Embracing Change with Extreme Programming

http://somamos.blogfa.com

فهرست

۱- پیش گفتار

در راستای تولید محتوای فارسی برای موضوعات مورد علاقه و باارزش، مطالب برگزیده پس از ترجمه توسط چند تن از عزیزان، از طریق وبلاگ در اختیار خوانندگان قرار خواهد گرفت. با وجود همه کمبودها، امیدوارم که این کار اثرگذار و مفید باشد.

متن زیر ترجمه‌ی مقاله Embracing Change with Extreme Programming نوشته Kent Beck است که در IEEE Software در سال ۱۹۹۹ منتشر شده است.

این مقاله توسط دوست گرامی جناب آقای مهندس **مهدی نگاهی** ترجمه شده است.

یوسف مهرداد بی‌بالان

somamos.blofa.com

۲۳ فروردین ماه ۱۳۹۲

۲- مقدمه

XP مسیر فرایند توسعه نرم‌افزار رایج را تغییر می‌دهد. به جای برنامه‌ریزی، تحلیل و طراحی برای آینده بسیار دور [و یک‌باره برای کل پروژه]، برنامه‌نویسان XP همه این کارها را در تمام مدت توسعه انجام می‌دهند اما هر بار، مقدار کمی از آن را.

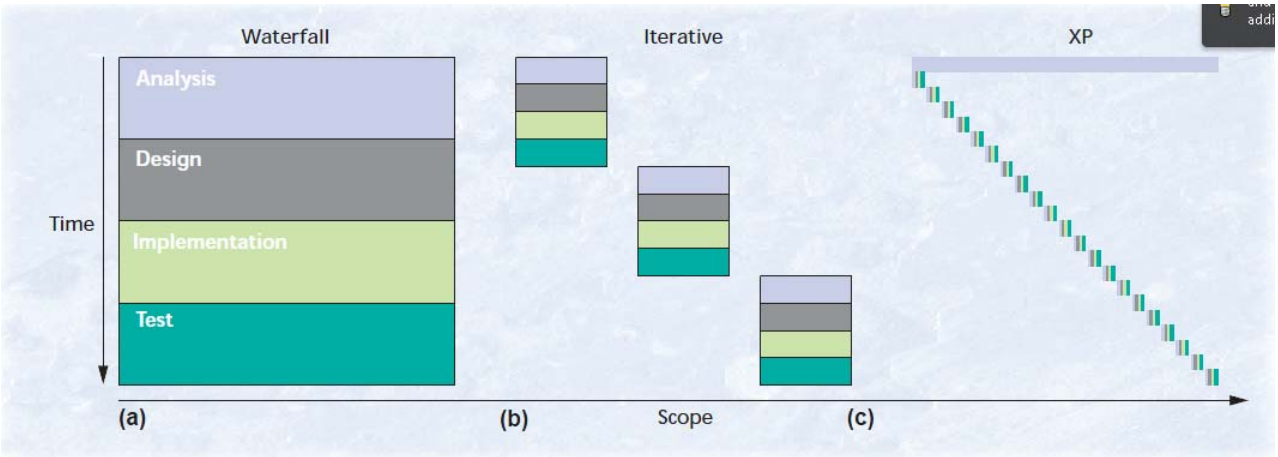
ابتدا مدل آبشاری به وجود آمد (شکل ۱) که در آن می‌خواستیم کاربران را مجاب کنیم یک‌جا و دقیق بگویند که چه نیازهایی دارند. می‌خواستیم سیستمی طراحی کنیم که همه ویژگی‌ها را داشته باشد، می‌خواستیم کد را بر اساس طراحی بنویسیم، می‌خواستیم نرم‌افزار را آزمایش کنیم تا از تحویل درست ویژگی‌ها مطمئن شویم. می‌خواستیم که همه چیز خوب باشد.

ولی همه چیز خوب نبود. کاربران یک‌جا و دقیق نمی‌گفتند که چه می‌خواهند. آنها نیازهایشان را نمی‌دانستند. حتی گفته‌های خود را نقض می‌کردند. فکرشان نیز تغییر می‌کرد. ولی مشکل فقط کاربران و مشتریان نبودند. ما برنامه‌نویسان نیز فکر می‌کردیم با انجام ¾ کار طبق برنامه، پیشرفت بزرگی در پروژه کرده‌ایم در حالی که عملاً فقط ۱/۴ کار انجام شده بود.

چرخه‌های بلندمدت توسعه خوب نبودند چرا که نتوانستند خود را با تغییرات سازگار کنند. این موضوع شاید بدین معناست که آن چه نیاز داشتیم ایجاد چرخه‌های کوتاه‌تر بود. بنابراین همان طور که شکل ۲ نیز نشان می‌دهد مدل آبشاری عامل ایجاد تکرارها[1] بوده است.

مدل آبشاری یکباره ظهور نکرد، بلکه واکنشی منطقی به تحقیق تکان‌دهنده‌ای بود که نشان می‌داد هزینه تغییر بخش کوچکی از نرم‌افزار با گذشت زمان به صورت چشم‌گیری افزایش می‌یابد. اگر این موضوع درست باشد، تمایل خواهید داشت بزرگ‌ترین و مؤثرترین تصمیمات را در زودترین زمان ممکن بگیرید تا مجبور به پرداخت هزینه کلان بابت آنها نشوید.

