



Agile Principles

Dr. Elham Mahmoudzadeh
Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

2023

متدلوژی که اینجا داریم اسکرام است
اسکرام --> یک تکنیک مدیریتی به ما میده
اگر اسکرام با همه جزئی کاری هاش رعایت نشه به شکست منجر میشه
اسکرام برای همه پروژه ها لزوما جواب نمیده
اگر موثر عمل نکنیم خودمون ضرر میکنیم

What is Software Engineering?

IEEE Computer Society Definition:

- “Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.”

تعریف مهندسی نرم افزار:

کل پکیج و فعالیت هایی مربوط به دولوپ نرم افزار و تخصیص افراد به پروژه و دادن بخش ها است می گن مهندسی نرم افزار

Software engineering(2)

- Doing the right thing
 - *Software that users want and need*
 - *Software that benefits society*
- Doing the thing right
 - *Following a good software process*
 - *Developing your programming skills*

از یه دیدگاه دیگه:

کار درست انجام دادن:

ینی همون پروژه ای که کارفرما می خواد ما انجام بدیم و محصوله محصول درستی باشه و منطبق با نیاز کارفرما باشه و موثر باشه
درست انجام دادن کار:

فرایند نرم افزاری درست بره جلو

یک بحثی که توی اسکرام است فقط به پروژه نگاه نمی کنه و فقط به نحوه انجام کار، بلکه ادم هایی که توی کار هستن هم خودشون هم رشد می کنن

Who is a software engineer?

- *Software engineers are the creative minds behind computers or programs.*
- A software engineer is the one who follows
 - *A systematic process that leads to understanding the requirements,*
 - *Working with teams and various professionals*
 - *Design and create the application software or components or modules*
 - *Fulfill the specific needs of the users successfully;*

مهندسی نرم افزار به عبارتی مغز رشد و تفکر پشت اون محصول است
و مهندس نرم افزار کسی است که اون پروسیسی که برای فهم ریکوارمنت هاست به نحو درستی
می ره جلو

مثلا ساختمانی که یک ادم تجربی میسازه و ساختمانی که یک مهندس عمران می سازه می تونیم
کاملا تفاوت رو توی این دو ساختمان ببینیم در مورد نرم افزار هم همینطوره --> نوع نگاهی که
یک مهندس نرم افزار داره با کسی که نداره خیلی فرق داره
نرم افزارکارها بدون کار تیمی نمی تونن کار موفق انجام بدن در حال حاضر

Software Development Methodology (SDM)

- A framework for applying software engineering practices with the specific aim of providing the necessary means for developing software-intensive systems.
- Have two parts.
 1. *A set of modeling conventions comprising a Modeling Language (syntax and semantics)*
 2. *A Process, which*
 - provides guidance as to the order of the activities,
 - specifies what artifacts should be developed using the Modeling Language,
 - directs the tasks of individual developers and the team as a whole,
 - offers criteria for monitoring and measuring a project's products and activities.

SDM یک فریم ورک است که بتوانیم ما فعالیت های نرم افزاری رو با یک هدف خاصی انجام بدیم که در نهایت اون هدف ما میشه ارائه سرویس به اون جامعه نرم افزاری متدلوژی ساخته شده از دوتا قسمت:

1. یکی بحث **Modeling Language** که سینتکسی است که ارائه میشه
2. فرایند:

فرایند ینی ترتیب انجام فعالیت هایی که برای رسیدن به یک هدف خاصی داره انجام میشه و هر تسکی یکسری نتیجه داره و یکسری معیارهایی برای نظارت که ایا طبق برنامه جلو رفتیم یا نه هم وجود داره

