



سؤال ۱.

چه تفاوتی بین درس مهندسی نرم‌افزار و تحلیل طراحی سیستم‌ها وجود دارد؟ چه مباحثی در زمینه این درس باید در ادامه حیات مهندسی خود بیاموزید؟ آیا شما برای مهندسی نرم‌افزار گزینه‌ی مناسبی هستید؟ چرا؟

• الف) درس مهندسی نرم‌افزار و تحلیل و طراحی سیستم‌ها موضوعاتی هستند که در دو رشته‌ی مختلف درس داده می‌شوند. در مهندسی نرم‌افزار تمرکز بر روی تولید و توسعه نرم‌افزار است؛ این در حالی است که تمرکز درس تحلیل و طراحی سیستم‌ها بر روی ساخت یک سیستم اطلاعاتی است. یک سیستم اطلاعاتی نمی‌تواند بدون آن‌که نرم‌افزار آن را بسازیم، ایجاد شود. علت این‌که در زمینه مهندسی نرم‌افزار در مقایسه با تحلیل و طراحی سیستم‌ها درس‌های بیش‌تری ارائه می‌شوند، «کیفیت» در این حوزه است. در واقع در درس تحلیل و طراحی سیستم‌ها بیش‌تر با روش‌گان‌ها^۱ سروکار دارد. به زبان دیگر اگر به منابع^۲ و کتاب‌های این دو نگاهی اجمالی بیندازیم، متوجه می‌شویم که در کتاب‌های تحلیل و طراحی سیستم‌ها به دنبال روشی برای طراحی مراحل^۳ سیستم‌های اطلاعاتی هستیم اما در کتاب‌های مهندسی نرم‌افزار بیش‌تر مدیریت پروژه نرم‌افزاری و ارزیابی کیفیت خروجی پروژه مد نظر است.

• ب) مهندسی نرم‌افزار شامل یک فرآیند، مجموعه‌ای از روش‌ها، تمرین‌ها و مجموعه‌ای از ابزارها است که به متخصصان اجازه می‌دهد، نرم‌افزارهای کامپیوتری با کیفیت بالا بسازند.

به همین دلیل، مهندس نرم‌افزار به مهارت‌های زیادی در حوزه‌های مختلف نیاز دارد که مهم‌ترین آن‌ها عبارتند از:

— چند کاری^۴: به عنوان یک مهندس نرم‌افزار نیاز است تا توانایی مدیریت کردن چندین پروژه را در محیط ددلاین محور را داشته باشیم. این مهارت را اگر به‌طور جزئی‌تر بخواهیم تحت نظر قرار دهیم شامل

- * سازمان‌دهی
- * اولویت‌بندی
- * ددلاین‌ها
- * مدیریت توقعات

می‌شود.

— کار تیمی: ساخت سیستم‌های نرم‌افزاری عموماً یک تلاش انفرادی است اما یک مهندس نرم‌افزار هنوز باید توانایی برقراری ارتباط با افراد و تیم‌های دیگر را داشته باشد و این ارتباط را به‌طور مرتب برقرار کند. به عنوان یک مهندس نرم‌افزار باید قادر باشیم تا نیازهای پروژه را بیان کنیم و اگر با چالش یا مشکلی مواجه شدیم، بحث کنیم. این بخش شامل

- * همکاری
- * سازش^۵
- * گوش کردن به‌طور فعال
- * حل اختلاف
- * ارتباط شفاهی
- * ارتباط نوشتاری

^۱ methodology
^۲ resources
^۳ phases
^۴ Multitasking
^۵ compromising

می‌باشد.

