# اقتصاد دیجیتال

## فرصت ها و چالش های فن آوری دیجیتال

### مشکلات عدم نیازسنجی های  قبل و بعد از ساخت نرم افزار

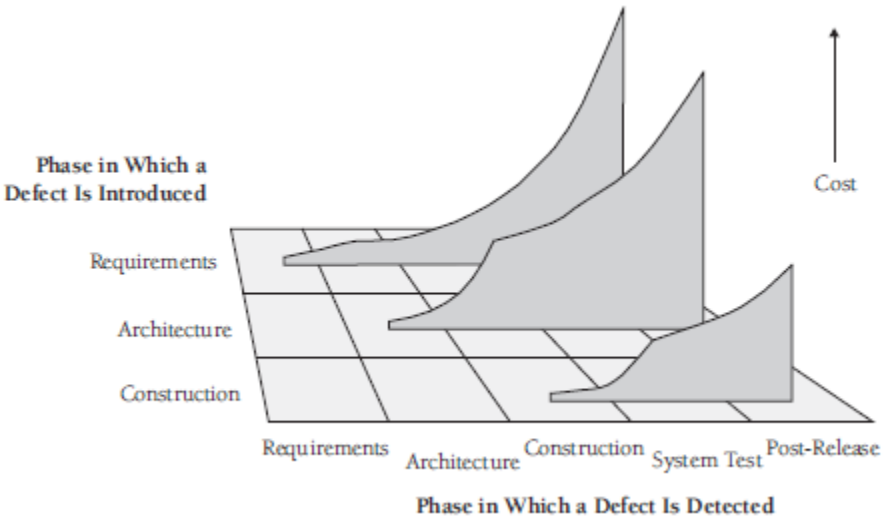
## مقدمه :

چرا تعداد زیادی از پروژه های نرم افزاری با شکست رو به رو می شود ؟

با وجود اینکه هم نیرو داریم و هم بودجه کافی و هم زیر ساخت مناسب .

چگونه می توان با مشکلاتی که در ساخت نرم افزار پدید می آید ، برخورد مناسبی داشته باشیم. به گونه ای که متحمل کمترین خسارت های مالی و زمانی (تاخیرات تحویل ) در ساخت نرم افزار خود شویم .

با توجه به نمودار مربوط به ساخت نرم افزار (برگرفته از کتاب Code Complete ) در فازهای مختلف ، متوجه میشویم که بهتر است که در همان ابتدا، نقص های نرم افزاری شناخته و بر طرف شود تا هزینه های سنگینی که در ادامه ساخت نرم افزار با آن مواجه می شویم را ، کاهش دهد.




## نیاز سنجی های نرم افزار

مشکلاتی که در ادامه بحث می شوند در چندین دسته مختلف قرار می گیرند که خلاصه ای از آنها را در زیل بحث کرده ام .

دسته اول :

دلیل این که تعداد زیادی از پروژه های نرم افزاری با شکست مواجه میشوند این است که ، از زمینه کاری مورد نظری که می خواهیم برای آن نرم افزار بسازیم ، شناخت کافی نداریم. اگر کمی جزئی تر شویم بهتر است بگوییم که نیاز های مستند (داکیومنت شده ) نداریم .

در این صورت بهتر است از یک  فرد خبره ی آن حوزه استفاده کنیم . از او بپرسیم کار کارمندان مختلف کارهای خودشان را چگونه به طور دستی ، انجام میدهند . چه نقص هایی در سیستم کاری دستی وجود دارد که ما می خواهیم آن را ماشینی  (نرم افزاری) کنیم و چه نیازهایی در آینده پیش می آید که قابل حل با سیستم دستی نمی باشد یا  هزینه های (مالی و زمانی ) آن در صورت امکان حل مشکل ،  بسیار سنگین می باشد .

این روش در مقابل روشی قرار می گیرد که برنامه نویسان و طراحان نرم افزار، بدون توجه به نیاز های حیاتی  که در حیطه  زمینه کاری مربوطه وجود دارد ، به طور خودسرانه شروع به طراحی نرم افزار می کنند. با توجه به این مشکل که آنها  شناخت اندکی از آن حوزه دارند و از روابط و ضوابط  و استاندارد های  موجود  در آن حوزه کاری ، اطلاعی ندارند یا اطلاعات ناقصی دارند .

دسته دوم :

عدم وجود یک فرد خبره در حوزه کاری مرتبط ، برای آشنایی با حوزه کاری مربوطه :

در بعضی از مواقع پیش می آید که می خواهیم برای یک سازمان یا یک شرکت یک نرم افزاری را طراحی کنیم  بطوریکه ، در آن  حوزه یک فرد خبره را نمی توان پیدا کرد که طراحان نرم افزار بتوانند از او به عنوان یک مرجع استفاده کنند . برای حل این مشکل دو راه حل احتمالی وجود دارد .

