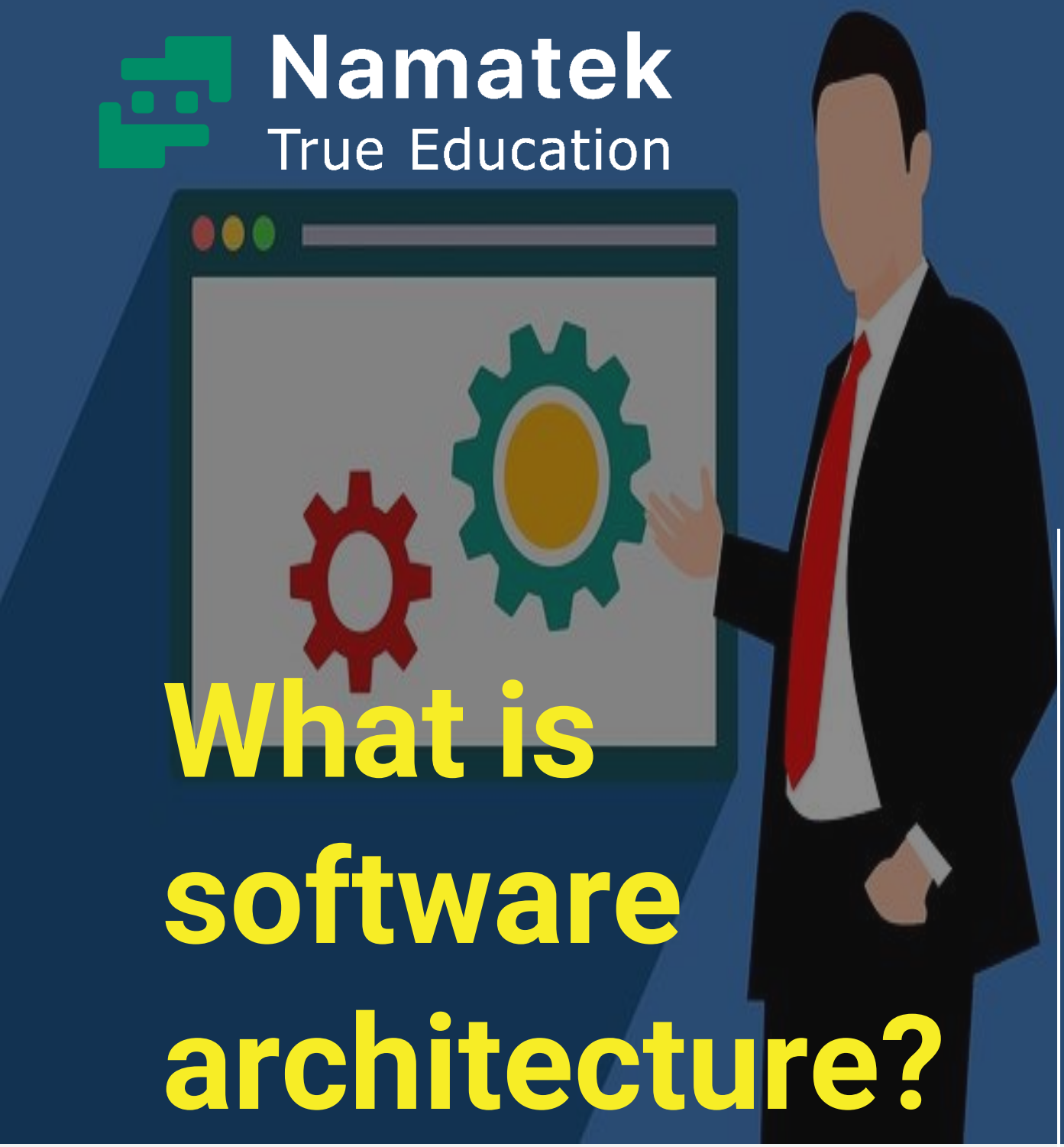
Namatek
True Education



What is software architecture?

www.namatek.com

معماری نرم افزار چیست؟

فهرست مطالب

احتمالا شما هم به عنوان یک فعال و یا علاقه مند در زمینه کارهای نرم افزاری بارها با عبارت معماری نرم افزار رو به رو شده اید. چه شباهت هایی بین نرم افزار و ساختمان است که یکی از گام های نخست اجرایی شدن طرح معماری آن هاست؟

در این مقاله سعی داریم به زبانی ساده بررسی کنیم مفهوم واقعی این اصطلاح چیست، اهداف و مزایای بکار بردن آن کدامند؟

#1 معماری نرم افزار چیست؟

معماری نرم افزار به ساختارهای بنیادی یک سیستم نرم افزاری و نظم ایجاد چنین ساختارها و سیستم هایی اشاره دارد. هر ساختار در نرم افزار شامل عناصر نرم افزاری، روابط بین آن ها و خصوصیات هر دو عنصر و روابط است.