جامعه دانشگاهی مهندسین نرم‌افزار که هزینه زیاد تغییر نرم‌افزار را به عنوان یک چالش معرفی کرده است، تکنولوژی‌هایی نظیر پایگاه داده رابطه‌ای، برنامه‌نویسی ماژولار[2] و مخفی‌سازی اطلاعات را ایجاد کرده است. چه می‌شد اگر این تلاش‌های دشوار نتیجه می‌داد؟ چه می‌شد که ما در کاهش هزینه تغییرات مداوم، خوب عمل می‌کردیم؟ چه می‌شد که مجبور نبودیم برای تسویه‌حساب با ساطور به جان مدل آبشاری بیفتیم؟ چه می‌شد اگر می‌توانستیم مدل آبشاری را در مخلوط‌کن بریزیم [و با هر چیز دیگری مخلوط کنیم]؟



تصویر ۱: تکامل مدل آبشاری (a) و چرخه طولانی آن (تحلیل، طراحی، پیاده‌سازی و آزمایش) به مدل کوتاه‌تر با چرخه‌های تکرارشونده به عنوان مثال مدل حلزونی[۳](b) و از آنجا به c) XP) با ترکیب فعالیت‌ها و انجام مقدار کمی از آنها در هر بار ولی مدوام در طول فرایند توسعه نرم افزار

۳- تجربه‌های[۴] XP

در زیر توضیح مختصری در مورد تجربه‌های مهم XP آورده شده است.

۱- بازی برنامه‌ریزی(Planning Game)

مشتریان محدوده و زمان‌بندی انتشار[۵] را بر اساس برآورد برنامه‌نویسان تعیین می‌کنند و برنامه‌نویسان فقط کارکردهای درخواست‌شده در داستان‌ها[۶] را در تکرار جاری پیاده‌سازی می‌کنند.

۲- انتشارهای کوچک(Small Releases)

سیستم طی چند ماه –کوتاه‌ترین زمان ممکن- و قبل از حل کل مسأله برای استفاده آماده و راه‌اندازی می‌شود. انتشارهای جدید به کرات در بازه‌های روزانه تا ماهانه ساخته و منتشر می‌شوند.

۳- استعاره (Metaphor)

چارچوب سیستم با استفاده از استعاره یا مجموعه‌ای از استعاره‌ها که بین مشتری و برنامه‌نویسان به اشتراک گذاشته شده، تعریف می‌شود.

۴- طراحی ساده(Simple Design)

در هر لحظه، طراحی همه آزمونها را پشت سر می‌گذارد، هر آن چه که برنامه‌نویسان نیاز دارند، در اختیارشان قرار می‌دهد و کد تکراری در آن وجود ندارد و حاوی کمترین تعداد کلاس و متد است. خلاصه این قاعده این است: "هر چیزی را یک بار و فقط یک بار بیان کنید."

۵- آزمایش‌ها(Tests)

برنامه‌نویسان لحظه به لحظه آزمونهای واحد[۷] می‌نویسند که باید تجمیع و به درستی اجرا شوند. در هر تکرار، مشتریان آزمونهای کارکردی[۸] را برای داستان‌ها می‌نویسند که اجرای آنها نیز ضروری است. اجرای آزمونها در پایان تکرار منجر به شناسایی ایرادهای محصول می‌شود. در چنین شرایطی نیاز به تصمیم‌گیری سازمانی است تا از بین تحویل محصول با ایرادهای شناسایی‌شده یا قبول دیرکرد انتشار محصول پس از رفع ایرادها، یکی انتخاب شود.

۶- بازسازی(Refactoring)

طراحی سیستم با تغییر طراحی موجود که همه آزمونها را پشت‌سر گذاشته، تکامل می‌یابد.

۷- برنامه‌نویسی دو نفره(Pair Programming)

همه کد برنامه به صورت دو نفره با یک نمایشگر، صفحه کلید و موشواره -یک دستگاه- نوشته می‌شود.

۸- یکپارچه‌سازی مداوم(Continuous Integration)

کد جدید حداکثر بعد از چند ساعت با سیستم موجود یکپارچه می‌شود. پس از آن، سیستم دوباره ساخته[۹] و همه آزمونها اجرا می‌شود، در صورت ناموفق بودن آزمایش، کد جدید رد می‌شود.

۹- مالکیت جمعی(Collective Ownership)

در صورت امکان بهبود، هر برنامه‌نویس موظف است هر کدی را در هر جایی از سیستم و در هر زمانی اصلاح کرده و بهبود بخشد.

۱۰- مشتری مقیم(On-Site Customer)

مشتری به صورت تمام وقت در کنار تیم تولید است.

۱۱- کارهفتگی ۴۰ ساعته(40-Hour Weeks)

هیچ کس نمی‌تواند دو هفته متوالی اضافه‌کاری داشته باشد. حتی اضافه‌کاری غیرمتوالی بیش از حد، نشانه وجود مشکلی عمیق است که نیاز به شناسایی و رسیدگی دارد.

۱۲- محیط کار باز(Open Workspace)

اعضای تیم در یک سالن بزرگ کار می‌کنند که دور تا دور آن دارای پارتیشن‌های کوچک است. برنامه‌نویسی دونفره بر روی کامپیوترهایی انجام می‌شود که در وسط سالن قرار دارند.

۱۳- فقط قواعد(Just Rules)

به عنوان بخشی از تیم XP متعهد به اجرای قواعد هستید. اما به یاد داشته باشید که اینها فقط یک سری قاعده‌اند. اعضای تیم می‌توانند قواعد را در هر زمانی تغییر دهند به شرطی که بر سر روش ارزیابی تغییرات ناشی از آنها توافق داشته باشند.

