AGILIS

Az agilis szemléletről elfoglalt vezetőknek (is)

TRAINING360

Avagy hogyan segíti az agilis szemlélet a csapatmunkát és a szervezeteket

TARTALOM

Mit jelent az agilis projektvezetés?

Az agilis projektvezetés egy iteratív megközelítés a szoftverfejlesztési projektek hatékony menedzselésére, amely elsősorban a folyamatos szállításra és ügyfél visszajelzések beépítésére fókuszál.

Azokban a szoftverfejlesztő csapatokban, ahol már ezen módszertan mentén dolgoznak, számos szempontból mérhető javulás tapasztalható:

- Növekszik a fejlesztés tempója,
- Hatékonyabban működnek együtt a csapattagok,
- Gyorsabban reagálnak a piacon bekövetkező trend változásokra.

Ez a dokumentum egy olyan praktikusan használható eszköztárat nyújt át az Olvasó kezébe, ami segít elsajátítani az agilis gondolkodásmód lényegét, támogatja az agilitás irányába történő elindulást és betekintést nyújt a szemlélet gyakorlati alkalmazásába.

Honnan indult mindez?

A módszertan alapjait általánosan az 1940-es évekből, a Toyota Lean szemléletű autógyártási koncepciójából származtatják. A gyökereit tekintve valóban nagyon hasonló a két koncepció, ugyanis a szoftverfejlesztő csapatok működésére is ugyanazokat az alapelveket alkalmazzuk:

- Csökkentjük a pazarlást, Növeljük a transzparenciát,
- Miközben gyorsan megoldjuk a vásárlók örökké változó igényeit.

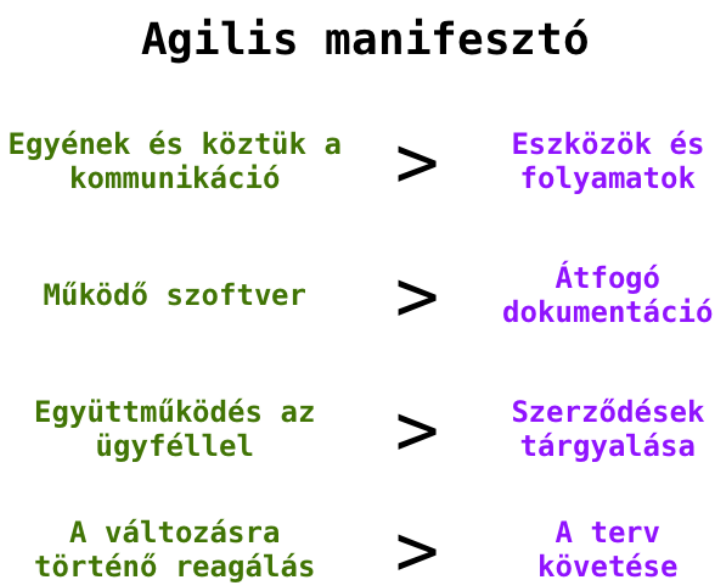
Ez valóban éles váltás, ugyanis 15 évvel ezelőtt a szoftver projektek jelentős része a hagyományosnak tekinthető vízesés-modell szerint működött. Akkoriban évekig dolgoztak a csapatok előre kidolgozott terveken, amire rengeteget költöttek, majd nagy felhajtással vezették be azokat, viszont időről-időre negatív meglepetést hozott a termék végső fogadtatása. Ehhez képest az agilis csapatok jobb együttműködéssel és minden eddiginél gyorsabb innovációval büszkélkedhetnek.

Korai alkalmazói az agilis fejlesztési módszertanoknak kis méretű önmagukban zárt csapatok voltak, akik kicsi és zárt projekteken dolgoztak. Zárt azért, mert minden szaktudás rendelkezésükre állt, nem volt semmilyen külső függőségük. Egyáltalán nem gyakori és szerencsés ez a felépítés, de találunk ilyet a gyakorlatban.

Ilyen környezetben bizonyították az agilis modellek először, hogy képesek működni, elégedettebbé és hatékonyabbá teszik a fejlesztő csapatokat. Manapság pedig már nagyobb szervezetek is eszerint működnek, de már nem egy csapatban vagy egy projektben gondolkodnak, teljes szervezeti egységek méretére felskálázzák azokat.

Természetesen ez nem jár kihívások nélkül, de nem is jelenti azt, hogy nem lehet megcsinálni!

Az agilis manifesztó négy alapértéke



2001. februárjában az utahi hegyekben 17 ember találkozott beszélgetni, síelni, pihenni, és dolgozni. Keresték a szoftverfejlesztési módszertanokban egy előremutató utat, és ennek lett az eredménye az agilis szoftverfejlesztési manifesztó jelent meg. Legyen szó extrém programozásról, Scrumról, DSDM szoftverfejlesztésről, Chrystal féle funkció-vezérelt fejlesztésekről, vagy pragmatikus programozásról, mind abban értettek egyet, hogy alternatívát kell kínálni a nehéz, dokumentációk által vezérelt szoftverfejlesztési folyamatoknak.

Ez a csoport magát Agilis Szövetségnek (angolul Agile Alliance) nevezte. Szoftverfejlesztésben dolgozó független gondolkodókról beszélünk, akik időnként egymás versenytársai voltak, mégis ez a csoport egyetértésben dolgozta ki és adta ki az agilis szoftverfejlesztés kiáltványát.

Nézzük meg közelebbről az Agilis manifesztót. Ahogy a fenti ábrán is láthatjátok mindez négy alapértékre épül. Tekintsük át ezt egészében és majd sorról sorra is elemezzük. Olyan szemmel elemezzük, hogyan tárják fel ezek a szoftverek fejlesztés egy jobb módját, azáltal, hogy cselekvésre, kollaborációra és egymás segítésére építenek.

- Az egyéneket és a köztük zajló kommunikációt többre értékeljük a folyamatokkal és az eszközökkel szemben.
- A működő szoftvert többre értékeljük az átfogó dokumentációval szemben.
- Az ügyféllel történő együttműködést többre értékeljük a szerződés tárgyalásokkal szemben.
- A változásokra való reagálást többre értékeljük egy terv követésével szemben.

Szeretném felhívni a figyelmeteket fontos részekre ezekben a mondatokban:

- Többre értékelünk valamit.
- Valamivel szemben.

Ezzel azt akarjuk kifejezni, hogy abszolút van érték a jobb oldali elemekben, viszont a bal oldali elemeket jobban értékeljük. Nagyon sok félremagyarázás van ebben, de nem szabad bedőlni, egyáltalán nem tanácsos elhagyni a jobb oldalon lévőket sem.

Vegyük példaként az első értéket, a folyamatok és az eszközök továbbra is nagyon fontosok, de inkább az egyéneket és az interakciókat kell előnyben részesítenie.

Most viszont tekintsük át részletesebben ezt a négy értéket.

Első alapérték

Az első alapérték esetében az egyéneket és köztük zajló kommunikációt vannak szemben a folyamatokkal és az eszközökkel. A szoftverrendszereket emberek építik, és ennek megfelelő elvégzéséhez mindannyiuknak együtt kell működni, illetve jó kommunikációnak kell kialakulnia az összes fél között. Nem csak szoftverfejlesztőkről beszélünk, hanem tesztelőkről, üzleti elemzőkről, projektmenedzserekről, termékgazdákról, line menedzserekről, a felső vezetésről és még nagyon sokan másokról, aki részt vesz a projektben a szervezeténél. A folyamatok és az eszközök fontosak, de mit sem érnek, ha a projekten dolgozó emberek nem tudnak hatékonyan együttműködni és kommunikálni.

Második alapérték

A második az alapértékek közül az, hogy a működő szoftverrel rendelkezzünk, és ne az átfogó dokumentációra fókuszáljunk. Nézzünk szembe a tényekkel.

Ki olvas el a 100 oldalas termék vagy technikai specifikációkat? Én egész biztosan nem.

Érdemes beszélni néhány különböző szerepkörű emberrel az üzleti oldalon, nekem az az általános tapasztalatom, hogy ők sokkal inkább azt szeretnék, ha kisebb és közepes fejlesztések gyorsan le legyenek szállítva, hogy ezekről visszajelzést adhassanak és gyűjthessenek. Azért gondolkodnak így, mert ezek a funkciók már akár elegendőek is lehetnek, ahhoz, hogy korai előnyökhöz jussanak.

De fontos kimondani, hogy a dokumentáció, nem rossz, sőt, fontos a megléte. Hosszú leírások helyett javaslok néhány olyat, amelyek a gyakorlatban nagy hasznot hoznak:

- A szoftverrendszer rétegeiről, a komponensekről és azok kapcsolatairól architekturális ábra vagy ábrák készítése.
- UML ábrák az adatbázis sémák, táblák, oszlopok felépítéséről és mindezek kapcsolatairól.
- A rendszernek és a komponenseinek telepítési útmutató.
- Esetleg néhány esettanulmány.

Érdemes ezeket A1 / A0 méretben kinyomtatni és a falakra ragasztani, hogy mindenki számára jól láthatóak legyenek. Az ilyen apró, hasznos dokumentumok szerintem felbecsülhetetlenek.

A több száz oldalas specifikációkat azonban nem tartom annak, tapasztalatom szerint 10-ből 9-szer elavultak, gyakran már akkor is, amikor elkészülnek.

Ne feledjétek tehát, hogy az elsődleges cél olyan szoftver fejlesztése, amely üzleti előnyöket nyújt, nem pedig átfogó dokumentációt.

Harmadik alapérték

A következő alapérték ügyfél-együttműködést helyezi előtérbe a szerződéses tárgyalásokkal szemben.

Az összes szoftvert a különböző belső és külső ügyfelek bevonásával kell megírni. Ahhoz, hogy sikeres legyen az elkészült termék, együtt kell dolgozni velük és gyakran be kell őket vonni a visszajelzésükért.

Ez pedig megannyi ponton megjelenik:

- Már a kutatási fázis során bevonjuk őket, különböző interjúk során a tapasztalatukat kérjük és támaszkodunk arra.
- A tervezési fázisban többször ellenőrizzük velük a terveket.
- A megvalósítási és finomítási fázisban az elkészült termékről kérjük a visszajelzésüket.
- Illetve bármelyik fázisban segíthetnek a csapattagokat motiválni, hogy igenis célja a van a munkájuknak.

A nap végén csak az ügyfél tudja megmondani, hogy mit akar valójában. Lehet, hogy kicsit ködös néha, amit mondanak, nem is adnak meg minden részletet, viszont szorosan együttműködhet velük, a termékcsapat jobban megértheti követelményeiket és inkább teljesíti is azokat.

Negyedik alapérték

A negyedik és egyben utolsó alapérték esetében reagálunk a változásokra és nem csak vakon követünk egy tervet.

Egészen általános, hogy az ügyfél vagy az egyik üzleti szponzor meggondolhatja magát abban, hogy mi készüljön el.

Ennek pedig több oka lehet:

- Új ötleteket merültek fel a korábban leszállított szoftverek használata során.
- A vállalat prioritásai megváltoztak.
- Új szabályozási változások léptek életbe.

A legfontosabb ebben a helyzetben az, hogy mindezt be kell tudnod fogadni.

