Mentor Klub

Agilis Product Owner képzés

Mi köze a szoftverfejlesztésnek a rögbihez?

Jóval az Agilis Kiáltvány 2001-es megjelenése előtt 1986-ban a Harvard Business Reviewban jelent meg Hirotaka Takeuchi és Ikujiro Nonaka "The New New Product Development Game" című írása. Ebben több japán- és amerikai vállalat termékfejlesztési gyakorlatát vizsgálták meg a fénymásolók, számítógépek és fényképezőgépek piacán. A saját területükön az összes vizsgált vállalat terméke technológiai áttörést jelentett, a piacon addig nem ismert funkciókat tartalmazott és a megjelenés után sikeresen teljesített gazdasági szempontból is.

A tanulmány a korábbi - rendszerint a vízesés modell alapján működő - fejlesztési folyamatok leírást egy váltóversenyhez hasonlította. A váltóversenyeken a váltóbotot adják át egymásnak a csapattagok ahogyan folyamatosan haladnak előre. Ezzel állították szembe az írásukban a rögbit ahol az egyes posztokon játszó specialisták közösen haladnak előre az ellenfél célterülete felé, miközben egymásnak passzolják a labdát. Az elemzésük során azt tapasztalták, hogy ezek az újító vállalatok a termékfejlesztés során olyan modelleket használtak amely sokkal inkább hasonlít a rögbicsapat működéséhez: minden specialista együtt dolgozott a sikeren és a fejlesztési ciklusok átfedték egymást nem pedig szigorúan elkülönültek egymástól.

A rögbi hasonlat tovább élt későbbi tanulmányokban is például Peter DeGrace és Leslie Hulet Stahl "Wicked Problems, Righteous Solutions" című könyvében is. Itt már Scrumként hivatkoztak rá - ami a rögbiből ismert formáció. Később ez terjedt el a szoftverfejlesztés világában és mind Ken Schwaber és Jeff Sutherland is dolgoztak már Scrum szerű rendszerben az 1990-es évek elején. Ők tovább finomították ezt a rendszert és 1996-ban közösen mutatták be az addigi tapasztalataik alapján megalkotott keretrendszert. Ezt végül 2001-ben foglalták végleges formába melyet Scrumnak neveztek el és az "Agile Software Development with SCRUM" című könyvben foglaltak össze. A Scrum keretrendszert ma legjobban a 2010-ben létrehozott Scrum Útmutatóból ismerhetjük meg, ez az a dokumentum ami lefekteti

a keretrendszer szabályait. A dokumentum folyamatosan fejlődik és kisebb-nagyobb változásokon megy keresztül. A legutóbbi nagy frissítésre 2020-ban került sor. A képzés alapját ez az útmutató fogja képezni.

A Scrum alapjai

Fontos rögtön az elején leszögezni, hogy miért nem egy módszertanról beszélünk, hanem egy keretrendszerről. A keretrendszerek meghatározzák ugyan azokat a határokat amiken belül a működésüket elképzelik, de kellő teret hagynak annak, hogy a rendszer használói saját eszközökkel és folyamatokkal töltsék ki azt. A módszertanok ezzel szemben sokkal szigorúbb elvárásokat fogalmaznak meg és abból indulnak ki, hogy az elvárások betartásával minden esetben ugyanaz lesz az eredmény a folyamat végén. A Scrumot keretrendszer mivolta miatt könnyű megérteni és elsajátítani, de nehéz magas szinten használni.

Ha elbizonytalanodunk, hogy jól használjuk-e a Scrumot mindig érdemes visszatérni az alapokhoz és megvizsgálni, hogy mire épül a keretrendszer. Vizsgáljuk most meg melyek is ezek.

A scrum az tapasztalatiságon (empirizmuson) és a lean gondolkodásmódon alapul. A tapasztalatiság alapján **a tudás alapját a tapasztalat képzi és a megfigyeléseken alapuló döntésekből ered**.

Mit is jelent ez a gyakorlatban? Vegyük példaként a Bumble randiapplikációt. Az app indulásakor a felhasználók a már meglévő Facebook fiókjukat használva tudtak regisztrálni. Miután azonban széles körben ismertté vált a Cambridge Analytica botrány ez a típusú regisztráció bizalmatlanságot keltett a felhasználókban és csökkeni kezdett az új regisztrációk száma. Miután a Bumble megfigyelte a jelenséget és feltárta a kapcsolatot a regisztrációk csökkenése és a Facebook fiókos regisztrációk iránt táplált bizalmatlanság között - azaz tapasztalati úton új tudásra tett szert - úgy döntöttek, hogy engedélyezik a felhasználók számára a regisztrációt

egyszerűsített formában, a telefonszámuk megadásával. A bevezetés után újabb növekedést tapasztaltak a regisztrációk számában azaz tapasztalati úton megerősítették a döntésüket, sikeresen alkalmazkodtak a változó környezethez.

A lean gondolkodásmód **a veszteségeket csökkenti és csak a szükségesre fókuszál**.

Egy izraeli startup azt a megoldást találta ki, hogy ne fizessen feleslegesen a felhő infrastruktúráért amikor nincs rá szükségük, hogy összekötötték a villanykapcsolót egy IoT gombbal amihez írtak egy algoritmust. Amikor az utolsó fejlesztő elhagyta az épületet a munkanap végén és lekapcsolta a villanyt az automatikusan elindította az algoritmust ami lebontotta a fejlesztéshez használt infrastruktúrát, így egészen a másnapi kezdésig ezért nem kellett fizetnie a cégnek.

A Scrum egy **ismétlődő (iteratív), inkrementális megközelítést alkalmaz a kiszámíthatóság optimalizálása és a kockázat kézben tartása érdekében**.

A változó piaci igények és a technológia folyamatos fejlődése olyan környezetet teremt ami minél távolabbra próbálunk előre tekinteni annál nagyobb bizonytalansággal tudjuk megmondani mi lesz a termékeinkkel kapcsolatos döntéseink kimenetele. A Scrum úgy oldja meg ezt a kérdést, hogy rövid, rögzített idejű legfeljebb egy hónapos ismétlődő eseményekbe - Sprintekbe - zárja be a fejlesztést, melyek folyamatosan ismétlődnek. Az előző Sprint végével automatikusan elindul a következő. Minden Sprintnek az a célja, hogy az ötletekből érték - működő szoftver - álljon elő. Azzal, hogy az időtávot rögzítjük csökkentjük a lehetséges kimenetelek számát ezzel kiszámíthatóbbá és kockázatmentesebbé tehetjük a termékünk piaci teljesítményét. Természetesen nem tudjuk teljesen kiszámíthatóvá és kockázat mentessé tenni a folyamatokat, de kezelhető szintre tudjuk csökkenteni a lehetőségeket és a kockázatot.

A Scrum tartóoszlopai

A Sprintek négy formális eseményt foglalnak magukba - ezeket később részletezzük. Minden a sprinten belüli esemény a megfigyelés (Inspection) és az alkalmazkodás (Adaptation) technikáját alkalmazzák. Ezek az események azért képesek működni, mert a gyakorlatban alkalmazzák a Scrum empirikus alappilléreit, azaz az **átláthatóságot, a megfigyelést és az alkalmazkodást**.

Átláthatóság

Az átláthatóság kiemelt szerepet játszik a szervezeten belül a Scrumban. Nem csak azok számára kell átláthatóvá tenni a folyamatokat akik a termék létrehozásán és továbbfejlesztésén dolgoznak, hanem azok számára is akik bármilyen szinten érdekeltek a termék létrejöttében és piaci teljesítményében.

Vegyünk példának egy olyan helyzetet amikor az egyik fejlesztő nehezen halad az egyik megoldandó problémával. Az általa ismert megoldási módok vagy nem működnek vagy túl időigényesek. Ha nem ad hangot annak, hogy elakadt akkor az veszélybe sodorhatja, hogy működő szoftver jöjjön létre a Sprint végén. Ha viszont nyíltan felvállalja a többi fejlesztő előtt, hogy nem tudja időben megoldani a problémát akkor segítséget kaphat a többiektől ami növeli a csapat összetartását és segíti a csapatban meglévő tudás megosztását is. Ha visszatekintünk az Agilis Kiáltvány hatodik alapelvére pontosan a személyes beszélgetés fontosságát láthatjuk kiemelve.

Nézzünk egy másik példát is: a Sprint véget ért, de az előzetes tervekhez képest nem sikerült minden funkciót leszállítani. A csapat ezt megpróbálja jobb színben feltüntetni vagy ködösíteni az információkat. A marketing részleg ezt úgy értelmezi, hogy minden a korábbi terveknek megfelelően halad ezért elindítják a termék marketing kampányát. A beharangozott időpontban azonban csak egy félkész

termékkel tudnának megjelenni a piacon a korábbi csúszás miatt és innentől nincs jó megoldása a kialakult helyzetnek: vagy egy rossz minőségű termék kerül a piacra a határidő lejártával vagy a cég megbízhatósága szenved csorbát a felhasználók szemében. A videójáték iparból ismerős lehet a “0-day patch” fogalma. Ez a megjelenés napján fennálló hibákat próbálja orvosolni és több mint valószínű, hogy abból ered, hogy nem volt átlátható a kommunikáció a szervezeten belül.

Megfigyelés

A Scrumban több esemény is arra hivatott, hogy megfigyelje a termék és a termék létrejöttéhez használt folyamatok aktuális állapotát. A megfigyelés mindig az alkalmazkodást segíti elő - önállóan nincs értelme. Minden megfigyelés célja a Scrumban az, hogy valamilyen pozitív változást idézzen elő.

Nézzünk egy példát arra amikor a szervezet egy termékkel kapcsolatos tudással lesz gazdagabb a megfigyelés során. Egy telekommunikációs vállalat elindítja az új weboldalát ahol a felhasználók különböző csomagokat választhatnak ki. Az indulás után azt tapasztalják, hogy az előző év ugyanilyen időszakához képest csökkent az oldal látogatottsága és a különböző csatornákon beérkező panaszok száma is megnövekedett. Valószínűleg összefüggés lehet az új oldal és az események között. Alaposabb vizsgálat szükséges ahhoz, hogy a vállalat kiderítse az okokat. Dönthetnek úgy, hogy ügyfél interjúkat készítenek és megpróbálják kideríteni mi lehet a trend oka. Vagy készíthetnek egy könnyen emészthető videót az új oldal használatáról. Esetleg bővíthetik az oldalt hiányzó funkciókkal. Bármilyen irányba is mozdulnak el mindig azon kell, hogy alapuljon a döntés amit a tapasztalati tények mutatnak. Ha kellően jól használja a szervezet ezt a pillért akkor ezekre még az előtt fény derülhet, hogy a tényleges termék a piacra kerülne.

Egy másik példában vizsgáljuk meg azt amikor nem a termékkel, hanem a folyamatokkal kapcsolatban kap új információt a szervezet. Az egyik Sprint után

szokatlanul megnövekszik a hibák száma a termékben. Ugyan a működést nem veszélyeztetik a hibák, de a javításuk időigényes és hátráltaja az értéket termelő funkciók létrehozását. Elképzelhető, hogy megváltozott a csapat összetétele és egy kevés tapasztalattal rendelkező munkatárs vét az átlagnál több hibát. Vagy a csapat kipróbált egy új technológiát és az ebben való jártasság hiánya okozza a problémákat. Előfordulhat az is, hogy külső nyomásnak engedve a csapat alább adott a minőségi elvárásaiból. A valós okok feltárásával úgy alakíthatják saját működésüket, hogy az kevesebb hibával járjon és hatékonyabban haladhassanak a fejlesztéssel. A jól alkalmazott Scrum egyik nagy előnye, hogy hamar a felszínre tudja hozni a szervezeti problémákat és ha ezek megismerésére és megoldására nyitott a szervezet akkor folyamatos versenyelőnyt tudnak belőle kovácsolni.

Alkalmazkodás

Ahogyan korábban fogalmaztunk a megfigyelés amelyet nem követnek tettek haszontalan a Scrumban. A termékfejlesztésben dolgozó szakemberek számára kellő felhatalmazást és önrendelkezést kell biztosítani annak érdekében, hogy a tapasztalataik alapján korrigálni tudják a terméket vagy a termék fejlesztéséhez szükséges folyamatokat abban a pillanatban ahogy azok az elfogadható határokon kívülre kerülnek.

Nézzük meg újra példákon keresztül, hogyan tud alkalmazkodni egy szervezet a megfigyeléskor gyűjtött tapasztalatai alapján.

Talán az egyik legsikeresebb történet ilyen szempontból az Amazonhoz köthető. Az alapvetően online értékesítéssel foglalkozó vállalat a saját belső infrastruktúrájuk megújítása során olyan megoldást dolgozott ki amellyel rendkívül gyorsan és megbízhatóan tudták azt skálázni az igényeknek megfelelően. A skálázhatósági lehetőség miatt fölös kapacitást is létrehoztak. Ebből jött az ötlet, hogy ezt szolgáltatásként is tudnák értékesíteni a piacon. Rövid időn belül nem csak egy új

terméket hoztak létre, hanem alapjaiban változtatták meg a webes fejlesztések működését - rohamosan fejlődésnek indultak a felhő megoldások.

Egy másik példában nézzük meg azt amikor a folyamatokon módosít egy fejlesztő csapat a hatékonyabb működés érdekében. Példánkban a csapat azt tapasztalta, hogy a termék funkciónak bővülésével drasztikusan megnövekedett a tesztelésre fordított idő, amit eddig leginkább manuálisan végeztek. Miután ezt tapasztalták azt a döntést hozták, hogy lépésről-lépésre időt szánnak arra a Sprintekben, hogy egyes teszteket automatizáljanak és ezt a gyakorlatot beépítik az új funkciók létrehozásába is. Ezen változás hatására a későbbi Sprintekben több időt tudtak arra fordítani, hogy jobb és gyorsabb termékkel lépjenek a piacra.

Scrum szerepkörök

> » A Scrum alapvető egysége a néhány emberből alkotott csapat, azaz a Scrum Csapat (Scrum Team). A Scrum Csapat egy Scrum Masterből, egy Product Ownerből és Fejlesztőkből (Developerekből) áll. A Scrum Csapaton belül nincsenek alcsapatok és nincs hierarchia. Egy szorosan összetartozó, szakértőkből álló egység, akik minden időben egyetlen célra, a Product Goalra fókuszálnak. «

A Scrum Csapat egészére jellemző néhány fontos tulajdonság.

Keresztfunkcionalitás

A Scrum Csapat összetétele olyan, hogy rendelkezik az összes olyan készséggel és képességgel amely szükséges ahhoz, hogy egy-egy Sprintben értéket állítsanak elő. Ez magában foglalja azt a Product Ownert aki képes megalkotni egy víziót a termékkel kapcsolatban és meghozni a megfelelő döntéseket a piaci viszonyok alakulásának kapcsán. Azt a Scrum Mastert aki felhatalmazás nélkül szolgáló

vezetőként segíti a csapat működését. És magában foglalja azokat a fejlesztőket akik megtervezik a termék architektúráját, létrehozzák és üzemeltetik a termék működéséhez szükséges infrastruktúrát és adatbázisokat, megtervezik és kivitelezik a felhasználói felületeket, tesztelik a működést és garantálják, hogy a termék folyamatosan a felhasználók rendelkezésére álljon.

Önmenedzselés

A Scrum Csapat tagjai kellő felhatalmazást kapnak arra, hogy a csapaton belül hozzanak döntést arról ki és milyen munkát végez el, mikor és hogyan. Ez a gyakorlatban azt jelenti, hogy a Scrum Csapat szabad kezet kap például abban, hogy milyen programnyelvet használnak a termék létrehozásához, hogyan osztják el egymás között a feladatokat, milyen sorrendben végzik el a működő szoftver létrehozásához szükséges teendőket, vagy hogy ki látja el és mennyi ideig a vezetői feladatokat.

A Scrum Csapat mérete

> » A Scrum Team megfelelően kicsi, hogy gyors maradjon és elég nagy, hogy jelentős mennyiségű munkával készüljön el egy Sprinten belül. **Tipikusan 10 vagy kevesebb emberből áll.** «

Miért pont ekkora a Scrum Csapat mérete? Az emberiségnek több tízezer éves tapasztalata van abban, hogy kis - család méretű - csoportokban oldjunk meg problémákat a mamut vadászattól egészen a Revolut első verziójának megalkotásáig. A tíz fő alatti csapatméret lehetővé teszi, hogy egyszerűen és hatékonyan kommunikáljanak egymással a csapat tagjai, megosszák egymással az információkat és a tudást és könnyen döntést tudjanak hozni az együttműködésük és céljaik kapcsán.

Ha a Scrum Csapat létszáma túllépi a 10 főt akkor érdemes megfontolni, hogy több, összefüggő és egymással szorosan együtt dolgozó csapattá váljanak szét. A szétválás után továbbra is minden csapat ugyanazon a terméken fog dolgozni és ugyanúgy önállóan is értéket állítanak elő.

A Scrum Csapat egészének vizsgálata után nézzük meg, hogy az egyes felelősségi körök pontosan mit takarnak és mitől lesznek az adott felelősségi körben dolgozó szakemberek csúcsteljesítők.

A kiváló Scrum Master kvalitásai

A Scrum Útmutató alapján a Scrum Master felelősségi köre:

> » A Scrum Master felelős a Scrum meghonosításáért a Scrum útmutatóban leírtaknak megfelelően. Ezt úgy éri el, hogy minden szereplőnek segít megérteni a Scrum elméletét és gyakorlatát, mind a Scrum Teamben, mind a szervezetben.
>
> A Scrum Master felelős a Scrum Team eredményességéért. Ezt úgy teszi meg, hogy lehetővé teszi a Scrum csapat számára, hogy saját munkamódszereit javítsa, a Scrum keretein belül maradva.
>
> A Scrum Masterek igazi vezetők, akik a Scrum csapatot és a nagyobb szervezetet szolgálják. «

A Scrum Master felelősségi köre három területre terjed ki. A Scrum Master szolgáló vezetőként támogatja:

- A Scrum Csapatot
- A Product Ownert
- A szervezetet

» A Scrum Master többféleképpen támogatja a Scrum Csapatot, többek között:

- A Scrum Master coacholja a csapattagokat az önmenedzsment és a keresztfunkcionalitás tekintetében;
- Segít a Scrum Csapatnak fókuszálni a Definition of Done-nak megfelelő, nagy értékű Inkrementumok létrehozásában;
- Elősegíti a Scrum Team haladását gátló tényezők megszüntetését; és
- Biztosítja, hogy az összes Scrum esemény megvalósuljon, pozitív és eredményes legyen és ne lépje túl az időkeretet.

A Scrum Master többféle módon szolgálja a Product Ownert, többek között:

- Segít megfelelő technikákat találni a hatékony Product Goal meghatározáshoz és Product Backlog kezeléshez
- Segít a Scrum Teamnek a világos és tömör Product Backlog elemek szükségességének megértésében
- Segít az empirikus terméktervezés meghonosításában a komplex környezetben; és
- Igény vagy szükség szerint facilitálja az együttműködést az érdekelt felekkel.

A Scrum Master többféle módon támogatja a szervezetet, többek között:

- A szervezetet vezeti, képezi és coacholja a Scrum meghonosításában
- Megtervezi a Scrum bevezetését, tanácsot ad a témában a szervezeten belül
- Segíti az alkalmazottakat és az érdekelt feleket a komplex munka empirikus megközelítésének megértésében és megvalósításában; és
- Eltávolítja az akadályokat az érdekelt felek és a Scrum Teamek között. «

A Scrum útmutató ugyan elég pontosan bemutatja azt, hogy milyen felelősségi körei vannak egy Scrum Masternek, de kevésbé segít abban, hogy hogyan lehet egy szakember kiváló ebben a szerepben. Nézzük meg részleteiben mi egy

csúcsteljesítő Scrum Master hozzáadott értéke a szervezetben és hogyan tudja a felelősségi körének megfelelő feladatokat magas szinten ellátni.

A fejlődési lehetőségek azonosítása

A Scrum Master feladata, hogy mind a Scrum csapatot, mind a szervezetet fejlessze és segítse, hogy minél inkább elsajátítsák az agilis gondolkodást és működést. Ahhoz, hogy ezt a célt el tudja érni azonban folyamatos megfigyelésre van szüksége, hogy a megfelelő területekre tudjon fókuszálni.

