**هنر توسعه agile**

در این فصل شما با اصولو شیوه هایی که توسعه agile را تشکیل میدهند آشنا خواهید شد.شما خواهید آموخت که توسعه agile بیشتر از آنکه یک تییر فرهنگی و فلسفی باشد، مجموعه ای از شیآغوه ها و پروسه هاست...

در این فصل همچنین درک کاملی ازagile manifesto خواهید یافت ،سندی که جنبش agile را آغازکرد.شما همچنین نکات کلیدی agile که به وسیله اصول و ارزش هایی گه توسط agile manifesto آشکار شده است را خواهید آموخت.اما اینکه همه اینها چه معنی دارند را در قسمت های بعد به طور کامل متوجه خواهید شد....

هدف این کتاب آماده سازی خوانندگان با دانشی پایه ای برای قدم گذاشتن در راه و تبدیل شدن به یک agile master بسیار فراتر از دوره های این کتاب می باشد.

**-چرابه agileنیاز داریم؟**

نیاز برای متدولوژی توسعه agile از کجا ایجاد شد؟ و چه چیزی درمورد پیشینیان agile بد بود؟ اینها سوالاتی هستند که دربخش های بعدی به آهنها رسیدگی خواهیم کرد.

**-این چیزی بود که من میخواستم نه چیزی که نیاز داشتم!**

پیش از این زمانی که یک تیم یک نرم افزار را توسعه میدادند آنها از توسعه plan-drive استفاده میکردند.این نوع از توسعه با gated-stage مشخص می شود که در تمام نیاز های مشتری برای پروژه جمع اوری شده و سپس یک تحلیل برای مشکل انجام می شود. در مرحله بعد کل برنامه طراحی می شود بدون اینکه حتی اولین خط کد نوشته شود.یکی از گسترده ترین متدولوژی های پذیرفته شده که به توسعه plan-driven پیوسته رویکرد آبشاری برای توسعه نرم افزار است.رویکرد آبشاری از g-s باری جمع اوری نیاز ها استفاده میکند ...

شکل 1-1

متد plan-driven برای صنعت هایی مانند صنعت ساخت و ساز که در آن نیازمندی ها در طول پروژه ثابت می مانند مناسب است.اما اینکار دارای اشکالاتی است، نیاز مندی ها زمانی که به صنعت وارد می شوند در طول حیات پروژه می توانند تغییر پیدا کنند...

در دنیای واقعی، تغییرات پروژه های نرم افزاری ، همه نیازمندی ها نمی توانند مشخص شوند، بعضی چیز ها فراموش می شوندو کسب و کار همواره در حال یادگیری راههای بهتری برای انجام کارهاست.ما می خواهیم نرم افزار بیشتر از نیازمندی های پروژه عمر داشته باشد نه اینکه نیازمندی های پروژه عمر بیشتری نسبت به نرم افزار داشته باشد. توسعه plan-driven به عدم تغییر در نیازمندی ها بستگی دارد.به بیان دیگر زمانی که نیازمندی ها جمع آوری شدند و بر روی آنها توافق انجام شد، نباید تغییر کنند.در صورتی که نیاز به تغییر دز نیازمندی ها باشدهزینه زیادی را برای تیم توسعه و همچنین برای مشتریبه بار خواهد آورد.این تصور که کسب و کار برای مدت نه الی سی وشش ماه، ثابت باقی خواهد ماند تصوری واهی و پوچ است. ذینفعان پروژه و کسب و کار به طور مداوم به دنبال بهبود پروسه و نوآوری در پروژه هستند و آنها نمی توانند این سیر تکاملی را به خطر بیاندازند چرا که آنها منتظر هستند ابزار نرم افزاری کامل شود. در طول چرخه حیات پروژه های تحت متد plan-driven ، کسب و کار به این نکته دست یافت که دریافت بازخورد برای نیازمندی ها و مستند سازی این پروژه ها کار دشواری است.از آنجایی که نیازمندی ها gated-stage هستند بسیاری از پروژه های تحت plan-driven بدون اینکه ذینفعان بدانند که چه چیزی باید تحویل داده شود پیش می رفت.بیشتر اوقات هم ذینفعان در مورد اینکه چه چیزی میخواستند مردد بودند.400 صفحه مستند برای نیازمندی ها راه ایده آلی برای مشخص کردن اینکه سیستم جدید چه کاری انجام میدهد نیست....