Igen, előfordulhat, hogy egyes kódokat kidobnak, és időveszteséget szenvedhet, de ha rövid iterációkban dolgoztok, akkor ez az időveszteség minimálisra csökkenthető. Ki kell mondani, a változás a szoftverfejlesztések valósága, a valóság, amelyet a szoftver folyamatának tükröznie kell.

Továbbá semmi baj azzal, ha létezik egy részletes projektterv. Valójában minden olyan projekt miatt aggódnék, amelyiknek nincs ilyenje. A projekttervnek azonban elég rugalmasnak kell lennie ahhoz, hogy megváltoztatható legyen. Helynek kell lennie a helyzet megváltozásához, különben a tervek hamar irrelevánssá válhatnak.

Az agilis manifesztó tizenkét alapelve

A négy alapértékének alátámasztására a manifesztó kimond 12 követendő alapelvet is. Ezek pedig három főbb csoportra oszthatók fel:

- A szoftver rendszeres szállítása,
- Csapatkommunikáció,
- És kiváló tervezés.

Vizsgáljuk meg egymás után ezt a három csoportot, és nézzük meg az egyes Agile elveket, mielőtt áttekintjük a különböző agilis módszertanokat.

Elvek első csoportja

Ez az első szakasz a szoftvereknek a felhasználókhoz történő rendszeres kézbesítésével kapcsolatos elveket vizsgálja.

Sok régebbi szoftverfejlesztési folyamat részletes tervekkel és Gantt-diagramokkal méri az előrehaladást. A működő szoftver az utolsó dolgok egyike, amelyet el kell juttatni az ügyfélhez.

Ezzel szemben az agilis elvek a szoftverre magára összpontosítanak, és ami a legfontosabb, már a kezdetektől és gyakran szállítsunk. Már az első állapottól számítva kiemelt fontosságú az ügyfelek kielégítése értékes szoftverek korai és folyamatos szállítása.

A csapatok akkor működnek együtt a legjobban, ha bíznak egymásban. Gyakran alakul ki feszültség az ügyfél és a fejlesztő csapat között, ami abból adódik, hogy az ügyfélnek valamilyen okból kifolyólag kétségei adódnak a szállítás kapcsán. Ezt a rossz trendet azonban meg lehet azzal fordítani, hogy a fejlesztők korán és folyamatosan képesek értékes fejlesztéseket szállítani, az ügyfél elégedetté válhat, a bizalom pedig kiépül.

Az értékes szó különösen fontos. A Scrumban az ügyfél és a termékgazdák közösen döntik el, hogy mi az értékes. A termék backlogot kiemelten kezeljük, és a legértékesebb funkciókat szállítjuk először.

A következő elv arról szól, hogy a csapat a működő szoftvert rövid – pár hetes, vagy 1-2 hónapos - iterációkban szállítja, és alapvetően a rövidebb időtartamot részesíti előnyben. Fontos, hogy gyakran szállítsunk szoftvert. A Scrum és az XP például erre az elvre épül. A Scrum és az XP alatt a funkciókat 2–4 hetes sprintekben szállítják, előnyben részesítve a 2 hetet.

A következő elv szerint a működő szoftver a fejlődés az elsődleges mérője. Ezzel azt mondjuk, hogy egy szoftvernek nemcsak értékesnek és gyakran szállítottnak kell lennie, hanem folyamatosan működnie is kell. A Scrum például elvárja annak definiálását, hogy mit tekintünk késznek. Egy szempont rendszerről beszélünk, aminek teljesüléseket a szoftver potenciálisan szállítható.

A következő elv azt mondja, hogy az agilis folyamatok elősegítik a fenntartható fejlődést. Az üzleti szponzoroknak, a fejlesztőknek és a felhasználóknak képesnek kell lenniük kvázi a végtelenségig állandó ütem fenntartani. Amint a csapatok bizalmat építenek és szoftvereket szállítanak újra és újra, állandó ütemben, fenntarthatóan, anélkül, hogy bárkit túladóznának. Ez lehetővé teszi a csapat számára, hogy addig dolgozzanak, amíg elegendő értéket képesek adni a termékhez. Ennek fontos szempontja a termék rendszeres kibocsátása. Ha a csapat például negyedévente szállítható terméket tud szállítani, az sokkal könnyebbé teszi az ügyfelekkel folytatott beszélgetéseket.

Elvek második csoportja

A második csoport a csapatkommunikációval kapcsolatos agilis alapelveket vizsgálja.

Tehát, a csapat akkor tud megbízhatóan sikeres lenni, ha a csapattagoknak hatékonyan kommunikálnak. Persze egyszerűbb ezt mondani, mint valóban eszerint működni. Olyan ez, mint amikor azt tanácsolják a tőzsdézés kapcsán, hogy úgy lehetsz sikeres, ha alacsony áron vásárolsz és magasan adsz el. Az alábbi alapelvek ennek javítására adnak néhány gyakorlatiasabb eszközt.

Az első elv azt az állítást teszi, hogy üzletnek és technikai oldalnak napi szinten együtt kell működniük a termék / projekt fejlesztése során. Továbbá az egész csapatnak könnyen elérhetőnek kell lennie egymás számára. A Scrum kritikus szinkronizációs mechanizmusként használja a napi standup értekezletet. Itt a csapat beszámol arról, hogy mi valósítottak meg a legutóbbi találkozó óta, mit fognak elérni a következő találkozóig, valamint szembesültek-e akadállyal a funkciók építése során? Ezek a megbeszélések korán feltárnak fontos a kérdéseket, így azokat még kritikussá válásuk előtt van lehetőség megoldani.

A következő alapelv azt mondja ki, hogy a leghatékonyabb módszer az információ továbbítására a külső és belső fejlesztői csapaton belül a személyes beszélgetés.

Ezeket az állításokat közel 20 évvel ezelőtt mondták, akkoriban a földrajzilag elosztott csapatok még nem messze nem voltak ennyire elterjedtek. Manapság meglehetősen gyakori, hogy a csapatok a világ minden táján megosztottak, így bár a rendszeres kicsit döcögve, de megoldható, azonban a személyes kommunikáció gyakran nem lehetséges. Online értekezletek és azonnali üzenetküldő eszközök állnak rendelkezésre a kommunikáció javítására a csapatok különválásakor.

Az egész csapatot magában foglaló találkozókat mindenképp több szervezést és némi kompromisszumot igényelnek mindegyik rész oldalról. Továbbá ez költséget jelent a projekt számára, mivel a csapat egyes részeinek az értekezlet központi helyszínére kell utazniuk. Viszont hozzáadott értéket képvisel a nagyobb és fontosabb találkozók, mint például verziók, kiadások és több sprint tervezése esetében.

A kiszervezett-erőforrások felhasználása esetén gyakran járunk el úgy, hogy a kiszervezett-csapat néhány képviselőjét pár hétre, vagy 1-2 hónapra a belső csapat mellé rotáljuk és közel dolgozhassanak egymáshoz. Ez lehetővé teszi a csapattagok számára, hogy személyesen kapcsolatot alakítsanak és megismerkedjenek.

Továbbá azt is lehetővé teszi, hogy az kiszervezett-csapat képviselői első kézből származó információkkal térjen haza, ami segít a távoli csapatnak értékes betekintést nyerni. Ez minden esetben előnyös helyzetet teremt, mert az kiszervezett-csapatok tagjai várják a tapasztalatokat a fő termékcsapattól.

Szerencsés a szorosan együtt dolgozó csapatokat egy tartani, egy helyen, legalábbis egy időzónában és kulturális körben. Ez lehetővé teszi az egyes csapatok számára, hogy folyamatosan szinkronizáljanak, valamint részesüljenek a személyes kommunikáció előnyeiből. Az elkülönített csapatok a gyakorlatban kihívást és költséget jelentenek, de nagy projekteknél a költségek növekedésével megérheti.

A következő elv kimondja, hogy a legjobb architektúrák, követelmények és általában eredmények az önszerveződő csapatoktól származnak. A csapat tudja, hogyan lehet a

legjobban megvalósítani valamit. Ők a szakértők; ez azonban nem jelenti azt, hogy a helyes eredmény mindig önmagában fog bekövetkezni. Minden egyén a saját növekedésében és karrierjében más és más helyen van. Létezik egy olyan kifejezés, hogy szolgáló vezető, az agilis közösségben alkalmazzák ezt szívesen, és a tipikus parancselvű projektmenedzsereket váltja ki. A szolgáló vezető megfigyeli, hogy a csapat tagjai hogyan dolgoznak és hogyan kommunikálnak egymással. Felteszi az esetleges problémák feltárásához szükséges kérdéseket, és rávilágít azokra a területekre, amelyeket az egyes csapattagok esetlegesen nem vesznek észre. Fontos, hogy a szolgáló vezető mindenkivel tisztelettel és méltósággal bánjon, még akkor is, ha adott esetben nagy a feszültség. A csapattagok közötti konfliktusok normálisok és gyakran helyes célok érdekében történnek. A jó vezető megérti ezt és támogatja az egészséges viták folytatását. Az önszerveződő csapatok nagyon ritkán állnak össze automatikusan, a vezető azonban megfelelő útmutatással és tanácsokkal nagy mértékben tudja támogatni azt.

A következő alapelv kimondja, hogy motivált egyének köré építsünk termékeket. Adja meg nekik a környezetet és a szükséges támogatást, és bízzon bennük a munka elvégzésében. Ez az önszerveződő csapatok kiterjesztése. Van ebben néhány fontos szó. Soha senki sem vallaná be, hogy nem motivált. A vezetők figyelnek a csapattagok törekvéseire és céljaira, és lehetőség szerint összehangolják ezeket a célokat a termék szükségleteivel. Az emberek akkor teljesítenek a legjobban, ha olyasmit csinálnak, amiért rajonganak.

A jó agilis vezető az esetek jelentős részében megvédi a csapatot a külső zavaró tényezőktől is. A Scrumban például egy csapat elkötelezi magát egy funkció-csomag megvalósítása mellett. Bármi, ami ettől elvonja a figyelmet, az kockázatot jelent. Azzal, hogy ott van a csapat mellett, a vezető a sikerhez szükséges környezetet és támogatást nyújtja számukra. A bizalom nem automatikus, de idővel kiépül, azonban könnyű elveszítheti. A csapattagoknak bízniuk kell egymásban, így a konfliktusok is produktívabbak.

A következő elv szerint a csapat rendszeres időközönként visszajelzéseket ad és kap. Ezt azért teszi, hogy tanuljon és ezáltal hatékonyabbá válhasson, ezeknek megfelelően igazíthassa viselkedését. A Scrum a retrospektív ceremóniákat használja erre a folyamatra. A csapatoknak jellemzően segítségre van szükségük ahhoz, hogy ez a tevékenység hatékony legyen. Az embereket kihívás érheti, amikor valódi önreflexióba kell bocsátkozni. Ez mind az agilis működésben megjelentik, és az agilis elvek mindegyike összefügg egymással.

A retrospektív ceremónia tökéletes hely a csapat számára az elmélkedésre és az önfejlesztésre. A scrum master feladata, hogy a felszínre hozz ezt az önreflexiót. Miután meghatároztuk a fejlesztendő területeket, azokon valóban fejlesztenünk is kell. Ha a csapatok erre időt szánnak, azonban nem történik ennek hatására semmi előrehaladás,

akkor ezeket a visszajelzéseket időpazarlásnak fogják tekintik. Ismét a scrum master feladata, hogy tapintatosan emlékeztesse a csapatot az általuk elfogadott fejlesztési területek haladására. Néhány csapat ezeket a pontokat jól látható helyekre szokta akasztani, van, ahol a falra is kikerül.

