**فصل ۳ اصول agile ------------------ سارا**

**مقایسه رویکرد های سنتی و agile :** هدف از مقایسه دو رویکرد این نیست که عنوان کنیم که روش های plan-driven و sequential development بد و ناکارآمد است و scrum خوب است بلکه هر دو روش جزو ابزار های توسعه دهندگان حرفه ای هست و هیچ کدام بد نیست تنها می توان گفت بسته به هدف و مسائل مختلف در هر زمان می بایست از روش مناسب استفاده شود. به عبارتی روش های سنتی و scrum برای دو دسته متفاوت از مسائل استفاده می شود.

برای مثال اگر نیازمندی ها مشخص نیست استفاده از روش plan-driven مناسب نیست اما اگر نیازمندی های سیستم دقیق و مشخص و تعیین شده باشد روش مناسبی است.

روش waterfall یکی از روش های plan-driven است اگرچه پروسس مدل های دیگری هم در این دسته جای می گیرند.

در پروسس های plan-driven ایده اصلی این است که هر چه planning بهتر باشد فهم مسئله (underestanding) بهتر و بنابراین اجرا و پیاده سازی (execution) هم بهتر خواهد شد.

The idea here is that the better the planning, the better the understanding, and therefore the better the execution

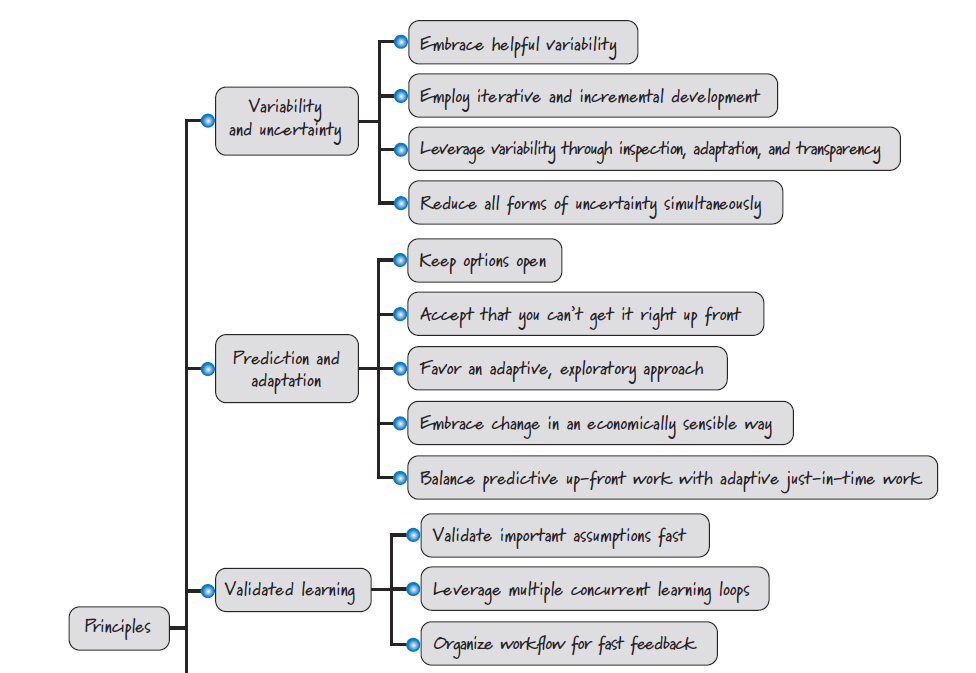
به این پروسس های plan-driven پروسس های sequential هم گفته می شود چون مراحل درون آن به ترتیب است.

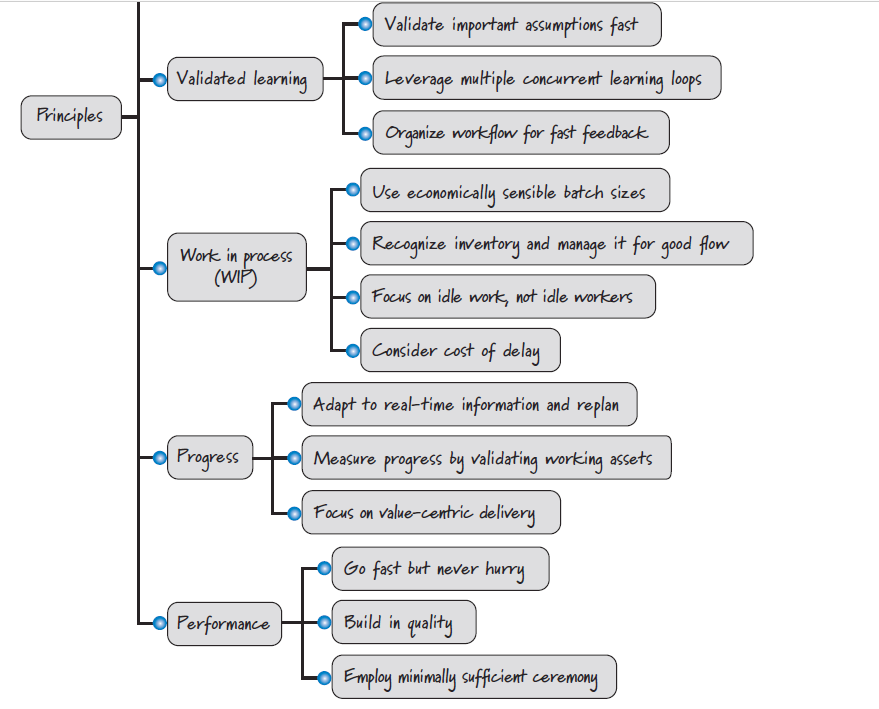
در این پروسس ها ابتدا نیازمندی ها به طور کامل مشخص می شود سپس طراحی (design)، کدزدن (coding) و ساخت سیستم (build) و نهایتا هم تست و ارزیابی (test) به طور کامل انجام میشود.

روش plan-driven در جایی مناسب است که نیازمندی ها و مسئله مشخص و تعیین شده (well defined) و قابل پیش بینی (predictable) باشد و احتمال اینکه تغییرات اساسی اتفاق بیتفد بسیار کم باشد.

به طور خلاصه روش plan-driven یعنی یک پروسس ترتیبی که میگه باید مسئله رو بفهمی، طراحی کنی، کد بزنی، تست کنی و deploy کنی. همه برساس یک plan و برنامه از پیش تعیین شده و مشخص. و نهایتا سیستم آماده شده باید کار کنه.

**اصول agile**

****

****

**Variability and Uncertainty**

اسکرام از تغییرپذیری و عدم قطعیت در تولید محصول استفده می کند تا راه حل های خلاقانه را ایجاد کند.

4 زیر اصلی که مربوط به این اصل هست شامل موارد زیر است:

\_ Embrace helpful variability. (پذیرش و استقبال از تغییرات مفید)

\_ Employ iterative and incremental development. (iterative , incremental به کارگیری یک رویکرد توسعه)

\_ Leverage variability through inspection, adaptation, and transparency.

(تغییرات در قالب بازبینی و تطبیق و شفافیت باشد.)

(کاهش همه انواع عدم قطعیت به طور همزمان) \_ Reduce all forms of uncertainty simultaneously.

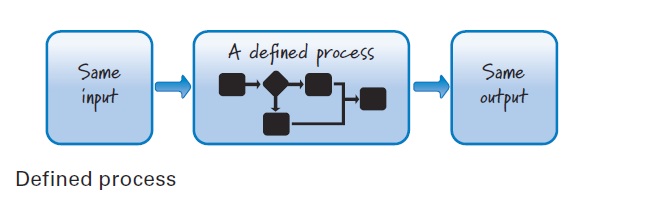
**Embrace Helpful Variability**

روش های plan-driven توسعه نرم افزار را مشابه تولیدات صنعتی تلقی می کنند. این روش ها از تغییر پذیری اجتناب می کنند و تشویق می کنند که از یک پروسه تعیین شده در تولید نرم افزار پیروی شود.

تولید محصول با توسعه نرم افزار متفاوت است. در تولید محصول صنعتی هدف ما این است که یک مجموعه نیازمندی ها و ریکوایرمنت های ثابتی رو برداریم و یک سری مراحل ترتیبی و مشخص و تعریف شده را طی کنیم تا تولید یک سری محصولات مشابه رو به اتمام برسانیم. اما در توسعه محصولات نرم افزاری هدف ایجاد یک نمونه منحصر به فرد از یک محصول است و تولید در حجم عظیم از یک نمونه نداریم که از یک روند خاص پیروی کنیم. این نمونه منحصر به فرد مشابه یک دستورالعمل منحصر به فرد است و ما قصد نداریم یک دستورالعمل را دوبار ایجاد کنیم. اگر چنین کنیم در حقیقت پول خود را هدر داده ایم. در عوض قصد داریم یک دستورالعمل منحصر به فرد برای یک محصول جدید ایجاد کنیم. اگر بخواهیم هر بار یک محصول متفاوت خلق کنیم به کار گیری مقداری تنوع و تغییر ضروری است.

در حقیقت هر ویژگی و فیچری که در یک محصول ایجاد میکنیم با یک ویژگی دیگر در همان محصول متفاوت است بنابراین حتی به تغییر و تنوع در سطح ویژگی های یک محصول هم نیازمندیم.

اگر چه که برخی از مفاهیم تولید، در توسعه محصولات هم کاربرد دارند و می توانند و باید اهرم کار باشند. مثلا شناخت و مدیریت فهرست کار هایی که باید انجام شود در هر دو ضروری است اما به طور کلی با توجه به ماهیت کار، توسعه محصول و تولید صنعتی محصول اصلا یکسان نیستند و هر کدام نیازمند فرآیند ها و پروسه های متفاوتی هستند.



**Employ Iterative and Incremental Development**

Plan-driven ها میگن همه چیز مشخص است و ترتیبی اما در اسکرام iterative, incremental یعنی ابتدا کار ها رو کلی انجام بدیم و بعد ارتقا بدیم و به جزئیات بپردازیم و محصول را گام به گام بهبود دهیم. مثلا یک پروتوتایپ اولیه داریم ک هیچ دکمه ای کار نمی کند بد در هر مرحله بخشی از دکمه هارو فعال میکنیم. یا مثل چاپ کتاب

Incremental یعنی اول هسته اصلی و core سیستم رو ایجاد کنیم و بعد از گرفتن فیدبک ها جزئیات را بهبود دهیم.

اسکرام بر مبنای توسعه iterative و incremental است اگر چه که این دو واژه معمولا به جای هم به کار می روند ولی در واقع توسعهiterative با توسعه incremental متفاوت است.

**Iterative development**

توسعه iterative ( تکراری ) تایید می کند که کار ها ممکن است اول اشتباه انجام شوند و بعدا درست شوند و ممکن است ما یک کار را قبل از اینکه خوب انجام دهیم ضعیف انجام داده باشیم. به این ترتیب توسعه تکراری یک استراتژی برنامه ریزی مجدد است. در حقیقت در این روش توسعه ما از چندین مرحله برای بهبود محصولی که ایجاد کردیم استفاده می کنیم تا نهایتا به سمت یک نمونه خوب از یک محصول همگرا شویم. برای مثال ممکن است ابتدا با ایجاد یک پروتوتایپ

مثلا در پروسه نوشتن یک کتاب یک فصل بار اول نوشته می شود و زمانی که بازخورد ها دریافت می شود و نویسنده متوجه می شه که چطوری میتونه نوشته خودش رو بهبود بده همان فصل چندین بار بازنویسی و بازبینی می شود. توسعه تکراری یک روش عالی برای بهبود محصول است اما بزرگترین نقطه ضعف توسعه تکراری این است که در حضور عدم قطعیت، این دشوار است که تعیین و برنامه ریزی کنیم چند مرحله (پاس) برای بهبود محصول لازم است.

**Incremental development**

توسعه incremental (افزایشی) براساس اصل قدیمی "بخشی از آن را بساز" پیش از اینکه تمام آن را بسازی، بنا شده است. به کمک این روش توسعه از داشتن یک بیگ بنگ در انتهاب کار اجتناب می کنیم ( منظورش اینه یهو یه حجم عظیمی کار رو در اخر سرهم کنیم )

در عوض ما محصول را به قطعات و بخش های کوچکتر تقسیم می کنیم تا بتوانیم برخی از آنها را بسازیم و یاد بگیریم که هر بخش چگونه در اون محیط و جایی که باید کار کند زنده مانده :/ بر اساس آنچیزی که یاد میگیریم سازگار شویم و بعد مقدار بیشتری از آن را بسازیم.

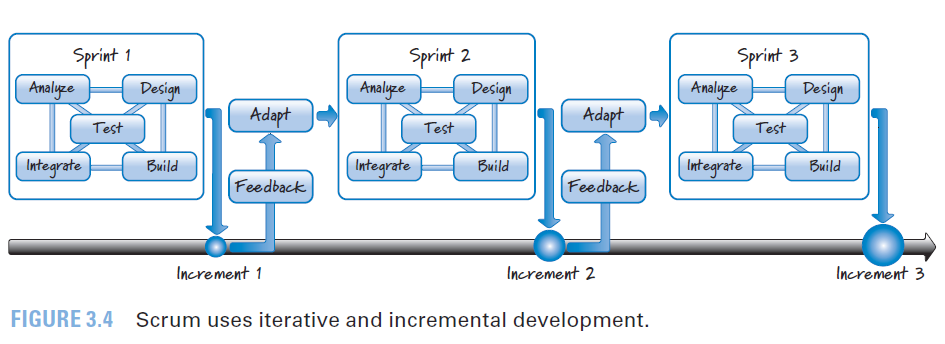
هنگام نوشتن یک کتاب در هر گام یک فصل نوشته می شود و همان فصل زمانی که کامل شد برای بازبینی و دریافت بازخورد ارسال می شود به جای آنکه تلاش کنیم کل کتاب را یکجا بنویسیم و یکجا برای کل آن بازخورد بگیریم. این کار این فرصت را به ما می دهد که از فیدبکی که برای یک فصل گرفته ایم در فصل های بعدی که قرار است بنویسیم استفاده کنیم و لحن و سبک و .. رو بر اساس اون تنظیم کنیم. به علاوه این فرصت را به ما می دهد که به تدریج یاد بگیریم و آنچه که در فصل های اول آموخته ایم را در نوشتن فصل های بعدی به کار بگیریم.

توسعه افزایشی به ما اطلاعات مهمی می دهد که این اطلاعات این امکان رو در اختیار ما قرار میدهد که روند توسعه خودمان را براساس ان تطبیق بدهیم و ادامه روند کار را تغییر بدهیم. بزرگترین عیب توسعه افزایشی این است که با ساخت بخش های کوچک ممکن است تصویر کلی (big picture) رو از دست بدهیم (درخت هارو میبینیم اما جنگل را نه). به عبارتی تصویر کلی از بخش هایی ک کنار هم قرار داده شدن نداریم.

اسکرام در حالی که از مزایای توسعه افزایشی و تکراری بهره می برد، معایب آن را خنثی می کند و این کار را با استفاده از هر دو ایده انجام می دهد در قالب یک مجموعه ی تطبیق پذیری از تکرار بازه های زمانی که اسپرینت نامیده می شوند.

در طول هر اسپرینت ما کلیه فعالیت ها و کار هایی را انجام می دهیم که برای ایجاد **یک بخشی از محصولی که کار می کند** ضروری و لازم است. در هر اسپرینت گام های analysis و design و build و integration و test کامل برای همان اسپرینت طی می شود.

این رویکرد یک مزیتی که دارد این است که فرضیاتی که ضمن توسعه ویژگی های اون بخش پیاده سازی شده اند سریعا ارزیابی و validate می شوند.



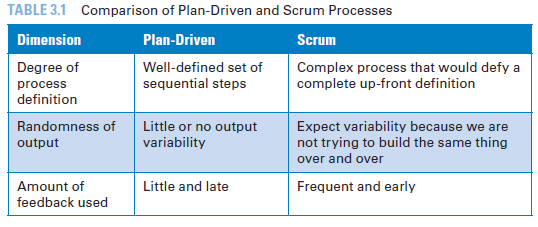
**Leverage Variability through Inspection, Adaptation, Transparency**

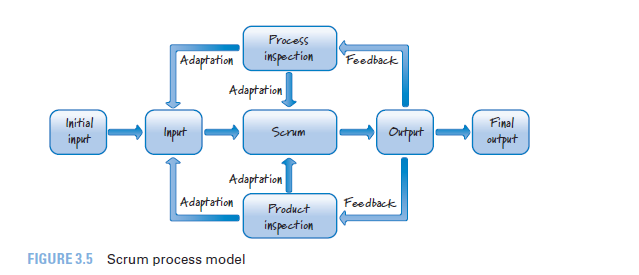
پروسس های plan-driven به طور بنیادی از جنبه های مختلف متفاوت هستند. در یک روند توسع plan-driven و ترتیبی فرض بر این است که تغییراتی نداریم و یا تغییرات ممکن بسیار جزئی است. این روش ها از یک مجموعه مراحل از پیش تعریف شده پیروی میکنند و تنها از اندکی بازخورد استفاده میکنند. به طور عکس اسکرام تایید میکند که بر اینکه در طول توسعه محصول نرم افزاری برخی تغییرات لازم و ضروری است برای اینکه یک محصول جدید ساخته شود.

اسکرام همچنین فرض می کند که فرآیند لازم برای تولید محصول پیچیده است. و بنابراین با یک تعریف کامل از پیش مشخص و تعیین شده مخالفت می کند. بنابراین به طور مداوم و زود بازخورد میگیره برای اینکه مطمئن‌شود که محصول صحیح (انچه که نیازمندی های تعیین شده را برطرف سازد) در حال ایجاد است. و روند توسعه محصول هم صحیح است.

در قلب اسکرام اصول بازبینی، تطبیق و شفافیت قرار گرفته اند. در اسکرام صرفا آنچیزی که ساخته ایم را مطابقت نمیدهیم و بازبینی نمیکنیم بلکه نحوه ساخت و اینکه چگونه محصول را ساخته ایم رو هم مورد بازبینی و تطبیق قرار می دهیم.

برای انجام این کار بر شفافیت تکیه میکنیم. همه اطلاعاتی که برای تولید یک محصول مهم است باید در دسترس افرادی باشد که در توسعه و ایجاد ان محصول درگیر هستند. شفافیت بازبینی را ممکن می سازد که بازبینی هم برای تطبیق محصول ضروری است . شفافیت به علاوه به همه کسانی که در ارتباط با توسعه محصول هستند این اجازه را می دهد که مشاهده کنند و بفهمند چه چیزی اتفاق افتاده. این امر باعث ارتباط بیشتر می شود و اعتماد را هم میان اعضای تیم و هم در فرایند ها برقرار می سازد.





**Reduce All Forms of Uncertainty Simultaneously**

توسعه یک محصول جدید تلاشی پیچیده است و حدی از عدم قطعیت در ان وجود دارد. این عدم قطعیت می تواند به دو دسته کلی تقسیم ‌شود.

**End uncertainty (***what* uncertainty**)**

عدم قطعیت در باره پایان و نتیجه کار و اینکه محصول نهایی چه ویژگی هایی را خواهد داشت. درواقع قاطعانه نمیتوان گفت چه چیزی نهایتا تولید می شود.

**Means uncertainty (***how* uncertainty**)**

عدم قطعیت درباره روش کار و اینکه فرایند ها و تکنولوژی و ابزار هایی ک در روند توسعه به کار گرفته می شوند. درواقع قاطعانه نمیتوان گفت از چه روش ها و تکنولوژی هایی برای توسعه محصول تا پایان کار استفاده خواهد شد‌.

**uncertainty** (*who* uncertainty) **Customer**

در محیط های خاص یا محصولات خاص ممکن است در مورد مشتری و کاربران نهایی سیستم هم عدگ قطعیت وجود داشته باشد و نتوان‌ قاطعانه گفت کاربر نهایی سیستم چه کسی است. برای مثال شرکت های استارت اپی شامل سازمان های بزرگی که بر روی

محصولات جدید تمرکز می کنند ممکن است فقط فرضیاتی داشته باشند در مورد اینکع کاربر نهایی محصول انها چه کسی خواهد بود. این عدم قطعیت باید به سمت مسیر صحیح هدایت بشه در غیر اینصورت ممکن هست محصولات درخشان و عالی اشتباها برای یک بازار نادرست و نامناسب تولید شود.

فرایند های توسعه ترتیبی و سنتی در وهله اول بر روی حذف همه end uncertainty ها تمرکز می کنند به این صورت که به طور دقیق تعریف می کنند که چه چیزی باید ساخته شود و سپس means uncertainty ها رو حذف میکنند. این رویکرد ساده و خطی جهت کاهش عدم قطعیت ها در فرایند پیچیده توسعه محصول مناسب نیست چون در این روند توسعه اقدامات ما و محیطی که در اون کار میکنیم به طور متقابل بر یکدیگر اثر دارد و میتواند یکدیگر رو محدود کند.

برای مثال ما تصمیم میگیریم که یک ویژگی رو در محصول ایجاد کنیم)اقدام ما(

در وهله بعد ما این ویژگی رو به مشتری نشان می دهیم و زمانی که مشتری اون رو میبیته نظرش درباره اینکه واقعا چه چیزی رو میخواد تغییر میکنه یا متوجه میشود که جزئیات ویژگی ها و اونچیزی که میخواسته رو دقیق منتقل نکرده) اقدام ما یک پاسخ از محیط دریافت و استنباط میکند(

و سپس ما طراحی رو براساس بازخوردی که دریافت کردیم تغییر می دهیم) عکس العمل محیط باعث شده ما یک اقدام پیش بینی نشده رو انجام بدیم(

در اسکرام ما با پرداختن به یکی از انواع عدم قطعیت ها پیش از آنکه به نوع دیگری بپردازیم خودمان را محدودی نمی کنیم. (نمیایم مرحله به مرحله عدم قطعیت ها رو حذف کنیم مثلا در روش سنتی اول همه end ها حذف می شد بعد means ها) در عوض رویکرد جامع تری را در پیش می گیریم و بر کاهش همزمان همه عدم قطعیت ها (end, means, customer, …) تمرکز می کنیم.

البته در هر برهه ای از زمان ممکن است است بر روی یکی از جنبه های عدم قطعیت نسبت به سایر جنبه ها بیشتر تمرکز کنیم. پرداختن همزمان به چندین نوع از عدم قطعیت ها با استفاده از رویکرد توسعه iterative , incremental ای که با بازبینی و تطابق و شفافیت هدایت می شود ساده تر است. چنین رویکردی به ما اجازه می دهد به طور فرصت طلبانه محیط رو کاوش کنیم برای آنکه مجهولات ناشناخته ای (unknown unknowns) که در حال ظهور هستند را یادبگیریم و شناسایی کنیم. (منظور از مجهولات ناشناخته چیز هایی هست که ما هنوزم نمیدونیم که اون ها رو نمیدونیم )

**Prediction and Adaptation**

وقتی که از اسکرام استفاده می کنیم به طور مداوم تلاش می کنیم تمایل به پیش بینی کار ها را با نیاز به تطابق بالانس و متعادل کنیم.

5 زیر اصلی که مربوط به این اصل هست شامل موارد زیر است:

\_ Keep options open. ( گزینه هاتو روی میز باز نگه دار )

\_ Accept that you can’t get it right up front.(بپذیریم که نمی توان همه چیز را از قبل و درست دریافت کرد)

\_ -Favor an adaptive, exploratory approach.( رویکرد انطباقی و اکتشافی را ترجیح دهید)

\_ Embrace change in an economically sensible way. (پذیرش تغییر با روشی که از نظر اقتصادی منطقی است)

\_ Balance predictive up-front work with adaptive just-in-time work.

(برقراری تعادل بین کارهایی که جلوتر پیش بینی میکنیم با کار هایی که درست سروقت تصمیم گیری میشوند و تطبیق پذیر هستند.)

**Keep Options Open**

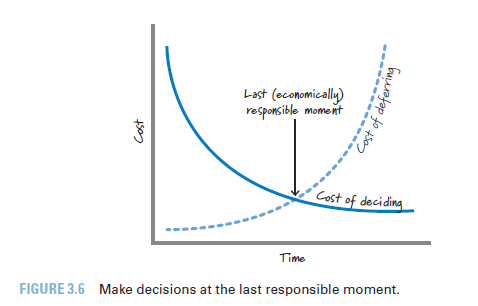
رویکرد های توسعه plan-driven و sequential مستلزم آن است که تصمیمات مهم در زمینه هایی مانند requirements و design در همان فاز مربوطه گرفته و بازبینی و تایید شوند. به علاوه این تصمیمات باید پیش از آنکه به فاز بعدی برویم اتخاذ شوند حتی اگر این تصممیات بر اساس دانش محدود باشد و دانش کاملی در آن فاز نداشته باشیم.

اسکرام عنوان می کند که نباید هرگز تصمیم زودرس بگیریم فقط به این دلیل که یک فرآیند عمومی و جنرال به ما دستور میده که الان زمان تعیین این تصمیم است. در عوض زمانی که از اسکرام استفاده می کنیم از استراتژی ای طرفداری می کنیم که به ما این امکان رو میده که گزینه ها و آپشن هایی که میتونیم از آن ها استفاده کنیم رو باز و open داشته باشد.

اغلب این اصل به last responsible moment (اخرین لحظات مسئولیت) شناخته می شود به این معنی که ما الزام به تصمیم را به لحظات اخر موکول می کنیم و تا آخرین لحظات مسئولیت تصمیمات مهم و برگشت ناپذیر نمی گیریم. و چه زمانی این کار را انجام می دهیم؟ ( چه موقع تصمیم بگیریم؟) زمانی که هزینه تصمیم نگرفتن بیشتر از هزینه تصمیم گرفتن باشد در ان لحظه تصمیم گیری می کنیم.

برای درک این اصل این را در نظر بگیرید: در اولین روز تلاش برای توسعه محصول، ما اطلاعات محدود و اندکی درباره انچه که می خواهیم انجام دهیم داریم. در هر روز از توسعه ما اندکی بیشتر یاد می گیریم پس چرا بیایم و تصمیمات مهم و حتی شاید بازگشت ناپذیر را در ابتدای کار بگیریم؟ اغلب ما ترجیح می دهیم صبر کنیم تا زمانی که اطلاعات بیشتری کسب کنیم و بنابراین بتوانیم یک تصمیم اگاهانه تر بگیریم.

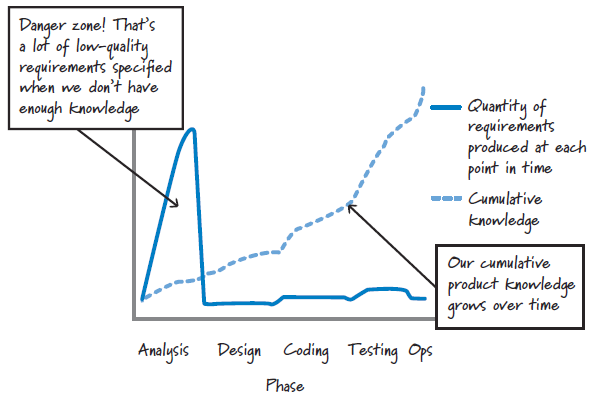
هنگام برخورد با تصمیمات مهم و برگشت ناپذیر اگر خیلی زود تصمیم بگیریم و این تصمیم اشتباه باشد در بخش نمایی هزینه تصمیم قرار خواهیم گرفت. هرچه درک بهتری در مورد تصمیم پیدا کنیم هزینه تصمیم گیری کاهش می یابد (احتمال گرفتن یک تصمیم بد کاهش می یابد به دلیل اینکه اطمینان و قطعیت بیشتری در مورد بازار محصول و کارهای فنی داریم) به همین دلیل است که باید صبر کنیم تا زمانی که اطلاعات بهتری داشته باشیم پیش ازین که در قبال تصمیمات تعهد پیدا کنیم.



**Accept That You Can’t Get It Right Up Front**

فرآیند های plan-driven نه تنها الزام می کند که نیازمندی ها و requirement ها کامل باشد و یک برنامه کامل داشته باشیم بلکه این فرآیند ها فرض می کنند که ما می توانیم این برنامه کامل را از قبل درست کنیم. اما واقعیت این است که بسیار نامحتمل است که بتوانیم همه نیازمندی های یا جزئیات مربوط به برنامه بر اساس آن نیازمندی ها را از پیش تعیین کنیم. آنچه بدتر است این است که زمانی که نیازمندی ها و requirement ها تغییر می کنند می بایست نیازمندی های اساسی و پایه و برنامه ها را مطابق با واقعیت فعلی اصلاح کنیم.

در اسکرام ما این امر را تصدیق می کنیم که نمی توانیم همه ی نیازمندی ها یا برنامه ها را پیش تر بدست آوریم. در حقیقت ما معتقدیم که تلاش برای انجام این کار می تواند خطرناک باشد زیرا با این کار ما احتمالا دانش مهم رو از دست می دهیم و این منجر به ایجاد مقدار زیادی نیازمندی های کیفیت-پایین می شود.



این شکل نشان می دهد که زمانی که از یک رویکرد توسعه ترتیبی و plan-driven استفاده می کنیم یک تعداد زیادی از نیازمندی ها پیش تر و زمانی که در مجموع دانش اندک و حداقلی درباره محصول داریم، تولید شده اند. این رویکرد ریسکی است زیرا یک توهمی وجود دارد که انگار ما end uncertainty (عدم قطعیت مربوط به محصول) ها را حذف کردیم. زمانی که دانش ما بهبود پیدا میکند و چیز ها تغییر می کنند اینکار بالقوه باعث هدر رفت است.

با استفاده از اسکرام ما هنوز هم برخی از نیازمندی ها و برنامه ها را از پیش تعیین و تولید می کنیم. اما فقط به میزان کافی و با فرض آنکه جزئیات مربوط به این نیازمندی ها و برنامه ها را بعدا که بیشتر درباره محصولی که میسازیم یاد گرفتیم، پر میکنیم.

از این گذشته، حتی اگر ما فکر کنیم که 100 درصد مطمئن هستیم که چه چیزی میخواهیم بسازیم و روند توسعه و ساخت آن چگونه است، به زودی خواهیم آموخت که در آن اشتباه کرده ایم ---

در آن نقطه همه ی واقعیت های ناخوشایند آنچه که واقعا مورد نیاز است ما را به سمت ایجاد تغییرات سوق می دهد.

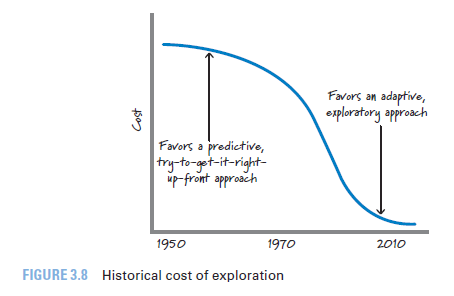
**Favor an Adaptive, Exploratory Approach**

فرآیند های plan-driven و ترتیبی بر روی استفاده یا بهره برداری از آنچه که در حال حاضر شناخته شده است و پیش بینی انچه که ناشناخته است تمرکز می کنند. اسکرام یک رویکرد تطبیق پذیر تر و قابل آزمایش و خطا براساس استفاده مناسب از اکتشاف را ترجیح می دهد. اکتشاف به مواقعی اشاره دارد که ما تصمیم میگیریم با انجام برخی فعالیت ها دانش کسب کنیم مانند ساختن یک پروتوتایپ، اثبات یک مفهوم، انجام یکسری مطالعات یا انجام یک آزمایش. به عبارت دیگر زمانی که با عدم قطعیت روبه رو می شویم اطلاعات را با استفاده از اکتشاف خریداری می کنیم. ابزار عا و تکنولوژی های ما به طور قابل توجهی بر هزینه اکتشاف تاثیر می گذارد. به طور تاریخی اکتشاف توسعه محصول نرم افزاری گران بوده است. یک واقعیتی که رویکرد پیشگو تر را ترجیح می دهد.

به طور مثال

خوشبختانه ابزار ها و تکنولوژی ها بهتر شده اند و هزینه اکتشاف کاهش یافته است و دیگر هیچ مانع اقتصادی برای اکتشاف و جستجو وجود ندارد. در حقیقت امروزه اغلب اینکار ارزان تر است که با بازخورد های کاربر تطبیق پیدا کنیم نسبت به اینکه تلاش کنیم که همه جیز را از پیش درست کنیم و بدانیم. این چیز خیلی خوبی است زیرا زمینه (تکنولوژی های اطراف) که در آن ها راه حل ما موجود است به طور فزاینده ای در حال پیچیده شدن است.

در اسکرام اگر ما دانش کافی برای انجام یک گام منطقی و آگاهانه داشته باشیم که در راستای راه حل ما است انگاه پیشرفت می کنیم. اما زمانی که با عدم قطعیت مواجه می شویم به جای اینکه تلاش کنیم آن را پیش بینی کنیم از یک اکتشاف کم هزینه برای دستیابی به اطلاعات مربوطه استفاده می کنیم که می توانیم یک گام منطقی و آگاهانه در راستای راه حل خود به جلو برداریم. بازخورد از کار و عمل ما به ما کمک می کند که نیاز و زمان نیاز خود به اکتشاف را مشخص کنیم.



**Embrace Change in an Economically Sensible Way**

همانطور که میدانیم در رویکرد های سنتی و ترتیبی هزینه اعمال تغییراتی که دیرتر اتفاق می افتد بسیار گران تر از تغییرات زود هنگام است. به عنوان مثال یک تغییری که در فاز analysis رخ می دهد ممکن از 1 دلار هزینه داشته باشد اما همان تغییر در فاز تست اگر رخ دهد می تواند 1000 دلار هزینه داشته باشد. چرا اینگونه است؟ اگر در فاز تحلیل و analysis یک اشتباه کنیم و آن را در همان فاز analysis متوجه شویم هزینه تعمیر آن گران نیست. اگر همین خطا تا فاز design و طراحی کشف نشود ما نه تنها باید نیازمندی ها و requirement ها نادرست را تصحیح کنیم **بلکه به طور بالقوه بخش هایی از طراحی ما که براساس آن نیازمندی های اشتباه صورت گرفته است نیز باید تصحیح شود.** این ترکیب از خطاها در مرحله های بعدی ادامه می یابد و باعث می شود **خطای کوچکی که در فاز analysis قابل تصحیح بوده به یک خطای بزرگتر برای اطلاح و تصحیح در فاز تست تبدیل شود.**

برای اجتناب از تغییرات دیرهنگام، فرآیند های ترتیبی یک کنترل و کوچک سازی تغییرات در requirement ها و design را با بهبود دقت پیش بینی درباره آنچه که درسیستم نیاز است انجام شود یا اینکه چگونه باید انجام شود

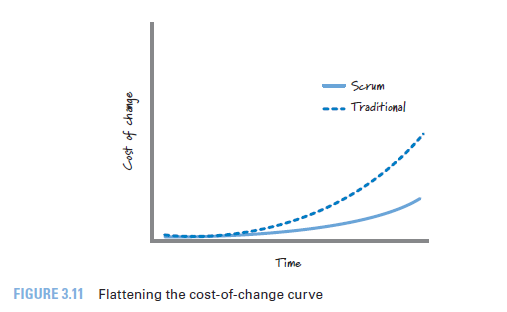
**متاسفانه پیش بینی های بیش از اندازه در فاز فعالیت های زودهنگام (فاز های اولیه) تاثیر عکس دارد. نه تنها تغییرات را حذف نمی کند بلکه در واقع باعث می شود که محصول دیرتر اماده شود و هزینه بیشتری را نیز متحمل شویم.** چرا این تناقض وجود دارد؟ اولا تمایل به حذف تغییرات گران و پر هزینه مارا وادار می کند که در هر فاز بیش از حد سرمایه گزاری کنیم با انجام کارهایی بیش از آنچه که نیاز و مورد کاربرد است. **ثانیا ما مجبور میشویم که براساس فرضیات مهم در ابتدای فرآیند و پیش از آنکه این مفروضات را با دریافت بازخوردهایی از ذینفعان محصول ارزیابی کنیم، تصمیم بگیریم.**

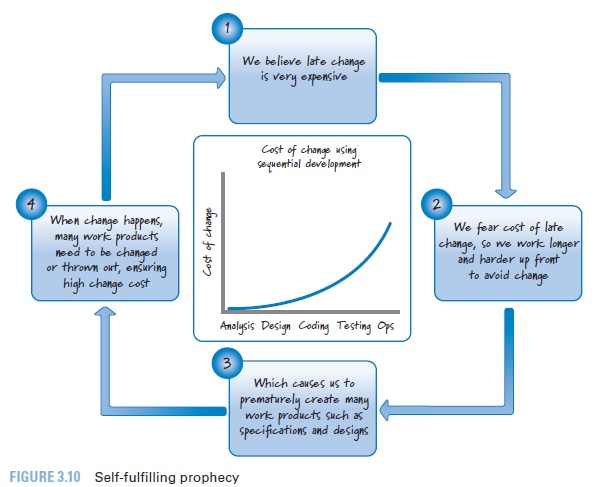
به عنوان یک نتیجه ما براساس این مفروضات یک بخش زیادی از کار را میسازیم و بعدا این بخش از کار ممکن است تصحیح شود یا ضمن اعتبارسنجی فرضیات کنار گذاشته شود (به دلیل معتبر نبودن) یا تغییر رخ دهد (برای مثال requirement هایی ظهور می کنند یا تکمیل می شوند).

در اسکرام ما فرض می کنیم که تغییر یک قاعده و نرم است. ما معتقد هستیم که نمیتوانیم عدم قطعیت ذاتی که در فرآیند توسعه محصول وجود دارد را با کار کردن طولانی تر و سخت تر پیش بینی کنیم. بنابراین ما باید برای پذیرش تغییر آماده شویم و زمانی که این تغییر رخ می دهد ما میخواهیم از نظر اقتصادی بهتر از رویکرد توسعه سنتی باشد حتی زمانی که تغییرات در توسعه محصول دیرتر رخ می دهند.

بنابراین هدف ما این است که منحنی هزینه تغییرات را تا جایی که ممکن است ثابت (مثل یک رابطه خطی) نگه داریم و حتی پذیرش تغییرات دیرهنگام تر هم از نظر اقتصادی معقول باشد. شکل زیر این ایده را نشان می دهد. ما می توانیم با مدیریت میزان کار در فرآیند و جریان کار به این هدف برسیم. درنتیجه هزینه تغییرات زمانی که از اسکرام استفاده می کنیم کمتر تحت تاثیر زمان هستند نسبت به روش های سنتی که در آن ها زمان تغییر تاثیر بسزایی در هزینه تغییر داشت. صرف نظر ار اینکه از چه رویکرد توسعه محصولی استفاده می کنیم میخواهیم این رابطه درست باشد: یک تغییر کوچک در نیازمندی ها باید به نسبت منجر به یک تغییر کوچک در پیاده سازی و بنابراین در هزینه شود. (بدیهی است که انتظار داریم یک تغییر بزرگ هزینه بیشتری داشته باشد.) یکی دیگر از ویژگی های مطلوب این رابطه این است که ما میخواهیم این رابطه برقرار و صحیح باشد صرف نظر از اینکه چه زمانی تغییر درخواست می شود.

با اسکرام ما بسیاری از بخش های کار (مانند requirmenet , design , test cases ) را به موقع (in a just-in-time fashion) تولید می کنیم. و از ایجاد مصنوعات و بخش های غیرضروری اجتناب می کنیم. در نتیجه زمانی که یک تغییر ایجاد میشود به طور معمول تعداد artifavt ها و تصمیمات محدود کننده کمتری براساس فرضیات ممکن است کنار گذاشته شود یا دوباره روی آن کار شود اما هزینه متناسب با اندازه تغییر درخواست شده است. تهش موند





**Balance Predictive Up-Front Work with Adaptive Just-in-Time Work**

یک باور اساسی در رویکرد توسعه plan-driven این است که نیازمندی ها و برنامه های جزئی که جلو جلو تعیین شده ((up-front مهم و حیاتی هستند و باید پیش از انتقال به مرحله بعد کامل شوند.) در اسکرام ما معتقد هستیم که کارهای زودتر از هنگام تعیین شده می توانند مفید باشند اما بدون زیاده روی در آنها.