A fejlődési lehetőségek azonosításához a Scrum Masternek meg kell figyelnie a csapatdinamikát anélkül, hogy közben beavatkozna a működésbe. Eközben fel tudja mérni, hol vannak azok a pontok, ahol a legszükségesebb lesz a változás. Ezt a feladatot el tudja végezni úgy is, hogy részt vesz a csapat működésében külső szemlélőként és úgy is, hogy a csapat tagjaival egyesével beszélget arról, hogy mit látnak a saját működésükkel kapcsolatban.

Nem csak annak a csapatnak a működését érdemes figyelemmel követnie, amelyikkel szorosan együtt dolgozik, hanem a szervezetben működő más csapatokét is. Az ott felmerülő problémák és megoldások akkor is hasznos információt rejthetnek, ha a vizsgált időpontban a saját csapatánál esetleg más események jelentenek kihívást.

A termék megalkotásában jellemzően nem csak a Scrum Csapat hanem más szervezeti egységek is részt vesznek. Érdemes tehát a Scrum Masternek modellezni és vizuálisan megjeleníteni azt a folyamatot ahogyan ez működik a szervezeten belül.

A termékfejlesztés folyamata során számos eszközt és gyakorlatot használ a Scrum Csapat. A Scrum Master feladata, hogy felmérje ezek hatékonyságát és azonosítsa az esetleges hiányosságokat. Hogyan tudja a csapat garantálni a folyamatos

integrációt és szállítást? Megfelelő szintű a tesztautomatizáció? Mi garantálja a termék folyamatosan magas minőségét? Ellenőrzik a csapattagok egymás munkáját? Kellő mértékben dolgoznak együtt a problémák megoldásán?

Naprakész agilis tudás

Az agilis működés motorja a szervezetben a Scrum Master. Ez nem csak a mindennapos üzemeltetés garantálását jelenti, hanem azt is, hogy a kiváló Scrum Master folyamatosan keresi azokat az új utakat, trendeket amelyek mentén még olajozottabbá teheti az agilis működést a szervezetben.

A felelősségi kör neve ugyan Scrum Master, de ez nem jelenti azt, hogy csak a Scrum keretrendszeren belül kereshet új megoldásokat. Ha az agilis alappillérek mentén de más keretrendszerből vagy folyamatból emel be új elemet a Scrum Master az ugyanúgy a hatékony működést segítheti elő.

Még a legjobb Scrum Masterekkel is előfordul, hogy egy vagy több facilitálási módszert túlhasználnak és ebbe belefárad a csapat. Ezt időben felismerve és az eszköztárat folyamatosan bővítve elébe mehet a Scrum Master annak, hogy fontos információ vesszen el működés közben.

A csapat és a szervezet coacholása

Miután kellő mennyiségű információ áll rendelkezésre a szervezeten belülről és megfelelő tudás gyűlt össze a szervezeten kívülről ideje, hogy a Scrum Master a gyakorlatba is átültesse ezeket a hatékony működés érdekében.

Ahhoz hogy ezt elérje érdemes rövid-, közép- és hosszú távú terveket kell készítenie azokra a területekre vonatkozóan ahol változást szeretne látni. A terveknek pontos célokat és határidőket is kell tartalmaznia. Ez kifejezetten fontos akkor, ha a szervezetben még új az agilis működés.

Ki kell használnia a Scrum egyik legnagyobb előnyét: a hatékony működést akadályozó tényezők gyors felszínre kerülését. Nyíltan beszélni a felmerülő problémákról és lehetőségekről annak érdekében, hogy azt közösen oldja meg a csapat vagy a szervezet. Tehet javaslatokat a megoldásra, de az elsődleges feladata az, hogy elősegítse a szervezet fejlődését és a csapat önszerveződő képességét azzal, hogy facilitálja a megoldáson dolgozók találkozóit.

Előfordulhat, hogy a Scrum Csapaton kívüli menedzserek szeretnének kívülről beavatkozni a folyamatokba. A Scrum Master feladata megértetni a menedzsmenttel, hogy milyen negatív hatásokkal járhat ez. Ha ez sikeres akkor segítheti a menedzsment munkáját azzal, hogy olyan menedzsment gyakorlatok bevezetését segíti elő amelyek kompatibilisek az agilis működéssel.

Az agilis működés bevezetése és üzemeltetése nem egyszerű feladat és gyakran szembe megy a szervezet rövid távú céljaival. A Scrum Master feladata, hogy ezen a viharos időszakon sikeresen átvezesse a szervezet hajóját.

Figyelnie kell minden lehetőségre - legyen az akár egy kötetlen beszélgetés egy kávé mellett - ahol információt gyűjthet és tanácsot adhat a helyes és hatékony agilis működéssel kapcsolatban.

A formális Scrum események facilitálása

A Scrum Útmutató szerint a Scrum Master biztosítja, hogy a Scrum események megvalósuljanak, hatékonyak legyenek és beleférjenek az időkeretbe. A Scrum Master szerepe változhat annak függvényében, hogy a csapat önszerveződési képessége milyen szinten van. Egy kezdő, az agilitással ismerkedő csapat esetén az összes eseménynél végig kell vezetnie a résztvevőket az esemény egészén. Ha szükséges akkor a megfelelő kérdésekkel tovább kell lendítenie az eseményt. Egy érettebb, magabiztosabb csapat esetén nyugodtan hagyhat nagyobb teret a

csapattagok között kibontakozó kommunikációnak és önálló problémamegoldásnak. Végső soron ez utóbbi állapot felé kell irányítania a csapatot.

A Scrum egyik nagy előnye, hogy az előre definiált eseményeken túl a fennmaradó időt a csapat zavartalanul tudja arra fordítani, hogy a lehető legnagyobb értékkel bíró inkrementumot állítsák elő. Előfordulhat, hogy szükségessé válik egy a megszokott eseményektől eltérő találkozó összehívása, azonban a Scrum Master felelőssége, hogy ennek előre meghatározott célja és elvárt kimenetele legyen.

Főleg a kezdeti időszakban érdemes minden egyes alkalommal tisztázni a Scrum események célját és a rájuk szánható maximális időkeretet.

Az események során a Scrum Master tartja kézben az időgazdálkodást. Különösen akkor van kiemelt szerepe, ha több területről kell egyszerre egyeztetnie a csapatnak az esemény során. Ügyelnie kell rá, hogy az egyes témákra kellő mennyiségű időt szánjanak és, hogy ne lépjék túl az előre megbeszélt időkeretet.

Ha úgy érzi, hogy egy téma megbeszélése során letért a csapat arról az útról ami az előrelépés felé vezethet akkor érdemes feltenni a kérdést: “A most tárgyaltak relevánsak az esemény céljának elérése szempontjából?”.

Az események levezetése során figyelmesen kell hallgatnia az elhangzottakat és megfelelő kérdéseket kell feltennie annak érdekében, hogy a csapat elérje az esemény célját. A Scrum Master nem feltétlenül kell, hogy járatos legyen a beszélgetések technikai részleteiben, de jól át kell, hogy lássa a nagy képet. Egy kiváló Scrum Master minden eseményre csak a kézzel írott jegyzetekhez szükséges eszközöket viszi magával a tudásán kívül.

Kapcsolatépítés a csapattagokkal

A Scrum működéséhez elengedhetetlen, hogy áttekinthető legyen a működés, a csapattagok bízzanak egymásban és tiszteljék a másik munkáját. Ennek kiépítése szintén a Scrum Master feladatainak egyike.

A kiváló Scrum Master nem csak a munkával kapcsolatos területeken ismeri a csapattagok működését, de gyakran beszélget velük a fejlesztéshez szorosan nem kapcsolódó témákról is. Ezt megteheti a személyes beszélgetések során is vagy akár egy kávé mellett. Az így kapott információ segítheti a munkáját például olyan helyzetekben amikor személyes konfliktus alakul ki a csapattagok között.

Feladata a csapatkohézió növelése. Ehhez szervezhet csapatépítő gyakorlatokat, amely lehet egy közös vacsora vagy egy szakmai műhelymunka, függetlenül attól, hogy a szervezet finanszírozza-e azt vagy sem.

A Scrum események megszervezése

Habár néha jelentkezhet arra igény a csapat részéről, hogy titkári feladatokat is ellásson a Scrum Master leszögezhetjük, hogy ez nem a feladata. Segít a csapatnak az események létrehozásában, de végső soron az a célja, hogy a csapat az önszerveződés jegyében saját maga feleljen a működéséért.

A csapattal közösen kialakítja az események helyét és idejét és segít abban - a lehetőleg ritka - esetben, ha ettől eltérő időpontban kell megtartani az eseményt. A komplexitás csökkentésére javasolt a Scrum eseményeket ugyanabban az időpontban és ugyanazon a helyen tartani minden egyes Sprintben.

Abban az esetben, ha a Fejlesztők külön eseményt igényelnek, hogy a termékkel kapcsolatos kérdéseket tisztázzák a Product Ownerrel bátorítja a feleket ennek megszervezésére és kérés esetén segít a hatékony levezetésben.

Akadályok elmozdítása a haladás útjából

Nem minden akadály elhárítása a Scrum Master feladata, de minden akadály azonosítása az!

Aktív hallgatóként a kiváló Scrum Master hamar felismeri, ha esetleg akadály került a csapat haladásának útjába. Az olyan mondatok mint az “Ebben nem egészen vagyok biztos…” vagy az “Ezen még dolgozom…” esetleg az “Ez nem világos…” mind azt sugallják, hogy a csapat akadályba ütközött. A Scrum Master feladata ilyenkor az, hogy a megfelelő kérdésekkel azonosítsa a probléma gyökerét, majd segítsen a csapatnak abban, hogy megoldást találjanak rá.

Ha a probléma bármilyen oknál fogva a csapat befolyásán kívül esik, akkor a Scrum Master feladata, hogy azonosítsa a szervezeten belül azt az erőforrást aki vagy ami megoldást jelenthet.

Az így feltárt akadályokat a Scrum Master folyamatosan nyomon követi és mindenki számára áttekinthetővé teszi. Ha változás történt egy akadállyal kapcsolatban, akár azért mert elhárul akár azért mert új információra van szükség a megoldásához arról a Scrum Master visszajelzést ad a csapat vagy a szervezet részére.

Képzési anyagok készítése

Amikor a Scrum Master elkezdi beazonosítani a fejlődési lehetőségeket a csapaton és a szervezeten belül valószínűleg bele fog futni abba a helyzetbe, hogy egy adott területen nem megfelelő a csapattagok képzettsége. Ez lehet az agilitás területén feltárt hiányosság vagy akár olyan specifikus terület szaktudásának a hiánya mint a tesztautomatizálás.

Ilyen helyzetben a Scrum Master feladata, hogy összegyűjtse a területről a hasznos és megbízható információkat. Ezek alapján összeállítsa a képzési anyagokat,

műhelymunka feladatokat vagy teszteket. Megszervezze vagy akár levezesse a képzést a csapattagok számára. A képzések végeztével pedig összegyűjtse a visszajelzéseket és maga is releváns kérdéseket tegyen fel, hogy megbizonyosodjon a képzés hatékonyságáról.

Megfelelő mérési rendszer kialakítása a csapat számára

A Scrum alapja a tapasztalatiság, azaz a tudás megszerzésének alapvető módja a tapasztalat. A döntéseink alapja pedig az amit az adott időpillanatban ismerünk. A Scrum Master az idő előrehaladtával egyre jobban megismeri azt a csapatot és azt szervezetet amellyel együtt dolgozik, de érdemes a működést mérhetővé is tenni, hogy hatékonyabban beazonosíthatóak legyenek azok a területek ahol a leginkább szükség van a fejlődésre.

A Scrum Master feladata annak felmérése, hogy egy adott helyzet megoldásához milyen mérés a legcélravezetőbb. Ez alapulhat azon, hogy ismeri a helyzet hátterét és tudja melyik mérőszám fogja a legtöbb információt adni. Vagy a kezdetekben használhat több fajta mérőszámot is és ezekből a tapasztalatok alapján a hasznosakat tartja csak meg.

Ha később csatlakozott egy csapathoz nyugodtan végezhet méréseket az az előtti időszakra vonatkozóan is, ha megfelelően képes azokat értelmezni és elő tudja ezzel segíteni a hatékonyabb működést.

Ha egy mérést kellően sokszor kell elvégezni akkor megtalálja annak a módját, hogyan lehet ezt automatizálni. Ideális esetben Sprintenként nem vesz 30 percnél több időt igénybe a mérések eredményének előállítása.

A mérések eredményeit átláthatóvá teszi a szervezet számára és elmagyarázza az egyes mérőszámok jelentését. Szükség esetén felülvizsgálja a mérések hatékonyságát a csapattal együtt és módosítja a mérések összetételét.

Kiemelten fontos, hogy a menedzsment számára egyértelmű legyen az, hogyan NE használják a mérések eredményeit. Az agilis mérések eredményeinek felhasználása például a csapat vagy egyének teljesítményének meghatározására kifejezetten ellenjavallt. Ez nagyon könnyen a bizalom mint alapérték romlásához vezethet.

Az agilis - főleg a termék teljesítményével kapcsolatos - metrikákról egy későbbi alkalommal részletesen fogunk foglalkozni.

Az ideális Fejlesztők adottságai

> » A Fejlesztők azon emberek a Scrum Teamben, akik elkötelezték magukat, hogy minden Sprintben előállítják egy használható Inkrementum valamilyen aspektusát. «

Bárki aki egy Scrum Csapatban azon dolgozik, hogy egy adott Sprint végén a felhasználók számára értéket jelentő és a csapat által meghatározott minőségi kritériumoknak megfelelő funkcionalitás jöjjön létre Fejlesztő. Tévedés, hogy csak azt tekintjük Fejlesztőnek aki valamilyen kódot írt ami működteti a funkciót. Mivel a Fejlesztő csapatnak rendelkeznie kell az összes olyan tudással amely lehetővé teszi az értékteremtést, ezért egy a felhasználói élmény tervezésében jártas dizájner, egy minőségbiztosítási területen tapasztalt tesztelő vagy az alkalmazás működési hátterét biztosító infrastruktúráért felelős rendszergazda ugyanúgy Fejlesztő a csapatban.

Korábban részleteztük, hogy a Fejlesztők között nincs hierarchia, de képesnek kell lenniük a mindennapi tevékenységeik hatékony megszervezésére, szükség esetén a Scrum Master segítségét igénybe véve. Ez a típusú működés nem a szervezeti hierarchia ellen irányul, hanem a problémák megoldásához legközelebb dolgozók kezébe ad kellő felhatalmazást a munkájuk hatékony elvégzéséhez.

> » A Fejlesztők felelősségi köre mindig kiterjed a következőkre:

- A Sprint tervének, azaz a Sprint Backlognak az elkészítése
- A minőség beépítése a Definiton of Done-ban foglaltak betartásával
- A terv napi szintű korrigálása a Sprint Goal elérésének érdekében; valamint
- Egymás elszámoltatása szakmai szempontból. «

A Fejlesztők felelősségi köreinek leírásában több olyan fogalom is található amivel eddig nem foglalkoztunk. A későbbiekben természetesen ezekre is fogunk időt szánni. Jelen pillanatban összefoglalóan annyit érdemes a Fejlesztőkről feljegyezni, hogy egy 10 főnél kisebb, önszerveződő, az összes szaktudás birtokában lévő csapat akik magas minőségi követelmények mellett gyakran és rendszeresen állítanak elő a felhasználók számára értékes funkcionalitást.

Az elsőrangú Product Owner készségei

> » A Product Owner felelős a Scrum Csapat munkájának eredményeként előállt termék értékének maximalizálásáért. Ennek megvalósítása a szervezeti formától, a Scrum Csapatoktól és egyénektől függően eltérő lehet. «

A Product Owner az a személy akinek egy jól körülhatárolható elképzelése van a termék víziójáról. Nem ismeri a jövőbeni termék működésének minden részletét, nem tudja pontosan hogyan fog működni a termék, de nagyon konkrét elképzelése van arról, hogy **miért** akarja a terméket fejleszteni és hogy **milyen** problémákat fog megoldani a termék és **kinek** szánjuk.

A fejlesztési folyamat fontos szereplői a stakeholderek. Ők azok akik használni, támogatni fogják a terméket vagy bármilyen formában hatással lesz az életükre vagy a munkájukra.

A stakeholderek igényei user storyk formájában jelennek meg a csapat asztalán. Például egy online áruház esetében a fizetés lehet egy ilyen story. A storyk leggyakrabban a termék egy funkcióját vagy a felhasználó egy lehetséges útját írják le. A stakeholdereknek rengeteg ötlete van, a Product Owner feladata, hogy ezeket egymástól független user storykká alakítsa át.

Valakinek meg is kell építenie a vízióban elképzelt terméket, akik pedig nem mások mint a Fejlesztők. A Fejlesztők kis méretű, keresztfunkcionális és önszerveződő csapatban, dolgoznak együtt - ideális esetben fizikailag is egy helyen.

Mivel agilis működésről beszélünk ezért a Fejlesztők nem egy nagy “Big Bang” indulásban gondolkodnak a termékkel kapcsolatban, hanem minél hamarabb szeretnék a felhasználókhoz eljuttatni a terméket és minél gyakrabban szeretnének új funkciókat hozzáadni. Átlagosan 4-6 user storyt tudnak fejleszteni egy Sprintben. Vannak nagyobb sztorik amik kétszer akkorák és vannak kisebbek amik csak fele akkorák, de az átlag 4-6 körül mozog.

Annak érdekében, hogy a csapat ezt a mennyiséget folyamatosan tudja hozni komoly hangsúlyt helyeznek az automatikus tesztelésre és a folyamatos integrációra.

A probléma abban rejtőzik, hogy a stakeholdereknek rengeteg ötlete és javaslata van. Egy-egy Sprintben jóval több mint 4-6. Sőt minél több dolgot látnak működés közben, annál nagyobb ihletet kapnak újabb és újabb igények megfogalmazására.

Mi történik, ha ezeket mind egyszerre akarják a Fejlesztők megvalósítani. A csapatot maga alá fogja temetni a munka. Tegyük fel, hogy egy Sprintben 10 feladatot vállal magára a csapat, de csak 4-6-ot tud megoldani. A túlterheltség hatására megpróbálnak egyszerre több feladaton dolgozni, elvesztik a fókuszt ami demotiváltsághoz és a minőség csökkenéséhez fog vezetni. Végül egy vesztes-vesztes szituációban kötünk ki.

Ez ahhoz hasonló helyzet mintha több papírt próbálnánk a nyomtatóba tenni annak érdekében, hogy gyorsabban nyomtasson. Ha figyelmen kívül hagyjuk azt az alapvetést, hogy a bemenet nagyobb a kimenetnél akkor az elkerülhetetlenül elakadáshoz vagy szivárgáshoz fog vezetni.

A Scrum ennek elkerülésére a "tegnapi időjárás" módszert használja. A csapat megnézi mennyi volt az előző Sprint kimenete, legyen ez mondjuk 4-6 user story. Ezek után megnézik melyik az a 4-6 user story amivel a következő sprintben tudnak foglalkozni. A Product Owner feladata, hogy a lehetséges összes user story közül kiválassza azt a 4-6-ot, amit a következő Sprintben a legfontosabbnak tart a stakeholderek szemszögéből.

A Kanban megoldása, hogy határt szab a párhuzamosan végezhető feladatoknak. Tegyük fel, hogy a csapat egységesen 5 feladaton tud dolgozni egyszerre úgy, hogy mindenkinek jut elegendő feladat és nem terhelődik túl a csapat. Ezért úgy döntenek, hogy 5 story lesz az egyszerre végezhető feladatok maximuma. Csak akkor kezdenek neki új story megvalósításának, amikor egyet már befejeztek, így garantálva azt, hogy folyamatosan egységes mennyiségű új funkciót szállítanak.

Mindkét megoldás működhet jól. Mindkét megoldás esetén felhalmozódik jónéhány igény, ami várja, hogy sorra kerüljön egy fejlesztési ciklusban. A Scrum ezeket hívja Termék Backlognak.

» A Product Owner felelős a Termék Backlog hatékony menedzseléséért is, beleértve:

- A Product Goal meghatározását és egyértelmű kommunikálását
- A Product Backlog elemeinek elkészítését és világos kommunikálását
- A Product Backlog elemeinek sorrendezését; valamint
- Annak biztosítását, hogy a Product Backlog könnyen áttekinthető, látható és mindenki számára érthető legyen «

Ezt a sort valakinek karban kell tartani. Ha Sprintenként 10 új igény érkezik be a stakeholderektől és a Fejlesztők csak 4-6 új funkciót tudnak szállítani, akkor elég gyorsan eljuthatunk oda, hogy egy-egy új elem akár 6 hónapig is várhat míg sorra kerül. Így kialakulhat az a helyzet, hogy olyan dolgot fejleszt a csapat, amit valaki 6 hónappal korábban kért. Ez nem túl agilis.

