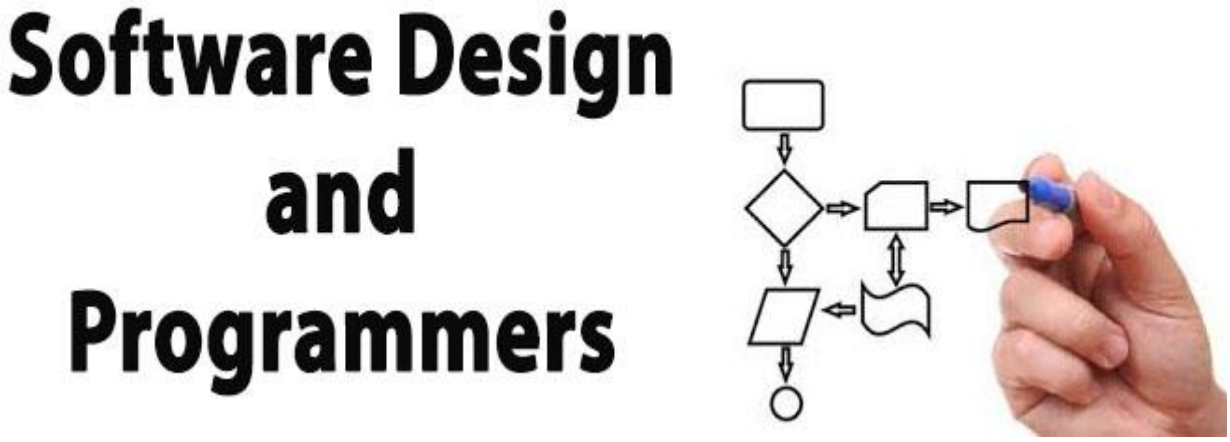
در این مقاله به توضیحاتی درباره طراحی نرم افزار و برنامه نویسی می پردازیم و رابطه بین طراحی،برنامه نویسی و توسعه نرم افزار را بازگو میکنیم .



مهارتهای طراحی نرم افزار وبرنامه نویسی چقدر مهم است؟

کار یک برنامه نویس قبل از هرچیز،نوشتن کد است.کد به عنوان زیرساخت نرم افزار نشان داده میشود، و قبل از شروع زیرساخت باید طراحی برنامه را تکمیل کرد. کار طراحی پروژه را طراحان نرم افزارهای تخصصی انجام میدهند. طراحان پس از طراحی ،طرح را به برنامه نویسان تحویل میدهندو برنامه نویسان آنها را تبدیل به کد میکنند.برنامه نویسان باید مهارت تبدیل طرح به کد را داشته باشند.

در اینجا میخواهیم یک طیف از توسعه نرم افزار را بازگو کنیم.

در پروژه های بزرگ و پیچیده به خصوص در سازمانهایی که دارای فرهنگ مهندسی نرم افزار سنتی هستند، در ابتدا سناریو به صورت دستی نوشته میشد.تخصصی شدن ،یک جزء کلیدی در این گونه پروژه ها است.تحلیلگران ،پروژه را تجزیه وتحلیل و نیازهای پروژه را جمع آوری می کند تا مشخصات تولید طرح را بدست آورد،که برنامه نویسان برای تولید کد نیاز به اطلاعات طراحی دارد.

در پایان ،مقابل این طیف ، بهترین مثال از برنامه نویسی) Extremeو XP مخفف Extreme Programming یا برنامه نویسی به روش سریع است. در اوایل دهـه نود Kent Beck در این فکـر بـود کـه روشـی بهتر برای طـراحی نرم افزار پیدا کند کـه سرانجام در سال 1966 به کمک همکارش Ward Cunningham در شـروع پروژه ای بـه نـام Daimler Chryslerبـه مفهـومی رسـید که به نتیجه آن Extreme Programming شد XP .شامل مجموعه ای از ارزشها، اصول

و روشهای مرتبط به هم است). نشان داده شده، که هیچکدام از آنها طراح نیستند،آنها فقط برنامه نویسانی هستند که مسئولیت طراحی سیستم را برعهده دارند.در این وضعیت جایی برای تخصص نیاز ندارند. برنامه نویسان نسبت به نیاز مردم به سمت مواردی مانند طراحی ,ساخت ,تست ,گسترش , داکیومنت نویسی ,آموزش و پشتیبانی ، برای توسعه نرم افزار میروند.