— توجه به جزییات: در صنعت، روش‌ها و استانداردهای خاصی وجود دارد که یک مهندس نرم‌افزار باید به آن‌ها تسلط کامل بیابد. این موارد شامل اصول اولیه، مانند استفاده از سیستم‌های کنترل نسخه (مانند گیت‌هاب) است تا کار قدیمی خود را از دست ندهیم. البته این موارد شامل چیزهای پیچیده‌تری مانند تحلیل و نگهداری کد قدیمی‌تر یا تولید روش‌های استراتژیک برای ایجاد یک چارچوب کدزنی می‌شود. بنابراین مهارت‌های لازم در این قسمت شامل

* مهارت‌های تحلیل

* اشکال‌زدایی

* مستندسازی فنی

* فرمول‌ها

* خلاقیت

* تفکر بحرانی

می‌باشد.

— زبان‌های برنامه‌نویسی: اکثر مهندسان نرم‌افزار به یک یا دو زبان برنامه‌نویسی تسلط و اشراف کامل دارند و این یک حوزه اصلی تخصص آن‌ها می‌شود.

• (پ) با توجه موارد گفته شده در قسمت «ب»، بنده خود را یک گزینه‌ی مناسب برای مهندسی نرم‌افزار می‌دانم؛ زیرا

۱. تجربه‌ی کار در صنعت به عنوان توسعه‌دهنده و مدیر پروژه را دارم.

۲. با متدولوژی‌های گوناگونی مانند UP، Agile و... آشنا هستم.

۳. درس‌های زیادی را در دانشکده‌ی مهندسی کامپیوتر با موضوع مهندسی نرم‌افزار و نمره‌ی عالی در این زمینه گذرانده‌ام.

۴. تعداد زیادی از مهارت‌های عمومی لازم و نه فنی (نظیر کار تیمی، آشنایی با ددلاین‌ها و...) را دارا هستم.

سؤال ۲.

فرض کنید امروز در شرکتتان محصولی نرم‌افزاری ایجاد کرده و به مشتری فروخته‌اید. سی سال بعد وضعیت نرم‌افزار شما چگونه خواهد بود؟ آیا هنوز مشتری شما از نرم‌افزار استفاده می‌کند؟ اگر چنین است شما در طول این زمان چه کارهایی روی نرم‌افزار انجام داده‌اید؟ اگر از نرم‌افزار شما استفاده نمی‌شود به نظرتان چه دلایلی موجب این اتفاق شده است؟

• الف و ب) وضعیت نرم‌افزار در آینده، به عوامل زیادی بستگی دارد که از مهم‌ترین آن‌ها می‌توان به نوع قرارداد، متدولوژی مورد استفاده و نحوه پیاده‌سازی اشاره کرد. برای وضعیت نرم‌افزار نیز چندین حالت می‌توان داشت:

۱. نرم‌افزار به خوبی کار می‌کند و قابلیت اضافه کردن ویژگی‌های جدید به آن وجود دارد.

۲. نرم‌افزار کار می‌کند اما قابلیت کم یا زیاد کردن ویژگی به آن وجود ندارد. اصطلاحاً به این سیستم، سیستم «مُرده» می‌گویند.

۳. نرم‌افزار کار نمی‌کند.

نکته: نباید پشتیبانی را با نگهداری اشتباه کرد. در پشتیبانی تغییری در کد داده نمی‌شود و هدف از آن بالا نگه داشتن سیستم است. برای مثال موارد

— logging

— نسخه‌ی بازگردانی^۷

— نسخه‌ی پشتیبان^۸

— و...

از جمله موارد پشتیبانی به حساب می‌آیند اما در نگهداری منبع کد دست‌کاری می‌شود و برای بالا نگه داشتن سیستم، تغییر روی آن اعمال می‌شود.