Csak egy módon lehet kordában tartani az egyre növekvő számú igények sorát. Ez pedig nem más, mint a **NEM** szó. Ez a legfontosabb szó egy Product Owner szókincsében. Érdemes is minden nap a tükör előtt gyakorolni. Egy új kérésre igent mondani rendkívül könnyű. A legnehezebb feladat egy Product Owner munkájában az, hogy eldöntse mi lesz az a funkció ami nem fog bekerülni a termékbe és természetesen vállalja a felelősséget az ezzel járó következményekért.

A Product Owner feladata, hogy eldöntse mi az amin a fejlesztőcsapat elkezd dolgozni és mi az a minőség ami a fejlesztési folyamat végén átadható a stakeholdereknek. Szintén az ő feladata az, hogy eldöntse mi az ami a következő iteráció során kerül be a fejlesztési folyamatba és mi az ami csak később. Nem könnyű feladat, és hatékonyan csak a Fejlesztők és a stakeholderek aktív bevonásával kivitelezhető.

Ahhoz, hogy a Product Owner megfelelő sorrendet tudjon felállítani az egyes elemek között ismernie kell, hogy egy-egy elem mekkora üzleti értéket képvisel és mekkora erőfeszítést igényel a fejlesztése. Néhány funkció kritikus a működés szempontjából, néhány pedig inkább csak hab a tortán. Néhány órák alatt elkészül másokhoz akár hónapok is szükségesek.

Mi lehet a különbség egy-egy user story értéke és mérete között? Így van - semmi! Az, hogy egy user story-t nagy energiabefektetéssel lehet csak megvalósítani, nem jelenti azt, hogy üzletileg ugyanolyan nagy értéket képvisel. Gondoljunk bele, hogy egy olyan egyszerű funkció, mint az automatikus mentés egy szövegszerkesztőben,

mennyire fontos lehet, mekkora értéket képvisel a felhasználóknak és ezek mellett mennyire alacsony a fejlesztési költsége. Ugyanakkor a Microsoft Office gemkapocs funkciója jelenetős energiákat emészthetett fel a fejlesztés terén, míg fogadtatása és haszna nem egészen azt mutatta, hogy a felhasználók túlzottan lelkesedtek érte.

Az egyes user storyk értéke és mérete azok a tulajdonságok amik, segítenek a Product Ownernek, hogy kellően megalapozott döntést hozzon amikor sorba rendezi őket.

Ha két azonos méretű user story között kell eldönteni a sorrendet, akkor a nagyobb értékű fog előre sorolni, míg két üzletileg azonos értékű közül a kisebb energia ráfordítást igénylő kerül előrébb a sorban.

Itt merülhet fel a jogos kérdés, honnan tudja a Product Owner az egyes user storyk méretét vagy értékét. A rossz hír az, hogy ezek nem határozhatóak meg egyértelműen. Egyszerűen a megérzéseire kell hallgatnia. A későbbiek során megismerkedünk majd olyan technikákkal amelyek a megérzéseknél egy fokkal jobban segítenek az egyes storyk értékének rangsorolásában.

A megérzésein túl be kell vonnia a stakeholdereket is, hogy folyamatosan felmérje, hogy az adott pillanatban mi az amit értéknek tartanak. Az egyes user storyk méretének becsléséhez pedig folyamatosan beszélnie kell a Fejlesztőkkel, hogy ők is visszajelzést adjanak a fejlesztésre fordítandó energiáról. Az összes ilyen becslés csak relatív és nem abszolút. Nem tudom megmondani segédeszköz nélkül, hogy mekkora egy alma és egy eper pontos súlya, de azt elég biztosan meg tudom állapítani, hogy az alma ötször nagyobb mint az eper és az eper nekem jobban ízlik. Ennyi elég is ahhoz egy Product Ownernek, hogy sorba tudja állítani a user storykat.

Egy új projekt indulásakor szinte az összes becslés nagyot fog tévedni, de ez teljesen rendben is van így. Ami számít ilyenkor az a becslések során elkezdődő párbeszédek és nem a becslés tényleges végeredménye. Minden egyes alkalommal

amikor a csapat szállít valami kézzel foghatót a felhasználóknak visszajelzést kapnak. Ezek a visszajelzések pedig egyre javítják a becsléseket mind méret, mind érték szempontjából. Ezért szükséges folyamatosan sorba rendezni a user storykat és újra megbecsülni azokat. A teljes víziót megbecsülni az elején elég rossz hozzáállás, hiszen a projekt elején áll rendelkezésünkre a legkevesebb információ. A visszajelzések a Product Owner legjobb barátai.

A user storyk sorba rendezése önmagában nem elég. Ahhoz, hogy a termék korán a felhasználóknál legyen és gyakran tudjuk frissíteni a meglévő user storykat apró falatnyi részekre kell bontani. Ideális esetben egy ilyen falat nem igényel néhány napnál több munkát. Egy tölcsérbe kell töltenünk a user storykat ahol a tölcsér szájánál kicsi és jól definiált elemek találhatóak, míg a bemenetnél a sokkal tágabban értelmezett és nagyobb elemek. Ha ezt a lebontást mindig akkor csináljuk meg, amikor az adott funkció közel kerül a fejlesztéshez, azzal tudjuk azt garantálni, hogy mindig a lehető legaktuálisabb információk alapján hozunk döntést az egyes funkciókról.

A fentieket - tehát az egyes user storyk méretének és értékének becslését, sorba rendezését és lebontását - összességében backlog refinementnek nevezzük. A Product Owner heti egyszer összehívja a Fejlesztőket - esetleg csatlakozik néhány stakeholder is - és egy műhelymunka keretében egyeztetnek a backlogról. A napirend változhat, néha a becslés, néha a user storyk lebontása a cél.

Az eddigiekből kirajzolódni látszik egy mintázat: kommunikáció! A Product Owner legfontosabb feladata a folyamatos kommunikáció. Ez teljesen összhangban áll az Agilis Kiáltvány első pontjával:

> **“Az egyéneket és a személyes kommunikációt (értékeljük)** a módszertanokkal és eszközökkel szemben”

A Product Owner feladata tehát nem az, hogy kiskanállal etesse a csapatot user storykkal. Ez egyrészről unalmas másrészről nem túl hatékony. A Product Owner feladata, hogy mindenkivel megértesse mi a víziója, biztosítsa a közvetlen kapcsolatot a Fejlesztők és a stakeholderek között és hogy elősegítse, hogy a valódi felhasználóktól érkező visszajelzések minél hamarabb beépülhessenek a termékbe. Ezzel segítve a Fejlesztőket abban, hogy egyre önállóbban tudjanak szakmai döntéseket hozni, míg a Product Owner a teljes képre fókuszál.

A Product Ownernek és a Fejlesztőknek jó pár kompromisszumos döntést kell meghozni a munkájuk során. Nézzünk néhány példát.

Elsőként kompromisszumot kell kötni a különböző értékek között. Egy új projekt elején a bizonytalanság és a kockázatok jelentik a legnagyobb veszélyt.

Létezik üzleti kockázat: biztos, hogy a megfelelő dolgot fejlesztjük? Csoport kockázat: a Fejlesztő csapat összetétele lehetővé teszi-e, hogy megvalósítsuk a terméket? Technológiai kockázat: a platform amin szeretnénk megvalósítani megfelelő lesz a terméknek? Mennyire lesz skálázható? Létezik még ezen túl a költségek és a határidő kockázata is: Reális idő- és költség korlátok között be tudjuk fejezni a fejlesztést?

A tudás a kockázat ellentéte. Amikor tehát nagy a bizonytalansági tényező akkor a tudás megszerzésére és rendszerezésére kell fókuszálni - ilyen például egy felhasználói felület prototípusának elkészítése, technológiák kipróbálása vagy egyszerű kísérletezés. Ezek nem túl érdekesek a termék felhasználói oldaláról, de ettől még értéket hordoznak, hiszen csökkentik a jövőbeni kockázatot.

A jövőben a bizonytalanság fokozatos csökkenésével a csapat folyamatosan egyre többet tud az ügyfelek számára is értékes dolgokra koncentrálni. Egyre inkább kikristályosodik, hogy mit és hogyan szeretnénk fejleszteni és már csak a tényleges

munka marad. Az értékesebb user storykkal kezdve pedig egy szép meredek értéknövekedést tud megvalósítani a csapat.

Az idő előrehaladtával ez a görbe ellaposodik. A legfontosabb funkciókat már használják az ügyfelek, a csapat már a bónusz funkciókra koncentrál, ez a hab a tortán. A fejlesztési folyamatnak ez a szakasza a legkényelmesebb, mivel bármely ponton az tudja mondani a Product Owner és a Fejlesztők, hogy abbahagyják az adott funkcióval kapcsolatos fejlesztést és más fontosabb funkcióval folytatják tovább. Ez az üzleti agilitás.

Ezek alapján amikor értékről beszélünk akkor a tudást és az ügyfél értéket értjük alatta és meg kell találnunk azt a kompromisszumot, ahol egyensúlyban van a kettő.

A második fontos kompromisszum a hosszú és rövid távú gondolkodás között lévő szakadékban rejlik. Mit fejlesszünk következőként? Előrébb vegyük-e a sürgős hibajavítást vagy fejlesszük inkább azt az új funkciót ami elkápráztatja az ügyfeleket, esetleg fejlesszük tovább a teszt rendszerünket ami majd gyorsabb fejlesztést tesz lehetővé a jövőben? Folyamatosan egyensúlyozni kell a reaktív és proaktív munka valamint a tűzoltás és a tűzvédelem között.

Ez egy másik kompromisszumhoz is kapcsolódik. Arra kell-e a hangsúlyt fektetnünk, hogy

“a megfelelő dolgot fejlesszük”

“megfelelő módon fejlesszük azt a dolgot” vagy

“gyorsan fejlesszük a dolgot”?

Ideális esetben mindhármat szeretnénk, de nem könnyű megtalálni az egyensúlyi pontot. Vegyünk egy egyszerű példát: a lehető legjobb terméket szeretnénk megvalósítani a lehető legtökéletesebb architektúrával. Ha túl sok időt töltünk a

tökéletesítéssel, akkor elmulaszthatjuk a megfelelő időpontot a piacra lépéshez és könnyen pénzügyi problémákba futhatunk.

Vagy vegyünk egy másik példát: már a prototípussal ki szeretnénk lépni a piacra. Ez rövid távon akár jó is lehet, de hosszú távon rendkívüli mennyiségű technikai problémát cipelünk magunkkal és a csapat értékteremtő képessége közel lesz a nullához.

Vagy építhetünk rekord idő alatt egy csodás katedrálist, ha kiderül, hogy az ügyfelek valójában egy lakókocsit szerettek volna.

A Scrum szerepek között létezik egy egészséges feszültség a fentiek kapcsán: a Product Owner szeretné a megfelelő dolgot a felhasználók kezeiben látni. A Fejlesztők szeretnék a leginkább megfelelő módon megvalósítani azt a dolgot. A Scrum Master pedig igyekszik a lehető legrövidebbre venni az utat amin megérkezik a visszajelzés a csapathoz.

A sebesség fontos tényező. Ha hamar kap visszajelzést a csapat az felgyorsítja a tanulást, így hamarabb kiderül, hogy mi a megfelelő termék és hogyan lehet megfelelően megvalósítani. Mindhárom tényező fontos ezért érdemes megtalálni az egészséges egyensúlyt közöttük.

A harmadik és utolsó kompromisszum pedig az új termék fejlesztése és a régi termék javítása között található. A termék backlog lehet egy megtévesztő fogalom, hiszen azt jelzi, hogy csak egy termék van. A projekt szintén lehet megtévesztő, hiszen azt jelzi, hogy a termék fejlesztése egyszer véget ér. Egy termék valójában soha nem lesz “kész”, mindig van karbantartási és javítási feladat egészen addig, amíg az adott termék elér az életciklusa végére és megszüntetjük.

Mi történik a régi termékkel, ha egy új fejlesztésébe kezdünk? Átadni a terméket egy másik csapatnak költséges és kockázatos lehet. Ami ennél sokkal gyakoribb, hogy

ugyanaz a csapat tartja karban a régi terméket aki fejlesztette azt és ezzel párhuzamosan fejlesztik az újat is. Így a csapat backlogja sokkal inkább kapcsolódik magához a csapathoz, mint egy konkrét termékhez - összességében olyan elemeket fog tartalmazni amelyeket a Product Owner szeretne megvalósítani a jövőben és akár több termékből is találhatóak meg funkciók benne. A Product Owner feladata pedig, hogy megtalálja ezen elemek között az egyensúlyt.

Néha egy-egy stakeholder megkeresi a Product Ownert és megkérdezi a következőket:

“Mikor lesz kész amit várunk?”

“Mennyi fog elkészülni belőle karácsonyig?”

A Product Owner feladata ebben az esetben a várakozások kezelése. Ami még ennél is fontosabb: a várakozások megtartása a realitás medrében. Ezt azt jelenti, hogy egy Product Owner semmiképpen nem fog hazudni a stakeholdereknek. Ez nem is annyira könnyű feladat ha belegondolunk, de az agilitás kemény dió tud lenni néha.

Előrejelzést nem annyira nehéz készíteni, egészen addig amíg nem kell túlságosan precíznek lennie. Ha megmérjük a csapat sebességét vagy az összes csapat együttes sebességét fel tudjuk rajzolni a “Burnup” grafikont. Ez a grafikon kumuláltan mutatja meg, hogy mennyi az összes eddigi szállított user story.

Vegyük észre a különbséget: ez a grafikon a fejlesztési folyamat kimenetét mutatja, ami nem egyezik meg az ügyfél értékkel. Nem az a célunk, hogy minél nagyobb legyen a kimeneti mennyiség, hanem az hogy minél több értéket tudjunk előállítani a mennyiség arányában. A kevesebb néha több.

Vegyünk egy burnup grafikont és rajzoljunk bele egy optimista és egy pesszimista trendvonalat. Ehhez nyúlhatunk a statisztika eszköztárához vagy egyszerűen

behúzhatjuk kézzel is. A két trendvonal közötti különbség mutatja meg mennyire bizonytalan az adott csapat sebessége. Ez idővel egyensúlyba kerül és a bizonytalansági tölcsérünk egyre szűkebb lesz.

Rendben, vissza a várakozások kezeléséhez!

Tegyük fel, hogy a stakeholderek azt kérdezik a Product Ownertől:

“Mikor lesz kész MINDEN amit szeretnénk?”

“Mikor érkezünk a következő mérföldkőhöz?”

Ezeknél a kérdéseknél a scope rögzített az idő pedig változó tényező. A Product Owner ránéz a burnup grafikonra és azt mondja:

“Körülbelül április-május környékén”.

Tegyük fel, hogy a stakeholderek azt kérdezik a Product Ownertől:

“Mi lesz kész karácsonyra?”

Ebben az esetben az idő rögzített míg a scope egy változó tényező. A trendvonalra újra ránézve meg tudjuk mondani:

“X funkció teljesen kész lesz, Y egy része is, de Z egyáltalán nem”.

A harmadik lehetséges kérdés:

“Meglesznek EZEK a funkciók karácsonyra?”

Ebben az esetben mind a scope mind az idő rögzített tényező. A legjobb válasz ilyen esetben a Product Ownertől:

“Nem, sajnos ez nem lesz kész addigra.”

majd közvetlenül utána

“Nagyjából EZT és EZT tudjuk karácsonyra szállítani”

vagy

“még ennyi időre van szükség.”

Általánosságban sokkal jobb a scope-ot módosítani mint a fejlesztésre szánt időt kiterjeszteni. Ha először a scope-ot vágjuk meg, később még mindig dönthetünk úgy hogy több időt hagyva fejlesszük a korábban kihagyott funkciókat. Fordítva nem működik, hiszen ha a fejlesztésre szánt idő elmúlt akkor azt visszaforgatni már nem tudjuk. A Product Owner végső válasza tehát:

“Tudunk szállítani egy részt a termékből a határidőre és a többit későbbre halasztjuk, vagy most nem szállítunk semmit és a kért funkciót EKKORRA tudjuk szállítani? Melyik lehetőség a szimpatikusabb?”

Az ehhez kapcsolódó számítások viszonylag egyszerűek, a Product Owner pedig folyamatosan tudja frissíteni az előrejelzését.

Fontos kiemelni, hogy a Product Owner ilyen esetben empirikus adatokra alapozza az előrejelzést a vágyálmok helyett. Ezen túl teljesen őszinte a bizonytalansággal kapcsolatban. Ez egy nagyon egyenes módja annak, hogy a stakeholderekkel kommunikáljon, akik jó esetben díjazzák is ezt. Ha a szervezetre nem jellemző az egyenes és őszinte kommunikáció az agilitás sem fog egyszerűen gyökeret verni.

Egy dologra azonban oda kell figyelni. Ha a Fejlesztők túl sok technológiai adósságot halmoznak fel - például nem írnak jó teszteket vagy nem fejlesztik folyamatosan az architektúrát - akkor a sebességük időről-időre lassulni fog, a burnup grafikon pedig ellaposodik. Ez szinte lehetetlenné teszi a becslést, ezért a Fejlesztők felelőssége, hogy egyenletes sebességgel dolgozzanak a Product Ownernek pedig el kell kerülnie, hogy olyan nyomást gyakoroljon a csapatra, amivel rossz döntésekre kényszerítheti őket.

Mi a helyzet akkor, ha egy nagyobb projekten dolgozunk több csapattal? Egy 5 fős csapat helyett mondjuk 3 csapat dolgozik a projekten. Több Product Owner is van és mindegyiknek saját backlogja is a termék különböző komponenseivel.

Összességében ugyanezeket az elveket alkalmazhatjuk ebben az esetben is. Szükség lesz a csapat kapacitásának kezelésére, az egyeztetésre a stakeholderekkel, Product Ownerekre akik tudnak nemet mondani, a backlog refinementre.. A projekt sebességét a csapatok kimeneteinek összessége fogja megadni és ezt is fel lehet használni előrejelzések készítéséhez. Vagy - ha ennek van értelme - a csapatok külön is készíthetnek előrejelzést.

Az ilyen több csapatos felállásokban azonban a Product Ownereknek van még egy felelősségük: egymással is beszélni kell. A csapatokat és azok backlogjait úgy kell kialakítani, hogy minimálisra csökkentsük a függőségeket. Biztosak lehetünk benne, függőségek mindig lesznek, ezért a Product Ownereknek egyeztetni kell arról, hogy megfelelő sorrendben kerüljenek a termék egyes komponensei a fejlesztési ciklusokba.

A termékvízió

A termék víziója annak egy általunk elképzelt ideális jövőbeni állapotát írja le.

Azok számára akik részt vesznek majd abban a folyamatban, hogy a termék létrejöjjön és elérje ezt az ideális állapotot a termékvízió mutatja majd az irányt mint egy világítótorony. A jól megalkotott termékvízió leírja, hogy milyen problémákra fog a termék megoldást nyújtani, kik lesznek a termék fő felhasználói és milyen egyedi értéket képes nyújtani a termék ezeknek a felhasználóknak. A jól megalkotott termékvízió segít abban is, hogy a szervezeten belül támogatást nyerjünk a termékünknek vagy a szervezeten kívülről forrásokat vonjunk be annak megalkotásához és fejlesztéséhez.

A termékvízió alapjául szolgál majd a stratégiai tervezésnek, a termékkel kapcsolatos mérföldkövek lefektetésének. Segít abban, hogy döntést hozzunk ha több irányt vehet a termék fejlesztési folyamata.

Mit kell tartalmaznia a termékvíziónak?

A cégek azért létezhetnek mert vannak ügyfeleik akik fizetnek azért hogy megoldást kapjanak egy problémára. Ezek lehetnek emberek vagy más vállalatok akik a cég termékét használják. Mivel a szervezet létezése függ ettől a tényezőtől az elsődleges tartalom a termékvízióban, hogy kik lesznek az ügyfelek akik a termékért fizetni fognak.

