Agile Principles(III)

Dr. Elham Mahmoudzadeh

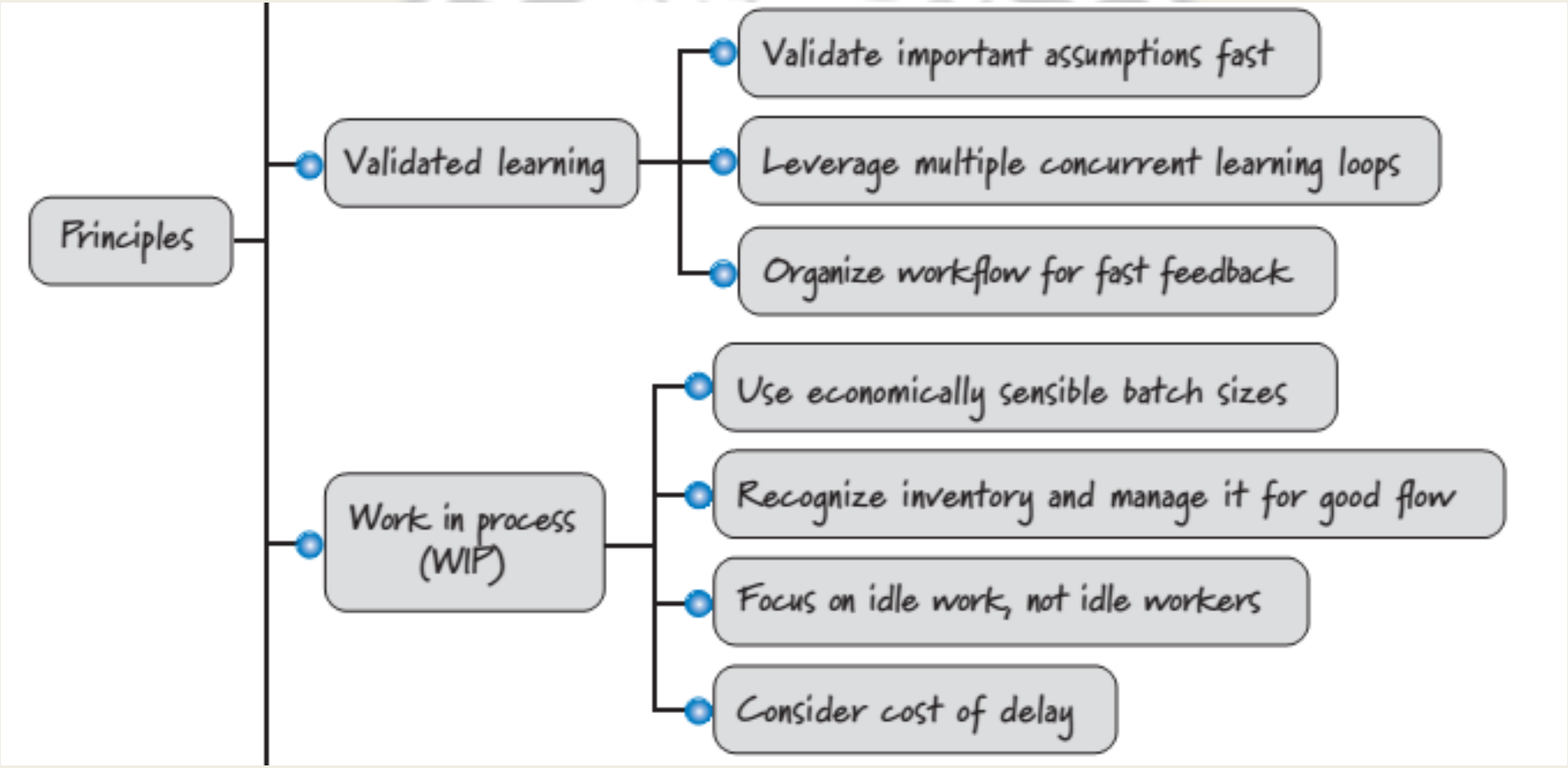
Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

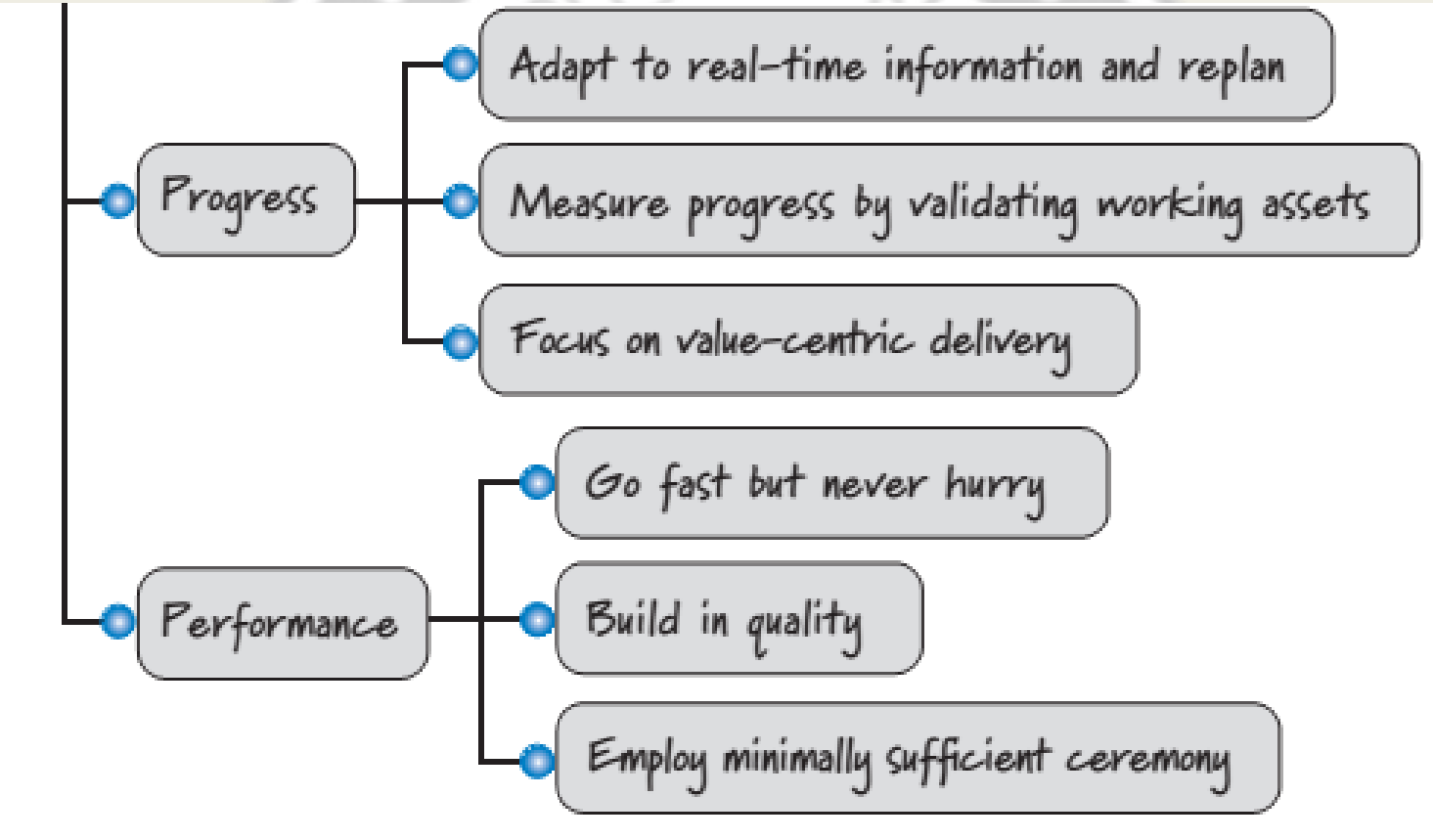
2022

Categorization of principles (Up)

Categorization of principles (Middle)



Categorization of principles (Bottom)



Prediction and Adaptation

- In Scrum, we are constantly balancing the desire for prediction with the need for adaptation.
- Five related principles
  1. *Keep options open.*
  2. *Accept that you can't get it right up front.*
  3. *Favor an adaptive, exploratory approach.*
  4. *Embrace change in an economically sensible way.*
  5. *Balance predictive up-front work with adaptive just-in-time work.*

Keep Options Open (I)

- Plan-driven, sequential development requires that important decisions in areas like requirements or design be made, reviewed, and approved within their respective phases.