با اسکرام ما تصدیق می کنیم که دستیابی به requirement ها و plan هایی که پیش از موعد تعیین شده و به صورت درست و دقیق باشند امکان ناپدیر است. این بدان معنی است که ماهیچ برنامه ریزی و نیازمدی ای رو از پیش تعیین نکنیم؟ البته که نه. اسکرام درباره یافتن تعادل بین کارهایی هست که باید پیش بینی شوند و کارهای که به وقتش در مورد آنها تصمیم میگیریم و تطبیق پذیر هستند.

**Validated Learning**

\_ Validate important assumptions fast.

\_ Leverage multiple concurrent learning loops.

\_ Organize workflow for fast feedback.

**Validate Important Assumptions Fast** (ارزیابی سریع فرضیات)

سریع فرضیات مهم را ارزیابی کنید یک فرض یک حدس یا اعتقاد است که درست ، واقعی یا قطعی فرض می شود حتی اگر ما هیچ یادگیری معتبری برای دانستن درست بودن آن نداریم. توسعه برنامه محور (Plan-driven) نسبت به Scrum نسبت به مفروضات طولانی مدت تحمل بیشتری دارد.(فرض هایی که تغییر نمی کنند.) با استفاده از Plan-driven، ما الزامات (requirements) و برنامه های گسترده پیش رو تولید می کنیم که احتمالاً بسیاری از مفروضات مهم را در خود جای داده است ، مفاهیمی که تا مرحله بعدی توسعه معتبر نخواهند بود.

فرضیات نشان دهنده یک خطر قابل توجه توسعه است. در Scrum ، سعی می کنیم تعداد پیش فرض های مهم موجود در هر زمان را به حداقل برسانیم. ما همچنین نمی خواهیم فرضیه های مهم بدون اعتبار سنجی برای مدت طولانی وجود داشته باشد. از ترکیب توسعه تکراری و افزایشی همراه با تمرکز بر اکتشاف کم هزینه می توان برای تأیید و ارزیابی سریع فرضیات استفاده کرد. در نتیجه ، اگر هنگام استفاده از Scrum یک فرض اساساً بد انجام دهیم ، احتمالاً به سرعت اشتباه خود را کشف خواهیم کرد و فرصتی برای بهبودی از آن خواهیم داشت. در توسعه ترتیبی و Plan-driven همان فرض بد اگر با تأخیر تأیید شود ، ممکن است باعث شکست قابل توجه یا کلی در تلاش توسعه شود.

**Leverage Multiple Concurrent Learning Loops** (یادگیری چندین چیز به صورت همزمان)

اهرم یادگیری چندین چیز به صورت همزمان در هنگام استفاده از توسعه ترتیبی اتفاق می افتد. با این حال ، شکل مهمی از یادگیری فقط پس از ساخته شدن ، تلفیق و آزمایش ویژگی ها اتفاق می افتد ، این بدان معنی است که یادگیری قابل توجهی در اواخر تلاش در روند توسعه رخ می دهد. یادگیری دیررس مزایای کمتری را به همراه دارد زیرا ممکن است زمان کافی برای استفاده از یادگیری وجود نداشته باشد و یا هزینه استفاده از آن بسیار زیاد باشد. در اسکرام ، ما می فهمیم که یادگیری مداوم رمز موفقیت ماست. هنگام استفاده از اسکرام ، ما حلقه های بازخورد را برای افزایش یادگیری شناسایی و مورد بهره برداری قرار می دهیم. یک الگوی تکراری در این سبک از توسعه محصول این است که یک فرض داشته باشد (یا هدف تعیین کنید) ، چیزی بسازید (برخی فعالیت ها را انجام دهید) ، در مورد آنچه که ساخته ایم بازخورد بگیرید و سپس از این بازخورد برای بررسی کارهایی که انجام داده ایم نسبت به آنچه که فرض کرده ایم استفاده کنیم. سپس بر اساس آنچه آموخته ایم با محصول ، فرآیند و یا اعتقادات خود سازگاری پیدا می کنیم (شکل 3.13 را ببینید). اسکرام از چندین حلقه یادگیری از پیش تعریف شده استفاده می کند. به عنوان مثال ، اسکرام روزانه یک حلقه روزانه است و بازبینی اسپرینت یک حلقه در تکراری است.

چارچوب اسکرام همچنین به قدری انعطاف پذیر است که بسیاری از حلقه های یادگیری دیگر را نیز در بر می گیرد. به عنوان مثال ، اگرچه توسط Scrum مشخص نشده است ، اما حلقه های بازخورد تمرین فنی ، مانند برنامه نویسی جفتی (pair programming) (بازخورد در چند ثانیه) و توسعه آزمون محور (بازخورد در دقیقه) ، اغلب با توسعه Scrum استفاده می شوند.

**Organize Workflow for Fast Feedback** (سازماندهی بارکاری برای گرفتن سریع بازخورد ها)

تحمل داشتن در برابر فرضیات طولانی مدت ، فرایندهای برنامه محور و Plan-driven را نسبت به یادگیری دیر هنگام مقاوم می کند ، بنابراین بازخورد سریع مورد توجه نیست. با اسکرام ، ما برای بازخورد سریع تلاش می کنیم ، زیرا این امر برای کمک به کوتاه کردن مسیرهای غلط زودرس بسیار حیاتی است و برای کشف سریع و بهره برداری سریع از فرصت های حساس و حساس به زمان بسیار حیاتی است. در یک برنامه توسعه Plan-driven ، هر فعالیتی برنامه ریزی شده است که بر اساس توالی فاز کاملاً مشخص در یک زمان مشخص انجام شود. این روش فرض می کند که فعالیتهای قبلی بدون بازخورد حاصل از فعالیتهای بعدی می توانند به اتمام برسند. در نتیجه ، ممکن است بین انجام کاری و دریافت بازخورد درباره آنچه انجام داده ایم مدت زمان طولانی باشد (از این رو حلقه یادگیری را می بندیم). بیایید از یکپارچه سازی مولفه ها به عنوان مثال استفاده کنیم. (component **integration**) بگویید ما سه جزیا مولفه را به طور موازی در حال توسعه داریم. در برخی از زمان ها ، قبل از اینکه یک محصول قابل حمل و ارائه داشته باشیم ، این قطعات باید یکپارچه و آزمایش شوند. تا زمانی که سعی نکنیم ادغام را انجام دهیم ، واقعاً نمی دانیم که آیا مولفه ها را به درستی توسعه داده ایم. تلاش برای یکپارچه سازی، بازخورد مهمی درباره کار توسعه مولفه ها فراهم می کند. با استفاده از توسعه پی در پی و ترتیبی ، یکپارچه سازی و آزمایش تا مرحله از پیش تعیین شده پایین دست ، که در آن بسیاری یا همه اجزا یکپارچه می شوند ، اتفاق نمی افتد. متأسفانه ، تصور اینکه ما می توانیم یک سری از مولفه ها را به طور موازی و سپس بعداً ، در یک مرحله ادغام ، هموار آنها را در یک کل منسجم جمع کنیم ، بعید به نظر می رسد. در حقیقت ، حتی با رابط های تصور شده خوبی که قبل از توسعه اجزا تعریف شده اند ، هنگام ادغام آنها مشکلی پیش خواهد آمد (شکل 3.14 را ببینید).

فعالیتهای بازخورد که مدتها پس از توسعه اتفاق می افتد ، عوارض جانبی ناخوشایندی دارد ، مانند تبدیل یکپارچه سازی به یک مرحله بزرگ تست و رفع ، زیرا اجزای سازنده به طور جداگانه از یکدیگر غالباً به طور یکپارچه ادغام نمی شوند. چقدر طول می کشد و هزینه رفع آن چقدر است فقط در این مرحله می توان حدس زد. در Scrum ، ما جریان کار را سازمان می دهیم تا از طریق حلقه یادگیری در شکل 3.13 حرکت کنیم و در اسرع وقت به بازخورد برسیم. با این کار ، اطمینان حاصل می کنیم که فعالیت های تولید بازخورد در فاصله زمانی نزدیک به اثر اصلی رخ می دهد. بازخورد سریع مزایای اقتصادی برتر را فراهم می کند ، زیرا هنگام تأخیر در بازخورد ، خطاها به وجود می آیند و منجر به شکست های بزرگتر می شوند. بیایید دوباره به مثال ادغام مولفه های خود نگاهی بیندازیم. وقتی قطعات را طراحی کردیم ، فرض های مهمی در مورد چگونگی ادغام آنها ارائه کردیم. براساس این پیش فرض ها ، ما یک مسیر طراحی را طی کردیم. در این مرحله ، ما نمی دانیم که مسیر طراحی انتخاب شده درست است یا غلط. این فقط بهترین حدس ماست. با این وجود ، هنگامی که راهی را انتخاب می کنیم ، تصمیمات دیگری نیز اتخاذ می کنیم که مبتنی بر آن انتخاب هستند. هر چه بیشتر منتظر تأیید فرض اصلی طراحی باشیم ، تعداد تصمیمات وابسته بیشتر خواهد بود. اگر بعداً (از طریق بازخورد در مرحله ادغام) تشخیص دهیم که تصور اصلی اشتباه بوده است ، ما نه تنها تصمیمات بد بسیاری خواهیم گرفت که باید دوباره مورد بررسی قرار گیرند. ما همچنین باید این کار را انجام دهیم بعد از گذشت زمان طولانی از جایی که خطا رخ داده، چون حافظه افراد نسبت به کار هایی ک انجام داده اند کمرنگ تر شده آنها وقت زیادی را صرف کار قبلی خود می کنند. وقتی ما کل هزینه بازسازی تصمیمات بد و تصمیاتی که وابسته به آن ها بودند و هزینه تأخیر برای ارائه محصول را فاکتور بگیریم ، منافع اقتصادی بازخورد سریع بسیار جذاب است. بازخورد سریع حلقه یادگیری را به سرعت بسته می کند و به ما امکان می دهد مسیرهای توسعه نامناسب را قبل از آسیب جدی اقتصادی کاهش دهیم.

**۱۰ ص آخر فصل ۳ ------------------ مریم**

**Work in Process (WIP)**

کار در حال انجام (یا WIP) به کاری گفته می شود که شروع شده اما هنوز نهایی نشده است. در طی توسعه محصول ، WIP باید شناخته شود و به درستی مدیریت شود.

من چهار اصل agile مرتبط با این موضوع را توصیف می کنم:

1. از دسته هایی(batch) با اندازه مناسب اقتصادی استفاده کنید.
2. موجودی را بررسی کنید و آن را برای جریان خوب مدیریت کنید.
3. روی کار عاطل تمرکز کنید ، نه کارگران عاطل. (عاطل = idle)
4. هزینه تأخیر را در نظر بگیرید.

**Use Economically Sensible Batch Sizes -1**

یکی دیگر از باورهای اساسی ذیل فرایند توسعه ترتیبی و plan-driven این است که ترجیح داده می شود همه کارهای از یک نوع را دسته بندی کرده و در یک فاز انجام دهید. من به این روش به عنوان رویکرد all-before-any اشاره کردم، جایی که ما همه (یا تقریباً همه) یک کار را قبل از شروع کار بعدی کامل می‌کنیم ، بگذارید بگوییم ما همه نیازمندی ها را در مرحله تجزیه و تحلیل ایجاد می‌کنیم. بعد ، کلیه نیازمندی ها را به فاز طراحی منتقل می کنیم. از آنجایی که ما مجموعه کاملی از نیازها را ایجاد کردیم ، اندازه دسته ما در این مثال 100٪ است.

رویکرد all-before-any ، تا حدی نتیجه اعتقاد به اصول تولیدی قدیمی از جمله economies-of-scale که برای توسعه محصول کاربرد دارد است. این اصل بیان می کند که با افزایش تعداد unit ها (افزایش batch size)، هزینه تولید یک واحد کاهش می یابد. بنابراین ، باور توسعه متوالی این است که دسته های بزرگتر در تولید محصول نیز از نظر economies-of-scale مقرون به صرفه است.

در اسکرام ، ما می پذیریم که اگرچه تفکرeconomies-of-scale بستری در تولید و توسعه بوده است، استفاده از آن به صورت جزمی در توسعه محصول آسیب اقتصادی قابل توجهی ایجاد خواهد کرد.

برخلاف چیزی که ممکن است تصور شود ، کار کردن با دسته های کوچکتر در حین تولید محصول مزایای زیادی دارد. Reinertsen در مورد موضوعات اندازه دسته ای به طور عمیق بحث می‌کند و جدول زیر شامل زیرمجموعه ای از مزایای small-batch-size است که وی توصیف می کند (Reinertsen 2009b).

اگر دسته های کوچک بهتر از دسته های بزرگ هستند ، آیا نباید فقط از دسته‌ای با سایز یک استفاده کنیم ، به این معنی که ما هر بار فقط روی یک نیاز کار می کنیم و آن را از طریق همه فعالیت ها جریان می‌دهیم تا زمان کامل شدن و آماده شدن برای مشتری ؟ برخی از افراد از این به عنوان جریان single-piece یاد می‌کنند. همانطور که در فصل های بعدی نشان خواهم داد ، اندازه دسته ای با سایز یک ممکن است دربعضی موارد مناسب باشد ، اما با فرض اینکه هدف دسته‌ای با سایز "یک" باشد ، ممکن است جریان و اقتصاد کلی ما را بهینه نکند.

|  |  |
| --- | --- |
| منفعت | توضیحات |
| کاهش زمان چرخه | دسته های کوچکتر کار کمتری را در انتظار پردازش دارند که به نوبه خود به معنای زمان کمتری برای انتظار انجام کار است. بنابراین ، کارها را سریعتر انجام می دهیم. |
| کاهش تنوع جریان | به رستورانی فکر کنید که مهمانی های کوچک در آن رفت و آمد می کنند (آنها به زیبایی از رستوران عبور می کنند). حال تخلیه یک اتوبوس تور بزرگ (دسته ای بزرگ) و تاثیری که بر جریان در رستوران دارد را تصور کنید. |
| بازخورد سریع | دسته های کوچک بازخورد سریع را تسریع می کنند و عواقب یک اشتباه را کوچکتر می کنند. |
| کاهش ریسک | دسته های کوچک موجودی کمتری را نشان می دهد که قابل تغییر است. دسته های کوچکتر نیز به احتمال کمتری شکست میخورند. (احتمال وقوع خرابی در ده کار بیشتر از پنج کار بیشتر است). |
| کاهش سربار | در مدیریت دسته های بزرگ سربار وجود دارد - به عنوان مثال ، نگه داشتن لیستی از 3000 مورد کار بیشتر از یک لیست 30 تایی نیاز به تلاش دارد. |
| افزایش انگیزه و فوریت | دسته های کوچک تمرکز و احساس مسئولیت را فراهم می کنند. درک تأثیر تأخیر و شکست در هنگام کار با دسته های کوچک در مقابل بزرگ بسیار آسان تر است. |
| کاهش هزینه و رشد برنامه | وقتی در دسته های بزرگ اشتباه می کنیم ، از نظر هزینه و زمانبندی بسیار خسارت وارد می کنیم. وقتی کارها را در مقیاس کوچک انجام می دهیم ، خسارت تا آن حد بزرگ نخواهیم داشت. |

**2- Recognize Inventory and Manage It for Good Flow**

در طول این فصل ، من به شما یادآوری کرده ام که تولید(manufacturing) و توسعه محصول(development) یک چیز نیستند و بنابراین باید متفاوت برخورد کرد. با این حال ، یک درس وجود دارد که تولید آموزش داده شده است که ما باید در توسعه محصول استفاده کنیم ، اما اغلب این کار را نمی کنیم. این درس مربوط به هزینه بالای موجودی کالا است که به آن کار با روند یا WIP نیز می گویند. جامعه نابغه توسعه محصول سالهاست که اهمیت WIP را می داند (Poppendieck and Poppendieck 2003؛ Reinertsen 2009b) و تیمهای Scrum از این مفهوم استقبال می کنند.

تولیدکنندگان کاملاً از موجودی کالا و پیامدهای مالی موجودی آن آگاهی دارند. چگونه ممکن است نباشند؟ موجودی به سرعت روی زمین جمع می شود و منتظر پردازش است. موجودی کارخانه نه تنها از نظر فیزیکی قابل مشاهده است. از نظر مالی نیز قابل مشاهده است. از مدیر مالی یک شرکت تولیدی بپرسید که چه مقدار موجودی (یا WIP) در کارخانه دارد یا در ماه گذشته چه مقدار تغییر کرده است و او می تواند جواب قطعی بدهد.

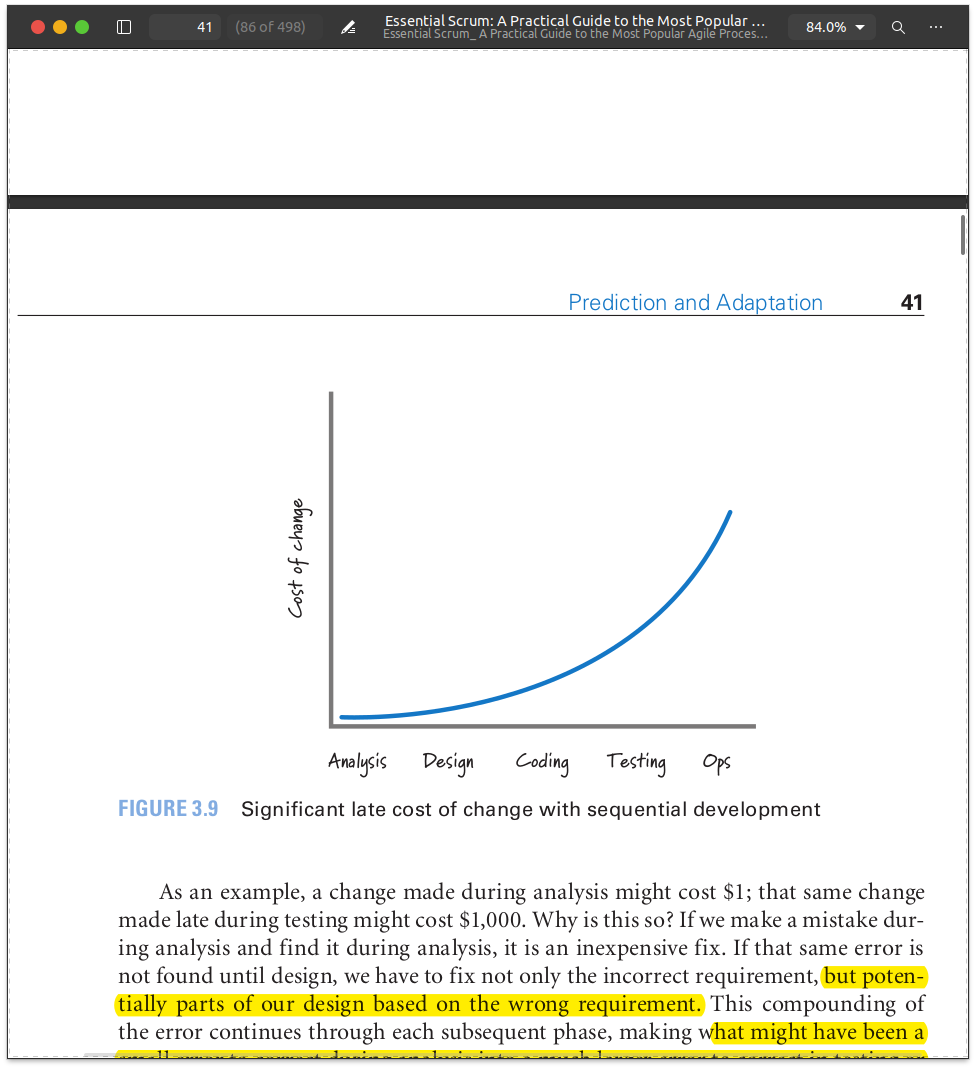
هیچ تولید کننده شایسته ای منتظر مقدار زیادی موجودی کالا نمی نشیند. قطعاتی که در طبقه کارخانه منتظر هستند تا کالاهای نهایی را وارد کنند ، در دفترچه های مالی مستهلک می شوند. از این بدتر ، چه اتفاقی می افتد اگر یک کامیون از قطعات را خریداری کنیم و سپس طراحی محصول را تغییر دهیم؟ ما با همه آن قسمت ها چه کار می کنیم؟ شاید قطعات را دوباره کار کنیم تا در طراحی جدید جای بگیرند. یا بدتر ، شاید ما قطعات را دور بیندازیم زیرا دیگر نمی توان از آنها استفاده کرد. یا ، برای جلوگیری از هدر رفتن قطعات برای قطعاتی که قبلاً خریداری کرده ایم ، آیا ما قصد نداریم طراحی خود را تغییر دهیم (حتی اگر انجام این کار انتخاب صحیحی برای طراحی باشد) بنابراین می توانیم از آنها استفاده کنیم - در معرض خطر تولید محصول با رضایت کمتر.

واضح است که اگر منتظر موجودی زیادی بنشینیم و سپس چیزی تغییر کند ، یک یا چند شکل از هدر رفت قابل توجه را تجربه خواهیم کرد. برای به حداقل رساندن خطرات ، تولیدکنندگان شایسته موجودی کالا را به روشی منطقی از نظر اقتصادی مدیریت می کنند - آنها مقداری موجودی را در دست دارند اما از دوز مناسب مدیریت موجودی just-in-time استفاده می کنند.

سازمان های توسعه محصول ، به طور کلی ، تقریباً از کار در جریان خود آگاهی ندارند. بخشی از این مشکل از این واقعیت ناشی می شود که در توسعه محصول ما با دانشی روبرو هستیم که از نظر فیزیکی به اندازه همان قطعات کف کارخانه قابل مشهود نیست. دانش بسیار ناگهانی است ، مانند کد روی دیسک ، یک سند در یک کابینت پرونده ها، یا یک صفحه بصری روی دیوار.

موجودی کالا در توسعه محصول نیز از نظر مالی قابل مشاهده نیست. از مدیر مالی یک سازمان توسعه محصول سوال کنید که چه مقدار موجودی در سازمان توسعه محصول وجود دارد و او احتمالاً نگاهی گیج کننده به شما می دهد و می گوید: "هیچ". در حالی که تیم مالی سایر اقدامات برای تولید محصول را ردیابی می كند ، احتمالاً موجودی توسعه محصول از این نوع را ردیابی نخواهد كرد.

متأسفانه موجودی کالا (WIP) متغیر مهمی است که باید در طول توسعه محصول مدیریت شود و رویکردهای سنتی توسعه محصول بر مدیریت آن متمرکز نیستند. همانطور که در بحث در مورد اندازه های دسته ای ذکر کردم ، با تنظیم اینکه اندازه دسته کاملاً بزرگ باشد (غالباً 100٪) ، توسعه سنتی در واقع ایجاد مقادیر زیادی موجودی را ترجیح می دهد. پیامد مهم داشتن WIP زیاد در تولید محصول این است که به طور قابل توجهی بر منحنی cost-of-change تأثیر می‌گذارد. (شکل 3.9 را ببینید).



اگرچه می خواهیم توسعه را شروع کنیم ، به برخی نیازمندیها نیاز داریم ، اما نیازی نیست که همه نیازمندیها را داشته باشیم. اگر نیازمندیهای زیادی داشته باشیم ، در صورت تغییر نیازمندیها ، پسماند موجودی را تجربه خواهیم کرد. از طرف دیگر ، اگر موجودی معادلات کافی نداشته باشیم ، جریان سریع کار را مختل می‌کنیم که نوعی هدر دادن نیز است . در اسکرام ، هدف ما یافتن تعادل مناسب بین موجودی کافی و موجودی بیش از حد است.

درک این نکته مهم است که نیازمندیها فقط یکی از موجودی های موجود در توسعه محصول است. در هنگام تولید محصول ، مکان ها و زمان های مختلفی وجود دارد که ما WIP داریم. ما همچنین باید به طور پیشگیرانه این موارد را شناسایی و مدیریت کنیم.

**3- Focus on Idle Work, Not Idle Workers**

در اسکرام ، ما معتقدیم که کار عاطل به مراتب بیشتر از کارگران عاطل هدررفت دارد و از نظر اقتصادی آسیب می رساند. کار عاطل کاری است که ما می خواهیم انجام دهیم (مانند ساختن یا آزمایش چیزی) اما نمی توانیم آن را انجام دهیم زیرا چیزی مانع ما می شود. شاید ما منتظر یک تیم دیگر برای انجام کاری باشیم و تا زمانی که آن تیم کار خود را به پایان برساند ، ما نمی توانیم کارهای خود را انجام دهیم. یا شاید ما فقط آنقدر کار داریم که نمی توانیم همه کارها را یک باره انجام داد. در این حالت ، برخی از کارها idle می مانند تا زمانی که ما آزاد شویم تا بتوانیم روی آن‌ها کار کنیم.

از طرف دیگر ، کارگران عاطل افرادی هستند که ظرفیت موجود برای انجام کار بیشتر را دارند زیرا در حال حاضر 100٪ از آنها استفاده نمی شود.

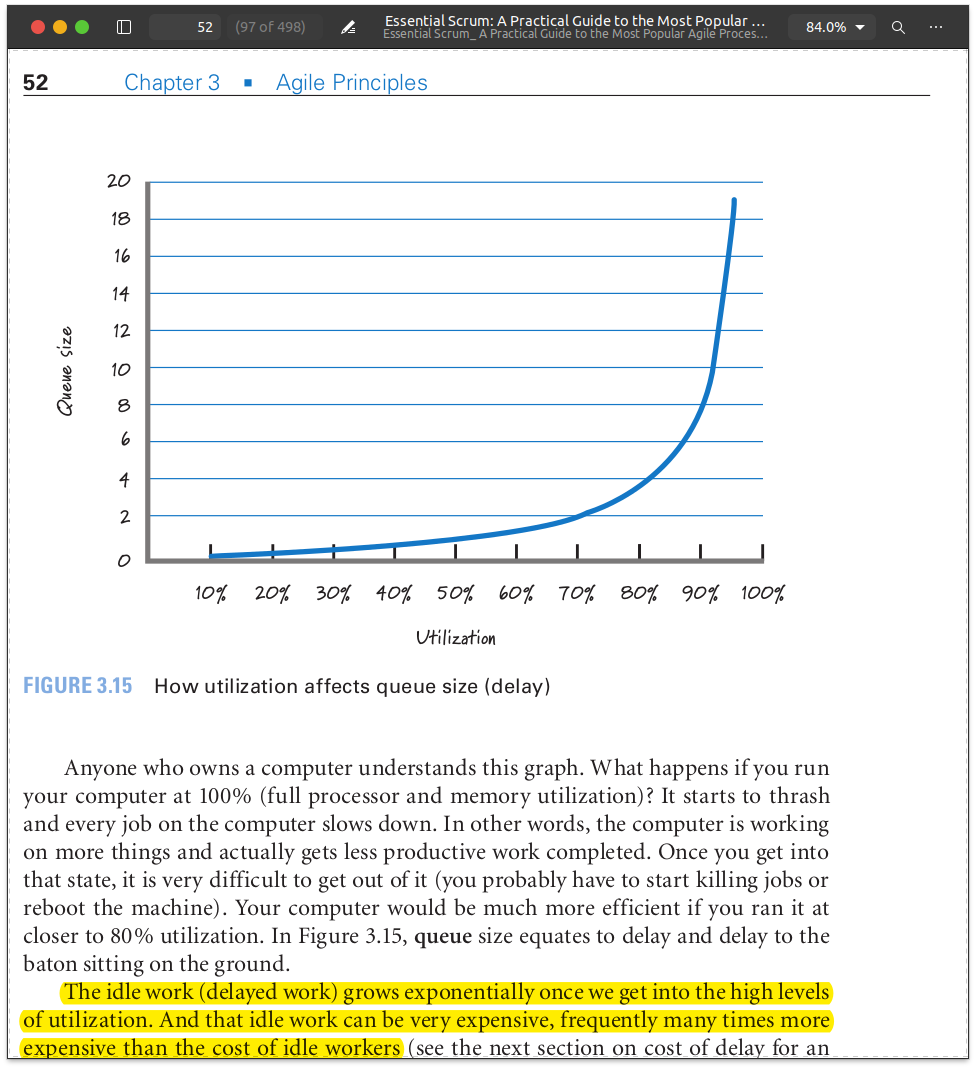
بسیاری از سازمانهای توسعه محصول بیشتر از بین بردن کارهای عاطل بر حذف کارگران عاطل تمرکز دارند. به عنوان مثال ، در تفکر سنتی ، اگر من شما را به عنوان تستر استخدام کنم ، انتظار دارم 100٪ از وقت خود را صرف تست کردن محصول کنید. اگر کمتر از 100٪ وقت خود صرف تست کردن محصول شود ، من آن را هدررفت میدانم (وقتی که می توانید تست کنید بیکار هستید). برای جلوگیری از این مشکل ، من کارهای بیشتری را برای شما در نظر میگیرم- شاید با مشغول کردن شما درچندین پروژه- تا از 100٪ توان شما استفاده کنم.

متأسفانه ، این روش یک نوع هدررفت (هدررفت کارگر بیکار) را کاهش می دهد در حالی که شکل دیگری از هدررفت (کار عاطل) را به طور گسترده افزایش می دهد. و در بیشتر اوقات ، هزینه کار عاطل به مراتب بیشتر از هزینه کارگر بیکار است. بیایید بررسی کنیم که چرا این درست است.

برای نشان دادن این موضوع ، بیایید استراتژی keep-workers-100%-busy را در مسابقات رله 4 × 100 متر در المپیک اعمال کنیم. بر اساس استراتژی مشغول نگه داشتن آنها ، این مسابقه بسیار ناکارآمد به نظر می رسد. من برای دویدن به مردم پول می دهم و به نظر می رسد که آنها فقط یک چهارم وقتشان را صرف دویدن میکنند. بقیه اوقات آنها فقط در اطراف ایستاده اند. خوب ، این درست نیست! من به آنها 100٪ حقوق می دهم بنابراین می خواهم 100٪ از وقتشان را بدوند. حال اگر آنها چوب را حمل نکنند و فقط بالا و پایین سکوها می دوند یا ممکن است مسابقه دیگری را در پیست مجاور اجرا کنند ، چطور؟ به این ترتیب در هنگام دویدن 100٪ از آنها استفاده خواهد شد

البته ، همه ما می دانیم که با 100٪ مشغول نگه داشتن دونده ها مدال طلای رله را کسب نمی کنید. ابتدا با عبور چوب از خط پایان مدال طلا را کسب می کنید. بنابراین ، مهمترین اقدام "تماشای چوب است ، نه دوندگان" (Larman and Vodde 2009). در زمینه تولید محصول ، چوب عاطل بر روی زمین برابر است با کاری که آماده انجام است اما در انتظار منابع لازم بلاک شده است. وقتی چوب روی زمین است ، در مسابقه برنده نمی شوید (محصولات را تحویل می دهید). (من تشبیه چوب و دونده را بسیار دوست دارم زیرا به خوبی نشان می دهد که ما باید کار را تماشا کنیم و نه کارگران را. با این حال ، مانند هر قیاس ، محدودیت های خود را دارد. در این حالت ، روش رله مسابقه برای تحویل کار دقیقاً یکی از جنبه های تولید سنتی و پی در پی محصول است که دوست داریم از آن اجتناب کنیم!)

همچنین ، همه از عواقب 100٪ مشغول نگه داشتن یک منبع اطلاع دارند. اگر یک نمودار را از تئوری صف وام بگیریم ، می توان آسیب آشکاری را که هنگام تلاش برای استفاده 100٪ استفاده شده است ، مشاهده کرد (شکل 3.15 را ببینید).



هر کسی که رایانه داشته باشد این نمودار را درک می کند. چه اتفاقی می افتد اگر با رایانه خود با سرعت 100٪ کار کنید (استفاده کامل از پردازنده و حافظه)؟ شروع به خرد شدن می کند و هر کاری در کامپیوتر کند می شود. به عبارت دیگر ، رایانه در حال کار بر روی چیزهای بیشتری است و در واقع کار موثر کمتری انجام می شود. به محض ورود به آن حالت ، خارج شدن از آن بسیار دشوار است (احتمالاً باید شروع به متوقف کردن تسک ها در رایانه یا راه اندازی مجدد دستگاه کنید). اگر رایانه شما با تقریباً 80٪ توانش کار کند ، بسیار کارآمدتر خواهد بود. در شکل 3.15 ، اندازه صف معادل تاخیر و تاخیر معادل با کارهای عاطل روی زمین است.

هنگامی که به سطح بالایی از عملکرد برسیم ، کار عاطل (کار با تأخیر) به صورت تصاعدی رشد می کند. و این کار عاطل می تواند بسیار گران باشد ، اغلب چندین برابر گرانتر از هزینه کارگران بیکار (برای مثال به بخش بعدی هزینه تاخیر مراجعه کنید). بنابراین ، در اسکرام ، ما کاملاً آگاه هستیم که یافتن گلوگاه های موجود در کار و تمرکز تلاش هایمان برای از بین بردن آنها ، فعالیتی کاملاً منطقی تر از تلاش برای 100٪ مشغول نگه داشتن همه افراد است.

**4- Consider Cost of Delay**

هزینه تأخیر، هزینه مالی مربوط به تأخیر در کار یا تأخیر در دستیابی به یک مرحله مهم است. شکل 3.15 نشان می دهد که با افزایش استفاده از نهایت ظرفیت ، اندازه صف و تأخیر نیز افزایش می یابد. بنابراین ، با کاهش هدررفت کارگران بیکار (با افزایش استفاده از آنها) ، به طور همزمان خسارات مرتبط با کار عاطل را افزایش می دهیم (کارهایی که در صف هایی منتظر سرویس قرار می گیرند). با استفاده از هزینه تأخیر ، می توانیم محاسبه کنیم که کدام هدررفت از نظر اقتصادی خسارت بیشتری دارد.

متأسفانه ، 85٪ سازمان ها هزینه تأخیر را محاسبه نمی کنند (Reinertsen 2009b). این را با این واقعیت ترکیب کنید که اکثر سازمانهای توسعه متوجه نیستند که آنها کار (موجودی) انباشته در صف را تشکیل داده اند و به راحتی می توان فهمید که چرا رفتار پیش فرض آنها تمرکز بر از بین بردن خسارت قابل مشاهده کارگران بیکار است.

در اینجا یک مثال ساده آورده شده است تا نشان دهد چرا هزینه کار عاطل معمولاً بسیار بیشتر از هزینه کارگر بیکار است. این سوال را در نظر بگیرید: آیا باید مستنداتی را در روز اول توسعه به تیم اختصاص دهیم یا در پایان توسعه؟ جدول زیرخلاصه ای از این دو گزینه را نشان می دهد (گزینه های دیگری وجود دارد که می توانیم استفاده کنیم).

|  |  |
| --- | --- |
| پارامتر | مقدار |
| مدت زمان با مستندساز تمام وقت | ۱۲ ماه |
| مدت زمان با مستندسازی در انتها (وقتی به حالت "همه مستند" رسیدیم) | ۱۴ ماه |
| هزینه زمان چرخه برای انجام اسناد در پایان | ۲ ماه |
| هزینه تأخیر به ازای هر ماه | ۲۵۰ هزار دلار |
| مجموع هزینه تاخیر | ۵۰۰ هزار دلار |
|  |  |
| هزینه سالانه کاملاً سنگین مستند | ۹۰ هزار دلار |
| هزینه ماهانه کاملا مستند | ۷.۵ هزار دلار |
| هزینه مستندساز تمام وقت | ۹۰ هزار دلار |
| هزینه برای مستند اگر در پایان تعیین شود | ۱۵ هزار دلار |
| هزینه افزایشی برای مستندساز تمام وقت | ۷۵ هزار دلار |

فرض کنید که ما مستندساز را به مدت 12 ماه به صورت تمام وقت برای کار بر روی این محصول به کار میگیریم ، حتی اگر 100٪ از اوقات به او نیازی نداشته باشیم. هزینه انجام این کار 75 هزار دلار افزایش دارد (این مورد را به عنوان خسارت کارگر بیکار در نظر بگیرید) بالاتر از هزینه ای که اگر ما محصول را به حالت "همه اما مستند شده. (all but documented) ".

اگر در پایان مستندساز را به انجام کلیه کارهای مدارک و مستندات اختصاص دهیم ، فقط به مدت دو ماه به او نیاز خواهیم داشت ، اما همچنین تحویل محصول را به همان دو ماه به تأخیر می اندازیم. اگر حمل و نقل محصول را دو ماه به تأخیر بیندازیم ، هزینه محاسبه شده تأخیر از نظر سود چرخه زندگی ۵۰۰ هزار دلار است (سود چرخه عمر کل پتانسیل سود یک محصول در طول عمر آن ؛ در این مثال ، این پتانسیل ۵۰۰ هزار دلار کاهش می یابد).

در این مثال ، هزینه کارگر بیکار 75 هزار دلار و هزینه کار عاطل 500 هزار دلار است. اگر ما به بهینه سازی استفاده از مستندساز بپردازیم ، به طور اساسی اقتصاد کل محصول را بهینه خواهیم کرد. در طول توسعه محصول ، این نوع داد و ستدها به طور مداوم با ما ارائه می شود. هزینه تاخیر یکی از مهمترین متغیرهایی است که باید در هنگام تصمیم گیری منطقی از نظر اقتصادی در نظر گرفت.

**Progress**

هنگام استفاده از اسکرام ، ما پیشرفت را با آنچه تحویل و تأیید کرده ایم اندازه گیری می کنیم ، نه با توجه به اینکه چگونه طبق برنامه تعریف شده از قبل پیش می رویم یا اینکه درکجای یک مرحله یا فاز خاص از توسعه قرار داریم. من سه اصل agile مرتبط با این موضوع را توصیف می کنم:

۱- با اطلاعات real-time سازگار شوید و دوباره برنامه ریزی کنید.

۲- پیشرفت را با اعتبارسنجی دارایی های در حال کار اندازه گیری کنید

۳- بر تحویل ارزش-محور تمرکز کنید.

**1- Adapt to Real-Time Information and Replan**

در یک فرایند پیوسته و طرح-محور ، برنامه منبع معتبری در مورد چگونگی و زمان کار باید باشد. به همین ترتیب ، انطباق با برنامه پیش بینی می شود. در مقابل ، در اسکرام معتقدیم که اعتقاد بیش از حد به برنامه ما را مرتباً نسبت به اشتباه بودن طرح کور خواهد کرد.

در مراحل توسعه در اسکرام ، هدف ما مطابقت داشتن با برنامه ها و برخی پیش بینی های پیش رو در مورد اینکه تصور میکردیم اوضاع چگونه پیش می‌رود نیست. درعوض ، هدف ما برنامه ریزی سریع و انطباق با جریان اطلاعات مهم اقتصادی است که به طور مداوم در طول تلاش برای توسعه به دست می آید.

**2- Measure Progress by Validating Working Assets**

پیشرفت در طی یک توسعه‌ی پیوسته و برنامه محور با تکمیل یک مرحله و مجاز بودن ورود به مرحله بعدی نشان داده می شود. در نتیجه ، اگر هر مرحله مطابق انتظار شروع و به پایان برسد ، به نظر می رسد توسعه محصول بسیار خوب پیش رفته است. با این وجود ، در پایان ، محصولی که مطابق با برنامه ایجاد کردیم ، ممکن است از نظر مشتری بسیار کمتر از آنچه پیش بینی شده است باشد. آیا اگر به موقع و طبق بودجه بندی پیش‌بینی شده تمام کنیم و در عین حال نتوانیم انتظارات مشتری را برآورده کنیم ، آیا واقعاً می توانیم ادعای موفقیت کنیم؟

با Scrum ، ما پیشرفت را با ساخت محصول معتبر تأیید شده که ارزش تولید می کنند و می توانند برای تأیید مفروضات مهم استفاده شوند ، اندازه گیری می کنیم. این بازخورد را به ما می دهد تا بدانیم قدم بعدی درست چیست. در اسکرام ، این در مورد میزان کار ما نیست. همه چیز در مورد کار با ارزش از نظر مشتری است که ما به پایان می رسانیم.