A termékvízió egy kívánatos jövőbeli állapotot ír le a termékkel kapcsolatban. Mivel a vízió időtávja általában több évet ölel fel ezért ennek az állapotnak vagy célnak elérhetőnek de kihívást jelentőnek kell lennie. Ez nem is olyan egyszerű mint elsőre hangzik. Ha túlságosan merészek az elképzeléseink akkor nagy valószínűséggel nem valósulnak meg és ez egyrészről demotiválja a szervezet tagjait másrészről bizalomvesztéshez vezethet a felhasználók körében. Ha túl alacsonyra tesszük a lécet akkor pedig nem fogunk kitűnni a versenytársak közül és vagy az ügyfélkör méretével vagy a piaci eredményekkel fogunk fizetni a középszerűségért. Az első esetre tipikus példa lehet a Theranos esete ahol az alapító azt ígérte, hogy néhány csepp vérből komplett egészségügyi diagnózist képes felállítani a termékük. Egy évtizeddel később most a börtönbüntetését várja az alapító, mivel nem csak a víziót nem sikerült megvalósítania, de a befektetőket is átverte a be nem váltott ígéretekre összegyűjtött dollár százmilliókkal. Az utóbbi eset pedig az összes olyan frissen induló cégre igaz akinek az a víziója, hogy “a termékünk olyan lesz mint a Facebook, csak más”.

Ha bárki elolvassa a termékvíziónkat értenie kell mik azok amik motiválják a szervezetünket abban, hogy elérje azt. A motivácó a “Miért?” kérdésre fog választ adni és kifejezetten nem a “Hogyan?”-t írja le.

Ha a termékvízió tartalmi elemeit létrehoztuk érdemes a formai részre is hangsúlyt fektetni és olyan rövid, könnyen megérthető alakot adni neki ami könnyen elülteti a bogarat a megfelelő fülekbe.

Az időtáv tekintetében a 3-5 éves periódus az ideális. Ennél rövidebb idő alatt nem tudunk kellően ambíciózus célokat megvalósítani az ennél hosszabb időtáv alatt pedig annyira jelentős változások következnek be a piaci és technológiai környezetben amelyek szükségessé teszik a termékvízió módosítását. Az időtávot nem kell feltétlenül rögzíteni a termékvízió kialakításakor, sokkal inkább egy olyan időszakot határozunk meg ezzel amely elteltével érdemes felülvizsgálni a termékkel kapcsolatos céljainkat és szükség esetén módosítani azokat.

Nézzünk néhány példát arra, hogy egyes piaci szereplők hogyan fogalmazták meg a termékvíziójukat.

Facebook

» Az emberek arra használják a Facebookot, hogy kapcsolatban maradjanak a barátaikkal és családjukkal, hogy felfedezzék mi történik a világban és hogy megosszák és kifejezzék mit tartanak fontosnak. «

Tesla

» A 21. század leglenyűgözőbb autó vállalatát hozzuk létre és mi leszünk a hajtóereje a világ elektromos járművekre való átállásának. «

Apple

» Hisszük, hogy azért vagyunk a Földön, hogy kiváló termékeket hozzunk létre, és ez nem változik. «

Szinte mindegyik komoly márka termékvíziója belesűríthető egy összetett mondatba, de ezek a mondatok semmiképpen nem a folyamat elején születnek, hanem hosszú kutatás, kísérletezés és közös gondolkodás előzi meg őket.

A következőkben olyan a Scrum keretein kívül létező, de az agilis működésbe jól beilleszthető eszközöket és módszereket ismerhetünk majd meg amelyeket jól használva a sikeres márkákhoz hasonló víziót alkothatunk meg a termékünkkel kapcsolatban.

Bevezetés a Service Design világába

A digitalizáció széles körű elterjedéséig elég könnyen meg lehetett különböztetni, hogy egy vállalat termékek vagy szolgáltatások értékesítésével foglalkozik. A technológiai fejlődéssel ez a határ azonban elmosódott és leginkább úgy írható le az új állapot, hogy egy-egy vállalat egy bizonyos spektrumot fed le abból a skálából amelynek egyik végén a kizárólag fizikai terméket előállító vállalatok találhatóak, míg a másik végén a szigorúan szolgáltatási tevékenységet végző cégek helyezkednek el. A jobb érthetőség kedvéért nézzünk néhány példát arra milyen spektrumon mozognak a modern márkák.

Ha a Ford-ot vesszük alapul akkor egész sokáig elmondhattuk, hogy kizárólag az autóikkal azaz fizikai termékekkel jelentek meg a piacon. Ha egy pár éve gyártott Fordba ülünk be azonnal láthatjuk a különbséget: fedélzeti navigáció, digitális rádió - mind olyan tartozékai a terméknek ami nélkül képtelenség lenne versenyképesnek maradni a piacon. Ezek a hozzáadott értéket képviselő szolgáltatások ugyanúgy része a terméknek és együtt alkotják a felhasználónak nyújtott élményt.

A skála másik oldaláról a navigációs megoldások piacát nézhetjük meg amelyek az induláskor külön fizikai termékként jelentek meg a piacon az okostelefonok elterjedésével azonban folyamatosan leépítették ezt a termék spektrumot és szinte kizárólag szolgáltatásokkal maradtak jelen a piacon.

A hagyományos kereskedelmi áruházakban - mint például a Tesco - is teret nyert a digitalizáció és a napi bevásárlás mellett már hűségprogramban is részt vehetünk vagy egy drágább termékre hitelt is igénybe vehetünk közvetlenül a kereskedőtől.

Ezek a folyamatok oda vezettek, hogy szükségessé vált egy olyan módszertan kialakítása amely az adott vállalat által az ügyfeleknek nyújtott csomagot egységesen tudja kezelni, mindenre kiterjedően tud folyamatokat kidolgozni és végig a tökéletes ügyfélélmény megalkotására törekszik. A Service Design ezt a feladatot igyekszik megvalósítani.

Egy új telefon előfizetés megvásárlásának példáján keresztül nézzük meg madártávlatból hogyan működik a Service Design.

1. Mielőtt az ügyfél kiválasztja a számára legmegfelelőbb csomagot körbe fog nézni a piacon, megvizsgálja több szolgáltató ajánlatát és összehasonlítja azokat egymással és a saját szokásaival.
2. Ha meghozta a döntését akkor kiválasztja a csomagot - és esetleg a készüléket amivel használni szeretné - majd megadja a szerződéskötéshez szükséges adatait.
3. A szerződés aláírásához és véglegesítéséhez elmegy egy kirendeltségre vagy digitálisan megköti a szerződést.
4. Kézhez kapja a szolgáltatás használatához szükséges eszközöket (SIM kártya és készülék) és elkezdi használni azokat.
5. A szolgáltatás díját adott időközönként fizeti a szolgáltatónak a kapott számlák alapján

6. Ha problémája vagy kérdése merül fel a szolgáltatással kapcsolatban megoldást kér a szolgáltatótól.
7. Ha elégedetlen a szolgáltatással megszünteti azt.

Az ügyfél a termékkel töltött életciklusa összes szakaszában más-más interakciót igényel. Minden egyes szakaszban azon dolgozik a Service Design, hogy a lehető legjobb élményben legyen része az ügyfélnek - igen még akkor is amikor megszünteti a kapcsolatát a vállalattal.

A Service Design alapvetően két dolgot valósít meg:

- Egy meglévő szolgáltatás próbál meg újragondolni és megkönnyíteni az azt használó ügyfelek életét.
- Egy teljesen új szolgáltatás létrehozásakor az első lépéstől az utolsóig megtervezi és az ügyfelek számára élménnyé teszi a szolgáltatás használatát

A két cél eléréséhez nem csak azokat az interakciókat, felületeket vagy eszközöket kell megvalósítani amelyekkel az ügyfél közvetlenül találkozik, hanem azokat a háttérben zajló részleteket is amelyek ezeket kiszolgálják.

Vegyük példának a telefonos előfizetés vásárlás harmadik lépését a szerződéskötést. Az ügyfél oldaláról ez annyit jelent, hogy aláírja a szerződést és kap belőle egy példányt, majd folytatja az útját a többi ponton. A háttérben ez egy sokkal szélesebb körű folyamatot indít el: a szerződést iktatni- és tárolni kell, kereshetővé kell tenni, az ügyfél adatait rögzíteni kell a számlázási rendszerben és biztosítani kell, hogy mind a szerződést mind az azon szereplő adatokat biztonságban tudja a cég.

Csakúgy mint egy mozifilm esetén amögött, hogy a néző jól érzi magát a moziban popcornnal és kólával a kézben több száz ember munkája áll a kamera mögött. A Service Design célja az, hogy ezek a szereplők is a lehető legjobban érezzék magukat és ezek a folyamatok is egyszerűek és jól használhatóak legyenek.

Mielőtt rátérnénk arra, hogy milyen eszközöket használ a Service Design nézzük meg melyek azok az alapelvek amelyekre a módszertan épül és segítenek abban, hogy a létrejövő folyamatok a legjobb ügyfélélményt hozzák létre.

Ügyfélközpontúság

Többször említettük a korábbiakban, hogy a Service Design fókuszában az áll, hogy azok akik megismerik vagy igénybe veszik a vállalatunk szolgáltatását azok boldogabbak legyenek a találkozás után.

Kiváló példa erre a szemléletre a Zappos akik már a termékvíziójukat is ennek függvényében fogalmazták meg: "Boldogságot szállítunk az ügyfeleinknek, kollégáinknak és a beszállítóknak." Az alapvetően cipők online értékesítésével foglalkozó vállalat ezt többek között úgy ültette át a gyakorlatba, hogy az ügyfélszolgálaton dolgozó kollégák hívás ideje nem volt maximálva, azaz addig beszéltek egy-egy ügyféllel amíg az kellően boldogan nem köszönt el. Még akkor is ha ez azt jelentette, hogy meghallgattak egy történetet az ügyfél kutyájának műtétjéről. Az ügyfelek imádnak ennél a cégnél vásárolni ami hatalmas versenyelőnyt jelentett számukra.

Közös alkotás

Egy új termék- vagy szolgáltatás megtervezésénél vagy egy meglévő továbbfejlesztésénél a Service Design fontosnak tartja, hogy bevonjuk a folyamat összes szereplőjét. Ez azt jelenti, hogy nem csak az ügyfelek véleményét hallgatjuk meg, hanem adott esetben az értékesítők, ügyfélszolgálatosok, adminisztratív dolgozók álláspontja is a folyamat részévé válik.

Egy saját projektemben egy POS alkalmazást fejlesztettünk amellyel lehetővé vált az ügyfelek számára, hogy akár közvetlenül a kiválasztott termék kiválasztása után fizessenek az értékesítőnél és ne kelljen bejárni az üzletet az adminisztráció miatt.

Még csak egy prototípus készült el, de már közvetlenül a későbbi felhasználókkal: az értékesítőkkel teszteltük a működést. Rengeteg értékes visszajelzést kaptunk ebben a folyamatban amivel jobbá tehettük a szolgáltatást és amikor ténylegesen kézbe kapták az értékesítők az eszközöket és a szoftvert nagyon szívesen használták azt ami az értékesítési számokon és az ügyfél elégedettségi mutatókon is látszott.

Sorba rendezés, szakaszokra osztás

A Scrumhoz hasonlóan a Service Design is arra törekszik, hogy komplex problémákat egyre kisebb és így egyre emészthetőbb, kézzel foghatóbb egységekre bontson. A telefon előfizetés vásárlása ügyfél oldalról egész egyszerűnek tűnik, de a háttérben ezt bonyolultabb folyamatok zajlanak. Ahhoz hogy jobban megértsük ezeket érdemes a teljes folyamatot szakaszokra bontani, azonosítani az egyes szakaszokban az érintett szereplők problémáit majd ezek mentén a teljes folyamatot magasabb szintre emelni.

A fenti példánál maradva a szerződés aláírása mind az értékesítőnek mind az ügyfélnek egy hosszú és kényelmetlen folyamat. Erre adhatjuk azt a választ az optimalizálás során, hogy az okmányokon szereplő ügyféladatokat nem kézzel visszük fel a rendszerekben, hanem lefotózzuk és szoftveres támogatással átemeljük őket közvetlenül a dokumentumba. Az aláírás pedig történhet egy tableten akár ujjal is. Ha újra megvizsgáljuk ezek bevezetése után a nagy képet akkor azt látjuk, hogy az adott szakaszban megoldottunk egy problémát és a többi szakaszban is lerövidült az ilyen szerződések feldolgozásának ideje.

Valódiság

A cég termékének vagy szolgáltatásának minden egyes alkalommal azt a minőséget és értékrendet kell tükröznie amit a cég, a márka az ügyfelek felé kommunikál.

Vegyünk egy hazai példát a kifli.hu-t. Nagyon személyesen kommunikálnak az ügyfelek felé és a termékek és szolgáltatások magas minősége jelenik meg az imidzsükben. Egy rosszul kiszállított vagy hibás termék reklamációját akár telefonon akár az online felületen keresztül ennek megfelelően rendkívül rugalmasan kezelik. Az ügyfélnek még arra is lehetősége adódik, hogy kis értékű visszatérítések esetén a visszakapott összeget felajánlja egy segélyszervezetnek. A szolgáltatás minősége és a vállalatról alkotott kép teljes összhangban van egymással.

Holisztikus nézet

Az ügyfelek számos ponton érintkezhetnek a szolgáltatásainkkal és kiemelten fontos, hogy minden alkalommal ugyanazt a kiváló élményt tapasztalják meg.

Ha az ügyfelünk kap egy biztosítási ajánlatot egy értékesítőtől és alszik rá egyet, majd úgy dönt, hogy megrendeli, de a telefonos ügyfélszolgálaton újra azzal kell kezdenie, hogy azonosítja magát és neki kell elmondania az ajánlat részleteit akkor rossz szájízzel fog indulni a kapcsolat. Ha a két csatorna összeköttetésben van, akkor a hívás elején már látható is a kapott ajánlat és elég a részletekről egyeztetni. Ez nem csak azért jó mert itt még lehetőség van további értékesítésre is, hanem azért is mert csatornától függetlenül halad előre a folyamat az ügyfél oldalán pedig zökkenőmentes ügyintézést tesz lehetővé.

Service Design eszközök

Az alapelvek után nézzünk meg néhány olyan eszközt amik ezekre épülve segítenek, hogy jobban megértsük az ügyfeleket ezáltal a számukra legmegfelelőbb szolgáltatást vagy terméket hozhassuk létre, vagy feltérképezzük azokat az interakciókat amelyek során az ügyfelek találkoznak a termékünkkel és megtaláljuk azokat a pontokat ahol optimalizálni tudjuk ezeket.

Hogyan alkossunk ügyfél Personát?

Az ügyfél Persona egy adatokon és tényeken alapuló, képileg megjelenített fiktív karakter, amely az ügyfelek egy adott csoportját testesíti meg. A jól megalkotott ügyfél Persona segít abban, hogy megértsük, közelebb hozzuk magunkhoz és folyamatosan láthatóvá tegyük az ügyfeleket - követve az ügyfélközpontúság alapelvét.

A létrehozásuk nem egyszerű feladat. Számos valós ügyféllel történt beszélgetés rögzítése előzi meg. Az interjúk során nem csak az ügyfeleket szeretnénk mélyen megismerni, hanem a termékünkhöz vagy szolgáltatásunkhoz való viszonyukat is. A beszélgetések során azonosítani kell azokat a pontokat amelyek a legnagyobb nehézséget okozzák az ügyfeleinknek és azokat is amelyek miatt újra és újra mellettünk döntenek. Érdemes újra megjegyezni, hogy valós, a vállalatunkkal kapcsolatban álló ügyfelek véleményét sűrítsük a kialakított Personába, mert ezzel tudjuk jobban megérteni a viselkedésüket és elérnia azt, hogy új ügyfeleket szerezzünk.

Miért hasznos az ügyfél Personák megalkotása a kimagasló termék- és szolgáltatás portfólió létrehozásában?

Az ügyfelek motivációjának és szükségleteinek megértése

Az ügyfél Personák megalkotása segít összegyűjteni és átláthatóvá tenni a termékeinket vagy szolgáltatásunkat használók egy adott csoportját. A később a termékfejlesztés során létrehozott funkciók könnyen összehasonlíthatóvá válnak az ügyféligényekkel és megfelelő mederben tartják a fejlesztést. Ha kétség merülne fel bennünk, hogy egy adott termékfunkció megfelelő lesz-e az ügyfeleink számára érdemes megvizsgálni, hogy a létrehozott Persona használná-e azt és ha igen hogyan. Ezzel elkerülhetjük azokat a helyzeteket, hogy egy akár magas költségek árán létrehozott funkció a termékünkben nem teremt értéket.

Az ügyfelek nehézségeinek megoldása

Az ügyfél interjúk során számtalan olyan információ fog a birtokunkba jutni amelyek arra vonatkoznak, hogy miért nehézkes vagy akár lehetetlen használni a termékünk egészét vagy egy-egy funkcióját. A termék fejlesztése során ezeket figyelembe véve olyan megoldást tudunk létrehozni amely könnyebbé teszi a felhasználók életét akik ezért elköteleződésükkel fognak fizetni.

Vásárlási szokások feltérképezése

Az ügyfeleink egymástól elkülönülő csoportjai másképpen fogják használni a termékeinket és a szolgáltatásainkat. Lesznek akik szeretik ha tudnak böngészni és részletesen tájékozódni arról amit kínálunk, mások jobban kedvelik, ha korábbi vásárlásaik alapján egyből ajánlunk nekik valamit. Ezek mind megjelennek egy jól megalkotott ügyfél Personában, így a fejlesztés során tudatos döntést hozhatunk arról, hogy melyik ügyfélkörünk milyen szokásának szeretnénk megfelelni.

Új lehetőségek felfedezése

Amikor beszélgetünk egy ügyféllel gyakran előfordul majd velünk, hogy olyan igényekkel találkozunk amiket nem egyértelműen közölnek velünk, de a sorok között olvasva olyan tudás birtokába juthatunk amely végső soron növelni fogja az elégedettséget. Henry Fordot idézve: “Ha megkérdeztem volna a vásárlóimat mit szeretnének, azt válaszolták volna, hogy gyorsabb lovakat.” Forddal ellentétben nekünk nagyon is érdemes folyamatosan megkérdezni az ügyfeleinket mit szeretnének, de több kérdéssel fel kell tudnunk azt mérni, hogy a felszín alatt is találunk-e rejtett igényeket.

Az egyértelműség csökkenti a kockázatot

Egy megalapozott és részletes ügyfél Persona világítótoronyként jelzi majd, hogy helyes irányba halad-e a termékfejlesztés. Ha kétségünk merül fel a folyamat során

elég lesz ránézni az ügyfél Personára és könnyen megállapíthatjuk, hogy a tervezett funkció lefedi-e az általa szimbolizált igényeket és ha igen milyen mértékben. Így segítségünkre lehet nem csak a megfelelő termékfunkciók kiválasztásában, hanem azok sorrendjének felállításában is.

Az alapelvek megismerése után nézzük meg a gyakorlatban milyen lépéseken keresztül hozhatunk létre egy jól használható ügyfél Personat.

Tulajdonságok meghatározása és összegyűjtése

Ahhoz, hogy egy jól körülhatárolható Personát alkossunk meg minél több oldalról meg kell vizsgálnunk azokat akik használják a termékünket vagy szolgáltatásunkat. Nincs előre meghatározott módja annak, hogy milyen adatokat gyűjtsünk be róluk, de az alábbi lista egy jó kiindulópont lehet:

- Foglalkozás
- Demográfiai adatok
- Személyes történet
- Családi állapot
- Vagyoni helyzet
- Iskolai végzettség
- Kihívások amelyekkel az életben vagy a termék használata során szembesül az ügyfél
- Mire van szüksége ahhoz, hogy ezekkel a kihívásokkal megküzdjön
- Egy idézetet amely jól leírja a személyiségét

Az így összegyűjtött és rendszerezett adatok jó alapot jelenthetnek ahhoz, hogy egy fiktív személyiségbe sűrítsük az ügyfelek adott csoportját.

A Persona finomítása

A termékünk vagy szolgáltatásunk létrehozását jól megalapozhatja ha egy kutatáson alapuló ügyfél Persona alapján döntünk a funkciókról vagy fejlesztjük azokat tovább. Ez azonban semmiképp se jelentse azt, hogy időről időre nem vizsgáljuk felül a létrehozáskor használt módszerekkel a már létező ügyfél Personákat. Ehhez számtalan forrásból szerezhetünk információt az ügyfél interjúkon túl is: olvassunk a termékeinkkel, piacunkkal kapcsolatos online fórumokat vagy a sajtóban megjelent elemzéseket. Kövessük nyomon a konkurencia megjelenéseit. Figyeljünk meg más akár a miénktől eltérő termékeket és trendeket és emeljünk át ötleteket ezekről a piacokról.