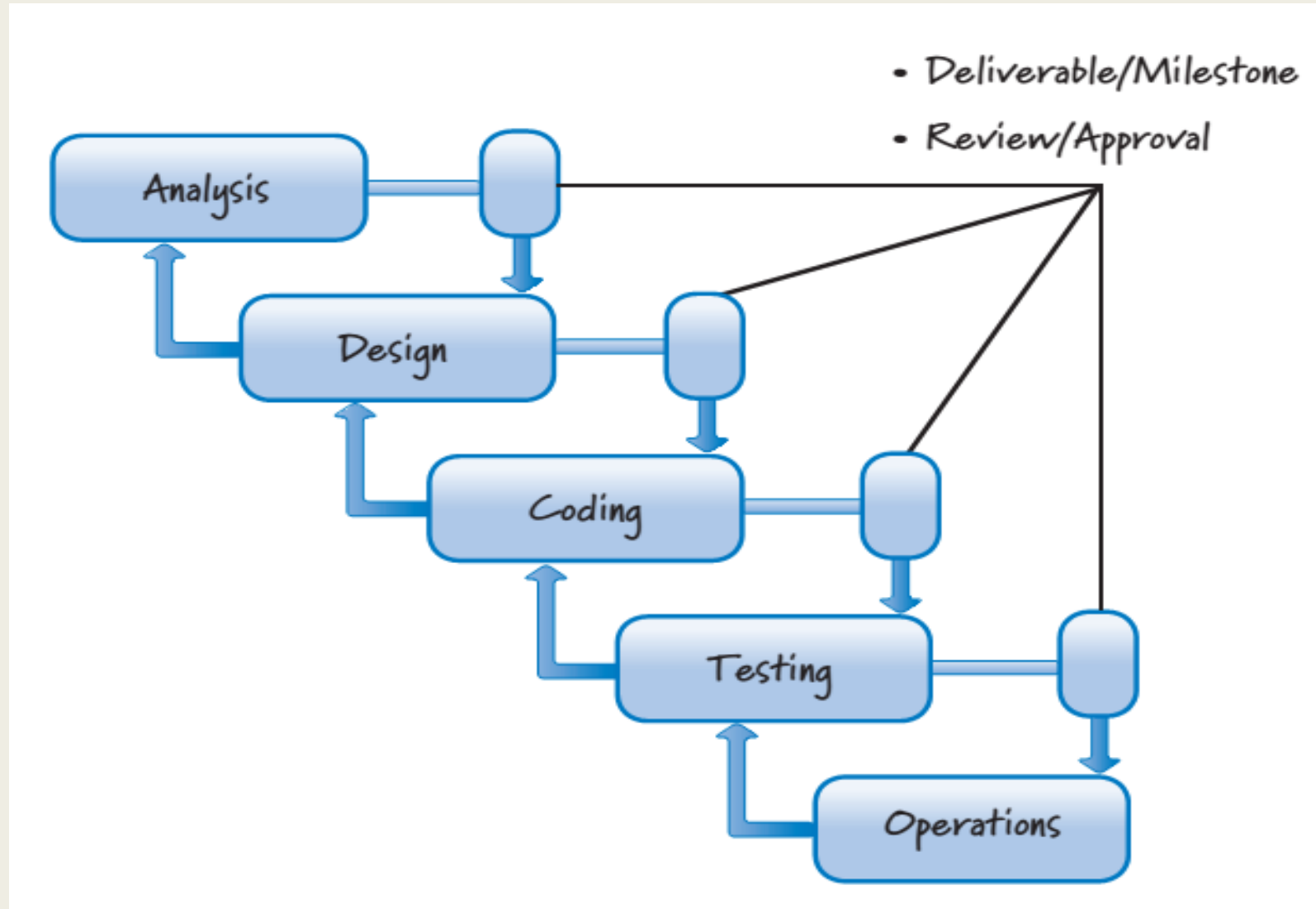
Agile vs. Traditional development process

- The goal of comparing agile principles with traditional development principles is not to make the case that **plan-driven, sequential development** is bad and that Scrum is good.
- Both are tools in the professional developer's toolkit; there is no such thing as a bad tool, rather just inappropriate times to use that tool.

Scrum and traditional, plan-driven, sequential development are appropriate to use on different classes of problems.

مقایسه رویکرد سنتی با رویکرد اسکرام:
مهمترین مسئله این است که هر دوی این ها ابزار فرایند تولید نرم افزارند
و هر دو مناسب اند ولی هر کدامشون برای پروژه های خاص خودشون
کاری که ما انجام میدیم اینجا مقایسه اسکرام با اون رویکرد سنتی است

Plan-driven process (Waterfall)



Waterfall به عنوان یک سمپلی از اون رویکردهای سنتی مثال می زنیم:

توی Waterfall ما چک پوینت ها اخر هر فاز است ینی تحلیل که تموم شد چک پوینت داریم که
یه تیمی میاد بررسی میکنه

پس ما یه عالمه چک پوینت داریم ولی اخر اون فاز است

توی Plan-driven ها پلن یا برنامه خیلی مهم است

Plan-driven process (I)

- Plan for and anticipate up front all of the features a user might want in the end product, and to determine how best to build those features.
- The idea here is that the better the planning, the better the understanding, and therefore the better the execution.
- Also called sequential processes because practitioners perform, in sequence, a complete requirements analysis followed by a complete design followed in turn by coding/building and then testing.