• پ) در صورتی که نرم‌افزار کار کند و قابلیت اضافه کردن ویژگی‌های جدید یا تغییر در آن وجود داشته باشد، این نرم‌افزار دارای قابلیت نگهداری^۹ دارد. ^{۱۰} در حالت کلی ۴ نوع نگهداری داریم:

۱. Perfective Maintenance: در این نوع از نگهداری به سیستم قابلیت‌هایی را اضافه کنیم.

۲. Corrective Maintenance: در این نوع از نگهداری مشکل^{۱۱} یک قابلیت را برطرف می‌کنیم.

۳. Adaptive Maintenance: در نوع از نگهداری سیستم را با پلتفرم جدید تطبیق و تغییر می‌دهیم.

۴. Preventive Maintenance: در این نوع از نگهداری برای اجتناب از بروز یک مشکل در آینده، تغییری در سیستم ایجاد می‌کنیم.

در روش‌های گوناگون، سیاست‌های متفاوتی برای ویژگی نگهداری وجود دارد که عبارتند از:

۱. سیاست مقیم^{۱۲}. قراردادی نوشته می‌شود تا به صورت مداوم سیستم نگهداری شود.

۲. سیاست on-demand. این سیاست به این معنی است که هر موقع سیستم دچار اشکال شد و یا کارفرما ویژگی جدیدی را خواست، از سیستم نگهداری می‌شود.

۳. سیاست دوره‌ای^{۱۳}. این سیاست به این معنی است که از سیستم در بازه‌های زمانی مشخص نگهداری می‌شود.

• ت) دلیل استفاده نشدن از نرم‌افزار ما، نداشتن قابلیت نگهداری و به اصطلاح «مُرده» بودن سیستم است. چند عامل ممکن است در از بین رفتن قابلیت نگهداری سیستم، تاثیر به سزایی داشته باشند؛ این عوامل عبارتند از:

^۶feature

^۷restore

^۸backup

^۹maintenance

^{۱۰}دشواری این کار بسیار وابسته به نوع متدولوژی و نحوه پیاده‌سازی است؛ زیرا اگر Coupling بالا و Cohesion کم باشد، کار نگهداری نرم‌افزار بسیار سخت می‌شود.