Minél több rendszerezett információ áll rendelkezésünkre annál jobban írja le a Personánk a valós ügyfeleket és azok viselkedését.

Mintázatok azonosítása

Kellőn nagy mennyiségű adat összegyűjtése után könnyen elképzelhető, hogy olyan mintázatokat látunk kialakulni amely alapján érdemes lesz határvonalakat meghúzni és több Persona segítségével leírni az ügyfél körünket. Természetesen nem lesznek éles határvonalak az ügyfélkörünkön belül több olyan pontot is fogunk találni amelyek összekötik őket. A mi feladatunk az a folyamatban, hogy azonosítsuk ezeket, meghatározzuk mennyire fontosak az egyes ügyfélköröknek és a termékünk fejlesztését ennek megfelelően priorizáljuk.

Az összegyűjtött információ rendszerezése

Már az előző pontban vázoltak alapján is láthatjuk, hogy kellő mennyiségű információ alapján elkezdenek kirajzolódni a leendő Personánk körvonalai. Egyes tulajdonságok és viselkedésminták gyakrabban fordulnak elő az ügyfelek egyik körénél mint a másiknál. Amit azonosítanunk kell, hogy mik azok a közös pontok

amelyek egyszerre több ügyfélnél is jelen vannak. Ne felejtsük el, hogy az a célunk, hogy egy fiktív személyiségbe sűrítsük az ügyfeleink körét. Ezért ha az összegyűjtött információk alapján egy fajta viselkedés mintázatot mutat 20 fodrász és 3 értékesítő akkor nyugodtan hagyjuk figyelmen kívül a 3 értékesítőtől kapott információkat. Ha az ügyfélkörünk nagy része 25 és 30 év közötti akkor ne vegyük figyelembe a Persona megalkotásakor az ezen az életkori tartományon kívül esők adatait.

Az adatok rendszerezése során nem az a cél, hogy mindenkinek találjunk egy skatulyát amelyikbe beletehetjük, épp ellenkezőleg: azonosítsuk a legnagyobb vagy leginkább fizetőképes azonos tulajdonságokkal rendelkező ügyfélkört és rájuk fókuszálva fejlesszük a termékünket a hatékonyság növelésének érdekében.

Az eredmények megosztása

Miután összegyűjtöttük, rendszereztük és csoportosítottuk az ügyfelkörünkről szóló információkat majd ezek alapján létrehoztuk a Personát osszuk meg az információt a szervezetünkben. Akkor végeztünk jó munkát, ha a Personánk szinte valós személyiségként van jelen a termékkel kapcsolatos döntések meghozatalakor. Ha a termék létrehozásán dolgozó összes csapattagunk felteszi magának a kérdést ha bizonytalan: “Hogyan reagálna erre az új funkcióra a Persona?” vagy “Tetszene-e a Personának, ha megváltoztatnánk ezt a funkciót?”.

A fenti folyamat során létrehozott Persona segít a szervezetnek abban, hogy folyamatosan olyan termékeket és szolgáltatásokat hozzon létre amelyek az ügyfelek véleményét figyelembe véve jöttek létre és így a lehető legnagyobb mértékben kielégítik az igényeiket.

Az ügyfél Personák létrehozását is lehet és ajánlott agilis megközelítéssel létrehozni. Első körben akár még kutatási adatok nélkül vagy meglévő, de nem aktuális kutatási adatokra alapozva létre tudunk hozni egy **Proto Personát.** Ez leginkább a

szervezetünk feltételezett ismereteit foglalja össze az ügyfelekről, de jó kiindulópont lehet, ha később kutatások alapján finomítjuk.

Ügyfélinterjúk, kisebb fókuszcsoportos interjúk alapján jobban megismerhetjük az ügyfél körünket és ezek alapján tovább tudjuk fejleszteni a Proto Personánkat egy **Kvalitatív Personává**.

Ha a kvalitatív kutatásokat kvantitatív kutatásokkal is meg tudjuk erősíteni még pontosabb információt kaphatunk az ügyfélkörünkről és ezek alapján a legreprezentatívabb **Statisztikai Personát** hozhatjuk létre.

Customer Journey Map

Az ügyfél Personák megalkotása során pontos képet kaphatunk arról, hogy **kik** azok akik használják a termékünket. A Customer Journey Map egy olyan eszköz amivel azt tudjuk jól körülhatárolni, hogy **hogyan** teszik ezt. Ha pedig jól ismerjük mind az ügyfeleket mind azok interakcióit a termékünkkel akkor sokkal könnyebben tudunk döntéseket hozni annak továbbfejlesztéséről.

A Customer Journey Map annak ábrázolása ahogyan az ügyfeleink a termékünkkel vagy szolgáltatásunkkal kapcsolatba lépnek. Bemutatja, hogy az találkozás és az érintkezés egyes pontjain milyen interakció zajlik le az ügyfelek és a vállalatunk között.

A Customer Journey Map létrehozása során érdemes a Persona létrehozáshoz hasonlóan a lehető legtöbb adatot összegyűjteni közvetlenül az ügyfelektől. Az adatgyűjtés célja az, hogy minél pontosabban felmérjük az ügyfeleink elvárásait a termékünkkel vagy szogáltatásunkkal kapcsolatban és összehasonlítsuk azzal amit ténylegesen nyújtunk számukra.

Az ügyfelek eltérő csoportjai mutathatnak ugyan hasonló viselkedést a termék használata során, de a használati szokások rendszerint eltérőek az egyes interakciók

alkalmával. Ahhoz, hogy ezt nyomon tudjuk követni és könnyebben át tudjuk tekinteni érdemes az ügyfélkörünket reprezentáló Personáknak külön-külön Customer Journey Map-et készíteni.

A megértésen túl a Customer Journey Map abban segít, hogy az ügyfelek elvárásai és a tényleges működés között fennálló hézagokat megszüntessük, az által, hogy módosítjuk a termékünket. Azaz a Customer Journey Map akkor jó, ha tényleges változásokat eredményez a termékkel kapcsolatban.

Hogyan építhetünk fel egy olyan Customer Journey Map-et amely ellátja ezeket a funkciókat?

Jól körülhatárolt ügyfél Personák és állomások

Az ügyfél Personák fontosságát már kifejtettük korábban. Amikor az ügyfeleink érintkeznek a termékünkkel vagy szolgáltatásunkkal egy jól megfigyelhető és szakaszokra bontható úton haladnak végig a felfedezéstől egészen a lezárásig. Gondoljunk vissza a korábbi telefon előfizetéses példánkra: a böngészéstől kezdve a szolgáltatás megszűntetéséig egyértelmű egymáshoz kapcsolódó szakaszokra lehet osztani a folyamatot ahol az ügyfél más-más pontokon kapcsolódik a termékhez. Amikor saját Customer Journey Mapet hozunk létre a korábbi ügyfél interjúk alapján azonosítanunk kell ezeket a szakaszokat.

Csatornák és érintkezési pontok

Az ügyfelek szinte biztosan nem csak egy csatornán keresztül fognak érintkezni a termékkel vagy szolgáltatással. Elképzelhető, hogy az ügyfél a weben rendel meg egy szolgáltatást, majd az applikációnkon keresztül választ kiegészítő szolgáltatást, ha problémája van akkor telefonon fog segítséget kérni majd egy személyes kirendeltségen fogja megszűntetni a szolgáltatást. A Customer Journey Mapnek minden egyes csatornát tartalmaznia kell ahol az ügyfelek találkozhatnak a

termékkel. Egy jól megtervezett szolgáltatás igénybevétele során az ügyfélélmény ugyanolyan magas szintű függetlenül attól, hogy egy adott szakaszban milyen csatornát választ az ügyfél vagy a folyamat egészén milyen úton halad végig.

Az ügyfelek döntéseinek érzelmi és értelmi háttere

Vegyük figyelembe, hogy az ügyfelek viselkedését nem kizárólag logikus döntések fogják irányítani, hanem az érzelmeik is. Amikor valaki a webshopunkban porszívót keres nem csak azt fogja megvizsgálni, hogy a konkurenciához képest mi mennyiért és milyen garanciával áruljuk a terméket. El fogja olvasni mind a termékkel mind a mi oldalunkkal kapcsolatos véleményeket, hagyatkozni fog korábbi tapasztalataira vagy egyszerűen arra mennyire ismert számára a mi márkánk. Ezeket a szempontokat érdemes megjelenítenünk a Customer Journey Mapen is, hogy jobban megértsük a döntések hátterét és ehhez tudjunk igazodni a kialakított folyamatokkal.

Lehetőségek és fájdalom pontok azonosítása

A jelenlegi helyzet feltérképezésén túl fontos, hogy a Customer Journey Map tartalmazza azokat a lehetőségeket amelyek elérhetővé teszik, hogy még magasabb szintű ügyfél élményt nyújthassunk a piacon. A térkép megmutatja nekünk azt is, hogy az egyes szakaszokban mik a legégetőbb problémák amik elvárásként érkeznek az ügyfelek részéről. Ha az ügyfél interjúk során azt tapasztaljuk, hogy komoly gondot jelent a várakozási idő a telefonos csatornán akkor erre lehet egy lehetőség egy hang alapú azonosítás ami az adategyeztetés idejét csökkentheti.

Felelősségi körök átlátható kijelölése

Miután felvázoltuk a fenti pontok alapján a Customer Journey Mapet tegyük jól láthatóvá a szervezeten belül. Akár ki is nyomtathatjuk és felragaszthatjuk a falra. Minden egyes szakaszban tüntessük fel, hogy a szervezetünk egyes tagjainak vagy

csoportjainak milyen felelőssége van abban, hogy a fájdalom pontok megszűnjenek a lehetőségek megvalósuljanak és összességében az ügyfelek minél jobb élményben részesüljenek a teljes folyamat során.

Mérés

Szeretnénk megállapítani azt, hogy valójában a jó irányba haladunk. Ehhez meg kell tudnunk mérni az egyes szakaszokban az aktuális állapotot és azt, hogy a változtatások milyen eredményt értek el. Ennek eléréséhez meg kell határoznunk azokat a mutatókat amelyek elérhetővé teszik ezt a célt. Ez szervezetenként változó, de vannak olyan széles körben elterjedt metrikák - mint például az Net Promoter Score - amelyek szinte minden terméknél vagy szolgáltatásnál hasznosak. A mérésekről bővebben fogunk beszélni a hatodik alkalom során.

Ha kellően részletes a kialakított ügyfél Personánk és alaposan feltérképeztük azokat a pontokat ahol értintkeznek az ügyfeleink a termékünkkel vagy szolgáltatásunkkal akkor egy komoly előrelépést tettünk abba az irányba, hogy a termékről kialakított víziónkat a gyakorlatba ültessük.

Prototípusok

Az előzőekben körülhatároltuk azt, hogy kinek és hogyan szeretnénk a terméket eladni vagy szolgáltatást nyújtani. Akár bele is kezdhetnénk az új termék létrehozásába vagy egyes funkciók továbbfejlesztésébe. Biztosak lehetünk benne, hogy ez az amire a piacnak szüksége van? Ha magabiztosak vagyunk akkor belevághatunk, de ha szeretnénk csökkenteni a kockázatot vagy szűkösek a rendelkezésre álló erőforrásaink érdemes validálni az ötletünket valódi ügyfelekkel vagy a szervezetünk döntéshozóival. A prototípusok ezt a lehetőséget adják a kezünkbe.

A prototípus készítés során a piacnak szánt megoldásunk egy alacsonyabb összetettségű modelljét készítjük el annak érdekében, hogy visszajelzést kapjunk a működéséről és felmérjük a piaci érdeklődést a termékünk iránt.

Számos módszer létezik arra, hogy ezeket a célokat elérjük egy prototípus elkészítésével, az alábbiakban ezekből ismerhetünk meg néhányat.

Forgatókönyv

Talán a legegyszerűbb és legkevésbé összetett prototípus a forgatókönyv. Ahogy a neve is mutatja néhány egymás mellé rendezett képpel és rövid szövegekkel mutatja be, hogyan használják az ügyfeleink a folyamatot. Egy előre felvázolt forgatókönyvet bemutatva egy ügyfélnek nagy vonalakban feltárhatjuk mi az amit szeretnek és mi az amin változtatnának az ügyfelek a termékünkön. Ha új terméket szeretnénk validálni, egy ügyféllel közös forgatókönyv alkotás során kiderülhet mi az amire ténylegesen vágynak a leendő felhasználók.

Papír prototípus

A digitális termékek létrehozása esetén akár egy kevésbé komplex prototípus előállítása is járhat magas költségekkel. Ha egyszerűen papíron tervezzük meg a felhasználói felületek elrendezését és kinézetét akkor az alapvető funkciókat ugyanúgy tesztelhetjük valódi felhasználókkal, sőt akár egyszerűen átrendezhetjük azt a visszajelzéseknek megfelelően. Ha papír prototípust használunk akkor általában egy specifikus funkció használatát vizsgáljuk meg: egy előre leírt helyzetben kérjük meg a kutatásban részt vevő személyt, hogy mondja el mit hogyan csinálna az adott felületen. Papír alapú modell esetén általában egy drótváz összetettségű modellt szoktunk tesztelni.

Digitális prototípus

A papír módszernél költségesebb cserébe részletesebb megjelenítést lehetővé tevő módszer. A digitális prototípus is lehet drótváz bonyolultságú, de a használt eszközöktől függően itt már lehetőség van arra, hogy képeket, színeket jelenítsünk meg. A digitális prototípusok általában még nem interaktívak, a kutatást végző személy segíti abban a résztvevőket, hogy a megfelelő prototípus változatot mutatja meg nekik akkor amikor egy interakció nyomán valamilyen változás történne. Egy magas összetettségű digitális prototípus esetén már kevés dolog van a prototípust használó fantáziájára bízva, így pontosabb, árnyaltabb képet kaphatunk arról, milyen legyen a megvalósítandó termék

Interaktív digitális prototípus

A valós termékhez legközelebbi prototípus változat az amelyik már képes kezelni az interakciókat. Limitált funkcionalitással, de végig lehet rajta próbálni egy folyamatot, úgy hogy az elágazásoknál vagy a folyamatban előre-hátra mozgásoknál nincs szükség külső beavatkozásra. Természetesen ez azzal jár, hogy mind költségek mind az előállításhoz szükséges idő tekintetében ez lesz a legerőforrásigényesebb cserébe a legpontosabb, a termék szempontjából legrelevánsabb információkat szolgáltató prototípus.

Átalában az olcsó- és gyors prototípus verzióktól érdemes a drágább és összetettebb verziók felé haladni, de érdemes elkerülni azt a csapdát, hogy a prototípusokat csiszoljuk a végtelenségig, ahelyett, hogy a valós terméket mérnénk meg a piacon valós felhasználókkal.

Business Model Canvas és Lean Canvas

A következőkben két olyan eszközzel ismerkedhetünk meg amelyek a teljes üzleti működésünket összefoglalják és megjelenítik egy oldalon. Egy hagyományos üzleti

tervhez képest jelentősen kevesebb részletet tartalmaznak - nem is a működés részletes leírása a céljuk - és a működés és a tapasztalataink alapján érdemes őket újra és újra felülírni és módosítani az előző verziókat. Egyszerűségért cserébe gyorsaságot kapunk: rövid idő alatt fel tudjuk vázolni ezen eszközök segítségével a termékünket és azt a piaci környezetet amelyben teljesítenie kell.

A két eszköz alapvetően abban tér el, hogy a Lean Canvas arra fókuszál, hogy a termékünk milyen problémára ad választ, míg a Business Model Canvas egy konkrét termék értékesítési potenciálját mutatja be.

Business Model Canvas

A Business Model Canvas egy dokumentum sablon amely egy meglévő üzlet tevékenységének a leírására vagy egy új üzleti modell megalkotására szolgál. A létrehozását Alexander Osterwalder nevéhez fűződik aki 2005-ben publikálta először.

Az első Business Model Canvasok startupok számára készültek és ebben az ökoszisztémában hamar ki is szorította ez a típusú vázlatos tervezés az üzleti terveket. Az egy oldalas dokumentum egyszerűsítette és átláthatóvá tette a tervezést, könnyen lehetett reagálni a piaci változásokra a segítségével.

A Business Model Canvas kilenc különböző területet fed le a tervezés során:

- A legfontosabb partnerek (Key partners):

  Kik azok a szervezeten kívüli személyek, cégek akiknek kulcsszerepe van a termék létrejöttében és fejlesztésében, abban hogy értéket teremtsünk az ügyfeleink számára? Ha saját üzleti működésünkben kiemelt szerepe van, hogy Facebook hirdetéseken keresztül generáljunk forgalmat a saját oldalunknak akkor a Facebook kiemelt partnerünkként tűnik majd fel a Business Model Canvasunkon.

- Legfontosabb tevékenységek (Key Activities):

  Mik azok az aktivitások amelyekkel értéket hozunk létre az ügyfeleink számára? Ha például tárhelyet biztosítunk az ügyfeleink képeinek tárolásához, akkor az egyik kulcs tevékenységünk az ehhez szükséges infrastruktúra zavartalan üzemeltetése

- Legfontosabb erőforrások (Key Resources):

  Mik azok a működéshez elengedhetetlen eszközök amelyekkel értéket teremt az üzletünk? Egy digitális termék létrehozásához például szükségünk lesz egy olyan csapatra akik képesek megalkotni és bevezetni a piacra azt: fejlesztők, ügyfélszolgálatosok.

- Értékajánlat (Value Proposition):

  Mi az a probléma amit megoldunk az ügyfeleink számára? Miért lenne arra szüksége az ügyfeleinknek, hogy megoldjuk ezt a problémát? Az Uber értékeajánlata például az “Uber Kényelem”. Röviden összefoglalva: gombnyomásra, kiszámíthatóan és készpénzmentesen nyújtanak személyszállítási szolgáltatást.

- Ügyfélkapcsolatok (Customer relationships):

  Milyen csatornákon keresztül léphetnek kapcsolatba az ügyfelek a termékkel? Egy induló applikációnál bőven elég lehet maga a letölthető alkalmazás mint egyetlen csatorna, de később elképzelhető, hogy webes felületen keresztül is elérhetőek a szolgáltatások. A Revolut például ilyen utat járt be.

- Célközönség (Customer segments):

  Kik lesznek azok akik használni fogják a termékünket? Van-e olyan tulajdonság amely mentén egymástól jelentősen különböző csoportokba

sorolhatjuk őket? A Kahn Academy célközönsége például könnyen csoportokra bontható a foglalkozás alapján: tanárok és diákok vagy az életkor alapján: általános vagy középiskolás diákok.

- Csatornák (Distribution channels):

  Milyen csatornákon keresztül érhetjük el a potenciális ügyfeleket? A híroldalak üzleti stratégiáját szinte teljesen felforgatta a Facebook széles körű elterjedése, hiszen jelentősen függ attól az egyes cikkek elérése - és így a potenciális reklámbevétel - hogy hány ember számára jelenik meg a Facebookon az írás.

- Költségek (Cost):

  Milyen működési költségeink merülnek fel miközben megvalósítjuk az értékajánlatban megfogalmazott céljainkat? Az első 100 ügyfél eléréséig szinte egészen biztosan megfelel a szerződések megkötéséhez egy ingyenesen elérhető sablon. Ha ez a szám eléri a tízezret akkor viszont érdemes lesz igénybe vennünk profi jogi segítséget.

- Bevétel (Revenue Streamers):

  Hogyan fogjuk az értékajánlatban megfogalmazott megoldásunkat fizetőképes keresletté konvertálni? A mobiljátékok piacán a freemium modellt használják bevétel generálásra: a játék ingyen letölthető és használható, de bizonyos előnyöket mikrotranzakciókon keresztül valódi pénzért lehet megvásárolni. A Netflix havi előfizetéses modellel dolgozik. A Wise pedig a használat alapján kalkulál díjat és vonja le azt a tranzakció során.

A Business Model Canvas jó eszköz arra, hogy azonosítsuk a kulcs tevékenységeket amelyek értéket teremtenek és bevételt generálnak, ösztönöz bennünket arra, hogy

felmérjük a stratégiai partnereinket és új kapcsolatokat alakítsunk ki, valamint egy korai fázisban kapunk értékelést arról a saját üzletünk hogyan teljesít majd a piacon.