Plan-driven: همون اول کار میاین بر مبنای ریکوارمنت هایی که وجود داره برنامه شون رو می چینن و مشخص می کنن که چجوری می تونن به اون پارامترها برسن --> کاملاً برنامه محورند

ایده این مسئله اینه که هرچی که برنامه بهتر باشه موفق تریم ینی هر چی برنامه خوب چیده بشه و هرچی این برنامه بدون ابهام باشه و شسته تر باشه خوبه و بهتر می ره جلو

همینطور متوالی هم هست ینی مرحله به مرحله می ره جلو ینی اول ریکوارمنت ها رو پیدا و بررسی می کنیم و بعد خطاهاشو در میاریم و بعد نهایی که شد وارد فاز تحلیل میشیم و بعدش وارد فاز طراحی میشیم و بعدش وارد پیاده سازی و در اخر وارد تست می شیم ینی همه چیز مرحله به مرحله می ره جلو ینی برنامه محوریت داره اینجا

Plan-driven process (II)

- Works well if you are applying it to problems that are well defined, predictable, and unlikely to undergo any significant change.
- The problem is that most product development efforts are anything but predictable, especially at the beginning.
- So, while a plan-driven process gives the impression of an orderly, accountable, and measurable approach, that impression can lead to a false sense of security.

After all, developing a product rarely goes as planned.

اگر برنامه کار بکنه ینی برنامه از اول خیلی خوب چیده شده

این Plan-driven ها برای پروژه هایی بدرد می خوره که همه چیزش از اول مشخص شده باشه ینی کاملاً معلوم باشه یا قابل پیش بینی باشه یا تغییر عمده ای توی اون پروژه اتفاق نیوفته ینی یک پروژه پیشنهاد میشه که این پروژه همه چیزش معلومه و کارفرما می دونه چی میخواد و مشکل خاصی نداره و ریسکش هم پایینه پس منطقی که ما طبق برنامه بریم جلو ولی مسئله اینه که اکثر پروژه های ما این خصوصیت رو ندارن ینی کارفرما نمیدونه چی میخواد و قابل پیش بینی نیستند و... پس Plan-driven به ما رویکردی میده که خیلی منطقی و منظم و حساب شده است ولی ما خودمون اینو میدونیم که همه چیز می ره جلو به جز برنامه ای که از اول چیدیم چون در واقعیت ممکنه یکسری اتفاق هایی اون وسط بیوفته که جای افراد توی تیم عوض بشه و.. و معمولاً طبق برنامه پیش نمی ره

پس ما بررسی میکنیم چه پروژه هایی طبق Plan-driven می تونیم بریم جلو و کدوما نه

Plan-driven process (III)

- Understand it, design it, code it, test it, and deploy it, all according to a well-defined, prescribed plan.
- There is a belief that it should work. If applying a plan-driven approach doesn't work, the prevailing attitude is that we must have done something wrong.
- Sure that if they just do it better, their results will improve. The problem, however, is not with the execution.
- It's that plan-driven approaches are based on a set of beliefs that do not match the uncertainty inherent in most product development efforts.

ریکوآرمنت ها و تحلیل و طراحی رو انجام میدیم و در نهایت بر مبنای اون فهمی که داشتیم پلنمون رو می چینیم

باور بر این است که اگر برنامه خوب پیش بره باید کار کنه

اگر اون Plan-driven ها درست کار نکرد و برنامه خوب بود قاعدتا ادم ها یک مشکلی دارند ینی تقصیر ادم هاست که خوب کار نکردن پس اگر کارشون رو خوب انجام میدادن نتیجه بهتر بود مسئله ما این است که یا پلن رو درست چیده نشده یا اگر درست چیده شده ادم ها درست کار انجام ندادن که این کار بره جلو پس باورهایی که وجود داره این باورها الان دیگه کار نمیکنه ینی ممکنه ادم ها خوب کار کرده باشن و پلن ها هم درست چیده شده باشه ولی مطابق با واقعیت نباشه و نتیجه موفقیت امیز نباشه

Traditional Pros.

- It is supremely logical.
- Think before you build.
- Write it all down.
- Follow a plan.
- Keep everything as organized as possible.