Elvek harmadik csoportja

Az agilis elveknek utolsó csoportja a tervezés kiválóságáról szól. Ha az agilis megközelítés hatékonyságát szeretnénk maximalizálni, akkor kiváló szoftveres megoldásokat kell megterveznünk és felépítenünk. A kiváló tervezés márpedig egy folyamatos feladat. Ahogy az agilis működés többi része is, így ez is éberséget és komoly figyelmet igényel a változó eseményekhez való alkalmazkodáshoz.

Minden előtt tehát, figyelmet kell fordítani a műszaki kiválóságra és a jó tervezésre, mert az fokozza az agilitást. Termékeink fejlődésével fokozott figyelmet kell fordítanunk a műszaki kiválóságra és a tervezésre. Nehéz megtalálni és megteremteni az egyensúlyt aközött, hogy a helyes terméket és funkciókat építsük és azokat a helyes módon építsük fel. Óvatosnak kell lennünk a törékeny rendszerekben történő szállítással kapcsolatban is. Például, ha megépítünk néhány új képességet és alkalmazásunk szétesik, mint egy kártyaház, akkor nem jó döntéseket hoztunk.

Az extrém programozás és érintőlegesen a Scrum is teszt-vezérelt fejlesztést és automatizált felépítést javasol a törékeny megoldások elkerülése érdekében. Ha megfelelő automatizált unit-tesztek állnak rendelkezésünkre, amelyek valamilyen automatizált CI/CD folyamat, akkor nagyon sok problémát már a buildelés során felfedezhetünk. Minél magasabb a kód lefedettsége és minél közelebb kerülünk a folyamatos integrációhoz, annál jobbak lesznek a megoldásaink.

Minden az egyensúlyról szól. Csak idő kérdése és a megoldásunk technikai adósságokat fog felhalmozni. Ahogy folyamatosan mérlegelünk és döntünk a helyes mindkét oldalán, úgy ez biztosan megtörténik. A legjobb, ha minden sprintre betervezzük néhány fontosabb technikai adósság javítását az ügyfeleknek értéket képviselő funkciók mellé.

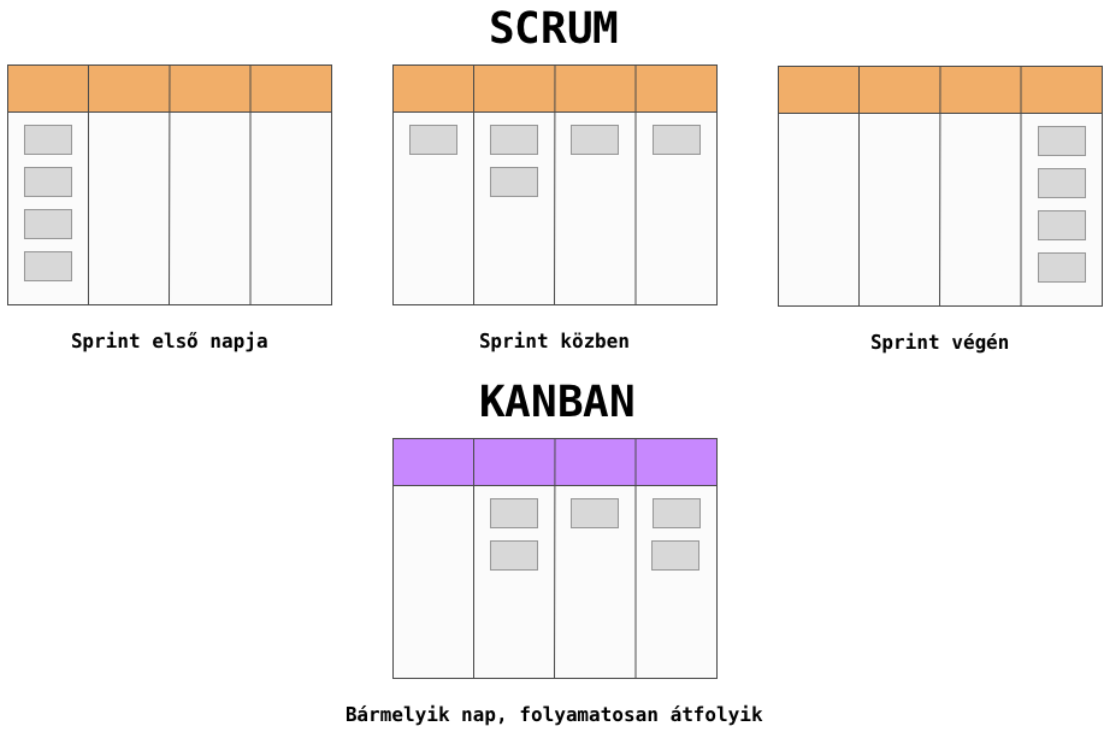
A következő elv szerint az egyszerűség elengedhetetlen, maximalizálni kell az el nem végzendő munka mennyiségét. Az agilitás arról szól, hogy bármikor megfelelő mennyiségű munkán dolgozzunk, és ne többön. Elég kicsi felhasználói történeteket kell kínálnunk a munka elvégzéséhez. Azt kell megépítenünk, amiről tudjuk, hogy most szükségünk van. Nem szabad olyasvalamibe jelentős időt befektetni, amire egyszer és talán szükség lesz. Kritikus az általunk használt szoftverek teljes megértése.

Ez a végső állítás az, hogy fogadjuk örömmel a változó követelményeket még a fejlesztések késői szakaszában is. Az agilis folyamatok megváltoztatják az ügyfelek versenyelőnyét. Ez meg fogja ijeszteni azokat a csapatokat, akik hozzászoktak a vízesés

projektekhez. Első pillantásra őrültségnek tűnik a fejlesztés végén változtatni. A késői fejlesztések gyakran a teljes termék későbbi kiadását jelentik. A Scrum például rövid iterációkban szállít funkciókat. Mivel a funkciókat rövid iterációkban szállítjuk, így a változás az egész folyamat része. A Scrumban a változást a termékgazda irányítja. Rajta múlik, hogy megértse, mi a versenyelőnye az összes backlogban maradt funkciónak.

Tehát az elmúlt fejezetekben megvizsgáltuk az agilis manifesztő alapvető értékeit és elveit. Ezt követően pedig vessünk egy pillantást a gyakorlatban használt különböző agilis módszerekre.

Agilis metodológiák áttekintése, avagy Scrum, Lean és Kanban



Ebben a részben megvizsgáljuk a manapság alkalmazott agilis módszereket. Először is a Scrumot vizsgáljuk. A Scrum egy kifejezetten könnyű menedzsment keretrendszer, amely széles körű vonzerővel bír számos iteratív és inkrementális projektek kezelésére.

Ken Schwaber, Mike Beedle, Jeff Sutherland és még sokan mások jelentősen hozzájárultak a Scrum evolúciójához az elmúlt másfél évtizedben. Különösen az elmúlt 5-6 évben a Scrum egyre nagyobb népszerűségnek örvend a szoftverfejlesztő közösségben, egyszerűsége, bizonyított termelékenysége, valamint azért, mert emellett számos más agilis mérnöki gyakorlatok is jól beépíthető lesz.

Ezt követi az extrém programozás vagy röviden csak XP. Az extrém programozást eredetileg Ken Beck találta ki, az egyik legnépszerűbb és legvitatottabb agilis módszerként jelent meg. Az XP a magas színvonalú szoftverek gyors és folyamatos

szállításának fegyelmezett megközelítése. Elősegíti az ügyfelek magas szintű részvételét, a gyors visszacsatolási ciklusokat, a folyamatos tesztelést, a folyamatos tervezést és a szoros csapatmunkát annak érdekében, hogy a működő szoftvereket nagyon gyakori időközönként, általában 1-3 hetente szállítsák. Az eredeti XP recept négy egyszerű értéken alapszik:

- egyszerűség,
- kommunikáció,
- visszacsatolás
- és bátorság.

És van még egy sor támogató gyakorlat. Ezek pedig sorban a következők...

- a tervezési játék,
- a kis kiadások,
- az ügyfél-elfogadási tesztek,
- az egyszerűsített tervezés,
- a páros programozás,
- a tesztvezérelt fejlesztés,
- az újrakezdés,
- a folyamatos integráció,
- a kollektív kód tulajdonjoga,
- a kódolási szabványok,
- a metaforák,
- és a fenntartható ütem.

A következő a Crystal módszertan-család. Ami mind közül az egyik legkönnyebben alkalmazkodó megközelítés a szoftverfejlesztéshez. Számos tényező vezérli, mint például:

- a csapat nagysága,
- a rendszer kritikussága,
- és a projekt prioritásai.

Azzal a felismeréssel foglalkozik, hogy minden projekt esetében kis mértékben szükség lehet irányelvekre, gyakorlatokra és folyamatokra, amelyek megfeleljenek a projekt egyedi jellemzőinek.

Ezután jutunk a dinamikus rendszerek fejlesztési módszertanokhoz, amelyek közül talán a DSDM a legismertebb. A DSDM 1994-re nyúlik vissza, és abból nőtt ki, hogy iparágilag szabványos projekt-keretet szerettek volna biztosítani az úgynevezett gyors alkalmazásfejlesztésnek, vagy röviden RAD-nek.

Az 1990-es évek elején a RAD rendkívül népszerű volt, azonban ez a megközelítése meglehetősen strukturálatlanul alakult. Ennek eredményeként 1994-ben hozták létre és hívták össze a DSDM konzorciumot azzal a céllal, hogy kialakítsák és előmozdítsák a gyors szoftver-szállítás közös ipari keretrendszerét.

1994 óta a DSDM módszertana úgy fejlődött, hogy átfogó alapot teremtsen az agilis és iteratív szoftverfejlesztési projektek tervezéséhez, kezeléséhez, végrehajtásához és méretezéséhez.

A DSDM sor kulcsfontosságú alapelven alapszanak, amelyet elsősorban az üzleti igények és értékek:

- az aktív felhasználói részvétel,
- a felhatalmazott csapatok,
- a gyakori kézbesítés,
- az integrációs tesztelés és az érdekelt felek közötti együttműködés körül forognak.

A DSDM kifejezetten az üzleti célokra való alkalmasságot nevezi meg a rendszer átadásának és elfogadásának elsődleges kritériumaként, amely a rendszer hasznos 80% -ára összpontosít, amely az idő 20% -ában telepíthető.

Ezután jutunk a funkció-vezérelt tervezéshez, amit röviden FDD-nek nevezünk. A módszertant eredetileg Jeff De Luca fejlesztette ki. Az FDD első inkarnációi De Luca és társa, Peter Coad együttműködésének eredményeként következtek be. Az FDD egy modell-vezérelt rövid iterációkra épülő folyamat.

- A folyamat a modell átfogó alakjának megállapításával kezdődik.
- Ezután jellemzően kéthetente alternálva ismétlődő tervezés és építkezés lépések sorozata.
- A funkciók célszerűen kis méretűek és magas értéket képviselnek az ügyfél szemében.