Hátrányaként érdemes megemlítenünk, hogy nagyon korai működési szakaszban csak alacsony valószínűséggel ad jó válaszokat mivel sok feltételezést tartalmaz. A termék aktuális állapotát leírja ugyan, de nem tartalmaz információt a létrehozásához szükséges lépésekről.

A fenti hiányosságok kezelésére 2010-ben Ash Maurya továbbfejlesztette az eszközt immár Lean Canvas néven.

Lean Canvas

A Lean Canvas célja, hogy a gyakorlatba könnyen átültethető és kivitelezhető üzleti modellt alkothassunk meg a segítségével, így jobban tudjuk kezelni a bizonytalanságot és a kockázatokat.

A Lean Canvas elődjéhez hasonlóan ugyanúgy kilenc területet fed le. Nézzük meg ezeket most egy cég az Uber példáján keresztül. Természetesen arra az időszakra vonatkoznak a példák amikor az Uber még éppen elindult. A lenti példa a Lean Canvas egy lehetséges kitöltési sorrendjét mutatja be, de nincs “egy jó megoldás” arra, hogy milyen sorrendben kell vagy érdemes megalkotni azt.

- Célközönség (Customer segments):

  Könnyen beleeshetett volna az Uber is abba a hibába, hogy az ő termékükre “mindenkinek szüksége van”. Még ha ez így is alakul a későbbiekben érdemes a kulcsfontosságú szereplőkre fókuszálni első körben és később kiegészíteni a modellünket. Az Uber esetében a két fő célcsoportot az olcsó utazásra vágyó jellemzően fiatal városi közönség és a szabadidejükben extra bevételre vágyó autótulajdonosok jelentették.

- Probléma (Problem):

  Az utazni vágyók azzal szembesültek, hogy a kényelem ára a hagyományos taxi szolgáltatóknál a viteli díjak magas mértéke. Ezen kívül nem mindig volt elérhető autó és ha tudtak is visszajelzést adni az utazási élményről az nem garantálta, hogy a következő alkalommal is ugyanolyan vagy jobb minőségű szolgáltatást kapnak. Az USAban az alternatív opciók mint például a tömegközlekedés olcsóságuk ellenére komoly kényelmetlenséggel jártak. Ezekre a problémákra adott választ az Uber.

- Bevétel (Revenue Streams):

  Mivel az Uber csak a szolgáltatás hátterét hozza létre ezért a fuvarok után csak azok díjának egy bizonyos részét fogja saját díjaként elszámolni. A tényleges tranzakciók a sofőr és az utas között zajlanak.

- Megoldás (Solution):

  Az Uber a taxis utazás kényelmét ötvözte az alacsony költségekkel, a korábban említettek alapján gombnyomásra érkezik a hívott jármű és a folyamatosan magas szolgáltatás minőséget az értékelési rendszer biztosítja.

- Értékajánlat (Value proposition):

  Az Uber kényelem koncepciója már az induláskor el lett vetve. Az Uber egy appban gyors és biztonságos utazást kínált a nap huszonnégy órájában, a hét és az év minden napján.

- Csatornák (Channles):

  Az induláskor az Uber két fontos pillérre támaszkodott a kommunikáció terén: a szájhagyományra és az elégedett ügyfelek ajánlásaira. Az utóbbit ingyenes vagy kedvezményes utazásokkal is ösztönözte.

- Legfontosabb mérőszámok (Key Metrics):

  Az Uber indulásakor a növekedés állt a fókuszban. Ennek megfelelően a sikert a regisztrált felhasználók- és a lezárt utazások számában, valamint a befolyt díjakban mérték.

- Költségek (Cost structure):

  Az app nem létezett ezért ennek fejlesztési költsége volt az egyik legjelentősebb költségelem az induláskor. Komoly összeget fordítottak arra is, hogy hatalmas marketing kampánnyal tegyék ismertté a márkát. A folyamatosan növekvő szervezet bérköltsége is jelentős tétel volt a költségvetésben.

- "Igazságtalan" előny (Unfair advantage):

  A sharing economy - a megosztáson alapuló gazdasági modellek - még nem igazán létezett az Uber megjelenése előtt a személyszállítás piacán. A modell olyan versenyelőnyt biztosított az Uber számára, amelyet a versenytársak sem másolni sem megszerezni nem tudtak és egészen a hasonló modellel dolgozó szolgáltatók megjelenéséig szinte egyeduralkodóvá tették az Ubert a saját piacán.

  Összefoglalva a Lean Canvas jobban teljesít abban, hogy egyértelművé teszi a problémát amit az üzlet szeretne megoldani. Komolyabb hangsúlyt fektet arra, hogy egyszerű könnyen validálható koncepciót alkosson. Mérhetővé teszi az elért eredményeket a mérőszámok bevezetésével.

  Amit érdemes figyelembe venni a használatakor, hogy erősen a belső működésre fókuszál a modell és nem veszi figyelembe a külső tényezőket. Az ügyfelek aktuális igényeire fókuszál és nincs benne távlati elképzelés. Az

erőforrások kihagyásával a modellben akár irreális termékötletek is jó színben tűnhetnek fel.

A két modell közül érdemes a saját termékünk vagy szolgáltatásunk életciklusának legmegfelelőbbet választanunk. Mivel viszonylag rövid idő alatt felvázolható mindkettővel egy üzlet piszkozatos formában ezért akár mindkettőt elkészíthetjük a tervezéskor és használhatjuk a saját igényeinkhez jobban illeszkedőt.

Scrum munkaanyagok

Korábban számos eszközt ismertünk meg amelyek segítségével kialakíthatjuk a termékkel kapcsolatos víziónkat. Ezekkel az eszközökkel jól körülhatároltuk azt, hogy kik azok akik majd használni fogják a termékünket és hogy milyen problémát szeretnénk megoldani a leendő felhasználóinknak. Azonosíthattuk azokat a pontokat is ahol a felhasználók majd találkozni fognak a termékünkkel. Ahhoz azonban, hogy ezek a magas szintű tervek és maga a vízió megvalósuljon olyan a gyakorlatban is kivitelezhető elemeket kell létrehoznunk amely alapján a Fejlesztők létre tudják hozni azt a funkció csoportot amely valódi értéket teremt majd a felhasználók számára. Ezek az elemek a Scrum munkaanyagok (artifactok).

» *A Scrum munkaanyagai munkát vagy értéket képviselnek. Arra lettek tervezve, hogy a legfontosabb információk átláthatóságát maximalizálják. Így minden elemzést végző fél ugyanazon alap alapján képes alkalmazkodni.*

Minden munkaanyag részét képezi egy kötelezettségvállalás is. Ezzel biztosítható, hogy az átláthatóságot és a fókuszt növelő információk álljanak rendelkezésre, valamint ezek alapot szolgáltatnak az előrehaladás méréséhez is:

- *A termék Backlog esetében ez a Product Goal.*
- *A Sprint Backlog esetében ez a Sprint Goal.*

- *Az Inkrementum esetében ez a Definition of Done.* «

Product Goal

> *A Product Goal a termék jövőbeni állapotát adja meg. A Scrum Team ennek elérése érdekében állítja össze a terveit. A Product Goal a Product Backlog részét képezi. A Product Backlog többi része folyamatosan alakul ki, ahogy megtaláljuk a választ arra kérdésre, hogy "mit" szükséges kifejleszteni a Product Goal elérése érdekében.*
>
> > *A termék értéknövelő eszköz, egyértelmű keretek között, ismert érdekelt felekkel, jól azonosított végfelhasználókkal vagy ügyfelekkel. A termék lehet szolgáltatás, egy tapintható fizikai termék vagy valami elvontabb.*
>
> *A Product Goal a Scrum Team hosszú távú célkitűzése. Ezt kell megvalósítaniuk (vagy elhagyniuk) mielőtt egy újabb célkitűzést állítanak maguk elé.* «

A Scrum a termék Backlog kötelezettségvállalásaként a Product Goalt határozza meg. Ahogy láthatjuk a Product Goal leírása nagyban hasonlít ahhoz amit az előző alkalom során a termék vízióról megismertünk.

A termék backlog

> *A Termék Backlog a termék fejlesztését szolgáló feladatok folyamatosan alakuló, sorrendezett listája. Ez a Scrum Csapat munkájának egyetlen hiteles forrása.* «

A termék backlog tehát minden olyan feladatot tartalmaz amelyről a Product Owner úgy döntött, hogy értéket teremt. Amikor egy új elem bekerül a termék backlogba a Product Owner elhelyezi a többi elem sorában ezzel egyértelmű prioritást rendel hozzá. Az elemek sorrendje természetesen változhat az idők során, hiszen ezzel lehet alkalmazkodni a piaci körülmények változásához, új technológiák

megjelenéséhez vagy a felhasználói visszajelzésekhez. Mindegyik prioritáshoz azonban csak és kizárólag egy elem tartozhat. A backlogban nem állhat semelyik helyen egyszerre két elem. A Product Owner feladata, kizárólagos joga és felelőssége, hogy a termék backlog sorrendje minden esetben a lehető legnagyobb érték előállítását segítse.

A mindennapi munka során elő fog fordulni, hogy a termék fejlesztésében érdekelt felek a szervezeten belülről egyaránt fontosnak fogják tartani a saját igényeiket. Például a marketing részleg szeretné ha push üzenetet tudna küldeni a felhasználók számára speciális ajánlatokkal, míg a vezérigazgató az applikációnk új designját szeretné látni. A Scrum elég egyértelműen fogalmaz a Product Owner szerepével kapcsolatban ilyen helyzetben:

> » *A Product Owner egyetlen személy, nem egy bizottság. A Product Owner sok érdekelt fél igényeit képviselheti a Product Backlogban. A Product Backlog megváltoztatására a Product Owner meggyőzésén keresztül nyílik lehetőség.* «

Egy ilyen esetben tehát nem az igény megfogalmazójának szervezeti hierarchiában betöltött szerepe vagy informális hatalma dönt arról, hogy a push üzenetek vagy az új design kerül a Fejlesztők asztalára először. A Product Owner az aki kellő felhatalmazással és termék ismerettel rendelkezik ahhoz, hogy meghatározza ezen elemek sorrendjét a backlogban, majd a döntését átláthatóvá tegye a teljes szervezet számára. Ha úgy dönt, hogy a push üzenetekkel nagyobb bevételt tud termelni a termék és az új ajánlatok növelik a felhasználók elégedettségét akkor ezt fogja magasabbra rangsorolni. Ha az új design kényelmesebbé teszi az oldal használatát és még több felhasználót ösztönöz az applikáció használatára akkor pedig ezt fogja priorizálni.

A termék backlog másik sajátossága, hogy az egyes elemei eltérő méretűek - azaz eltérő mértékű idő és tudás ráfordítással lehet őket megvalósítani - és eltérő részletességű leírást tartalmaznak arról, hogy mi is a mögöttük rejlő pontos igény.

Amikor a termék víziót elkezdjük lebontani kivitelezhető egységekre akkor először mindig a nagyobb egységek születnek meg először. Vegyünk most példának egy szakmai hírekkel foglalkozó portált. Ha egy ilyen terméket szeretnénk megvalósítani akkor szükségünk lesz arra, hogy az olvasók el tudják olvasni azokat a híreket amelyeket megjelenítünk. Kell egy olyan felület ahol a cikkek szerzői létre tudják hozni az olvasók számára a tartalmat. Ha a portál bevételét nagyrészt reklámok adják akkor a reklámkampányok anyagait is kezelni kell.

Máris van három, igencsak nagy méretű, de már kézzelfogható backlog elemünk amely mentén elkezdődhet a fejlesztés. Persze ha ezekkel az igényekkel állunk a Fejlesztők elé, akkor jó eseteben rengeteg kérdést, rosszabb esetben keresetlen szavakat kapunk tőlük. A fenti sorrendet követve az olvasói felület kapcsán el kell döntenünk olyan kérdéseket, hogy “Hány hír jelenjen meg egyszerre az oldalon?”, “Legyenek-e kiemelt hírek?”, “Akarunk-e külön rovatokat és eszerinti csoportosítást?” és így tovább. Ezen kérdések nem csak az olvasói, de a szerkesztői felület vagy a kampány menedzsment felület kapcsán is fel fognak merülni.

Ha a Product Owner jól végzi a munkáját akkor már a fejlesztési folyamat elején és onnan folyamatosan bevonja a Fejlesztőket a backlog finomításába, kisebb egységekre történő bontásába. Ez azért is hasznos mert így a Fejlesztők is beleszólást kapnak a termék kialakításába amely számukra is motivációt ad, valamint számos olyan kérdést feltehetnek amely új backlog elemek létrehozására ösztönzi majd a Product Ownert.

Miután az egyes backlog elemeket kisebb részekre osztottuk érdemes újra megvizsgálni a kisebb, jobban definiált elemek sorrendjét. Egy kezdetleges

hírportálnál például elég ha formázás nélkül megjelennek a cikkek, de már meg tudunk jeleníteni egyszerű kép formátumú hirdetéseket az egyes cikkek alatt.

A Minimum Viable Product (MVP)

Egy új termék indulásakor komoly kihívást jelent, hol húzzuk meg azt a határt ahol a termékünk már elég funkciót tartalmaz, amellyel be tudjuk vonzani a korai felhasználókat és validálni tudjuk a koncepciónkat a piacon. Ezt a már piaci környezetben is életképes, de a funkciók nagy részét még nem tartalmazó terméket nevezzük Minimum Viable Productnak vagy MVP-nek. Az MVP koncepciója párhuzamban áll a Scrum működési elveivel: minél hamarabb és minél gyakrabban piacra lépni a termékünkkel. Az MVP egy hírportál esetén egy egészen lecsupaszított funkciókkal működő termék is lehet. Ha egymás alatt megjelennek a híreink - akár úgy hogy a tartalmakat formázás nélkül közvetlenül a kódhoz adjuk hozzá - akkor már van egy olyan termékünk amelyet oda tudunk adni valódi felhasználóknak, hogy visszajelzést adjanak rá. Egy ilyen termék persze hosszú távon fenntarthatatlan lenne ezért a hírportál szerzői biztosan jelezni fogják, hogy szeretnének egy olyan felületet ahol közvetlenül le tudják írni és publikálni tudják a cikkeiket. Egy következő sprintben már erre az igényre fókuszálhatunk, persze ha nem lesz fontosabb, hogy képek is legyenek a cikkekben, mert úgy szívesebben olvassák az olvasók.

Az MVP egyik legnagyobb előnye, hogy rámutat a fejlesztendő backlog elemek valódi értékére és fontosságára és ezen keresztül segíti a megfelelő prioritások felállítását a termék backlogban a fejlesztés kezdeti szakaszában.

Agilis Roadmap

A Scrum és általánosságban az agilis megközelítés alapvetően rövid egymást követő időtávlatokba zárja be a komplex és gyorsan változó valóságot. Ez azonban korántsem jelenti azt, hogy ne lehetne hosszú távú terveket felállítani a termék fejlesztésével kapcsolatban. Ha folyamatosan kapcsolatban vagyunk a termék

felhasználóival és figyeljük mik azok a hiányzó funkciók amelyekre vágynak, vagy mik azok a funkciók amelyeket a legkényelmetlenebb használni akkor ehhez tudjuk igazítani a fejlesztés irányát is. Ehhez mérten fel tudjuk vázolni azokat a nagyobb egységeket a termék fejlesztésével kapcsolatban amelyeket egy távolabbi jövőben szeretnénk megvalósítani.

Ezt a magas szintű dokumentumot amely tartalmazza a termék vízió megalkotásához szükséges stratégiai lépéseket nevezzük roadmapnek (ütemtervnek).

Az áttekinthetőségen túl ez az eszköz lehetőséget ad arra is, hogy a szervezet számára transzparensen kommunikáljuk a termék víziót és megvalósításának lépéseit. A létrehozásakor mindig vegyük figyelembe ezt a célt és tartsuk meg a részletességét magas szinten. A roadmap ne tartalmazza a megvalósítás részleteit csak a stratégiai szempontból fontos elemeket. A hírportálunk esetében ha szeretnénk bevezetni, hogy az olvasók kommentelni tudjanak az egyes cikkek alatt akkor a roadmapen ennyi információ bőven elég erről a funkcióról. Az, hogy mekkora terjedelmű lehet egy komment, lehet-e a másik felhasználó kommentjére reakciót tenni vagy szükséges-e a moderáció a kommentek között már nem kell, hogy része legyen egy agilis roadmapnek.

Az agilis roadmap nem állandó - nincs kőbe vésve. Ha változik a felhasználók igénye akkor a roadmapen felvázolt sorrendet is változtathatjuk. A rugalmasság és a magas szinten definiált elemek miatt nem ajánlott hosszú időtávlatra felvázolni a roadmapet, sőt általánosságban óvatosan kell időtávlatot rendelni egy-egy elem mellé, mivel azzal olyan elvárásokat támaszthatunk a termékkel szemben amit később nem fogunk tudni teljesíteni.

A roadmap egyes elemei között lehetnek függőségek. Ez fakadhat a felhasznált technológiából vagy egyszerűen abból, hogy bizonyos funkciók önállóan nem

értelmezhetőek csak valamilyen másik funkció megléte után adnak értéket a felhasználóknak. A hírportálunk esetén a hirdetések szerkesztésére alkalmas felület csak azután valósulhat meg, miután már legalább egy típusú hirdetést meg tudunk jeleníteni a cikkekben.

Az hogy milyen időtávlatban alakítjuk ki a roadmapet nagyban függ a terméktől, annak életciklusától és a szervezetünk rugalmasságától is. Egy életciklusának korai szakaszában lévő termék esetén felesleges túl hosszú időtávra terveznünk, hiszen nagyon kevés információ áll rendelkezésünkre a felhasználókról és arról hogyan is használják a termékünket. Egy érett, a felhasználók által évek óta használt termék esetén azonban lehetünk megengedőbbek és egy-egy új funkció megvalósításának terve szükség esetén túlnyúlhat egy negyedéves cikluson is.

Ha a szervezet kevésbé alkalmazkodott az agilus működéshez akkor szintén érdemes rövidebb időtávlatokra felvázolni a roadmapet és folyamatosan kommunikálni azt, hogy rugalmasan változni fog az idő előrehaladtával. Ezzel megakadályozhatjuk azt, hogy a roadmapen szereplő tervet ígéretként kezelje a szervezetünk és lehetetlen elvárásoknak kelljen megfelelni a fejlesztés során. Hívjuk fel továbbá a figyelmet, hogy a roadmapen szereplő elemek még csak nagyvonalú becslést sem tartalmaznak arra vonatkozóan, hogy mekkora ráfordítást igényel majd a megvalósításuk.

A roadmap összefoglalva egy kiváló eszköz lehet arra, hogy a termékvízió megvalósítását nagyobb stratégiai elemekre bontva jelenítsük meg és átláthatóvá tegyük a szervezetünk számára.

Saga, Epic, User story

A termék backlog létrehozásakor már tárgyaltuk azt, hogy a backlogban szereplő elemek eltérő méretűek és részletességűek lehetnek. A tényleges megvalósíthatósághoz időben legközelebb eső elemek ideális esetben a

legrészletesebbek és legkisebbek, míg a távolabbiak kevesebb részletet tartalmaznak és funkciók nagyobb körét ölelik fel. A Scrum csupán azt határozza meg, hogy a termék backlog elemekből áll és ezeket a Scrum csapat folyamatosan finomítja:

> » Azon termék Backlog elemek, melyek egy Sprinten belül elvégezhetőek a Scrum Team által, készen álltnak tekinthetőek a Sprint Planningen történő kiválasztásra. Ezt az átláthatóságot általában finomítási tevékenységek során érik el. A termék Backlog finomítása a termék Backlog elemeinek lebontása és tovább definiálása kisebb, pontosabb tételekre. Ez egy folyamatos tevékenység, amely részletekkel, például leírással, sorrendezéssel és méretek megadásával bővíti az elemeket. Ezek az attribútumok a munka területétől függően változhatnak. «

A legelemibb ilyen termék Backlog elemek leírásának egyik módszere a User Storyk létrehozása. A User Story alkotás nem része a Scrum keretrendszernek, de egy jó kiegészítője lehet annak.

A User Story nem más mint egy a termékkel kapcsolatos igény megfogalmazása a felhasználó szemszögéből. A legnépszerűbb és legegyszerűbb formátum alapján a User Story felépítése a következő:

> {Persona}ként
>
> Azt akarom, hogy {a várt termékfunkció leírása}
>
> Azért, hogy {az érték amit adni fog nekem ez a funkció}

Felmerülhet a kérdés, hogy miért van szükség arra, hogy ilyen körülményes formában írjunk le minden új funkciót? Előfordulhat, hogy valami nagyon triviális dolgot akarunk megvalósítani, akkor rengeteg időt vesz igénybe csak a leírás

létrehozása. Ha csak annyit szeretnék a termékben látni, hogy a felhasználó be tudjon jelentkezni egy adott felületen az mindenki számára egyértelmű, nem?