- Furthermore, these decisions must be made before we can transit to the next phase, even if those decisions are based on limited knowledge.

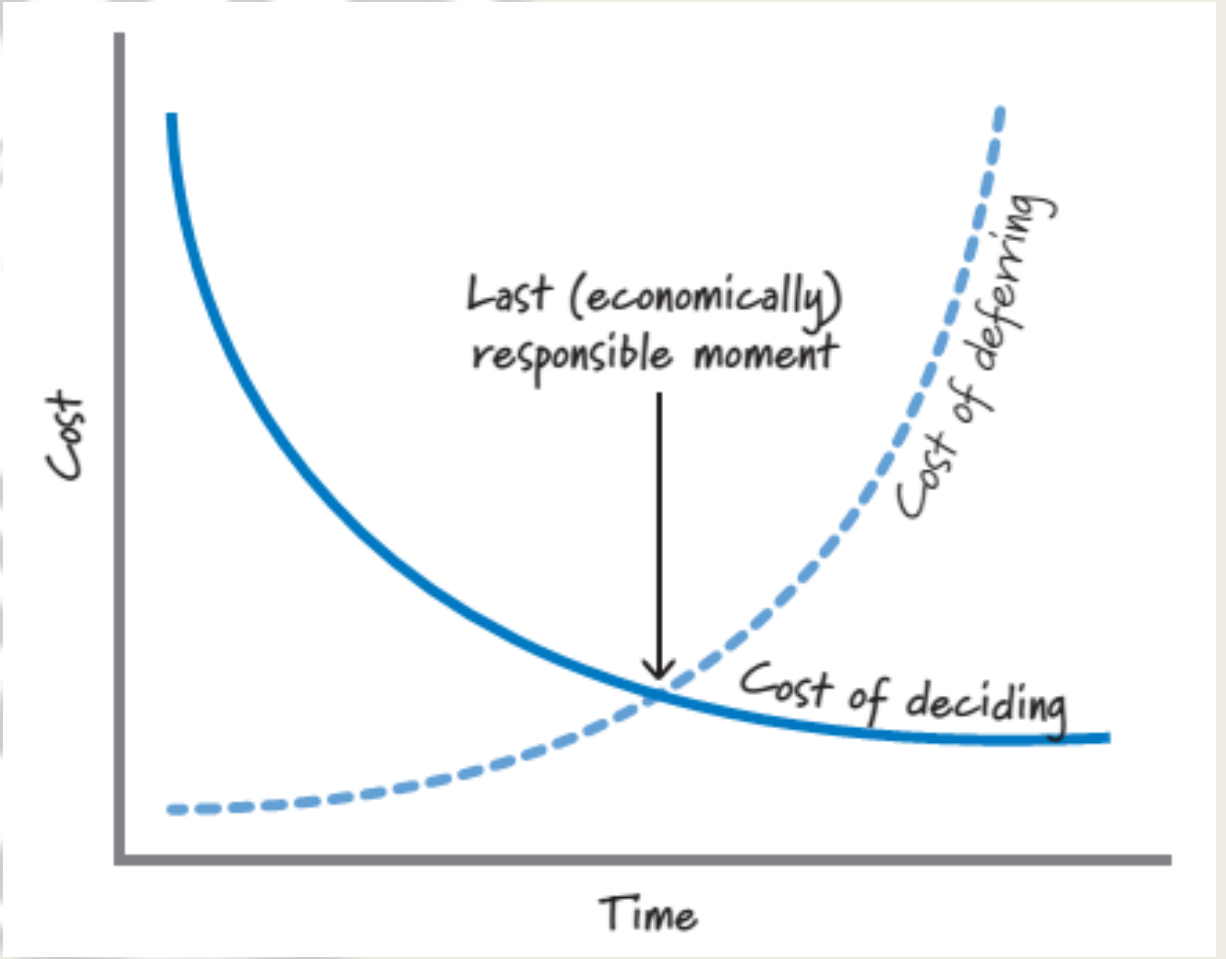
-
یکی از نقاط ضعف Plan-driven ها اینه که اول کار که ما داریم ریکوارمنت هارو دریافت میکنیم درسته که همه تلاش رو روی این می ذاریم که نیازمندی ها رو کسب بکنیم و با جزئیات همه چیز رو متوجه بشیم ولی یکی از نقاط ضعفش اینه که ما داریم تصمیمات مهم رو داریم وقتی می گیریم که مینیمم اطلاعات رو داریم ینی اول پروژه اطلاعات ما نسبت به فازهای بعدی کمتر است --> تصمیمات مهم ما وقتی اتفاق می افته که قبل از اینکه وارد مرحله اصلی کار بشیم با حداقل دانش و اطلاعات تصمیمات مهم می گیریم

Keep Options Open (II)

- Scrum contends that we should never make a premature decision just because a generic process would dictate that now is the appointed time to make one.
- In Scrum, we favor a strategy of keeping our options open.
- Often this principle is referred to as the **last responsible moment (LRM)**, meaning that we delay commitment and do not make important and irreversible decisions until the last responsible moment. LRM is when the cost of not making a decision becomes greater than the cost of making a decision. At that moment, we make the decision.

-
توی اسکرام: زمانی که داریم می ذاریم استراتژیش این است ک ما هیچ وقت تصمیم رو زمانی که اطلاعات لازم و کافی براش نداریم نگیریم --> تصمیم نگیریم تا وقتی که به زمان LRM برسیم
اون زمان کیه؟ زمانی که هزینه تصمیم نگرفتنه بیشتر از هزینه تصمیم گرفتن باشه ینی ما همینطوری به تاخیر بیندازیم و اطلاعات بیشتری به دست بیاریم و فکر کنیم و بحث کنیم و.. تا وقتی که مثلا می رسه که قرارداد رو از دست میدیم یا تا وقتی که ...

Keep Options Open (III)



هزینه تصمیم گرفتن اول کار خیلی زیاد است چون ما حداقل اطلاعات رو داریم و امکان تصمیم گرفتن اشتباه خیلی زیاد است
هرچی جلوتر می ریم هزینه تصمیم گرفتن کم میشه چون برنامه افراد تیم مشخص می شه و... و امکان اشتباه تصمیم گرفتن کمتر میشه و از یه جایی به بعد تقریبا ثابت است
هزینه تصمیم نگرفتن اول کار کم است ولی از یه جایی به بعد هزینه تصمیم نگرفتن نمایی زیاد میشه مثلا انگیزه تیم از دست می ره و کارفرما خسته می شه و زمان از دست می ره و..

پس زمانه مهمه که تا چه موقعی تصمیم نگیریم

پس option ها رو تا وقتی که زمان داریم باز بذاریم برای خودمون که اطلاعات کسب کنیم و با اطلاعات بیشتر تصمیمات منطقی تر بگیریم که ریسک اشتباهات کم باشه

Keep Options Open (IV)

- On the first day of a product development effort, we have the least information about what we are doing.
- On each subsequent day of the development effort, we learn a little more.
- Most of us would prefer to wait until we have more information so that we can make a more informed decision.

Keep Options Open (V)

- When dealing with important or irreversible decisions, if we decide too early and are wrong, we will be on the exponential part of the cost-of-deciding curve.

- As we acquire a better understanding regarding the decision, the cost of deciding declines (the likelihood of making a bad decision declines because of increasing market or technical certainty). That's why we should wait until we have better information before committing to a decision.

-

این تصمیمات مخصوصا اول پروژه خیلی مهم هستند مثلا اگر می خوایم زیر ساخت پروژه رو انتخاب بکنیم پس اگر این تصمیمات همینطوری باشه و بدون در نظر گرفتن شرایط متفاوتی باشه این تصمیمات هزینه های هنگفتی برامون داره و یه موقع هایی رو جبران ناپذیر است پس تصمیمات مهم اهمیت دارند که روش فکر بشه پس وقتی فهم بهتری پیدا کردیم و تا جایی پیش رفتیم که باید دیگه تصمیم رو بگیریم

Accept That You Can’t Get It Right Up Front(I)

- Plan-driven processes not only mandate full requirements and a complete plan; they also assume that we can “get it right” up front.
- The reality is that it is very unlikely that we can get all of the requirements, or the detailed plans based on those requirements, correct up front.
- What’s worse is that when the requirements do change, we have to modify the baseline requirements and plans to match the current reality.