خوبی های این Plan-driven این است که:

همه چیز منطقی است ینی همه چیز شسته و رفته داریم اینجا

قبل از ساختن فکر کنیم

همه چیز رو بنویسیم

طبق برنامه بریم جلو

همه چیز منظم و سازماندهی بره جلو

What's Wrong With Traditional Software Development?

Humans are involved.

- Creativity is inhibited.
- Written documents have their limitations.
- Bad timing.
- No crystal balls.
- Too much work and no fun.
- Sub-optimized results.

نقاط منفی:

ادم رو به عنوان یک فردی که کننده کاره در نظر نمی گیره

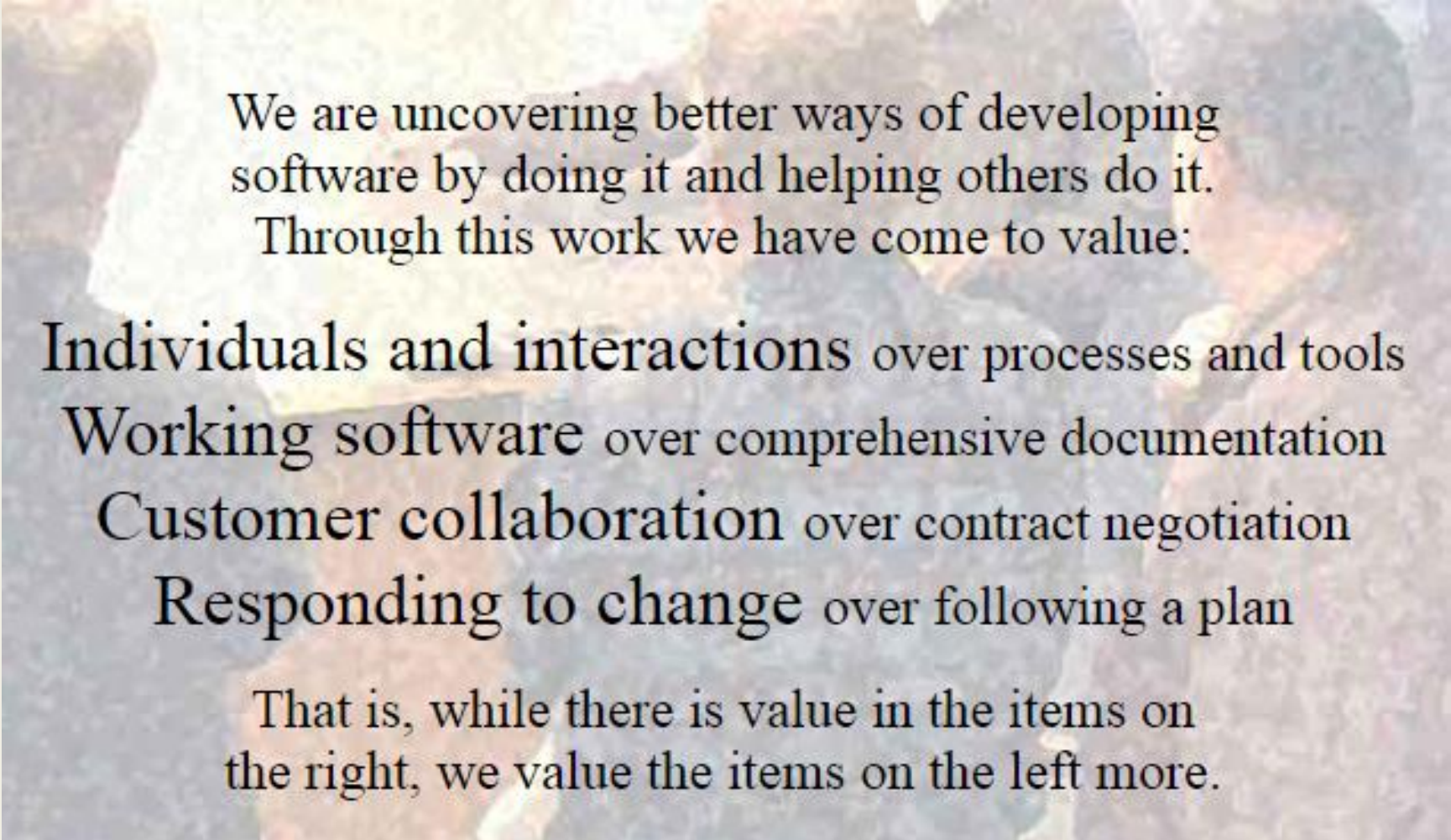
قاعدتا یک انسان خلاقیت داره ینی الان ممکنه چیزی به ذهن ما نرسه ولی بعدا برسه ینی اون چیزی که اول می گیم با اون چیزی که نتیجه می شه ممکنه خیلی متفاوت باشه --> ما نمی تونیم خلاقیت رو به عنوان یک موضوعی که توی ذهن ادم ها هست حذفش کنیم