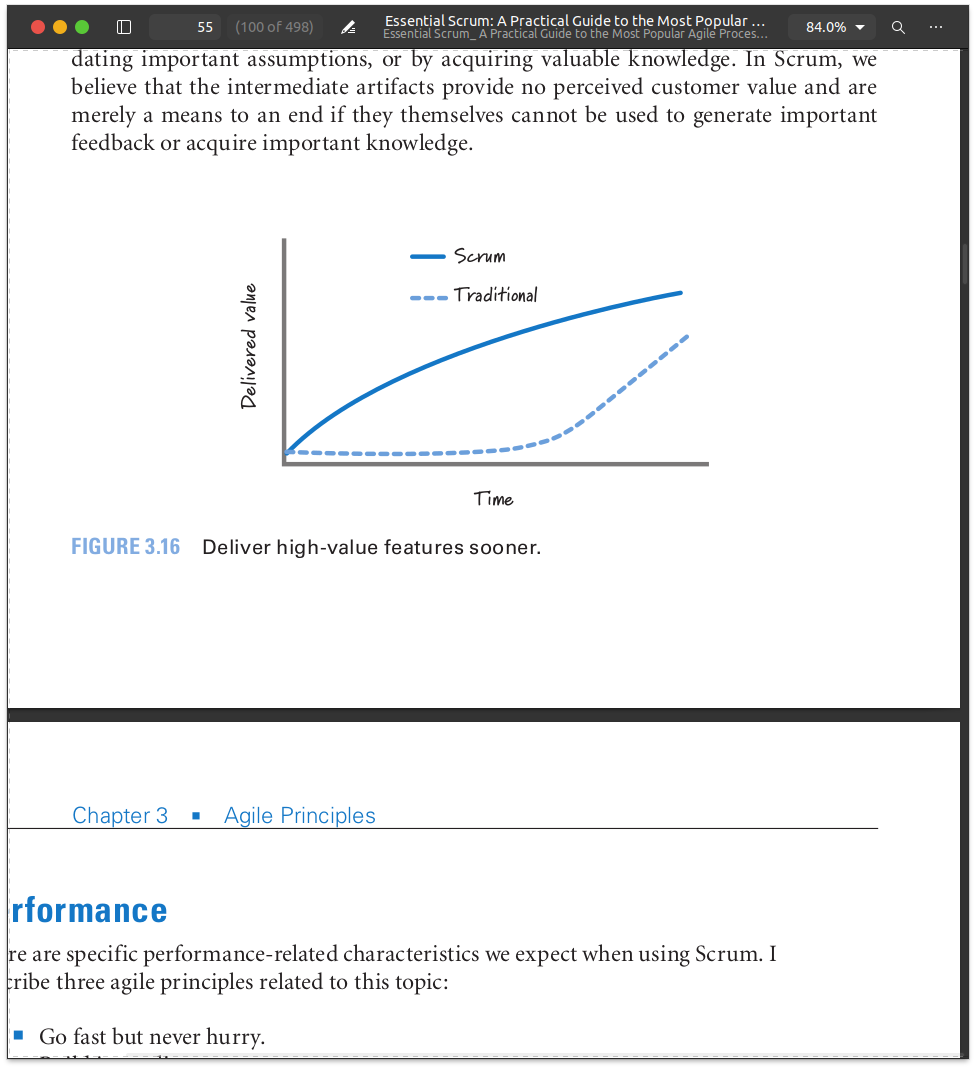
**3- Focus on Value-Centric Delivery**

توسعه‌ی برنامه محور و متوالی بر پیگیری مجدانه فرآیند متمرکز است. با ساختارمندی خود ، یکپارچه سازی و تحویل فیچرها در طول توسعه متوالی در پایان کار اتفاق می افتد (شکل 3.16 را ببینید). با استفاده از این رویکرد ، این خطر وجود دارد که منابع ما (وقت یا هزینه) کم شود قبل از اینکه همه مقادیر مهم را به مشتریان خود تحویل دهیم.

یک اعتقاد مربوط به توسعه سنتی این است که مصنوعات برنامه ریزی و اسنادی که در مسیر ارائه ویژگی ها تولید می شوند ، خود ارزشمند هستند. اگر این مصنوعات واقعاً ارزشمند باشند ، بیشتر اوقات فقط برای فرآیند پایین دستی و نه برای مشتری ارزش دارند. و اگر آنها برای مشتری ارزشمند باشند ، تنها در صورت تحویل نهایی محصول مطلوب به مشتری ، این ارزش حاصل می شود. تا زمانی که این اتفاق نیفتد ، این مصنوعات برای مشتری ارزش مشتری مستقیم ندارند.

از طرف دیگر ، اسکرام نوعی توسعه customer-value- محور است. این بر اساس یک مدل افزایشی، با اولویت بندی تحویل است که در آن ویژگی های با بالاترین ارزش به طور مداوم ساخته و در iteration بعدی تحویل داده می شوند. در نتیجه ، مشتریان زودتر جریان مداوم ویژگی های با ارزش بالا را بدست می آورند.

در اسکرام ، ارزش با تولید محصول در حال کار به مشتری ، با تأیید مفروضات مهم یا کسب دانش ارزشمند تولید می شود. در اسکرام ، ما معتقدیم که مصنوعات میانی هیچ گونه ارزشی از نظر مشتری ارائه نمی دهند و صرفاً وسیله ای برای رسیدن به هدف هستند اگر آن‌ها خودشان برای تولید فیدبک های مهم یا کسب دانش مهم استفاده نشوند.



**Performance**

در هنگام استفاده از اسکرام خصوصیاتی خاص که مربوط به عملکرد هستند وجود دارد. من سه اصل agile مرتبط با این موضوع را توصیف می کنم:

۱- سریع برو اما هرگز عجله نکن.

۲- با کیفیت بسازید.

۳- از تشریفات حداقل کافی استفاده کنید.

**1- Go Fast but Never Hurry**

در توسعه برنامه محور معتقدیم که اگر برنامه را دنبال کنیم و اولین بار کارها را درست انجام دهیم ، از انجام دوباره کاری های پرهزینه و وقت گیر جلوگیری خواهیم کرد. حرکت سریع از مرحله به مرحله البته مطلوب است ، اما هدف اصلی نیست.

در اسکرام ، یک هدف اصلی ، زیرک بودن ، سازگاری و سریع بودن است. با سریع رفتن ، سریع تحویل می دهیم ، سریع بازخورد می گیریم و زودتر محصول را به دست مشتری می رسانیم. یادگیری و واکنش سریع به ما امکان می دهد زودتر درآمدزایی کرده و یا هزینه ها را کاهش دهیم.

با این حال پیشروی سریع را با عجله کردن اشتباه نگیرید. در Scrum ، زمان مهم است ، اما ما برای انجام کارها عجله نداریم. با انجام این کار احتمالاً اصل سرعت پایدار اسکرام نقض می شود - افراد باید بتوانند با سرعتی کار کنند که بتوانند برای مدت زمان طولانی ادامه دهند. علاوه بر این ، عجله کردن احتمالاً به قیمت کیفیت پایین ختم می شود.

یک مثال ممکن است به شما کمک کند تا تفاوت بین پیشروی سریع و عجله را متوجه شوید. من موی تای (کیک بوکسینگ تایلندی) می خوانم. همانطور که در بیشتر ورزش های رزمی صدق می کند ، عملکرد موی تای با سرعت ، بالا می رود. توانایی انجام سریع و دقیق کاتا یا اسپارینگ لذت ورزش و نتیجه را افزایش می دهد. با این حال ، عجله در حرکات با هدف انجام کار ، تأثیر آنها را به میزان قابل توجهی کاهش می دهد و می تواند باعث صدمه جدی به بدن در حین مسابقه شود. هنگام اجرای موی تای ، با سرعت ، چابک و پیوسته حرکت می کنید و سریع با شرایط سازگار می شوید. به عبارت دیگر ، شما باید سریع باشید ، اما هرگز عجله نکنید.

**2- Build In Quality**

در طول توسعه برنامه محور ، این باور وجود دارد که از طریق عملکرد دقیق و متوالی کار ، ما محصولی با کیفیت بالا بدست می آوریم. با این حال ، تا زمانی که محصول یکپارچه را تست نکنیم، که در مرحله آخر فرآیند اتفاق می افتد ، نمی توانیم این کیفیت را تأیید کنیم. اگر آزمایش نشان دهد که کیفیت کم است ، ما باید برای آزمایش کیفیت وارد مرحله پرهزینه تست و رفع مشکل شویم. همچنین ، به دلیل اینکه تیم های مختلفی اغلب روی هر مرحله کار می کند ، تیم تست اغلب به عنوان مسئول کیفیت محصول شناخته می شود.

در Scrum ، کیفیت چیزی نیست که یک تیم تستردر پایان "تست" کند. این چیزی است که تیم Scrum که دارای عملکرد متقابل است مسئول آن میباشد و به طور مداوم در هر اسپرینت ساخته و تأیید می شود. هر گامی که برداشته می شود تا سطح بالایی از کیفیت تکمیل می شود و امکان تولید یا فروش به مشتری را دارد (برای بحث بیشتر در مورد تعریف انجام شده به فصل 4 مراجعه کنید). در نتیجه ، نیاز به هرگونه آزمایش نهایی قابل توجه برای کنترل کیفیت به طور قابل ملاحظه ای کاهش می یابد.

**3- Employ Minimally Sufficient Ceremony**

فرایندهای برنامه محور معمولاً رویکردهایی با تشریفات بالا ، اسنادمدار و فرآیند سنگین هستند. یک عارضه جانبی ارزش-مدار بودن اسکرام این است که تأکید بسیار کمی روی تشریفات فرآیند-محور می شود. منظور من این نیست که اشاره کنم همه تشریفات بد است. به عنوان مثال ، "مراسم" رفتن به میخانه برای معاشرت و پیوند هر جمعه پس از کار ، مراسم خوبی خواهد بود. من به مراسمی اشاره می کنم که غیرضروری است.

رسمیت برخی ممکن است آن را "فرایند به خاطر فرایند" بنامند. چنین مراسمی هزینه دارد اما ارزش کمی دارد یا هیچ ارزشی ندارد (به عبارت دیگر ، این یک نوع ضرر است).

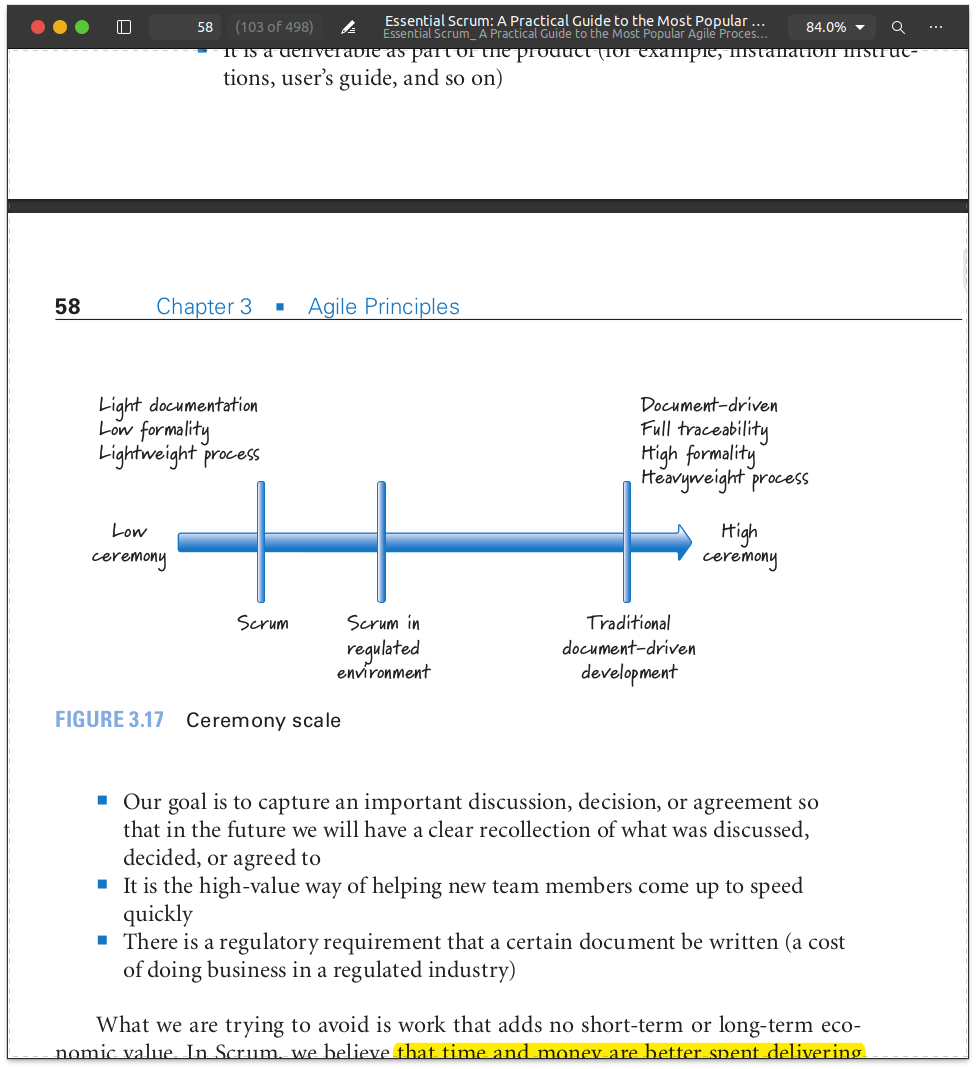
تشریفات نمونه ای که ممکن است رسمی و غیرضروری باشد شامل موارد زیر است:

۱- قبل از اینکه اجازه شروع تست را داشته باشیم ، برای تأیید و انتقال کد از محیط توسعه به محیط QA ، یک فرآیند سه روزه و سنگین وزن لازم است.

۲- تمام ناهنجاری ها باید در یک ابزار نرم افزاری وارد شوند تا بتوان آنها را ردیابی و گزارش کرد ، حتی اگر فقط بتوانم روی شانه شخصی که کنارم نشسته است ضربه بزنم و بگویم: "هی ، این کار نمی کند. میتونی درستش کنی؟ " و از او بخواهید تا تعمیر کند تا بتوانم کارم را ادامه دهم.

۳- من یک سند می نویسم زیرا اکنون زمان تعیین شده برای نوشتن آن سند است ، حتی اگر کسی نمی تواند بگوید چرا آن سند ضروری است یا ارزشمند است.

در Scrum ، هدف ما حذف تشریفات غیرضروری است. بنابراین ، نوار تشریفاتی را در سطح پایین تنظیم می کنیم ، نواری که حداقل کافی باشد (برخی آن را به سختی می گویند) یا به اندازه کافی خوب است. البته ، آنچه که حداقل کافی یا خوب باشد ، می تواند از یک سازمان به سازمان دیگر متفاوت باشد. اگر ما در حال ساخت یک وب سایت جدید رسانه های اجتماعی هستیم ، ممکن است نیاز ما به مراسم فوق العاده کم باشد. از طرف دیگر ، اگر در حال ساخت پیس‌میکر هستیم و مشمول مقررات متعدد دولتی هستیم که به انواع خاصی از مراسم احتیاج دارند ، نوار حداقل کافی بالاتر تنظیم می شود (شکل 3.17 را ببینید).



غالباً تمرکز اسکرام به داشتن تشریفات حداقلی ، به اشتباه به مواردی مانند "اسکرام ضد مستند است." تعبیر میگردد ولی اسکرام ضد مستندات نیست. در عوض ، هنگام استفاده از اسکرام ، ما یک دیدگاه اقتصادی اتخاذ می کنیم و اسناد خود را با دقت بررسی می کنیم. اگر سندی بنویسیم که مناسب قفسه باشد و هیچ ارزشی نداشته باشد ، وقت و هزینه خود را صرف ایجاد یک سند مرده کرده ایم. با این حال ، همه اسناد مرده نیستند. به عنوان مثال ، ما احتمالاً اگر سندی بنویسیم :

۱- این به عنوان بخشی از محصول قابل تحویل است (به عنوان مثال ، دستورالعمل نصب ، راهنمای کاربر و موارد دیگر)

۲- هدف ما گرفتن یک بحث ، تصمیم یا توافق مهم است تا در آینده خاطره روشنی از آنچه در مورد آن بحث شده ، تصمیم گرفته شده یا توافق شده است داشته باشیم.

۳- این روش با ارزش برای کمک به اعضای جدید تیم است که به سرعت می آیند.

۴- یک الزام نظارتی وجود دارد که یک سند خاص نوشته شود (هزینه تجارت در یک صنعت تنظیم شده)

آنچه ما سعی می کنیم از آن اجتناب کنیم کاری است که هیچ ارزش اقتصادی کوتاه مدت یا بلند مدت به آن اضافه نمی کند. در اسکرام ، ما معتقدیم که وقت و پول بهتر است صرف تهیه محصولی مشتری پسند می شود.

**Closing**

در این فصل من به توصیف اصول اصلی agile - اعتقادات اساسی که باعث پیشرفت ما با اسکرام می شود ، اشاره کردم. با این کار ، من مقایسه کردم که این باورها با اعتقادات اساسی در کتاب درسی ، توسعه سنتی ، برنامه محور و برنامه ریزی شده (که در جدول زیر آمده است) متفاوت است.

هدف من از انجام این مقایسه این نیست که شما را متقاعد کنم که متودولوژی waterfall بد است و Scrum خوب است. در عوض ، هدف من این است که نشان دهم باورهای اساسی waterfall ، آن را برای طبقه دیگری از مسئله مناسب تر از Scrum می کند. شما می توانید خود ارزیابی کنید که سازمان شما به چه نوع مشکلاتی رسیدگی می کند و بنابراین ابزار مناسب تری برای آن استفاده است. در فصل های بعدی این کتاب شرح مفصلی از چگونگی تقویت این اصول و ارائه رویکرد قدرتمند به توسعه محصول ارائه می شود.

|  |  |  |
| --- | --- | --- |
| موضوع | Plan-Driven Principle | Agile Principle |
| شباهت بین توسعه و ساخت | هر دو روند مشخصی را دنبال می کنند. | توسعه تولید نیست ؛ توسعه دستورالعمل محصول را ایجاد می کند. |
| ساختار فرایند | توسعه مبتنی بر فاز و متوالی است. | توسعه باید iterative و incremental باشد. |
| درجه فرآیند و تنوع محصول | سعی میکند تنوع فرآیند و محصول را از بین ببرد. | تنوع را از طریق بازرسی ، انطباق و شفافیت اعمال کنید. |
| مدیریت عدم قطعیت | ابتدا عدم قطعیت نقطه پایانی را از بین میبرد وسپس عدم اطمینان را معنا میکند | عدم اطمینان را به طور همزمان کاهش دهید. |
| تصمیم گیری | هر تصمیم را در مرحله مناسب خود میگیرد | گزینه ها را باز نگه دارید. |
| اولین بار درست کردنش | فرض می کنیم که همه اطلاعات صحیح را در جلو داریم تا نیازها و برنامه ها را ایجاد کنیم | ما نمی توانیم آن را از پیش به شکل درست آماده کنیم. |
| اکتشاف در مقابل بهره برداری | از آنچه در حال حاضر شناخته شده است بهره برداری کنید و از آنچه شناخته شده نیست پیش بینی کنید | یک رویکرد انطباقی و اکتشافی را ترجیح دهید. |
| تغییر / ظهور نیازمندی | تغییر در برنامه ها مخل و گران است ، بنابراین باید از آن اجتناب شود. | پذیرش تغییر به روشی منطقی از نظر اقتصادی. |
| پیش بینی در مقابل سازگار | روند بسیار قابل پیش بینی است. | تعادل کار پیش‌بینی شده پیش رو با کار انطباقی در زمان مناسب. |
| مفروضات (دانش غیر معتبر) | این فرایند توانایی تحمل کردن فرضیات طولانی مدت را دارد. | فرضیات مهم را سریع تأیید کنید. |
| فیدبک | یادگیری چالشی در یک حلقه اصلی آنالیز-طراحی-کد-تست رخ می دهد. | استفاده از چندین حلقه همزمان یادگیری |
| فیدبک سریع | این فرایند توانایی تحمل کردن یادگیری دیر هنگام را دارد. | گردش کار را برای بازخورد سریع سازمان دهید. |
| اندازه دسته (قبل از شروع فعالیت بعدی چه مقدار کار به پایان رسیده است) | دسته ها بزرگ هستند ، غالباً 100٪ - all before any. اقتصاد مقیاس باید اعمال شود. | از اندازه های دسته ای کوچکتر و منطقی از نظر اقتصادی استفاده کنید. |
| موجودی / کار در حال اجرا (WIP) | موجودی بخشی از سیستم مورد نظر نیست ، بنابراین روی آن تمرکز نمیشود. | موجودی را بشناسید و برای دستیابی به جریان خوب آن را مدیریت کنید. |
| افراد در برابر ضایعات کاری | افراد را برای دستیابی به سطوح بالای کارکرد اختصاص دهید. | روی کار عاطل تمرکز کنید ، نه کارگران عاطل. |
| هزینه تاخیر | هزینه تأخیر به ندرت مورد توجه است | همیشه هزینه تأخیر را در نظر بگیرید. |
| انطباق با برنامه ریزی | انطباق وسیله اصلی برای دستیابی به نتیجه خوب در نظر گرفته می شود. | به جای انطباق با یک برنامه ، خود را سازگار کنید و دوباره برنامه ریزی کنید. |
| پیشرفت | پیشرفت را با میزان پیشروی در مراحل یا فازها نشان دهید. | پیشرفت را با اعتبارسنجی محصولات در حال کار اندازه بگیرید. |
| مرکزیت | فرآیند محور - روند را دنبال کنید. | ارزش محور - ارزش(value) را تحویل دهید. |
| سرعت | روند را دنبال کنید اولین بار کارها را درست انجام دهید و سریع پیش بروید. | سریع پیش برو ولی عجله نکن |
| چه زمانی به کیفیت بالایی میرسیم | کیفیت پس از یک مرحله تست و رفع مشکل در انتهای فرایند به پایان می رسد. | از ابتدا کیفیت را بسازید. |
| تشریفات (مراسم) | رسمیت (روش ها و ایست های بازرسی کاملاً مشخص) برای اجرای موثر مهم است. | از مراسم و تشریفات حداقلی استفاده کنید. |

**فصل۴ اسپرینت ------------------ مرضیه**

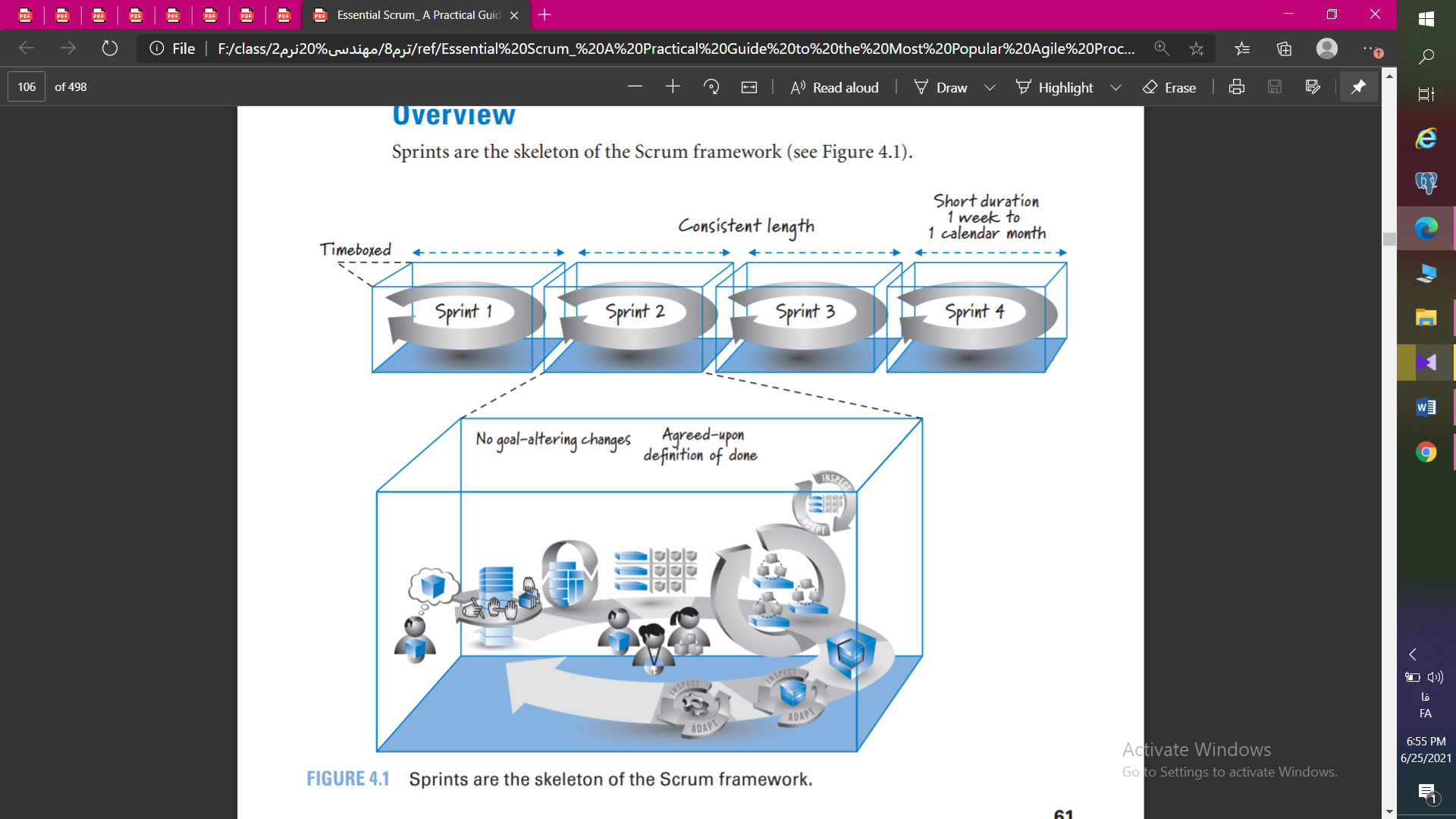
فصل چهارم : SPRINTS

scrum کارها را به صورت تکرار ها(iterations) یا چرخه های حداکثر یک ماهه به نام sprint سازمان می دهد.

در این فصل توضیحات دقیق تری از اینکه sprint چیست، داده می شود. سپس چندین ویژگی اصلی sprint مورد بحث قرار می گیرد: آنها دارای جعبه زمان هستند(timeboxed)؛ دارای یک مدت زمان ثابت و کوتاه هستند؛ دارای هدفی هستند که پس از شروع sprint نباید تغییر کند؛ و باید به آن حالت نهایی مشخص شده توسط definition of done تعریف شده ی تیم، برسد.

مرور

Sprint ها اسکلت scrum هستند.



پیکان اصلی حلقه ای خاکستری در شکل، که از product backlog از طریق حلقه اجرای sprint کشیده شده و اعضای تیم Scrum را در بر می گیرد، نشان دهنده sprint است. که سایر دست آورد ها و فعالیت های اسکرام بر اساس زمان نسبی وقوع آنها در sprint نشان داده شده است. اگرچه اجرای sprint با "the sprint" بودن اشتباه گرفته می شود؛ اما در واقع این فقط یکی از فعالیت هایی است که در طول sprint انجام می شود. همراه با آن، برنامه ریزی(planning) sprint، مرور(review) sprint و گذشته نگر(retrospective) sprint هم انجام می شود.

همه اسپرینت ها دارای جعبه زمان هستند، به این معنی که آنها تاریخ شروع و پایان مشخصی دارند. اسپرینت ها همچنین باید کوتاه باشند و طول آن ها بین یک هفته و یک ماه تقویم باشد. اسپرینت ها باید از نظر طول ثابت باشند، اگرچه در برخی شرایط استثنا مجاز است. به عنوان یک قاعده، هیچ تغییری در هدف و یا تغییر پرسنل در طول اسپرینت مجاز نیست. سرانجام، در طی هر sprint، یک محصول بالقوه قابل حمل(potentially shippable) مطابق با تعریف انجام شده تیم اسکرام(definition of done) توسعه می شود.

اگرچه هر سازمانی پیاده سازی منحصر به فرد خود را در Scrum خواهد داشت ، اما این ویژگی های sprint، به استثنای چند مورد استثنایی که ما در آنها کاوش خواهیم کرد ، برای هر sprint و هر تیم اعمال می شوند.

جعبه زمانی(Timeboxed)

Sprint ریشه در مفهوم timeboxing دارد ، که یک روش مدیریت زمان است و به سازماندهی عملکرد کار و مدیریت scope کمک می کند. هر اسپرینت در یک بازه زمانی با تاریخ های شروع و پایان مشخص برگزار می شود ، که جعبه زمان نامیده می شود. در این جعبه زمانی ، انتظار می رود که تیم با سرعت پایدار کار کند تا مجموعه ای از کارهای انتخاب شده را که با یک هدف اسپرینت (sprint goal) هماهنگ است ، به پایان برساند.

Timeboxing به چند دلیل مهم است(مزایا):

1. محدودیت WIP ایجاد می کند

Timeboxing روشی برای محدود کردن میزانWIP (work in process) است. WIP نمایانگر فهرستی از کار ها است که آغاز شده اند اما هنوز به پایان نرسیده اند. عدم مدیریت صحیح آن می تواند عواقب جدی اقتصادی داشته باشد. از آنجا که تیم قصد دارد فقط روی مواردی کار کند که فکر می کند می تواند در طول آن اسپرینت، شروع کند و به پایان برساند ، جعبه زمانی، محدودیت WIP برای هر اسپرینت را تعیین می کند.

2. ما را مجبور به اولویت بندی می کند

Boxing time ما را مجبور می کند که کمترین کار را که بیشترین اهیت را دارند در اولویت قرار دهیم و انجام دهیم. این کار، تمرکز ما را بر انجام سریع کار با ارزش، بیشتر می کند.

3. پیشرفت را نشان می دهد

Timeboxing همچنین به ما کمک می کند با تکمیل و اعتبار سنجی کارهای مهم تا تاریخ معلوم (پایان اسپرینت) پیشرفت های مربوطه را نشان دهیم. این نوع پیشرفت با عدم تمرکز بر روش های غیر قابل اطمینان گزارشگیری پیشرفت، مانند انطباق با برنامه، خطر و ریسک سازمانی را کاهش می دهد. Timeboxing همچنین به ما کمک می کند تا پیشرفت feature های بزرگی که برای تکمیل آنها به بیش از یک جعبه زمان نیاز داریم را نشان دهیم. تکمیل برخی کارها در جهت دستیابی به این feature ها، به ما این اطمینان را می دهد که در هر اسپرینت پیشرفت ارزشمند و قابل سنجشی حاصل شده است. همچنین به سهامداران و تیم کمک می کند تا دقیقاً از آنچه که باید برای ارائه کل feature انجام شود، آگاه شوند.

4. از کمال گرایی غیرضروری جلوگیری می کند

timeboxing کمک می کند تا از کمال گرایی غیرضروری جلوگیری شود. همه ما زمان زیادی را صرف تلاش برای بدست آوردن چیزی "عالی" یا انجام کاری "عالی" کرده ایم، درحالی که "به اندازه ی کافی خوب" کفایت می کرد. timeboxing با تعیین یک تاریخ پایان ثابت برای sprint که باید تا آن زمان یک راه حل خوب انجام شود ، ما را مجبور به پایان کار بالقوه بی حد و مرز می کند.

5. باعث بسته شدن و پایان می شود

timeboxing باعث بسته شدن و پایان کار می شود. تجربه نشان می دهد که وقتی تیم ها تاریخ مشخصی دارند ، احتمالاً همه چیز به نتیجه می رسد. این واقعیت که پایان اسپرینت مهلت سفت و سختی (hard deadline) را با خود به همراه می آورد ، اعضای تیم را تشویق می کند تا سخت کوشانه خود را وادار کنند تا کار را به موقع انجام دهند. بدون تاریخ پایان مشخص ، احساس فوریت برای تکمیل کار کمتر است.

6. پیش بینی را بهبود می بخشد

Timeboxing قابلیت پیش بینی را بهبود می بخشد. اگرچه ما نمی توانیم دقیقاً کاری را که یک سال دیگر به پایان خواهیم رساند ، با اطمینان زیادی پیش بینی کنیم ، اما کاملا منطقی است که انتظار داشته باشیم در اسپرینت کوتاه بعدی ، کاری را که می توانیم به اتمام برسانیم ، پیش بینی کنیم.

مدت زمان کوتاه(short duration)

اسپرینت های کوتاه مدت مزایای زیادی را به همراه دارند:

1. سهولت برنامه ریزی

اسپرینت های کوتاه مدت باعث سهولت برنامه ریزی می شود. برنامه ریزی کار چند هفته ای آسانتر از کار شش ماهه است. همچنین ، برنامه ریزی در چنین افق های زمانی کوتاه به تلاش بسیار کمتری نیاز دارد و به مراتب دقیق تر از برنامه ریزی با افق طولانی تر است.

2. بازخورد سریع

اسپرینت های کوتاه مدت بازخورد سریع ایجاد می کند. در طول هر اسپرینت ، ما یک working software تولید می کنیم و سپس این فرصت را داریم تا آنچه را که ساخته ایم و نحوه ساخت آن را مورد بازبینی و انطباق قرار دهیم. این بازخورد سریع ما را قادر می سازد تا به سرعت مسیرهای نامطلوب محصول یا رویکردهای نامطلوب توسعه را قبل از تصمیم بد با بسیاری از تصمیمات بعدی که با تصمیم بد همراه شده اند ، کوتاه کنیم. بازخورد سریع همچنین به ما امکان می دهد تا فرصت های اضطراری حساس به زمان را سریعتر کشف و از آنها بهره برداری کنیم.

3. بهبود بازگشت سرمایه

اسپرینت های کوتاه مدت نه تنها از طریق بازخورد سریع باعث بهبود اقتصاد می شوند؛ آنها همچنین امکان تحویل زودرس و مکرر(deliverables) را دارند. در نتیجه ، ما فرصت داریم زودتر درآمدزایی کنیم و بازده کلی سرمایه گذاری را بهبود ببخشیم.

4. خطای محدود شده

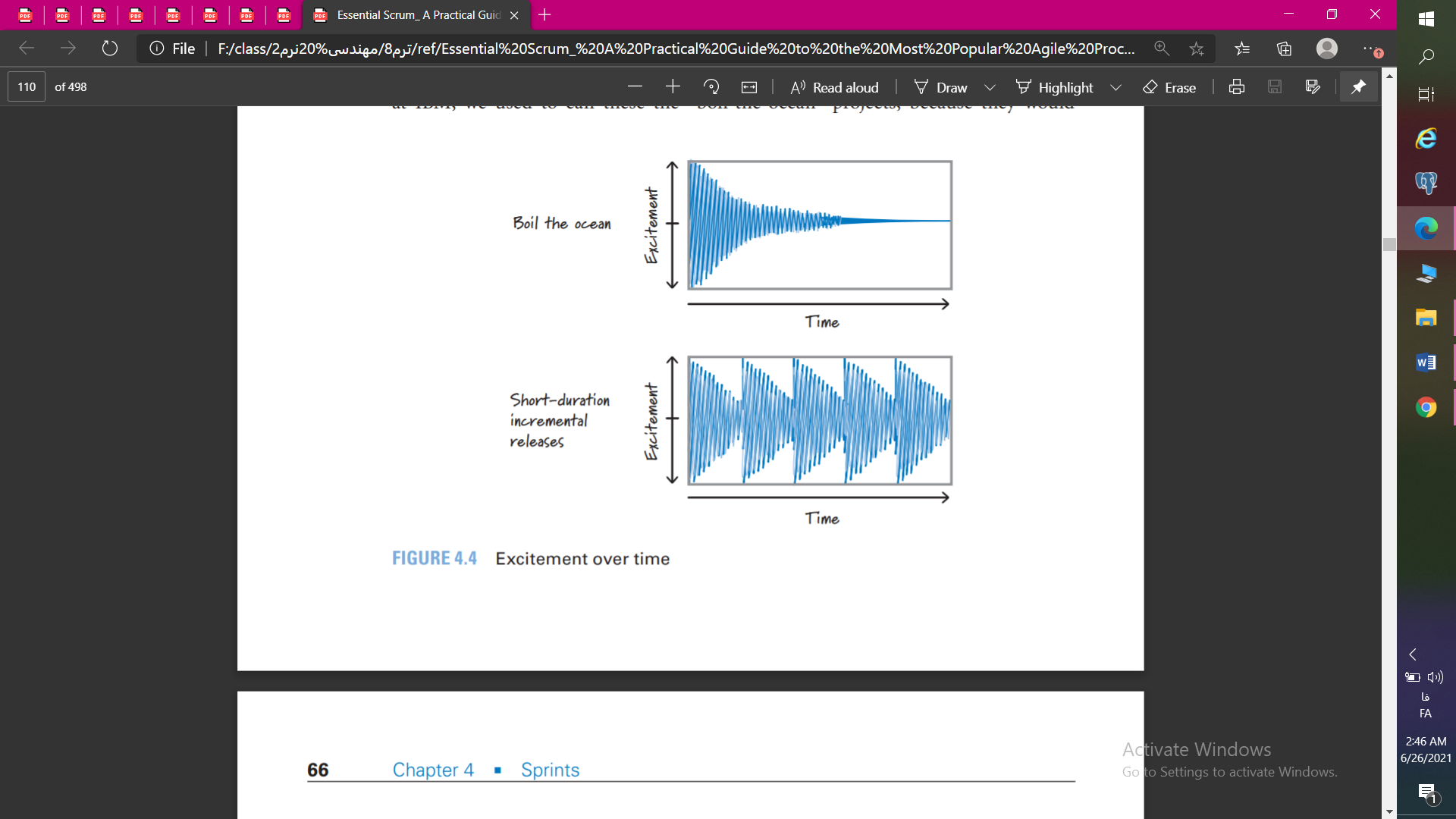
اسپرینت های کوتاه مدت خطا را محدود می کنند. چقدر می توانیم در یک اسپرینت دو هفته ای اشتباه کنیم؟ حتی اگر همه چیز را کورکورانه جلو ببریم، فقط دو هفته ضرر کرده ایم. ما اصرار داریم که اسپرینت های کوتاه مدت را انجام دهیم زیرا آنها هماهنگی و بازخورد مکرر را ارائه می دهند. به این ترتیب ، اگر اشتباه کنیم ، حداقل در یک راه کوچک اشتباه می کنیم.

5. تجدید هیجان

اسپرینت های کوتاه مدت می توانند به تجدید شدن هیجان کمک کنند. ذات انسان در برابر علاقه و هیجان اینطور است که هر چه بیشتر منتظر رضایت ماندن را رد می کند.

اگر ما روی یک پروژه طولانی مدت کار کنیم ، نه تنها احتمال شکست ما وجود دارد؛ بلکه احتمالاً در نهایت، اشتیاق خود را برای تلاش از دست خواهیم داد. (در IBM این پروژه ها را "جوشاندن اقیانوس" می نامند، زیرا اگر بخواهید اقیانوس را بجوشانید، اگر هرگز انجام شود، انجام آنها بسیار طولانی خواهد بود و تلاش زیادی لازم می شود.) بدون پیشرفت و پایانی قابل مشاهده، افراد شروع به بی علاقگی می کنند. در اواخر آن ، آنها ممکن است مایل باشند به کسی پول بدهند تا سراغ یک محصول متفاوت بروند!