-

اول کار نمی تونیم همه اطلاعات رو داشته باشیم --> مهمترین دلیلش اینه که اول کار ایده ها کلی است مثلا کارفرما اول کار نمیدونه چی میخواد و.. که این مسئله طبیعی است --> بپذیر که اول کار نمی تونی همه اطلاعات رو داشته باشی که این هم یک اصل است و رویکردهای سنتی این واقعیت رو نمی پذیرن ینی میگن اول کار باید همه چیز مشخص باشه

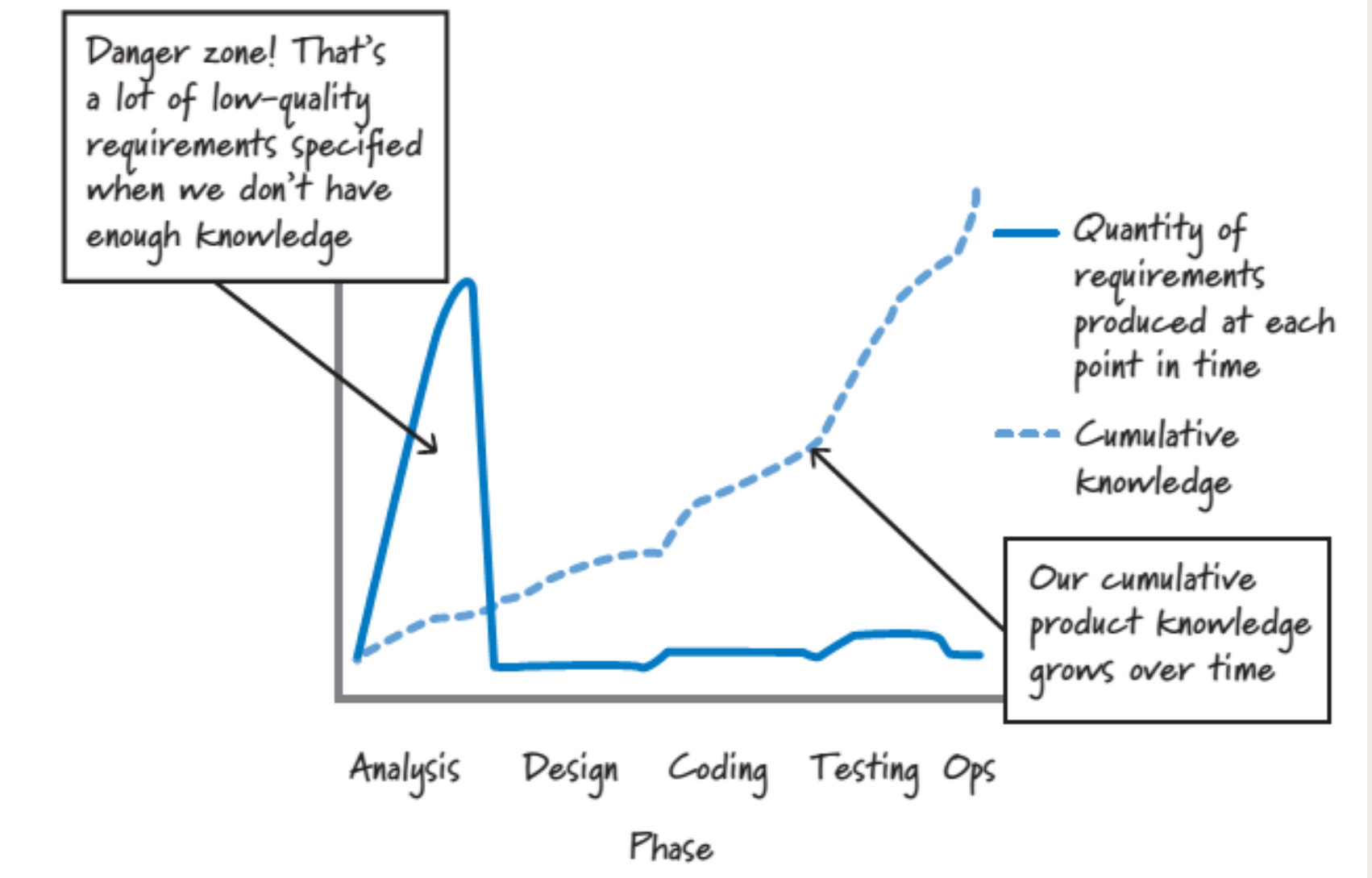
Accept That You Can’t Get It Right Up Front(II)

- In Scrum, we acknowledge that we can’t get all of the requirements or the plans right up front.

- In fact, we believe that trying to do so could be dangerous because we are likely missing important knowledge, leading to the creation of a large quantity of low-quality requirement.

-
توی اسکرام میگه من اینو می پذیرم و قبول میکنیم که همه ریکوارمنت هارو نمی تونم از اول داشته باشم و برمبنای همین واقعیت میاد میگه که اول کار نباید تصمیمات مهم رو بگیریم

Plan-driven requirements acquisition relative to product knowledge



دانش تجمعی

--

Accept That You Can’t Get It Right Up Front(IV)

- With Scrum, we still produce some requirements and plans up front, but just sufficiently, and with the assumption that we will fill in the details of those requirements and plans as we learn more about the product we are building.
- After all, even if we think we’re 100% certain about what to build and how to organize up front the work to build it, we will learn where we are wrong as soon as we subject our early incremental deliverables to the environment in which they must exist. At that point all of the inconvenient realities of what is really needed will drive us to make changes.

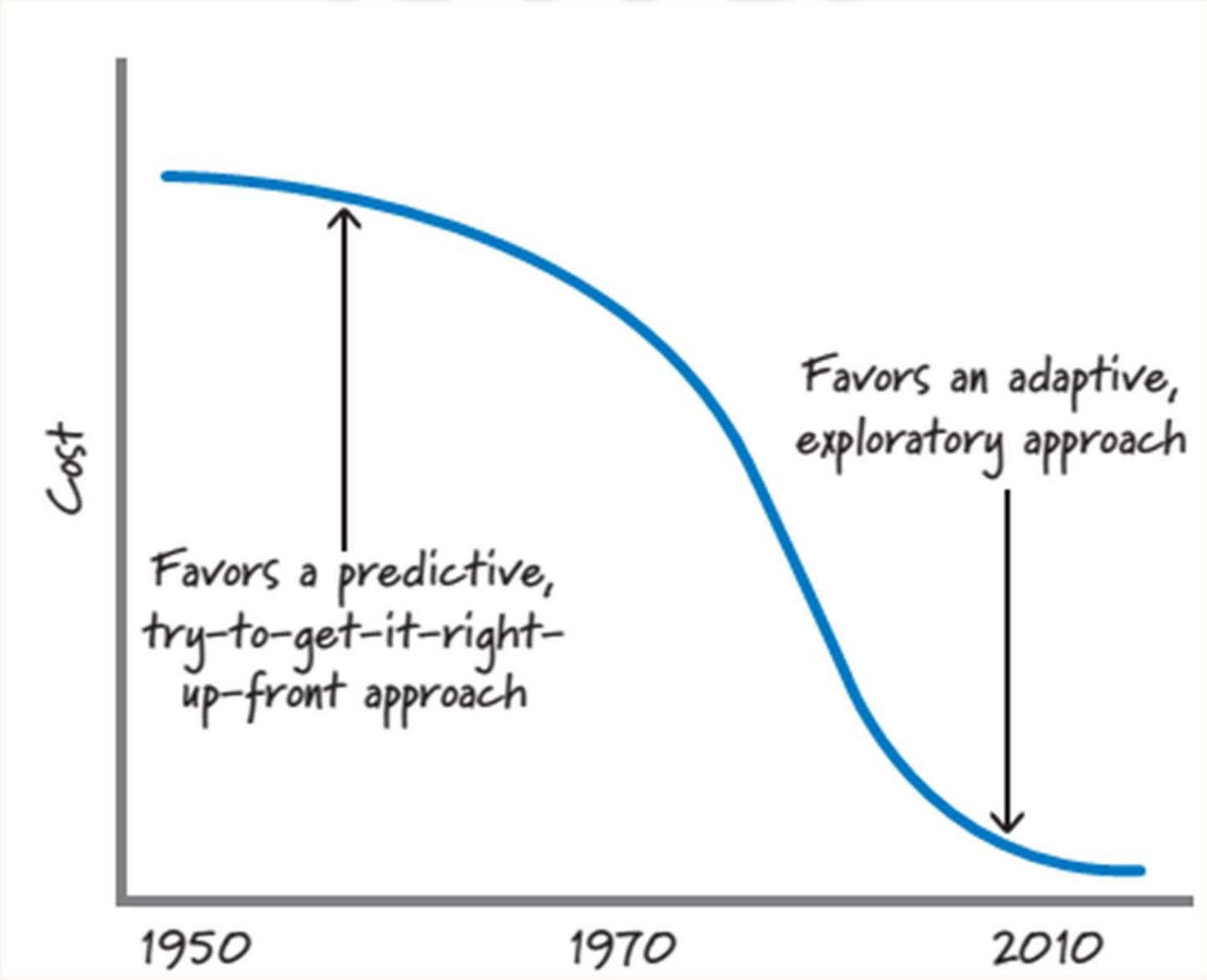
-
توی اسکرام نمیاد اول کار برنامه رو بچینه
اگر ما 100 درصد هم مطمئن باشیم که همه اطلاعات رو داریم ممکنه که increment اولمون که میاد یک اتفاقی بیوفته و با یکسری واقعیت هایی بربخوریم و یکسری نیازمندی هایی تغییر بکنه که نیاز به تغییر داشته باشیم