در این نوع پروسه های gated-stage تیم توسعه برای نمایش کارهای انجام شده و همچنین ذینفعان برای ارائه بازخورد های خود بر روی کار مکانیزم مناسبی وجود ندارد.بنابراین اولین فرصتی که ذینفعان باید بازخورد های خود را برای پروژه پیشنهاد کنند.در طول مرحله تضمین کیفیت (quality assurance) خواهد بود.

این بدین معنی است که ذینفعان برای مشکلات درخواست راه حل میکنند اما پاسخی از طرف تیم توسعه دریافت نمیکنند حتی شاید باری یک سال یا بیشتر! این یک نوع black-box از محیط توسعه است که مشتری ایده های خود را میفرستد اما راه حل مناسبی حتی تا یک سال یا بیشتر دریافت نمیکند.در این وضعیت ذینفعان و کسب و کار باید قبول کنند که نیازمندی ها همتن نیازمندی هایی هستند که در ابتدای پروژه تعریف شده اند، حتی اگر نیازهای کسب و کار در طول پروژه تغییر یابند. یک رویکرد plan-driven می تواند فقط انتظار ارائه نیازمندی هایی که در ایتدای پروژه بر روی آنها توافق شده است را داشنه باشد.

چیزهایی که کسب و کار میدانست توسط همه چیزهایی که حال میداند تحت الشعاع قرار گرفته ، این شاید باعث ایجاد افزونگی در نرم افزار شود یا بدتر منسوخ شود.

یکی از بزرگترین مسائل پروژه های plan-driven عدم بازگشت واقعی سرمایه برای کسب و کار تا زمان پایان پروژه در طول مرحله deployment است.سود محسوسی برای کسب و کار در طول ماه ها طراحی و مستند سازی جزئیات وجود ندارد. کسب و کار نمتواند فقط 400 صفحه مستند سازی نیازمندی ها را بردارد و در فعالیت های روزانه خود استفاده کند.فقط زمانی که پروژه به طور کامل به پایان رسید کسب و کار می تواند انتظار ارائه ارزش و سود مورد نظر را داشته باشد.متد plan-driven برای ناشناخته ها هیچ قید و شرطی ندارد.می تئان که هدف متد plan-driven از بین بردن ناشناخته های پروزه است چون هیچ مکتنیزم مناسبی برای مقابله با ناشناخته ها ندارد. از این رو به این دلیل به gated-stage نیازداریم که: تا زمانی که ناشناخته های پروژه از بین نروند نمیتوان از مرحله ای به مرحله دیگر رفت. چون هیچ قید و شرطی برای تغییر طراحی اولیه و برای تغییر در نیازمندی ها وجود ندارد که به حذف ناشناخته ها از پروژه نیازمندیم. نیاز به حذف ناشناخته ها از پروژه راهی برای تخمین زدن است. با حذف ناشناخته ها و توافق بر روی زمان تخمین زذه شده پروژه، تمام تاخیرهایی که در پروژه وجود دارند در آخر پروژه انباشته می شوند.در توسعه plan-driven مکانیزم صحیحی برای تخمین خطا ها وجود ندارد و ...

این درست است که پروسه توجهی بر روی مهارت های فنی توسعه دهندگان که عملیات را پیش میبرند ندارد.این توسعه دهندگان مسئول اجرای احتمالی پروژه هستند.

کوچکترین خطا در برنامه نویسی می تواند عواقب بزرگی داشته باشد که برای سالهای سال مشخص نشود. اما بهتر است بدانیم که توسعه دهندگان به عنوان مهندس ، تصمیمات بی شماری را برای طراحی های فنی گرفته اند و همچنین با مشکلات زیادی در طول روزکاری دست و پنجه نرم کرده اند. توسعه plan-driven در شرایطی به کار می آید که مدیران تولید و ذینفعان کسب و کار دقیقا می دانند که چه می خواهند و افکارشان تغییر نمیابد، اولویت بندی های مشخصی دارند و از این مطمعن هستند که پروسه کسب و کار تغییر نخواهد کرد ...

اختصاص زمان زیاد و تاکید بر روی طراحی و جمع آوری نیازمندی ها می تواند تهدیدی برای کسب و کار در هر دو حوزه هزینه و فرصت ها باشد.نیاز برای یک رویکرد قابل اطمینان تر و تکرار شونده که ریسک را کمتر میکند واینکه می تواند حداگثر بازگشت سرمایه را داشته باشد ، باعث به وجود آمدن agile شد.