در واقع معماری یک سیستم، اجزای اصلی آن، روابط آن ها (ساختارها) و نحوه تعامل آن ها با یکدیگر را توصیف می کند. معماری یک سیستم نرم افزاری مشابه معماری یک ساختمان است.

این معماری به عنوان یک طرح اصلی برای سیستم و پروژه در حال توسعه عمل می کند و وظایفی را که باید توسط تیم های طراحی انجام شود، مشخص می کند.

معماری نرم افزار در واقع انتخاب ساختاری اساسی است که تغییر آن پس از اجرا پر هزینه است. گزینه های معماری نرم افزار شامل گزینه های ساختاری خاص از امکانات موجود در طراحی نرم افزار است.

به عنوان مثال، سیستم هایی که وسیله نقلیه پرتاب شاتل فضایی را کنترل می کنند باید بسیار سریع و بسیار قابل اعتماد باشند.

بنابراین، باید یک زبان محاسباتی در زمان واقعی مناسب انتخاب شود. علاوه بر این برای برآوردن نیاز قابلیت اطمینان بودن می توان چندین نسخه اضافی و مستقل از برنامه تولید کرد و این نسخه ها را روی سخت افزار مستقل اجرا کرد و نتایج را بررسی کرد.

2# تصمیمات در معماری نرم افزار

معماری نرم افزار شامل مجموعه ای از تصمیمات مهم در مورد سازمان مربوط به توسعه نرم افزار است و هر یک از این تصمیمات می تواند تاثیر قابل توجهی بر کیفیت، نگهداری، عملکرد و موفقیت کلی محصول نهایی داشته باشد.

این تصمیمات شامل موارد زیر است:

- انتخاب عناصر ساختاری و رابط هایی که سیستم توسط آن ها ساخته شده است
- رفتار در همکاری بین این عناصر
- ترکیب این عناصر ساختاری و رفتاری به زیر سیستم بزرگ
- همسو بودن تصمیمات معماری با اهداف تجاری

- سبک های معماری سازمان را راهنمایی کند

#3 اهداف معماری نرم افزار

هدف اصلی این معماری شناسایی نیازهایی است که بر ساختار برنامه تأثیر می گذارد. یک معماری خوب خطرات تجاری مرتبط با ایجاد راه حل فنی را کاهش می دهد و پلی را بین نیازهای فنی و تجاری ایجاد می کند.

برخی از اهداف معماری نرم افزار عبارتند از:

1. ساختار سیستم را در معرض دید قرار دهد اما جزئیات اجرای آن را مخفی کند.

2. تحقق بخشیدن به تمام موارد استفاده و سناریوها
3. سعی در رسیدگی به الزامات ذینفعان مختلف
4. رسیدگی به هر دو الزامات عملکردی و کیفیت
5. کاهش هدف مالکیت و بهبود موقعیت سازمان در بازار
6. بهبود کیفیت و قابلیت های ارائه شده توسط سیستم
7. بهبود اعتماد خارجی به هر دو سازمان و سیستم

#4 محدودیت های معماری نرم افزار

معماری نرم افزار هنوز یک رشته در حال ظهور در مهندسی نرم افزار است که محدودیت های زیر را دارد:

- کمبود ابزار و روش های استاندارد برای نمایش معماری
- کمبود روش تجزیه و تحلیل پیش بینی اینکه آیا معماری منجر به پیاده سازی مطابق با الزامات خواهد شد یا خیر.
- عدم آگاهی از اهمیت طراحی معماری در توسعه نرم افزار
- عدم درک نقش معمار نرم افزار توسط ذینفعان و ارتباط ضعیف بین آن ها
- عدم درک فرآیند طراحی، تجربه طراحی و ارزیابی طرح

#5 نقش معمار نرم افزار چیست؟

یک معمار نرم افزار راه حلی را ارائه می دهد که تیم فنی می تواند آن را در کل برنامه ایجاد و طراحی استفاده کند.