Favor an Adaptive, Exploratory Approach (I)

- **Exploration** refers to times when we choose to gain knowledge by doing some activity, such as building a prototype, performing a study, or conducting an experiment. In other words, when faced with uncertainty, we buy information by exploring.
- Plan-driven, sequential processes focus on using (or exploiting) what is currently known and predicting what isn't known.
- Scrum favors a more adaptive, trial-and error approach based on appropriate use of exploration.
- Our tools and technologies significantly influence the cost of exploration.

-
وقتی به یه جاهایی برخوردیم که ابهاماتی وجود داره و نمیدونیم چه اتفاقی می خواد بیوفته وقت بذاریم برای کسب اطلاعات
پس کاوش کردن ینی تلاش برای کسب اطلاعات
توی Plan-driven ها تلاششون روی exploit بوده ینی برمبنای ان چیزی که داریم تصمیم میگیریم و ان چیزی را هم که نداریم پیش بینی میکنیم
توی اسکرام میگه اون چیزی که داریم و مطمئن هستیم اوکی هست و اون چیزی هم که نداریم بریم اطلاعاتش رو کسب بکنیم و بعد روش تصمیم بگیریم --> یکی از دلایلی هم که اصل Keep Options Open رو داشتیم همین بود ینی فعلا گزینه هارو باز بذاریم که بریم کسب اطلاعات انجام بدیم
توی اسکرام رویکردش، رویکرد تطبیقی بیشتر است و سعی و خطا بریم جلو و برمبنای اکتشاف که هست و خوبیه این تکنیک این است که ابزارهای امروزی به ما اجازه exploration کردن رو میده
توی اسکرام اگر اطلاعات کافی داری که مطمئنی روش بریم جلو ولی اگر با عدم قطعیت مواجه شدیم به جای اینکه پیش بینی کنیم بیایم جمع اوری کنیم و بحث کنیم ینی explor کنیم که بتونیم تصمیماتی که می گیریم منطقی تر باشه و ریسک تصمیم اشتباه بیاد پایین

Historical cost of exploration



Favor an Adaptive, Exploratory Approach(III)

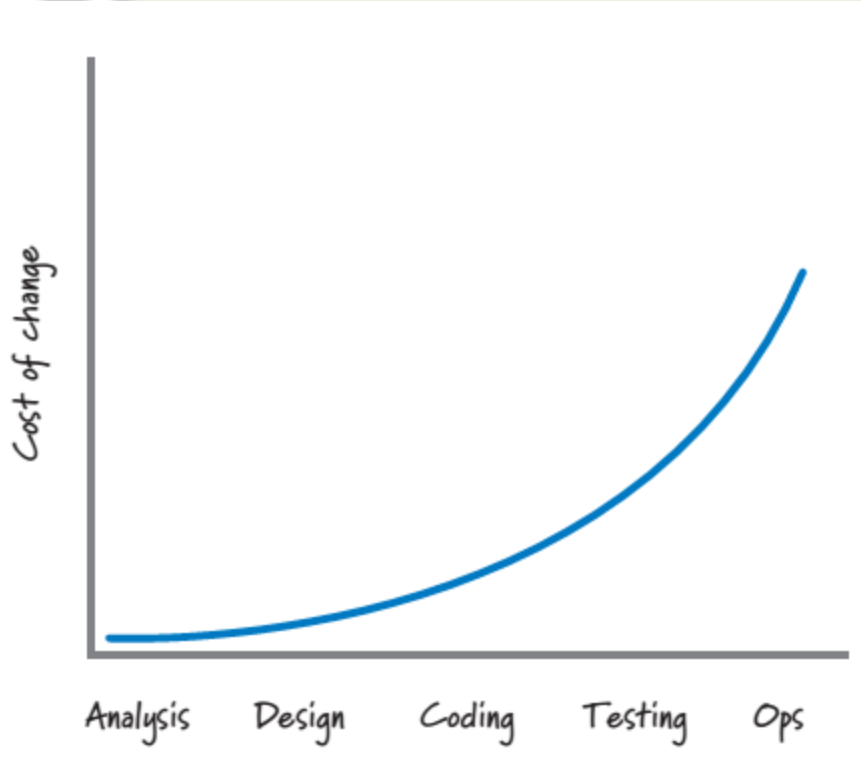
- Tools and technologies have gotten better and the cost of exploring has come way down.
- In fact, nowadays, it's often cheaper to adapt to user feedback based on building something fast than it is to invest in trying to get everything right up front.

Favor an Adaptive, Exploratory Approach(IV)

- In Scrum, if we have enough knowledge to make an informed, reasonable step forward with our solution, we advance.
- When faced with uncertainty, rather than trying to predict it away, we use low-cost exploration to buy relevant information that we can then use to make an informed, reasonable step forward with our solution.
- The feedback from our action will help us determine if and when we need further exploration.

Embrace Change in an Economically Sensible Way(I)

- When using sequential development, change, as we have all learned, is substantially more expensive late than it is early on.
- To avoid late changes, sequential processes seek to carefully control and minimize any changing requirements or designs by improving the accuracy of the predictions about what the system needs to do or how it is supposed to do it.



-
توی رویکرد سنتی تغییر خیلی هزینه بر است و هرچی ما این تغییر رو بذاریم مراحل اخر هزینه ما بیشتر میشه
توی رویکردهای سنتی چون می خوایم از اون هزینه نمایی جلوگیری بکنیم در خواست خود تغییر رو محدود میکنیم ینی می گیم تغییر نباید اتفاق بیوفته چون هزینه بر است پس کنترل میکنیم تغییر ریکوارمنت ها ینی از اول می بندیم با کارفرما که اگر خواستی تغییری بدی اجازه تغییرندی --> محدوده ی تغییر رو کنترل میکنیم