۴- آناتومی(کالبدشناسی) XP

XP مسیر فرایند رایج توسعه نرم‌افزار را تغییر می‌دهد. XP با استفاده از کاهش هزینه اعمال تغییرات نرم افزار، به جای انجام یک‌باره برنامه‌ریزی، تحلیل و طراحی برای آینده‌ای بسیار دور، آنها را به صورت مستمر در تمام مدت توسعه و در هر بار مقدار کمی از آنها را انجام می‌دهد. (بخش "تجربه‌های XP" در صفحه قبل، نگرش فلسفی و تجربه‌های XP را نشان می‌دهد. این تجربه‌ها به گونه‌ای طراحی شده‌اند که امکان استفاده همزمان از آنها وجود داشته باشد و تلاش برای استفاده یکی از آنها، خیلی زود منجر به استفاده از بقیه گردد.

۴-۱- چرخه توسعه XP

در شکل ۲، XP در دوره‌های زمانی مختلفی از سالیانه تا روزانه نشان داده شده است. مشتری انتشار[1] بعدی را با انتخاب باارزش‌ترین ویژگی‌ها (که در XP داستان نامیده می‌شوند) از بین داستان‌های موجود مشخص می‌کند. مشتری انتخاب را با اطلاع از هزینه پیاده‌سازی هر یک از داستانها و سرعت پیاده‌سازی تیم انجام می‌دهد.

مشتری داستانهای تکرار بعدی را نیز با انتخاب باارزش‌ترین داستان‌های باقی‌مانده از انتشار و اطلاع از هزینه هر یک از آنها و سرعت تیم مشخص می‌کند. برنامه‌نوسان داستان‌ها را به وظیفه‌های[2] کوچک‌تری تبدیل می‌کنند تا توسط هر یک از آنها قابل انجام باشد.

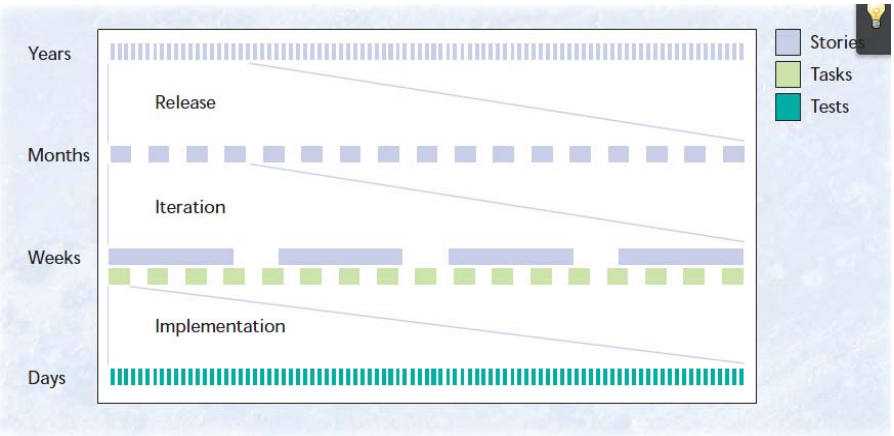
سپس هر برنامه‌نویس یک وظیفه را به مجموعه‌ای از موردهای آزمون[3] تبدیل می‌کند. موردهای آزمون نشان‌دهنده پایان درست هر وظیفه هستند. هر برنامه‌نویس با همراهی یکی دیگر از برنامه‌نویسان -همکار-، ابتدا موردهای آزمون را می‌نویسد(در

[1] Release

[2] Tasks

[3] Test cases

این مرحله اجرای آنها مؤفقیت‌آمیز نیست چون کد برنامه نوشته نشده است) و سپس با طراحی و نوشتن کدهای برنامه باعث اجرای درست موردهای آزمون و پشت‌سر گذاشتن آنها می‌شود. طراحی با رعایت اصل "حفظ ساده‌ترین طراحی ممکن برای کل سیستم" انجام می‌شود.



شکل ۲: XP در بازه‌های زمانی مختلف. در بازه ماه و سال، داستانهای انتشار جاری و انتشارهای آینده وجود دارند. در بازه هفته و ماه، با داستان‌های تکرار جاری و داستانهای باقی‌مانده از انتشار جاری سروکار دارید. در بازه روز و هفته، با وظیفه‌هایی که روی آنها کار می‌کنید و بعد با وظیفه‌های باقی‌مانده از تکرار جاری رو به رو هستید. در بازه دقیقه و روز، با موردآزمونهایی که روی آنها کار می‌کنید و بعد بقیه موارد آزمون قابل تصور سروکار دارید.

۴-۲- داستان‌ها

XP به دوره قبل از اولین ورود سیستم به مرحله بهره‌برداری[۴] توجه ویژه‌ای دارد[دوره منجر به تولید نسخه یک نرم‌افزار]. این دوره می‌تواند منجر به ناهنجاری خطرناکی در پروژه شود و از این رو باید در سریع‌ترین زمان ممکن طی گردد. به هر حال پروژه باید از جایی شروع شود.

این تصمیم که سیستم چه کاری می‌تواند انجام دهد و چه کاری بهتر است انجام دهد، اولین تصمیم پروژه است. این تصمیم معمولاً در حوزه تحلیل است(مستطیل آبی کم رنگ در بالای شکل C.۱). تا زمانی که ندانید چه چیزی باید پیاده‌سازی شود، نمی‌توانید برنامه‌نویسی را شروع کنید.

