای تغییر و پنهان‌سازی اطلاعات استفاده کرد و باتوجه‌به معماری N لایه انتخاب شده، لایه‌های سیستم جفت‌شدگی کمی دارند و به‌صورت مستقل هر لایه کار مربوط به خود را انجام داده و خروجی را به لایه‌های بعدی منتقل می‌کند.

**3-3-6 ساده و احمقانه فرض کن**

قانون ساده و احمقانه فرض کن، طراحی‌های ساده، سرراست، و قابل‌فهم را توصیه می‌نماید. در این نگاه اشیا به‌صورت نادان در نظر گرفته می‌شوند؛ به این معنا که هر شیء تنها توانایی انجام یک کار بخصوص را دارد و روش انجام سایر کارها را نمی‌داند. تقسیم‌بندی سیستم سبتا این قانون را رعایت کرده و در هرکدام از لایه‌ها به‌مانند لایه واسط کاربر گرافیکی و لایه کسب‌وکار برای اجرای توابع، کلاس‌ها و اشیا به ساده‌ترین شکل ممکن تعریف شده‌اند و در نتیجه می‌توان اذعان کرد که سبتا دارای اشیای احمق است.