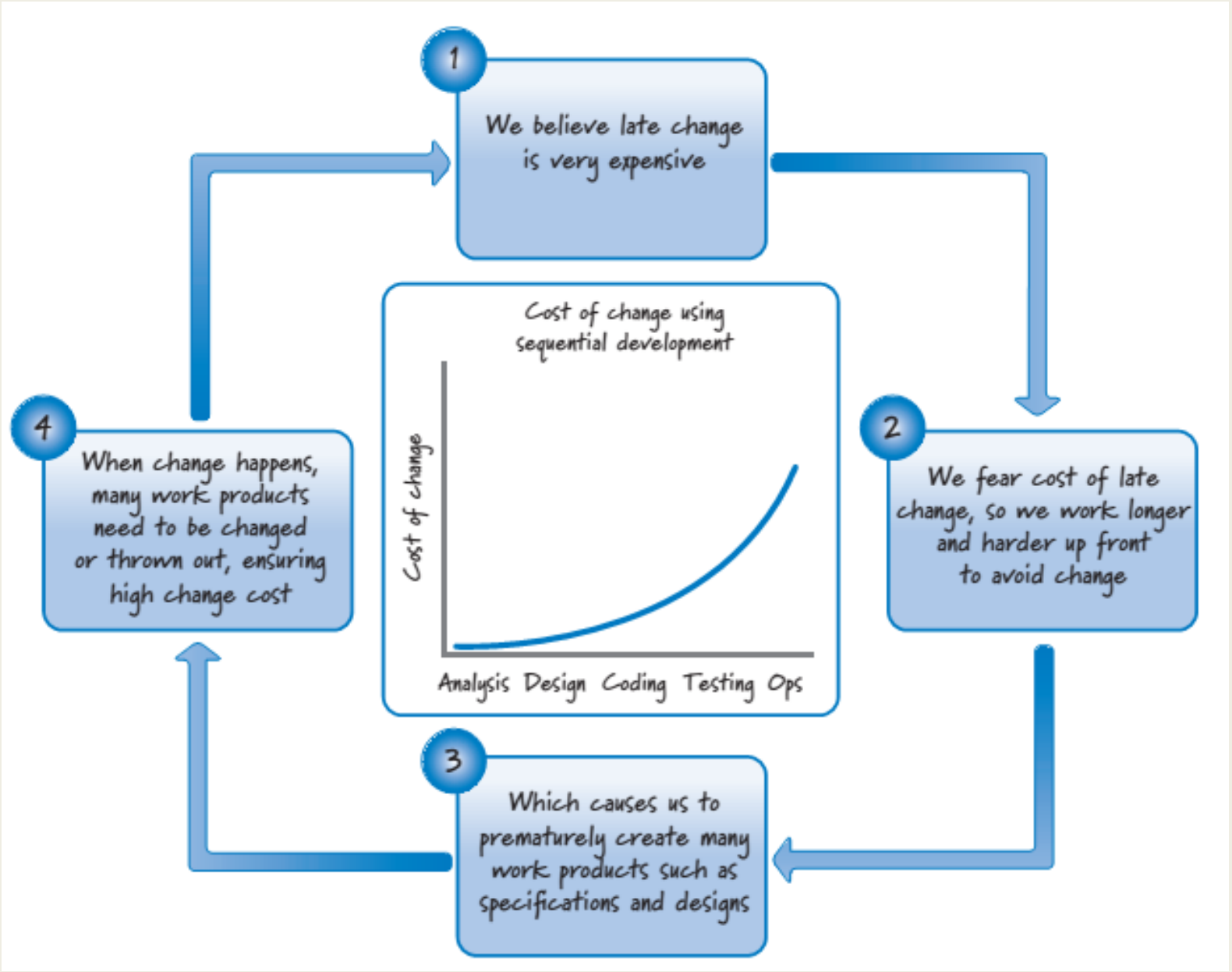
دقت پیش بینی رو بالا می بریم --> سعی میکنیم تا جایی که میشه تمام تلاشمون رو بکنیم که اون ریکوارمنت ها درست تنظیم بشه --> از اول سعی میکنیم تمام تلاشمون رو بکنیم که هرانچه ابهام وجود داره و هر انچه که نمیدانیم رو پیش بینی بکنیم ینی فرض بذاریم و برمبنای اون فرضیات ببندیم ریکوارمنت هارو

Embrace Change in an Economically Sensible Way(II)

- Unfortunately, being excessively predictive in early-activity phases often has the opposite effect.
- First, the desire to eliminate expensive change forces us to overinvest in each phase—doing more work than is necessary and practical.
- Second, we're forced to make decisions based on important assumptions early in the process, before we have validated these assumptions with feedback from our stakeholders based on our working assets. As a result, we produce a large inventory of work products based on these assumptions. Later, this inventory will likely have to be corrected or discarded as we validate (or invalidate) our assumptions, or change happens.

-
متاسفانه رویکرد وسواس گرایانه توی پیش بینی ریکوارمنت ها نتیجه عکس میده چون ما میخوایم یک تغییراتی رو بدیم

یکسری فرضیاتی اول پروژه قبول میکنیم که این فرضیات مهم است و هرچه زودتر این فرضیات اعتبارسنجی بشن ینی صحت و عدم صحتشون مشخص بشه مثلا ما فرضمون اینه که کارفرما یک سلیقه خاصی از مشتریان رو می خواد ارائه بده باید اینو مطمئن بشیم که این فرض درست است یا غلط پس اگر از فرضیات مهم زود بازخورد نگیریم یا درست و غلط بودنش رو زود متوجه نشیم این باعث میشه که ما تصمیمات بعدی رو برمبنای یک سری فرضیاتی بذاریم که اصلا معلوم نیست درست است یا غلط
پس رویکرد مقابله با تغییر با سخت کار کردن یا همون تلاش بیش از حد اول کار جواب نمیده


1
We believe late change is very expensive
2
We fear cost of late change, so we work longer and harder up front to avoid change
3
Which causes us to prematurely create many work products such as specifications and designs
4
When change happens, many work products need to be changed or thrown out, ensuring high change cost
Cost of change using sequential development
Cost of change
Analysis Design Coding Testing Ops

باورمون اینه که تغییرات توی دیرهنگام اگر اتفاق بیوفته خیلی هزینه بر است پس میخوایم هزینه این تغییرات رو کم میکنیم پس تمام تلاشمون رو میکنیم و همه بررسی ها رو انجام میدیم که روی اون تصمیم مطمئن باشیم و تاجایی که میشه احتمال تغییر رو کم کنیم
3- وقتی که کار خیلی زیاد انجام بشه متعاقبش یه عالمه خروجی داریم بعد اگر همین خروجی ها تغییراتی روش اتفاق بیوفته هزینه خیلی زیادی داره --> پس باید بذاریمش کنار و مجددا بهش بپردازیم (گام 4 میشه ینی جلوتر می فهمیم اینارو نمی خواستیم و باید بذاریمش کنار)
هرچی گام های بیشتری در راستای انجام پروژه برداریم چون artifactهای بیشتری تولید میشه پس هزینه تغییرات بالا می بره --> این سخت کار کردن منجر به پیشرفت روی اون نمودار میشه --> پس به افزایش هزینه تغییرات کمک میکنه --> پس کار کردن زیاد و پیش بینی کردن اطلاعاتی که در اختیار ما نیست نمی تواند از یه حدی به بعد هزینه رو کم بکنه --> چون همون کار کردن ینی انرژی گذاشتن و artifact های بیشتر --> تغییرات هزینه داره پس توی یک لوپ منفی گیر میکنیم

یک مسئله دیگه ای که اینجا مهمه --> اول کار یکسری فرضیاتی داریم که این فرضیاته مهم است و هرچی زوردتر این فرضیات اعتبارسنجی بشن ینی صحت یا عدم صحتشو مشخص بشه ما زودتر می فهمیم تصمیم درست بوده یا نه ولی اگر دیر مشخص بشه این اعتبارسنجی باعث میشه تصمیمات بعدی برمبنای یکسری فرضیاتی باشه که معلوم نیست اصلا این درست است یا نه
نکته: رویکرد مقابله با تغییر با سخت کار کردن اول کار جواب نمیده --> اینجا داره زیادی کار کردن اول کار رو زیر سوال می بره