Az FDD a fejlesztési folyamat többi részét a szolgáltatásnyújtás köré alakítja, a következő nyolc gyakorlat felhasználásával:

- domain objektumok modellezése,
- fejlesztés funkciónként,
- komponensek és osztályok tulajdonosának definiálása,
- feature-csapatok kialakítása,
- ellenőrzések elvégzése,
- konfigurációkezelés,
- rendszeres kiadások,
- a fejlődés és az eredmények láthatóvá tétele.

A példáinkat a Lean módszertannal folytatjuk. Ami szintén egy iteratív szoftverfejlesztési módszertan, amelyet egy házaspár, Mary és Tom Poppendieck fejlesztettek ki. A Lean szoftverfejlesztés alapelvei és gyakorlatai sokat köszönhetnek a Lean nagyvállalati mozgalomnak és az olyan vállalatok gyakorlatának, mint például a Toyota.

A Lean szoftverfejlesztésben két értékre összpontosítanak a csapatok:

- értéket juttassanak el az ügyfélhez,
- valamint az értéket szolgáltató mechanizmusok hatékonyságára.

A Lean alapelvei közé tartoznak:

- a pazarlás megszüntetése,
- a tanulás fejlesztése,
- a döntések kitolása,
- a lehető leggyorsabb teljesítés,
- a csapat felhatalmazása,
- az integritás építése
- és az átfogó képre fókuszálás.

A Lean kiküszöböli a pazarlást azáltal, hogy csak a rendszer valóban értékes tulajdonságait választja ki, prioritásként kezeli és kis tételekben szállítja azokat. Hangsúlyozza a fejlesztési munkafolyamat sebességét és hatékonyságát, valamint a fejlesztők és az ügyfelek közötti gyors és megbízható visszajelzésekre támaszkodik.

A példáink sorát a Kanbannal zárjuk. A Kanban-t a szoftverfejlesztésben pull-base tervezési és végrehajtási módszerként alkalmazzák. Ahelyett, hogy előre megtervezné a munkadarabokat, és a csapat munkasorába helyezik azokat, a csapat pedig folyamatosan jelzi, ha készen állnak a további munkára, és behúzza azokat a végrehajtásba.

Kanban historikusan kártyákat használ arra, hogy jelezze egy tétel szükségességét. A szoftverfejlesztő csapatok számára ezeket a kártyákat egy kanban táblára helyezik, amelyet oszlopokba és sorokba rendeznek.

Az *oszlopok* a munkaelem különböző állapotait mutatják a kezdeti tervezéstől a vevői elfogadásig. A csapat által használt speciális oszlopoknak meg kell felelniük a csapat igényeinek, és a környezetükhöz kell igazítani.

A kanban tábla *sorai* a munkaelemeket képviselik. A munkaelemeket néha területeken belül csoportosítják, például a funkció-készletek vagy a kategória-típusok.

A Kanban a csapat teljesítményének maximalizálására összpontosít. A cél elérésének egyik módja a *work-in-progress* (röviden *WIP)* határértékek alkalmazása az egyes

oszlopok esetében. Ez azt jelenti, hogy az egyes oszlopokban maximalizáljuk az oda bemozgatható kártya sorok számát.

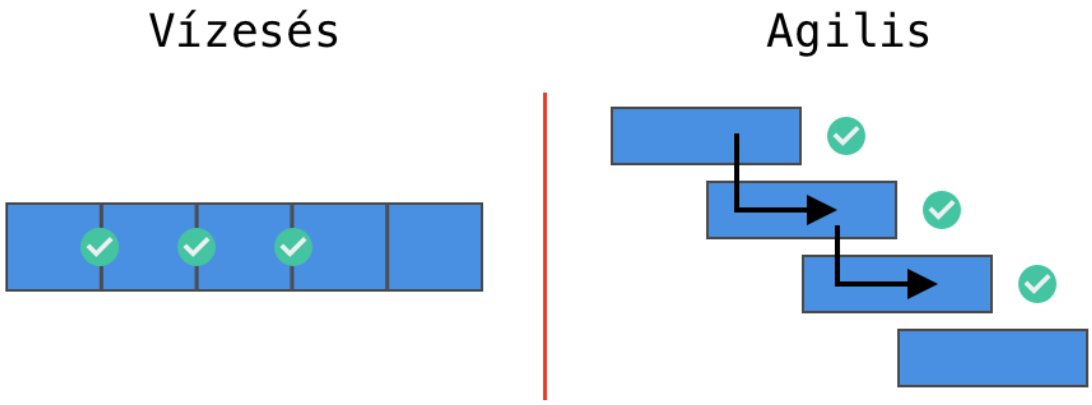
Ez a módszer felhívhatja a figyelmünket a pazarlásra. Célunk ezen rendszeren keresztül a munkaelemek optimális áramlására összpontosítson, minimalizálva a kapcsolódó pazarlást.

Kanban lehetővé teszi a csapatok számára a folyamatok optimalizálását, miközben tiszteletben tartják és biztosítják a fenntartható tempót.

Vízesés vs agilis

Tekintsük át az alapokat, avagy miben tér el az agilis gondolkodás az őt megelőző szemléletektől, módszertanoktól?

A tradicionális projekt menedzselési stílusok - mint amilyen a vízesés is - fázisokra építenek. Az alábbi ábra jól szemlélteti a vízesés modell lényegi koncepcióját. Ez a stílus lényegében a teljes termékfejlesztés sikerét – nem ritkán 2-3-5 év munkáját – a folyamat végén egyetlen „Nagy Bumm" kiadásra teszi fel. Ebben nagy a kockázat, igaz? Amikor a projekt lezár egy fázist, akkor meglehetősen nehéz felülvizsgálni azt és eltérni a korábbi döntésektől, mivel a módszertan mindig csak előre tolja a csapatokat.



A tradicionális projekt vezetési stílusok gyakran hoznak létre ún. „kritikus utakat", amikor is az egész projekt, vagy legalábbis annak egy része, nem haladhat tovább, amíg egy blokkoló probléma feloldásra nem kerül. Sőt, fogalmazzuk meg ezt még fájdalmasabban: a vásárló addig nem léphet párbeszédbe a termékfejlesztő csapattal vagy annak képviselőjével, amíg a termék nincs teljesen készen. Még akkor sem, ha fontos problémák jelentkeztek a termék tervezése vagy implementálása során, ezek bizony egészen a kiadásig ott fognak rejtőzködni, és ott szembesül vele mindenki.
Állítsuk ezt szembe az agilis projektvezetési stílussal, ahol a fejlesztést iteratívan közelítjük meg, és ahol rendszeresen visszajelzéseket kapnak a csapatok. Ezek az iterációk lehetővé teszik a csapatoknak, hogy a korábban említett blokkoló problémák megoldásáig áttérjenek más területekre és valóban produktívak lehessenek.

A Scrum alapvető értékei

A Scrum egy agilis projektvezetési keretrendszer, amely az alapjait tekintve kifejezetten egyszerű, de az ördög ezúttal is a részletekben rejtőzik.

Az alap gondolatmenet lényege az alábbi pár pontban fogalmazható meg:

- A csapatokat és a feladatokat jól kezelhető, kis méretű részekre osztja fel
- Folyamatos, jól megszokható ritmust biztosít a munkának
- Törekszünk arra, hogy a csapatokban minden a hatékony szállításhoz szükséges szakértelem megtalálható legyen

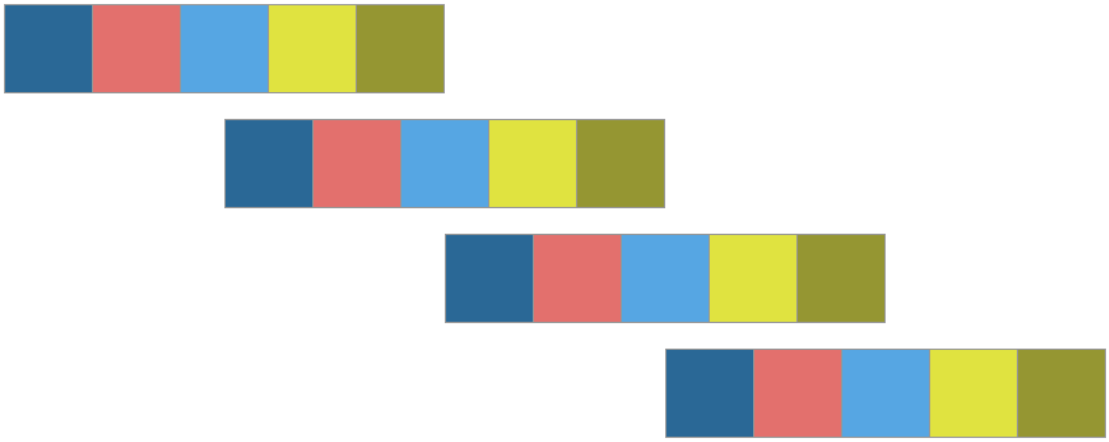
Oszd meg és uralkodj!

Mi emberek úgy működünk, hogy jellemzően sokkal kiszámíthatóbban tudunk dolgozni, ha a feladat kevésbé komplex, az időtartam rövidebb, és kevesebb emberrel kell együttműködni.

Ezért a Scrum tervezőinek az egyik első lépése az volt, hogy a jellemzően nagy dolgokat kisebb részekre osztották:

- Az időt előre meghatározott méretű iterációkra, ún. sprintekre osztották.
- A csapatokat a szakértelmek mentén zárt és teljes egységekre.
- A feladatok esetében is a kisebb részekre darabolás az egyértelmű ajánlás.

Az idő felosztása



Az idő esetében onnan indult az alapítók gondolkodása, hogy sokkal könnyebben elúsznak a határidők és nehezebb észre venni, ha valami időközben félremegy, ha a feladatok kontrolláltalanul folynak előre.

Ha azonban fix és jól tervezhető iterációkra osztjuk fel az időt, akkor kezelhetőbb lesz az egész munkafolyamat. Folyamatos ellenőrzőpontok alakulnak ki, amiknek köszönhetően sokkal jobban követhető a tényleges haladás.

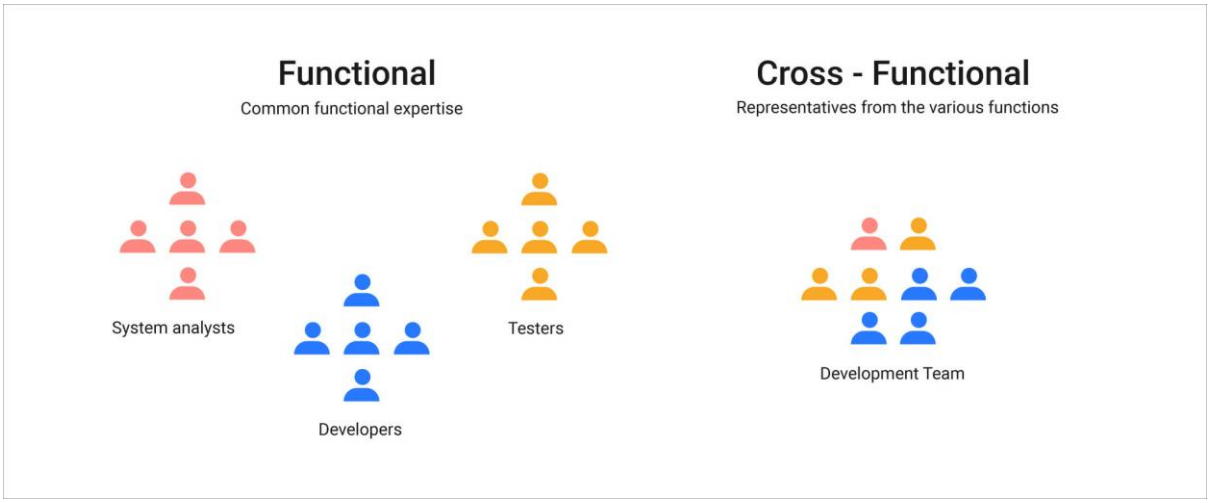
Érdekes szóhasználat egyébként a sprint, mert ugye a szoftver projektekre leginkább a maraton szó jellemző, legalábbis én így tapasztaltam. Valójában arról van szó, hogy a nagy maratont sok kis rövidebb sprint etapra osztjuk fel és azokat folyamatosan mérjük. Meglehet, a futás szempontjából nem ez a leginkább tanácsolt megközelítés, de mint hasonlat megállja a helyét.