اسپرینت های کوتاه مدت با تحویل مداوم دارایی های کاری، هیجان شرکا را بالا نگه می دارند. رضایت از تحویل های زود هنگام و مکرر ، علاقه و تمایل ما برای ادامه کار در جهت هدف را تجدید می کند.

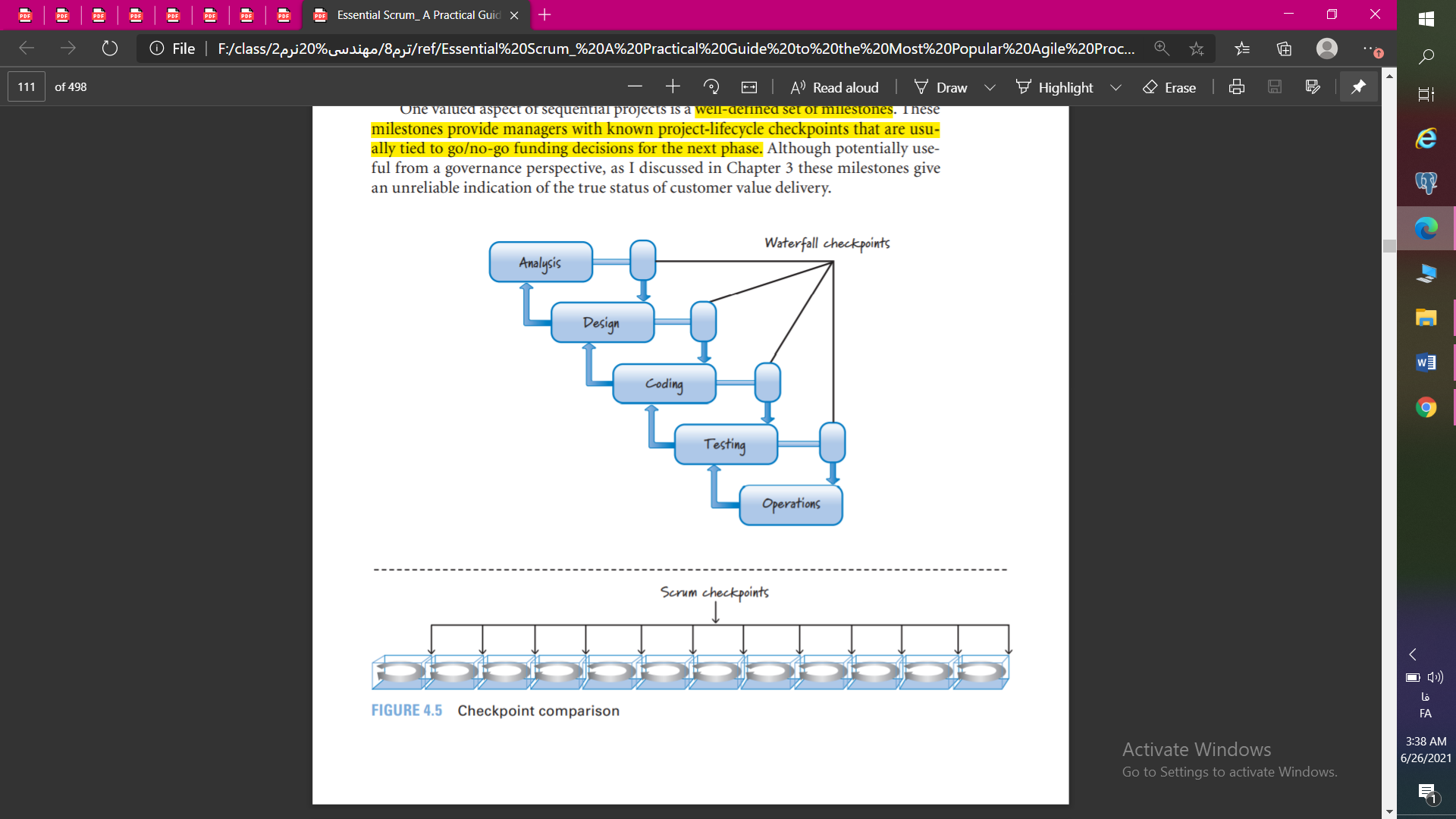


6. Checkpoint های مکرر

اسپرینت های کوتاه مدت چندین Checkpoint معنی دار ارائه می دهند.

یکی از جنبه های ارزشمند پروژه های sequential ، مجموعه ای کاملاً مشخص از milestone هاست. این milestone ها، Checkpoint های شناخته شده برای چرخه حیات پروژه را برای مدیران فراهم می کند که معمولاً با تصمیمات مالی مربوط به مرحله بعدی مرتبط هستند. اگرچه از منظر حاکمیت به طور بالقوه مفید است ، اما این milestone ها نشانه ای غیر قابل اطمینان از وضعیت واقعی ارزش محصول قابل تحویل مشتری است.

اسکرام تعداد زیادی Checkpoint نسبت به پروژه های sequential به مدیران ، سهامداران ، دارندگان محصولات و دیگران ارائه می دهد. در پایان هر اسپرینت کوتاه یک Checkpoint معنی دار وجود دارد (مرور اسپرینت) که به همه اجازه می دهد تصمیمات خود را بر اساس feature هایی که کار می کنند و قابل شرح هستند، قرار دهند. افراد وقتی فرصت بیشتری برای بازرسی و سازگاری دارند، بهتر می توانند با یک محیط پیچیده کنار بیایند.



مدت زمان ثابت(Consistent Duration)

به عنوان یک قاعده ، در توسعه، تیم باید یک مدت زمان ثابت را برای اسپرینت خود انتخاب کند و آن را تغییر ندهد مگر اینکه دلیل قانع کننده ای داشته باشد.

دلایل قانع کننده ای ممکن است شامل موارد زیر باشد:

1. در نظر دارید که برای کسب بازخورد بیشتر ، از اسپرینت چهار هفته ای به اسپرینت دو هفته ای بروید اما می خواهید قبل از تصمیم گیری نهایی ، اسپرینت دو هفته ای را امتحان کنید.

2. تعطیلات سالانه یا پایان سال مالی اجرای اسپرینت سه هفته ای را عملی تر و کاربردی تر از اسپرینت های معمول دو هفته ای می کند.

3. release محصول در یک هفته اتفاق می افتد ، بنابراین اسپرینت دو هفته ای بی فایده خواهد بود.

این واقعیت که تیم نمی تواند تمام کارها را در طول اسپرینت فعلی انجام دهد ، دلیل قانع کننده ای برای افزایش طول اسپرینت نیست. همچنین رسیدن به آخرین روز اسپرینت، و فهمیدن اینکه کار شما تمام نمی شود و تحمیل کردن یک روز یا هفته اضافی مجاز نیست. اینها علائم اختلال عملکرد و فرصت های بهبود است. و دلیل خوبی برای تغییر طول اسپرینت نیستند.

به عنوان یک قاعده، بنابراین، اگر تیمی موافقت کند اسپرینت دو هفته ای انجام دهد، تمام اسپرینت ها باید دو هفته داشته باشند. به عنوان یک موضوع عملی ، اکثر تیم ها (و نه همه) دو هفته را به معنای ده روز هفته در تقویم تعریف می کنند. اگر یک روز تعطیل یا تمرین یک روزه در طول اسپرینت انجام شود ، این قضیه ظرفیت(capacity) تیم را برای آن اسپرینت کاهش می دهد، اما تغییر طول اسپرینت را ضروری نمی کند.

استفاده از طول یکسان برای اسپرینت ها از مزایای هم آهنگی و هم وزنی بهره می برد و برنامه ریزی را ساده می کند.

مزایای هم آهنگی و هم وزنی (Cadence Benefits)

اسپرینت های با مدت زمان یکسان، ریتم منظم، قابل پیش بینی یا به صورت ضربان قلب را برای توسعه Scrum فراهم می کنند. ضربان قلب ثابت و سالم به تیم اسکرام و سازمان این امکان را می دهد تا آشنایی موزون مهمی را در مورد شرایطی که برای دستیابی به جریان سریع و انعطاف پذیر ارزش تجاری اتفاق می افتد ، کسب کنند. این اتفاق می افتد زیرا آهنگ منظم، فعالیت های پیش پا افتاده اما ضروری را عادی می کند ، در نتیجه توانایی ذهنی را برای تمرکز بر کار سرگرم کننده و با ارزش افزوده آزاد می کند.

داشتن یک Sprint کوتاه موزون، تمایل به تسریع شدت کار ایجاد می کند. بر خلاف یک پروژه sequential سنتی که در مراحل آخر شاهد افزایش شدید شدت هستیم، هر اسپرینت دارای یک شدت شبیه به سایر اسپرنیت هاست.

اسپرینت با آهنگ منظم باعث کاهش قابل توجه overhead هماهنگی می شود. با اسپرینت های با طول ثابت می توانیم به طور همزمان برنامه ریزی اسپرینت ، مرور اسپرینت ، و فعالیت های گذشته اسپرینت را برای بسیاری از اسپرینت همزمان انجام دهیم. از آنجا که همه می دانند چه زمانی این فعالیت ها اتفاق می افتد ، هزینه های اضافی برای برنامه ریزی برای یک دسته بزرگ از اسپرینت ها به طور قابل توجهی کاهش می یابد.

به عنوان مثال ، اگر ما قرار است یک توسعه ی یک ساله را در اسپرینت های دو هفته ای انجام دهیم، می توانیم رویداد تکراری را در تقویم همه برای 26 مرور(review) اسپرینت بعدی ارسال کنیم. اگر اجازه دادیم مدت اسپرینت ها از یک اسپرینت به اسپرینت دیگر متفاوت باشد، تلاش بیشتری را برای هماهنگی برنامه ذینفعان در مورد اطلاع رسانی بازنگری اسپرینت لازم داریم. این فرض می کند که ما حتی می توانیم زمانی مناسب را برای مجموعه اصلی ذینفعان که برنامه آنها احتمالاً هفته ها قبل پر شده است، پیدا کنیم.

سرانجام، اگر چندین تیم در یک پروژه واحد داشته باشیم ، اینکه همه تیم ها دارای اسپرینت های هماهنگ باشند، امکان همگام سازی کار در همه تیم ها را فراهم می کند.

برنامه ریزی را آسان می کند(Simplifies Planning)

استفاده از مدت زمان ثابت، فعالیت های برنامه ریزی را نیز ساده می کند. وقتی طول تمام اسپرینت ها یکسان باشد(حتی اگر به دلیل تعطیلات، یک روز یا ظرفیت کمتری در هر اسپرینت داشته باشند) ، تیم با مقدار کارهایی که می تواند در یک اسپرینت معمول انجام دهد، راحت است(به عنوان velocity آن گفته می شود). velocity به طور معمول در اسپرینت تحت قاعده در آورده می شود. اگر طول اسپرینت بتواند متفاوت باشد ، ما واقعاً یک واحد مشخص در اسپرینت نداریم. گفتن جملاتی از جمله " velocityمتوسط تیم در هر اسپرینت 20 امتیاز است" معنادار نخواهد بود.

در حالی که مطمئناً می توان velocity تیم را حتی اگر از اسپرینت های با طول متغیر استفاده کند محاسبه کرد، اما پیچیده تر است. چسبیدن به یک مدت ثابت اسپرینت، محاسبات ما را در داده های velocity گذشته ی یک تیم ساده می کند.

مدت زمان ثابت اسپرینت، بقیه ریاضیات برنامه ریزی را هم ساده می کند. به عنوان مثال ، اگر ما در حال کار بر روی یک release با تاریخ مشخص شده هستیم ، و ما اسپرینت هایی با مدت زمان ثابت داریم ، محاسبه تعداد اسپرینت در release به راحتی تمرین ریاضیات تقویم است (ما تاریخ امروز را می دانیم ، تاریخ انتشار را می دانیم و ما می دانیم که تمام اسپرینت ها یک طول دارند). اگر مدت اسپرینت مجاز به تغییر باشد ، محاسبه تعداد اسپرینت در release می تواند به طور قابل توجهی چالش برانگیزتر باشد (زیرا ما باید برنامه ریزی اولیه گسترده ای را انجام دهیم) ، شامل overhead غیر ضروری می شود و احتمالاً نسبت به مدت زمان ثابت اسپرینت بسیار کمتر قابل اطمینان است.

بدون تغییر در تغییر هدف(No Goal-Altering Changes)

یک قانون مهم اسکرام بیان می کند که پس از تثبیت هدف و اسپرینت و شروع اجرای آن، هیچ تغییری بر هدف آن اثر بگذارد، مجاز نیست.

هدف اسپرینت چیست؟

هر اسپرینت را می توان با یک هدف اسپرینت خلاصه کرد که هدف تجاری و ارزش اسپرینت را توصیف می کند. به طور معمول هدف اسپرینت، تمرکز مشخص و واحدی دارد، مانند:

1. از تولید گزارش اولیه پشتیبانی کنید.

2. داده های نقشه آمریکای شمالی را بارگیری و تصحیح کنید.

3. توانایی ارسال پیام متنی از طریق یک نرم افزار ، سیستم عامل و پشته سخت افزار یکپارچه را نشان دهید.

مواردی وجود دارد که ممکن است یک هدف اسپرینت چند وجهی باشد ، به عنوان مثال ، "چاپ اولیه را انجام دهید و جستجو را براساس تاریخ پشتیبانی کنید."

در طول برنامه ریزی اسپرینت ، تیم توسعه باید به بهبود و موافقت با هدف اسپرینت بپردازد و از آن برای تعیین موارد product backlog که می تواند تا پایان اسپرینت کامل کند استفاده کند. این موارد موجود در product backlog برای توضیح بیشتر هدف اسپرینت است.

تعهد متقابل

هدف اسپرینت، پایه و اساس تعهد متقابل تیم و product owner است. تیم متعهد می شوند که هدف را تا پایان اسپرینت برآورده کنند و product owner متعهد می شود که هدف را در طول مدت اسپرینت تغییر ندهد.

این تعهد متقابل اهمیت اسپرینت را در ایجاد تعادل در نیازهای کسب و کار برای انطباق پذیری با تغییر نشان می دهد ، در حالی که به تیم اجازه می دهد تمرکز کند و به طور کارآمد استعداد خود را برای ایجاد ارزش در مدت زمان کوتاه و ثابت اعمال کند. با تعریف و پیوستن به یک هدف اسپرینت، تیم اسکرام قادر است تمرکز خود را (در یک حوزه) روی یک هدف کاملاً مشخص و ارزشمند حفظ کند.

تغییر vs شفاف سازی

اگرچه هدف اسپرینت نباید تغییر کند ، اما شفاف سازی هدف مجاز است. بگذارید این دو را از هم متمایز کنیم.

چه چیزی تغییر ایجاد می کند؟ تغییر به هر دگرگونی در کار یا منابعی گفته می شود که امکان ایجاد هدررفت اقتصادی با معنی، ایجاد آسیب در جریان کار یا افزایش قابل توجه scope کار در اسپرینت را داشته باشد. افزودن یا حذف یک آیتم product backlog از یک اسپرینت، یا تغییر قابل توجه scope یک آیتم product backlog که در اسپرینت قرار دارد، معمولاً تغییر محسوب می شود. مثال زیر، یک تغییر را نشان می دهد:

- product owner: "وقتی گفتم که ما باید بتوانیم در پایگاه داده پلیس برای یک مجرم نوجوان جستجو کنیم ، منظور من فقط نام خانوادگی و نام خانوادگی نبود. منظور من همچنین این بود که ما باید بتوانیم پایگاه داده را بر اساس تصویری از خال کوبی بدن مظنون جستجو کنیم! "

اضافه کردن توانایی جستجو بر اساس تصویر احتمالاً نشان دهنده تلاش بیشتر است و تقریباً بر توانایی تیم برای انجام تعهد برای ارائه جستجو براساس نام خانوادگی و نام خانوادگی تأثیر می گذارد. در این حالت ، product owner باید در نظر داشته باشد كه یک آیتم جدید برای ویژگی سرچ با عکس، به product backlog اضافه كند تا در اسپرینت بعدی روی آن كار شود.

چه چیزی شفاف سازی را تشکیل می دهد؟ توضیحات جزئیات اضافی ارائه شده در طول اسپرینت است که به تیم در رسیدن به هدف اسپرینت خود کمک می کند. تمام جزئیات مربوط به موارد موجود در product backlog ممکن است در آغاز اسپرینت به طور کامل مشخص نشوند. بنابراین ، کاملا منطقی است که تیم در طول اسپرینت، سوال هایی در باره ی شفاف سازی از product owner بپرسد و او پاسخ دهد. مثال زیر شفاف سازی را نشان می دهد:

- تیم توسعه: "هنگامی که شما گفتید جستجوی برای یک مجرم باید در یک لیست نمایش داده شود ، آیا شما ترجیحی برای نحوه قرارگیری آن لیست دارید؟ "

+ صاحب محصول: "بله ، آنها را بر اساس نام خانوادگی حروف الفبا مرتب کنید."

- تیم توسعه: "خوب ، ما می توانیم این کار را انجام دهیم."

به این ترتیب ، مالک محصول می تواند (و باید) شفاف سازی هایی را در طول اسپرینت ارائه دهد.

پیامدهای تغییر

ممکن است به نظر برسد که این قانون که هیچ تغییری در هدف رخ ندهد، در تعارض مستقیم با قاعده ی اصلی اسکرام است که ما باید تغییر را بپذیریم. ما تغییر را پذیرفته ایم ، اما می خواهیم آن را به روشی متعادل و منطقی از نظر اقتصادی بپذیریم.

پیامدهای اقتصادی یک تغییر با افزایش سطح سرمایه گذاری ما در کار تغییر یافته ، افزایش می یابد.

ما بر روی آیتم های product backlog سرمایه گذاری می کنیم تا بتوانیم آنها را برای کار در اسپرینت آماده کنیم. با این حال ، به محض شروع اسپرینت، سرمایه گذاری ما بر روی موارد product backlog افزایش می یابد (زیرا ما در طول اسپرینت برای برنامه ریزی و بحث در سطح task ها وقت صرف میکنیم). اگر بخواهیم پس از برنامه ریزی اسپرینت، تغییری ایجاد کنیم ، نه تنها سرمایه گذاری که بر روی برنامه ریزی کرده ایم را به خطر می اندازیم، بلکه برای ایجاد تغییرات مجدد در طول اسپرینت نیز هزینه های اضافی متحمل خواهیم شد.



علاوه بر این ، هنگامی که اجرای اسپرینت را شروع می کنیم ، سرمایه گذاری ما در کار، با انتقال آیتم های موجود در product backlog به حالت های "باید انجام شود" (کار هنوز شروع نشده است) ، "در حال انجام" (کار در حال انجام است) و "انجام شده" (کار به اتمام رسیده) ، حتی بیشتر می شود.

بگذارید بگوییم ما می خواهیم ویژگی X را که در حال حاضر بخشی از کاری است که تعهد کرده ایم در اسپرینت انجام دهیم، و ویژگی Y که تعهد نکرده ایم آن را در این اسپرینت انجام دهیم، جایگزین کنیم. حتی اگر کار بر روی ویژگی X را شروع نکرده ایم ، باز هم برای برنامه ریزی ها، اتلاف و هدررفتی را متحمل می شویم. علاوه بر این ، ویژگی X ممکن است با سایر ویژگی های اسپرینت نیز وابستگی داشته باشد ، بنابراین تغییری که بر ویژگی X تأثیر بگذارد ، می تواند یک یا چند ویژگی دیگر را نیز تحت تأثیر قرار دهد ، بنابراین تأثیر بر هدف اسپرینت را تقویت می بخشد.

اگر کار بر روی ویژگی X از قبل آغاز شده باشد، علاوه بر هدررفت های ذکر شده در بالا، ما می توانیم اتلاف های بالقوه ی دیگری نیز داشته باشیم. به عنوان مثال ، ممکن است تمام کارهایی که قبلاً روی ویژگی X انجام شده است باید کنار گذاشته شود. و ممکن است در حذف کار نیمه تمام ویژگی X که ممکن است در آینده هرگز استفاده نکنیم، هدر رفت اضافه داشته باشیم (محصول بالقوه قابل حملی(potentially shippable product) که در انتهای اسپرینت تهیه می کنیم، نباید کار های نیمه تمام را هم شامل شود).

و البته ، اگر ویژگی X از قبل تکمیل شده باشد ، ممکن است ما سرمایه گذاری کاملی را که در ویژگی X انجام داده ایم ، تلف کنیم. همه این اتلاف ها تجمیع می شوند!

علاوه بر پیامدهای اقتصادی مستقیم اتلاف، مسائل اقتصادی می تواند به طور غیر مستقیم تحت تأثیر تخریب بالقوه انگیزه و اعتماد تیمی، که می تواند توأم با یک تغییر باشد، قرار گیرد. وقتی product owner تعهدی در عدم تغییر هدف اسپرینت ببندد و سپس تعهد خود را نقض کند، طبیعتاً تیم بی انگیزه می شود. که تقریباً به طور قطع بر تمایل وی برای کار سخت کوشانه اش برای تکمیل سایر موارد product backlog تأثیر می گذارد. علاوه بر این ، نقض تعهد می تواند به اعتماد در تیم اسکرام آسیب برساند، زیرا تیم توسعه دیگر اعتماد نخواهد کرد به اینکه product owner مایل است به تعهدات خود پایبند باشد.

عمل گرا بودن

قاعده ی عدم تغییر هدف، تنها یک قاعده است، نه یک قانون. تیم اسکرام باید عملگرا باشد.

اگر شرایط کسب و کار به گونه ای تغییر کند که ایجاد تغییر در هدف اسپرینت ضروری به نظر برسد، چه می شود؟ فرضاً یک رقیب در طول اسپرینت ما محصول جدید خود را به بازار عرضه می کند. پس از بررسی محصول جدید ، نتیجه می گیریم که باید هدفی را که برای اسپرینت فعلی خود تعیین کرده ایم ، تغییر دهیم؛ زیرا کاری که ما انجام می دهیم با توجه به آنچه رقیب ما انجام داده است ، اکنون از نظر اقتصادی بسیار کم ارزش است. آیا باید کورکورانه از قانون عدم تغییر در هدف گذاری پیروی کنیم و اسپرینت خود را تغییر ندهیم؟ احتمالا نه.

اگر یکی از سیستم های حیاتی مان خراب شده باشد و بعضی یا همه افراد تیم ما تنها کسانی هستند که می توانند آن را تعمیر کنند، چه می شود؟ آیا برای رفع آن نباید اسپرینت فعلی را قطع کنیم؟ آیا باید به کسب و کار و بیزینس بگوییم که در اولین فرصت در اسپرینت بعدی، آن خرابی سیستم را برطرف می کنیم؟ احتمالا نه.

در پایان، عمل گرایانه بودن، قاعده ی عدم تغییر هدف را مغلوب می کند. ما باید از نظر اقتصادی منطقی عمل کنیم. همه اعضای تیم اسکرام می توانند قدر آن را بدانند. اگر اسپرینت فعلی را تغییر دهیم، پیامدهای اقتصادی منفی را که قبلاً در مورد آن صحبت کردیم ، تجربه خواهیم کرد. با این وجود ، اگر پیامدهای اقتصادی این تغییر بسیار کمتر از پیامدهای اقتصادی به تعویق انداختن تغییر باشد ، ایجاد این تغییر، تصمیمی هوشمندانه برای کسب و کار است. اگر دستاورد های اقصادی تغییر در مقابل عدم تغییر، بی اهمیت باشد، هیچ تغییری در هدف اسپرینت نباید ایجاد شود.

در مورد انگیزه و اعتماد تیمی، وقتی product owner در مورد ضرورت تغییر، بحث صریح و متمرکز اقتصادی با تیم داشته باشد، اکثر تیم ها این نیاز را درک می کنند و از آن استقبال می کنند، بنابراین وجود انگیزه و اعتماد، کمک کننده خواهد بود.

خاتمه غیر عادی

اگر هدف اسپرینت کاملاً پوچ و نامعتبر شود، ممکن است تیم اسکرام تصمیم بگیرد که ادامه دادن اسپرینت فعلی منطقی نیست و به product owner توصیه کنند اسپرینت را به طور غیر عادی خاتمه دهد. هنگامی که یک اسپرینت به طور غیرطبیعی خاتمه می یابد، اسپرینت فعلی به طور ناگهانی پایان می یابد و تیم اسکرام برای دوره ی کار های گذشته (retrospective)ی اسپرینت جمع می شوند. سپس تیم، با product owner یک جلسه ی برنامه ریزی برگزار می کنند تا اسپرینتی با هدف متفاوت و مجموعه ای متفاوت از موارد product backlog، برنامه ریزی کند.

خاتمه اسپرینت در شرایطی مورد استفاده قرار می گیرد که از نظر اقتصادی یک رویداد مهم اتفاق افتاده باشد، مانند اقدامات یک رقیب که اسپرینت یا اعتبار مالی محصول را کاملاً بی اعتبار کرده باشد.

اگرچه product owner گزینه لغو هر اسپرینت را برای خود محفوظ می داند، اما به تجربه، نادر است که product owner ها از این گزینه استفاده کنند. معمولاً اقدامات با شدت کمتری وجود دارد که یک تیم اسکرام می تواند برای سازگاری با شرایط موجود انجام دهد. به یاد داشته باشید که، اسپرینت ها کوتاه هستند و به طور متوسط، وقتی که یک وضعیت ایجاد کننده ی تغییر، به وجود آید، تیم تقریباً در نیمی از راه اسپرینت قرار دارد. از آنجا که ممکن است فقط یک هفته یا بیشتر از زمان اسپرینت باقی مانده باشد، که اسپرینت نیازمند تغییر شود، دستاورد های اقتصاد پایان کار ممکن است مطلوب تر از ادامه روند باشد. و در بسیاری از مواقع می توان یک تغییر کوچک تر ایجاد کرد، مثلاً به جای خاتمه اسپرینت، یک feature را کاهش دهیم تا زمان بیشتری برای رفع خرابی پیش آمده سیستم حیاتی داشته باشیم.

مهم است که بدانیم پایان زودهنگام اسپرینت علاوه بر تأثیر منفی بر روحیه، اختلال جدی در جریان سریع و انعطاف پذیر feature ها دارد و بسیاری از مزایای اسپرینت های با طول ثابت را که قبلاً ذکر کردیم نفی می کند. خاتمه ی اسپرینت باید آخرین گزینه باشد.

در صورت خاتمه اسپرینت، تیم اسکرام باید طول اسپرینت بعدی را تعیین کند.

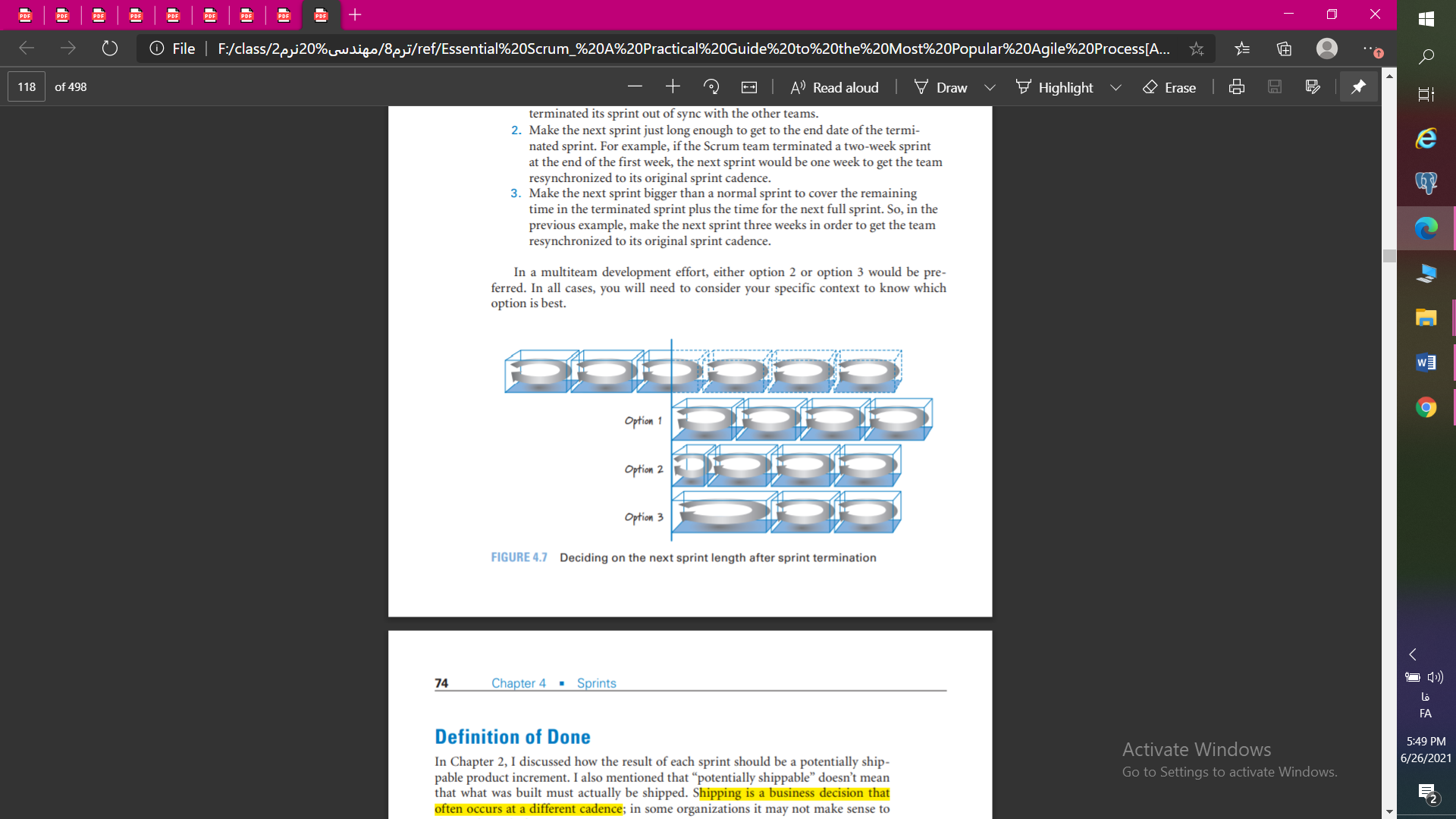
سه گزینه برای این کار وجود دارد:

1. با همان طول اسپرینت اصلی ادامه بدهیم. این کار، این مزیت را دارد که طول اسپرینت یکنواخت را در طول توسعه حفظ می کند (البته به جز اسپرینت پایان یافته). اگر چندین تیم اسکرام در یک توسعه یکسان مشارکت داشته باشند، استفاده از طول اسپرینت اصلی باعث می شود تیم اسکرام که اسپرینت خود را به پایان رسانده است، با سایر تیم ها هماهنگ نشود.

2. اسپرینت بعدی را فقط به اندازه ای طولانی کنید تا به تاریخ پایان اسپرینت پایان یافته برسید. به عنوان مثال ، اگر تیم اسکرام اسپرینت دو هفته ای را در پایان هفته اول خاتمه دهد، اسپرینت بعدی یک هفته خواهد بود تا تیم مجدداً با اسپرینت اصلی خود هماهنگ شود.

3. طول اسپرینت بعدی را بزرگتر از اسپرینت عادی قرار دهید تا مدت زمان باقیمانده در اسپرینت پایان یافته بعلاوه زمان اسپرینت کامل بعدی را پوشش دهد. بنابراین، در مثال قبلی، اسپرینت بعدی را سه هفته تنظیم کنید تا تیم بتواند با اسپرینت اصلی خود هماهنگ شود.

در توسعه هایی که دارای چند تیم هستند، گزینه 2 یا گزینه 3 ترجیح داده می شود. در هر وضعیتی، شما باید زمینه و شرایط خاص خود را در نظر بگیرید تا بدانید کدام گزینه بهترین است.



**۵ ص آخر فصل۴ ---------------------- مریم**

**Definition of Done**

در فصل 2 ، من بحث کردم که چگونه نتیجه هر اسپرینت باید یک محصول به طور بالقوه عرضه کردنی باشد. من همچنین اشاره کردم که “potentially shippable” به این معنی نیست که آنچه ساخته شده است باید در‌واقع قابل عرضه باشد. عرضه کردن یک تصمیم تجاری است که غالباً با ریتم متفاوتی اتفاق می افتد. در برخی از سازمان ها ممکن است منطقی نباشد که در پایان هر اسپرینت محصول عرضه شود.

به طور بالقوه قابل عرضه بودن بهتر است به عنوان یک حالت اطمینان بدانیم که آنچه در اسپرینت ساخته شده است در واقع done شده است ، به این معنی که در اسپرینت کار done نشده مهم از نظر زیرساختی وجود ندارد (مانند آزمایش مهم یا یکپارچه سازی و غیره) که باید قبل از اینکه ما آن را از اسپرینت کامل کنیم انجام شده باشد البته اگر عرضه کردن خواسته تجاری ما باشد. برای تعیین اینکه آیا محصول تولید شده به طور بالقوه قابل عرضه است ، تیم Scrum باید یک تعریف کاملاً مشخص و توافق شده از done شده داشته باشد.

**What Is the Definition of Done**

از نظر مفهومی ، تعریف done شده لیستی از انواع کارهایی است که انتظار می رود تیم با موفقیت به اتمام برساند ، قبل از اینکه اعلام کند کار خود قابل عرضه است (به جدول زیر مراجعه کنید).

|  |  |
| --- | --- |
| تعریف done شده | |
|  | طراحی بازنگری شده است |
|  | کد کامل شده است :   * کد refactor شده است * کد در فرمت استاندارد است * کد کامنت گذاری شده است * کد checked in شده است * کد inspected شده است |
|  | مستندات کاربر به روز شده است |
|  | تست شده است :   * Unit tested (تست واحد) * Integration tested (تست یکپارچه سازی) * Regression tested (تست رگرسیون) * Platform tested (سیستم عامل آزمایش شده است) * Language tested (زبان تست شده) |
|  | نواقص مرتفع شده است |
|  | قابل قبول بودن تست شده است |
|  | روی سرور بارگذاری شده است. |

بدیهی است که موارد خاص موجود در چک‌لیست فوق به تعدادی از متغیرها بستگی دارد:

۱- ماهیت محصول در حال ساخت

۲- فناوری های مورد استفاده برای ساخت آن

۳- سازمانی که در حال ساخت آن است

۴- موانع فعلی که بر آن ممکن است تأثیر بگذارد

در اکثر مواقع ، تعریف حداقل و ناچیز done شده یعنی باید یک قطعه کامل از عملکرد محصول را داشته باشد ، تعریفی که طراحی ، ساخته ، تلفیق ، آزمایش و مستند شده است و customer value معتبری را ارائه می دهد. با این وجود برای داشتن یک چک لیست مفید ، این موارد کار در سطح بزرگتر باید بیشتر اصلاح شوند. مثلاً آزمایش یعنی چه؟ Unit تست شده؟ Integration تست شده؟ سیستم تست شده؟ سیستم عامل آزمایش شده است؟ بین‌المللی کردن تست شده؟ احتمالاً می توانید به بسیاری از اشکال دیگر آزمایش که مخصوص محصول شماست فکر کنید. آیا همه این انواع تست در تعریف انجام شده وجود دارد؟

بخاطر داشته باشید که اگر در هر اسپرینت تست مهمی را انجام ندهید (مثلاً تست عملکرد) ، مجبورید گاهی اوقات آن را انجام دهید. آیا قرار است در آینده اسپرینت تخصصی داشته باشید که تنها کاری که انجام می دهید تست عملکرد است؟ اگر بله ، در این صورت تست عملکرد برای "done شدن کار" ضروری است ، ولی شما در هر اسپرینت محصول بالقوه قابل عرضه ای که به صورت افزایشی کامل شده است ندارید و حتی بدتر از آن ، هنگامی که شما تست عملکرد را بعداً انجام دهید و کاملاً مطابق برنامه پیش نرود ، نه تنها در اواخر مرحله یک مشکل مهم کشف خواهید کرد ، بلکه باید زمان و پول بسیار بیشتری نیز صرف کنید برای رفع آن نسبت به اینکه اگر آزمایش عملکرد را زودتر انجام داده بودید.

بعضی اوقات تست ممکن است بیشتر از مدت زمان اسپرینت باشد. اگر این اتفاق بیفتد زیرا تیم توسعه بخش زیادی برای تست دستی به دست آورده است ، تیم باید آزمایشات خود را شروع کند تا آزمایش در یک اسپرینت انجام شود. اگر این امر به دلیل ماهیت آزمون رخ دهد ، ما باید شروع آزمون در یک اسپرینت و پایان آن در برخی از اسپرینت های آینده را بپذیریم. به عنوان مثال ، سازمانی که من در آن مربیگری می کردم ساخت دستگاهی متشکل از سخت افزار ، سیستم عامل و نرم افزار بود. یکی از تست های استاندارد آن تست سوختن 1500 ساعته بود که در آن مدت زمان دستگاه به طور مساوی اجرا شد تا ببیند آیا خراب است. این آزمون را نمی توان در یک اسپرینت دو هفته ای به پایان رساند ، بنابراین تیم Scrum تعریف انجام شده را تنظیم کرد تا حتی اگر آزمون 1500 ساعته هنوز به پایان نرسیده باشد ، یک اسپرینت انجام شده تلقی شود.

غالباً از من سال می شود ، "اگر نقص قابل توجهی در آخرین روز اسپرینت باقی بماند ، چه می شود. آیا آیتم مانده در backlog محصول done شده است؟ " نه ، این کار تمام نشده است! و چون قاعدتاً ، ما اسپرینت را تا انتهای timebox برنامه ریزی شده افزایش نمی دهیم ، اسپرینت را یک یا دو روز برای رفع نقص در اسپرینت فعلی تمدید نمی کنیم. در عوض ، در انتهای برنامه ریزی شده اسپرینت ، مورد ناقص مانده در backlog را از اسپرینت فعلی خارج می‌شود و بر اساس سایر مواردی که در حال حاضر در backlog محصول موجود هستند ، با ترتیب مناسب مجدداً به backlog محصول وارد می شود. پس از آن ممکن است مورد ناقص در اسپرینت بعدی به پایان برسد.

تیم های اسکرام باید یک تعریف قوی از done شده داشته باشند ، تعریفی که اطمینان بالایی را نسبت به آنچه که می سازند با کیفیت بالا و قابل عرضه است ، فراهم می کند. هر چیزی که بنا به صلاحدید خود فرصت تجاری عرضه را از سازمان سلب می کند و می تواند منجر به تعهد بدهی فنی شود (همانطور که در فصل 8 بحث خواهم کرد).

**Definition of Done Can Evolve Over Time**

شما می توانید تعریف انجام شده را به عنوان تعریف وضعیت کار در پایان اسپرینت انتخاب کنید. برای بسیاری از تیم های با عملکرد بالا ، وضعیت نهایی هدف کار آن را امکان پذیر می کند تا به طور بالقوه قابل عرضه باشد - و این حالت نهایی نسبت به چرخه عمر توسعه نسبتاً ثابت باقی می ماند.

به عنوان مثال ، زمانی که من در سال 2007 صاحب محصول برای پروژه طراحی مجدد وب سایت Scrum Alliance بودم ، ما اسپرینت های یک هفته ای را اجرا کردیم. حالت نهایی تعریف ما از done شده می تواند به صورت "live on the production servers" خلاصه شود. من و تیم تشخیص دادیم که این شرایط کاملاً معقول و منطقی است تا بتوانیم به هر اسپرینت برسیم. ما این وضعیت نهایی را در ابتدای مسیرتوسعه تعریف کردیم. در پایان زمانی که مالک محصول برای سایت بودم ، وضعیت نهایی هدف تغییر نکرد.

با این حال ، بسیاری از تیم ها کار خود را با تعریف done شده آغاز می کنند که در وضعیتی که تمام فیچرها در حدی که قابل عرضه شدن باشند به پایان می رسد. برای برخی ، موانع واقعی ممکن است مانع رسیدن آنها به این حالت در آغاز توسعه شود ، حتی اگر این هدف نهایی باشد. در نتیجه ، آنها ممکن است (لزوما) با وضعیت پایان کمتری شروع کنند و اجازه دهند با حذف موانع سازمانی ، تعریف done شدن آن‌ها به مرور تکامل یابد.