خروجی طرحی که از معمار نرم افزار دریافت می شود شامل موارد زیر می شود:

- مجموعه ای واضح، کامل، سازگار و قابل دستیابی از اهداف عملکردی
- شرح عملکرد سیستم، حداقل با دو لایه تجزیه
- مفهومی برای سیستم
- طرحی به شکل سیستم، با حداقل دو لایه تجزیه
- ایده ای از زمان کلی، ویژگی های اپراتور و برنامه های اجرا و بهره برداری

- یک سند یا فرآیند که تضمین می کند تجزیه عملکردی دنبال شده، و نوع رابط کنترل می شود

برای رسیدن به نتایج بالا یک معمار نرم افزار باید در زمینه های زیر تخصص داشته باشد:

#1-5 تخصص طراحی

- متخصص در طراحی نرم افزار، از جمله روش ها و رویکردهای متنوع مانند طراحی شی گرا، طراحی رویداد محور و غیره
- رهبری تیم توسعه و هماهنگی تلاش های توسعه برای یکپارچگی طراحی
- باید قادر به بررسی پیشنهادات طراحی و سبک و سنگین کردن آن ها جهت استفاده باشد

#2-5 تخصص دامنه

- متخصص روی سیستم در حال توسعه و برنامه ریزی برای توسعه نرم افزار باشد.
- کمک در روند بررسی نیاز، اطمینان از کامل بودن و ثبات
- هماهنگ سازی در تعریف مدل دامنه برای سیستم در حال توسعه

#5-3 تخصص فناوری

- متخصص در فن آوری های موجود که در اجرای سیستم کمک می کند.
- هماهنگ سازی انتخاب زبان برنامه نویسی، چارچوب، سیستم عامل ها، پایگاه داده و غیره

#5-4 تخصص روش شناسی

- متخصص روش های توسعه نرم افزار که ممکن است در طول SDLC (چرخه عمر توسعه نرم افزار) اتخاذ شود.
- رویکردها و روش های مناسبی را برای پیشرفت انتخاب کند که به کل تیم کمک کند.

#5-5 نقش پنهان معمار نرم افزار

- تسهیل کار فنی در میان اعضای تیم و تقویت رابطه اعتماد در تیم
- متخصص اطلاعات که دانش مشترک دارد و تجربه زیادی دارد
- از اعضای تیم در برابر نیروهای خارجی که باعث حواس پرتی آن ها می شود و ارزش کمتری برای پروژه دارند محافظت می کند.

#6 ویژگی های کیفی در معماری نرم افزار

کیفیت معیاری برای تعالی یا عاری بودن از کمبود و نقص است. ویژگی های کیفی خصوصیاتی از سیستم هستند که از عملکرد سیستم جدا هستند. پیاده سازی ویژگی های کیفیت، تمایز یک سیستم خوب از یک سیستم بد را آسان می کند.

این ویژگی ها به طور کلی عواملی هستند که بر رفتار زمان اجرا، طراحی سیستم و تجربه کاربر تاثیر می گذارند.

ویژگی های کیفی را می توان در موارد زیر طبقه بندی کرد:

#6-1 ویژگی های کیفیت استاتیک

انعکاس ساختار یک سیستم و سازمان که مستقیماً با معماری، طراحی و کد منبع مرتبط است. این موارد برای کاربر نهایی نامرئی هستند اما بر هزینه توسعه و نگهداری تاثیر می گذارند.

به عنوان مثال: ماژولار بودن، قابلیت تست، قابلیت نگهداری و غیره

#2-6 ویژگی های کیفیت پویا

این ویژگی ها بازتاب رفتار سیستم در هنگام اجرای آن است که مستقیما با معماری، طراحی، کد منبع، پیکربندی، پارامترهای استقرار، محیط و بستر سیستم ارتباط دارند.

این موارد برای کاربر نهایی قابل مشاهده هستند و در زمان اجرا وجود دارند.

به عنوان مثال توان عملیاتی، قدرت، مقیاس پذیری و غیره

#7 سناریوهای کیفی در معماری نرم افزار

سناریوهای کیفی نحوه جلوگیری از خطا در زمان خرابی را مشخص می کنند. آن ها را می توان بر اساس مشخصات ویژگی آن ها به شش بخش تقسیم کرد:

- منبع: (Source) موجودیتی داخلی یا خارجی مانند افراد، سخت افزار، نرم افزار یا زیرساخت های فیزیکی
- محرک: (Stimulus) شرایطی که هنگام ورود به سیستم باید مورد توجه قرار گیرد.
- محیط: (Environment) محرک هایی که با شرایطی خاص در محیط ایجاد می شوند.
- سازه ها: (Artifact) کل سیستم یا بخشی از آن مانند پردازنده ها، کانال های ارتباطی، ذخیره سازی مداوم، پردازش ها و غیره
- واکنش: (Response) فعالیتی که پس از رسیدن محرک انجام می شود. مانند شناسایی خطاها، بهبود خطا، غیرفعال کردن منبع رویداد و غیره
- اندازه گیری پاسخ: (Response measure) باید پاسخ های رخ داده را اندازه گیری کرد تا بتوان الزامات را آزمایش کرد.