تغییر تکرار شونده

در حقیقت توسعه نرم افزار وسیله ای برای رسیدن به اهداف می باشد.توسعه نرم افزار سازمان ها را قادر میسازد که اتوماتیک و ساده تر کار کنند و پروسه های خود را به منظور رسیدن به اهداف خود بهبود ببخشند. با اتخاذ متدولوژی agile و ایده های آن از توسعه plan-driven قادربه درک و مواجه شدن با کسب و کار امروز خواهید بود و همین طور قادر خواهید بود سود و ارزش بیشتری را برای ذینفعان خود ارائه کنید.بازخورد مکرر و تعامل بین اعضای تیم توسعه، ذینفعان، کارشناسان domain و اسپانسر ها به این معنی است که در پروژه های Agile ارائه ارزش بشیار سریع می باشد.اولویت بندی وظایف این اطمینان رابرای ما حاصل میکند که نیازهای اورژانسی زودتر بر طرف می شوند. چرخه توسعه تکرار شونده ریسک را کمتر میکند و ارائه منظم کار نرم افزار باعث تحویل راحت تر محصول نهایی می شود، همین طور رضایت کاربر را به دنبال دارد ، باعث کمتر کردن آموزش و کاهش هزینه ها می شود. چون نظم توسعه نرم افزار به تکامل رسیده متدولوژی Agile یک تکامل بزرگ در توسعه نرم افزار نسبت به متدولوژی های قبلی به حساب می آید.

متدولوژی Agile بیشتر از آنکه یک تغییر فلسفی باشد به عنوان یک تغییر روند است. Agile تاکید زیادی بر روی رضایت کاربر و بازگشت سریع سرمایه از طریق رویکرد تکرار شونده دارد. شکل 2-1 پروسه کارس Agile را نمایش میدهد.

به جای طراحی و برنامه ریزی مراحل که تلاش میکردند ناشناخته ها را از پروژه قبل از توسعه آن حذف کنند، Agile بر روی تکرارهای کوچک feature-driven که برای حل مشکلات خاص پروژه تلاش می کند تمرکز دارد.این تکرار ها معمولا جعبه زمانی ثابت که به طور معمول 2 تا 4 هفته طول می کشد را اشغال می کنند. این تکرار ها شامل همه مراحل پروسه plan-driven هستند و کسب وکار را قادر می سازند که به طور مکرر بازخوردهای خود را در زمان بسیار کوتاه ارئه کنند. تفوت بین تکرار و انجام پروژه نرم افزاری با متدولوژی plan-driven در این است که هر تکرار بر روی قسمت های کوچکی از کل پروژه کار میکند. این بخش ها قسمت هایی از پروژه هستند که ذینفعان در اولویت بندی نیازمندی ها آنها را در اولویت های بالاتر قرار داده اند.

توانایی اینکه نرم افزار را در مدت زمان کم به کسب و کار بدهیم.کسب و کار را قادر می شازد که کارش را با نرف افزار شروع کند و به اهدافش از این طریق برسد. از آنجایی که شامل agile با کسب و کار بسیار نزدیک است، کارشناسان domain به اعضای تیم درجه یک رسیدگی میکنند و معمولا با تیم توسعه ملاقات میکنند.

بر خلاف رویکرد آبشاری که دارای شروعی آرام و پایانی سخت است، agile با سرعت قابل تحملی کارش را پیش میبرد. با شکستن تحویل پروژه به قطعات کوچکتر که می تواند در یک تکرار کامل شود agile مکانیزمی برای بهبود دقت تخمین تیم فراهم می کند. این مکانیزم در پروژه های تحت plan-driven وجود ندارد. به طور معمول بعد از 3 یا 4 تکرارتیم یک تخمین نسبتا دقیق دست خواهد یافت. با تخمین های دقیق مدیران پروژه می توانند یک پیش بینی دقیق از زمان مورد نیاز برای تکمیل گل سیستم ارائه دهند.