به عنوان مثال ، من از سازمانی بازدید کردم که سیستم انفورماتیک بالینی را ایجاد می کند. محصول آن در یک کلینیک پزشکی نصب شده و اطلاعات کلینیکی مختلفی را جمع آوری می کند (برخی حتی به طور مستقیم از دستگاه هایی که آزمایش های تشخیصی را انجام می دهند). این تیم می دانست که قبل از عرضه شدن، آزمایش بالینی که شامل نصب محصول در آزمایشگاه بالینی برای اطمینان از کار با سخت افزار بالینی است ، مورد نیاز است. با این حال ، از آنجا که آنها به طور منظم به آزمایشگاه دسترسی نداشتند ، تیم در ابتدا آزمایش بالینی را در تعریف خود انجام نداد. در عوض ، این شامل اسپرینت های آزمایش بالینی در پایان هر نسخه بود.

در بحث های ما ، من فهمیدم که بازاریابی و تیم از این آزمایشات بالینی قبل از انتشار متنفر هستند. هیچ کس نمی تواند پیش بینی کند که چند اسپرینت برای رفع تمام نقص ها لازم است و تا رفع نقص نمی توان محصول را آزاد کرد. همانطور که ما در حال ایده پردازی برای راه حل های بالقوه بودیم ، معاون فنی مهندسی در این زمینه توجیه کرد. وی از تیم خود پرسید ، "اگر به آزمایشگاه بالینی دسترسی داشتید ، آیا می توانید آزمایش های بالینی را در هر اسپرینت انجام دهید؟"

اعضای تیم در مورد سوال او بحث کردند و پاسخ دادند ، "بله ، اما این بدان معناست که ما در هر اسپرینت ویژگی های کمتری را تکمیل می کنیم." معاون فنی مهندسی موافقت کرد برای دسترسی تیم به آزمایشگاه بالینی دانشگاه محلی موانع را برطرف کند. صاحب محصول موافقت کرد که داشتن ویژگی های کمتری در هر اسپرینت منطقی است زیرا می داند ویژگی های ارائه شده از نظر بالینی آزمایش شده است. در آن مرحله تیم توانست تعریف خود را از done شده تکامل دهد تا در واقع به اصل "بالقوه قابل عرضه شدن" برسد و موجب افزایش اعتماد به نفس بالاتر افراد دخیل در کار در هر اسپرینت شود.

در مواقع دیگر ممکن است تیمی مانعی داشته باشد که می داند بلافاصله نمی توان آن را از بین برد. در نتیجه ، آنها می دانند که تعریف done شده در طول تلاش برای تولید محصول تکامل می یابد. نمونه متداول محصولی است که شامل سخت افزار و نرم افزار است. من دیده ام که اسکرام در توسعه بسیاری از این محصولات استفاده شده است و اغلب این نرم افزار را می شنوم که مردم می گویند: "سخت افزار همیشه دیر می رسد!" در مواردی از این دست ، اگر تیم در حال ساخت نرم افزار است و از سخت افزار واقعی برای آزمایش نرم‌افزار برخوردار نیست ، واقعاً نمی تواند ادعا کند که نتایج تولید شده در انتهای اسپرینت حداکثر قابل عرضه است. در بهترین حالت ممکن است ادعا کند "شبیه سازی انجام شده است" ، زیرا آزمایش در اوایل اسپرینت به طور معمول در برابر شبیه ساز نرم افزار از سخت افزار واقعی انجام می شود. بعداً ، وقتی سخت افزار واقعی در دسترس باشد ، تعریف done شده به معنای بالقوه قابل حمل یا حداقل چیزی نزدیک به آن است.

**Definition of Done versus Acceptance Criteria**

تعریف انجام شده مربوط به increment محصول در حین اسپرینت است. افزایش محصول از مجموعه ای از موارد باقیمانده در backlog محصول تشکیل شده است ، بنابراین هر مورد باقیمانده در backlog باید مطابق با کار مشخص شده در چک لیست تعریف done شده کامل شود.

همانطور که در فصل 5 بحث خواهم کرد ، هر مورد باقیمانده محصول که به اسپرینت وارد می شود باید دارای مجموعه ای از شرایط پذیرفته شدن (conditions of satisfaction) باشد (معیارهای پذیرش خاص کالا) ، که توسط صاحب محصول مشخص شده است. این معیارهای پذیرش (acceptance criteria) سرانجام در آزمونهای پذیرش تأیید می شود که مالک محصول تأیید می کند آیا مورد باقیمانده به شکل مورد نظر عمل می کند. به عنوان مثال ، اگر مورد باقیمانده محصول "مجاز بودن مشتری برای خرید با کارت اعتباری" باشد ، شرایط پذیرش ممکن است" کار با AmEx ، Visa و MasterCard "باشد. بنابراین هر مورد باقیمانده محصول مجموعه معیارهای پذیرش مناسب خود را خواهد داشت. این معیارهای خاص علاوه بر معیارهای done شده مشخص شده در لیست بررسی تعریف done شده است که برای همه موارد باقیمانده محصول اعمال می شود.

آیتم های باقیمانده محصول فقط زمانی قابل done شدن است که معیارهای پذیرش خاص آیتم (مثلاً "با همه کارتهای اعتباری کار کند") و تعریف done شده در سطح اسپرینت (مثلاً "روی سرور فعال باشد ") برآورده شده باشد.

اگر رجوع به موارد باقیمانده محصول(backlog) که معیارهای پذیرش خود را به صورت کامل ارضاء کرده است با عنوان done شده گیج کننده است ، آنها را تکمیل یا پذیرفته شده (completed / accepted.) صدا بزنید

**Done versus Done-Done**

برخی از تیم ها مفهوم done شده را در مقابل "done-done" را پذیرفته اند. به نوعی done-done قرار است بیشتر از done انجام شده باشد! تیم ها نباید به دو مفهوم متفاوت احتیاج داشته باشند ، اما من باید بپذیرم که هم از اصطلاحات با پسرم و هم از مشق شب استفاده می کنم. من از پسرم می پرسیدم که آیا او با تکالیفش "تمام شده" است یا خیر و او به من می گوید بله. سپس من به عنوان پدر و معلم به مدرسه او رفتم و هنگام بحث و گفتگو با معلم او پرسیدم: "بنابراین ، وقتی او تکالیف خود را تحویل می دهد ، آیا این کار done می شود؟" او گفت ، "نه واقعاً!"

بعد از یک بحث بیشتر با پسرم ، فهمیدم که تعریف او از انجام شده این بود که "من به همان اندازه که آمادگی انجام کار را داشتم کار کردم!" بنابراین ، از آن مرحله به بعد من شروع به استفاده از اصطلاح "done-done" کردم ، که هر دو توافق کردیم به معنای "انجام کار تا جایی که معلم شما فکر کند شما تمام کرده اید" است.

تیم هایی که عادت ندارند کارها را زود انجام دهند و غالباً بیشتر از کارهای done-done شده به عنوان عصا استفاده می کنند. برای آنها ، استفاده از done-done شده به این نکته اشاره می کند که done شده (انجام کار به همان اندازه که آمادگی انجام کار را دارند) وضعیت متفاوتی از کار done-done شده است (انجام کار مورد نیاز برای اینکه مشتری باور کند که انجام شده است). تیم هایی که داخلی سازی کرده اند تنها در صورتی done می شود که تمام کارهای لازم برای جلب رضایت مشتری را انجام داده باشید ، نیازی به دو حالت ندارند. برای آنها ، done به معنای done-done است!

**Closing**

در این فصل من بر نقش حیاتی اسپرینت در چارچوب اسکرام تأکید کردم. Sprints اسکلت ضروری Scrum را که بیشتر فعالیت ها و مصنوعات دیگر می توان بر روی آن قرار داد فراهم می کند. اسپرینت ها کوتاه ، دارای جعبه زمان و از نظر طول مدت سازگار هستند. آنها به طور معمول با یک sprint goal تعریف می شوند ، هدفی که نباید بدون داشتن دلیل اقتصادی مناسب تغییر یابد. اسپرینت ها باید محصول بالقوه قابل عرضه تولید کنند که مطابق با تعریف توافق شده done شده کامل شود. در فصل بعدی ، من به ورودی های اسپرینت – نیازمندی ها و نمایش مشترک آنها ، user story توجه خواهم کرد.

**فصل ۹ product owner نیمه اول ---------------------- مهدی**

مالک محصول

در این فصل شرح نقش مالک محصول را توضیح می دهم. من با توضیح هدف این نقش نسبت به سایر نقش های اسکرام شروع می کنم. سپس جزئیات اصلی مسئولیت ها و ویژگی های صاحب محصول را شرح می دهم. بعد من "روزی از زندگی" صاحب محصول را در طول چند هفته ارائه می دهم. سپس بحث می کنم که چه کسی باید برای انواع مختلف توسعه محصول، صاحب محصول باشد. من در پایان با توصیف چگونگی ترکیب نقش صاحب محصول با نقشهای دیگر و چگونه می توان آن را در تیم صاحب محصول مقیاس بندی کرد.

بررسی اجمالی

صاحب محصول نقطه مرکزی قدرتمند رهبری محصول است. این یکی از سه نقش همکاری است که هر تیم Scrum را تشکیل می دهد (دیگر نقش ها ScrumMaster و تیم توسعه هستند).

صاحب محصول باید حداقل در دو جهت به طور همزمان نگاه کند (شکل 9.1 را ببینید).

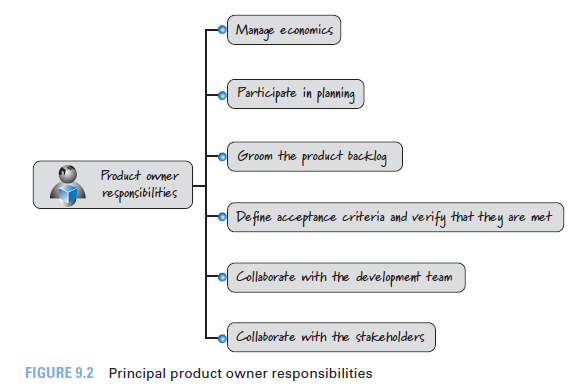
از یک طرف ، صاحب محصول باید نیازها و اولویت های ذینفعان سازمانی ، مشتریان و کاربران را به خوبی درک کند تا بتواند به عنوان صدای آنها عمل کند. از این نظر صاحب محصول به عنوان مدیر محصول اطمینان حاصل می کند که راه حل صحیحی ارائه شده است.

از طرف دیگر ، مالک محصول باید با تیم توسعه دهنده آنچه را که باید بسازد و ترتیب ساخت آن را اعلام کند. صاحب محصول همچنین باید اطمینان حاصل کند که معیارهای پذیرش ویژگی ها مشخص شده و بعداً آزمایش هایی که این معیارها را تأیید می کند برای تعیین کامل بودن ویژگی ها انجام می شود. مالک محصول تست های سطح جزئیات(منظورش خیلی جزعی) را نمی نویسد اما اطمینان حاصل می کند که سطح بالایی(کلیات تست های ضروری) نوشته شده است تا تیم بتواند تعیین کند که صاحب محصول ویژگی را کامل می داند. از این نظر صاحب محصول بخشی از تجزیه و تحلیل کسب و کار(بیزینس انالیز) و آزمایش کننده(تستر) است.

مسئولیت های اصلی

شکل 9.2 مسئولیت های اصلی صاحب محصول را نشان می دهد.

این مشخصاً یک نقش تمام وقت با مسئولیت های قابل توجه است. در حقیقت ، وقتی توضیحاتی را که در زیر می خوانید ، بخوانید فکر می کنید که مسئولیت رسیدگی به همه این مسئولیت ها یا داشتن تمام ویژگی های لازم برای موفقیت در این نقش برای یک شخص عملی نیست. در بیشتر موارد ، یک شخص می تواند و باید نقش صاحب محصول را پر کند. با این حال ، تحت شرایط خاص ، تیم های صاحب محصول یا پروکسی های صاحب محصول ممکن است عملی باشند. هر دو مفهوم بعداً در فصل بیان خواهد شد.



مدیریت اقتصاد

صاحب محصول مسئول این اطمینان است که تصمیمات اقتصادی خوبی به طور مداوم در سطح انتشار(ریلیز) ، sprint و سطح product backlog محصول گرفته می شود (شکل 9.3 را ببینید).

اقتصاد در سطح انتشار(ریلیز)

در سطح انتشار(ریلیز) ، مالک محصول به طور مداوم در زمینه(scope) ، تاریخ ، بودجه و کیفیت، trade-off انجام میدهد که از نظر اقتصادی مهم هستش در طول توسعه محصول وجود دارد. معاملات انجام شده(ترید-عاف) در ابتدای انتشار ممکن است دیگر در صورت وجود اطلاعات جدیدی که هنگام انتشار به دست می آید ، مناسب نباشد.

به عنوان مثال ، اگر چند هفته به یک تلاش 6 ماهه و با تاریخ ثابت برای توسعه فرصت پیدا کنیم ، اگر یک هفته اضافی (4٪ (schedule slip)ورقه برنامه) یک ویژگی اضافی را برای اضافه کردن به نسخه اختصاص دهیم ، 50٪ درآمد را افزایش می دهیم؟ آیا باید یک هفته وقت و هزینه اضافی برای درآمد اضافی معامله کنیم؟ صاحب محصول بر این تصمیم نظارت می کند. در بسیاری از موارد او می تواند یک جانبه تصمیم بگیرد. در مواقعی دیگر ممکن است صاحب محصول تصمیمی را توصیه کند اما همچنان برای اطمینان از ورودی آنها (و در بعضی مواقع تأیید) برای اجرای تصمیم با دیگران کار می کند.

همچنین ، در پایان هر sprint، صاحب محصول بر تصمیم گیری در مورد تأمین بودجه sprint بعدی نظارت می کند که تامین کند یا نه. اگر پیشرفت خوبی در جهت هدف ریلیز حاصل شود یا در صورت دیگری که sprint بعدی توجیه اقتصادی داشته باشد ، sprint بعدی تأمین می شود. اگر پیشرفت ضعیفی حاصل شود یا اقتصاد نصرفد ، ممکن است تلاش لغو شود.

یک product owner راضی اگر ببینه که محصول اماده اراعه هستش،میتونه تصمیم بگیره که بودجه رو برای توسعه بیشتر ادامه بده یا نه.

به عنوان مثال ، بگذارید بگوییم ما برای ده sprint برنامه ریزی کردیم. پس از sprint 7 ، صاحب محصول موارد باقیمانده محصول باقیمانده را بررسی می کند و نتیجه می گیرد که هزینه ایجاد آن موارد بیشتر از ارزشی است که آنها تحویل می دهند. صاحب محصول ممکن است به این نتیجه برسد که اراعه زودهنگام محصول به جای ادامه کار با طرح اولیه sprint ، از نظر اقتصادی منطقی تر است. این انعطاف پذیری برای تحویل زودهنگام با اطمینان از اینکه ابتدا روی کالاهای با ارزش بالاتر در بالای محصول کار می شود ، کار می شود و تیم در حال انجام هر sprint مطابق با تعریف قوی انجام شده است.

و البته صاحب محصول ممکن است به این نتیجه برسد که ما باید در پایان sprint یکبار دیگر بودجه را متوقف کنیم زیرا خصوصیات اقتصادی اصلی تغییر کرده است. به عنوان مثال ، اگر ما در حال ایجاد یک محصول خاص برای یک کشور هستیم و یک آژانس نظارتی در آن کشور قوانین خود را تجدید نظر می کند ، فروش محصول را برای ما بی سود یا حتی غیرقانونی می کند، چه میشود؟ در مواردی از این دست ، یک مالک محصول ممکن است بر لغو تلاش برای توسعه نظارت کند حتی اگر اوضاع در غیر این صورت خوب پیش برود.

اقتصاد در سطح Sprint

علاوه بر اقتصاد در سطح انتشار(ریلیز) ، صاحب محصول ، اقتصاد در سطح sprint را نیز مدیریت می کند ، اطمینان حاصل کند که از هر sprint بازده خوبی برای سرمایه گذاری ارائه می شود. صاحبان کالاهای خوب با پول سازمان خود طوری رفتار می کنند که گویی پول خودشان است. در بیشتر موارد صاحب محصول از هزینه sprint بعدی اطلاع دارد (مدت زمان و ترکیب تیم در این sprint مشخص است). با این دانش ، مالک محصول باید در برنامه ریزی sprint از خود سوال کند ، "آیا می توانم از حساب بانکی خودم چک بکشم که برابر با هزینه این sprint باشد تا ویژگی هایی را که می خواهیم در این sprint بسازیم بدست آورم؟" اگر پاسخ منفی باشد ، یک مالک خوب محصول هم پول سازمان را خرج نمی کند.

اقتصاد در product backlog

همانطور که در فصل 6 بحث کردم ، مالک محصول وظیفه دارد اولویت Product Backlog را تعیین کند. وقتی شرایط اقتصادی تغییر می کند ، اولویت های موجود در Product Backlog نیز احتمالاً تغییر خواهد کرد.

به عنوان مثال ، بگذارید بگوییم که در ابتدای انتشار(ریلیز) ، صاحب محصول معتقد است که یک ویژگی برای درصد زیادی از کاربران هدف ارزشمند است و تیم معتقد است که فقط یک تلاش متوسط برای ایجاد آن لازم است. با این حال ، پس از چند sprint ، تیم کشف کرد که تکمیل این ویژگی به تلاش زیادی احتیاج دارد و فقط برای بخشی از کاربران هدف ارزشمند است. از آنجا که نسبت هزینه / سود این ویژگی به طرز چشمگیری تغییر کرده است ، صاحب محصول باید product backlog محصول را مجدداً درجه بندی کند تا این دانش را منعکس کند - شاید با انداختن موارد product backlog مرتبط با این ویژگی.

شرکت در برنامه ریزی

صاحب محصول ، یک شرکت کننده اصلی در فعالیت های برنامه ریزی نمونه کارها(portfolio) ، محصولات ، انتشار(ریلیز) ، و sprint است. در طول برنامه ریزی نمونه کارها (نگاه کنید به فصل 16) ، مالک محصول با ذینفعان داخلی (شاید کمیته تأیید یا هیئت حاکمه) کار می کند تا محصول را به طور صحیح در مجموعه سهام نمونه قرار دهد و زمان شروع و خاتمه توسعه محصول را تعیین کند. در طول برنامه ریزی محصول (نگاه کنید به فصل 17) ، مالک محصول با سهامداران همکاری می کند تا محصول را تصور کند. در طول برنامه ریزی برای انتشار(ریلیز) (نگاه کنید به فصل 18) ، مالک محصول با سهامداران و تیم برای تعریف محتوای نسخه بعدی کار می کند. در طول برنامه ریزی sprint (نگاه کنید به فصل 19) ، صاحب محصول با تیم توسعه همکاری می کند تا هدف sprint را تعیین کند. وی همچنین ورودی ارزشمندی را فراهم می کند که تیم توسعه را قادر می سازد مجموعه ای از موارد باقیمانده محصول) product backlog) را که تیم می تواند واقعاً در پایان sprint ارائه دهد ، انتخاب کند.

Groom the Product Backlog

صاحب محصول بر ارایش پروداکت بک لاگ نظارت می کند ، که شامل ایجاد و پالایش ، برآورد و اولویت بندی پروداکت بک لاگ است (نگاه کنید به فصل 6). صاحب محصول شخصاً همه کارهای ارایش و grooming را انجام نمی دهد. به عنوان مثال ، ممکن است او همه پروداکت بک لاگ را ننویسد. دیگران ممکن است به او کمک کنند. صاحب محصول نیز موارد را تخمین نمی زند (تیم توسعه دهنده چنین کاری می کند) اما در هنگام برآورد برای سوالات و تخمین زمانی در دسترس است. صاحب محصول ، در نهایت ، مسئول اطمینان از انجام فعالیت های grooming به گونه ای است که جریان روان ارزش محصول ارائه شده را ارتقا بخشد

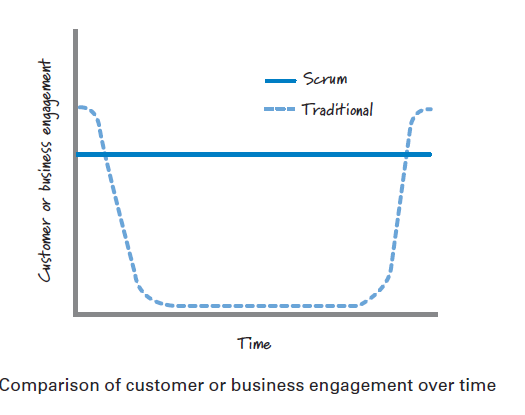
Define Acceptance Criteria and Verify That They Are Met

صاحب محصول وظیفه دارد acceptance criteria را برای هر ایتم از پروداکت بک لاگ تعریف کند. این شرایطی است که در آن صاحب محصول از برآورده شدن الزامات عملکردی و غیر عملکردی راضی خواهد بود. صاحب محصول همچنین ممکن است acceptance tests متناسب با acceptance criteria را بنویسد ، یا می تواند از متخصصان موضوع (SME) یا اعضای تیم توسعه کمک بگیرد. در هر صورت ، صاحب محصول باید اطمینان حاصل کند که این acceptance criteria (و acceptance tests غالباً خاص) قبل از بررسی مورد در جلسه اسپرینت پلنینگ ایجاد شده است. بدون آنها ، تیم درک کاملی از ایتم ها ندارند و آماده نیستند که آن را در یک اسپرینت اماده کنند. به همین دلیل ، بسیاری از تیم های اسکرام وجود acceptance criteria واضح را به عنوان ماده ای در لیست چک تعریف خود (definition-of-ready checklist () قرار داده اند (به فصل 6 مراجعه کنید)

صاحب کالا در نهایت مسئول تأیید acceptance criteria شده و یک بار دیگر ، صاحب کالا ممکن است خود تصمیم به انجام acceptance tests داشته باشد یا ممکن است از کمک کاربران خبره برای تأیید پروداکت بک لاگ استفاده کند و تایید اینکه این پروداکت بک لاگ شرایط پذیرش را میت میکنه. این تیم ممکن است به ایجاد زیرساخت آزمایشی کمک کند که مالک محصول یا SME های خاص را قادر به اجرای این موارد کند آزمایشات با کارآیی بیشتری انجام می شود ، اما صاحب محصول باید قاضی نهایی در مورد اینکه آیا یک مورد انتظارات را برآورده می کند ، باشد. برای مالک محصول مهم است که معیارهای پذیرش را در هنگام اجرای اسپرینت بررسی کند نه اینکه منتظر بازبینی اسپرینت باشد. با انجام برخی از آزمایش ها با تکمیل ویژگی ها ، صاحب محصول می تواند اشتباهات و سو تفاهماتی را که تیم می تواند قبل از بررسی سرعت انجام دهد ، شناسایی کند. همچنین ، از آنجا که تیم مجاز است فقط ویژگی های تکمیل شده را هنگام بازبینی نشان دهد ، مالک محصول باید اطمینان حاصل کند که acceptance tests قبل از بررسی انجام می شود تا تیم بداند کدام ویژگی ها واقعاً تعریف انجام شده (definition of done) را دارند.

**Collaborate with the Development Team** (همکاری با تیم توسعه)

صاحب محصول باید بطور مکرر از نزدیک با تیم توسعه همکاری کند. صاحب محصول ، نقشی درگیر ، متعهد و روزمره است. بسیاری از سازمان ها که تازه شروع به پذیرش اسکرام می کنند ، در ایجاد تعامل کافی صاحب محصول با تیم توسعه موفق نیستند ، بازخورد اساسی را به تأخیر می اندازند و اینکار ارزش آن بازخورد را در صورت دریافت به میزان قابل توجهی کاهش می دهد. این عدم تعامل می تواند زمانی نیز اتفاق بیفتد که افراد تازه وارد نقش صاحب محصول تصور می کنند میزان مشارکت آنها هنگام استفاده از اسکرام باید با مشارکت آنها در طول توسعه مبتنی بر فاز شباهت داشته باشد. شکل زیر سطح معمولی تعامل مشتری در یک روند توسعه ترتیبی را با آنچه از صاحب محصول هنگام استفاده از اسکرام انتظار می رود مقایسه می کند. با استفاده از توسعه سنتی و مبتنی بر فاز ، الگوی تعامل شبیه منحنی U شکل یا وان شکل است. در ابتدا مشتریان به طور قابل توجهی درگیر مقدمات کار برای کمک به تعریف مجموعه کامل نیازها هستند. هنگامی که تلاش به مراحل فنی بیشتری (مانند طراحی ، کدنویسی و انواع خاصی از تست ها) منتقل شد ، دیگر "مشتری" مورد نیاز نیست. به همین ترتیب ، میزان مشارکت آنها در اکثر تلاش ها کاملاً پایین است یا وجود ندارد. در واقع ، در طول توسعه سنتی ، مشتریان تا زمانی که مجبور به انجام تست پذیرش کاربر بر روی آنچه که ساخته شده نباشند ، مجدداً وارد فرآیند نمی شوند. چیزی که مشتریان در این مرحله به طور معمول کشف می کنند این است که آنچه ساخته شده دقیقاً همان چیزی نیست که آنها می خواستند. برای اینکه اوضاع را بدتر کنیم ، برای آنها معمولاً خیلی دیر یا خیلی گران است که تغییرات را انجام دهند - حداقل در این نسخه. مشتریانی که در عوض انتظار خوشحال شدن دارند ، متعجب و ناامید شده و این زمانی است که انگشت اتهام به سمت یکدیگر می گیرند. مشتری ادعا می کند ، "اگر شما بچه ها سند نیاز من را با دقت بیشتری مطالعه می کردید ، آنچه را که واقعاً می خواستم می ساختید" و تیم توسعه پاسخ می دهد ، "خوب ، اگر سند خود را با وضوح بیشتری می نوشتید ، ما چیز دیگری می ساختیم . ما خواسته های شما را ساختیم! " با استفاده از Scrum ، ما یک ویژگی را همزمان می سازیم ، نه فاز را همزمان. این بدان معنی است که ما تمام فعالیت ها را برای ایجاد یک ویژگی خاص (طراحی ، کد ، یکپارچه سازی ، آزمایش) در طول یک سرعت انجام می دهیم. بنابراین ، سطح بالایی از تعامل ثابت توسط صاحب محصول ضروری است. با چنین تعامل نزدیک در تکرارهای کوتاه و تایم باکس ، احتمال قطع ارتباط صاحب محصول و تیم توسعه به مراتب کمتر است. یک مزیت ثانویه این است که هنگام انجام اسکرام انگشت اتهام به سمت یکدیگر نمی گیرند!



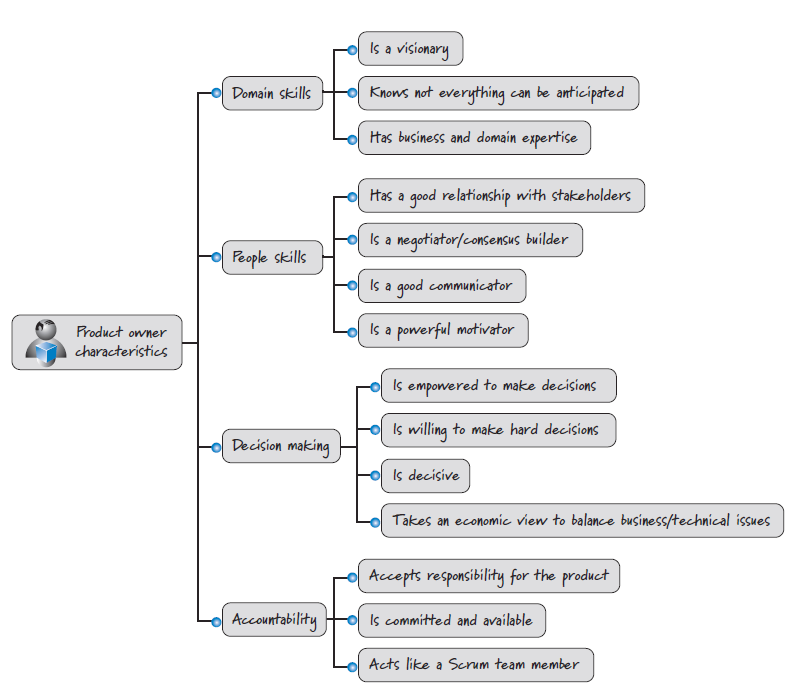
**Collaborate with the Stakeholders** (همکاری با مشتریان)

صاحب محصول یک صدای واحد برای همه جامعه ذینفعان محصول است. هم ذینفعان داخلی هم خارجی. ذینفع داخلی (**Internal stakeholders**) می توانند شامل صاحبان سیستم های تجاری، مدیر اجرا، مدیر برنامه بازاریابان و فروشندگان باشد. ذینفع خارجی (**External stakeholders**) می تواند شامل مشتریان، کاربران، شرکا، نهادهای نطارتی و دیگران باشد.صاحب محصول می بایست با جامعه همه ذینفعان کاملا نزدیک کار کند تا ورودی (اطلاعات) جمع آوری کند و یک چشم اندازه برای هدایت روند توسعه محصول را بسازد. اگر صاحب محصول بیش از حد تحت فشار قرار بگیرد برای او سخت خواهد بود که هم با تیم توسعه و هم با ذینفعان در حد مورد نیاز همکاری داشته باشد. در برخی شرایط ممکن است حجم و بارکاری بیش از این باشد که یک نفر بتواند ان را انجام دهد. در این شرایط صاحب محصول ممکن است از دیگران کمک و همکاری بگیرد یا بتواند مسئولیت های نقش خود را کاملا انجام دهد. به این مورد در بخش تیم صاحب محصول (product owner team) بیشتر می پردازیم.

**Characteristics/Skills**

شکل زیر مشخصات مهم نقش product owner را نشان میدهد. اگرچه که ویژگی های بی شماری باید یک صاحب محصول خوب داشته باشد اما این ویژگی ها در 4 دسته قابل تقسیم است:

domain skills, people skills, decision making, accountability



**Domain Skills**

صاحب محصول فردی است که می تواند چشم اندازه کلی محصول را طراحی کرده و باعث شود تیم توسعه بتوانند به ان چشم انداز (دید کلی از محصول) دست یابند. دید داشتم به این معنا نیست جزئیات این چشم اندازه و راه های دستیابی بع آن دقیقا مشخص و واضح باشد. یک صاحب محصول خوب می داند که همه چیز از قبل قابل پیش بینی نیست و مشتاق است در برابر تغییرات سازگار باشد.

صاحب محصول برای اینکه در ایجاد و اجرای دیدگاه و چشم انداز موثر باشد باید دانش مناسبی در حوزه تجاری و دامنه کار داشته باشد (domain & business knowledge). این دشوار است که یک صاحب محصول خوب باشید اگر در حوزه و دامنه محصول تازه کار و نااشنا باشید. و چگونه می توانید در میان ویژگی های زیاد اولویت ها را تعیین کنید اکر که موضوع اصلی محصول را ندانید؟

**People Skills** (مهارت های افراد)

فرد صاحب محصول همچنین باید صدای مشتری باشد که این کار یک رابطه خوب با سهامداران و ذیففعان را می طلبد. زیرا اغلب چندین مشتری وجود دارد که ممکن است نیاز های متناقض داشته باشند، صاحب محصول باید یک مذاکره کننده و توافق کننده خوب نیز باشد. صاحب محصول حلقه اتصال بین جامعه ذینفعان (مشتریان) و سایر اعضای تیم اسکرام (توسعه دهندگان) است. در این موقعیت صاحب محصول به مهارت های ارتباطی خوب نیاز دارد تا بتواند به زبان مناسب با هر دو گروه (تیم توسعه و مشتریان) تعامل داشته باشد و اطلاعات را به آنها انتقال دهد. یک ارتباط گیرنده خوب همچنین خصوصیات زیر را باید داشته باشد: مایل به صحبت کردن است حتی اگر این کار خلاف وضعیت موجود باشد.() ، نسبت به ایده های خود اعتماد به نفس و اطمینان دارد، نسبت به موضوع دانش و اگاهی دارد، می تواند به سادگی و راحت ارتباط برقرار کند، راه بلد و مورد اعتماد و معتبر است. به علاوه صاحب محصول یک انگیزه دهنده قوی است. زمانی که کار سخت پیش میرود صاحب محصول می تواند به افراد یاداور شود که چرا آنها تلاش می کنند و به مردم کمک می کنند (یه چیز چرتیم ادامش بود ک نفهمیدم البته چیز مهمی نیست.)

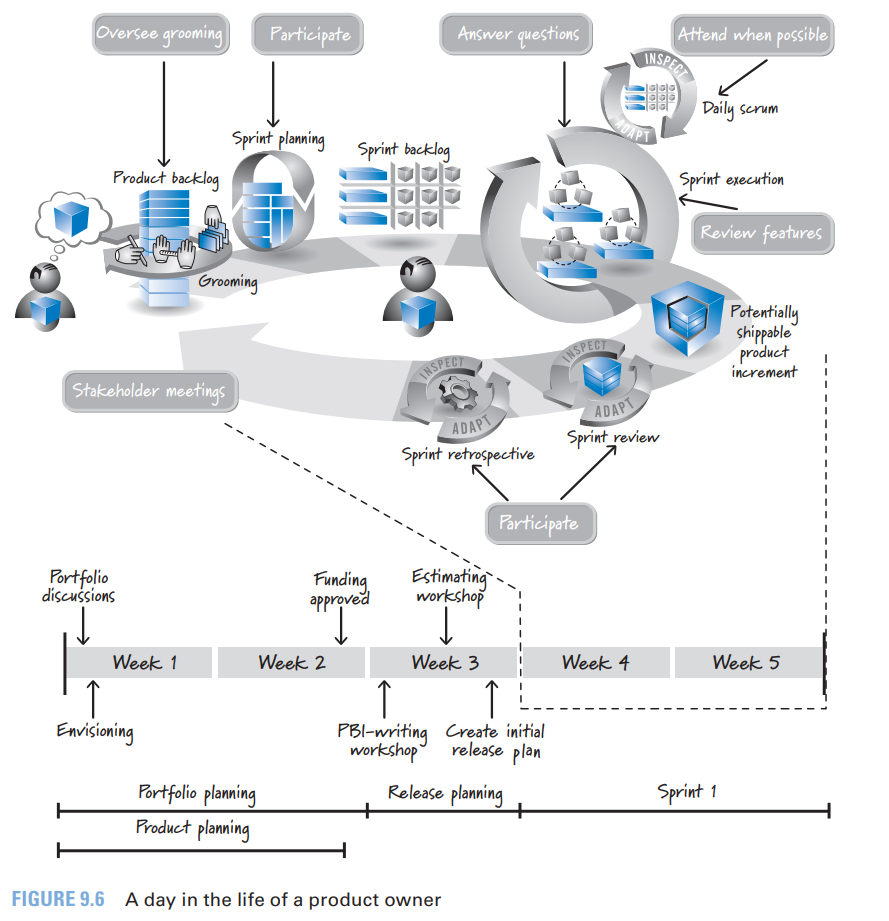
**Decision Making** (تصمیم گیری)

صاحب محصول باید برا تصمیم گیری توانمند باشد. یک مانع مکرر برای سازمان هایی که تازه با اسکرام اشنا شده اند و در استفاده از آن تازه کار هستند این است که فردی انتخابی به عنوان صاحب محصول در تصمیم گیری های مهم توانمند نیست. چنین فردی صاحب محصول نیست. به علاوه صاحب محصول باید مشتاق باشد که تصمیمات جدی و دشوار بگیرد حتی گاها با درنظر گیری trade off ای که بین محدودیت ها در حوزه هایی مثل زمان و بودجه وجود دارد. این تصمیمات باید به موفع گرفته شوند و نباید بدون دلیل خوب و قانع کننده ای معکوس شوند. به عبارت دیگر صاحب محصول باید یک تصمیم گیرنده قاطع باشد. در گرفتن این تصمیمات یک صاحب محصول باید تعادل مناسب بین نیازهای بیزینسی و واقعیت های فنی را حفظ و برقرار کند. اگرچه که تیم اسکرام به همگی زمانی که سیستم دچار یک سطح غیرقابل پیش بینی از بدهی های فنی میشود مسئول هستند اما تصمیمات صاحب محصول و ان دسته از از تصمیاتی که تاثیرات سطح سیستم را در نظر نمی گیرند می تواند مکررا یک عامل موثر و قابل توجه باشد.

**Accountability** (پاسخگویی)

صاحب محصول باید نتایج خوب تجاری را ارائه دهد و در قبال اینکار مسئول است. این پاسخگویی صاحب محصول دلیل نمی شود که سایر اعضای تیم اسکرام در قبال تلاش برای تولید و ارائه یک نتیجه خوب در ازای سرمایه گزاری که صورت گرفته پاسخگو و مسئول نباشند. با این حال صاحب محصول مجبور است که مطمئن شود منابعی که در ضمن توسعه صرف می شوند از نظر اقتصادی معقو هستند و اگر این چنین نباشد باید مسئولیت این کار را بپذیرد. در انتها صاحب محصول فرصت های زیادی دارد که product backlog را تغییر دهد، اولیت ها را مجددا تنظیم کند یا حتی کاملا روند توسعه رو لغو کند. صاحب محصول باید متعهد و در دسترس باشد برای هم سهامداران و مشتریان و هم برای سایر اعضای تیم اسکرام. صاحب محصول بودن یک شغل تمام وقت است و تلاش پاره وقت برای آن منجر به شکست می شود. در نهایت صاحب محصول یک عضو از تیم اسکرام است و بنابراین متوجه است که نتایج خوب اقتصادی بدون تلاش همکارانه همه اعضای تیم اکرام غیر ممکن است. بنابراین ضاحب محضول با اعضای تیم توسعه و اسکرام مستر با احترام رفتار می کند و اعتماد می کند که آن ها در ارائه نتایج مطلوب شریک هستند. همه اعضای تیم اسکرام باهم باید یک تفکر تفنگدار (should share a Musketeer attitude) داشته باشند (گفته این اصطلاحو بعدا توضیح میدم تو فصل 11 ). و هیچ نگرش من در برابر بقیه ای نباید وجود داشته باشد (منم منم نداریم و منافع همه مشترکه). صاحب محصول، اسکرام مستر و تیم توسعه یک مجموعه واحد هستند که باهم برای تحقق یک هدف مشترک کار می کنند.

**فصل ۹ product owner نیمه دوم ---------------------- حمید**

بعد از اتمام مرحله ی Product Planning، محصول پیشنهادی به Portfolio Planning ارائه می شود، که در آنجا از فیلتر اقتصادی سازمان گذر داده شود و مشخص شود که آیا روی توسعه ی محصول سرمایه گذاری می شود یا خیر و چه زمانی کار می تواند شروع شود. در تصویر 9.6 اینطور نمایش داده شده که همه ی این مراحل بلافاصله پس از مرحله ی Product Planning انجام می شود؛ با اینحال در بسیاری از سازمان ها، بین زمان اتمام مرحله ی Envisioning و زمانی که کمیته ی تصویب کننده یا هیئت نظارت (governance board)، سرمایه گذاری را بررسی و تایید کند و کار شود، تاخیری وجود خواهد داشت.

Product owner در هفته ی سوم به برنامه ریزی انتشار اولیه ( initial release planning) می پردازد (فصل 18). در این مرحله معمولا PBI-writing workshop (story-writing) برای ذینفعان (Stakeholders) داخلی، اعضای تیم، و در صورت امکان ذینفعان خارجی انجام می شود تا یک Product Backlog سطح بالا تولید شود که بتوان در مرحله ی برنامه ریزی انتشار (Release planning) از آن استفاده کرد. چون قبل از این مرحله سرمایه گذاری تایید شده است، اعضای تیم توسعه باید در دسترس باشند و بتوانند در تیم شرکت کنند. در صورت ضرورت و در صورتی که هنوز تیم توسعه تشکیل نشده می توان از یک تیم جایگزین استفاده کرد.