بسیاری از واقعیت ها جایی بین دو قطب (A،تقسیم مهندسی نرم افزار، که در آن طراحان، طرح را بسیار دقیق کامل میکنند وبه برنامه نویسان تحویل میدهند،و B،برای توسعه از تیم برنامه نویسی Extrem، بطور صحیح یا ضمنی استفاده میکنند که برنامه نویسان مسئولیت بخشی از طراحی را داشته باشد، اما نه همه آنرا. بخشی از کار برنامه نویس پر کردن نواقصی است که در طراحی وجود دارد.

یکی از چیزهایی که از همه نظر در این طیف مشترک ،وجود دارد : از نظر مهندسی نرم افزار برنامه نویسانی که فقط کد مینویسند،و همچنین طراحان نرم افزار ،همگی جزء برنامه نویسان هستند. همچنین همه برنامه نویسان ،طراح نرم افزار نیز هستند.متاسفانه اغلب آنها به رسمیت شناخته نشده اند که این منجر به تصورات غلطی درمورد ماهیت توسعه نرم افزار ،و نقش برنامه نویس و مهارتهایی که نیاز به وجود برنامه نویسان است؛ میشود.

مقاله ای درباره نرم افزار IEEE با نام "مهندسی نرم افزار کافی نیست"

نقل قول از مقاله:

تصور کنید که شما چیزی در مورد توسعه نرم افزار نمیدانید. بنابراین ، شما یک کتاب یا چیزی مشابه آن درباره مهندسی نرم افزار انتخاب و تهیه کنید ،تا در مورد آن چیزهایی یاد بگیرید.بدیهی است که مطالبی که میخوانید درباره مهندسی نرم افزار است .

میتوانید تصور کنید و سپس طراحی را شروع کنید ،آیا در این صورت به این نتیجه می رسید که نوشتن کد ساده تر از طراحی است. که کد فقط ترجمه ای از یک طراحی است ؟

وقتی که از آن نگهداری و پشتیبانی میشود،ممکن است این نتیجه گیری دوراز ذهن نباشدکه :تنها در روند طراحی درسطح کد نویسی ، جزئیات کوچکی که قادر به طراحی و اجرای رویه به کد گذاری هستند، تصمیم گیری میشود .

واقعا ؟چند بار یک سیستم کوچک و با اهمیت را به یک زبان برنامه نویسی بدون داشتن مشکلی طراحی کرده اید؟ و در پاسخ این را میگوییم که طرح ها در وهله اول برنامه نیستند. طرح جزییات انتزاعی بسیاری دارد که در نهایت باید کد گزاری شوند .

قسمت ترسناک از متون مهندسی نرم افزار که Whittaker و Atkin به طرز ماهرانه ای متون استانداردی که در دوره برنامه نویسی نرم افزار در کالج مورداستفاده قرارمیگیرند،را مورد تمسخر قرار دادند.

نهایتا،وقتی شما سعی در پیدا کردن بخشهایی که درباره برنامه نویسی است را دارید، تصمیم میگیرید کتاب مهندسی نرم افزار را بخوانید که متوجه میشوید بخشهایی از کتاب مهندسی نرم افزار اشتباه است، نگاهی به جدول محتویات میندازید اما قسمتهای دیگری را نشان میدهد.به عنوان مثال مهندسی نرم افزار :دسترسی حرفه ای، مسابقه بهترین متن مهندسی نرم افزار McGraw-Hill's، ویک برنامه لیست شده و یک طراحی که به برنامه ترجمه شده باشد ندارند. در عوض، این کتاب مملو از مدیریت پروژه، برآورد هزینه، و مفاهیم طراحی شده است.