Agile بسیار شبیه به کسب وکار است که همیشه بر روی بهبود پروسه به وسیله یادگیری و تصحیح این پروژه ها تمرکز می کند. حلقه های بازخورد دائمی از طرف کسب و کار و ذینفعان به ارتقا بخشیدن به این پروسه ها و مهارت ها کمک میکند و تیم توسعه را قادر می سازند که تحویل بسیار موثر و نرم افزار با ارزشی داشته باشند. در پابان با اضافه کردم متدولوژیagile و استفاده از متد توسعه نرم افزار value-driven شما در حال تحویل نرم افزاری خواهید بود که مطابق با نیاز های کسب و کار در بازه های زمانی تکرار شونده تکمیل می شود. حال شما مجبور هستید با یک سری از مشکلات دست و پنجه نرم کنید،در واقع agile برای مقابله با انها طراحی شده است.بیایید نگاه دقیق تری به توسعه دهندگان agile داشته باشیم. ابتدا به agile manifesto و بعضی از نکات کلیدی که داخل آن است توجه کنید.

معرفی agile

در این بخش تعریف روشنی از از پروسه توسعه agile و بعضی از نکات کلیدی آن را خواهیم داشت.

Agile manifesto

در سال 1990 چند نفر در مورد تغییر روش توسعه نرم افزار باهم به گفتگو نشستند.این گفتگو در سال 2001 زمانی که تعدادی از بزرگان توسعه نرم افزار از جمله مارتین فلاور، کنت بک، باب مارتین، کن شاوبر، جس ساترلند و دیو توماس با هم در کوههای Wasatch شهر Utah ملاقات کردند به سرانجام رسید.نتایج این جلسه با نام Agile manifesto شناخته می شود. شکل 3-1

بر اساس manifesto دوازده اصول agile در شکل 4-1 برای برای تشریح اظهارنامه manifesto ساخته شد.

ویژگی های کلیدی agile

نگاهی به manifesto به ویژه اصول 12 گانه agile بیاندازید.ما میتوانیم برخی از نکات و ویژگی های کلیدی که در پروسه تعریف شده اند را معرفی کنیم.

* استقبال از تغییر با فهمیدن نیازهای کسب و کار: Agile بودن، تحقق این مسئله است که تغییر غیر قابل اجتناب است. Agile به وسیله پذیرش تغییر و یادگیری با کسب و کار، به انجام رسید، با این اوصاف agile توانایی انطباق و منعطف بودن را برای ذهن را تعریف کرد تا تغییرات را با آغوش باز بپذیرند به جای اینکه در برابر آن مقاوت کنند یا از اینکه اهداف تغییر کرده شکایت کنند. تیم agile تغییرات را باتمام وجود می پذیرد و فعالیت ها تغییرات در برنامه را تعریف می کنند که این ارزش کسب و کار را بالا می برد.
* تمرکز بر روی ارزش کسب و کار و نرخ بازگشت سرمایه (ROI): توسعه agile توسعه ای است که ذهن را تغییر داد و بر روی تلاش ها و اولویت بندی ها تمرکز کرد. تکنیک های فراوانی در این کتاب تعریف شده است که می تواند برای اینکه یک توسعه دهنده agile باشید شما را یاری کند.ابزار، متدولوژی های پروژه و متدهای برنامه نویسی می تواند راهی برای کمک به agile شدن باشد. اما این قابلیت که بتوان این تکنیک ها را برای کسب و کاری که همواره در حال تغییر است اعمال کرد قابل تقدیر است. اساسا شما باید دامنه کاری که کار می کنید را درک کنید و تلاش ها و شیوه ها و پروسه هایی که به ارزش ها تحقق می بخشند را دسته بندی کنید.
* تحویل مداوم از طریق توسعه تدریجی و تکرار شونده : agile بودن در کل درباره تحویل هرچه زودتر کار نرم افزار است. موفقیت توسعه پروژه در مقدار طراحی شده از پروژه نسبت معیار موفقیت کسب و کار در کار کردن نرم افزار است ، این می تواند معیار شما از پیشرفت نیز باشد.
* بهبود مداوم با یادگیری از طریق کسب و کار : به عنوان بخشی از تیم توسعه ، این کار ماست که زبان و فرایند های کسب و کار را به سیستم نرم افزار تبدیل کنیم.بر این اساس انجام این کار بسیار مهم است که ما با کارشناسان domain خود کار کنیم که همان مردمی هستند که با نرم افزار کار خواهند کرد . کاربران همیشه کارشناس domain نیستند ، آنها تجربه استفاده از پروسه موجود را دارند اما لزوما دلیل آن را نمی دانند. اینجا جایی است که متخصصان domain وارد عمل می شوند. در ادامه شما به عنوان یک توسعه دهنده در مورد کسب و کار بیشتر خواهید فهمید .
* نگه داشتن پروسه ها به صورت lean به وسیله انعکاس (بازگشت) مستمر روی پروسه و حذف ضایعات: Lean نگه داشتن پروسه و تمرینات ، در کل در مورد حذف ضایعات است. با دیدن حجم زیاد مستندات قبل از توسعه سیستم نگران نشوید، مستندات را زمانی تهیه کنید که نیاز است . شما باید قادر باشید از عهده دیاگرام های معماری که هرکدام از اعضای تیم می توانند آن را روی وایت برد بکشند برآیید.به جای حجم زیاد یاز مستندات نیازمندی ها از کارت های وظایف (task card)استفاده کنید وfeature هایی که می توانند به عنوان یادآورنده در گفت و گو عمل کنند را بنویسید. بسیاری از مستندات برای کسب وکار خوب نیستند، به طور ساده ارزشی درآن ها نیست. بخشی از مستندسازی که در agile ساخته شده با عنوان نیازمندی ها است. توسعه agile می خواهد اطلاعات غیر مفید را حذف کند. کدها و داستان های کاربری با معیاری ها مقبولیت وارد، مستندات پروژه می شود نه 400 صفحه نیازمندی های راکد!
* تمرکز قوی بر روی تلاش تیم: Agile در مورد کار تیمی و تمرکز بر روی تیم و تلاش برا ی بهبود تمرینات و تحویل نهایی ارزش بیشتر برای کسب و کار است. کارشناسان domain ، مدیران تولید، تحلیل گران کسب و کار، ذینفعان زیرساخت امنیت و آی تی و تستر ها در طول پروژه باید جز شهروندان درجه یک به همراه توسعه دهندگان باشند. غیر توسعه دهندگان در تیم برای بالا بردن سطح دانش و به استراک گذاستن مالکیت و کاهش شکاف میان توسعه دهندگان و مشتری کمک میکنند.