نتایج تحلیل در قالب داستانها در کنار هم قرار می‌گیرند. می‌توانید آنها را مجموعه‌ای از موردهای کاربرد[۵] فرض کنید که هر یک روی کارت نمایه‌ای[۶] نوشته شده است. هر داستان باید کسب‌وکار محو[۷]، آزمون‌پذیر و قابل برآورد باشد.

[4] Production

[5] Use case

[6] Index card

[7] Business-oriented

یک ماه زمان مناسبی برای شناسایی داستانهای یک پروژه ده نفر سال (ده نفر در یک سال) است. قبول داریم که این زمان برای شناسایی **کامل** همه موارد کافی نیست. اما توجه داشته باشید که تا پیاده‌سازی شروع نشود، نمی‌توان همه موارد را کامل و دقیق شناسایی کرد حتی اگر تا ابد هم وقت داشته باشیم.

۴-۳-انتشار[8]

توجه کنید که مطابق شکل ۲، همه داستانها در ابتدا پیاده‌سازی نمی‌شوند. بلکه مشتری کوچکترین مجموعه از با ارزش‌ترین داستانهایی را انتخاب می‌کند که در کنار هم معقول و منطقی هستند. ابتدا این مجموعه از داستانها پیاده‌سازی شده و در محصول نهایی قرار می‌گیرند و سپس باقی‌مانده‌ها پیاده‌سازی می‌شوند.

انتخاب داستانهای هر انتشار تا حدودی شبیه خرید مواد غذایی از فروشگاه است. وقتی با ۱۰۰ دلار به فروشگاه می‌رویم، به خریدهای ضروری(با الویت) فکر می‌کنیم، به قیمت اجناس نگاه می‌کنیم و بعد تصمیم می‌گیریم که چه چیزهایی خریداری کنیم. [در اینجا بودجه موجود برای خرید -۱۰۰ دلار- قابل افزایش نیست. رویکرد دیگر این است که ابتدا اجناس ضروری انتخاب می‌شوند و بعد بودجه لازم برای خرید آنها با جمع قیمتها، براورد می‌شود-مثلاً ۱۵۰ دلار-].

در بازی برنامه‌ریزی(فرایند برنامه‌ریزی XP)، داستانها معادل اجناس و برآورد هر داستان معادل قیمت است[به عنوان مثال اگر براورد پیاده‌سازی داستانی، ۳ نفر-روز باشد، یعنی قیمت آن ۳ واحد است]. بودجه با اندازه‌گیری خروجی تیم محاسبه می‌شود که مبنای آن جمع برآورد داستانهای انجام شده در واحد زمان است.

مشتری می‌تواند به دو شیوه زیر عمل کند: مجموعه‌ای از داستانها را انتخاب کند و برنامه‌نویسان تاریخ پایان پیاده‌سازی را محاسبه کنند [موقع خرید، اجناس را انتخاب می‌کنید و فروشنده، مبلغ قابل پرداخت را محاسبه می‌کند] یا تاریخ پایان را مشخص کند و برنامه‌نویسان بودجه را محاسبه کرده و سپس مشتری داستانها را یکی پس از دیگری انتخاب کند تا جمع برآوردهای آنها با بودجه برابر شود.[موقع خرید، بودجه را به فروشنده اعلام می‌کنید و سپس یکی یکی اجناس را انتخاب می‌کنید و هر بار جمع قیمتها محاسبه می‌شود تا از بودجه بیشتر نشود].

۴-۴- تکرار(Iteration)

هدف هر تکرار، افزودن داستانهای جدیدِ آزمایش شده و آماده‌ی استفاده به محصول است. فرایند با طرحی شروع می‌شود که در آن داستانهای انتخاب شده برای پیاده‌سازی و چگونگی انجام آنها توسط تیم مشخص شده است. هنگامی که تیم در حال

[8] Release

پیاده‌سازی است، مشتری آزمونهای کارکردی[9] را مشخص می‌کند. آزمون‌ها در پایان تکرار اجرا شده و تیم برای تکرار بعدی آماده می‌شود.

برنامه‌ریزی تکرار با درخواست دوباره از مشتری برای انتخاب باارزش‌ترین داستان‌ها آغاز می‌شود با این تفاوت که این‌بار، داستانها از بین داستانهای باقی‌مانده از انتشار انتخاب می‌شوند. داستان‌ها توسط تیم به مجموعه‌ای از وظایف[10] شکسته می‌شوند –وظیفه کاری است که یک نفر می‌تواند طی چند روز انجام دهد. در صورت وجود وظایف فنی– مانند ارتقای[11] پایگاه داده به نسخه جدید–، آنها نیز به لیست وظایف افزوده می‌شوند.

پس از آن، برنامه‌نویسان برای پذیرش انجام وظایف، اعلام آمادگی می‌کنند. وقتی گفت‌وگو درباره وظایف به پایان می‌رسد، برنامه‌نویس مسئول، زمان انجام وظیفه‌اش را بر مبنای روز ایده‌آل[12] برآورد و اعلام می‌کند[روز ایده‌آل یکی از واحدهای براورد اندازه(سایز) داستان است. روز ایده‌ال مدت زمان انجام یک کار است به شرطی که مجری فقط همان یک کار را انجام دهد، وقفه‌ای در انجام کار به وجود نیاید و منابع لازم برای کار نیز فوراً آماده گردد .]