با توجه به جهانی که در آن coding بی اهمیت به نظر می رسد؛ نظر غالب این است که بسیاری از کسانی که روی نرم افزارهای حرفه ای کار میکنند، فکر کنند تا یک راه جدید در مورد رابطه بین ماهیت طراحی و ساختار پیدا کنند. یکی از این روشها که به عنوان یک جایگزین برای روش های مهندسی نرم افزار بوجود آمده ،رویکرد مبتنی بر مهارت است، که تاکید بر اهمیت فرآیندهای پیچیده، تخصصی، و در دست دارد. برنامه نویسی Extremeیک مثال از یک روش مهارت محور است.

برنامه نویسی Extreme ، و تکنیک های وابسته به آن مانند Refactoring و " اولین تست از طراحی،" از کار توسعه دهندگان اسمالتاک ،کنت بک وکانینگهام با هم به وجود آوردند. بک و کانینگهام با هم مشغول به کار بودند تا بخشی از یک جنبش رو به رشد شی گرا، که در آن زبان اسمالتاک و عموم نقش مهمی بازی می کردند، دست یافتند. با توجه به پیت McBreen در پرسش برنامه نویسی Extreme، "این ایده که سورس کد است که طراحی را در جامعه اسمالتاک از s1980 گسترده است."

برنامه نویسی Extreme ، در هسته خود ایده ای از کد طراحی شده دارد و بهترین راه برای رسیدن به بهترین طراحی و بهترین کد با بالاترین کیفیت این است که عملیات طراحی و کد نویسی توسط همان برنامه نویسان به طور همزمان حفظ شود . Refactoring ، مفهوم کلیدی از XPاست ،که برای تغییر مجموعه ای از روش تدریجی ، به شیوه ای کنترل شده ودرج کدها برای طراحی، که بیشتر در نقش برنامه نویس به عنوان طراح اعمال نفوذ میکنند،بکار برده میشوند. دو مفهوم دیگر از کلید XP، "اولین تست طراحی" و تست اتوماتیک واحدها است ، در این ایده ، تنها کد طراحی شده است، اما اساس طراحی کامل نیست مگر اینکه آن از طریق آزمایش تایید شده باشد. البته، برنامه نویس از طریق واحد تست ،طراحی را بررسی میکند.

بسیاری از آنها به نتیجه رسیده اند که این یکی از دلایلی است که برنامه نویسی (Extremeوروشهای جنبش عمومی (Agile محبوبند ،مخصوصا برای کسانی که عشق به نوشتن کد دارند explicitly). یا (implicitly برنامه نویسان در تشخیص طراحی نرم افزار نقش حیاتی بازی میکنند ،حتی زمانی که به آنها برای ایجاد یا تغییر در طراحی مسئولیت داده نشده باشد .دانشگاهیان و پزشکان که طرفدار سنتی دیدگاه مهندسی نرم افزار بودند، از نظر اغلب آنها جای تاسف دارد که نتایج تحقیقات و انتشارات خود را دور بریزند و سریعا کارهای عملی که توسعه دهندگان نرم افزار انجام داده اند را اجرا کنند.

ریوز یک مقاله تاثیرگذار به نام "طراحی نرم افزار چیست؟ "در مقاله C ++مجله McBreen .منتشر شده. این مقاله که کنت بک تو ریوز ارائه می دهد برخی از بینش کلیدی نسبت به بحث در دست را دارد:

C++عمومیت یافته است زیرا نرم افزار طراحی و برنامه همزمان و آسان تر ساخته می شود .ریوز، [1992]

پس از بررسی در چرخه حیات توسعه نرم فزار و نتیجه گرفتن از آن ،از آنچه که درک میشود را برای برآورد کردن معیارهای مهندسی و طراحی سورس کدها و فهرست کردن آنها رابصورت داکیومنت نرم افزار درمی آوریم.ریوز، [1992]

مشکل قریب به اتفاق با توسعه نرم افزار که همه چیز بخشی از فرآیند طراحی می باشد. کدهای طراحی، تست و اشکال زدایی بخشی از طراحی،وآنچه ما درباره طراحی نرم افزار صحبت میکنیم نیز جزءی از طراحی است ".ریوز، [1992]

تست کردن فقط برای این نیست که بداند طراحی فعلی درست است، این مرحله را برای مشاهده روند پالایش ،طراحی کرده اند. ریوز، [1992]

مثالی دیگر : در تعریف کتاب آمده است که هر برنامه نویس باید درباره ی طراحی شی گرایی بداند، جونز میگوید: یکی از اهداف این کتاب این است که برنامه نویسان به صراحت از الگوهای طراحی که در کد خود ایجاد کردند ،آگاه باشند.