Hogy mi az ideális időtartam, azt természetesen csapata válogatja. Az általános ajánlás 1-4 hét szokott lenni. A legtöbb csapat, akivel találkoztam az arany középutas 2 hetes időtartamok mellett tette le a voksát, mert így ideális a szükséges ceremóniákra fordított idő és az elvégezhető munka aránya. Ez legtöbbször így is van.

Újonnan összeállt csapatoknál, vagy ahol sok változás történt, sok új belépő, érdemes szerintem megfontolni az 1 hetes sprinteket is. Sokan félnek ettől, úgy gondolják túl sok időt elvesznek a ceremóniák. Ez a kezdeti időszakban minden bizonnyal így is lesz, de kitartó optimalizálással ez bőven viselhető időtartamra szorítható.

Láttam arra is példát, hogy egy csapatnak a 6 hetes időtartam működött jól. Ez is lehet jó döntés egy jól összeszokott és misszió orientált csapat esetében. Azonban ez erősen összefügg a termékfejlesztési döntésekkel, érdemes ezeket jól összehangolni.

A csapatok átalakítása



Kép forrása

Régi igazság, hogy minél nagyobb egy csapat, akik közös célokért dolgoznak, annál nehezebb azt megfelelő hatékonysággal egyben tartani. Csak egy a sok probléma közül, hogy egyesek nem figyelnek oda, hogy miken dolgoznak a többiek és simán ugyanarra a feladatra fordulnak rá.

A másik nagy problémakör, hogy megannyi szervezetben az emberek funkciók szerint silókba gyűlnek össze. Példa az ilyen silókra, amikor külön szervezeti egységben vannak a fejlesztők, a grafikusok, a UX kutatók, az üzleti elemzők, az üzemeltetők, adatmérnökök, markertingesek, értékesítők stb.

Ilyen esetekben az egyes projektek vezetőinek komoly előkészületeket kell tennie:

- Erőforrástervezést végezni,
- Gantt-diagramokat kezelni,
- Egyezkedni az egyes silók vezetőivel,
- Csúszás, betegség, vagy hiba esetén folyamatosan sakkozni az erőforrásokkal.

Nem könnyű feladat, és nagyon ritkán szokott működni. A tapasztalatom az, hogy ezek a feladatok rengeteg értékes időt vesznek el a valódi értékteremtéstől.

A Scrum mindkét problémakörre tud megoldást javasolni:

- Osszuk fel a csapatokat kisebb, jól kezelhető részekre
- Szervezzük azokat kereszt-funkcionálisan

A csapatok ideális mérete ebben a rendszerben 6-8-10 fő között van. Jelentős mértékben függ a kitűzött céloktól, van példa kisebbre és nagyobbra is. Attól is, hogy milyen

szakértelemre van szükség. Milyen szintű emberekből épül fel a csapat. Vezetői szempontból ez egy bizonyosan jól kezelhető méret.

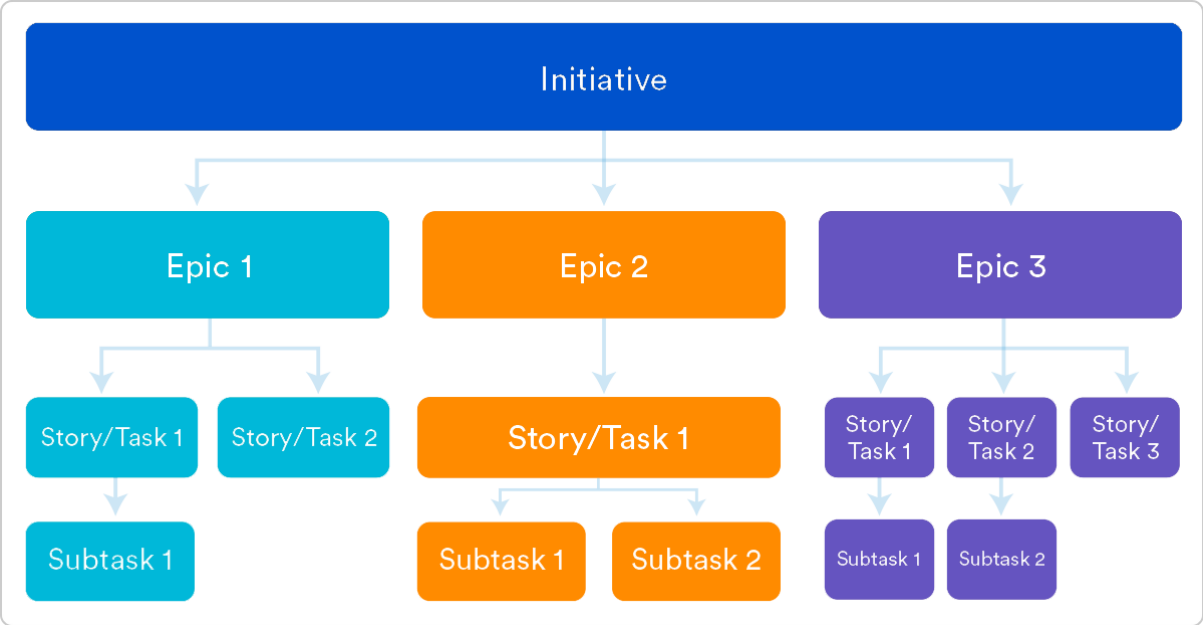
A kereszt-funkcionális csapat azt jelenti, hogy minden szükséges szerepkörű kolléga egy csapatban van, egy vezetővel. Nincs szükség alkudozásra, az erőforrás mindig jelen van. Ez egy ideális helyzet, de persze el is kell tudni látni minden embert megfelelő mennyiségű munkával.

Például pazarlásnak gondolom teljes munkaidőben grafikus a csapatban tartani, ha mindössze havi 4-5 napra elegendő valódi értékteremtő munkája van. Másik példa a marketing szakember, mivel a fejlesztés fókuszú hónapokban nem kell feltétlen végig a csapatok mellett lennie, elég, ha később lép csatlakozik a csapathoz.

A kulcs a valódi értékteremtés. Emiatt számos szervezet dönt úgy, hogy a kereszt-funkcionális csapatokba valóban a legtöbb szükséges szerepkör megtalálható, de bizonyos funkciókat megtart silókban, amikhez szükség esetén lehet nyúlni.

Még egy utolsó fontos szempont. Széleskörű tapasztalatok támasztják alá, hogy a hosszú távon együtt dolgozó csapatok hatékonyabbá képesek válni, mint az ad-hoc és rövid időre összeverbuváltak.

A feladatok felosztása



Kép forrása

Ahogy arról már szó volt pár fejezettel korábban, a szoftver projektek leginkább a maratonhoz hasonlíthatók, és bizony nagyon sok feladat hosszú időt vesz igénybe.

Ezzel pedig két tipikus probléma szokott megtörténni:

- Minden kontroll ellenére túlfolynak a rájuk szabott határidőn
- Nem az készül el, amit terveztünk, vagy az, amire az ügyfél vágyott

Meglehetősen gyakori a projektek során, hogy mindkettő megtörténik. Csúszik is, nem is az készül el, és akkor még többet csúszik. Ez meglehetősen sok stressz forrása lehet.

A Scrum megoldási javaslata erre az, hogy a kitűzött feladatokat daraboljuk fel kisebb, jobban kezelhető részekre. Hogy mekkora részekre, arról örök hitviták lesznek az agilis vezetésben jártas szakemberek között.

Vannak sokan, akik szeretnek mindent feladatot pár órás, legfeljebb egy napos részekre darabolni, mindent külön megbecsülni.

Ez jó megközelítés lehet, ha még nincs összeszokva a csapat és nem túl tapasztaltak a kollégák. Viszont meglehetősen munkaigényes, jól kell felmérni, hogyan osztják be az idejüket a vezetők.

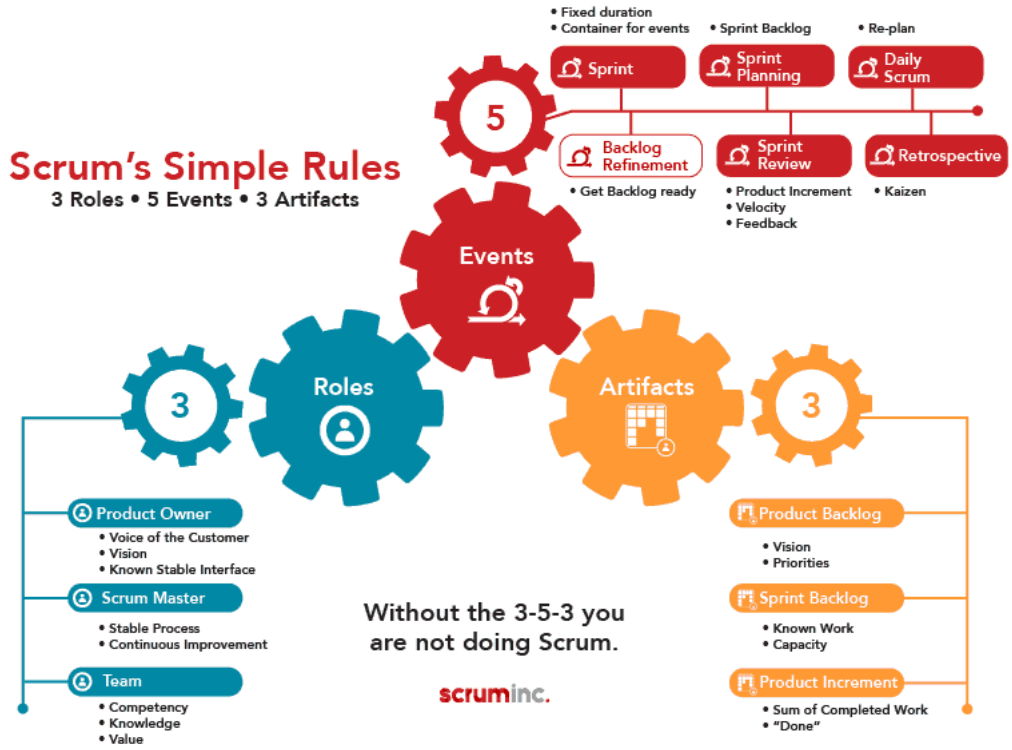
Vannak mások, akik azonban egyben hagynak több napos, vagy akár 1-2 hetes fejlesztéseket is, mert azt mondják, hogy így teremtenek valódi értéket, és nem akarnak elveszni a sok pici feladat között.