مستندسازی خوبه ولی محدودیت های خودش هم داره ینی نمیتونم همه رو وابسته کنیم به اون مستندها بلاخره ممکنه فهم ادم ها از یک نوشته متفاوت باشه - ممکنه یک نوشته خلاصه باشه ینی ادبیات ادم ها با هم متفاوته --> برداشت از اون داکيومنت ممکنه برداشت متفاوتی باشه تایمش --> اگر ما یک کاری رو رو عقب بندازیم قاعدتا همه چیزها به تاخیر می افته و استفاده بهینه از زمان نمیشه

قاعدتا هیچکس نمیتونه آینده رو پیش بینی کنه --> آینده ای که یک رقیب جدید بیاد یا یک نیاز جدیدی بیاد یا ...

در نهایت اون لذت و آرامش که باید در طول انجام کار داشته باشه توی Plan-driven ها نمی بینیم و در نهایت نتایج یک نتایج نیمه بهینه میشه

Manifesto for Agile Software Development



We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

ادم ها و روابطشون مهمتر از اون فرایندها و ابزارهاست
نرم افزاری که کار میکنه بهتر و اولویت داره نسبت به داکيومنت ینی وقت رو بیشتر بسازیم روی
اون خروجی نرم افزار که کار بکنه تا داکيومنت نوشتن
ارتباط با کارفرما اهمیت داره نسبت به قراردادی که قبلا داشتیم ینی پاسخ به تغییرات مهمتر از
تبعیت از برنامه است ینی ما اینجا اصل Plan-driven رو داریم نقض میکنیم ینی میگیریم برنامه
مهمه و ادم باید برنامه داشته باشه ولی ادم ها مهمتر است

اسکرام چارچوبش رو طبق این اصول می چینه ینی به نحوی چارچوبه رو چیده که بشه به تغییرات
پاسخ داد و به نحوی این اعضای تیم رو چیده که ارتباط با کارفرما برقرار باشه و به نحوی چیده
شده که خروجی هر sprint یک working software باشه و به نحوی چیده شده که بتونیم اون
افراد رو در نظر بگیریم پس همه فعالیت های اسکرام از این اصول تبعیت میکنه

Principles Behind the Agile Manifesto(I)

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.

- اصول 12 گانه ای وجود داره:

1. مهمترین اولویت اینه که مشتری رو راضی نگه داریم چجوری؟ با سریعترین تحویل دادن اون پروژه بهش ینی هم سریع بهش تحویل بدیم و هم پیوسته باشه و در نهایت اون چیزی که بدرش می خوره بهش تحویل بدیم
2. ریکوارمنت ها اگه قراره تغییر بکنه ایرادی نداره حتی اگر اخرهای پیاده سازی باشه و ما زیرساختمون قوی باشه که تغییرات رو با حداقل هزینه بپذیریم --> فرایندهای agile تغییرات رو می پذیرن --> برای کارفرما این مسئله خیلی مهم است که قفلش نکنیم ینی اگر واقعا یک نیازمندی داد و بعدا پشیمون شد نگیم که نمیشه دیگه
3. working software رو مکرر تحویل بدیم ینی هفته به هفته یا ماهانه یا.. ینی توی تایم های کوچکتر تحویلش بدیم
4. افراد تجاری و توسعه دهندگان باید در قالب یک تیم کار بکنند ینی اون فردی که نماینده بازاره برای ما با افراد تیم با هم دیگه کار رو ببرن جلو و انجام بدن

Principles Behind the Agile Manifesto(II)