1. استفاده از نرم افزار های مشابه ساخته شده در آن حوزه ی کاری و بررسی و اعتبار سنجی آن :
در صورت وجود نرم افزارهای مشابه می توان آنها را خریداری کرد و نیازهای اولیه را با توجه به شناختی  ابتدایی که از نرم افزارهای مشابه حاصل میشود را با حوزه کاری مربوطه هماهنگ (Synchronize) کرد و تغییرات مربوطه را در نیاز سنجی های خود ، مستند کنیم .

2. عدم دسترسی به نرم افزارهای  مشابه برای شناخت حوزه کاری مربوطه به دلایل مختلف اعم از اختصاصی بودن نرم افزار یا مشکل مالی در تهیه نرم افزار های مشابه .

در این صورت تنها راهی که باقی می ماند و بهترین راه موجود در حال حاضر  ، این است که تعدادی از طراحان نرم افزار را در آن شرکت به جایی افرادی  (کارکنانی ) که می خواهیم برای آنها سیستم نرم افزاری طراحی کنیم، قرار دهیم .
واضحتر بگویم . یعنی بگذاریم که ساعاتی از روز را طراحان نرم افزار در بخش های مختلفی که می خواهیم برای آن نرم افزار طراحی کنیم و روال های کاری آن را ماشینی کنیم ، مانند کارمندان کار کنند و نیاز های مربوطه را مستند کرده و  آن ها را با هم مورد بررسی قرار داده و در نهایت با هم به یک جمع بندی کلی برسند.

دسته سوم : عدم استفاده از ابزارهای مستند سازی و مدیریت ساخت  نرم افزار :

این موضوع را باتوجه به این نکته در نظر می گیریم که شناخت کافی از زمینه کاری مربوطه داریم ، اما ابزاری مناسب جهت مستند سازی نرم افزار نداریم . فکر می کنیم که مستند سازی نرم افزار را می توان با نرم افزار هایی مانند WORD و EXCEL و VISIO انجام دهیم .کاری اشتباهی که اکثر مهندسان نرم افزار به علت عدم آشنایی با ابزارهای استاندارد( ویژه ) مستند سازی نرم افزار انجام می دهند . نرم افزارهایی مانند IBM Rational Software Architect و Visual Paradigm برای پروژه هایی که زمان بیشتر و مستندات دقیق تری می خواهد و ابزار های Confluence و JIRA هم برای نرم افزارهایی که با متودولوژی های چابک (Agile) ساخته می شوند .

دسته چهارم :

درطی روال ساخت نرم افزار در صورتی که مستندات ما خوب است و از ابزار های استاندارد مستند سازی و طراحی نرم افزار استفاده می کنیم ، مشاهده می کنیم که مشکلاتی پیش می آید. علت به وجود آمدن اکثر این مشکلات ، عدم شناخت و دانش کافی در حوزه طراحی نرم افزار است که منوط به بروز مشکل در یکی از دسته های زیر می باشد .

1. عدم استفاده از متودولوژی مناسب جهت طراحی و ساخت نرم افزار.
2. ناآشنایی با الگوهای مناسب طراحی از قبیل :Analysis Patterns, Refactoring Patterns, Design Patterns
3. به موقع تست نکردن نرم افزار و عدم استفاده از Test Patternهای مناسب .

دسته پنجم :

بروز مشکل به دلیل تغییرات در سازمان مربوطه در حین طراحی نرم افزار:

این مشکل کشنده ترین مشکل است که باید طراحی نرم افزار را به گونه ای انجام داد که بتوان امکان Continuous Delivery و Continuous Integration را داشته باشیم که الگو ها و منابع خاص خود را در این زمینه دارد .

منابع :


1.Karl Wiegers and Joy Beatty. 2013. Software Requirements . Microsoft Press

2. David Budgen . 2003 . Software Design . PEARSON Education.

3. Martin Fowler . 2007. Analysis Patterns . Addison Wesley.

4.Serge Demeyer, Stephane Ducasse, Oscar Nierstrasz . 2008.Object-Oriented Reengineering Patterns . Addison Wesley .

5.Roger Pressman . 2010. Software Engineering . Mc Graw Hill .

6.Steve McConnell. 2004 . Code Complete . Microsoft Press.

7.Gerard Meszaros . 2007. XUNIT TEST PATTERNS . Addison Wesley Martin Fowler Signature Book

8.Pual M.Duvall , Steve Matyas, Andrew Glover . 2007. Continuous Integration . Martin Fowler Signature . PEARSON Education.

9.JEZ Humble , David Farley . 2001. Continuous Delivery . Martin Fowler Signature . PEARSON Education .

10. FREEDERICK P.BROOKS , JR . 2010 . The Design of Design . PEARSON Education.