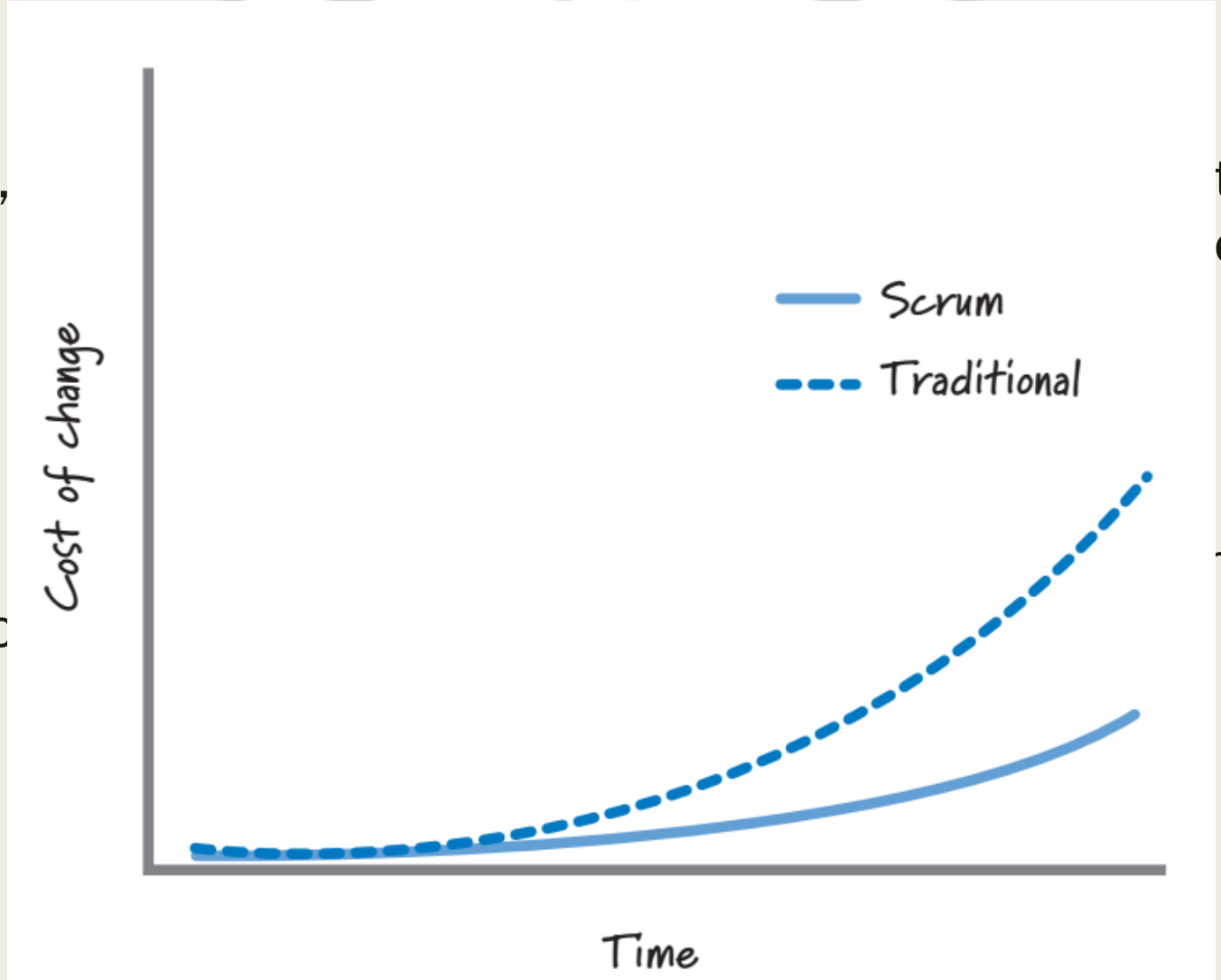
Embrace Change in an Economically Sensible Way(IV)

- In Scrum, we assume that change is the norm.
- We believe that we can’t predict away the inherent uncertainty that exists during product development by working longer and harder up front.
- Thus, we must be prepared to embrace change. And when that change occurs, we want the economics to be more appealing than with traditional development, even when the change happens later in the product development effort.

-
توی اسکرام اول فرض میکنیم تغییر یک نرم یا طبیعت است ینی طبیعی اتفاق میافته
خیلی مهم است که ما به یک پدیده دید مثبت داشته باشیم یا دید منفی ینی یک پدیده رو بپذیریم یا نپذیریم
اول دید نسبت به تغییر دید منفی نباشه چون ممکنه هیچ فردی مقصر نباشه و این اتفاق بیوفته و طبیعت اون کاری است که داریم انجام میدیم
از طرف دیگه نمی تونیم پیش بینی کنیم از اول کار همه چیز رو
پس باید چی کار کنیم؟ باید زیر ساختمون رو برای دریافت تغییرات اماده کنیم
پس هدف اینه که cost of change رو تا حد ممکن ثابت نگه داریم

Embrace Change in an Economically Sensible Way(V)

- Our goal, t for as long as possi e even late change.
- We can of work in process nange when using Sc sequential projects.



-
زیر ساخت رو جوری می چینن که هزینه تغییر حتی زمانی که دیره ثابت باشه البته از یه جایی به بعد نمایی میشه ینی از یه جایی به بعد که سیستم کاملا مستقر شد دیگه نمی شه اون نقطه عطف حذف کنیم ولی تا قبل از اون نقطه عطف ثابت نگهش میداره

هدف ما اینه که هزینه تغییرات برمبنای زمان تغییر نکنه

سوال امتحانی هستن:
اسکرام چگونه تغییر رو مدیریت میکند؟
1- زیرساختش جوری است که هزینه تغییرات ما تقریبا نسبت به زمان ثابت باشه تا جایی که میشه
2- سایزش خطی باشه --> سایز تغییر نسبت به سایز effort یا فعالیتی که برای اعمال تغییرات انجام میده نسبتا متناسب باشه ینی با یک تغییر کوچیک یک حجم زیادی از تغییر رو اعمال نکنه
یا
1* چارچوب رو جوری تنظیم میکنه که سایز اعمال تغییر با سایز فعالیت هایی که انجام میدیم برای اینکه تغییر اعمال بشه تقریبا متناسب است
2* تا حد ممکن چارچوب جوری است اون هزینه تغییره وابسته به زمان نباشه ینی نمودار رو صاف نگه میداره

Embrace Change in an Economically Sensible Way(VI)

- Regardless of which product development approach we use, we want the following relationship to be true.
  - *A small change in requirements should yield a proportionally small change in implementation and therefore in cost (obviously we would expect a larger change to cost more).*
- Another desirable property that we want it to be true regardless of <u>when the change request is made</u>.

-
تغییرات نسبت به سایز effort که باید انجام بدیم نسبت با سایز درخواسته خطی باشه --> یه وقت هست میگیم که یه تغییر کوچیک یه عالمه کار توی سیستم باید بکنه ینی سایز effort با سایز تلاشی که میکنیم با هم دیگه ناهمخوانه ولی یکی از اهداف اسکرام اینه که سایز تغییرات نسبت به سایز effort خطی باشه

وقتی که داریم روی یک ریکوارمنت کار میکنیم و همونو توی sprint انجام میدیم و از همون بازخورد میگیریم اگر تغییر هم بیاد روی همون تغییر رو اعمال میکنیم

Embrace Change in an Economically Sensible Way(VII)

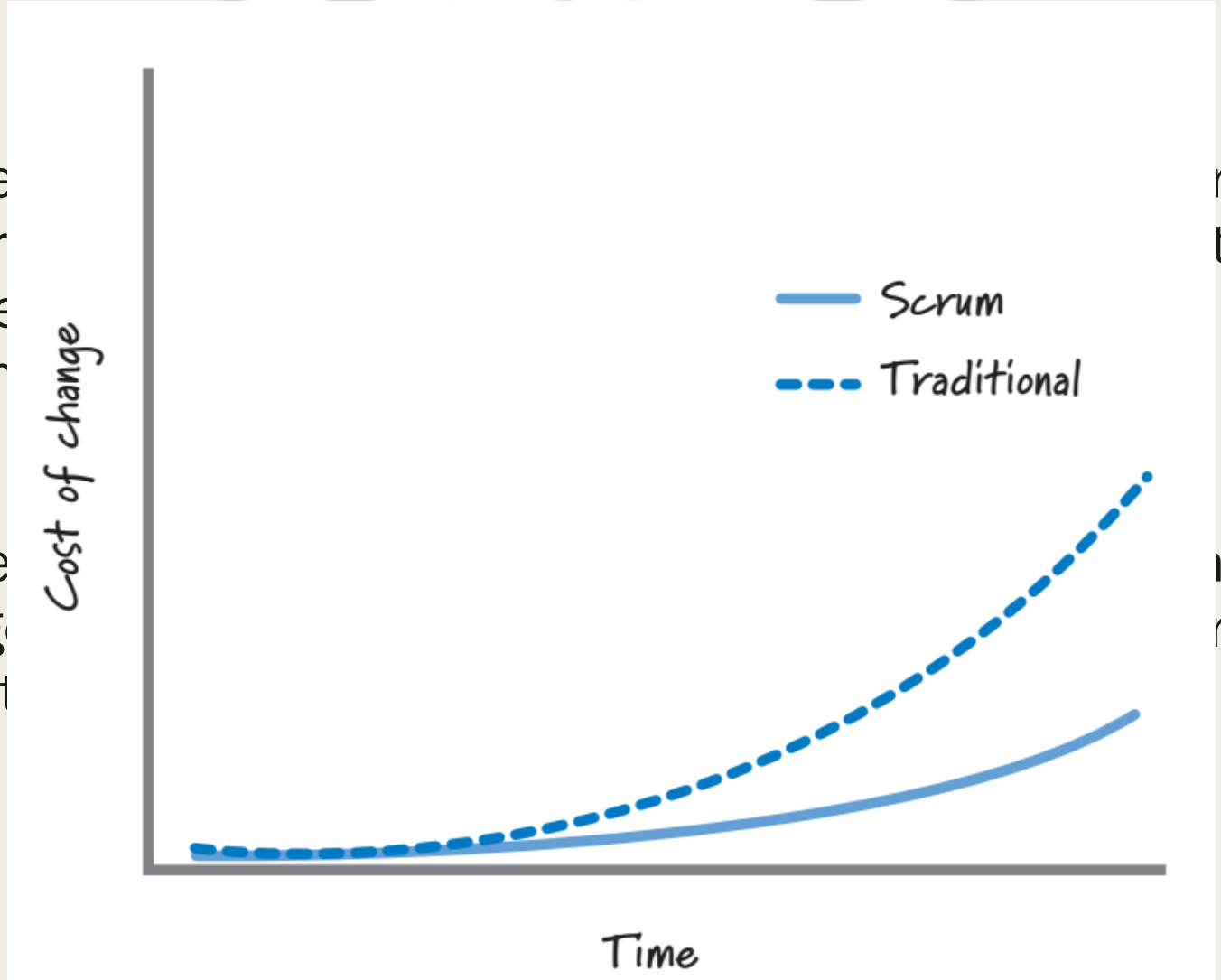
- With Scrum, we produce many work products (such as detailed requirements, designs, and test cases) in a just-in-time fashion, avoiding the creation of potentially unnecessary artifacts.