Flavor of agile

فرم های مختلفی از agile وجود دارد اما همه آنها مشخصات تقریبا یکسانی دارند. در واقع این متدولوژی های مختلف، شاخه هایی از یک آیین هستند. اساس هر شاخه ایده ای برای رضایت مندی مشتری است. نکته کلیدی که در مورد همه این شاخه ها صادق است این است که همه آنها از نوع تکرار شونده هستند.

+ Scrum

متدولوژی Scrum از یک سری Sprint ها که معمولا به انداره 2 تا 4 هفته طول میکشد، تشکیل شده است. هر بخش کاری ارسالی به صورت بالقوه یک shippable software است. حجم کاری هر یک از این sprint ها از یک سری backlog تشکیل شده است. Product backlog شامل ویژگی های جدید ، رفع اشکالات، technical dept و هر چیز دیگری که در تحویل نهایی وجود دارد است. صاحب محصول با کمک مشتری Product backlog ها را اولویت بندی میکند و به وسیله جلسات منظم stand-up و sprint retrospective با تیم کار میکنند. جنبه تکرار شونده scrum این است که چرخه تکرار آن بارها و بارها تکرار می شود تا زمانی که پروژه تکمیل شود. در مورد scrum در فصل 2 بیشتر خواهید آموخت.

+ برنامه نویسی افراطی XP

Xp تمرکز بسیاری بر روی تعامل و مشارکت با مشتری دارد که دارای پنج ارزش زیر است:

* سادگی
* ارتباط
* بازخورد
* شجاعت
* احترام

همچنین xp شیوه های زیر را دنبال میکند:

* برنامه ریزی بازی
* Release های کوچک
* آزمون پذیرش برای مشتری
* طراحی ساده
* برنامه نویسی جفت pair
* توسعه test-driven
* فاکنورگیری مجدد
* ادغام مداوم
* مالکیت جمعی کد
* استاندارد های برنامه نویسی
* متافور ها
* سرعت پایدار

در xp داستان های کاربری برای نیازمندی ها از سوی مشتریان تولید می شوند. این داستان ها توسط توسعه دهندگان تخمین زده می شوند و توسط مشتریان اولویت بندی می شوند و در نهایت به یک نرم افزار بر روی یک پایه تکرار شونده توسعه پیدا میکنند. برنامه ریزی مستمر و پشتیبانی از تحویل فرایند xp را نظم بخشیده است. شایان ذکر است که بسیاری از شیوه هایی که در xp هستند با بقیه انواع agile از جمله scrum به اشتراک گذاشته شده اند. در فصل 3 نگاه دقیق تری بر روی xp خواهیم داشت.