Ez is lehet jó megközelítés, de jellemzően összeszokott és tapasztalt csapatokkal szokott jól működni.

Akármelyik is választom, érdemes folyamatosan kommunikálni a fejlesztők felé, hogy mi az a cél, amiért dolgoznak és összességében mit is akarunk leszállítani az ügyfélnek. Kerüljük el azt a problémát, hogy mindenki csak a maga pici feladatával foglalkozik és azon túl felteszi a kezét.

A Scrum eszközkészlete

A Scrum tervezőinek egyértelmű célja volt, hogy egyszerűen átlátható eszköztárt tegyenek le mindenki elé, hogy a megértés biztosan ne jelentsen akadályt. Események, szerepkörök és dokumentumok, az alábbi 3x2x4-as tábla jól összefoglalja mindezeket.



Kép forrása

Szerepkörök	Scrum Master	Product owner	Csapat	
Dokumentumok	Termék backlog	Sprint backlog		
Események	Tervezés	Napi Standup	Demó	Retrospektív

Egyébként ez jó ellenőrzése is az agilis metodológiai bevezetésének, ha bármelyik hiányzik, akkor bizonyosan érdemes kérdéseket feltenni a miértek kapcsán.

Szerepkörök

Most pedig tekintsük át ezeket!

Scrum Master

A tradicionális projektekben a csapatokokat projektmenedzserek vezették, az agilis e pozíció helyére vezette be a Scrum Master szerepkört. Gondolkodásmódjukat a következő főbb célok határozzák meg:

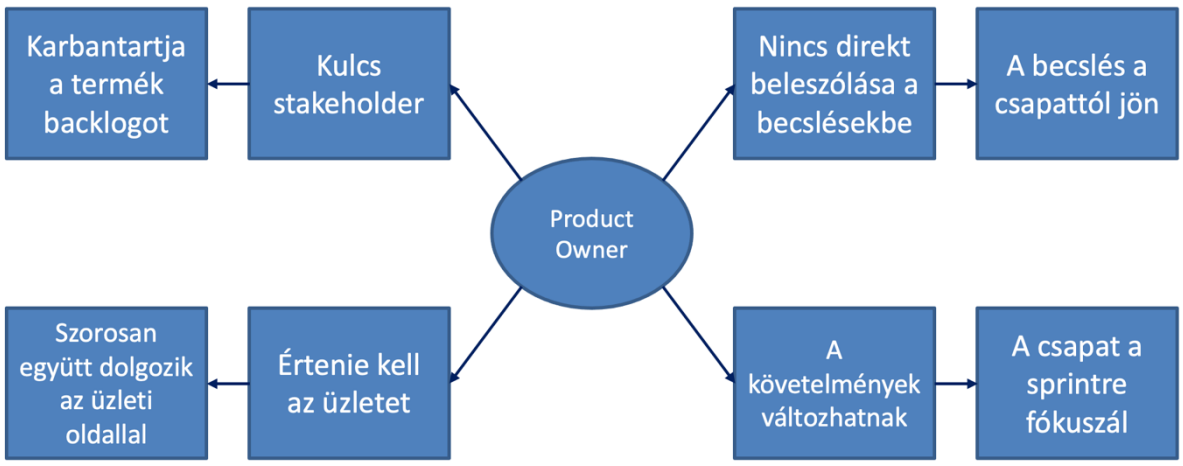
- A Scrum folyamatok védelme, betartatása,
- A csapat elé kerülő akadályok elhárítása, biztosítani azt, hogy a csapat zavartalanul tudjon dolgozni

Nagyon fontos különbség, hogy a projektmenedzserekkel szemben, hogy a Scrum Master hivatalosan, hagyományos értelemben véve nem főnöke a csapatban senkinek sem.

Gyakori viták tárgya ez a szerepkör, a radikálisak a létjogosultságát vitatják, a szelídebbek inkább csak azt, hogy szükség van-e rá teljes időben. Megint mások pedig azt vetik fel, hogy egy technikai vezető hasznosabb lenne.

Mindegyikben van igazság, valóban dolgozni kell azon, hogy adott csapat esetében meglegyen a megfelelő értékteremtése. Számos szervezet esetében láttam olyat, hogy egy Scrum Master több csapatot kiszolgált. De arra is vannak példák, hogy ezt a szerepkört egy termékgazda és egy technikai vezető közösen átveszik.

Product Owner

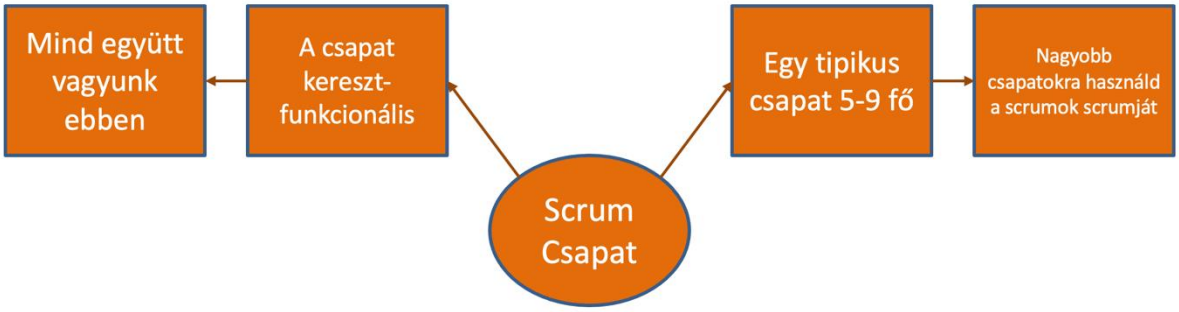


Sokan úgy fogalmazzák meg - szerintem viszonylag találóan -, hogy ez a szerepkör jelenti az ügyfél hangját a projektekben. Meghatározzák, hogy minek kell elkészülnie, és a prioritásokon keresztül azt is, hogy melyek képviselik a legmagasabb hozzáadott értéket.

A trükk az, hogy a Product Owner sem főnöke senkinek sem a csapatban, a csapattal együtt dolgozik, akikkel így közösen hoz létre értéket.

Nagyon nehéz feladatkörről van szó. Ahogy mondtam a szoftverek fejlesztés szinte mindig maraton, a sikerét döntések ezrei fogják meghatározni. A döntésekre való képesség, az azokból való tanulás és az irány finomítása nagy mértékben meghatározza, hogy mennyire lesz sikeres valaki ebben a szerepkörben.

Csapat



A Scrum szabályai szerint a csapat célja az, hogy folyamatosan szállítsa a szoftverek kisebb-nagyobb növekményeit.

Korábbi fejezetben már volt arról szó, hogy érdemes egy logikus szintig kereszt-funkcionális csapatokban gondolkodni, erre nem is térnék vissza. Inkább még egy plusz gondolat: véleményem szerint egy jól működő csapatot az ownership különbözteti meg a többitől. Ne csak azt szállítsák le betűről-betűre, ami le van írva, akkor is, ha tudják, hogy rossz, hanem tényleg jó dolgot akarjanak kiadni a kezükből. Ezt egyébként nagyon nehéz elérni, ügyesen kell tudni kommunikálni számukra a missziót és folyamatosan szem előtt tartani a célokat.

Hogyan működjünk jól agilisan?

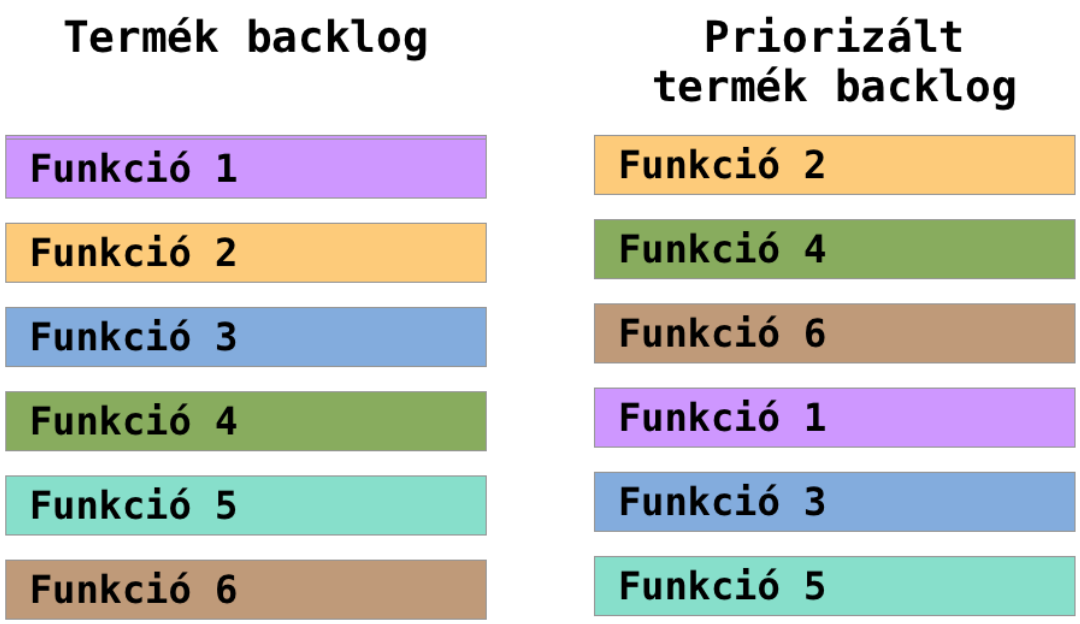
Amikor hagyományosról az agilis útra térünk át, akkor a csapatnak és a döntéshozóknak két fontos koncepciót kell elfogadniuk:

- A termékgazda fókusza az, hogy optimalizálja az értékét annak a munkának, amivel a fejlesztő csapat elkészül.
- A fejlesztő csapat tehát arra épít, hogy a termékgazda a legmagasabb értékű feladatokat priorizálja előre.
- A fejlesztő csapat csak olyan munkát fogad el, amire kapacitása van. A termékgazda nem nyom a csapatra munkákat vagy kötelezi őket önkényesen munkára.
- A fejlesztő csapat akkor húz be új munkát a backlogból, amikor elkészült az előzővel.

Ezek egyébként leírva nagyon jól hangzanak, de implementálni nem mindig egyértelmű, a tapasztalat az, hogy mindenhol egy kicsit másképp kezelik ezt. Bizony a határidő sokszor nagy úr.

Dokumentumok

Termék backlog



A Scrumban az agilis termék backlog a feature-ök priorizált listáját jelenti. A backlog minden feature esetében egy rövid leírással párosul, ami az elképzelt termék funkcióit írja le. Ezek a rövid ismertetők nagy könnyebbséget jelent, ugyanis így nem kell előre jelentős energiákat tenni terjengős követelmény dokumentumok megírásába.

Az első sprint tipikusan azzal kezdődik, hogy a termékgazda és a csapat közösen kialakítják a backlog első priorizációját, ami már szinte minden esetben elégséges lesz a kezdéshez. Ezt követően ez egy élő dokumentum lesz, ahogy halad a csapat előre, növekszik a szoftver, többet tanulunk az ügyfelektől a környezetükről és a termékről, úgy fog ezekkel együtt fejlődni, adaptálódni a backlog is.