در پایان ممکن است برنامه‌نویسی وظایف بیشتر یا کمتر از ظرفیت خود داشته باشد. در این صورت کسی که وظایف کمتری دارد، وظایف بیشتری برمی‌دارد تا تعادل برقرار گردد.

برنامه نویسان وظایف خود را طی تکرار انجام می‌دهند. با پایان یافتن هر وظیفه، برنامه‌نویس کد نوشته شده را با کد سیستم یکپارچه کرده و آزمون‌ها را اجرا می‌کند. همه آزمونها باید اجرا و پشت سر گذاشته شوند، در غیر این صورت کدها اجازه یکپارچه‌شدن با سیستم را نخواهند داشت.

در طول تکرار، آزمونهای کارکردی تحویل شده توسط مشتری به مجموعه موجود افزوده می‌شوند. همه آزمونهای واحد و آزمونهای کارکردی در پایان تکرار اجرا می‌شوند.

۴-۵-وظیفه[13]

برای پیاده‌سازی هر وظیفه، برنامه‌نویسِ مسئول ابتدا باید یک همکار پیدا کند، زیرا همه کدهای برنامه به صورت دو نفره پشت یک کامپیوتر نوشته می‌شود. اگر پرسشی در مورد محدوده یا روش پیاده‌سازی به وجود آید، دو همکار(برنامه‌نویس و همکار وی) جلسه کوتاهی (۱۵ دقیقه‌ای) با مشتری، برنامه‌نویسان مرتبط یا هر دو برگزار می‌کند. برنامه‌نویسان مرتبط کسانی هستند که دانش بیشتری در مورد کدی دارند که در طول پیاده‌سازی وظیفه تغییر خواهد کرد.

با توجه به جلسات، دو همکار فهرستی از موارد آزمون را تهیه می‌کنند که باید قبل از پایان وظیفه اجرا شوند. دو همکار یک مورد آزمون را از فهرست انتخاب می‌کنند که اطمینان دارند قادر به پیاده‌سازی آن بوده و مطالبی در مورد وظیفه به آنها یاد

[9] Functional tests

[10] Tasks

[11] Upgrade

[12] Ideal day

[13] Task

خواهد داد. آنها کد مورد آزمون را می‌نویسند. اگر مورد آزمون اجرا شد، این روند تا زمانی که کار به پایان رسد، ادامه می‌یابد. گرچه معمولاً کارهایی باید انجام شود تا مورد آزمون به درستی اجرا شود.

وقتی مورد آزمونی وجود دارد که اجرا نمی‌شود:

- یا یک راه تمیز برای اجرای مورد آزمون پیدا شده، که در این صورت همان راه پیش برده می‌شود؛
- یا یک راه ناپسند برای اجرای مورد آزمون پیدا شده، اما راه تمیزی هم وجود دارد که نیاز به تغییر طراحی فعلی دارد. در این حالت، سیستم بازسازی[14] می‌شود تا راه تمیز قابل اجرا شود؛
- یا یک راه ناپسند برای اجرای مورد آزمون پیدا شده و راه تمیزی حتی با بازسازی سیستم نیز متصور نیست. در این حالت، همان راه ناپسند پیش برده می‌شود تا مورد آزمون اجرا شود.

پس از اجرای مورد آزمون، اگر راهی برای بازسازی سیستم و تمیزتر کردن آن به ذهنمان رسید، انجام می‌دهیم. در هنگام پیاده‌سازی مورد آزمون، ممکن است مورد آزمون دیگری که بهتر است اجرا شود به ذهنمان برسد. در این حالت، مورد آزمون جدید در فهرست یادداشت شده و کار ادامه می‌یابد. هم‌چنین ممکن است لازم شود بخشی از سیستم بازسازی شود که در محدوده مورد آزمون جاری نمی‌گنجد. این مورد هم یادداشت شده و کار ادامه می‌یابد .

هدف این روش، حفظ تمرکز تیم در کار است. با این شیوه، می‌توان کار فعلی را به خوبی انجام داد و در عین حال بینش و درک کلانی از سیستم داشت. این بینش از تعامل زیاد با کد ناشی می‌شود.

۴-۶-آزمایش

اگر بتوان از تکنیکی به عنوان قلب XP نام برد، بی‌شک این تکنیک، آزمون واحد خواهد بود. همان طور که دیدید، آزمون واحد بخشی از کار روزانه هر برنامه‌نویس است. لازم به یادآوری است که درXP ، ترکیب دو استراتژی معمولی و مرسوم

[14] Refactor

آزمون، موجب افزایش فوق‌العاده کارآمدی آزمون شده است: اول آن که برنامه‌نویسان، آزمون کارشان را خودشان می‌نویسند و دیگر این که آزمون را قبل از کد برنامه می‌نویسند. اگر برنامه‌نویسی به یادگیری ارتباط دارد و یادگیری به دریافت بازخوردهای فراوان در زودترین زمان ممکن، پس می‌توان از آزمونهایی که شخص دیگری روزها یا هفته‌ها بعد از کدنویسی نوشته، نکات فراوانی یاد گرفت [چون آزمونها منجر به بازخورد و بازخورد منجر به یادگیری می‌شود][نتیجه‌گیری: بهتر است آزمونها زودتر نوشته شوند]. از طرف دیگر، اصولاً XP این نکته منطقی را که برنامه‌نویسان معمولاً قادر به آزمایش کد خود نیستند، با اجباری کردن برنامه‌نویسی دونفره پذیرفته است.