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

5. کار کردن با انگیزه و ریلکس است ینی پروژه رو طبق ادم هایی که انگیزه دارن می بریم جلو و زورشون نمی کنیم --> خود ادم ها کار می کنند ینی محیط و بستر و امکانت رو بهشون میدیم که اون ها کار بکنند ینی بستر انجام کار رو براشون فراهم میکنیم و ازشون انتظار داریم که کار بکنند و در نهایت بهشون اعتماد می کنیم که کارشون رو انجام بدن که اعتماد جز یکی از اصول agile است ینی ادم ها اونقدر مسئولیت پذیر هستن که اون کاری که می گن انجام می دن رو انجام میدن
6. روش موثر و کارآمد برای انتقال اطلاعات روش حضوری است ینی بهترین روشی که ما با کمترین هزینه بتونیم اطلاعات رو دریافت بکنیم روش حضوری است ینی اگر بخوایم یک عالمه داکيومنت بنویسیم و به طرف منتقل بکنیم هم هزینه نوشتن زیاد میشه و هم هزینه فهم اون و هم ریسک سو برداشت خیلی بالا می ره ولی وقتی که توی جلسات حضوری باشه هم سریعتر است و هم موثرتر است و هم خطاش کمتر است --> روش انتقال چهره به چهره مهمترین و به عبارتی کانال ارتباطی موثر که بیشترین اطلاعات رو با کمترین خطا منتقل می کنه
7. مهمترین معیار ارزیابی **working software** است ینی باهاش کار بکنه کارفرما پس **working software** مهمترین معیار ارزیابی میزان پیشرفت پروژه است
8. سرعت انجام کار در کل روند پروژه ثابت نیست ینی همیشه اینطوری که اول کار پروژه یک وقت زیادی می داریم و بینشون با ارامش کار می ره جلو و یهو اخرش که می خوایم پروژه رو تحویل بدیم زمان زیادی می داریم ولی در فرایند **agile** تقریباً سرعت ثابت است چون توی تایم های کوچک داریم پروژه رو تحویل میدیم پس از همون اول تا اخر سرعت تقریباً ثابت است پس در اخر باید یک سرعت ثابتی رو از اول تا اخر پروژه حفظ بکنیم

Principles Behind the Agile Manifesto(III)

- 9. Continuous attention to technical excellence and good design enhances agility.
- 10. Simplicity—the art of maximizing the amount of work not done—is essential.
- 11. The best architectures, requirements, and designs emerge from self-organizing teams.
- 12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

9. اخرش پیوسته و مکرر داریم کار رو می بریم جلو --> یکی از ایراداتی که به agile وارد شده اینه که agile چون که میگه چابک برو جلو ما داریم از کیفیت کار می زنیم ولی چون داریم مداوم بازخورد می گیریم و به کارفرما ارائه میدیم و گام به گام می ریم جلو این کار باعث میشه کیفیتش بالا بره--> چون مداوم است هم کیفیت کار رو تضمین میکنه و هم چابکی کار می ره جلو (گام های کوچیک باعث میشه کار به سرعت میره جلو)

10. تا جایی که میشه شاخ و برگش رو بکنیم تا به core سیستم برسیم --> ینی تا جایی که میشه ساده اش بکنیم ینی هنر اینه که شاخ و برگ رو بیشتر بکنیم ینی کارهایی که نباید انجام بدیم رو بیشتر بکنیم تا برسیم به core اصلی --> اگر این اتفاق نیوفته ممکنه اول کار سردرگم بشیم پس ساده کردن مسئله ضروری است

11. بهترین معماری ها و بهترین نیازمندهای ... از دل تیم هایی میاد که خودسازمانی میشن ینی مدیر تیم هم خودش کننده کار است ینی یک فردی بالای سر تیم نیست که بگه این کارو بکنین یا نکنین ینی تیم هایی هستنند که خودشون، خودشون رو مدیریت می کنن --> سینک کردن افراد تیم باهم دیگه توسط تیم انجام میشه (رویکرد اسکرام هم طبق همین منطق است ینی تا جایی که میشه مسئولیت کارت رو خودت بپذیر و برای اون کار هم به صورت خودمختار اون کار رو انجام بده)

12. توی بازه های منظم تیم منعکس میکنه فعالیت هاش رو --> یکی از کارهایی که اخر هر sprint تحویل میدیم همون working software است- یک چیز دیگه ای هم که اخر کار sprint باید ارزیابی بشه روند انجام کار است --> ینی اخر کار میایم خودمون رو قضاوت میکنیم و انعکاس فعالیت های خودمون رو دریافت میکنیم --> پس اخر اون sprint هم ارزیابی محصول رو داریم هم ارزیابی خودمون رو داریم که این توی پیشرفت ادم ها خیلی موثر است --> رفتارمون رو بر مبنای اون نتایج قضاوت تنظیم می کنیم و این باعث میشه به طور مداوم رشد بکنیم