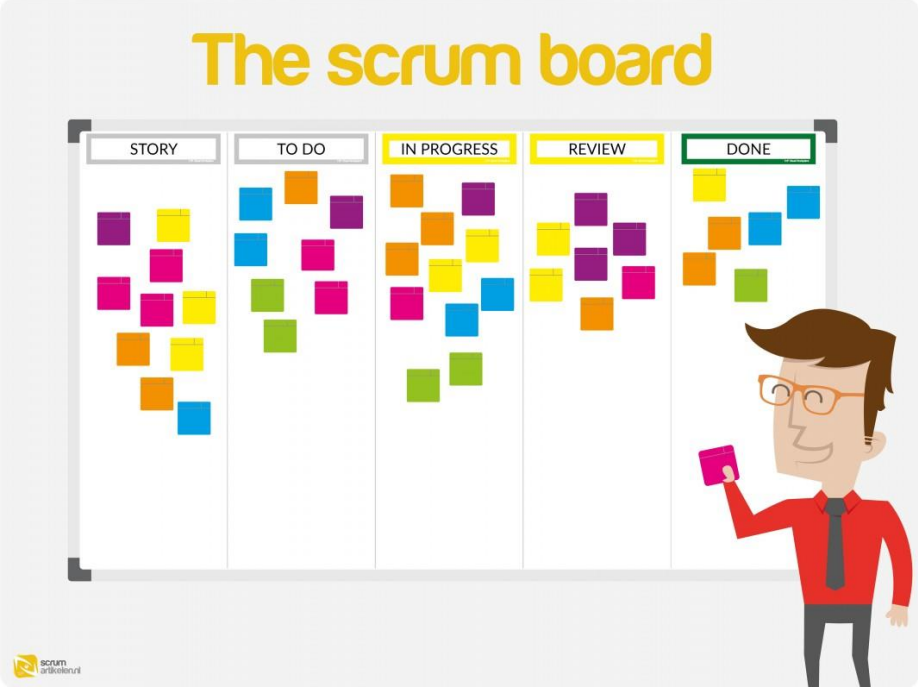
A backlog szokásosan a következő típusú elemekből épül fel:

- Feature elemek
- Hibák
- Technikai munkák
- Tudás szerzés

Ha valaki még mélyebben szeretne kategorizálni, akkor természetesen bővíthető ez a lista. Nekem ennyi már elég szokott lenni az induláshoz.

Nagyon fontos fogalom a backlogok esetében a prioritások. Minden esetben a termékgazdák priorizálják a munkát és vezetik azt ebben a dokumentumban. A csapat a termék backlogra úgy tekint, mint az igazság egyetlen forrására, ez alapján végzi a munkáját és választja ki a következő feladatot.

Sprint backlog



Kép forrása

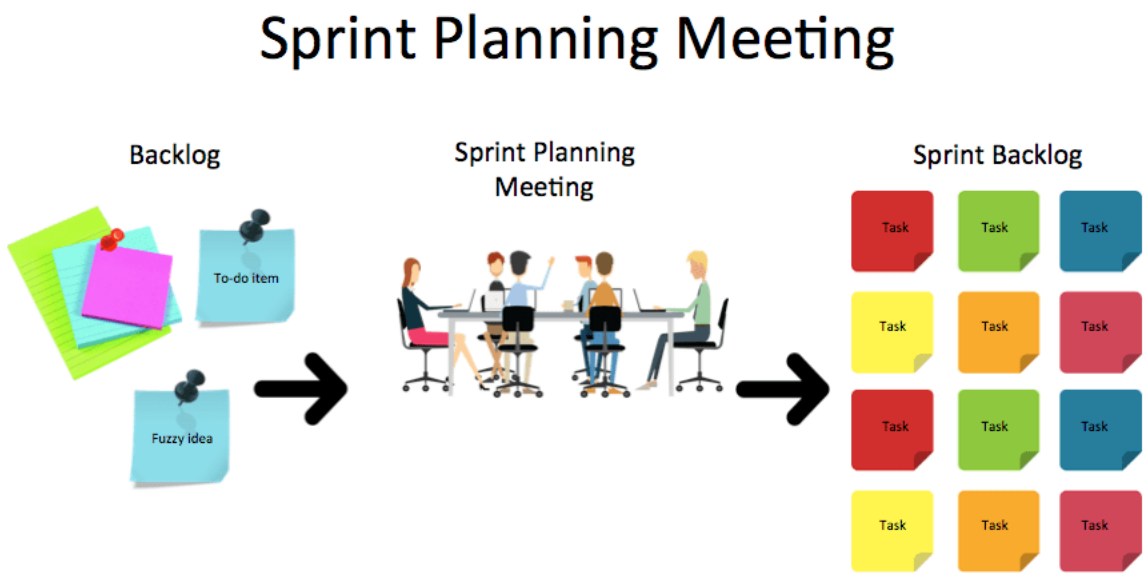
Lényegében ez a fent látható Scrum táblát jelenti, amit arra szoktunk használni, hogy vizualizáljuk az adott sprint alatt elvégzendő munkát. A sprint tervező meeting alatt a termékgazda és a csapat közösen mozgatnak át elemeket a termék backlogból a sprint backlogba. Ezen a táblán számos folyamat lépés olvasható, így például:

- Csináld meg (To do)
- Folyamatban (In progress)
- Kész (Done)

Ez a tábla egy központi eleme annak a transzparens működésnek, amit az agilis vezetés egyik előnyeként szoktam emlegetni.

Ceremóniák

Tervezés



Kép forrása

A csapat közösen megtervezi azokat a feladatokat, amiket a következő sprintben le szeretne zárni.

Kettős a ceremónia célja:

- Mindenki tisztában legyen azzal, hogy mivel halad; a többiek mivel haladnak; kivel dolgozik együtt; ha valami függősége van, akkor kivel kommunikálhat
- A csapat hétről-hétre egy irányba haladjon

A termékmenedzser minden tervezésre felkészül, gyakran a ceremónia előtt a felülvizsgálja a backlog tartalmát és a prioritásokat, annak érdekében, hogy hozzá igazítsa azt a csapat kapacitásához és az üzlet céljaihoz.

Jellemzően 30-60 percet vesz igénybe egy átlagos csapat esetében.

Napi standup

Kép forrása

Mini meeting, ahol mindenki képbe kerül a munkák haladásával kapcsolatban.

Két célja van:

- Ez a napindítója a csapatnak
- A csapattagok minden nap képet kapjanak arról, hogyan állnak a többiek

Fontos kimondani, hogy ez nem a vezetőknek szól, minden csapattag képet kell kapjon a teljes csapat haladásáról.

Mindenki körbe áll és sorban adunk egy státuszt, törekszünk arra, hogy rendkívül lényegre törőek legyünk:

- Mi történt tegnap? Mit kezdtél el? Mit fejeztél be? Mi került ki élesbe? Amit lezártál azt át kell vennie valakinek?
- Mit tervezek mára?
- Mi blokkol és miért?
- Kitől van szükségem segítségre?
- Kinek tartozok még én segítséggel?

Fontos, hogy a hétre tervezettek alapján adjunk összefoglalást a haladásról.

A hosszabb témákat innen mindenképp ki kell vinni, nem a teljes csapat előtt tárgyaljuk. Jellemzően 10-15 percet vesz igénybe egy átlagos csapat esetében.

Demó

Kép forrása

Egy közös meeting a teljes csapatra, ahol a csapat megmutatja, hogy tudott szállítani a héten.

A célja ismét a csapat szinkron:

- Egy terméken dolgozunk, egy csapat vagyunk
- Mindenki legyen tisztában azzal, hogy hol állunk közösen

Haladunk sorban, mindig ugyanabban a sorrendben, és egyenként végig megyünk a heti terveken, prezentálva azt, hogy mi készült el.

Tapasztalatom szerint két iskolája van a demóknak:

- Csak a teljes és kész, az ügyfélnek szállítható képességeket demózzuk egyben.
- Mindenki demózza minden héten, amivel Ő kész lett.

Csapata és helyzete válogatja, de általában jobban kedvelem az utóbbi megoldást. Viszont ennél gyakori megjegyzés, hogy nem tudnak a csapattagok demózni. Nos, ezzel nem értek egyet, szerintem mindig lehet demózni!

Valahogy mindenképp meg fogsz arról győződni, hogy az a valami működött, ha pedig így van, akkor pedig meg is tudod mutatni másoknak is, például:

- Mutass Postman-ban egy előkészített API hívást
- Mutass egy elkészült dokumentumot
- Mutasd a tesztek lefutását
- Mutasd meg a logokat
- Mutasd a kigenerált Excelt vagy számlát
- Mélyen technikai témáknál mutass kódot

Száz szónak is egy a vége, valami mindig prezentálható.

Jellemzően 30-40 percet vesz igénybe egy átlagos csapat esetében.

Retrospektív

Kép forrása

Legyen akármilyen jó is egy csapat, mindig van lehetőség a fejlődésre. Egy jó csapat folyamatosan keresi a fejlődési lehetőségeket, a csapatnak minden sprint végén el kell különítenie egy rövid, dedikált időszakot, hogy szándékosan átgondolja, hogyan teljesít, és lehetőségeket tárjon fel a javulásra. Erre a retrospektív találkozón kerül sor.

A retrospektív rendszerint az utolsó ceremónia a sprintben, ez a zárás. A teljes csapatnak, a scrum masternek és a termékgazdának egyaránt részt kell vennie.

Legfeljebb egy órát tarthat, ami jellemzően elegendő erre. Időről-időre sok aktuális téma merülhet fel, vagy esetleg egy csapaton belüli konfliktus kiéleződik, így a retrospektív is elhúzódhat.

Retrospektív ceremónia során a csapatnak válaszolnia kell a következő kérdésekre.

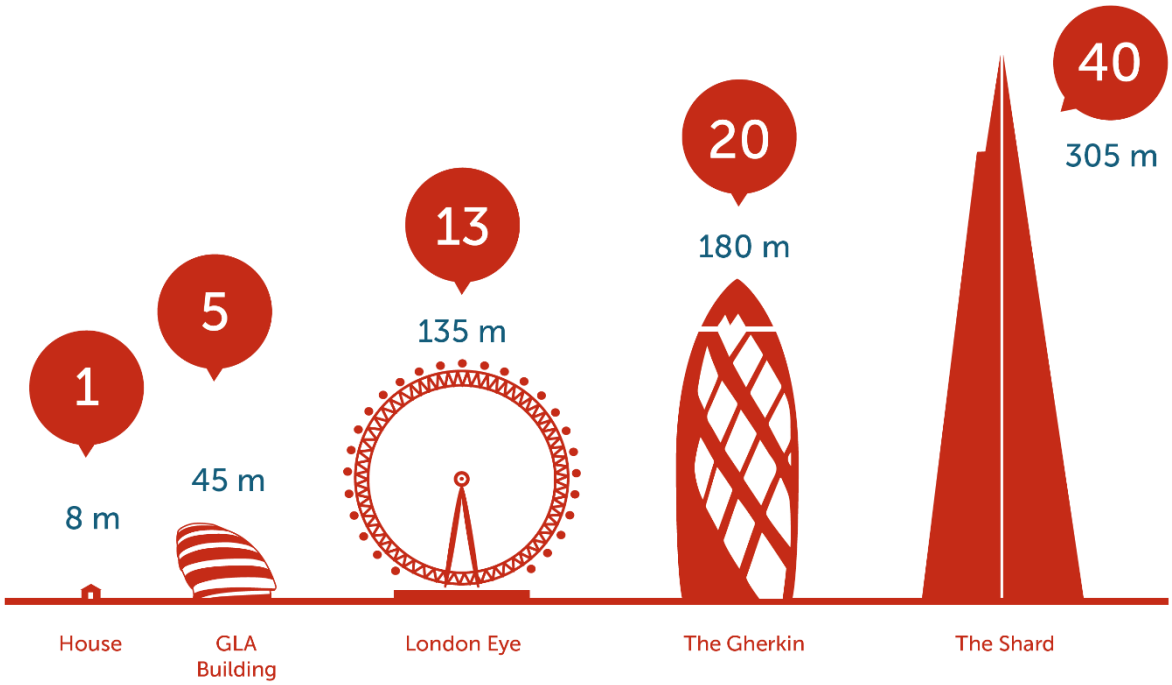
- Mit kezdjünk el csinálni?
- Mit kellene abbahagynunk?