- As a result, when a change is made, there are typically far fewer artifacts or constraining decisions based on assumptions that might be discarded or reworked, thus keeping the cost more proportional to the size of the requested change.

-
تغییر به عنوان یک رویکرد طبیعی است توی جریان تولید یک محصول نرم افزاری
توی agile میگه چون این مسئله طبیعی است و ممکنه ما بعد از یک مدتی تغییر داشته باشیم نباید artifacts های غیرلازم رو جمع کنیم ینی در هر لحظه فقط کارهای اون مرحله رو انجام بدیم مثلا اول کار به همه ریکوارمنت ها می پردازیم و بعد وارد فاز بعدی می شیم ولی اینجا می گه این کارو نمی کنیم و ریکوارمنت هایی که مهم هستن رو اول بهش می پردازیم و اونایی که مبهم هستن یا اطلاعات کافی ازشون نداریم یا... روفعلا می ذاریم کنار
یکی از راه های مدیریت تغییر این است که به مهمترین چیزها بپردازیم و به ریکوارمنت هایی که می دونیم اولویت n ام هستن نپردازیم چون شاید بعدا اصلا اونا تغییر بکنن
اینده نزدیک و اینده دور این دوتا مهم هستن: اینده نزدیک ینی اون چیزهایی که قراره در اینده نزدیک بهش بپردازیم اونا فعلا باید روش کار بشه و اینده دورها رو فعلا نریم سراغش
توی اسکرام تغییر رو با این نحو مدیریت می کنه
وقتی که تغییر میاد تعداد کارهایی که کردیم و باید کنارش بذاریم کمتر است نسبت به رویکردهای سنتی چون یک بخش کوچیکی رو که خیلی مهم بوده رو بهش پرداختیم --> پس داریم هزینه اون تغییر رو تا حد ممکن کم می کنیم پس هزینه تغییر نسبت به سایز اون effort که داریم انجام میدیم تقریبا داره خطی میشه
قسمت سبز رنگ: ما یک سایز ریکویست داریم مثلا کتابی داریم که یکسری فیچر داره و یکسری فیچر دیگه هم می خوایم بهش اضافه کنیم و برمبنای همون فیچره هم میخوایم سرچ رو انجام بدیم
پس درخواست تغییر اینجا یک سایز داره - اون تغییراتی که روی کد میدیم که این تغییره اعمال بشه و این ریکوئست تغییرات اپلای بشه اینم یک سایزی داره --> حالا با اضافه کردن اون فیچر یا اینه که فقط همین ماژول توی محصول اومده بالا و هر تغییری بخوایم بدیم روی همون ماژول می دیم یا اینکه همه ماژولها رو طراحی کردیم و ریکوارمنت هاشو بردیم جلو و این تغییر علاوه بر اینکه روی این ماژول باید اعمال بشه روی ماژول های دیگه هم اعمال میشه پس سایز اون ریکوئست نسبت به سایز effort که برای اپلای این تغییره داریم خیلی کوچیک تر است ینی یک تغییر کوچیک باعث تغییرات خیلی زیاد توی سیستم میشه چون توسعه پیدا میکنه

Embrace Change in an Economically Sensible Way(VIII)

- Using se rtifacts and push for t the cost of a change causes the inflection
- When de hen the cost of chang request, but this point



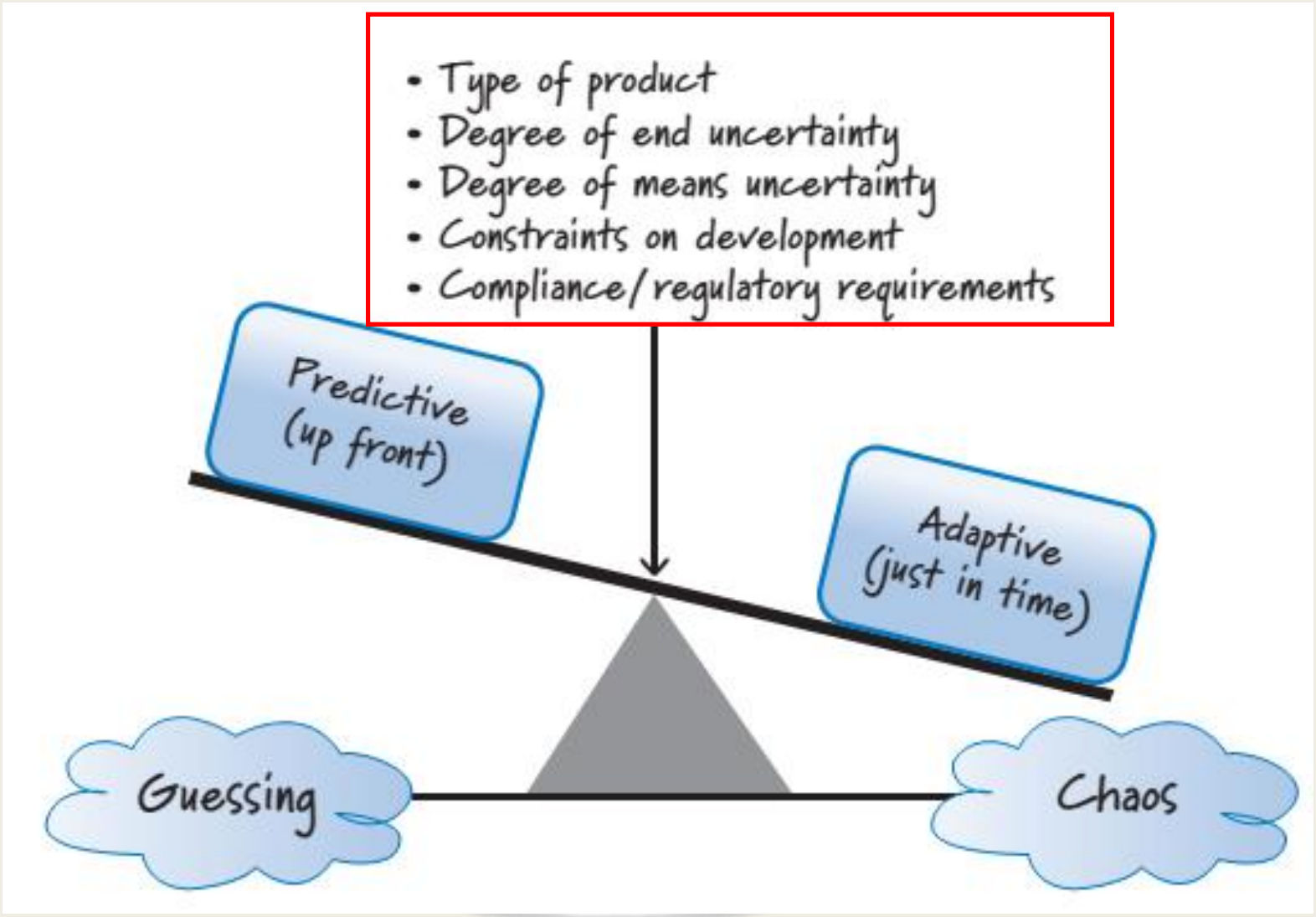
-
ادامه صفحه قبلی:
این تکنیک اسکرام که میگه در لحظه به اینده نزدیک نگاه کن و بقیه رو کاری نداشته باش این باعث میشه تا حد ممکن تغییره روی همون artifacts های لازم اعمال بشه