^{۱۱}bug

^{۱۲}resident

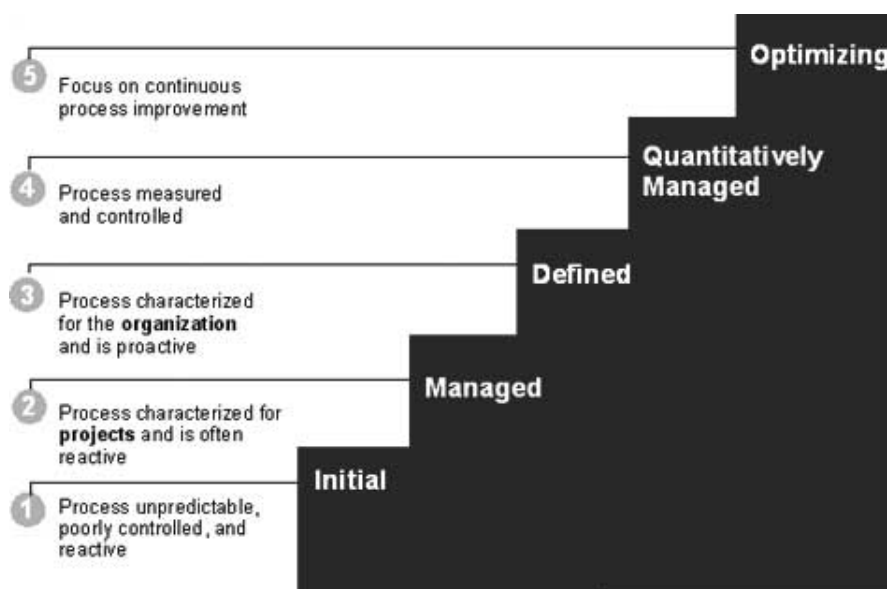
^{۱۳}periodical

۱. در صورتی که منبع کد به صورت نابهمینه و بدون در نظر گرفتن الگوهای طراحی نرم‌افزار^{۱۴} و bad smellها پیاده‌سازی شوند، تکثیر^{۱۵} در کد بسیار بالا خواهد بود. این تکثیر باعث بالا رفتن coupling و در نتیجه پایین آمدن cohesion می‌شود.
 ۲. نوع قرارداد و متدولوژی در مرحله ایجاد و توسعه نرم‌افزار. ممکن است کارفرما با توجه به شرایطی که وجود دارد، پروژه و محدودیت‌های زمانی و مالی را طوری تعریف کند که ویژگی نگهداری برایش اهمیت کمتری داشته باشد.
 ۳. وجود کلاس God. به کلاسی کلاس God گفته می‌شود که ویژگی‌ها و عمل‌کردهای زیادی در آن پیاده‌سازی شود در حالی که آن کلاس قابل شکسته شدن به کلاس‌های کوچک‌تر باشد.
 ۴. وجود دید تراپا. ^{۱۶} اگر همه‌ی دیدها در روابط به صورت دوطرفه تعریف شود در حالی که به آن نیاز نیست، باعث به وجود آمدن این عارضه می‌شود.
- همان‌طور که مشخص است، ویژگی نگهداری را می‌توان به عنوان مهم‌ترین ویژگی در ایجاد و توسعه نرم‌افزار محسوب کرد؛ به همین دلیل، در اکثر متدولوژی‌ها بیش‌ترین هزینه و زمان صرف محقق کردن این ویژگی می‌شود.

سؤال ۳.

شرکت‌ها و به‌ویژه شرکت‌های نرم‌افزاری هم مانند انسان‌ها مراحل رشد و شکوفایی دارند. برای بررسی این بلوغ از روش‌هایی مانند CBA-IPI که در فصل سوم کتاب مطرح شده، استفاده می‌شود. لطفاً با انجام تحقیق کوچکی مراحل مختلف بلوغ سازمان‌ها را در این روش به اختصار مشخص کنید.

سطح بلوغ یک تکامل خوش‌تعریف^{۱۷} جهشی رو به جلو برای دستیابی به یک فرآیند نرم‌افزاری بالغ است. هر سطح بلوغ یک لایه‌ای را در پایه برای بهبود مداوم فرآیند فراهم می‌کند.



شکل ۱: مراحل بلوغ در سازمان‌ها

در مدل CMMI^{۱۸} از پنج سطح بلوغ تشکیل شده است:

۱. سطح اولیه^{۱۹}

۲. سطح مدیریت شده

۳. سطح تعریف شده

۴. سطح مدیریت شده از نظر کمی Quantitatively Managed

۵. سطح بهینه‌سازی

حال هر سطح از سطوح بالا را به‌طور دقیق‌تر بررسی می‌کنیم:

• سطح اولیه

در این سطح فرآیندها معمولاً نامشخص هستند. سازمان معمولاً محیط پایداری را فراهم نمی‌کند و موفقیت در این سازمان‌ها بیشتر سستی به رقابت افراد و اعضای و نه استفاده از فرآیندهای اثبات شده دارد. در این مرحله بعضی اوقات محصول یا سرویسی تولید می‌شود اما قیمت تمام‌شده‌ی آن‌ها معمولاً بالاتر از بودجه‌ی اختصاص داده شده است.

• سطح مدیریت شده

در این سطح سازمان به همه‌ی اهداف کلی و خاص رسیده است. به بیان دیگر، پروژه‌های سازمان تضمین کرده‌اند که نیازمندی‌ها مدیریت می‌شوند و هم‌چنین فرآیندها نیز برنامه‌ریزی، انجام، اندازه‌گیری و کنترل می‌شوند.

در این مرحله تعهدات بین سهام‌داران ایجاد شده و در صورت نیاز مورد بازبینی و تجدید نظر قرار می‌گیرد.

^{۱۷}well-defined
^{۱۸}Integration Model Maturity Capability
^{۱۹}Initial

- **سطح تعریف شده**

در این سطح فرآیندها به خوبی مشخص و درک شده و در قالب استانداردها، رویه‌ها^{۲۰}، ابزارها و روش‌ها بیان می‌شوند. تفاوت اصلی و مهم بین سطح شماره‌ی «۳» و «۲» در محدوده استانداردها، توضیحات فرآیندها و رویه‌ها می‌باشد. در سطح «۲» استانداردها، توضیحات فرآیند و رویه‌ها ممکن است در هر نمونه خاص از فرآیند (مانند یک پروژه خاص) کاملاً متفاوت باشند اما در مورد «۳» این تفاوت وجود ندارد و سازمان یک‌پارچه است.