پس از PBI-Writing Workshop، مالک محصول در Estimating Workshop مشارکت می کند (احتمالا شامل یک مجموعه ای از جلسات طی یک یا دو روز می شود) که طی آن اعضای تیم توسعه دهنده (یا تیم جایگزین اگر هنوز تیم اصلی تشکیل نشده) اندازه و حجم موارد پر ارزش Product Backlog را حدس می زنند.

در مرحله ی بعد، مالک محصول به تسهیل و تشکیل جلسه ای برای طرح اولیه ی برنامه ریزی انتشار (release-planning) می پردازد (برنامه ریزی بلند مدت تر). چون از قبل بعضی از آیتم های Product Backlog تخمین زده شده اند، تمرکز این مرحله بر اولویت بندی Product Backlog و ایجاد تعادل بین قیدهای حوزه (وسعت)، جدول زمانی و بودجه خواهد بود (فصل 18). مشارکت کنندگان اصلی این activity، ذینفعان هستند؛ با اینحال همه یا برخی از اعضای تیم توسعه دهنده نیز باید در مقطعی، در این فعالیت مشارکت داشته باشند تا بتوانند وابستگی های تکنیکالی که ممکن است بر روی ترتیب آیتم های Product Backlog تاثیر بگذارد را شناسایی کنند.

هدف از این مرحله این است که به قدر کافی در release planning پیش رفته باشیم که به سطح قابل قبولی از وضوح در مورد انتشار کلی (overall release) رسیده باشیم و به سوالات تجاری یک پاسخ اولیه داده شده باشد مثلا سوالاتی مثل اینکه محصول نهایی که تحویل می شود چه خواهد بود و چه زمانی تحویل داده می شود. این activity برای اغلب محصولات نباید بیشتر از یک یا دو روز طول بکشد. همانطور که در فصل 18 اشاره خواهم کرد، release planning یک activity مداوم و ادامه دار (ongoing) خواهد بود و به همین دلیل، در این مرحله نباید سرمایه گذاری بیش از حدی برای رسیدن به دقت زیاد انجام دهیم؛ بلکه با به دست آوردن اطلاعات بهتر در طی توسعه، نقشه ی انتشار (Release Plan) را به روز رسانی می کنیم.

پس از release planning ، تیم اسکرام، اولین sprint را انجام می دهد (شکل 9.6 یک اسپرینت دو هفته ای در هفته های 4 و 5 را نشان می دهد). در ابتدای sprint، مالک محصول، اکتیویتی برنامه ریزی اسپیرینت (Sprint-Planning) را بررسی می کند (به فصل 19 مراجعه کنید). در حین اجرای sprint (نگاه کنید به فصل 20) ، مالک محصول سعی می کند تا در اسکرام های روزانه ی تیم شرکت کند، این کار ممکن است همیشه امکان پذیر نباشد، اما روش و عادت خوبی است. در طی اسکرام روزانه ، مالک محصول سعی می کند درک کند که روند فعلی پیشرفت sprint به چه صورت است و راه ها و فرصت های کمک به تیم توسعه را شناسایی می کند. مثلا شاید یکی از اعضای تیم ذکر کند که ابهاماتی در مورد جزئیات یکی از آیتم های product backlog دارد و قبل از انجام Task فعلی خود به توضیحاتی نیاز داد. اگر توضیحات کوتاه باشند، مالک محصول ممکن است توضیحات آن را در جلسه ی طی اسکرام روزانه ارائه دهد. ولی اگر پاسخ، یک پاسخ بیش از چند ثانیه ای باشد، بهتر است که مالک محصول بگوید "خوشحال خواهم شد که بعد از اسکرام روزانه، آن را با شما در میان بگذارم"

مالک محصول همچنین باید (به طور معمول هر روز) در دسترس باشد تا به سوالاتی که ممکن است ایجاد شود پاسخ دهد و با قابل بررسی شدن امکانات و ویژگی ها (features)، به تست آن ها بپردازد. اگر مالک محصول بداند که برای انجام این مسئولیت ها نمی تواند هر روز در دسترس باشد، باید این مسئولیت ها را به یک شخص مناسب محول کند تا تیم توسعه به بن بست نخورد.

مالک محصول همچنین در حین اجرای Sprint با ذینفعان داخلی و خارجی ملاقات می کند تا از صحیح بودن اولویت های Sprint های آینده اطمینان پیدا کند و ورودی های با ارزش کاربران را که بر ویژگی های انتخاب شده برای Sprint های آینده تأثیر می گذارد محفوظ کند.

مالک محصول همچنین به پیرایش مکرر Product Backlog می پردازد. که این کار شامل نوشتن آیتم های جدید برای product backlog و تصحیح و پالایش موارد موجود و سپس کار با تیم برای تخمین زدن آنها و کار با تیم و ذینفعان برای اولویت بندی آنها می شود.

در پایان Sprint، مالک محصول در دو فعالیت end-of-sprint و inspect-and-adapt شرکت می کند: مرور Sprint (Sprint Review) (فصل 21) و گذشته نگر (!!) Sprint (Retrospective Sprint) (فصل 22). بعد از انجام این موارد، چرخه Sprint تکرار می شود و مالک محصول در اکتیویتی Sprint Planning بعدی شرکت می کند.

# **مالک محصول باید چگونه شخصی باشد؟**

اکثر سازمان های غیر اسکرام احتمالاً نقشی با عنوان "مالک محصول" نخواهند داشت. پس چه کسی در داخل سازمان باید نقش مالک محصول را پر کند؟

همانطور که قبلاً در این فصل اشاره کردم، مالک محصول باید همزمان با دو سمت روبرو شود: سمت ذینفعان داخلی و خارجی و سمت تیم توسعه. بنابراین، نقش مالک محصول ترکیبی از اقتدار و مسئولیت هایی است که به طور تاریخی در چندین نقش سنتی یافت می شوند. در فراگیرترین بیان آن، مالک محصول، عناصری از نقش مدیر محصول (product manager)، بازاریاب محصول (Product Marketer)، مدیر پروژه (در فصل 13 بیشتر بحث شده) و تحلیلگر تجارت (Business Analyst)، و تست کننده ی پذیرش (Acceptance Tester) را در خود جای داده است.

اینکه دقیقاً چه کسی باید مالک محصول باشد به نوع محصول تحت توسعه و سازمان بستگی دارد. جدول زیر برای انواع مختلف توسعه، نامزدهای مناسب برای نقش مالک محصول را پیشنهاد می دهد و مشخص می کند.

|  |  |
| --- | --- |
| نوع توسعه | نامزد نقش مالک محصول |
| توسعه ی درونی (Internal Development) | نماینده یا مشتری از حوزه کسب و کار که از راه حل و محصول سود می برد |
| توسعه ی تجاری (Commercial Development) | پراکسی یا نائب داخلی (Internal Proxy) برای مشتریان و کاربران حقیقی (معمولاً یک مدیر محصول ، بازاریاب محصول یا مدیر پروژه) |
| توسعه ی برون سپاری شده (Outsourced Development) | نماینده یا مشتری ازکمپانی هزینه کننده برای محصول که از راه حل و محصول سود می برد |
| Component Team (Architectural Development)  توسعه ی معماری ای (!) | به طور معمول یک فرد فنی که می تواند به بهترین شکل آیتم های فنی Backlog را اولویت بندی کند |

# **توسعه ی داخلی:**

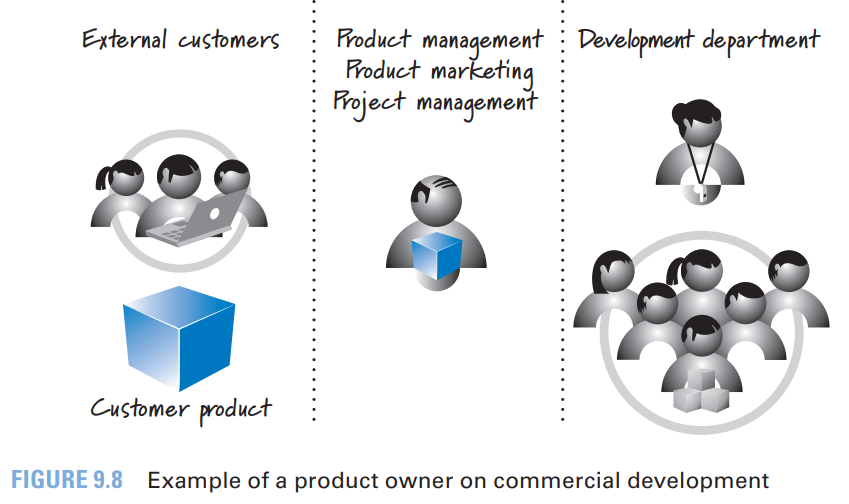
در توسعه داخلی ، یک فرد توانمند از گروه که از توسعه سود می برد، باید مالک محصول باشد. به عنوان مثال، اگر یک گروه IT داخلی، سیستمی را برای گروه بازاریابی ایجاد کند، یک فرد توانمند از تیم بازاریابی باید مالک محصول باشد (تصویر 9.7 را ببینید)



برخی از سازمان ها (معمولاً آنهایی که هنوز اهمیت داشتن یک فرد تجاری به عنوان مالک محصول که به صورت روزانه درگیر می شود را نیاموخته اند) ممکن است از یک فرد IT بخواهند مسئولیت های روزمره مالک محصول را بر عهده بگیرد. وقتی بعداً در این فصل به مفهوم تیم مالک محصول (Product Owner Team) بپردازم، اشکالات این کار را مرور خواهم كرد.

# **توسعه ی تجاری**

در توسعه تجاری - به عنوان مثال، شرکتی که محصولی را برای فروش به مشتریان خارجی می سازد - مالک محصول باید یکی از کارمندان سازمان باشد که به عنوان صدای مشتریان واقعی عمل کند. غالباً این فرد از رده های بالای مدیریت محصول (product management) یا بازاریابی محصول (product marketing) می آید (شکل 9.8 را ببینید).

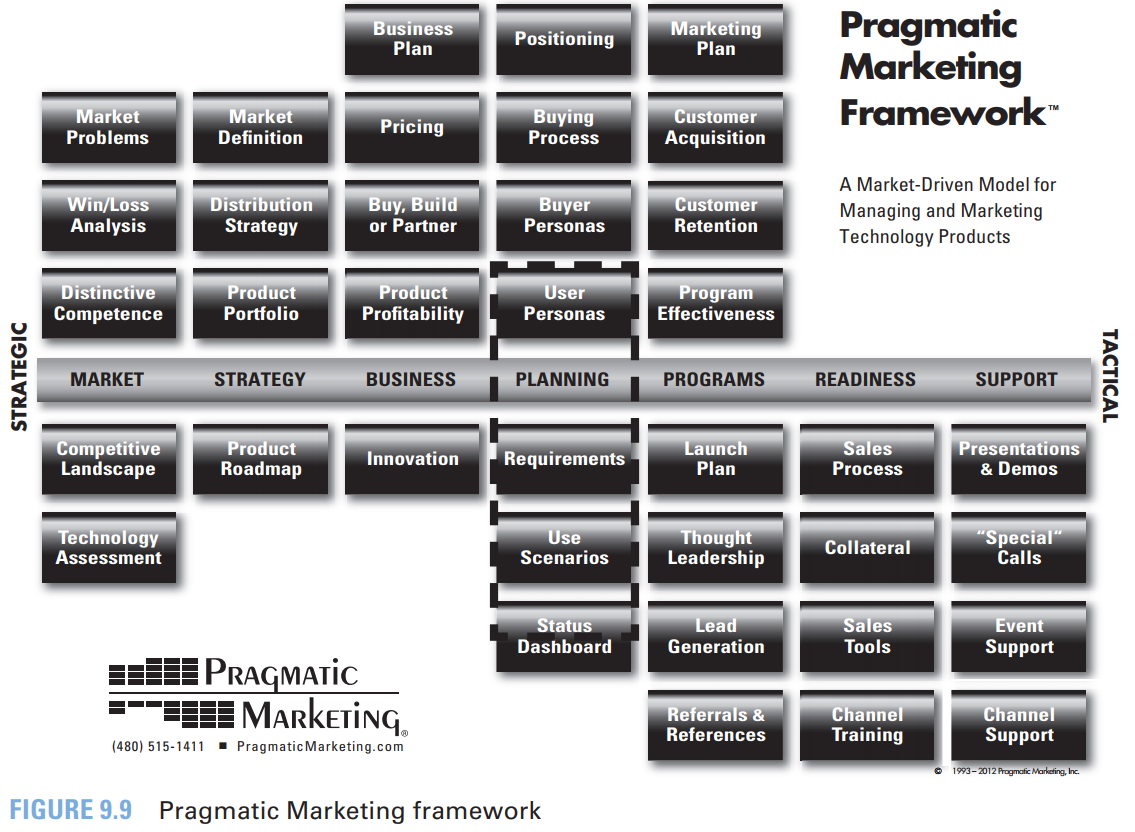


متخصصین اسکرام در این مورد که آیا نقش مالک محصول در اسکرام (و Agile)، فقط یک تغییر نام از نقش مدیر محصول است یا نه به طور داغ بحث کرده اند. برخی معتقدند این دو نقش مترادف یکدیگر هستند. برخی دیگر حامی این دید هستند که نقش مالک محصول بزرگتر از نقش مدیر محصول است. و البته، برخی هم معتقدند که نقش مدیر محصول بزرگتر است. نظر من به این صورت است:

زمینه ها و حوزه های مدیریت محصول و بازاریابی محصول کاملاً گسترده است. یک شرکت مشهور و مورد احترام در زمینه مدیریت و بازاریابی محصول به نام Pragmatic Marketing، چارچوبی بسیار مشهور ایجاد کرده است که نقش ها و مسئولیت های تیم مدیریت محصول و بازاریابی محصول را مشخص می کند (شکل 9.9 را ببینید).

برای پوشش دادن همه این فعالیت ها،Pragmatic Marketing توصیه می کند که به چندین نقش از جمله قهرمانان استراتژی محصول (product strategy champions)، مدیران فنی محصول (technical product managers) و مدیران بازاریابی محصول (marketing product managers) نیاز است. اکثر افراد با این ایده موافقند که اگر یک سازمان تجاری نیاز به انجام همه ی این فعالیت ها برای یک محصول بزرگتر داشته باشد، احتمالاً تیمی از افراد لازم است.

آیا از مالک محصول انتظار می رود که همه این فعالیت ها را انجام دهد؟ کسانی که معتقدند که نقش مالک محصول زیرمجموعه ای از نقش سنتی مدیر محصول است، استدلال می کنند که مالک محصول همان "مدیر فنی محصول" است و بنابراین در درجه اول روی تعداد اندکی از فعالیت های نشان داده شده در خط چین در شکل 9.9، تمرکز می کند. آنها معتقدند که چون مالک محصول باید به صورت روزمره در دسترس تیم باشد، فرصت نخواهد داشت که روی سایر فعالیت ها تمرکز کند.



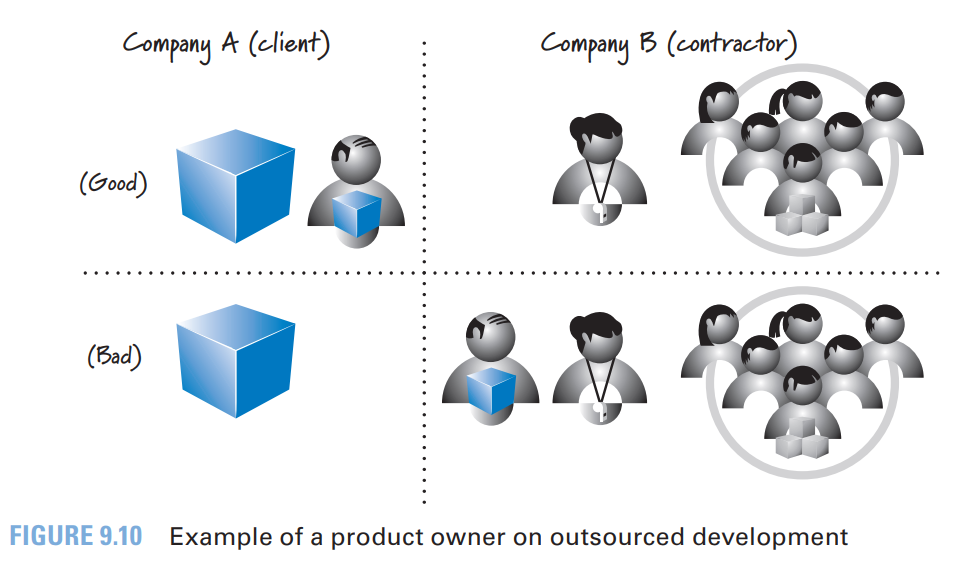
مطمئناً مالک محصول وظیفه انجام فعالیت های درون خط چین (در شکل 9.9) را بر عهده دارد، اما من معتقدم که نقش مالک محصول، مسئول فعالیت های بیشتری نیز هست. در واقع، من معتقدم که نقش مالک محصول باید وظیفه انجام بسیاری از فعالیتهای نشان داده شده در شکل 9.9 را، تا آنجایی که برای مالک محصول ضروری و عملی است، داشته باشد. میزان وسعت این مسئولیت به سازمان، محصول مورد نظر و مهارت های شخصی که به عنوان مالک محصول انتخاب شده، بستگی دارد. به عنوان مثال، سازمانی که یک برنامه تبدیل واحد ساده برای فروش در فروشگاه برنامه های دستگاه های تلفن همراه تولید می کند، به اندازه ی سازمانی که نسخه بعدی محصول Business Intelligence را توسعه می دهد، نیاز به انجام فعالیت ها ندارد. بنابراین، تعریف عمومیِ میزان و گستره ی مسئولیت مالک محصول، نسبت به چارچوب Pragmatic Marketing کاربردی نیست.

همانطور که به زودی اشاره خواهم کرد، ممکن است مواردی پیش بیاید که دامنه ی فعالیت های مالک محصول به قدری زیاد و گسترده باشد که از توان یک یک فرد خارج باشد. در چنین مواردی، ممکن است ما یک تیم مالک محصول داشته باشیم که شامل افرادی باشد که روی استراتژی و بازاریابی تمرکز دارند. با این حال، برای یک تیم اسکرام همیشه یک فرد واحد وجود دارد که در نقش مالک محصول فعالیت می کند.

# **توسعه ی پروژه ی برون سپاری شده**

در مورد توسعه برون سپاری شده - به عنوان مثال، شرکت A که با شرکت B برای ساختن راه حل و محصول قرارداد می بندد - یک نماینده از شرکت A باید مالک محصول باشد. شرکت B ممکن است به یک شخص داخلی این نقش را محول کند که از نزدیک با مالک محصول ارتباط برقرار کند، اما مالک محصول باید از شرکتی باشد که هزینه راه حل و محصول را پرداخت می کند و سود آن را دریافت می کند (شکل 9.10 را ببینید).

اگر شرکت A و شرکت B یک قرارداد توسعه سنتی با قیمت ثابت منعقد کنند، نقش مالک محصول پیچیده می شود. در چنین شرایطی، شرکت B به صورت قطع به یقین احساس می کند که چون خطر یک قرارداد قیمت ثابت را به جان خریده، او باید انجام بسیاری از مسئولیت های مالک محصول را به عهده بگیرد. اگرچه در واقعیت، این شرکت A به عنوان مشتری واقعی است که باید نقش مالک محصول را پر کند. یک قرارداد مناسب تر این است که شرکت A تیم توسعه حرفه ای و ScrumMaster را از شرکت B اجاره کند و شرکت A نقش مالک محصول را ارائه دهد.



# **توسعه ی مولفه ها (اجزا):**

در نهایت، برخی از سازمان ها ممکن است از تیم های component استفاده کنند (فصل 12) که بخش هایی از راه حل ها و محصولات مشتری را می سازند و نه راه حلهای کامل را. این تیم ها معمولا به ایجاد components ها یا asset های دیگری می پردازند که توسط تیم های دیگر برای راه اندازی و ساخت راه حل های Customer-Valuable مورد استفاده مجدد قرار می گیرد. از آنجا که این تیم ها در سطح component های فنی متمرکز هستند، نقش مالک محصول برای آنها معمولاً افرادی با گرايش فنی هستند که قادر به تعریف و اولويت بخشي به مشخصات فنی در Backlog خود هستند (شکل 9.11 را ببینید).

در شکل، سه تیمِ ویژگی تجاری محور (Business-oriented feature team) وجود دارد که ویژگی ها و امکاناتی ایجاد می کنند که برای کاربران نهایی ارزشمند هستند. هر تیم ویژگی، مالک محصول جداگانه ای دارد که بر ویژگی های (feature های) آن تیم متمرکز شده است. همچنین هر یک از این تیم های ویژگی، از نتیجه ی کار یک تیم component استفاده می کنند که asset لازم برای تکمیل ویژگی های آنها را برایشان فراهم می کند. تیم component به یک مالک محصول نیاز دارد که بتواند توسعه ی درخواست های مختلف در سطح مولفه ها (component-level) که توسط تیم های ویژگی ارائه می شود را اولویت بندی و نظارت کند. مالک محصول تیم component، احتمالاً فنی تر از مالک محصول تیم های ویژگی است.

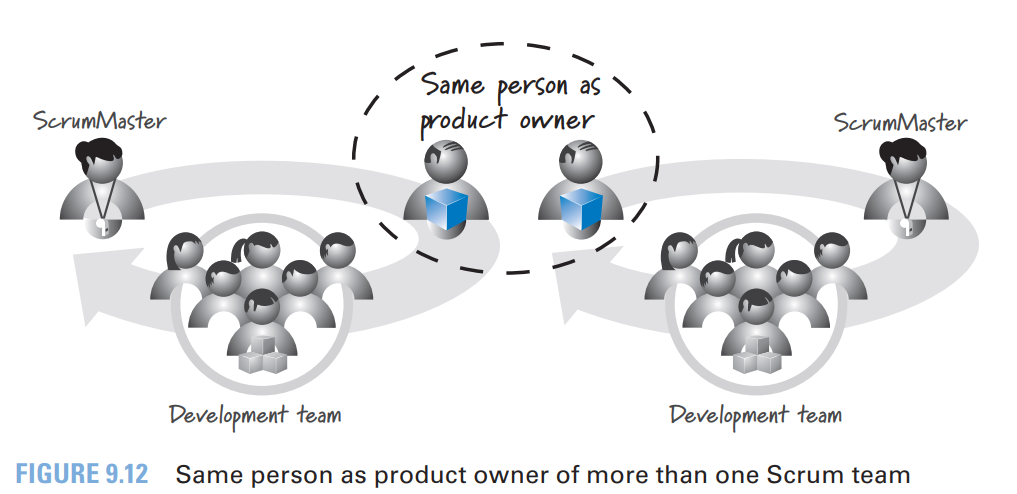
# **مالک محصول در ترکیب با نقش های دیگر**

اگر ظرفیت اجازه دهد، یک شخص می تواند به عنوان مالک محصول برای بیش از یک تیم Scrum عمل کند (شکل 9.12).

در اینصورت معمولا برای این شخص راحت تر است که مالکِ محصولِ تیم هایی باشد که برای توسعه ی یک پروژه ی یکسان تشکیل شده اند، زیرا کار این تیم ها به احتمال زیاد ارتباط زیادی با هم خواهد داشت.

اگرچه برخی مواقع ممکن است یک فرد همزمان مالک محصول و عضوی از تیم توسعه دهنده باشد اما اینکه در یک تیم Scrum، یک نفر به طور همزمان هم مالک محصول و هم ScrumMaster باشد، ایده ی بدی محسوب می شود. این دو نقش یکدیگر را موازنه می کنند. داشتن یک نفر در هر دو نقش تضاد منافع ایجاد می کند که باید سعی کنیم از آن اجتناب کنیم.

# 



# **تیم مالک محصول**

در هر تیم اسکرام باید تنها یک شخص وجود داشته باشد که به عنوان مالک محصول شناخته شود و او تنها فردی باشد که در انجام وظایف مالک محصول برای آن تیم اسکرام اختیار و مسئولیت داشته باشد.

آیا خوب است که به یک تیمی از افراد اجازه بدهیم که در نقش مالک محصول عم کنند؟ اگر منظور ما از تیم گروهی از افراد است که به طور مشترک تصمیم گیر و مسئول هستند، قطعاً اینگونه نیست. برای استفاده صحیح از اسکرام، ما به **یک** فرد نیاز داریم که مالک محصول باشد، تصمیم گیری کند و به عنوان صدای واحد ذینفعان به تیم اسکرام عمل کند.

با اینحال، برخی از سازمان ها ممکن است آنچه را که "تیم مالک محصول" می نامند تشکیل دهند زیرا تشخیص می دهند که با شرایطی که دارند، مالک محصول نمی تواند بدون راهنمایی و ورودی گرفتن از گروهی از افراد منتخب، کار خود را انجام دهد. در برخی شرکت های دیگر، حجم کاری یک مالک محصول ممکن است بیشتر از حد توان و کاری که یک کارمند تمام وقت می تواند به طور منطقی انجام دهد باشد. در این موارد، مالک محصول برخی از مسئولیت های مالک محصول را به افراد دیگر واگذار می کند. تشکیل یک تیم مالک محصول در هر یک از این شرایط قابل قبول است؛ البته تا زمانی که در نهایت یک نفر در تیم، تصمیم گیرنده ی نهایی باشد و تا زمانی که این تیم تبدیل به یک کمیته طراحی نشود به طوری که مثلا هر تصمیم گرفته شده نیاز به تایید 8 نفر دیگر داشته باشد.

هنگام ایجاد تیم های مالک محصول باید مراقب بود. Product owner هایی که مهارت کافی برای عمل کردن به عنوان صاحب اختیار و نقطه مرکزی رهبری محصول را ندارند، به کمیته نیازی ندارند، بلکه به نقش دیگری نیاز دارند (باید در یک نقش دیگر عمل کنند). به همین ترتیب، product owner هایی که بیش از حد مشغول انجام مسئولیت های خود هستند، ممکن است اصلا به تیم احتیاج نداشته باشند. شاید مشکل اصلی این باشد که سازمان تصمیم گرفته است به طور همزمان کارهای توسعه ی (development effort) زیادی را آغاز کند، یا تعداد بسیار کمی از product owner ها برای پوشش دادن محصولات ضروری وجود دارد.

از سوی دیگر، شاید محصولی که در حال ساخت آن هستیم خیلی بزرگ است و بهتر باشد که به یک سری قطعات کوچکتر با انتشارهای مکرر تقسیم شود. با این قطعات کوچک، یک شخص ممکن است به راحتی نقش مالک محصول را پر کند و مسئولیت ها را انجام دهد. همچنین، اگر در تشکیل ساختار تیم هایمان ضعیف عمل کرده باشیم (فصل 12)، یا در درک ساختار Product Backlog ضعیف عمل کرده باشیم (فصل 6)، ممکن است انجام وظایف یک مالک محصول برایش سخت به نظر برسد. اطمینان پیدا کنید که تیم های مالک محصول شما واقعاً لازم هستند و تشکیل این تیم ها سرپوششی بر یک مشکل اساسی نیست. در غیر این صورت، وضعیت پیچیده تر می شود و نتیجه کلی شما را به خطر می اندازد.

# **پراکسی مالک محصول**

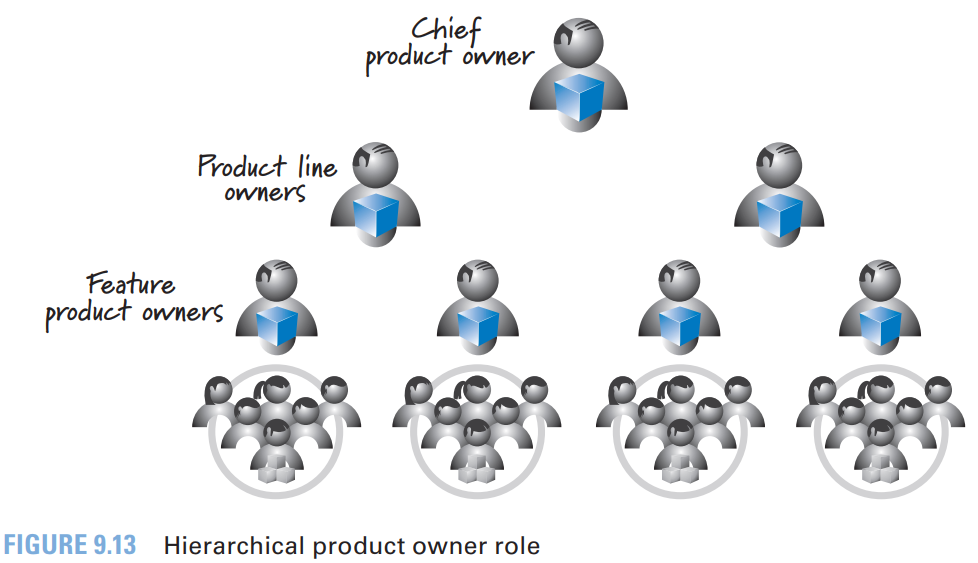
همانطور که قبلاً اشاره کردم، برخی از شرکتهایی که در حال توسعه داخلی (Internal Development) هستند از یک فرد IT (به عنوان مثال، یک تحلیلگر تجارت یا مدیر توسعه) می خواهند که مالک محصول باشد زیرا افراد واحد تجاری بیش از حد مشغول کارهای دیگر هستند. از آنجا که همه می دانند که شخص IT اختیار تصمیم گیری های مهم و نهایی را ندارد (که این تصمیم گیری یکی از مسئولیتهای اصلی هر مالک محصول است)، سازمانهایی که این کار را انجام می دهد در واقع نقش مالک محصول را به طور غیر موثر و گیج کننده ای پر کرده اند. راه حل بهتر این است که به اندازه ی کافی وقت فرد واحد تجاری را آزاد کنند تا به عنوان مالک واقعی محصول عمل کند، ولی شخص IT را به عنوان پراکسی مالک محصول در تعاملات خاص با تیم تعیین کند.

پراکسی مالک محصول شخصی است که مالک محصول از وی کمک می گیرد تا در برخی شرایط خاص به جای او عمل کند. همه ی اعضای تیم Scrum می دانند که پراکسی مالک واقعی محصول نیست، اما همچنین می دانند که مالک محصول به پراکسی این اختیار را داده است که حداقل برخی تصمیمات تاکتیکی را به جای او بگیرد. یک مثال متداول این است که مالک محصول وقت زیادی را صرف ملاقات با مشتریان و کاربران می کند تا مطمئن شود که نبض بازار را در دست دارد. چنین شخصی به طور حتم به صورت روزانه در دسترس تیم توسعه نیست. در این حالت، مالک محصول ممکن است از کمک پراکسی استفاده کند تا تعامل روزانه ی خود با تیم توسعه دهنده در مورد آیتم های Product backlog را انجام دهد.

برای اینکه این رویکرد ثمربخش باشد و کار کند، ضروری است که مالک محصول، واقعا به پراکسی اختیار تصمیم گیری بدهد و با رد کردن بدون دلیل و غیر منطقی تصمیمات گرفته شده توسط پراکسی، اعتبار پراکسی نزد تیم را زیر سوال نبرد. به یاد داشته باشید که حتی با اینکه مالک محصول می تواند به دیگران برای کمک به او اختیاراتی بدهد، اما نمی تواند مسئولیت نهایی برای اطمینان از اتمام کار را به او واگذار کند – بلکه او هنوز خودش برای این کار در مضیقه خواهد بود.

# **Chief Product Owner**

وضعیت دیگری که بعضا تیم مالک محصول تشکیل می شود، مربوط به محصولات بسیار بزرگ است. قبلاً اشاره کردم که ممکن است یک فرد یکسان در دو تیم Scrum مالک محصول باشد، اما در مورد سناریوهایی که تیم های زیادی این وضعیت را دارند چه؟ به عنوان مثال، من در سازمانی آموزش میدادم و مربیگری میکردم که در حال توسعه ی محصولی بودند که بالغ بر 2500 نفر را درگیر خود کرده بود. هر تیم به طور میانگین از کمتر از 10 نفر تشکیل می شد و بیشتر از 250 تیم در این روند توسعه حضور داشتند. یک نفر نمی تواند مالک محصول برای 250 تیم باشد. در حقیقت، یک نفر نمی تواند برای بیش از چند تیم، نقش مالک محصول روزمره را ایفا کند. در اینگونه موارد، نقش مالک محصول باید به صورت سلسله مراتبی به صورتی که در شکل 9.13 نشان داده شده است گسترش یابد.



شخصی که در شکل 9.13 برچسب Chief Product Owner را دارد، مالک محصول برای کل محصول است. با این حال، Chief Product Owner برای اطمینان حاصل نمودن از اینکه نقش مالک محصول به درستی در سطوح پایین تر در سلسله مراتب پر شده است، تیمی از مالکان محصول دارد. اگر خواستید از این روش استفاده کنید، اطمینان حاصل کنید که تک تک تیم های مالک محصول، قدرت و اختیار این را داشته باشند که تصمیمات مربوط به سطح خود را خودشان بگیرند و نیاز به ارسال چنین موضوعاتی به سطوح بالاتر برای تصمیم گیری، نداشته باشند.

**فصل ۱۰ scrum master---------------------- فریده**

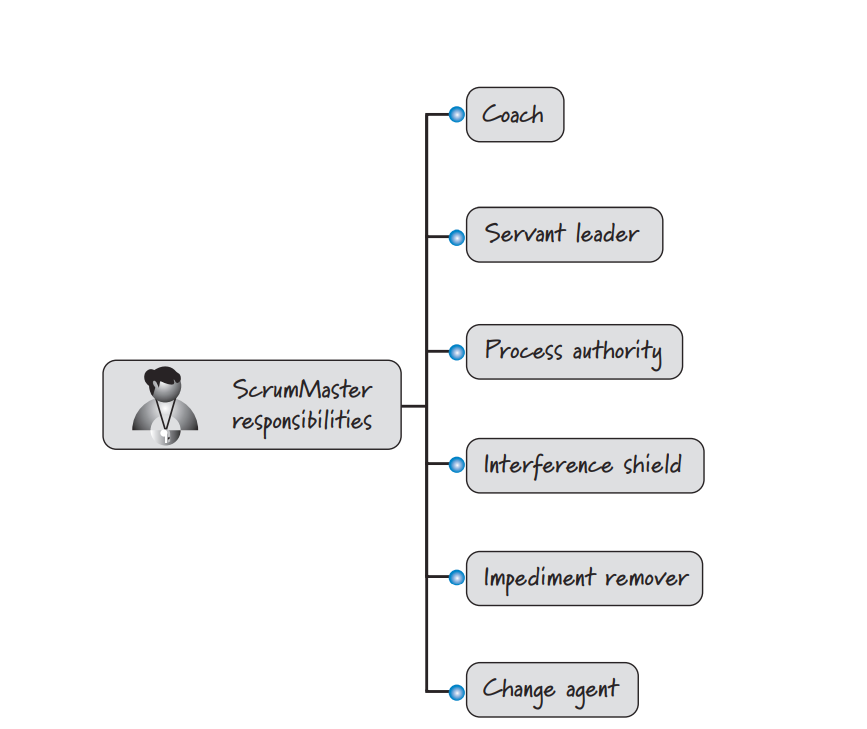
در این فصل نقش تیم توسعه توصیف می‌شود. با بحث در مورد پنج مسئولیت اصلی این نقش شروع می‌کنیم و بعد با توصیف ده ویژگی آن بحث را به پایان می‌رسانیم.

یکی از سه نقشی که یک تیم اسکرام رو تشکیل می‌ده نقش ScrumMaster هست. اون دو نقش دیگه Product Owner (بعضی جاها ترجمه کردم مالک محصول)و تیم توسعه هستند.

در حالی که صاحب محصول بر روی ساخت محصول درست و مناسب تمرکز داره و تیم توسعه هم بر روی ساختن محصول صحیح متمرکزاند، scrum master بر روی اینکه همه مفهوم و اصول و روش اسکرام رو صحیح یاد گرفته باشن، مترکز هست.

اسکرام مستر مثل یک مربی برای تیم توسعه و صاحب محصول عمل می‌کنه. همچنین یک اسکرام مستر رهبری اون فرآیند رو می‌کنه و به تیم اسکرام و بقیه‌ی سازمان‌ها کمک می‌کنه تا اون رویکرد اسکرام سازمان خودشونو با بالاترین کیفیت اجرا کنن.

شکل زیر به خوبی مسئولیت‌های مربوط به اسکرام مستر رو نشون می‌ده:



Coach :

اسکرام مستر مثل یک مربی agile هست برای تیم اسکرام. هم برای صاحب محصول هم تیم توسعه. با مربی‌گری بین این دو نقش، اسکرام مستر می‌تونه موانع بین نقش‌ها رو از بین ببره و صاحب محصول رو مستقیماً به سمت توسعه هدایت بکنه.

مشابه مربی یه تیم ورزشی، اسکرام مسترمی‌تونه نحوه‌ی اجرای اسکرام در تیم رو مشاهده بکنه و برای بالاتر بردن سطح اجرا به اون‌ها کمک بکنه.

وقتی یه مشکلی برای تیم پیش می‌آد، نگرش اسکرام مستر مثل هر مربی خوبی باید اینجوری باشه که بگه من اینجا نیستم که مشکلات‌تون رو براتون حل کنم بلکه اینجام که کمک‌تون کنم تا مشکلات‌تون رو حل کنید.

اسکرام مستر مثل یک مربی به مدیر محصول کمک می‌کنه تا وظایفش رو به خوبی بشناسه و درست بهشون عمل بکنه. هنگام کمک اسکرام مستر به مدیر محصول، تازه بهش کمک می‌کنه فعالیت‌هایی مثل درست راسی کردن پروداکت بک‌لاگ رو هم انجام بده.

علاوه بر این مثل مقایسه تیم ورزشی، رابطه‌ی اسکرام مستر با مدیر محصول بسیار شبیه رابطه‌ی مربی تیم ورزشی با مالک تیم هست: به مدیر محصول کمک می‌کنه که بیشترین و بهترین نتیجه‌ی خود از کار با اسکرام رو ببره، مطمئن می‌شه که مدیر محصول داره هرچی که تیم نیاز داره رو برآورده می‌کنه و به شکایات و درخواست‌ها گوش می‌ده و اونارو در جهت پیشرفت تیم به کار می‌بره.

Servant Leader :

اسکرام مستر مثل یک رهبر خدمت‌گزار برای یه تیم اسکرام تعریف می‌شه. حتی وقتی که مثل یک مربی عمل می‌کنه باز قبل از هر چیزی خدمت‌گزار تیم اسکرامه و اطمینان حاصل می‌کنه که اولویت‌های بالاتر حتماً بررسی می‌شن.

یه رهبر خدمت‌گزار هیچ‌وقت نمی‌پرسه که امروز قراره چه کاری برای من انجام بدی؟؛ به جاش می‌پرسه که که من چه کاری می‌تونم امروز انجام بدم تا برای تیم موثرتر واقع بشه؟

Process Authority :

اسکرام مستر مرجع فرآیند تیم اسکرامه. در این قسمت اسکرام مستر می‌تونه اطمینان حاصل بکنه که تیم اسکرام، اصول و روش‌های اصلی اسکرام رو بهش پایبند هستن یا نه و در مسیر رویکرد اسکرام اون سازمان پیش می‌رن یا خیر.

اسکرام مستر در جهت به بالابردن ارزش تجاری به طور مداوم به تیم اسکرام کمک می‌کنه تا فرآیند ها رو بهبود ببخشه.

اختیار در این زمینه شبیه اختیار داشتن در مدیریت پروژه‌های دیگه نیست. برای مثال اسکرام مستر وظیفه‌ی استخدام یا اخراج افراد رو نداره و اختیار اینکه به افراد تیم بگه که چه تسکی رو الان انجام بدن یا چه جور انجام بدنش رو نداره.