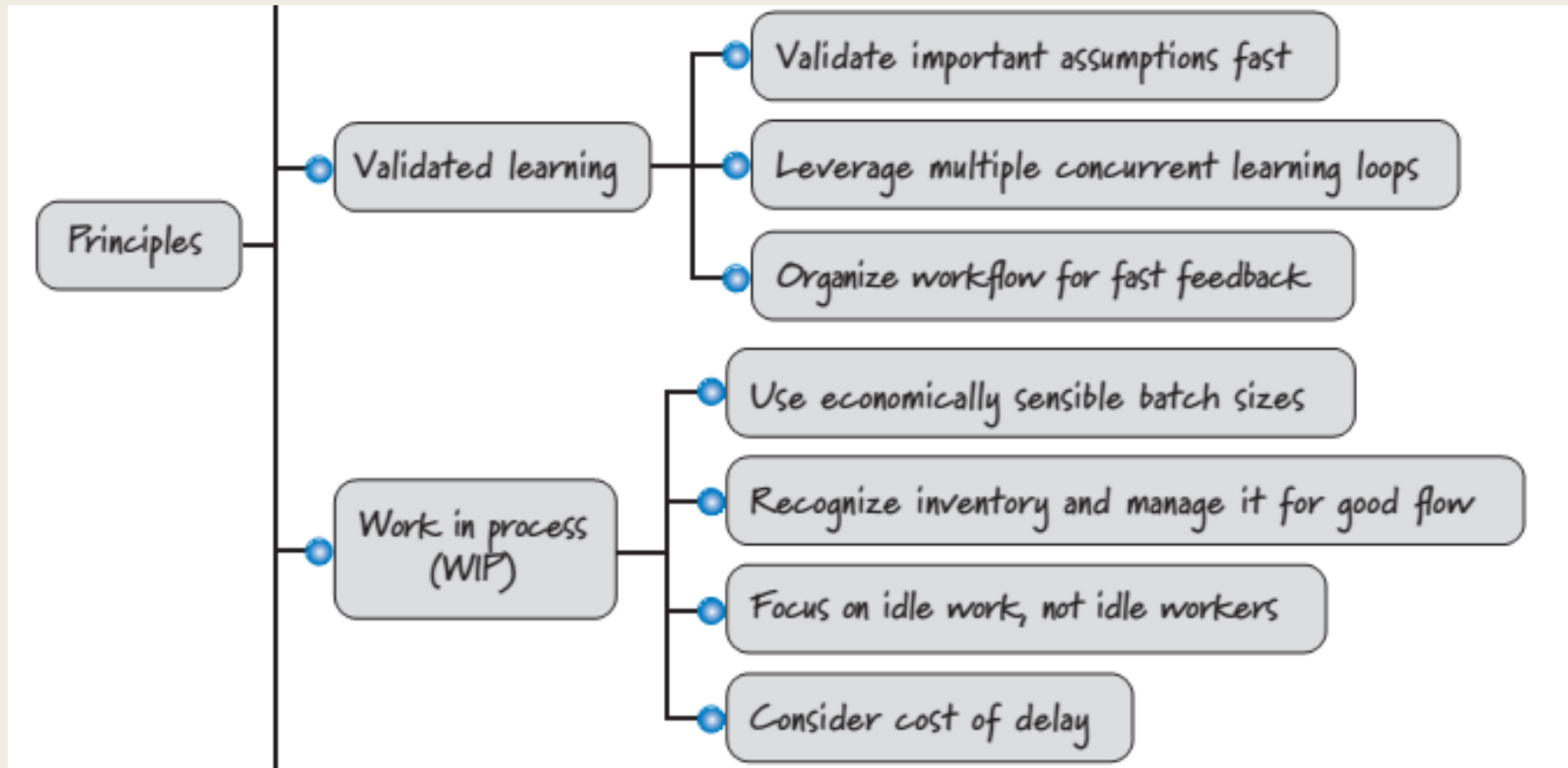
Categorization of principles (Up)