چه چیزی در مورد نقش برنامه نویس روی تعمیر و نگهداری یک پروژه ،و اهمیت بخشهایی از طراحی که متعلق به برنامه نویس است ،وجود دارد؟ بسیاری از برنامه نویسان از ابتدا روی توسعه سیستم هایی با نام تجاری جدید کار نمی کنند (حتی اگر همه آنها را ترجیح دهند) .در عوض، آنها در حال تعمیر و نگهداری سیستم های موجود هستند، ویژگی های جدیدی را درحالت تثبیت اشکال و هم در حالت توسعه، به سیستم اضافه میکنند و یا در غیر این صورت از مسیرهای جدیدی برای تکامل نرم افزار استفاده میکنند.

مهارتهای طراحی از لحاظ برنامه نویس مهمتر از زمانی است که یک پروژه در حالت تعمیر و نگهداری است . به همین دلیل بسیاری از شرکتها وقتی سیستم ها به حالت تعمیر و نگهداری میروند،کارکنان خود را کاهش میدهند .در واقع، هنگامی که به ویژگی های جدید برای توسعه نیاز است ،و یا زمانی که بخشی در سیستم نیاز به طراحی مجدد و یا تنظیم عملکرد دارند، اغلب از برنامه نویسانی که طراحی و پیاده سازی را انجام داده اند درتعمیر و نگهداری استفاده میشوند. (در ضمن، به ویژه یکی از مزیت هایی که برای یک توسعه دهنده کم تجربه می توان نام برد افزایش فرصت های یادگیری و کار با یک توسعه دهنده تعمیر و نگهداری است.)

در کتاب معروف تجزیه و تحلیل سیستم و طراحی، جرالد واینبرگ می نویسد:

طراحان موفق و دانا باید از وسوسه طراحی همه چیز در یک توده بزرگ جلوگیری کنند .در عوض، بهتر است آنها بخشی کوچکی بسازند، و رفتار واقعی آن بخش را تجزیه و تحلیل ، و پس از آن روند را برای قسمت بعدی تکرار کنند .این راه ، طراحی تکاملی است .

این فرایند از طراحی افزایشی در دو سطح ،ذهن طراح و واقعیت قرار میگیرد. اکثریت قریب به اتفاق تصمیمات طراحی ،امروزه توسط برنامه نویسان تعمیر و نگهداری (maintenance)گرفته میشود ، نه طراحان .

به این معنا نیست که در وضعیت بسیار خوب از طراحی کسی که میتواند تعمیر و نگهداری را انجام دهد به همان اندازه نیز میتواند باعث بدترشدن وضعیت سیستماتیک شود .

برنامه نویسی Extreme به طور کامل پذیرای ایده وینبرگ " تعمیر و نگهداری طراحی" با این نظر از ریوز که " فقط ساخت طرح و تست آن نسبت به انجام هر چیز دیگری ارزان تر و ساده تر است" ،است آنها را با هم پیوند میزند . در نتیجه یک فرایند ،که واینبرگ به آن برچسب خطرناک و ریسک را در نظر گرفته است. این کار باعث میشود برنامه نویسی Extreme ،که با استفاده از روش هایی مانند تست طرح برای اولین بار ، تست خودکار، تکرار کوتاه، برنامه نویسی pair، و refactoring به طور همزمان برای کاهش ریسک انجام دهندو ریسک را به یک مزیت تبدیل کند.

اگر واینبرگ درست بگوید که "اکثریت قریب به اتفاق تصمیمات طراحی که امروزه توسط برنامه نویسان maintenanceبه اجرا گذاشته شده ، نه طراحان, که از نظر بسیاری از ما، مهارت های طراحی موجود در برنامه نویسان برای پروژه های تعمیر و نگهداری ، مهم تر می باشند. در نهایت، مراقبت ویکپارچگی مورد نیاز است که منجر به تولید و گسترش یک سیستم نرم افزاری میشود.