همچنین اسکرام مستر مسئولیت اینکه چک بکنه که کار انجام شده رو، نداره. در عوض اسکرام مستر به تیم کمک می‌کنه تا روند خودشون رو در جهت رسیدن به اتمام محصول، مشخص بکنن.

Interface Shield :

اسکرام مستر از تیم توسعه‌دهنده در برابر عوامل رابط خارجی محافظت می‌کنه تا تیم توسعه بتونه تمرکز خودش رو بر روی کسب ارزش، حفط بکنه.

مهم نیست که این منبع مداخله‌گر چیه؟ چون اسکرام مستر پیگیر و رهگیر هستش و به اون رسیدگی می‌کنه تا تیم بتونه که بر روی ارزش‌ها تمرکزش رو حفظ کنه.

Impediment Remover

همچنین اسکرام مستر مسئولیت رفع موانعی که باعث کاهش بهره‌وری تیم می‌شوند رو نیز داره. (در زمانی که اعضای تیم نمی‌تونند اون موانع رو از بین ببرند).

برای مثال تیم اسکرامی بود که مکرراً نمی‌تونست به اهداف خودش توی هر اسپرینت برسه. مانع‌شون سرورهای تولید غیر استیبل (ناپایدار) بود که وقت تست کردن ازش استفاده می‌کردن. خود تیم کنترلی بر روی این سرورها نداشت. (مسئولیتش با قسمت عملیات VP بود). حالا بخاطر اینکه خود تیم نتونست اون مانع رو برطرف بکنه اسکرام مستر با همکاری VP of Operation و سایر افرادی که می‌تونستند کمکی در ارتباط با این مسئله بکنند، پایداری سرور رو برگردوند.

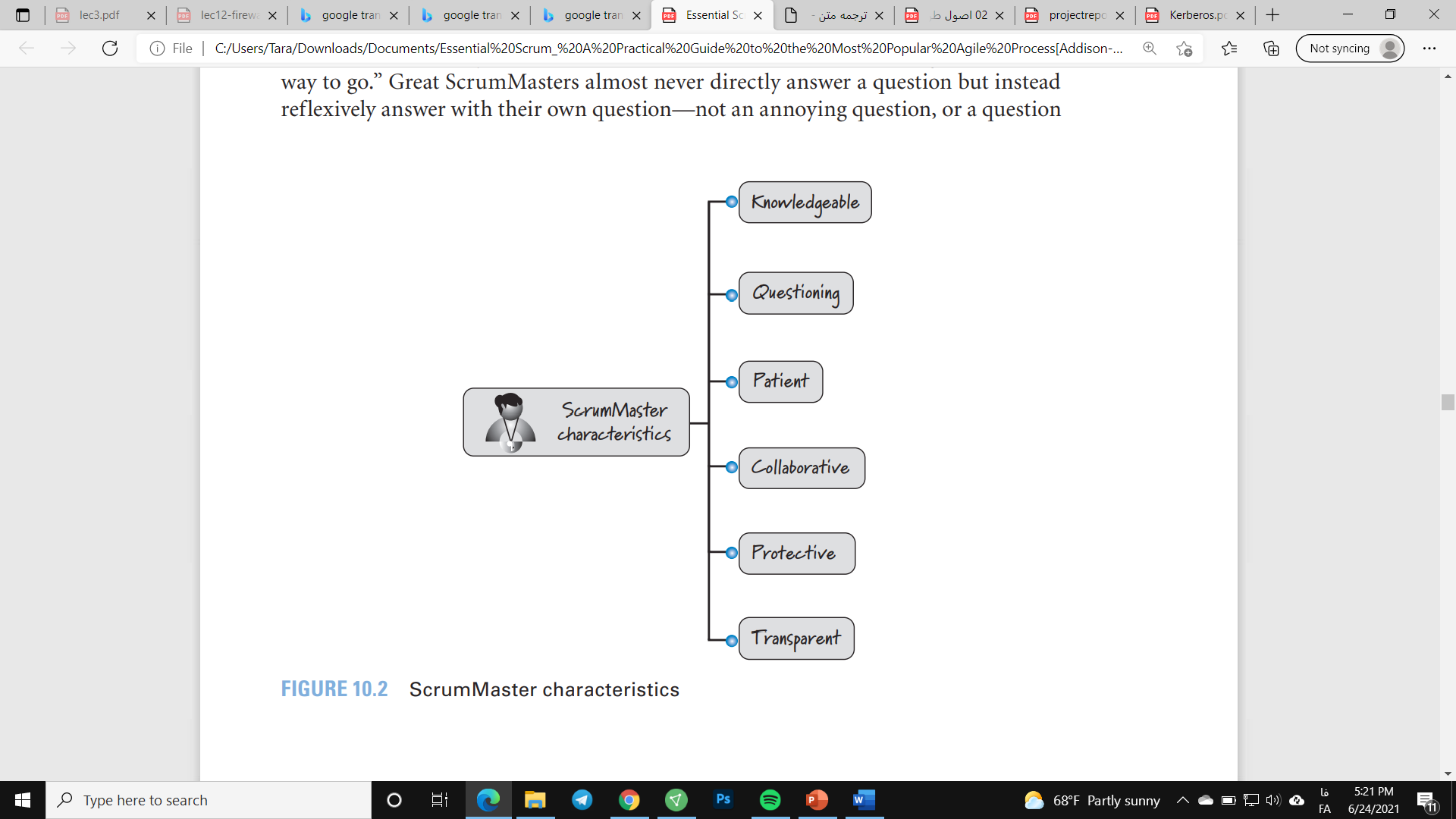
Change Agent

اسکرام مستر باید بیش از تغییر یه سرور خراب و موانع مشابه به این، به تیم کمک بکنه. یه اسکرام مستر خوب اونیه که تغییرات ذهنی رو به خوبی اجرا بکنه.

درسته که تغییرات مورد نیاز برای موفقیت در متد اسکرام کار راحتی نیست اما اسکرام مستر به بقیه کمک می‌کنه تا متوجه بشن که این تغییرات واقعاً نیاز هستند و اینکه میزان تأثیر متد اسکرام بر روی خروجی کار تیم چقدر بالاست و مزایای گسترده‌ای می‌تونه توسط اسکرام کسب بشه.

اسکرام مستر همچین می‌تونه این اطمینان رو بقیه بده که این تغییرات مؤثر داره در تمام سطح سازمان اعمال می‌شه و نه تنها در موفقیت کوتاه مدت که در موفقیت بلند مدت اثر داره و در دراز مدت می‌تونیم اثرات استفاده از اسکرام رو متوجه بشیم.

شکل زیر ویژگی‌های یک اسکرام مستر رو نشون می‌ده:



Knowledgeable

برای اینکه یک مربی مؤثر واقع بشه نیازمند این هست که علم کافی رو داشته باشه. در اینجا نیاز هست که اسکرام مستر از متد اسکرام به طور کامل آگاهی داشته باشه. همچین باید مسائل فنّی مورد نیازِ رو بدونه که در مواقع لزوم بتونه آدرس درستی در ارتباط با تکنولوژی مورد نیاز برای تیم رو بده.

همچنین نیاز نیست که اسکرام مستر مسائل خیلی ریز مربوط به اون حرفه رو بدونه (برخلاف صاحب محصول). اما به طور کلی داشتن دانش در حیطه‌ی اون حوزه تجارت مفیده.

Questioning

اسکرام مستر از مهارت مربی‌گری خودش در کنار فرآیند، دانش فنی و علم کسب و کاری که می‌دونه استفاده می‌کنه تا بتونه سؤال‌های خوبی بپرسه.

اسکرام مسترها سؤالات عمدی‌ای می‌پرسند به طوری که یک سری از مرم دست از کارشون بکش و به خودش بگن: هممم در موردش فکر نکرده بودم و حالا که ازم پرسیدی باعث شد بهش فکر کنم و راه دیگه‌ای براش در پیش بگیرم.

اسکرام مسترها هیچ‌وتق به یک سؤال پاسخ نمی‌دهند اما در عوض با انعکاس جواب با پرسش‌شون، پاسخ می‌دهند. البته که سؤالات آزارهنده نمی‌پرسند یا اینکه سؤال بپرسن که صرفاً سؤال پرسیده باشن بلکه سؤالی که باعث ایجاد یه تفر عمیق در اون فرد بشن و اون خودش با اون سؤال به جواب خودش برسه. (از مدل سؤالات سقراطی)

Patient

چون که اسکرام مستر‌ها ترجیج می‌دن که پاسخی ندهند، نیاز دارن که بسیار صبور باشن و به تیم فرصت بدن که خودشون جواب خودشونو به درستی بدن.

بعضی وقتا اسکرام مستر بودن خیلی سخته؛ چون می‌بینی تیم با مشکلاتی رو به رو هست که تو جوابشونو می‌دونی یا حداقل فکر می‌کنی که می‌دونی! امّا نمی‌تونی پاسخی بدی؛ و این باعث می‌شه که تصور کنی که از میزان هوش تو از میزان هوش جمعی بالاتره و اون زمان صبور بودن خیلی سخته.

Collaborative

یک اسکرام مستر باید مهارت همکاری فوق‌العاده‌ای داشته باشه تا بتونه با مدیر محصول، تیم توسعه و همه‌ی بخش‌های مختلف که حتی شاید با اسکرام مستقیماً در ارتباط نیستن، همکاری داشته باشه.

همچنین به عنوان مربی یه فرآیند، اسکرام مستر دنبال فرصت‌هایی‌ست که بتونه اون رقابت داخل تیم رو با استفاده از به نمایش گذاشتن مهارت‌های هر فرد، بالا ببره.

Protective

اسکرام مستر باید از اعضای تیم محافظت بکنه. مثل این تشبیه رایح که اسکرام مستر مثل چوپان یک گله گوسفند می‌باشد که گله را از گرگ‌هایی که ممکن است قصد حمله و آسیب رسوندن رو دارن، حفظ می‌کنه. حالا در حوزه‌ی کاری ما گرگ‌ها می‌تون اون سازمان‌های یا افرادی باشن که با کارای مختلف باعث مانع پیش بردن اهداف ما بشن.

اسکرام مستر توی زمینه‌ی حفاظت از تیم ماهره مخصوصاً اگر در زمینه‌ی اقتصادی باشه. با حساسیت شدید به سمت محافظت تیم و نیازهای بیزینیس می‌ره و به تیم اسکرام کمک می‌کنه که یه بالانس خوبی برقرار کنن.

اسکرام مستر همچنین اگه ببینه کسی داره از گله دور می‌شه بهش کمک می‌کنه تا به جای رویکردهای مرسوم، به گله برگردند و به اونها کمک می‌کنه تا با تقویت نحوه‌ی استفاده‌شون از اسکرام، بر مشکلات‌شون غلبه کنن.

Transparent

در آخر اسکرام مستر باید در تمامی ارتباط‌هایش شفاف عمل کند. وقتی که داره با اعضای تیم کار می‌کنه دیگه جایی برای برنامه‌های پنهانی وجود نداره. اونچه که از اسکرام مستر می‌بینید و می‌شنوید باید همونی باشه که انجام می‌دین.

همچین اسکرام مستر ارتباطات شفاف خارج از تیم اسکرام رو هم جلو می‌بره.

بدون شفافیت دسترسی به اطلاعات کار سختی می‌باشد.

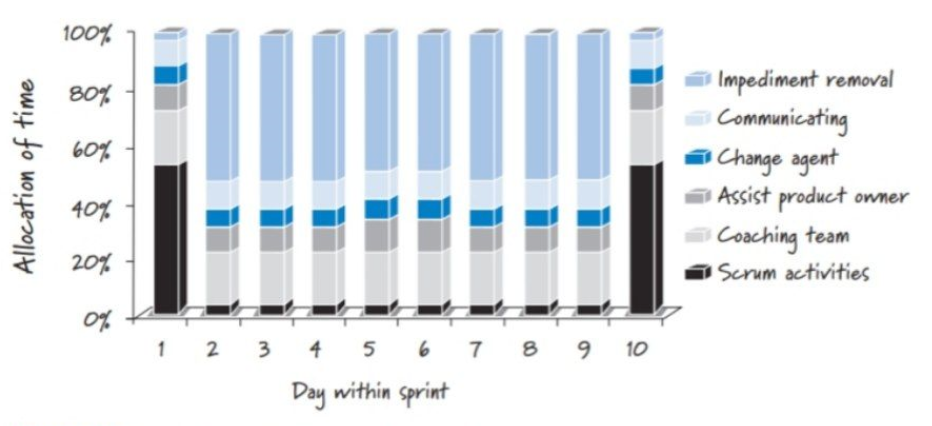
یک روز از زندگی

زندگی برای یک استاد اسکرام در حین یک اسپرینت چگونه است؟ شکل 10.3 نشان میدهد که یه اسکرام مستر یک گروه تازه تشکیل شده تقریبا چقدر زمان برای هر کار در فعالیت در طول یک اسپرینت باید بگذراد. درصد های تخصیص داده شده ممکن است برای اسکرام مستر یک گروه سطح بالا که چند سال است باهم کار میکنند متفاوت باشد.

همانطور که در شکل نشان داده شده، اسکرام مستر هرروز میزانی زمان برای مرتب سازی (organize ) و تسهیل کردن (facilitate) فعالیت های اسکرام صرف میکند که شامل پلنینگ اسپرینت، اجرا(excecute) کردن اسپرینت، review های اسپرینت ، اسپرینت های نگاه به عقب (resrospective) و اسکرام های روزانه است. این شامل تنطیم کردن فعالیت ها، بازنگری اجرای آنها و enable کردن بقیه ی تیم اسکرام برای فعالیت در سطحی است که hig\_value resalut ها آرشیو شده اند.

اسکرام مستر همچنان هرروز میزانی وقت صرف میکند تا به اعضای تیم آموزش دهد تا به آنها کمک کند استفاده ی خود از اسکرام و عملیات های تکنیکال (technical practice) را ارتقا بخشند. اسکرام مستر همچنین باید یک سری آموزش های به مبتدی ها (refresher trainnig) را اجرا کند برای مثال یاداوری به یک تیم جدید درمورد قوانین palnning poker در هنگام تخمین آیتم های پروداکت بک لاگ. هم چنین بخشی از روز مختص به برقرار کردن ارتباط است(برای مثال آپدیت کردن اسپرینت و انتشار نمودار های برن آپ و برن داون یا بحث کردن با اعضایی که در تیم اسکرام نیستند)

در طی اسپرینت، اسکرام مستر زمانی را نیز صرف میکند تا با مالک محصول روی gromming فعالیت های پروداکت بک لاگ زمان بگزارد مثلا (نوشتن و اولویت بندی کردن آیتم های جدید پردواکت بک لاگ) اسکرام مستر همچنین با مالک پردواکت در این جهت کار میکند تا مطمین شود trade off های سودمند اقتصادی با توجه به متغیر های مهمی مثل فیچر ها، دیتا، بودجع و کیفیت در حال انجام است.



اسکرام مستر همچنین وقتی را صرف میکند تا به عنوان یک change agent ? به سازمان کمک کند اسکرام را در بیشتر طول زنجیره ی مقادیر (value chain) در آغوش بگیرد!(در بر بگیرد شاید) {از بخش فروش ، بازاریابی، HR ، قرارداد ها و ...)

اسکرام مستر یک وقت متغیری را صرف میکند تا هرروز موانع را برطرف کند. او ممکن است هرروز یک مقدار ثابت زمان را بودجه بندی کند مخصوص برطرف کردن موانع . قطعا موانع در هر زمانی میتوانند طاهر شوند و آنها ممکن بزرگ و حساس به زمان باشند پس اسکرام مستر ممکن است نیاز داشته باشد تا به طور داینامیک تایم را برای رسیدگی تنظیم کند.

بیشتر تیم ها و سازمان ها که اسکرام برایشان جدید است موانع زیادی برای شروع و تمرکز بر روی موانعی دارند که حل آنها سریع و قابل دسترس است. اما این به این معنا نخولهد بود مه همی موانع برطرف کردنشان آسان خواهد بود. در واقع مرحله ی بعد موانع بسیار سخت تر و زمان بر تر خواهد بود هم شناسایی و هم برطرف کردنش. به طور کلی برطرف کردن موانع یه متغیر بزرگ در روز یک اسکرام مستر هست و میتونه به سادگی تخصیص های زمانی نشان داده شده روی نمودار را عوض کنه.

ایفای نقش

هنگام در نظر گرفتن نقش اسکرام مستر ما نیاز دارم تصمیم بگیریم چه کسی برای این نقش بهترین هست و اینکه عایا شغل باید تمام وقت باشه یا به طور پاره وقتم انجامش ممکنه و اینکه عایا میتونه با هرکدوم از نقش های درون یا بیرون اسکرام هم تلفیق بشه یا نه. بیایید تا تمام حالات را در نظر بگیریم.

چه کسی باید اسکرام مستر باشد؟

سازمان هایی که اسکرام براشون جدیده افرادی رو در نقش اسکرام مستر ندارن. بنابراین از کجا باید این شخصو پیدا کرد؟ افراد متنوع و زیادی از نقش ها و کارهای متنوعی دیده شدن که اسکرام مستر های عالی ای شدن.بعضی اسکرام متسر ها قبلا مدیر پروژ] بودن یا ممدیر محصول البته مدیر محصولا به احتمال زیاد به سمت مالک محصول شدن پیش میرن .بعضی اسکرام مسترا قبلا ئولوپر تستر بودن یا کلا یه بک گراند تکنیکی داشتن. تا زمانی که شخص ویژگی های لازمو داشته باشه که زوتر درموردش حرف زدم، و خودش مایل باشع کع این مسیولیتو بگیره میتونه یه اسکرام مستر موثر باشه.

بعضی ارگان ها احساس میکنن که سرپرست بخش های تکنیکی و توسعه باید اسکرام مستر باشن. این افراد درواقع اسکرام مستر های خیلی خوبی از آب در میان اما ممکنه بهترین انتخاب برای این نقش نباشن. افرادی که در موقعیت های مدیریت تکنیکی هستنبرای یه دلیلی اونجان! اونا از نظر تکنیکی توی کارشون خیلی خوبن. نقش اسکرام مستر نقشی نیست که دانش تکنیکی فوق العاده بتونه به حد اعلای خودش مورد استفادع و شکوفایی قرار بگیره.هر زمانی که مدیر های تکنیکی کار اسکرام مستر رو انجام میدن در عین حال مدیریتشون تو بخش تکنیکی کمتر میشه. اینکه اونارو اسکرام مستر کنیم ممکنه نتایج قابل توجه و بدی روی خروجی تکنیکال داشته باشه. در ادامه در مورد اینکه عایا کسی از تیم دولوپر ها میتونه اسکرام مستر بشه یا نه صحبت خاهم کرد.

مدیر های بخش های عملیاتی یا مدیران منابع میتونن همچنین اسکرام مستر های موفقی بشن اگه مهارت انجامشو داشته باشن. بهتره این افراد دیگه مدیریت های قبلی خودشونو بعد از اسکرام مستر شدن ول کنن حداقل برای کسایی که تو اسکرام هستن. چون یه اسکرام مستر هویت مدیریتی داشته باشه برای افراد تیم و ممکنه این برای افراد گیج کننده بشه که عایا مث یه مدیر با این طرف برخورد کنن یا نه. من ترجیح میدم این موقعیت رو ازش دوری بکنم و افراد مجبور نباشن به اسکرام مسترشون گزارش هم بدن (ینی اسکرام مستر مدیرشونم باشه). در بعضی ارگان ها ولی ممکنه که این موقعیت غیرقابل دوری کردن باشه و کجبور بشیم این کارو بکنیم. پس باید سعی کنیم یاد بگیریم با هر تناقض ممکنی به خوبی رویارویی کنیم.

آیا اسکرام مستر بودن یک شغل تمام وقت است؟

هر تیم اسکرامی یک اسکرام مستر دارد انا عایا اسکرام مستر یه نقش تمام وقته؟ احتمالا نه. یه تیم اسکرام که باهم در یه زمان بلندی از زمان کار کردن و در مورد اسکرام و تعامل با همدیگه خیلی حرفه ای شدن حتما زمان کمتری برای آموزش و کمک های اسکرام مستر نیاز خواهند داشت تا یه تیمی که تازه جمع شدن دورهمو جدیدن و تا حالا باهم کار نکردن و اسکرامم براشون جدیده.

هرچند اسکرام مستر کمتر نیاز داره که هرروز با حرفه ای ها تیم وقت بزاره، ولی به هرحال نقش اسکرام مستر بنیادی عه برای موفقیت اسکرام در اون شرکت. معمولا هرچه نیاز تیم اسکرام به اسکرام مستر کم تر میهش، نیاز اسکرام مستر هم به تمرکز روی موارد فراتر مثل برطرف سازی مانع عا و یه change agent بودن و تمرکز بر روی زنجیره ی ارزش ها بیشتر میشه.

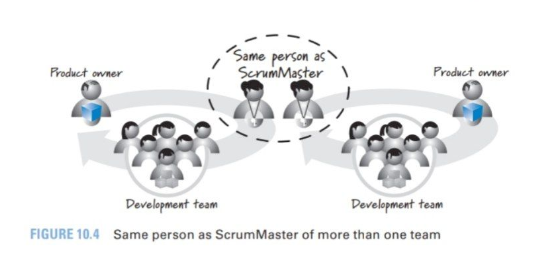
در بیشتر موارد نقش اسکرام مستر تو یه زمان مشخصی که تعهد شده باقی میمونه. در اون مواردی که این تعهد تمام وقت نیست ممکنه یه ترکیبی از نقش های دیگه بقیه زمانو پر کنه و جایگزینش بشه.

ترکیب اسکرام مستر با نقش های دیگه

اگه ظرفیت اجازه بده و یه شخص هم یه اسکرام مستر با استعداد باشه و هم عضو تیم توسعه دهنده ها. اون شخص ممکنه بتونه توی هر دو نقش به خوبی عمل کنه. هرچند این ترکیب ممکنه مشکل تداخل علاقه مندی ها و اهدافو داشته باشه وقتی که شخص تلاش کنه به هردوکارش برسه. برای مثال چی میشه اگه کسی که دوتا کارو داره یه اکتیویتی مهم اسکرام مستری مث برطرف کردن مودانع داشته باشه و یه بخش خیلی مهم عملی توسعه دهندگی هم بهش داده باشن تو اون بازه اززمان؟ چون هردوتاش مهمه بی توجهی به هرکدوم موثر بودن تیم اسکرام رو کاهش میده. د رواقع پیچیده شدن این trade off میتونه یه مانعی باشه که رسیدکی بهش خییلی سخت و غیرمنتظره و زمان بر باشه. این حتی باعث میشه که سخت تر بشه پیش بینی اینکه یک اسکرام مستر به عنوان یه عضو از تیم چقدر زمان در دسترس خواهد داشت برای انجام کارهای توسعه دهندگیش.

ولی به هرحال یه رویکرد متفاوتی هست که اغلبا بهتره. اگه یه اسکرام مستر واقعا ظرفیت موجود داشته باشه در خیلی از موارد این اپشن وجود داره که اسکرام مستر بیش از یک تیم باشه.

یک اسکرام مستر خوب بودن مستلزم اینهحکه یه سری مهارت های با ارزش و غیرعام داشته باشی. من ترجیح میدم یه شخص یکه اون مهارت هارو داره با چندین تیم اونارو به اشتراک بزاره تا اینکه بیاد در کنار اسکرام مستر به کارای نامربوط به اسکرامو انجام بده.ولی این فقط یه ترجیح شخصی عه و اسکرام مستر های موفقی هم دیدم که از هر دو روش ممکنه استفاده کرده باشن. هیچ جواب درست یا غلطی به این سوال وجود نداره درواقع ولی ممکنه برای یه ارگان و اشخاص خاص جواب درست یا غلط ینی جواب قطعی وجود داشته باشه.



همونطور که اشاره کردم توی فصل 9، یک ترکیب از نقش ها برای یک نفر که خیلی منع شده اسکرام مستر و مالک محصول هست. اسکرام مستر مربی تیم اسکرام باید باشه و این به این معنی هست که اسکرام مستر مربی مالک محصول هم هست. خب خیلی سخئته که تو بتونی مربی خودت باشی.به علاوع مالک محصول واقعا صاحب اون محصول هست و میتونه از تیم درخواست داشته باشه(؟؟؟مثلا شاید ینی بهشون دستور بده؟)

اسکرام مستر مرتبا مثل یک مامور ایجاد توازن بین درخواست های مالک محصول، نیازمندی ها و توانایی های تیم توسعه عمل میکنه. اینکه یه نفر هم اسکرام مستر باشه هم صاحب محصول میتونه باعث ایجاد گیجی بشه که خب به سادگی میشه اینکارو نکینم و ازش دوری کنیم.

جمع بندی

در این فصل من نقش اسکرام مستر رو تشریح کردم. من مسیولیت های اسکرام مستر رو مثل مربی ، رهبر خدمت گذار،مدیر فرایند،زره دخالت(!interfrence sheild)، برطرف کننده موانع و change توصیف کردم . سپس در اینمورد بحث کردم که اسکرام مستر باید چه دانشی از اسکرام داشته باشه، سوالای خوب بپرسه، با صب ر صب کنه تا تیم مشکلاشو حل کنه، با همه در تعامل باشه، از مداخله و دخالت بی مورد جلوگیری کنه، و به طور شفاف و مشخصی ارتباط برقرار کنه. بعد گفتیم که زمان یه اسکرام مستر چجوری باید تقسیم بندی و صرف بشه و بعد هم اینکه کی باید اسکرام مستر باشه و چه نقشایی باهاش سازگار هستن یا نیستن و اینکه این نقش فول تایم هست یا نه.

**فصل ۱۱ development team ---------------------- مائده**

در این فصل قصد داریم در مورد نقش development team حرف بزنیم. اول با بیان 5 مسئولیت این نقش شروع می کنیم و در ادامه 10 کاراکتری که باید یک تیم توسعه داشته باشد را بیان میکنیم.

Overview:

رویکرد های سنتی توسعه نرم افزار تایپ های مختلفی از job یاامور را تعریف میکنند مثل معمار ، برنامه نویس، تست کننده ، ادمین دیتابیس ، UI designer و ... ولی اسکرام نقش تیم توسعه را به گونه ای که یک مجموعه cross functional دارد که این وظایف را انجام میدهند، تعریف میکند. تیم توسعه یکی از 3 نقش موجود در هر تیم اسکرام است. اعضای تیم توسعه ، تجربه های لازم برای deliver کردن ارزش بیزینس درخواستی از product owner را دارند. عبارت تیم توسعه، ممکن است برچسب اشتباهی برای استفاده در یک تیم باشد که بیش از فقط توسعه دهندگان تشکیل شده است. از برچسب های دیگری استفاده شده است ، مانند تیم تحویل ، تیم طراحی آزمایش ، و فقط تیم. مشخص نیست که هر یک از این برچسب ها از تیم توسعه مناسب تر ، کمتر مبهم یا استفاده از آن راحت تر است. در حال حاضر ، جامعه اسکرام در استفاده از اصطلاح تیم توسعه همگرا شده است ، و من از این اصطلاح در این کتاب استفاده خواهم کرد.

**Role-Specific Teams ( تیم های نقش ویژه)**

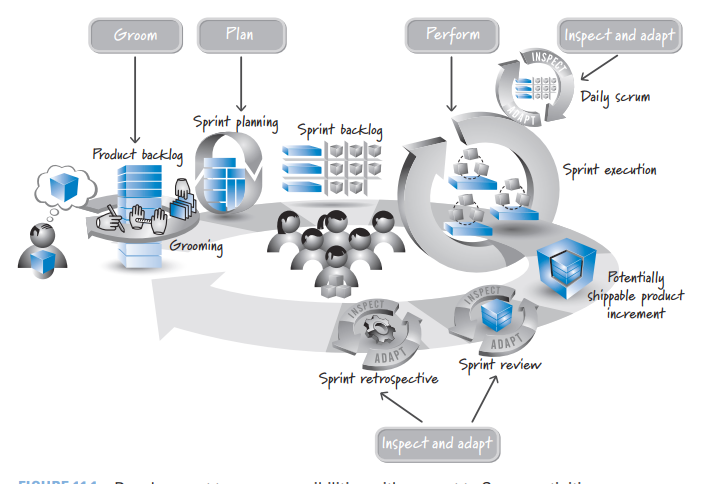
بسیاری از سازمانها عادت دارند که عمداً نقشهای مختلف کاری را به تیم های تخصصی و نقش ویژه تقسیم کنند. این سازمانها ممکن است یک تیم از طراحان ، یک تیم از توسعه دهندگان و یک تیم دیگر از آزمایشگران ( یا تیم تست ) داشته باشند. این تیم ها کار را به همدیگر واگذار می کنند وقتی که یک تسک وقتی کامل شده باشد و عملکرد آن کم و بیش مستقل از بقیه باشد. دراسکرام تیم توسعه باید تمام امور لازم جهت تولید یک یا چند بخشی که در راس کار و مهم هستند را در هر اسپرینت انجام دهد که شامل طراحی، توسعه، integration یا یکی کردن ( هماهنگی – یک پارچگی) و تست آن عملکرد می باشد. پس به تیمی نیاز داریم که در همه موارد ذکر شده باتجربه باشد.

برخی سازمان ها سعی میکنند که تیم جداگانه ای برای تست داشته باشند ( در تیم اسکرام). البته الان هم وجود چنین تیم جداگانه ای برای تست تاحدودی خوب و لازم است ولی اکثر اوقات چنین نیازی وجود ندارد. عملیات تست باید همزمان با ارائه اسپرینت ها انجام شود و لذا لازم است که خود تیم توسعه این کار را به عهده گیرند.

پس میتوان نتیجه گرفت که یک تیم باید همه جانبه کار کند در همه زمینه ها. و ایجاد تیم های مختلف تر و واگذاری وظایف به ان ها، مانعی برای عملکرد بهینه اسکرام است.

**Principal Responsibilities ( اصول مسئولیت ها)**

نمودار زیر بیانگر امور اسکرام است همراه با تفسیر اصول مسئولیت های تیم توسعه. در ادامه هر یک از این مسئولیت ها را توضیح میدهیم.



اصول به صورت زیر است :

* **Perform sprint execution(اجرای عملکرد اسپرینت)**

در مدت اجرای اسپرینت ، اعضای تیم توسعه روی طراحی، معماری، یک پارچگی و تست ایتم های پروداکت بک لاگ کار میکند تا خروجی و داکیومنت های قابل ارسال و ارائه، به سرانجام برسد. بدین منظور ان ها خودشان برنامه ریزی میکنند و اجماعا تصمیمات لازم جهت برنامه ریزی ، مدیریت ، انجام کار و امور مرتبط دیگر را میگیرند. در کل تیم توسعه اکثر تایم خود را رو اجرای اسپرینت میگذارند.

* **Inspect and Adapt Each Day (بازرسی و سازگاری هر روزه)**

از هر یک از اعضای تیم توسعه انتظار میرود که در اسکرام های روزانه شرکت کند ، ان هم زمانی که اعضای تیم انتظار پیشرفت در کار را دارند تا به هدف ان اسپرینت برسند. اگر کسی از تیم توسعه در جلسات شرکت نکند، تیم ممکن است بخشی از big picture یا شمای کلی کار را از دست بدهند و در ادامه ممکن است به هدف ان اسپرینت نرسند.

* **Groom the Product Backlog ( تهیه منظم از پروداکت بک لاگ)**

بخشی از یک اسپرینت باید برای اماده شدن برای اسپرنت بعدی صرف شود. بخش بزرگی از این کار با تمرکز روی نوشتن پروداکت بک لاگ انجام میشود که شامل ایجاد و پالایش (؟) ، حدس و تخمین و اولویت بندی ایتم های پروداکت بک لاگ می شود. تیم توسعه باید حداقل 10% از ظرفیت اماده خود را به کار گیرد تا امور لازم برای هر اسپرینت انجام شود تا با product owner هم در انجام این فعالیت ها یاری کنیم.

* **Plan the Sprint( برنامه ریزی اسپرینت)**

در ابتدای هر اسپرینت تیم توسعه در برنامه ریزی ان اسپرینت شرکت میکنند . در هماهنگی با product owner و با کمک scrum master ، تیم توسعه هدفی برای اسپرینت اینده اش قرار میدهد. در ادامه تیم تعیین میکند که کدام یک از ایتم های پروداکت بک لاگ با اولویت بالا باید هدف قرار گیرند. برای یه اسپرینت دو هفته ای ، برنامه ریزی اش نصف روز طول میکشد. پس برنامه ریزی به طور مرتب انجام میشود. به جای تمرکز روی امور بزرگ و بی اهمیت و برنامه بیش از حد دقیقی که در ابتدای اسپرینت برنامه ریزی شده ، تیم توسعه مجموعه ی کوچکتر و مهم تری از ان امور را برای انجام برمیگزیند و همچنین در اغاز هر اسپرینت برنامه ریزی دقیق تری انجام میدهد.

* **Inspect and Adapt the Product and Process(**بررسی و تطبیق دهی محصول و فرآیند**)**

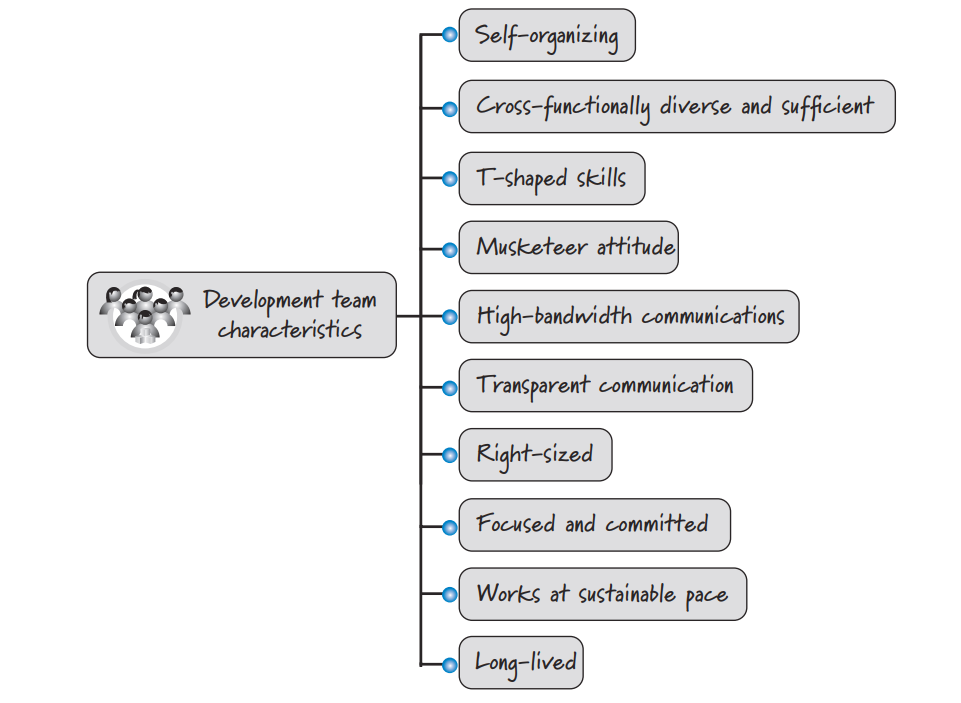
در پایان هر اسپرینت تیم توسعه در فعالیت های بررسی و تطبیق شرکت میکنند که شامل sprint review و sprint retrospective( گذشته نگری) است.

Sprint review: زمانی است که تیم توسعه، product owner، اسکرام مستر ، اسپانسرها ، ذی نفعان، مشتری ها و افراد علاقه مند از تیم های دیگر، فیچر ها ( ویژگی ) های تازه تکمیل شده از اسپرینت فعلی را مرور میکنند و بحث میکنند که بهترین حرکت در اینده چیست.

Sprint retrospective: زمانی است که تیم اسکرام بررسی میکند و تطبیق میدهد پروسس اسکرام را با تمریین های تکنیکی تا ارزش عملکرد به روش اسکرام را در بیزنس ارتقا دهند.

**Characteristics/Skills (ویژگی ها و مهارت ها )**

شکل زیر مهم ترین ویژگی های یک تیم توسعه را بیان میکند:



**بررسی ویژگی ها :**

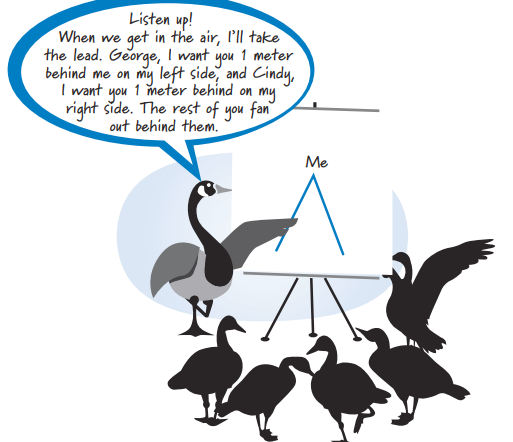
1. **Self\_organizing**

اعضای تیم، خود را سازماندهی می کنند تا بهترین راه برای رسیدن به هدف اسپرینت را تعیین کنند و هیچ project manager یا مدیران دیگری نیستند که به انها بگویند که چیکار کنند. Self\_organization یک اقدام مهم سیستم از نوع bottom-up است و هیچ نیروی اعمالی خارجی سنتی از نوع top-down و یا command and control management وجود ندارد.

مثلا برندگان را که با نظمی زیبا پرواز میکنند. در چنین سیستم هایی ، بسیاری از موجودات به طرق مختلف با یکدیگر تعامل می کنند ، و این تعاملات با قوانین ساده و محلی انجام می شود که در یک زمینه از بازخورد مداوم کار می کنند. این دسته از سیستم ها بیانگر مشخصه های جالب هستند مانند به طور قابل توجهی مقاومت و تولیدی شفت انگیز.