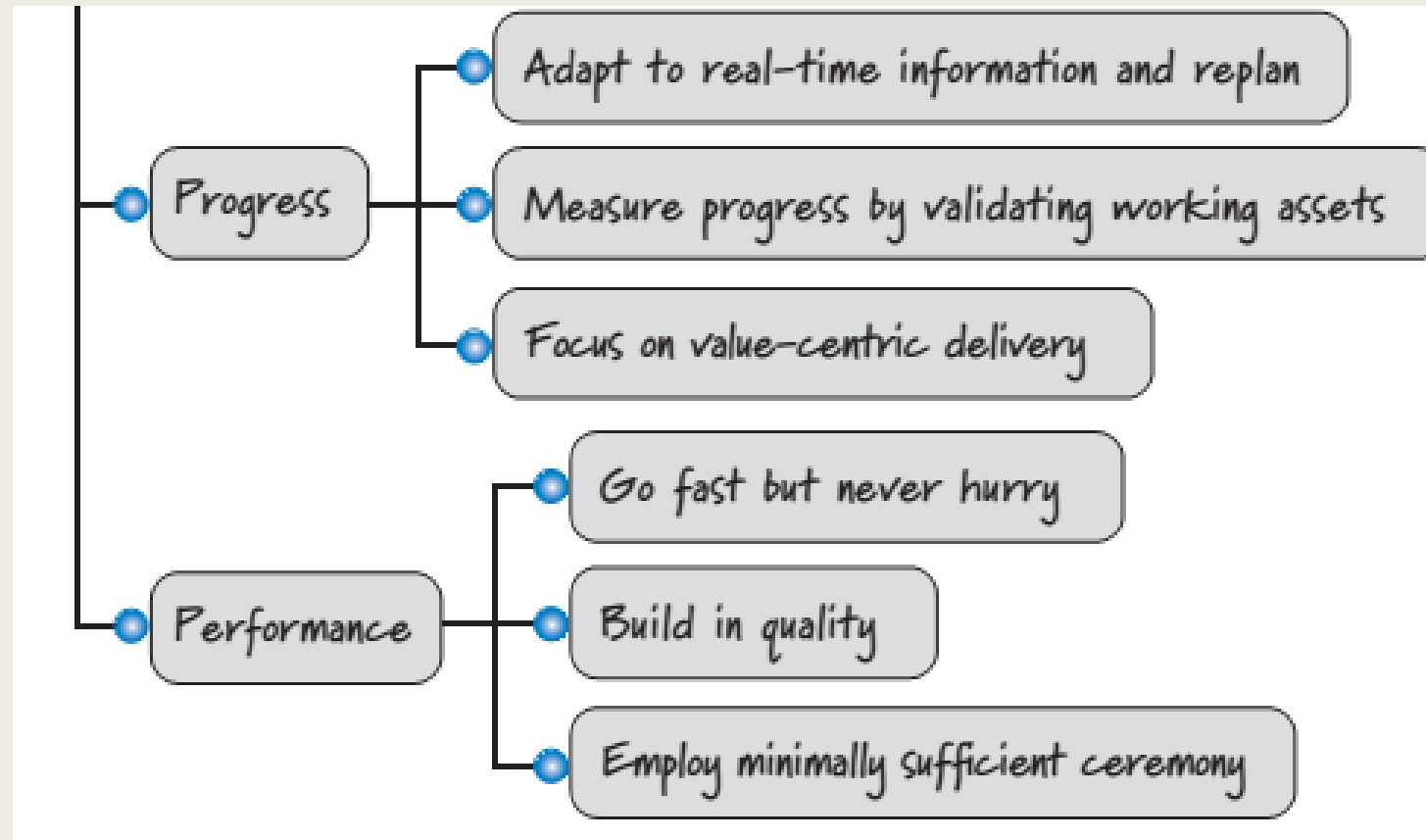
این اصول 12 گانه تقسیم می شن به 6 تا اصل و یکسری زیر اصل ها:

- 1- تغییر پذیری و عدم قطعیت
- 2- پیش بینی و تطبیق
- 3- چیزهایی دیگه ای که بتونیم به صورت پیوسته از پروژه یاد بگیریم
- 4- میزان کارهای ناتمامی که داریم مدیریت کنیم
- 5- میزان پیشرفت کار
- 6- پرفرمنس

Categorization of principles (Middle)



Categorization of principles (Bottom)



References

- 1- K. S. Rubin, “Essential Scrum, A Practical guide to the most popular agile process,” 2013.
- 2- J. Sutherland, “Scrum handbook,” 2010.