- **سطح مدیریت شده از نظر کمی**

در این سطح فرآیندهای فرعی که به‌طور قابل توجهی در عملکرد^{۲۱} کلی فرآیند تاثیرگذارند، انتخاب می‌شوند. این فرآیندها با استفاده از روش‌های آماری و دیگر تکنیک‌های اندازه‌گیری کمی کنترل می‌شوند. اهداف کمی به عنوان معیاری برای مدیریت فرآیندها ایجاد می‌شوند. این اهداف براساس نیازهای مشتری، کاربران نهایی^{۲۲}، سازمان‌ها و مجریان فرآیند هستند. تفاوت اصلی این سطح با سطح شماره‌ی «۳» «قابلیت پیش‌بینی عملکرد فرآیندها» است.

- **بهینه‌سازی**

در این سطح فرآیندها به‌طور مداوم بر اساس درک کمی از دلایل مشترک تغییر ذاتی فرآیندها بهبود می‌یابند و بیش‌تر تمرکز آن بر روی پیشرفت ادامه‌پذیر عمل‌کرد فرآیند از طریق کارهای تکراری-افزایشی^{۲۳} است. این سطح در مقایسه با سطح «۴» در این مورد تفاوت دارند که در آن مشکلی برای کافی بودن عوامل برای دستیابی به نتایج اهداف وجود ندارد

نکته هر کدام از سطوح گفته شده ضروری هستند و نباید از آن‌ها پرش کرد.^{۲۴} در صورت پرش کردن، هر چه سطح بالاتر باشد، احتمال رسیدن به موفقیت کم‌تر می‌شود.

^{۲۰} procedure

^{۲۱} performance

^{۲۲} users end

^{۲۳} incremental and iterative

^{۲۴} skip

سؤال ۴.

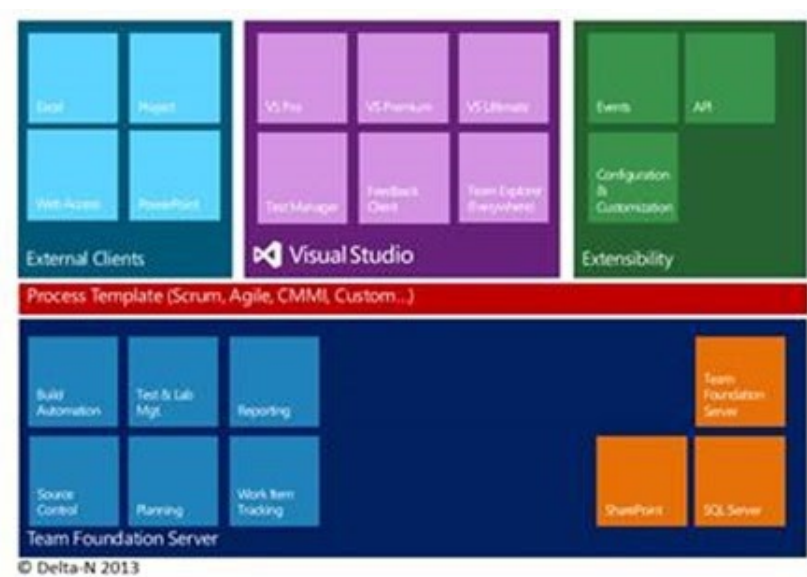
Agile مفهومی است که این روزها در دنیای مهندسی نرم افزار خیلی مطرح می شود. شما با مفاهیمی مانند XP و Scrum در کلاس کمی آشنا شده اید. در شرکت هایی که از Scrum استفاده می کنند، معمولا از ابزاری مانند tfs برای مدیریت مراحل مختلف در فرآیند استفاده می شود. لطفا در خصوص این نرم افزار تحقیق کنید و نتایج آن را مختصرا گزارش دهید. تحقیقتان در حیطه مباحثی باشد که در کلاس مطرح شده است. یعنی ببینید موارد مطرح شده در کلاس در نرم افزار چگونه دیده شده است.