Két okból is érdemes azonban ragaszkodni a fenti formátumhoz.

Egyrészről kapcsolatot hozunk létre a funkcióhoz kapcsolódó részletek között. Meghatározzuk ki a felhasználó aki számára értéket termet majd a funkció, mi lesz konkrétan az ami megteremti majd ezt az értéket és leírjuk azt is miért szeretnénk ezt az értéket megteremteni. A felhasználó jelenléte a User Storyban segíti a Fejlesztőket abban, hogy azonosulni tudjanak a megoldandó problémával, a miért pedig megadja a lehetőséget arra, hogy kellő szabadságot kapjanak a funkció megvalósításában.

Másrészről a "lehessen bejelentkezni" jellegű termék Backlog elemek nem nyújtanak elég információt egy új csapattag részére, sőt ha valamilyen okból vissza kell tekinteni egy korábbi User Storyra akkor szintén kevésbé lesz érthető, hogy miért hoztunk létre egy adott funkciót.

A fenti formátumot a három rész megtartásával szabadon bővíthetjük, hogy még pontosabb képet tudjunk adni a Fejleszőknek arról mi is az amivel bővíteni szeretnénk a terméket.

Példaként vegyünk most egy csomagszállítással foglalkozó vállalkozást és annak a feladó számára létrehozott felületét. Szeretnénk ha a feladók le tudnák kérdezni a feladott csomagjait állapotát. Egy ilyen esetben a User Story a már ismert formátumban így nézhet ki:

> **Feladó**ként
>
> **Azt akarom**, hogy lássam az elküldött csomagjaim állapotát
>
> **Azért hogy**, tájékoztathassam a címzettet a várható szállítási időről

Ezt a user storyt használva több megoldás is szóba jöhet. A feladónk számára az jelenti az értéket, hogy a címzett és ő is értesül a szállítás várható időpontjáról. Csinálhatunk egy egyszerű listát az összes feladott csomagról ahol a feladó megkeresheti az aktuáliast és saját maga értesítheti a címzettet a várható időről.

Pontosítsuk most egy új hozzáadott elemmel a User Storyt.

> **Feladó**ként **aki** most adott fel egy új csomagot
>
> **Azt akarom**, hogy lássam az elküldött csomagjaim állapotát
>
> **Azért hogy**, tájékoztathassam a címzettet a várható szállítási időről

Ebben a megfogalmazásban szűkítettük a megoldási lehetőségeket, hiszen egy adott csomag - az épp most feladott - szállítási időpontját szeretnénk látni. Ez történhet a feladás sikerességét visszaigazoló képernyőn vagy akár egyszerűen úgy, hogy időrendi sorrendben jelenítjük meg a listát.

Ahogyan a fenti példából is látjuk az, hogy az érték hogyan fog előállni, nincs meghatározva az User Storyban. Ez lehetőséget ad arra, hogy a Fejlesztők még a megvalósítás előtt több megoldási javaslatot tegyenek és a felhasználók a lehető legjobbat kapják majd kézhez ezek közül. Az hogy ténylegesen mennyi értéket adtunk persze továbbra is a felhasználók visszajelzéseiből fog látszani.

Attól függően, hogy milyen komplexitású megoldásra váró problémát fogalmazunk meg egy User Storyban elképzelhető, hogy nem lesz megvalósítható egy Sprintem belül. Némi irodalmi párhuzamot behozva egy-egy User Storyt úgy is elképzelhetünk mint egy bekezdést vagy egy fejezetet egy nagyobb műből. Ha ennek a műnek az egyik kötetét vesszük akkor az már egy hőskölteményt (Epic). Ha pedig kötetek során rágjuk át magunkat, mondjuk a viking mitológiát végigjárva akkor pedig egy egész Sagaról beszélhetünk. Mindegyik egység önmagában is értelmezhető, közösen pedig egy nagy egységet alkotnak.

Ezt a párhuzamot a backlog elemek létrehozásakor is szokták használni általában úgy, hogy egy nagyobb egységet bont le a Product Owner a Fejlesztők bevonásával kisebb, alacsonyabb komplexitású és lehetőség szerint egy Sprinten belül megvalósítható elemekre. A nagy átfogó elemek - például a felhasználói adatkezelés - lesz a Saga. Ennek egyik fontos eleme a bejelentkezés már egy kisebb Epic szintű elem. Ha pedig ezt még tovább bontjuk akkor a felhasználói fiók megjegyzése egy adott eszközön már lehet egy User Story.

Bárhogyan is nevezzük azonban a backlog elemeit azok alapvetően méretükben és komplexitásukban különböznek csak egymástól azonban közös tulajdonságuk, hogy a felhasználó szemszögéből íródnak, megfogalmazzák, hogy mivel és hogyan teremtenek értéket. Akár a User Story akár más formátumot használunk a backlog elemek leírására érdemes egy adott termékhez kapcsolódóan egy fajta módszert használni.

INVEST

A fentiekben megismerhettük a User Storyk formátumát és alapvető tulajdonságait. Egészítsük ezt most ki egy olyan módszerrel amely alkalmazásával még jobb User Storykat hozhatunk létre. Az INVEST egy betűszó és minden egyes eleme a kiváló User Story egy-egy tulajdonságát mutatja be. Nézzük meg ezeket egyesével példákon keresztül.

Független (**I**ndependent)

A User Story függetlensége azt jelenti, hogy nincs átfedésben másik User Storyval. Függőségek természetesen lehetnek egyes User Storyk között például a regisztráció nélkül értelmetlen a felhasználó profil oldalt létrehozni. Az egymással átfedésben lévő storyra akkor találunk példát, ha nem keresztfunkcionális csapat dolgozik a megvalósításon. Maradva a regisztrációs folyamatnál szükséges ehhez nagyvonalakban szükségünk lesz egy felületre ahol a felhasználók megadják az

adataikat és egy adatbázisra ahol ezeket tároljuk. Ha ezen két elemet külön-külön backlog elemként hozzuk létre akkor azok nem csak kölcsönösen függnek majd egymástól, de nagy valószínűséggel a Sprint végén önállóan kevés vagy semennyi értéket nem fognak képviselni. Az ilyen horizontális típusú backlog finomítást, amikor az egyes backlog elemeket nem önállóan is használható egységekre hanem a technológia mentén bontjuk kisebb részekre érdemes a gyakorlatban elkerülni, mert lassítja a piacra lépést és feleslegesen pazarolja a szűkös erőforrásokat.

Tárgyalható (**N**egotiable)

Amíg a Product Owner azt mondja meg, hogy mi az amit megvalósítsunk addig a Fejlesztők azt, hogy hogyan. Egy User Story akkor működik jól, ha teret ad a kreativitásnak és arra ösztönzi a Fejlesztőket, hogy több választ is adjanak a felhasználók problémáira. Vegyünk először egy olyan példát ami több elemében ellentmond ennek így kerülendő:

> Product Ownerként
>
> Azt akarom, hogy az új felhasználók végignézzenek egy videót a regisztráció után
>
> Azért hogy, a marketing részleg videóba fektetett munkája értelmet nyerjen

A fenti - szinte minden elemében rossz - User Story kiválóan példázza azt, hogy a formátum megtartásával, de a szellemiség semmibe vételével milyen zombit lehet alkotni. Nézzük sorban a hibákat. Nem a felhasználó szemszögéből van megfogalmazva, hanem egy utasítást fogalmaz meg, pedig a Product Ownernek nincs ilyen felhatalmazása. Pontosan leírja hogyan kellene megoldani a problémát: vetítsük le a felhasználónak a videót - nincs lehetőség más megoldásra. A miértre adott válasz pedig szinte biztos, hogy a Product Ownerre gyakorolt nyomás hatására jön létre.

Nézzük meg, hogyan lehetne ezt jól átírni.

> Új felhasználóként aki most fejezte be a regisztrációt
>
> Azt akarom, hogy részletes információt kapjak a termék működéséről
>
> Azért, hogy könnyen megtehessem az első lépéseket

Ez a User Story a felhasználó szempontjából mutatja be a problémát. Az, hogy nem írja le a megoldást, csak azt hogy a felhasználónak több információra van szüksége nyitva hagyja a lehetőséget arra, hogy a Product Owner és a Fejlesztők közösen találjanak megoldást, ami lehet egy videó is, de egy szöveges dokumentum vagy képsorozat esetleg egy tutorial is. Az első lépések mint a probléma fókusza pedig a segítség időbeli pontját és komplexitását is körbeírja, de nem határolja le teljesen ezért újabb beszélgetést fog igényelni.

Értékteremtő (**V**aluable)

A termék Backlog feladata, hogy garantálja azt, hogy a felhasználók mindeg egyes Sprint után a lehető legnagyobb értéket kapják meg. A backlog elemeit ezért mindig úgy kell létrehozni, hogy az elemek által teremtett érték egyértelmű legyen. A User Story formátuma lehetővé teszi azt, hogy az “Azért, hogy…” részben teret adjunk annak, hogy miért is fog értéket teremteni a storyban megfogalmazott funkció. Gyorsabban tudja használni az applikációnkat a felhasználó? Kényelmesebben? Egy régen fennálló problémájára adunk választ? Bármelyik fenti kérdésre is felelünk igennel az értéket fog teremteni a felhasználónak. Az érték megfogalmazása a user storyban pedig segít a Fejlesztőknek, hogy a munkájuk során lássák miért dolgoznak és közel hozzuk hozzájuk a felhasználót.

Becsülhető (**E**stimable)

A User Storyk azért születnek, hogy alapul szolgáljanak a későbbi fejlesztésnek. A Fejlesztők a munka elkezdése előtt megpróbálják hozzávetőlegesen megbecsülni, hogy egy adott Sprint alatt mennyi és milyen komplexitású feladatot tudnak elvégezni, hogy előálljon az egyes User Storykban megfogalmazott érték. A becsléshez arra van szükség, hogy az egyes storyk jól körülhatárolható igényeket fogalmazzanak meg. A becsülhetőséghez a Fejlesztők számtalan kérdést és javaslatot fogalmaznak meg a Product Owner felé és ebből születik meg a pontos igény majd az ehhez kapcsolódó becslés.

Ennek részleteiről és a becslési technikákról a következő alkalommal lesz szó.

Kicsi (**S**mall)

Az, hogy egy User Story kicsi nem arra utal, hogy milyen hosszú vagy részletes leírást tartalmaz a szövege. Ideális esetben a User Story nem nagyobb annál, hogy egy teljes Sprintet igénybe vegyen a megvalósítása. Már akkor is érdemes elgondolkodni azon, hogy tudjuk-e kisebb részekre bontani a storyt ha önmagában egy teljes Sprint szükséges ahhoz, hogy létrehozzuk a benne megfogalmazott értéket, de ha ennél is több idő szükséges biztos, hogy további finomításra van szükség.

Fontos megjegyezni, hogy egy User Story méretét nem a Product Owner, főleg nem a szervezeten belüli más stakeholder mondja meg, csak és kizárólag a Fejlesztők. Az ezzel ellentétes működés egy olyan negatív spirált tud kialakítani ahol becslések helyett elvárások fogalmazódnak meg a csapat felé, akiknek csökken a motivációja és még kevesebb értéket tudnak előállítani aminek hatására a stakeholderek még komolyabb elvárásokat fogalmaznak meg és így tovább. Ez persze nem jelenti azt, hogy nem elvárható a fejlődés a Fejlesztők részéről, de egy magasan motivált, kellő felhatalmazással rendelkező és egy komolyabb termék ismerettel rendelkező csapat ezt automatikusan szállítani fogja.

Tesztelhető (**T**estable)

Nem elég ha értéket állítunk elő az User Story megvalósításával, de úgy kell ezt tennünk, hogy az megfeleljen bizonyos minőségi kritériumoknak is. Ezeket a minőségi kritériumokat a Fejlesztők állítják fel a saját munkájukkal szemben. A tesztek elvégzése szintén az ő feladatuk és ez a tevékenység része a Sprintben folyó munkának. A Product Owner feladata ebben az, hogy a Fejlesztőkkel közösen megfogalmazza azokat az üzleti szempontból releváns elvárásokat amelyek mentén elfogadja a Scrum Csapat, hogy egy User Story értéket teremtett.

Vegyünk példának egy repülőjegy vásárlással kapcsolatos termék backlogjából egy User Storyt.

> Utazóként aki kiválasztotta a jegyeit
>
> Azt akarom, hogy láthassam a kiválasztott jegyeimet a fizetés előtt
>
> Azért, hogy módosíthassak rajtuk szükség esetén.

Ehhez a User Storyhoz a lentihez hasonló acceptance criteria tartozhat:

- A jegyekre kattintva a kiválasztott jegyek egy új ablakban nyílnak meg
- Az új ablakban látszik a kiválasztott ülőhely
- A felhasználó tudja módosítani az ülőhelyet
- A jegyek összértéke látszik a felhasználó számára
- Az utazás időpontja látszik a felhasználó számára
- A fizetést választva a felhasználó kiválaszthatja az elérhető fizetési lehetőségeket
- A fizetési opciót kiválasztva átirányítjuk a felhasználót a banki felületre

A fenti lista nem teljes, de ennyi részlet mentén a Fejlesztők jól körül tudják határolni azokat a feladatokat amelyeket el kell végezniük, hogy a leírás alapján

működő funkció jöjjön létre és vissza is tudják ellenőrizni, hogy a munkájuk megfelel-e az összes üzleti kritériumnak.

Definition of Ready

Megismerhettük, hogy az egyes backlog elemeket hogyan érdemes felépíteni. Az Agilis Kiáltvány alapján a személyeket és a köztük létrejött interakciót nagyobbra értékeljük a módszertanokkal és az eszközökkel szemben. Ennek szellemében előfordulhat, hogy a Fejlesztők megfogalmaznak olyan elvárásokat az egyes backlog elemekkel szemben, amely megkönnyítheti a közös munkát. Az átláthatóság jegyében ezt érdemes mindenki számára elérhető módon dokumentálni. Ezt a dokumentumot fogjuk Definition of Readynek nevezni.

Egy ilyen dokumentum tartalma nem előre rögzített, hiszen egy párbeszédből fog kialakulni, de néhány elemet érdemes megfontolni a megalkotása során:

- Fejlesztésre kész egy backlog elem, ha minden Fejlesztő kellően jól érti milyen értéket hoz létre és ez a megértés közös a Product Ownerrel
- Tartalmazza a technikai specifikáció megalkotásához szükséges részleteket (például egy felület drótvázát)
- Megfelel a korábban bemutatott INVEST tulajdonságoknak
- Nincs átfedésben más backlog elemmel

Ha közös megállapodás alapján születik meg a Definition of Ready akkor a Fejlesztők mondhatják azt, hogy addig nem tudnak egy User Story megvalósításán dolgozni, amíg az nem felel meg a Definition of Readynek. Fontos azonban az egyensúly: ez nem egy fegyver, hanem a közös megértést segítő eszköz. Akár a Product Owner hanyagolja el a termék Backlog elemeinek kezelését, akár a Fejlesztők nem mozdítják a kisujjukat sem amíg pontos specifikációt nem kapnak azt jelzi, hogy a Scrum csapat letért az útról. Egy jó Scrum Master felismeri az ilyen helyzeteket és segít elmozdítani ezeket az akadályokat.

A Sprint Backlog

> *» A Sprint Backlog a Sprint Goalból (miért csináljuk), a termék Backlog Sprintre kiválasztott elemeiből (mit csinálunk) és a Sprint Goalt megvalósító Inkrementum elkészítése érdekében összeállított tervből (hogyan csináljuk) áll.*
>
> *A Sprint Backlog egy terv; a Fejleszők által, a Fejlesztőknek. A Fejlesztők a Sprint Goal*
>
> *érdekében a Sprint során elvégzendő munkájának kiemelt helyen közzétett, valós idejű állapota. Azaz ahogy a megszerzett tudás növekszik, a Sprint Backlog folyamatosan frissül a Sprint során. Elég részletes kell legyen ahhoz, hogy a Developerek vizsgálni tudják előrehaladásukat a Daily Scrum során. «*

A Sprint Backlog lényegében a jéghegy csúcsa, amely éppen kilátszik a vízből. Miután a Product Owner meghatározta, hogy melyek azok az elemek a backlogból amelyek a legnagyobb értéket képviselik és ez alapján sorba rendezte őket a Fejlesztők feladata, hogy kiválasszák azokat az elemeket a termék Backlog tetejéről amelyeket egy Sprint alatt meg tudnak valósítani és elkészítsék a megvalósításhoz szükséges tervet valamint meghatározzák azokat a technikai feladatokat amelyeket el kell végezni a megvalósításhoz. A Sprint Backlog része a Sprint Goal azaz a sprint célja is.

> *» A Sprint Goal a Sprint egyetlen célkitűzése. Bár a Sprint cél a Developerek kötelezettségvállalása, elég rugalmassággal rendelkezik az eléréséhez szükséges munka tekintetében. A Sprint Goal koherenciát és fókuszt teremt, arra ösztönzi a Scrum Teamet, hogy az egyéni kezdeményezések helyett a közös munkát részesítsék előnyben. «*

A Sprint Goal a Product Goalhoz hasonlóan egy iránymutatás de az értelmezési tartománya időben csak a Sprint idejére terjed ki. Segít továbbá abban, hogy ha

bármilyen okból változna a Sprint Backlogban szereplő termék Backlog elemek összetétele a Sprint során akkor az úgy történjen, hogy az elérendő célt tartsuk szem előtt.

SMART

A Sprint Goal megalkotásához használt egyik módszer a szintén betűszóként ismert SMART. Mitől is lesz okos egy kitűzött cél?

Konkrét (**S**pecific)

Határozzuk meg úgy a Sprint célját, hogy az egy probléma jól definiált megoldását jelentse. Egy előfizetéses szolgáltatást kínáló termék esetén a Sprint Goal lehet a következő:

> Csökkentsük a lemondások számát azzal, hogy összegyűjtjük a szolgáltatást lemondó felhasználók visszajelzéseit

Ez a cél elég konkrétumot tartalmaz ahhoz, hogy fókuszt adjon a fejlesztésnek, de kellő szabadságot ahhoz, hogy a célok eléréséhez használt eszközöket önállóan válassza ki a csapat.

Mérhető (**M**easurable)

Honnan fogjuk tudni a Sprint végén, hogy ténylegesen értéket állítottunk elő? Ha meg tudjuk mérni az új funkciók hatását akkor meggyőződhetünk arról, hogy ténylegesen jó irányba haladunk-e. Ezek alapján a korábbi Sprint Goalt módosíthatjuk úgy, hogy számszerűsítjük az elvárt eredményt:

> Csökkentsük a lemondások számát 20% azzal, hogy összegyűjtjük a szolgáltatást lemondó felhasználók visszajelzéseit

Ha létrehoztuk a visszajelzéseket összegyűjtő megoldásunkat és a visszajelzések alapján alakítunk a termék árazásán akkor megnézhetjük, hogy sikerült-e elérni a kitűzött 20%-os célt a lemondások csökkentése kapcsán.

Elérehtő (**A**chievable)

A termék vízióhoz hasonlóan a Sprint Goal is ambíciózus, de megvalósítható részleteket kell, hogy tartalmazzon. Szinte kizárt, hogy egy mozi számára készült applikáció regisztrációs és értékesítési felületét egyszerre el tudjuk készíteni egy Sprint alatt.

> A néző saját maga választhassa ki a széket ahonnan nézni szeretné a filmet.

Ez viszont egy kihívást jelentő, de rövid idő alatt megvalósítható cél lehet az alkalmazásunkban.

Fontos (**R**elevant)

A fontosságot az fogja meghatározni, hogy egyrészt összefüggésben áll-e a Sprint kitűzött célja a termék víziójával és céljaival valamint segít-e elérni a szervezet üzleti céljait.

> Növeljük a kezdő oldalra látogatók fizető ügyféllé konvertálását 15%-al.

A fenti cél üzleti eredményeket tartalmaz (fizető ügyfél konverzió) és a felhasználók számának növelésével támogatja a vízió megvalósulását.