در مرحله تعمیر و نگهداری ،برنامه نویس /طراح به طور مداوم باید بین دغدغه های موجود در رقابت، و حفظ ماندگاری کد و رابط خارجی در سیستم موجود، تعادل ایجاد کنند و تمایل به استفاده از بهترین و صحیح ترین راه حل؛ تمایل به استفاده از آخرین تکنیک ها؛ تمایل به ایجاد بهترین تجربه برای کاربر؛ نیازی نقض منطقی و یا بی ثبات کردن سیستم تولید ندارد ؛ شارژ برای محافظت از اطلاعات مهم تولید شده ، تمایل یا نیاز به بهبود طراحی و یا کدی که در حال حاضر وجود دارد .و به یک مهارت و راه حل نیاز داریم تا درصورت امکان در سیستم های موجود با حداقل اختلال و از کار افتادگی مستقر شود .

آلن کوپر، که در سال 2002 گفت :

این دو جنبش در دنیای نرم افزار، مهندسی و مهارت به نظر می رسد در جهت مخالف در حال حرکت اند. مشکل این است که شما نمی توانید هم به روش مهارتی تمرکز کنیدهم بر مشکلات مهندسی و بالعکس .بنابراین در موقعیتهایی باید آنها را جداکنیم وتلاش کنیم تا از روش مهارتی منحصر به فرد برای انجام پروژه های بزرگ استفاده کنیم. در کجا حک شده که ما مجبور به استفاده از یک روش برای تمام پروژه ها هستیم؟

در تاکید بر نقش برنامه نویس به عنوان طراح ،از نظر طراحان متخصص ،یک ایده بد است.برنامه نویسی که روی پروژه ای که شخص دیگری طراحی آن را انجام داده ، نباید ازطرحها چشم پوشی و آنها را رد کند .کسانی که در چند پروژه ایده ی، برنامه نویسی را به عنوان طراح پذیرفته اند(مانند پروژه . (xpآنها خطر کرده اند ؛ معمولا در برنامه نویسان شناختی از نقش ذاتی طرح وجود ندارد.

خطری که در هر دو حالت وجود داردWhittaker .و اتکین نوشتند" :ماهیت طرح این است که بسیاری از آنها جزئیات انتزاعی هستند که در نهایت باید کد گذاری شود ".طراحان باید به چیزهایی که درطرح های خود خلاصه میشوند حساس باشند و برنامه نویسان ، برای انجام بهتر کار باید مقداری از طراحی که مربوط بکار خود است را درک کنند،و مهارتهای طراحی خود را افزایش دهند تا آنها را به بهترین شکل ممکن انجام دهند و برای طراحی بطور مداوم طرحی از فریم ورکها به آنها داده میشود.

مشکلاتی که یک برنامه نویس و طراح ممکن است به آن برخورد کنند این است که توسعه دهندگانی که روی طرح طراحان کار میکنند،از روشها و برندهای جدیدی روی پروژه استفاده میکنند،که ممکن است با طرح طراحان مناسبت نداشته باشد.ویا ممکن است برنامه نویس تازه کار باشد و دانش کمی از پایگاه داده داشته باشد و همه چیز را درباره سیستم های پیچیده ندانند.همچنین، هرگز از قبل باهم کار نکرده باشند، و هیچ آگاهی از نحوه کد نویسی هم نداشته باشند.

با توجه به مجموعه ای از شرایط ،طرحی بسیار دقیق است.که با اسپل کردن تک تک جزئیات ،با استفاده ازنوشتن شبه کدها و بسیاری از عبارتها ی SQLکه برنامه نویسان نیاز دارند ، هرکجا که میتوانیم توضیحاتی برای درک بهتر اضافه کنیم.ما نمیتوانم درباره بخشی از طراحی که برنامه نویسان برداشت غلطی از روی قصد یا منطق دارند، شانسمان را از دست بدهم. اما این روش را ظرف مهلت مقرر بایدانجام دهیم ، وبرای آنها بسیار قابل اعتماد باشیم.