برخی از متدولوژی‌ها مانند Cleanroom ، برنامه‌نویسان را از آزمایش و بعضی مواقع حتی از کامپایل برنامه خود منع می‌کنند. فرایند معمولی بدین گونه است که برنامه‌نویس کدی را می‌نویسد، آن را کامپایل می‌کند، مطمئن می‌شود که درست کار می‌کند و سپس آن را به واحد سازمانی مسئول آزمون می‌فرستد. سپس آزمایش میزکار[۱۵،۱۶] در یک مرحله روی همه کد انجام می‌شود. در این آزمایش، متغیرهای کد بررسی می‌گردند، خروجی دستورات چاپی تفسیر و کنترل می‌شوند و چندین دکمه فشار داده می‌شود تا چک لیست آزمون تکمیل و تأیید گردد .

استراتژی آزمون XP نیاز به کار بیشتری نسبت به استراتژی‌های آزمایش میزکار ندارد. در واقع XP فقط شکل انجام آزمون‌ها را تغییر داده است. به جای انجام فعالیت‌هایی که نتایج آنها در پایان مانند اتر محو می‌شوند، آزمون‌ها به شکل دائمی ذخیره می‌شوند. این آزمون‌ها به صورت خودکار امروز اجرا می‌شوند و بعد از ظهر، فردا، هفته آینده و سال آینده بعد از بکپارچه‌سازی نیز اجرا خواهند شد .اطمینان ناشی از اجرای آزمونها به تدریج و با زیاد شدن آنها افزایش می‌یابد و از این رو تیم XP به مرور زمان، به سیستم اطمینان پیدا می‌کند.

همانطور که قبلا اشاره شد، آزمون‌ها توسط مشتریان نیز تعیین می‌شوند. مشتریان در ابتدای هر تکرار اعلام می‌کنند که در چه صورتی خواهند پذیرفت که داستان‌های کاربر به درستی پیاده‌سازی شده‌اند. نظرات مشتریان به آزمون سطح سیستم[۱۷] تبدیل می‌شود. تبدیل می‌تواند مستقیماً توسط خود مشتری با استفاده از زبانهای اسکریپتی متنی یا گرافیکی یا توسط برنامه‌نویسان با استفاده از ابزارهای آزمون انجام شود. این گونه آزمون‌ها نیز موجب افزایش اطمینان به سیستم می‌شوند با این تفاوت که اطمینان مشتری را نسبت به عملیات درست سیستم افزایش می‌دهند[آزمونهای واحد اطمینان برنامه‌نویسان را افزایش می‌دهند].

---

[۱۵] آزمایش میزکار: آزمایشی که روی ماشین، قطعه یا نرم‌افزار، قبل از تحویل آن برای استفاده با هدف کسب اطمینان از درستی آن انجام می‌شود.

[16] Bench test

[17] System-wide test

۵- اشتباهات

وقتی متدی به خوبی کار می‌کند، صحبت کردن در مورد دلیل خوب بودنش مثل این است که بخواهید دلایل فرورفتن پیچ فولادی در آب را با دقت و جزئیات توضیح دهید. آن چه جذابیت دارد، شرح دقیق عملکرد ما در هنگام روبرو شدن با اتفاقات و پدیده‌های غیرمنتظره و نامطلوب است. در اینجا به چند اشتباه رایج و واکنش XP درباره آنها اشاره می‌کنیم.

۵-۱- دست‌کم گرفتن کارها (برآورد کم)

گاهگاهی کارهایی را توافق و تعهد می‌کنیم که بیش از توان ماست. اگر این امر ناشی از برآورد نادرست است(برآورد کم زمان انجام)، باید دفعات وقوع این عارضه را با به‌کارگیری روشهای متعدد برآورد تا حد امکان کاهش دهید . اگر همیشه میزان تعهد بیش از توان شماست، ابتدا سعی کنید مسأله را در داخل تیم حل کنید. آیا در به‌کارگیری تجربه‌ها دچار اشتباه و انحراف شده‌اید؟ آیا آزمون، برنامه‌نویسی دونفره، بازسازی کد و یکپارچه‌سازی را به درستی انجام می‌دهید؟ آیا محصولی که به مشتری تحویل می‌دهید، بیش از نیاز فعلی وی است؟
اگر هیچ راهی برای افزایش سرعت تیم پیدا نکردید، مجبورید از مشتری درخواست کمک کنید. تداوم و اصرار بر متعهد ماندن به کاری که واقعاً بیش از حد توان شماست باعث ناامیدی، کاهش کیفیت و فرسودگی شغلی(کاهش اثربخشی)می‌شود. این کار را نکنید .

بر اساس اطلاعات و یافته‌های جدید پروژه، دوباره برآورد کنید، سپس از مشتری بخواهید که دوباره زمان‌بندی کند. مثلاً بگویید با اطلاعات جدید فهمیدیم که فقط می‌توانیم دو داستان از سه داستان را پیاده‌سازی کنیم؛ با این اوصاف کدام دو داستان، پیاده‌سازی و کدام داستان باید به تکرار یا انتشار بعدی منتقل شود؟ آیا داستانی وجود دارد که شامل یک بخش مهم و یک بخش کم‌اهمیت‌تر باشد؟در این صورت می‌توانیم آن را به دو داستان تفکیک کنیم و بخش مهم را الان و بخش کم‌اهمیت را بعداً تحویل دهیم؟

۵-۲-همکاری نکردن مشتریان

با مشتری‌ای که در بازی شرکت نمی‌کند – مشارکت نمی‌کند- چه می‌کنید؟ هیچ آزمونی را مشخص نمی‌کند، در مورد الویت‌ها تصمیم نمی‌گیرد، داستان‌ها را نمی‌نویسد .