مانند یک دسته از پرندگان، یک تیم توسعه هیچ قدرت top-down و یا command and control ای ندارد که به تیم بگوید یک کار را چگونه انجام دهند. به جای آن، یک تیم تقسیم وظایف متقابل دارند که خودشان را سازماندهی میکنند که در بهترین راه کار را به سرانجام رسانند. در واقع ، آنچه ظاهر می شود معادل الگوی V تیم است. در هر حال مدیران نقش اساسی ای در اسکرام دارند. انها بار ها و بار ها محیطی برای self-organize کردن تیم فراهم می اورند. ( اقا لپ کلام خود تیم تصمیم میگیرن و برنامه ریزی میکنن که چه غلطی بکنند در هر اسپرینت .( مثل پرندها که دیگه خودشون یادمیگیرن منظم پرواز کنن و با هم باشن.) حالا مدیر اسکرام هم یه نقش نظارتی داره دیگه این وسط. )

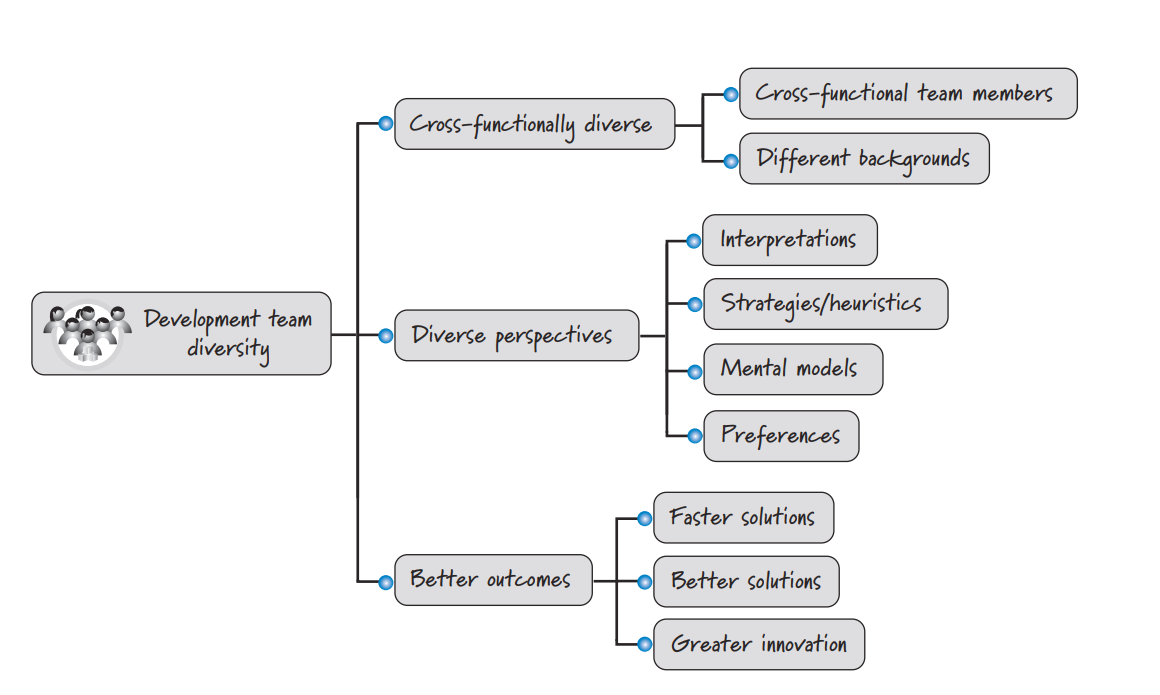
اینم این شکل v که میگفت ☹



1. **Cross-Functionally Diverse and Sufficient(عملکرد متقابل متنوع و کافی)**

اعضای تیم توسعه باید تنوع عملکرد داشته باشند. اجماعا ان ها باید مجموعه ای ضروری و کافی از تجربه ها و مهارت ها را برای به سرانجام رساندن امور داشته باشند. یک تیم خوب و well form میتواند یک ایتم از پروداکت بک لاگ را بردارد و یک فیچر کارامد با کیفیت خوب را ایجاد کند که مطابق انچه که تیم اسکرام انجام داده است ، باشد. :/

تیم هایی که فقط از افراد با مهارت های یکسان تشکیل شده اند (تیم های silo ی سنتی) می توانند حداکثر بخشی از کار را انجام دهند. در نتیجه ، تیم های سیلو (silo)( نمیفهمم یعنی چی :/) در نهایت محصولات کار را به سایر تیم های سیلو تحویل می دهند. به عنوان مثال ، تیم توسعه کد را در اختیار تیم تست (آزمایش) قرار می دهد ، یا تیم UI طراحی صفحه را به تیم منطق تجارت تحویل می دهد. این تحویل دادن یا همان HANDS OFF فرصتی عالی است برای ارتباط نادرست و اشتباهات پرهزینه ممکن. داشتن تیم های متنوع تعداد این hands off را به حداقل می رساند. و ایجاد تیم های متنوع مانع ما نمیشود که چندین عضو تیم که ممکن است مهارت بالایی داشته باشند ، را داشته باشیم. تیم های متنوع از نظر عملکرد نیز چندین دیدگاه را به همراه دارند که منجر به نتایج بهتر می شوند.



یک تیم متنوع عملکردی دارای اعضایی با بک گراندهای مختلف است. هر عضو تیم برای تیم مجموعه ای از ابزار های شناخته شده را برای حل مسائل فراهم می اورد. این ابزار ها و امکانات میتواند حاوی تفسیر های مختلف ( از دیتای یکسان) ، استراتژی های مختلف برای حل مشکلات، مدل های ذهنی مختلف از چگونگی کارکردها و اجراهای مختلف برای هر دو رویکرد ها و راهکار ها. این نوع از تنوع معمولا منجر میشود به خروجی بهتر از نظر راه حل های سریع تر ، محصولات قابل تحویل با کیفیت تر و خلاقیت بهتر و کلا هر ان چه که به ارزش اقتصادی بالاتری ما را می رسانند.

همچنین ما باید تلاش کنیم برای این تنوع تیم با داشتن یک ترکیب خوب از افراد باتجربه و بی تجربه تا بتوانیم تیم را همسطح کنیم. وجود تعداد زیادی از افراد با تجربه ، ممکن است باعث ایجاد یک اشفتگی غیرضروری شوند مثل اینکه در یک اشپزخانه چند اشپز بامهارت داشته باشیم.در هر حال وجود تعداد زیادی افراد بی تجربه در این زمینه هم ممکن است باعث این شود که مهارت کافی برای انجام کار در تیم وجود نداشته باشد. یک ترکیب خوب از این افراد یک محیط یادگیری سالم و مشترک را ایجاد میکند.

1. **T-Shaped Skills( مهارت های تی شکل)**

تیم های توسعه ی FLEXIBLE از افرادی با مهارت های تی شکل تشکیل شده اند. مهارت های تی شکل به این معنی است که یک عضو تیم دارای مهارت هایی عمیق در زمینه ارجح خودش دارد . مثلا فلانی یک UX کار حرفه ای است که ترجیح میدهد در این زمینه همیشه کار کند. البته که یک فرد میتواند خارج از حوزه تخصصی خود هم کار کند ولی خب به خوبی نیمتاوند مثل حوزه تخصصی خودش در ان ها عمل کند ولی باز هم میتواند نقش کمکی داشته باشد زمانی که حجم کار زیاد است و باید سر موقع تمام شود.

باور اینکه هر شخصی در یک تیم بتواند روی هر تسکی کار کند، غیرواقعی و یک هدف مغرورانه است. برای مثال در حوزه هایی با تخصص شدید مثل توسعه VIDEO GAME ها ، که یک تیم میتواند یک ارتیست، انیمیتر و مهندس آدیو ، هوش مصنوعی و تستر داشته باشد، این غیرمنطقی است که همه توانایی های یکسانی در انجام امور داشته باشند.

لذا مدیران باید روی ایجاد و تشکیل یک تیمی که بهترین مجموعه مهارت های تی شکل را داشته باشند، تمرکز کنند. با این حال ممکن است امکان پذیر نباشد که دقیقا مجموع مهارت ها از اول بی نقص بوده باشد، پس مجموعه مهارت های موردنظر میتواند با گذشت زمان و تکامل نیاز های توسعه محصول ،تکامل یابد. بنابراین مهم و ضروری است که یک محیطی داشته باشیم که در ان اعضای تیم، مداوم در حال یادگیری و اموزش باشند و به تجربه های شان بیفزایند. ( که کلا شامل حوزه دانش، دانش تکنیکی، مهارت های فکری و سایر توانایی ها میشود.) مدیریت باید اعضای تیم را در این راستا کمک کند. ( از نظر جنبه های مدیرتی منظورمه)

حالا ایا اکی هست که یک متخصص تمام معنا در تیم داشته باشیم؟

فرض کنیم شخص علی یک طراح UX هست ولی این تمام کاری است که او میتواند انجام دهد. و چون ما تعداد بسیار کمی طراح یو ایکس داریم ، بجز این کار ، از علی وظیفه و انتظار دیگری نداریم. حتی نیاز به تجربه اش در تیم داریم . ولی فقط میتوانیم حدود 10% از تایم او را در کار با تیم پر کنیم. در این موارد یک سولوشن واضح این است که تایم علی را بین تیم های مختلف پخش کنیم. البته در این صورت علی شکسته میشود و میتواند به راحتی یک bottle neck شود. باتوجه به فصل 3 که هدف ما نباید جوری باشد که 100% وقت افراد گرفته شود، به جای آن بیشتر باید نگران کار های بیهوده بود که زمانی رخ میدهد که ما به یک منبع بیش از حد استفاده شده ، بیش از حد اعتماد میکنیم. بنابراین ما ممکن است علی را به عنوان یک متخصص به تعداد معقولی از محصولات، اختصاص دهیم. ولی نه زیاد چون او دلیل و باعث موفقیت یا شکت است . ( جمله کتاب این بود : he is the cause of baton dropping)

از سوی دیگر ، از انجایی که هدف ما دستیابی به جریان خوب با یک تیم است که مهارت های تی شکل گسترده ای دارد، ما باید علی را تشویق کنیم تا دانش تخصصی خود را به سایر اعضای تیم هم اموزش دهد تا دیگر نیازی به اعتماد زیاد به متخصصان نداشته باشیم.

به طور خلاصه هدف این است که یک تیم با اعضایی که دارای مهارت های مناسب جهت کاور کردن زمینه های تخصصی اصلی و به طور کلی دارای برخی همپوشانی ها در مهارت ها ( برای انعطاف بیشتر) هستند، تشکیل دهیم ( داشته باشیم.) برای دیستیابی به این هدف، بیشتر اعضای تیم باید مهارت های تی شکل داشته باشند ولی باز هم متخصصانی در تیم ها حضور دارند.

**. Musketeer Attitude4 (نگرش تفنگدار)**

اعضای تیم توسعه (و به طور کامل اعضای تیم scrum) به داشتن 3 ویژگی مشترک به عنوان musketeer نیاز دارند. "همه برای یکی و یکی برای همه"

این نگرش musketeer، این نکته را تقویت می­کند که اعضای تیم به درستی مسئولیت گرفتن شغل را دارند. آن­ها به عنوان یک تیم برنده شده و یا به عنوان تیم می­بازند.

در یک scrum team دارای عملکرد خوب، من هرگز از هیچ کس انتظار ندارم که بگوید: من قسمت خودم را انجام دادم. شما قسمت خود را انجام ندادید. بنابراین ما شکست می­خوریم.این نگرش این نکته را ازدست می­دهد که تمام اعضای تیم با همدیگر، در یک قایق مشترک قرار دارند.



اعضای تیم بایستی شکرگزار باشند که باید درکنارهم کار کنند تا به تعهد برسند؛ زیرا اگر شکست بخورند، درنهایت این مورد، مشکل همه­ی آن­ها خواهد بود. داشتن اعضای تیم دارای نگرش musketeer لازمه­ی به دست آوردن یک موفقیت مشترک است.

داشتن اعضای تیم دارای مهارت­های T-shaped، نگرش آن­ها را تقویت می­کند و آن را کاربردی می­سازد؛ زیرا مردم در کار مهارت بیشتری نسبت به نوعی وظیفه دارند. از هیچ یک از این تیم ها انتظار نمی رود تا یکی از اعضای توانای آن ­ها بگوید: این شغل من نیست.

با این حال از آن­جایی که همبشه ممکن نیست که یک نفر همه­ی کارها را انجام دهد، من ممکن است بشنوم که کسی می­گوید: من برای انجام این کار توانا نیستم. در این مورد، تیم می­تواند انتخاب کند که یک فرد بدون مهارت را به عنوان شاگرد همراه با یک فرد دارای مهارت داشته باشد تا در آینده توانایی­های متراکم بزرگتری داشته باشد.

حتی اگر محدودیت مهارت افراد، مانع از کار عملکردی متقابل آن­ها شود، اعضای تیم هنوز می­توانند کار آن­ها را سازماندهی کنند تا یک جریان خوب را از طریق با حداکثر سرعت دویدن تضمین کند، به طوری که هیچ یک از اعضای تیم بار بیش از حد به دوش نکشد. برای مثال، برگزاری تمام آزمایش کار تا پایان با حداکثر سرعت دویدن به طوری که "تست کننده" می تواند کار را انجام دهد قطعا نسخه ای برای شکست است. فصل 20 را برای بحث عمیق تر ببینید که چگونه تیم باید جریان در طول اجرا با حداکثر سرعت دویدن را مدیریت کند.

بنابراین با نگرش Musketeer هیچ کس تنها "همراهی برای سواری" نیست. هریک از اعضای تیم مسئول هستند تا این اطمینان را ایجاد کنند که در تمام زمان ها استخدام شده است.

این موضوع اغلب به معنای صحبت کردن و درگیر شدن در فعالیت های خارج از تخصص ویژه خود برای اضافه شدن به بحث­های گوناگون است. برای مثال، اگرچه ممکن است تخصص هریک از اعضای تیم تست شود، اگر فرد فکرکند مشکلی در طراحی تیم با آینده­ای معین درحال به وجود آمدن است، وظیفه­ی اوست که آن را بیان کند، به جای اینکه بی اعتنایی کند و بگوید: این کار من نیست. آن­ها به هرحال بهتر از من می­دانند.

**High-Bandwidth Communications5. (ارتباطات با پهنای بالا)**

اعضای تیم توسعه به برقراری ارتباط با یکدیگر، همچنین با صاحب محصول و ScrumMaster، در شیوه ای با پهنای باند بالا نیاز دارند که در آن، اطلاعات ارزشمند، به سرعت و کارآمد با حداقل سربار (overhead) مبادله شوند.

ارتباطات High-Bandwidth به طور همزمان تکرار و کیفیت انتقال اطلاعات را افزایش می­دهد. درنتیجه تیم scrum، فرصت­های بیشتری برای رسیدگی و سازگاری دارند تا به تصمیم گیری بهتر و سریع تری منجر شود؛ زیرا ارزش اقتصادی اطلاعات، حساسیت زمان است. افزایش سرعت اشتراک اطلاعات به تیم اجازه می­دهد تا ارزش خود را به حداکثر برساند. با بهره برداری سریع از فرصت های حاضر و به رسمیت شناختن موقعیت های هدر رفته، تیم می­تواند با پایین بردن اشتباهات، از خرج کردن منابع بیشتر خودداری کند.

موارد زیر، تعدادی از راه­ها هستند که تیم می­تواند با استفاده از آن­ها، ارتباطات با پهنای بالا را به دست آورد:

مانیفست چابک (بک و همکارانش ۲۰۰۱) بیان می کند که ارتباط رو در رو یک رویکرد ترجیحی است. قطعا اعضای تیمی که به صورت فیزیک جدا شده­اند و یا در درجه اول از ارتباطات بدون تاثیر بر یکدیگر (مانند مدارک) استفاده می­کنند، اشکالی برای مرتب کردن (collocate) اعضای تیم استخدام شده در زمان واقعی، همکاری چهره به چهره است. (face to face)

هرگاه که ممکن باشد، من ترجیح می­دهم اعضای تیمم مرتب (collocate) شوند. به هرحال، بسیاری از سازمان­ها برای دلیل­های متنوع تجارت، تیم­های توزیع شده را خلق کرده­اند؛ بنابراین مرتب سازی همیشه ممکن و یا کاربردی نخواهد بود. من با بیزاری از تیم­های توزیع شده که مزایای ارتباطات با پهنای بالا را به دست آورده بودند، کار کرده ام. بنابراین رو در رو بودن تنها راه رسیدن به هدف نیست؛ اما جایگاهی عالی برای شروع است اگر وضعیت­های تجارت اجازه دهد.

برای تیم­های توزیع شده، سطح معینی از حمایت تکنولوژی می­تواند به بهتر شدن ارتباطات با پهنای بالا کمک کند. من با سازمان­ها کار کرده بودم که اعضای تیم در آن­ها به طور گسترده ای توزیع شده بودند. من از طریق استفاده از بعضی تکنولوژی­های تلفنی با تاثیر کمتر، در بحث­هایی شرکت کردم که احساس می­شد همه در آن بزرگ شده اند. آیا این به اندازه نظم و ترتیب خوب است؟

نه. اما تکنولوژی راه بلندی را طی می­کند تا ارتباطات با پهنای بالا را بین اعضای تیم بهتر کند.

داشتن تیم های تشکیل شده از اعضای تیم متقابل، یک گام بحرانی به سمت دستیابی به ارتباطات با پهنای باند بالاست. چنین تیم هایی کانال های ارتباطی ساده تری دارند؛ زیرا آن­ها دسترسی آسانی به مردمی دارند که نیاز دارند کار خود را تمام کنند. همچنین چنین تیم­های متنوع متقابلی، به احتمال کمتری handoff های رسمی (که معمولا به صورت اسناد نوشته شده) از تیمی به تیم دیگر خواهند داشت. با هریک از افراد در یک تیم، فرکانس و تشریفات handoffs کاهش می یابد که باعث بهبود ارتباطات می شود.

همچنین ما باید زمان سپری شده برای تشریفاتی را کاهش دهیم که در آن­ها، اعضای تیم مراحل مختلفی را اجرا می­کنند که ارزشی نداشته یا ارزش کمی دارند. برای مثال، اگر اعضای تیم مجبورند از طریق سه سطح غیرمستقیم قبل از اینکه با یک مشتری واقعی و یا کاربر صحبت کنند، عمل کنند، تشریفات صحبت با یک مشتری احتمالا مانعی جدی برای ارتباطات با پهنای بالاست.

اجبار به خلق کردن مدارک بی ارزش یا دارای ارزش کم و نیاز به تصویب طولانی و بالقوه غیرضروری و روش های ثبت نام، پهنای باند را کاهش می­دهد. ما باید این موانع را شناسایی کنیم و از بین ببریم تا عملکرد ارتباطی کلی تیم کلی را بهبود ببخشیم .

6. **Transparent Communication (ارتباطات شفاف)**

علاوه بر بالا بودن پهنای باند (سریع و کارآمد با حداقل سربار) ، ارتباطات درون تیم باید شفاف باشد. ارتباطات شفاف فهم تمیزی را از آنچه دقیقا اتفاق افتاده است فراهم می­کند تا از اتفاقات غیرمنتظره جلوگیری شود و کمک می کند تا بین اعضای تیم اعتماد ایجاد شود. من همیشه احساس کرده ام که تیم ها باید به شیوه ای ارتباط برقرار کنند که با اصل حقیقت با حداقل تعجب هم ردیف شود. به عبارت ساده تر، مردم باید به گونه ای با یکدیگر ارتباط برقرار کنند که کمترین احتمال تعجب را داشته باشد. به عنوان مثال ، به یاد دارم که در یک تیم اسکرام که آن را مربی گری کردم، یک فرد خاص به طور مداوم کلمات خود را در طول scrums روزانه انتخاب می­کند تا به چیزی که انجام داده و کاری که برای انجام آن برنامه ریزی می­کرد، برسد. اغلب مردم شگفته زده بودند تا بعدا یاد بگیرند که ارتباطات آن فرد عمدا غیرشفاف بوده و برای گمراهی طراحی شده بود. این امر منجر به اعتماد نکردن دیگر اعضای تیم شد.

به جای آن، مانع از توانایی تیم در سازماندهی و رسیدن به اهداف با حداکثر سرعت شد.

**Right-Sized7. (راست اندازه)**

Scrum به تیم­های کوچک توجه می­کند. قانون کلی این است که داشتن 5 تا 9 نفر در تیم، بهترین حالت است. تحقیقات منتشر شده ای وجود دارد که از این ادعا حمایت می کند که تیم های کوچک

تمایل به کارآمدترین بودن دارند. (Putnam 1996; Putnam and Myers 1998)

تجربه من در طول 25 سال گذشته این است که تیم های پنج تا هفت نفر، نقطه شیرینی برای ارائه ارزش کسب و کار به سرعت هستند.

مایک کوهن تعداد انگشت شماری از دلایل کوچک نگه داشتن تیم ها را فهرست می کند که شامل موارد زیر می شود (کوهن ۲۰۰۹):

1- اجتماع های کمی با افرادی که وقت خود را تلف می کنند وجود دارد. مردم تلاش کمی به کار می برند زیرا باور دارند که دیگران سستی را جبران می کنند.

2- تعامل سازنده بیشتر در تیم های کوچک رخ می دهد.

3- زمان کمتری صرف هماهنگی تلاش ها می شود.

4- هیچ کس نمی تواند در پس زمینه محو شود.

5- تیم های کوچک بیشتر به اعضای خود رضایت می بخشند.

6- احتمال رخ دادن موارد بیش از حد مضر، کمتر است.

این امکان وجود دارد که یک تیم بیش از حد کوچک باشد. مثلا ً تیمی خیلی کوچک است اگر

افراد لازم برای انجام کار را نداشته باشد، و یا اگرتعداد بیش از حد کمی از افراد داشته باشد که به طور کارآمد عمل می کنند.

فقط به این دلیل که scrum به تيم هاي کوچک علاقه دارد، به اين معنا نيست که ما نميتوانيم از آن برای تلاش های توسعه ای بزرگتر استفاده کنيم.

Scrum اغلب برای ساخت محصولاتی استفاده می شود که به بیش از 9 نفر نیاز دارند. با این حال ، به جای داشتن یک تیم بزرگ اسکرام با ، می گویند ، 36 اعضای تیم توسعه ، ما به جای چهار یا چند تیم اسکرام ، هر کدام با یک تیم توسعه از 9 یا کمتر نفر داشته باشد. به هرحال، به جای گفتن داشتن یک تیم scrum بزرگ، بگویید 36 اعضای تیم توسعه یافته. ما می توانیم به جای آن، 4 تیم scrum یا بیشتر داشته باشیم که هرکدام با یک تیم توسعه با 9 نفر یا کمتر داشته باشد.

مقیاس پروژه اسکرام نه با داشتن یک تیم توسعه بزرگتر، بلکه با داشتن چند تیم اسکرام انجام می شود. تیم های متعدد اسکرام می توانند در راه های متنوعی با یکدیگر هماهنگ شوند. یک رویداد مشترک به نام scrum of scrums شناخته می شود که جایی است که در آن، اعضای تیم هر scrum، دورهم جمع می شوند تا یک scrum روزانه معادل با سطح بالاتر اجرا کنند.(برای جزئیات بیشتر فصل 12 را مشاهده کنید)

**Focused and Committed8. (متمرکز و متعهد)**

اعضای تیم باید روی هدف تیم متمرکز و متعهد باشند.

تمرکز یعنی هریک از اعضای تیم باید متعهد شوند، روی آن تمرکز کنند و توجه خود را به هدف تیم اختصاص دهند.

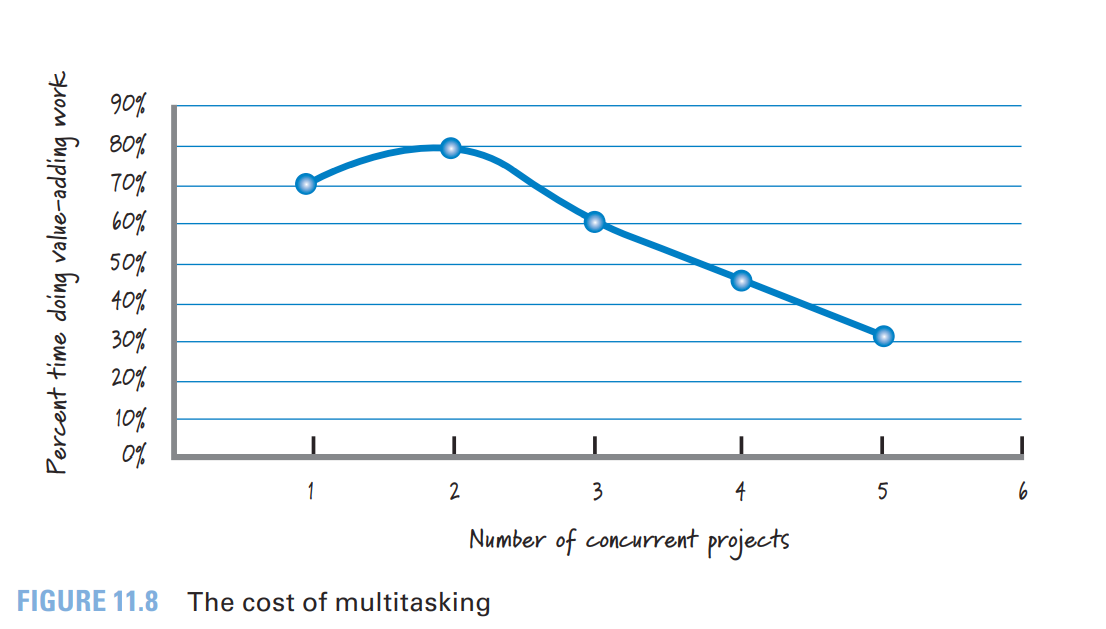
تعهد یعنی در طول زمان های خوشی و بدی، هریک از اعضای تیم برای نشست هدف مشترک تیم اختصاص داده شوند.

اگر کسی تنها برروی یک محصول کار کند، کار آن بسیار راحت تر از فردی است که متمرکز و متعهد است. هنگامی که درخواست می شود یک نفر به طور همزمان بر روی تلاش های توسعه محصول متعدد کار کند، ان فرد باید زمان خود را در سراسر آن محصولات تقسیم کند و تمرکز و تعهد خود در تمام محصولات را کاهش دهد.

از یک فرد که برروی محصولات متعددی کار می کند، درباره تمرکز و تعهد آن بپرسید و شما احتمالا پاسخی مشابه این را می شنوید: من کارهای زیادی برای انجام دادن دارم، به طوری که من تنها تلاش میکنم تا کارها را به بهترین شکل انجام دهم. من می توانم روی یک محصول کارکنم و روی محصول بعدی بپرم. من هرگز احساس نمی کنم زمانی دارم تا بتوانم روی یک محصول تمرکز کنم و آن را به خوبی انجام دهم. اگر زمانی یک موقیعت اورژانسی برای چند محصول پیش آید، من به سادگی نمی توانم به همه آن ها کمک کنم.

این موضوع برای اعضای تیم سخت تر است که تا کار خود را با کیفیت خوب انجام دهد وقتی از محصولی به محصول دیگر می پرد. همچنین سخت تر است. به جای بودن در یک قایق با اعضای تیم، تیم دارای وظایف متعدد از قایقی به قایق دیگر حرکت می کند. اگر تعداد زیادی قایق به طور همزمان سوراخ شوند، فرد چگونه می تواند انتخاب کند کدام یک از خدمه کشتی به کمک احتیاج دارد؟ اگر فردی آنجا نباشد که آب را منتقل کند، اعضای تیم برای آن تیم متعهد نیستند. در بهترین حالت آن فرد با تیم درگیر است. برای عادل بودن با دیگر اعضای تیم درگیر در تیم باید ان را به خوبی شفاف سازی کرد که آن فرد تنها برای این زمان درگیر بوده و ممکن است در زمان های بحرانی بعدی حضور نداشته باشد.

داده های قابل توجهی برای حمایت از این باور به طور گسترده برگزار شده که نشان می دهد قرار گرفتن بر روی محصولات متعدد (یا پروژه ها) یا تیم های متعدد بهره وری را کاهش می دهد.



این داده نشان می دهد که هیچ کس به طور 100 درصد سازنده نیست. این تنها یک overhead است تا یک شهروند شرکتی خوب باشد. درواقع بهره وری با دو پروژه بهتر از یک پروژه به نظر می رسد. این اتفاق می افتد زیرا ممکن است در یک پروژه متوقف شود. بنابراین داشتن مورد سوم برای تعویض به فرد اجازه می دهد تا به طور فزاینده ای موثر باشد.

براساس این داده، کار کردن روی سه پروژه یا بیشتر به صورت همزمان، یک انتخاب بد اقتصادی است؛ زیرا زمان بیشتری صرف هماهنگی، به یاد آوردن، و ردیابی اطلاعات و زمان کمتری صرف انجام کار برای ارزش افزوده می شود. بنابراین فرد باید چند پروژه و محصول (و احتمالا تیم های مختلف) را به صورت همزمان داشته باشد؟ احتمالا نه بیشتر از دوتا.

من یک ترجیح قوی برای یکی دارم. زیرا در این دنیای قوی ارتباطات امروزه با ایمیل، پیامرسانی فوری، توییتر، فیس بوک و سایر فناوری های دیگر، بودن یک شهروند خوب احتمالا معادل بودن در یک پروژه است.

حالا برای آن دسته از متخصصان که ممکن است به اختصاص دادن همزمان به چند محصول نیاز دارند، چگونه است؟ پیش از این از مثال علی (طراح UX) استفاده کردم که 10 درصد زمان خود را به تیم اختصاص داده بود.(با بقیه وقتش برای رفتن به تیم های دیگر)

به همان اندازه که دوست داریم مانند علی روی یک یا دو محصول کار کنیم، چطور است برنامه زمانی او را برای 5 دقیقه ببینیم؟

به عنوان یک رویکرد عملی، اجازه دهید متخصص تصمیم بگیرد که فرد روی چند محصول می تواند به طور همزمان تمرکز و تعهد کند. اگر فرد گفت نمی تواند روی بیش از یکی متعهد شود، او را به محصول بعدی یا تیم ارجاع ندهید. اگر از دیدگاه کسب و کار، ما از تصمیم فرد که محصول دیگری را برنداشته، ناراحت هستیم، (اجازه دهید بگوییم علی با تنها سه محصول به صورت همزمان راضی بود) ، شاید ما باید به دنبال یک راه حل جایگزین برای این مشکل باشیم.

در اینجا چندمورد را بررسی می کنیم:

1- تعدا کمی پروژه را به صورت همزمان انجام دهید. این مورد قطعا راه حل درست است؛ زیرا سازمان های زیادی انتخاب کردند که تا تعداد زیادی پروژه را در یک زمان شروع کنند (فصل 16 را ببینید).

2- راه حل دیگر این است که متخصص های بیشتری را استخدام کنیم تا بار مسئولیت تقسیم شود

3- کمک وسیع به سایر افراد تا مهارت های تخصصی را به آن ها بیاموزیم.

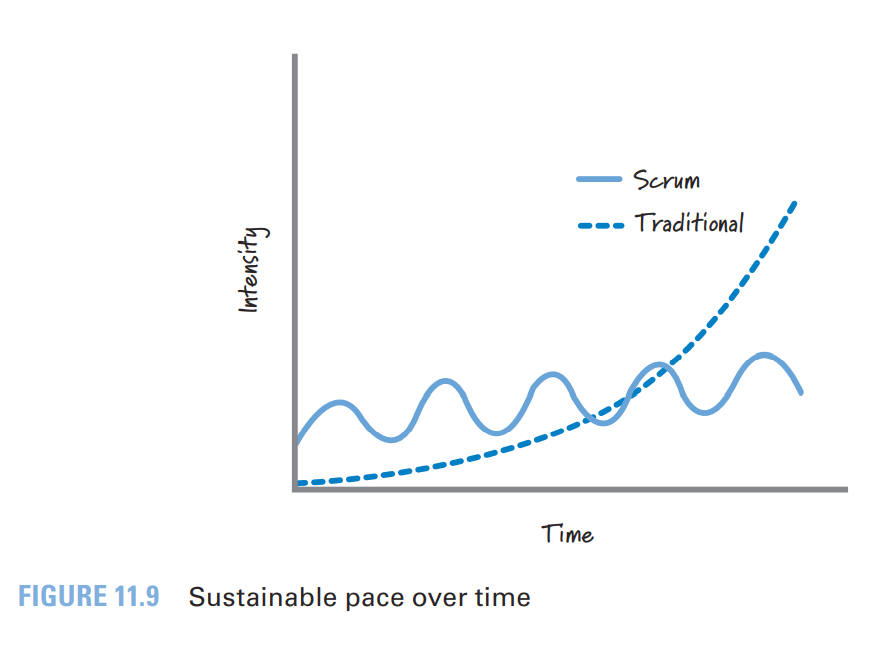
4- راه حل چهارم ترکیبی از سه راه حل قبلی است.

درنهایت وادار کردن افراد به کار کردن روی تعداد زیادی پروژه یا تیم به صورت همزمان، تمرکز و تعهد را کاهش داده و درامد تجاری را به خطر می اندازد.

**Working at a Sustainable Pace9. (کار با سرعت پایدار)**

یکی از اصول هدایت کننده scrum این است که اعضای تیم باید در یک سرعت پایدار کار کنند. (دیگر راهمپیمایی مرگ نیست). با این کار محصولات در سطح جهانی را تحویل می دهند و از نظر حفظ محیط زیست سالم و سرگرم کننده است.

با استفاده از توالی توسعه ، فعالیت های مهمی مانند یکپارچگی را خنثی می کنیم و تست تا نزدیک به پایان، زمانی که به طور معمول حجم کار خرد کننده از مسائل وجود دارد به با نزدیک شدن به تاریخ تحویل برخورد کنید. نتیجه افزایش شدید در شدت در فازهای اخیر است.



این زمان شدید ( تحت فشار) نمایان میشود با قهرمانانی که که شبانه روز کار میکنند و حتی در روز های تعطیل هم میخواهند که خروجی را release کنند. برخی مردم در این نوع از کار پیشرفت میکنند ، توجه را دوست دارند و میخواهند که برای این تلاش مضاعف شان پاداشی دریافت کنند. استرس بر دیگران بسیار طاقت فرسات . به عنوان یک سازمان باید از خود بپرسیم : چرا ما این همه حتی اخر هفتها یا شب ها باید کار کنیم و چه چییزی را باید عوض کنیم؟

هنگام استفاده از اسکرام، جایی که ما به طور مداوم در حال توسعه، ازمایش و یک پارچه سازی ویژگی های کاری هستیم، با مشخصات معمول قابل مقایسه نیست. در طی هر اسپرینت اعضای تیم باید از تکنیک های عملی خوب مثل بازسازی، یک پارچه سازی مداوم و تست های اتوماتیک استفاده کنند تا مطمئن شوند که میتوانند خروجی مطلوب بدون فشار زیاد (و خودکشی) تحویل دهند.

بنابراین در یک اسپرینت داده شده، ما موقع نزدیکی به انتهای اسپرینت، فشار بیشتری را لمس میکنیم که مطمئن شویم که همه امور با تعریف قوی ما و انچه قابل انتظار است، مرتبط و هماهنگ است. با این حال، شدت کلی کار در طی هر اسپرینت به طور تقریبی مرتبط و وابسته است به فشاری که در اسپرینت قبلی تحمل کرده ایم.

نتیجه کار ، تسطیح کار است. در قسمت های بزرگ یا انفجار های بزرگ کار نمی اید؛ بخصوص دیر که مضر تر است. این تسطیح به این معنی است که تیم های اسکرام ساعات اضافه کاری کمتری خواهند داشت و لذا کمتر کار میکنند.

10. **Long-Lived (عمر طولانی)**

استفاده موثر از اسکرام به تیم هایی نیاز دارد نه گروه ها.

یک تیم تشکیل شده از یک مجموعه ی متنوع و cross-functional از افراد هماهنگ با هم که با هم در یک vision (چشم انداز) یکسان همشو شده باشند و باهم کار کنند تا به ان چشم اندازه دست یابند.

یک گروه به مجموعه ای از افراد گفته میشود با یک لیبل مشترک است. اعضای یک گروه بجز اشتراک نام گروه ، موارد دیگری به اشتراک نمیگذارند و به طور موثر مسئولیت هایی را که برای نقش تیم توسعه توضیح دادیم را انجام نمیدهند.

به عنوان یک قانون، تیم ها باید طولانی عمر باشند. و تا جایی که میشود باید تیم را حفظ کرد که باهم باشند. و اقتصاد هم برای تیم های طولانی عمر مطلوب است. ( ازنظر اقتصادی یعنی خوبه که تیم عمر طولانی ای داشته باشه) . تحقیقات نشان میدهند که تیم های طولانی عمر عملکرد بهتری نسبت به گروه های تازه شکل گرفته دارند. بنابراین ، ثابت شده است که صمیمیت درون تیمی میتواند روی خروجی تیم اثر بگذارد. و خروجی خوب و با کیفیت و کارامد منجر به نتایجی خوب در بیزنس میشود.

اگر ما با یک تیمی کار کنیم که تا به حال با هم کار نکرده اند، باید زمان و هزینه بیشتری صرف کنیم تا به یک تیم کارامد و عالی تبدیل شوند. بیشتر گروها نیاز دارند که انتقال از طریق فاز ها، مثل forming ( تشکیل ) و storming ( هنجار ایجاد کردن ) به تیم های بسیار کاربردی تبدیل شوند. زمانی که یک تیم با عملکرد خوب داشته باشیم، یک بیزینس واقعی داریم. که اعضای ان میدانند که چگونه باید باهم کار کنند و اعتماد یکدیگر را به دست اورده اند. علاوه بر این، تیم اطلاعات مهم تاریخی مثل team’s velocity را گرداوری کرده است . ☹

اگر تیم را منحل کنیم یا ترکیب ان را به میزان زیادی عوض کنیم، این سابقه کار باهم و اطلاعاتی مثل velocity ها دیگر بطور مستقیم قابل استفاده نیست.

گاها سازمان ها در ارزیابی ارزش دارایی تیم ها کوتاهی میکنند. بیشتر سازمان ها مهارت ها و پروسس هایشان را افزایش دادند تا شکل تیم به اعضای خود بدهد. درواقع رواج agile با تیم است. در حقیقت ، یکی از ارزش های هسته ای بیانیه ی اجایل این است : افراد و تعاملات. به عبارتی دیگر، تیم یک دارایی مهم است.

جابجایی افراد از تیمی به تیمی دیگر بی عیبی یک تیم را از بین میبرد. و ورود و خروج افراد از تیم و به تیم ، احتمالا به صداقت و اعتماد و درستی کار لطمه میزند.

بیشتر سازمان ها این سیاست را قبول کرده اند که حداقل هسته تیم هایشان را تا جای ممکن برای تولد محصولات مضاعف ( از یک محصول به محصول دیگر) حفظ کنند. از نظر اقتصادی، جابجایی یک تیم خوب و well-formed بهتر و بهینه تر از زمانی است که افراد را از تیم به تیم جابجا کنیم.

این به این معنا نیست که باید همیشه همیشه تیم را برای مدت زمان طولانی حفظ کرد. مثلا اگر ما یک تیمی داشته باشیم که انطور که ما انتظار داشتیم با هم match نشده باشند، یا مثلا از نظر عملکردی با هم خوب نباشند( یعنی خروجی مشکل داشته باشه مثلا) ، در این صورت بهتر است که نسبت به تغییر تیم اقدام کنیم.

در یک مورد دیگر، میتوان اگاهانه تیم اسکرام با عملکرد بالا را به عنوان بخشی از استراتژی جدا کنیم تا یک تیم دیگه را گسترش دهیم . (ولی مثلا اگر تیم کارش تمام شده باشد، خب اعضای تیم را باید برای پروژه های بعدی به تیم های جدید اختصاص دهیم.) به جای آن ، ما تیم را تقسیم میکنیم چون به این باور داریم که این ارزش بیشتری دارد که 6 تیم اسکرام جدید تشکیل شود که در هرکدام از انها یک متخصص وجود داشته باشد ، نسبت به اینکه یک تیم حرفه ای داشته باشیم.

در اخر چون تیم ها دارایی ما هستند، انها واحد ظرفیتی ای هستند که باید از انها استفاده کنیم تا پایه گذاری کنیم محدودیت مناسب wip ( work in proccess) در تعداد و انواع تلاش هایمان برای توسعه محصولات. ( این تلاش باید مداوم باشد.) ( کلا مهم نیست این بند اخر ، تو اسلایدا هم که نبود اصلا)

خلاصه فصل:

در این فصل درمورد نقش تیم توضیح دادیم. تاکید کردیم که چگونه تیم ایتم های پروداکت بک لاگ را عملی و اجرا میکند. همچنین گفتیم مسئولیت های تیم را در هر اسپرینت و 10 خصوصیت قابل انتظار از تیم. به هرحال توقع است که تیم به خوبی برنامه ریزی و تقسیم وظایف داشته باشند و عملکرد های متنوع و مهارت های کافی تا بتوانند کار را به خوبی تمام کنند. کار داده شده به تیم باید انجام شود ان هم با ترکیبی از مهارت های تی شکل موثر در موقعیت های انبوه و حجم بالای کاری. اگر مردم هنوز در تجربه های شان عمیق نباشند، لازم است که به تجربه های شان بیفزایند تا این عمق در تجربه ایجاد شود. همچنین از اعضای تیم درخواست میشود که همگی با هم باشند ( همه برای یکی یکی برای همه) . تیم ها ایجاد شده اند که ارتباطات گسترده (؟) به عمل بپیوندد. و خب کلا ترجیح بر تیم ها با تعداد کم است. برای حفظ ارتباط و تمرکز روی کار، ترجیح میدهیم که اعضای تیم روی یک یا دو تلاش توسعه ای در یک تایم کار کنند. و قطعا دنبال این هستیم که یک تیم بتواند به مددت طولانی به کارشان باهم ادامه دهند.