TFS یک ابزار مدیریت چرخه ی حیات نرم افزار^{۲۵} است.

در واقع TFS مجموعه ای از ابزارهای مختلف تعاملی برای تولید نرم افزار را فراهم می کند که همه ی تیم ایجاد و توسعه نرم افزار از آن استفاده می کنند. این ابزارها شامل

- کنترل نسخه^{۲۶}
- ابزارهای گزارش دهی
- مدیریت نیازمندی ها
- مدیریت توزیع ها^{۲۷}
- ... و

است. به طور کلی با توجه به مطالب گفته شده می توان این نتیجه را گرفت که از آن می توان به عنوان یک backend استفاده کرد. کاربرد اصلی آن در مستقرسازی^{۲۸} است که مفهومی به نام DevOps در آن بسیار پررنگ است.



شکل ۲: نمای سطح بالای TFS

همان طور که در شکل شماره «۲» مشاهده می شود (بدون پرداختن به هر جنبه به طور مستقل)، TFS کاربردهای بسیار زیادی دارد و اساس بسیاری از موارد است. این اجازه را به ما می دهد تا منابع، مشکلات، برنامه ریزی و ساختن تست و... را به خوبی مدیریت کنیم. مهم ترین ویژگی TFS توسعه پذیر^{۲۹} آن است که آن را فوق العاده می کند؛ به این صورت که هر چیزی که وجود ندارد، یا می تواند ساخته شود و یا می تواند تغییر کند. این دقیقا همان چیزی است که استفاده از TFS را برای سازمان ها مناسب می کند. ویژگی دیگر آن تنظیم پذیری^{۳۰} و گستردگی آن است

-
- Application Lifecycle Management (ALM)^{۲۵}
 - version control^{۲۶}
 - release management^{۲۷}
 - deployment^{۲۸}
 - extensibility^{۲۹}
 - adjustability^{۳۰}

که هم می‌تواند تبدیل به نقطه قوت و هم نقطه ضعف آن شود؛ زیرا این ابزارها خاص منظوره هستند و عمل‌کرد آن‌ها باید بسته به کاربردها مقایسه و بررسی شود.

نحوه‌ی کار TFS بدین صورت است که با process templateها کار می‌کند. این قالب^{۳۱}ها از بخش‌های مختلفی شامل

- مورد کاری^{۳۲}
- گزارش‌ها
- مستندات
- نوع^{۳۳}ها
- و...

است و اجباری در انتخاب فرآیند انتخاب شده ندارد و دست مهندس نرم‌افزار را باز می‌گذارد.

TFS می‌تواند در هر متدولوژی‌ای استفاده شود. بسته به نوع متدولوژی‌ای که از آن استفاده می‌کنیم، مورد کاری تعریف می‌شود. این انعطاف‌پذیری زیاد باعث می‌شود که ساختار همه‌ی آن‌ها یکسان باشد.

^{۳۱}template
^{۳۲}work item
^{۳۳}type

سؤال ۵.

تصور کنید که پس از اتمام تحصیل تصمیم به راه اندازی شرکتی نرم افزاری با دوستانتان گرفته اید. ساختار این شرکت از لحاظ نیروی انسانی به چه شکلی است؟ چه نیروهایی دارد؟ می توانید برای پاسخ گویی بهتر حوزه فعالیت شرکت را به دلخواه پیش نهاد دهید. مثلاً بگویید شرکت بناست در حوزه تولید نرم افزارهای مالی فعالیت کند لذا این نیروها را خواهد خواست.