ابتدا با کامل کردن تدیجی نرم‌افزار طی چندین تکرار و فراهم‌سازی امکانی برای کنترل بر توسعه توسط مشتری، سعی کنید رابطه‌ای مبتنی بر اعتماد با وی برقرار کنید. اگر اعتماد شروع به کم شدن کرد و اشکال از شما بود، راه حلی برای آن پیدا کنید. ببینید می‌توانید کاری برای بهبود ارتباط انجام دهید یا خیر؟

اگر به تنهایی نتوانستید مشکل را حل کنید، باید از مشتری درخواست کمک کنید. برنامه‌نویسان XP به سادگی و بر اساس حدس و گمان خود، کار را پیش نمی‌برند. نتایج به دست آمده را به مشتری توضیح دهید. اگر تغییری در آنها به وجود نیامد، نگرانی‌های خود را آشکارا بیان کنید. اگر کسی به نگرانی‌های شما و حل مشکل توجهی نکرد، احتمالاً ادامه پروژه خیلی برایشان اهمیت ندارد.

Ward Cunningham

۵-۳- جابه‌جایی اعضای تیم

اگر کسی تیم را ترک کند، چه اتفاقی خواهد افتاد؟ بدون مستندات و اسناد بازنگری گیر نخواهیم کرد؟ جابه‌جایی در حد متعارف هم برای تیم و هم برای اعضا مفید است. در هر صورت دوست داریم که اعضا، تیم را به دلایل مثبت و خوشایند ترک کنند. اگر برنامه‌نویسان در پایان هر هفته ببینند و لمس کنند که کارهایشان چه نتایج خوبی برای مشتری به ارمغان آورده، شاید کمتر ناامید شوند و به فکر ترک تیم بیفتند.

وقتی کسی تیم XP را ترک می‌کند این‌طور نیست که هر چه را که فقط وی می‌دانسته با خود برده باشد. چرا که هر خط از کد سیستم توسط دو نفر نوشته شده است. از طرف دیگر، هر اطلاعاتی که از تیم خارج شده باشد، نمی‌تواند تیم را خیلی آزار دهد، زیرا که با هر تغییری،می‌توان آزمونها را اجرا کرد تا از خراب نشدن سیستم موجود در اثر بی‌اطلاعی از بخشی از آن مطمئن شد.

اعضای تازه وارد به تیمXP ، در چند تکرار اول حضور خود، با اعضای باتجربه تیم به صورت دونفره کار می‌کنند، آزمون‌ها را می‌خوانند و با مشتری صحبت می‌کنند .وقتی احساس آمادگی کردند، مسئولیت انجام کارهارا قبول می‌کنند. طی تکرارهای بعدی، سرعت کار فردی آنها بالا خواهد رفت تا نشان دهند که میتوانند کارها را سروقت تحویل دهند. بعد از چند ماه، نمی‌توان بین آنها و نیروهای قدیمی تمایز قائل شد.

برنامه‌نویسانی هم که با تیم همکاری نمی‌کنند، مشکل‌آفرین هستند .XP فعالیتی به شدت اجتماعی است و هر کسی نمی‌تواند آن را یاد. اجرای XP نیازمند کنارگذاشتن عادت‌های گذشته است که کار دشواری است به خصوص برای برنامه‌نویسان با تجربه. در آخر با توجه به وجود روشهای مختلف دریافت بازخورد در XP مشخص می‌شود که چه کسی کار می‌کند و چه کسی کار نمی‌کند. کسی که پی‌درپی کارهایش را به پایان نمی‌رساند، یکپارچگی کدهایش باعث بروز مشکلات برای دیگر اعضا می‌گردد، کدهایش را بازسازی نمی‌کند، دونفره کار نمی‌کند، آزمون انجام نمی‌دهد و .... همه اعضای تیم از این اتفاقات مطلع هستند. معلوم است که چنین کسی بهتر است از تیم کنار گذاشته شود هر چند بسیار توانا و ماهر باشد.

۵-۴- تغییر نیازمندها

غول [بزرگ‌ترین و ترسناک‌ترین] مشکلات بسیاری از روشهای توسعه نرم‌افزار، فقط در حد یک مشکل ساده در XP است. با به کارگیری دستورالعمل «انجام طراحی فقط برای مسأله‌های امروز»، سیستمی که بر اساس XP در حال ساخت است، روز بعد برای پیمودن هر مسیری آمادگی دارد. انجام کارهای مشابه با کارهای قبلی، به دلیل ماهیت تکنیک بازسازی[۱۸] در برآورده کردن اصل «یک بار و فقط یک بار»، ساده تر خواهد بود. لازم به یادآوری است که کارهای مشابه در یک پروژه زیاد است. با این حال، با اعلام یک نیازمندی اساسی و متفاوت-نامشابه-، برای انجام آن مجبور نیستید پای‌بند بسیاری از مکانیزمهای قبلی باشید.