- És mit kellene folytatnunk?

A scrum master megkönnyítheti a sprint retrospektív találkozót, ha mindenkit arra kér, hogy csak az ülés során kiabálja ki ötleteit. A scrum master sokszor körbe járkál a szobában, hogy azonosítsanak egy dolgot, amit el kell kezdeni, le kell állítani vagy folytatni érdemes.

Miután ötletekből egy lista formálódik, akkor a csapatok általában szavazni fognak, hogy mire lehetne összpontosítani.

Agilis projekt becslés



Kép forrása

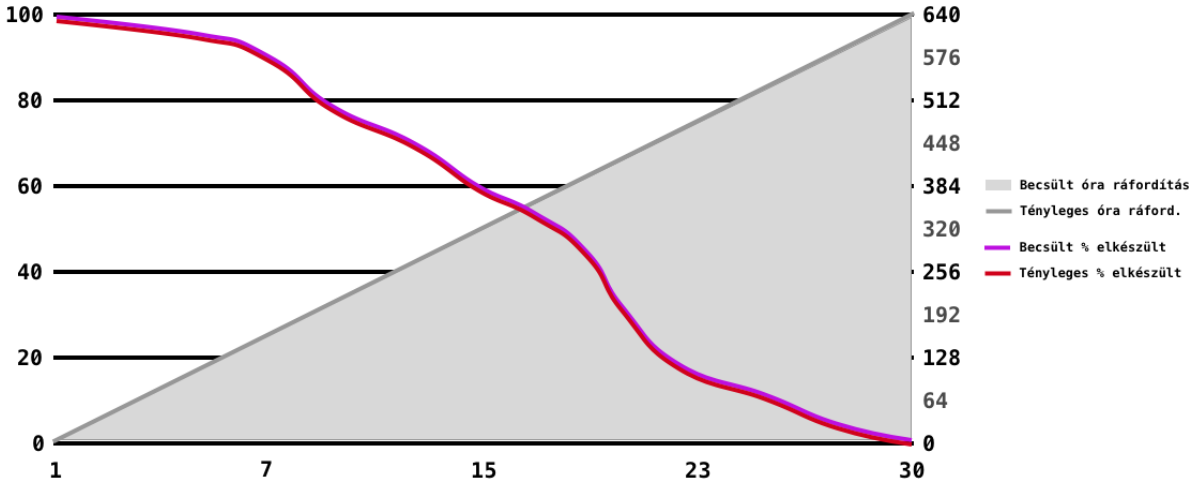
Minden fejlesztési módszertan esetében szükség van arra, hogy lásd a csapatod haladását, erre építve tudsz a jövőre nézve tervezni. Az agilis becslések segítenek megérteni az adott csapat kapacitását.

A Scrum csapatok számos módszert használnak arra, hogy megbecsüljék mennyi munka végezhető el egy adott sprint alatt. A teljesség igénye nélkül pár technika ezek közül:

- Planning poker
- Ideális órák
- Sztori pontok

Alapvetően mind egy irányba mutat, valahogy egy számérték keletkezik belőle. Ezt lehet használni a tervezéshez, valamint a retrospektívek kapcsán is előkerül, amikor egy sprint eredményeit értékeli a csapat.

Agilis jelentések



Ezek a jelentések mutatják meg hogyan alakul a csapatok teljesítménye az idők során. Ilyen például a Burndown ábra, ami megmutatja, hogy mennyi sztori pont készült el egy adott sprint során. Folyamatosan jól követhetővé válik ezáltal a csapat teljesítménye.

További metrikák a csapat haladási hatékonyságának mérésére

Szeretnélek titeket megismertetni még néhány fontos mérőszámmal, amely sokat segíthet a munkátokban.

Velocitás

A csapat átlagos áteresztőképességét méri egy sprint alatt. Érdemes több sprintet alapul venni a méréshez, az előrejelzés pontossága nagyban függ ettől, minél több az iteráció, annál pontosabb az előrejelzés.

A mértékegység egyaránt lehet óra vagy sztoripont. Megmutatja, hogy a csapat hogyan képes haladni a backloggal. Egy friss csapat esetében az első időkben a sebesség jellemzően fejlődik, de idővel be kell álljon egy kvázi állandó szintre.

Hogy a teljesítmény folyamatosan biztosítható legyen fontos a mérés. Ha a sebesség csökken, ez annak a jele, hogy a csapatnak meg kell javítania valamit.

Átfutási idő

Az átfutási idő az az időszak, amely a termék kézbesítésének igénylése és a tényleges szállítás között van. Minden folyamat lépést, amely a termék befejezéséhez szükséges

összes érdemes mérni átfutási idő szempontjából. Ez magában foglalja az üzleti követelmények kidolgozását és a hibák kijavítását is. Az átfutási idő fontos mutató, sokat segíthet a csapat részfolyamatainak optimalizálásban.

Leszállított érték

Itt a termékgazda minden követelményhez egy bizonyos értéket rendelnek. Lehet forintosítani, vagy absztrakt értéket adni.

Egyértelmű, a nagyobb értékű funkciók megvalósításának kell a legfontosabbnak lennie. A mutató emelkedő tendenciája azt mutatja, hogy a dolgok jó úton haladnak.

Azonban a csökkenő tendencia nem jó jel. Ez azt jelenti, hogy az alacsonyabb értékű szolgáltatások megvalósítása folyamatban van. Ebben az esetben a csapatnak érdemes lehet át strukturálnia a munkáját, akár egy-egy funkció fejlesztésének a megállítása is elképzelhető.

Felhasználók elé jutó hibák

Ha hibák vannak az éles rendszeren az bizonyosan sok váratlan kárt okoz. Problémákat vetnek fel, és a csapatnak foglalkoznia kell velük. Az átcsúszott hibák mutatói segítenek a hibák azonosításában, amikor kiadásra kerülnek szoftver növekmények. Ez a minőség egyik komoly mérőszáma.

Sikertelen telepítések

Szintén hasznos minőségi mutató. Segít az általános telepítések számának felmérésében. Sőt, a csapatok meghatározhatják a tesztelési és az éles környezet megbízhatóságát. Ez a mutató azt is meghatározza, hogy egy sprint során elkészült mutató kikerült-e a felhasználók elé.

Az agilitáshoz kölcsönös bizalomra van szükség

Végül még egy témát szeretnék megosztani az olvasóval.

Egy agilis szervezet akkor fog igazán működni, ha a résztvevők között magas szintű bizalom épül ki. Ehhez pedig egyenes kommunikációra van szükség, hogy nehéz beszélgetésekbe is bele tudjunk menni, annak érdekében, hogy a termék érdekét tudjuk előtérbe helyezni.

A módszertan rendszeres párbeszédet kezdeményez, így az ötletek és a kétségek folyamatosan kifejezésre kerülnek. Ez azt is jelenti, hogy a csapattagoknak folyamatosan magabiztosnak kell lenniük egymás képességében (és akaratában), hogy a döntések, amelyek ezek folyamán születnek, végrehajthatók legyenek.

A bevezetés lépései, avagy így érdemes haladni

Minden szervezet és csapat más, más az üzlet, a kontextus, az emberek, nagyon sok a változás. Nincs az áttérésre egy tuti recept, ami minden helyzetben működni fog, de alább felvázolok egy tervet, ami bizonyosan segíthet.

- Alaposan fel kell mérni a szervezet / csapat jelenlegi állapotát, látni kell, hogy honnan indulunk
    - Már ezen a ponton érdemes megfelelő metrikákat definiálni, egyrészt, hogy értsük a jelen helyzetet, másrészt, hogy a változás eredményeit jobban átláthassuk
- Ki kell tűzni célokat, hogy honnan és hova szeretnénk eljutni, a feljebb meghatározott metrikák ebben sokat segíthetnek
- Érdemes megbízni egy agilis coachot, aki segítheti az átállás folyamatát
- El kell tudni magyarázni a szervezet különböző szintjein, hogy miért váltunk agilisre
    - Milyen előnyökkel jár? Milyen problémákat vagy pazarlásokat szüntetünk meg így?
        - *Például: A hibák gyorsabb megtalálása jobb minőségű kibocsátásokat eredményez, ami alacsonyabb költségeket és technológiai adósságot jelent*
    - Mi fog változni? Milyen szokatlan új dolgokra készüljenek?
    - Kell a vezetőség jóváhagyása, anélkül nem érdemes belekezdeni, erről volt is már szó
    - Kell egy kommunikációs terv, hogy miképpen lesz ez a csapatok elé tárva
- Dolgozzon ki részletes oktatási tervet, fontos, hogy a csapatok és a stakeholderek egy nyelvet beszéljenek
- Határozzon meg egy pilot csapatot, ahol bevezetésre kerül a módszertan
- Definiálni kell és összehangolni a Scrum szerepköröket és a hozzájuk tartozó felelősségi köröket
    - Szükség lehet igazításokra HR oldalon
        - *Például: "Projektmenedzser voltam, és most scrum master vagyok."*
    - A teljesítményértékelés elvárásainak változása
- Ki kell alakítani a technikai és infrastrukturális körülményeket, hogy akár a távoli csapatok is hatékonyan tudjanak együtt dolgozni
- Folyamatosan a mérések alapján igazítsd az agilis szervezeti folyamatokat
    - *Például: A sprinthez szükséges dokumentumok száma*
    - Folyamatosan értékelj újra

A hatékonyságot pedig a helyes metrikák megválasztása és a fegyelmezett mérés fogja meghatározni. Ezekről pár fejezettel korábban már tettem ajánlásokat. A cél az, hogy mindig az adott helyzetnek megfelelő célokat rajzoljunk fel és ezekhez igazítsuk.

Hol kaphatok támogatást?

A Training 360-nál egy kiváló csapat gyűlt össze, akik a folyamat mindegyik lépésében képesek támogatást nyújtani. A különböző szerepköröknek szóló oktatásokkal, segítenek a kollégák szükséges szemléletének és alaptudásának felépítésében.

Azonban ezen túl komplex tanácsadást is kínálunk, amely a szervezet alapos felmérésével kezdődik, részletes programot dolgozunk ki a bevezetésre, amit folyamatosan támogatunk, amíg a saját belső csapat teljes mértékben képes lesz átvenni a napi munka minden aspektusát.

Részletes információkért keresse fel honlapunkat: http://www.training360.com/agilis

1117 Budapest, Budafoki út 56., South Buda Business Park, III. emelet
Felnőttképzési nyilvántartási szám: E-000398/2014