Crystal

کریستال از متدولوژی agile تمرکز بیشتری بر انسان ها دارد تا بر روی پروسه ها. کریستال یک مجموعه ساده از اصول است که باعث افزایش کارگروهی با تمرکز بر روی ارتباطات و حذف نویز از مدیریت پروژه می شود. همچنین کریستال سیستم ها را بر روی اولوبت بندی ها و مسیر های بحرانی توسعه نرم افزار متمرکز میکند. همانند scrum و xp ، کریستال نیز بر روی تحویل مکرر تاکید دارد.

متد توسعه سیستم های پوبا DSDM

متد توسعه سیستم های پویا بر پایه قانون 80-20 است. که در آن باید 80 درصد از سود پروژه از 20 درصد از نیازمندی ها به دست آید. برای بهره برداری اولویت بندی شده، بر روی قسمت هایی که برای سیستم مهم هستند کار کنید. این همان اولین 20 درصد از نیازمندی های پروژه است. DSDM به اصطلاح متد MoSCoW نام گرفته است. .....

توسعه feature-driven

این نوع از توسعه با ایجاد یک مدل از دامنه در حال توسعه آغاز می شود. پس از تکمیل این پروسه، پروسه تکرار شونده طراحی feature ها و اجرای آنها شروع می شود. Feature ها یک گروه بندی مفید از قابلیت ها برای مشتری ارائه می کنند.

FDD از یک سری فعالیت های ساده تشکیل شده است:

* توسعه domain object model
* ساختن لیست feature ها
* برنامه ریزی با feature ها
* طراحی با feature ها
* ساخت با feature ها

توسعه نرم افزار lean

توسعه نرم افزار lean از اصول ساخت lean که از سیستم های تولیدی توسط Toyota ساخته شده، مشتق شده است. Lean بر روی سود بیشتر برای مشتری و حذف اضافات از پروژه تمرکز دارد. Lean دارای این اصول می باشد:

* از بین بردن اضافات : انتخاب Feature های ارزشمند برای مشتری
* تقویت یادگیری : یادگیری به وسیبه آزمایش کردن و انجام عملی نه به وسیله مستندات
* تصمیم گیری هر چه دیرتر: عقب انداختن تصمیم گیری ها به منظور جمع اوری اطلاعات هر چه بیشتر و یافتن تغییراتی که باید انجام شوند.
* تحویل هر چه سریع تر: نرم افزار هر چه سریع تر تحویل داده شود بازخورد ها زود تر می رسند و باعث رفتن به release های بعدی و بازگشت هر چه سریع تر سرمایه خواهد شد.
* توانمد سازی تیم : مسئولبت پذیر کردن تیم و ایجاد انگیزه برای تیم با شرکت دادن همه اعضای تیم در پروسه های تصمیم گیری.
* ایجاد یکپارچگی : بهبود منظم کد برای انعطاف پذیر و انطباق پذیری برای تغییر
* دید کلی : مطمعن شدن از گستردگی دانش در کل تیم به منظور تعریف مشکلات در هر سطحی از سیستم

در این کتاب ما از متدولوژی های بسیار پرطرفدار agile از جمله scrum و xp استفاده کردیم تا برای شما اینکه Agile بودن چه معنی دارد را نشان دهیم.

خلاصه فصل

این فصل برای شما دلیل نیاز به Agile و ایده های آن از توسعه value-driven را معرفی میکند. شما ابتدا در مورد عدم موفقیت رویکرد plan-driven برای توسعه نرم افزار مطالبی را مطالعه میکنید. بسیاری از اینها ...

نیازمندی ها تغییر نمی کنند

هیچ مانع و سورپرایزی وجود نخواهد داشت.

کسب و کار می تواند برای مدت چند ماه قبل از اینکه اولین ورژن نرم افزار را ارائه دهد صبر کند.

پروسه جدیدی که می تواند تغییرات را با آغوش باز بپذیرد و واکنش نشان دهد