در ابتدا درکی از میزان توانایی XP برای مواجهه با تغییر نیازمندی‌ها نداشتم. در اولین نسخهXP ، تعیین این که هر داستان در کدام تکرار انجام شود، بخشی از برنامه‌ریزی انتشار[۱۹] بود. تیم به مرور به این نتیجه رسید که می‌توان با کاهش زمان برنامه‌ریزی به نتایج بهتری دست یافت؛ کافی است از مشتری بخواهید فقط داستانهای تکرار جاری را انتخاب کند. در این روش، با شناسایی هر داستان جدید، نیازی نیست ترتیب داستانهای موجود در تکرارهای باقی‌مانده را بهم ریخته و دوباره مرتب کنید تا تکرار انجام داستان جدید مشخص شود. تنها کاری که باید انجام دهید، قراردادن داستان جدید در بین داستانهای انجام‌نشده است. یک یا دو هفته بعد، اگر داستان جدید هنوز هم برای مشتری اهمیت داشته باشد، وی آن را برای انجام در تکرار پیش‌رو انتخاب خواهد کرد.

(مترجم :
فرض کنید که در سه تکرار پیش رو، قرار است داستانهای زیر انجام شود(اندازه هر داستان جلوی آن مشخص شده است). سرعت تیم برای هر تکرار را ۷ فرض کنید .

تکرار ۱:	داستان A = ۳	داستانB = ۴	
تکرار ۲:	داستان C = ۵	داستانD = ۲	
تکرار ۳:	داستان E = ۲	داستانF = ۳	داستان G = ۲

[18] Refactoring

[19] Release Planning

حال اگر داستان Z با اندازه ۲ به تازگی شناسایی شود و مشخص گردد باید که بعد از داستان A انجام شود، ترتیب انجام داستان‌ها و تکرارهای متناظر آنها دچار تغییر می‌شود که در زیر نمایش داده شده است. این تغییر فقط یک جابه‌جایی ساده نیست، بلکه واقعاً به‌هم ریختن داستان‌ها و مرتب‌سازی دوباره است(به ترتیب حروف الفبای انگلیسی در بالا و پایین دقت کنید)

تکرار ۱:	داستان A = ۳	داستانZ = ۲	داستانD = ۲
تکرار ۲:	داستان B = ۴	داستانF = ۳	
تکرار ۳:	داستان C = ۵	داستانE = ۲	
تکرار ۴:	داستانG = ۲		

این روش برنامه‌ریزی که در آن هر بار فقط تکرار بعدی برنامه‌ریزی می‌شود، موجب خودمانایی[۲۰] مطلوبی می‌گردد. بدین شکل که در بازه ماهانه و سالانه، با **دو** دسته داستان روبرو هستید: داستانهای انتشار جاری و داستانهای باقی‌مانده برای انتشارهای بعدی. در بازه هفتگی و ماهیانه نیز با **دو** دسته داستان روبرو هستید: داستانهای تکرار جاری و داستان‌های باقی‌مانده از انتشار جاری. در بازه روزانه و هفتگی هم با **دو** دسته کار(وظیفه) سروکار دارید: کارهای در دست انجام و کارهای باقی‌مانده از تکرار جاری. همچنین در بازه دقیقه و روزانه نیز با **دو** دسته مورد آزمون روبرو هستید: موردهای آزمون در دست انجام و موردهای آزمون باقی‌مانده.

Ron Jeffries

۶- سخن پایانی

XP به هیچ‌وجه ایده‌ای کامل، بی‌عیب و پایان‌یافته‌ای نیست. محدوده و گستره کاربرد آن شفاف و مشخص نیست. در شرایط فعلی، پذیرش و استفاده از آن نیازمند شجاعت و انعطاف‌پذیری است وگرنه تمایل به بی‌توجهی و رهاکردن پروژه انتخاب شده، موجب شکست XP خواهد شد .

استراتژی من در استفاده از XP این است که ابتدا آن را در جایی که شرایط مناسب دارد، اجرا کنم: پروژه‌های برون‌سپاری یا

[20] self-similarity

داخل سازمانی مربوط به سیستم‌های کوچک و متوسط که نیازمندی‌های آن نامشخص و احتمالاً متغیر هستند. با شروع اجرای XP، می‌توانیم تلاش برای کاهش هزینه تغییرات در محیط‌های پرتنش و چالش را نیز شروع کنیم. اگر قصد دارید از XP استفاده کنید، به خاطر خدا سعی نکنید آن را یک مرتبه قورت دهید. ابتدا یک مشکل حاد را در فرایند جاری انتخاب کنید و سعی کنید آن را با XP حل کنید. وقتی که مشکل حل شد، این کار را دوباره تکرار کنید. در هر لحظه، اگر پی‌بردید که اقدامات قبلی دیگر مفید نیستند، انجام آنها را متوقف کنید[کاری که قبلاً مشکلی را حل می‌کرده، ولی در حال حاضر کمکی نمی‌کند]

این روند به‌کارگیری XP، به شما امکان می‌دهد تا سبک[۲۱] توسعه مختص خودتان را ایجاد کنید-کاری که نه تنها در استفاده از XP، بلکه همواره باید در پی انجام آن باشید. این شیوه به‌کارگیری به شما کمک می‌کند تا ریسکهای ناشی از نامناسب بودن XP برای تیم خود را مدیریت کنید و در کنار تغییر فرایند، تحویل محصول نیز دچار مشکل بیشتری نشود .

[21] Style