توی توسعه متوالی چون داریم روی حجم زیادی از ریکوارمنت ها کار میکنیم و همون اول کار داریم به همه ریکوارمنت ها می پردازیم --> میزان تصمیم هایی که هنوز پشت اون تصمیم اطلاعات کافی نداریم زیاده پس وقتی که یک تغییر میاد سایز این تغییره خیلی زیاده و از نقطه عطف که از اونجا به بعد به صورت نمایی تغییره زیاد میشه خیلی زود اتفاق می افته ولی توی اسکرام اینطوری نیست و نقطه عطف خیلی دیرتر اتفاق می افته

Balance Predictive Up-Front Work with Adaptive Just-in-Time Work(I)

- A fundamental belief of plan-driven development is that detailed up-front requirements and planning are critical and should be completed before moving on to later stages.
- In Scrum, we believe that up-front work should be helpful without being excessive. With Scrum, we acknowledge that it is not possible to get requirements and plans precisely right up front.
- Scrum is about finding balance—balance between predictive up-front work and adaptive just-in-time work.

-
ما باید یک توازنی بین Adaption و Prediction داشته باشیم:
دو سر طیف رو در نظر بگیر --> یک سرش اون سری است که سنتی ها انجامش می دادن که همه چیز رو طبق برنامه بریم جلو --> ما فقط راجع به یکسری چیزا اطلاعات داریم و بقیه رو نداریم --> حالا اونایی که اطلاعات داریم که همونا است و اونایی نیستن رو پیش بینی می کنیم که در نهایت تمام ریکوارمنت ها رو داشته باشیم که این خیلی اسیب زننده است
توی اسکرام میگه که خوبه که اون چیزی که داریم رو روش برنامه بچینیم ولی نباید افراطی باشه این موضوع --> ینی روی اطلاعاتی که داریم تصمیم می گیریم ولی اونایی که نداریم رو فعلا در انتظار نگه می داریم ینی یک توازنی باید اول کار داشته باشیم و خیلی خودمون رو نکشیم روی این موضوع و اینو بپذیریم که توی اسکرام قرار نیست همه ریکوارمنت ها رو از اول داشته باشیم --> پس توی اسکرام می گه نه پیش بینی صد در صد و نه adaptation صد در صد --> ینی نه اینطوری باشه که اول کار همه چیز رو بدونیم یا حدس بزنیم و نه از اونور هم بگیم که فعلا چیزی رو تنظیم نکنیم و بگیم بریم جلو --> پس توازن باید حفظ بشه
اگر همه چیز از اول بخوایم پیش بینی بکنیم اونم به صورت افراطی --> پس برمبنای یکسری اطلاعات نداشته تصمیم میگیریم که احتمال تغییرش خیلی زیاده است و غیر از این تصمیمات بعدیمون هم ممکنه اشتباه باشه
اگر همه چیز هم بر مبنای رویکرد adaptation بره جلو اینم خیلی خوب نیست ینی اخرش ما باید یک مسیر داشته باشه پروژمون --> چون اگر این کارو نکنیم برمبنای هر تغییری و هر درخواستی و هر اتفاقی و.. ممکنه مسیر عوض بشه ینی کاملا بی نظم می ریم جلو
پس در عین حال که یک حداقلی از پیش بینی رو داریم باید یک حداکثری هم از adaptation داشته باشیم


• Type of product
• Degree of end uncertainty
• Degree of means uncertainty
• Constraints on development
• Compliance/regulatory requirements
Predictive (up front)
Adaptive (just in time)
Guessing
Chaos

نقطه توازن کجاست؟
این نقطه بستگی به فاکتورهای متفاوتی داره - بستگی به خود محصول داره
مثلا ممکنه محصول ما محصولی است که قبلا زده شده ینی تقریبا همه چیزاشو میدونیم و پیش بینی توی این حالت بالاتر است و خیلی جایی برای adaptation نداریم و فقط میخوایم این محصول رو تغییر بدیم
ولی یک موقع است ک محصول خیلی نامشخص است حالا یا کارفرما نمی دونه یا خودمون یک ایده جدید داریم و این ایده خیلی خام است و.. در این حالت باید پیش بینی کم باشه و adaptation بیشتر باشه
1- پس به محصول برمی گرده
2- به درجه end uncertainty , means uncertainty هم برمیگرده: ینی مخاطبمون چه مخاطبی است --> مخاطب کاملا شفافی است و سختی نداره با اون مخاطب و اونا خودشون میدونن چی میخوان و چه اتفاقاتی می افته
3- به تکنولوژی وسط مسیر هم برمیگرده
4- به تیم و افراد تیم هم برمیگرده --> یک وقت ادم های تیم ادم هایی هستن که سابقه تغییر خیلی کمی دارند یا برناممون مشخص است مثلا میگیم تا یک سال اینده پلن ادامه تحصیل نداریم

پس این نقطه توازن رو باید پیدا بکنیم --> ینی تا یه جایی پیش بینی رو متوقف کنیم و محوریت کار مشخص باشه و روی اطلاعاتی که داریم تصمیم بگیریم و تصمیماتی که برمبنای اطلاعاتی هست که نداریم فعلا معلق بذاریم ینی اپشن ها رو فعلا برای خودمون باز بذاریم تا برسیم به یک درصدی از اطمینان و بعد اون کار رو انجام بدیم

Balance Predictive Up-Front Work with Adaptive Just-in-Time Work(III)

- When developing a product, the balance point should be set in an economically sensible way to maximize the amount of ongoing adaptation based on fast feedback and minimize the amount of up-front prediction, while still meeting compliance, regulatory, and/or corporate objectives.

- Exactly how that balance is achieved is driven in part by the type of product being built, the degree of uncertainty that exists in both what we want to build (end uncertainty) and how we want to build it (means uncertainty), and the constraints placed on the development.

-
حالا بالانس کجا اتفاق می افته؟
پیش بینی باید مینیمم بشه و adaptation ماکزیمم --> چرا؟ چون توی اون Danger zone اون منطقه ای که اول کار داریم با حداقل ترین اطلاعات ینی کم کیفیت ترین اطلاعات مواجه هستیم پس تا جایی که می تونیم پیش بینی رو میاریم پایین ولی صفر نمی کنیم چون یه عالمه ریکوارمنتی وجود دارن که low level هستن از نظر کیفیت پس پیش بینی بخاطر این Danger zone مینیمم است و از یه طرفی یه عالمه عدم قطعیتی که در اینده باهاش مواجه هستیم پس adpatation باید ماکزیمم باشه
یه جا باید این نقطه تعادل رو پیدا بکنیم --> برمبنای اطلاعات مهم و کلیدی و محوری پیش بینی میکنیم و یه عالمه تصمیمات رو باز می ذاریم تا در اینده اطلاعات بیشتر و مشخص تری که اومد بعد می بندیمش

Balance Predictive Up-Front Work with Adaptive Just-in-Time Work(IV)

- Being overly predictive would require us to make many assumptions in the presence of great uncertainty. Being overly adaptive could cause us to live in a state of constant change, making our work feel inefficient and chaotic.

- To rapidly develop innovative products we need to operate in a space where adaptability is counterbalanced by just enough prediction to keep us from sliding into chaos.

- The Scrum framework operates well at this balance point of order and chaos.

-
اگر خیلی پیش بینی طور بریم جلو --> داریم برمبنای اطلاعات نامعلومی تصمیم میگیریم که ممکنه در اینده تغییر کنه
و اگر adaptation بریم جلو --> به یه عالمه بی نظمی برخورد میکنیم

Reference

1- K. S. Rubin, “Essential Scrum, A Practical guide to the most popular agile process,” 2013.