در وضعیت دیگر در پروژه های مختلف (این یکی یک پروژه تعمیر و نگهداری)، نیاز به طراحی یک خصوصیت داریم که باید آن را بسازیم که با توجه به کسب و کار مردم ،تجزیه و تحلیل را انجام میدهم، و بدنبال چشم اندازی درباره چگونگی استفاده از ، خصوصیت های جدید در کار باشیم. به عنوان مثال برنامه نویسی که مدتی بر روی این پروژه بوده ، و دانش خوبی از دامنه و سیستم موجود دارد. علاوه بر این، میدانیم که او یک رمز گذار عالی با مهارت های طراحی و با استعداد است .در این وضعیت، ما مجبور به نوشتن یک طرح بسیار دقیق نیستیم .برای او : توضیحی کوتاه درباره نیاز / طراحی داکیومنت ، که شرایط کلی که متناسب با نرم افزارهای موجود است را توصیف کند ، به چند قانون کلیدی کسب و کار و محدودیت ها در آن نوشته شود پی ببرد، و مفهوم یک رابط کاربری را توصیف کند ، و برخی از جزئیات که لازم است در آن قید شود را می نویسیم.

در جلسه ای کوتاه با توسعه دهنده باید داکیومنت و توضیحات طراحی را به آن بدهیم که تا پایان کارش از آن استفاده کند. و داکیومنت را در این دوره حفظ کند.

به عنوان اخرین مثال ، وقتی طراحان،در مواقع نیاز ،جلساتی با توسعه دهندگان نداشته باشند و در این صورت در پروژه داکیومنتی برای تجزیه و تحلیل وجود نخواهدداشت. اگر از روشهای مناسب استفاده نکرده باشنددر این صورت طرح شان از نظر برنامه نویسان ناقص می باشد.

همان طور که تصور میکنید ،توسعه دهندگان در این وضعیت بسیار ناامید میشوند .آنها برای پر کردن جاهای خالی شروع به پرسیدن سوالات مختلف از مردم میکنند .آنها مجبورند برگردند و از طراحان سوالاتشان را بپرسند که بیشتر ناامیدشان میکنند. زمانی که توسعه دهندگان تفسیر و طراحی همه چیز را خود انجام میدهند که بنظرشان بهترین فکر است، ولی از نظر طراحان اینگونه نیست وآنها می گویند" نه، آن راه آنچه تصور میشود نیست" که البته این مورد باعث سرخوردگی در هر دو طرف میشود .

همه این بحث در مورد رابطه بین طراحی و ساخت و ساز در نظر گرفته شده است . سه نکته برای تأکید:

اول :طراحان خصوصیاتی ایجاد میکنند و آنها را به افراد دیگر واگذار میکنند که باید از تکمیل نبودن هر طرح آگاه باشند. باید توجه خود را برای طراحی به مخاطبین بدهند. و همچنین نگرانی موقعیتی مانند زمانبندی و ریسک به آنها داده شده باشد.

دوم: بسیار مهم است که برنامه نویسان دانشی از اصول و تکنیک های طراحی نرم افزارداشته باشند؛ و در صورت نیاز در مورد آن مطالعه کنند و در زمان طراحی کد بزنند. تنها تفاوت برای یک برنامه نویس از یک وضعیت به بعد موضوع رتبه است :بهترین تصمیمات را بگیرند و دانش خود را از طراحی یک اثر مستقیم بر روی کیفیت کار خود بگذارندوبرای بدست آوردن محصولی بابهترین کیفیت تلاش کنند. تسلط به زبان ، ابزار، و پلت فرم ها کافی نیست.

سوم:مدیران و رهبران مسئول در توسعه پروژه ها نقش طراحان را در مرحله ساخت و ساز به رسمیت بشناسد و به عنوان یک مزیت مهم در فرایند کلی فرصتهایی برای نفوذ و اطمینان ایجاد کنند . با چشم پوشی از این واقعیت که برنامه نویسان همان طراحان هستند.واینکه طرح برنامه نویسان نمیتواند کامل باشد ،سبب میشود انواع اشتباهات هزینه بر ،مشکلات کیفیتی ،و برنامه های Overrun به وجود بیایند.