Időhöz kötött (**T**ime-bound)

A Sprint Goal megvalósítása mindig az adott Sprinthez, így annak időkeretéhez kötött. Mivel rövid időszakról beszélünk ezért könnyebb úgy meghatározni a Sprint célját, hogy az elérhető legyen.

> Tegyük elérhetővé a PayPal fizetési módot az ügyfelek számára.

Egy konkrét fizetési mód hozzáadása a termékhez jó esetben rövid idő alatt is elérhető. Ha ennél többet szeretnénk látni akkor érdemes több Sprinten átívelően megvalósítani a célunkat.

A Sprint Goal, a kiválasztott termék Backlog elemek és a Sprint terv tehát együttesen alkotják a Sprint Backlogot amely alapján létrejön majd a Sprint végén a működő funkció.

Az Inkrementum

> *» Egy Inkrementum egy szilárd lépcsőfok a Product Goal irányában. Minden Inkrementum ráépül az összes megelőzőre, valamint részletesen ellenőrzött, biztosítván az Inkrementumok együttműködését. Annak érdekében, hogy értéket jelentsen, az Inkrementumnak használhatónak kell lennie.*
>
> *A Sprinten belül több Inkrementum is létrehozható. Az empirikus gondolkodás jegyében az Inkrementumok összességét a Sprint Review-n mutatjuk be. Ugyanakkor, egy Inkrementum a Sprint vége előtt is leszállítható az érdekelt felek számára. A Sprint Reviewt ne tekintsük kapunak az értékszállítás folyamatában.*
>
> *A Definition of Done-nak nem megfelelő munkát nem tekinthetjük egy Inkrementum részének. «*

Az Inkrementum egy működő termékfunkció ami a Sprint végén szerves részévé válik a terméknek és megfelel az összes olyan minőségi kritériumnak amelyet előzetesen a Fejlesztők meghatároztak.

Az ehhez kapcsolódó feladatok mindegyike a Sprint során valósul meg. Azaz, ha egy új funkciót adunk a termékhez - például bevezetünk egy új fizetési megoldást - akkor az ehhez kapcsolódó integrációs és tesztelési feladatokat a Sprinten belül kell megvalósítani. Nem feltétlenül szükséges, hogy minden Sprint végén ez elérhető

legyen a felhasználók számára is, de úgy kell, hogy elkészüljön, hogy ha a szervezet emelett dönt akkor ez megvalósítható legyen.

Ez akkor is így van, ha több csapat dolgozik a termék megvalósításán.

Definition of Done

> » *A Definition of Done az Inkrementum állapotának formális leírása abban az időpillanatban, amikor megfelel a termék számára előírt minőségi kritériumoknak.*
>
> *Abban a pillanatban, ahogy a Product Backlog egy eleme megfelel a Definition of Done-nak, egy új Inkrementum áll elő.* «

A Definition of Done segítségével határozzák meg a Fejlesztők azokat a saját maguk felé támasztott minőségi elvárásokat, amelyek mentén garantálják, hogy folyamatosan magas minőségű terméket adjanak ki a kezükből és törekedjenek a minőségi elvárások növelésére.

A Definition of Done tartalmazhat a kódolás sztenderdjeire vonatkozó elvárásokat, az elkészült kód ellenőrzésének folyamatát, a tesztek szintjeinek meghatározását és az automatizálásuk arányát, a dokumentáció folyamatát és elvárásait vagy előírásokat a felhasználható erőforrások kapcsán.

A kialakított Definition of Done kötelező elvárás a Fejlesztők munkája felé, akkor is, ha saját maguk alakították ki a szabályokat, és akkor is ha szervezeti szinten történt a meghatározás. Egy termékkel kapcsolatban egy Definition of Done határozható meg amelyet az összes a terméken dolgozó csapatnak be kell tartania.

Scrum események

Korábban létrehoztuk a termék víziót és legalább magas szinten megalkottunk egy termék backlogot amely alapján elindulhat a fejlesztési folyamat. A következőkben megismerkedünk a Product owner számára fontos Scrum eseményekkel miközben végigkísérjük egy Scrum csapat hétköznapjait. Lássuk milyen események történnek velük a Sprint során.

Backlog refinement

Időpont: szerda 15:00 - az előző Sprint zárása előtt két nappal. A Fejlesztők és a Product Owner a nap zárásaként egy backlog refinement megbeszélést tervezett. A találkozón részt vesz a Scrum Master is, hogy elősegítse a hatékony kommunikációt. Az összejövetel előtt a Product Owner naprakész állapotba hozta a backlogot és elérhetővé tette a Fejlesztők számára. A Fejlesztők is felkészülten érkeznek: előzetesen átnézték a backlog magas prioritású elemeit és felkészültek azokkal a kérdésekkel amelyekkel jobban megérthetik az üzleti igényt valamint azokkal a javaslatokkal amelyek mentén a Product Ownerrel közösen döntést hozhatnak a megvalósítás mikéntjéről.

A megbeszélés célja, hogy a teljes Scrum csapatnak egységes megértése legyen a user storykban megfogalmazott üzleti igényekről, a Fejlesztők pedig ezek alapján átlássák, hogy milyen feladatokat kell elvégezni, hogy az igényből működő termék legyen. Mivel a user storyk nem konkrét megoldást tartalmaznak, hanem egy ügyfél számára megoldandó problémát, ezért a csapatnak azt is el kell döntenie hogy fog választ adni a user storyban megfogalmazott szükségletekre. Ha egy user story túl általánosan fogalmaz meg egy igényt, akkor ezen az összejövetelen van arra lehetőség, hogy a megfelelő kérdéseket feltéve kisebb, pontosabb, de önállóan is értéket képviselő egységekre bontsák tovább azt. Ha az igény jól körülhatárolható akkor pedig a Fejlesztők több javaslatot is megfogalmazhatnak a megoldás kapcsán

és az erőforrás igények valamint a létrehozott érték figyelembe vételével a csapat dönthet a megvalósításról.

A backlog refinement során arra törekszik a Scrum csapat, hogy a lehető legtöbb backlog elem megfeleljen az előző alkalommal megismert INVEST kritériumoknak.

A Scrum csapat jelenleg egy hangoskönyv applikáción dolgozik, és a termék életciklusának elején járnak. A megbeszélés elején a Product Owner felvázolja a jelenlegi prioritásokat.

A legfontosabb elem a backlogban, hogy az ügyfelek regisztrálni tudjanak. A regisztráció után a sorban következő elem, hogy a felhasználók le tudják játszani a hangoskönyveket. Ezután következik, hogy a kiadók fel tudják tölteni a saját hangoskönyveiket, majd legvégül az ügyfeleknek nyújtott fizetési megoldás van terítéken.

Az egyik fejlesztő a megbeszélés elején jelzi, hogy több kérdése is felmerült a regisztrációval kapcsolatban ami befolyásolja azt, hogy milyen feladatok lesznek majd ezzel a Sprintben.

- Milyen adatokat szeretnénk kérni a regisztráció során az ügyféltől? Szükség van-e arra, hogy valamilyen közösségi média profillal tudjon regisztrálni? Szükség lesz-e arra, hogy visszaigazoljuk a regisztráció során megadott e-mail címet?

A Product Owner átgondolja az üzleti igényt és a következő választ adja:

- Jelenleg elég, ha az ügyfelek megadják az e-mail címüket és a keresztnevüket a regisztrációkor. A megadott e-mail címet minden esetben validáljuk, mert ezzel kiküszöbölhető, hogy más nevében regisztráljon valaki, vagy rossz szándékkal tömegesen érvénytelen regisztrációt hozzon létre. A közösségi

média profilok közül a Facebook, Google és Apple fiókokkal is lehessen regisztrálni.

A beszélgetés során az eddigi egy user story máris legalább öt különböző kisebb elemre bontható (adatok megadása, e-mail cím visszaigazolása, három különböző közösségi média profil kezelése). Ezeket a csapat közösen dokumentálja és ezzel frissíti a termék backlogot. A backlog refinement egyik lényege pont ez: a nagyobb, vagy kevésbé világos elemeket kisebb, könnyen értelmezhető és legfőképpen működő szoftverré konvertálható egységekre bontani.

A megbeszélés folytatása során az egyik elsősorban tesztelésben járatos fejlesztő a következő kérdést teszi fel:

- Én úgyis csak a teszteket fogom megírni. Szóljatok ha én jövök addig játszom a telefonomon.

A Scrum Master itt tesz egy javaslatot:

- A tesztelés ugyanolyan fontos része a folyamatnak mint bármelyik másik lépés. El tudnád mondani, hogy a regisztráció során mit tervezel tesztelni?
- Ahogy a klasszikus vicc is mondja: egy tesztelő bemegy a bárba és kér egy sört. Majd kér mínusz egy sört. Majd kér végtelen sört. Végül kér “szarvas” korsó sört. Komolyra fordítva a szót: ha a felhasználó érvénytelen e-mail címet ad meg akkor sikertelen lesz a regisztráció. Ha pedig nem védjük le a beviteli mezőt akkor nyitva hagyunk egy kiskaput a támadásoknak.

A Scrum Master megköszöni az értékes hozzászólást és megerősíti a fejlesztőt abban is, hogy a tesztelési feladatok előzetes vizsgálata is adhat értéket a párbeszédhez.

A mostaniból egyből két minőségi kritériumot is hozzá tud adni a csapat a regisztrációs user storyhoz: ellenőrizzük a megadott email címet, hogy valós-e és ne

engedjünk olyan tartalmat tovább a beviteli mezőből ami kárt tehet a szoftverünkben.

A backlog refinement másik fontos eleme, hogy nem csak az egyértelmű pozitív, hanem a valamilyen okból kifolyólag negatív eseményeket is meg tudjuk vizsgálni és az ezekre adott választ be tudjuk építeni a minőségi kritériumokba.

A csapat hasonlóan halad tovább és egyre jobb képet kapnak az üzleti igényekről és a minőségi elvárásokról. Az idő azonban lassan a végéhez közelít és a Scrum események ideje mindig rögzített, a Scrumban nincs túlóra. A Scrum Master ezért azt javasolja, hogy elégséges mértékben bontotta részekre a backlog elemeket a csapat, de még nem tudják, hogy mekkora energia ráfordítással tudják őket megvalósítani. Szükség lenne a csapattól egy erre vonatkozó becslésre.

Becslési technikák

Mielőtt megismerjük a lehetséges becslési technikákat le kell szögeznünk, hogy ezek semmi esetre sem jelentenek elköteleződést vagy ígéretek. A becslés kizárólagos célja, hogy a rendelkezésre álló információk alapján egy előrejelzést adjon arra, hogy milyen ütemben haladhat a fejlesztés **a fejlesztőcsapat számára**. A becslések pontosságát utólag érdemes megvizsgálni és a tapasztalatok alapján akár módosítani a még fejlesztés előtt álló backlog elemek becsült méretét. Ahogy a Scrum csapat tapasztaltabb és éretteb lesz valamint egyre több piaci visszajelzést kapnak a becslések pontossága is egyre nagyobb lesz.

A becslés folyamata is adhat értéket a backlog refinementhez. Az összes technika arra épül, hogy az egyes csapattagok becslései nem ismertek a többi csapattag számára. Önállóan becsülnek először, majd egyszerre mondják/mutatják meg a becsült értéket a többieknek. Ha nem egységes a becslés akkor minden csapattag elmondja ő miért azt az értéket adta. Ilyenkor kiderülhet, hogy a különböző értékek között különböző elképzelések vannak a megvalósításról. Ezek összehangolásával

nem csak a becslés lesz pontosabb, de jobb megértést kap a csapat az üzleti igényről és annak megvalósításáról valamint újabb minőségi kritériumok kerülhetnek hozzáadásra. Egy elem becslése akár több körös is lehet attól függően mennyire egyértelmű a megvalósítandó üzleti érték.

Póló méret

A legegyszerűbb és emiatt talán legkevésbé pontos becslést a póló méretek alapján kategorizáló módszer adja. A csapattagok minden backlog elemet 4 kategóriába sorolnak: S, M, L, XXXXL. Az első három kategória a könnyű, közepes, nehéz feladatokat fedi le a legnagyobb pedig vagy további információt igényel az üzleti igényről vagy annyira összetett, hogy nem kivitelezhető egy Sprintben, esetleg olyan technikai tudást igényel amely nem áll a csapat rendelkezésére. Már ebben a becslési technikában is látható az az elv, hogy az egyes backlog elemeket nem önmagukban értékeljük, hanem egymáshoz viszonyítva nézzük meg mennyi ráfordítás szükséges a megvalósításukhoz.

A póló méret típusú becslés inkább nagyobb egységek becsléséhez javasolt.

Vödör módszer

A vödrök alkalmazásával nagy mennyiségű backlog elemet tudunk gyorsan kategóriákba rendezni. Leginkább akkor érdemes alkalmazni, ha a termék már érettebb életciklusban van és számos lehetséges fejlesztési irány szerepel a backlogban amelyekről hozzávetőlegesen szeretnénk látni a ráfordítás mértékét. A pólókhoz hasonlóan az egyes vödrökhöz is egy-egy érték tartozik, de itt már viszonyszámok. A 0,1,5,8 értékekhez az alaposan körüljárt, jól értelmezhető vagy egyszerűen triviális értékek tartoznak míg az 50,70,100,200-as értékű vödrökbe a komplex, tisztázatlan backlog elemek.
A vödrök alkalmazásakor véletlenszerű sorrendben vizsgálunk meg backlog elemeket és rövid megbeszélés után belehelyezzük őket az egyik kategóriába. Ha az

előrehaladás során nem megfelelő egy elem relatív mérete áthelyezzük egy másik vödörbe. A becslés mérete egy fokkal finomabb a több kategória miatt, jól elkülönülnek egymástól a megértett és részletezendő igények és nagy mennyiséget tudunk gyorsan feldolgozni. Ugyanakkor kevesebb teret ad ez a módszer az interakcióra. A mennyiség érdekében minőségi kompromisszummal fizetünk.

Planning poker

A planning poker akár az előző módszerek ötvözeteként is értelmezhető. A poker során az egyes backlog elemeket egyesével becsli a csapat - akár többször is megismételve a becslést az egyes elemeknél. A becslés nagyságát pedig a Fibonacci sorozat értékeivel adják meg, azaz: 1,2,3,5,8,13... értékeket rendelnek az egyes backlog elemek mellé. Az értékek itt is egymáshoz viszonyított méretet jelentenek egy viszonylag részletes skálán.

Ennél becslési módszernél ki kell emelni egy gyakori hibát: a Fibonacci sorozat értékeit (vagy ahogyan a módszer nevezi story pontokat) semmiképpen se feleltessük meg az egyes feladatok időigényének. Ez gyakorlatilag egyenértékű azzal, mintha elvárásokat fogalmaznánk meg azzal kapcsolatban, hogy egy elemnek mikorra kell elkészülnie, pedig az információk legfeljebb részben állnak rendelkezésünkre.

Visszatérve a megbeszélésünkre a csapat a planning poker megoldást használva relatív értékeket rendel a korábban részletezett backlog elemekhez. A becslés eredményei alapján a regisztrációs folyamatra 5, a visszaigazoló emailre pedig 3 pontot adott a csapat. Mivel a Facebook fiókkal történő regisztrációban már jártas a csapat ezért az 8 pontot kapott. A Google és Apple fiókok használatában még nincs tapasztalatuk, ezért ott szükségesnek tartják, hogy egy új elem egy technikai spike kerüljön be a backlogba. Ez az elem egy rögzített időt kap és ebben az időkeretben

a csapat megszerzi és dokumentálja azokat az ismereteket amelyekkel pontos becslést tud adni a feladat mértékére. Addig 21-21 pontot adnak a két elemre a bizonytalanságot jelezve.

A becslések végeztével a megadott időkeret is lejár, így az esemény is véget ér. A Product Owner és a Fejlesztők is elégedetten távoznak, hiszen magasabb szinten értik az üzleti igényeket és a termék backlog elején található elemek készen állnak arra, hogy a következő Sprintben inkrementum legyen belőlük.

Sprint planning

> *» A Sprint Planning a Sprintet úgy indítja el, hogy meghatározza a Sprint alatt elvégzendő munkákat. Ez a terv az egész Scrum Team közös munkájának eredménye. «*

Hétfő 9:15 - a Sprint indulásának napja. A Fejlesztők és a Product Owner azért ültek le egy megbeszélésre, hogy létrehozzák az adott Sprint backlogját. Az előző fejezetből már ismerjük, hogy ez a munkanyag három elemből áll: a Sprint Goalból, a termék Backlog Sprintre kiválasztott elemeiből és az Inkrementum elkészítéséhez szükséges tervből.

A fentiek alapján először az első kérdést kezdi el tisztázni a csapat: miért értékes ez a Sprint?

- Az előző refinement alkalmával sikerült a regisztrációt kisebb részekre bontani és becslést is adott a csapat ezekre a backlog elemekre. Mivel ez alapvető funkció azt javaslom, hogy a Sprint célja az legyen, hogy a felhasználók tudjanak regisztrálni az applikációba. - kezdi a megbeszélést a Product Owner
- Ez jó célnak tűnik, de én nem hagynám ki belőle a biztonság kérdését sem - veti közbe az egyik fejlesztő.

- Mi lenne, ha mérhetővé is tennénk a célunkat? - teszi fel a kérdést a Scrum Master.
- Köszönöm a javaslatokat! - veszi vissza a szót a Product Owner - Mit szólnátok ahhoz a Sprint célhoz, hogy "A következő Sprint végéig legyen 100 biztonságos folyamaton keresztül regisztrált ügyfelünk"?

A Sprint cél egyöntetűen tetszik a csapatnak, hiszen megfelel a SMART célkitűzés elvárásainak. Ezzel a sprint célt elfogadottnak tekintik.

A következő lépésben a csapat feladata annak meghatározása lesz, hogy mi végezhető el az adott Sprint alatt.

- A backlog első néhány elemét elég jól megértette a csapat és szorosan kapcsolódnak is a Sprint céljához, nézzük meg ezek közül mi valósítható meg a Sprintben? - indítja el a beszélgetést a Scrum Master.
- Szerintem a regisztrációs felületet és a visszaigazolást biztosan meg tudjuk csinálni, ha minden rendben fog menni, akkor a Facebookos regisztráció is bele fog férni. - javasolja az egyik fejlesztő.
- Azt vedd figyelembe, hogy ugyan ketten már csináltatok ilyet, de én például nem vagyok túl járatos a Facebook API használatában. - érvel egy másik fejlesztő.
- Ha párban programoznánk akkor egyrészt ezt át tudjuk hidalni, másrészt neked is lesz gyakorlati tapasztalatod benne. - válaszol az első fejlesztő.
- Rendben ez jól hangzik, így szerintem el tudunk készülni ezzel is.

A Product Owner elégedetten hallgatja a beszélgetést, de azért rákérdez:

- A Google regisztráció semmiképpen sem fér bele?
- Sajnos abban nincs tapasztalatunk még és a fejlesztési ciklus elején járunk, tehát nem tudjuk, hogy milyen problémák léphetnek fel miközben kialakítjuk

az adatbázisokat és megírjuk a teszteket. Ha ebbe is belekezdünk akkor veszélybe fog kerülni a Sprint Goal. - válaszolja az egyik fejlesztő.

- Köszönöm a visszajelzést! Akkor ezt inkább engedjük el a Sprintben.

A csapat a beszélgetés végén kiválasztja az első három elemet a termék Backlogból. Ezeket az elemeket szeretnék működő szoftverré konvertálni.

Záró lépésként a Fejlesztők azt határozzák meg: hogyan végzik el a kiválasztott munkát, milyen konkrét kódolási, tesztelési, infrastrukturális és egyéb technikai feladatokat kell elvégezni, hogy létrehozzák az inkrementumot.

- Ha biztosak akarunk lenni a dolgunkban, hogy végzünk akkor elég lenne a visszaigazoló e-maileket manuálisan tesztelni. - javasolja az egyik fejlesztő.
- Rendben, a Definition of Done csak azt szögezi le jelenleg, hogy minden elemet tesztelni kell. Vegyük viszont figyelembe, hogy ahogy bővül a termékünk a manuális tesztelés egyre több időt fog igénybe venni ami hosszú távon technikai adósságként fog jelentkezni. Ez később azzal járhat, hogy új funkciók létrehozása helyett ezeket a hiányosságokat kell foltoznunk. - mondja egy másik fejlesztő.
- Vállaljátok ezt a kockázatot? - kérdezi meg a Scrum Master.

A csapat úgy dönt, hogy jelenleg ez vállalható kockázat majd folytatják a feladatok részletezését.

A Sprint célja, a kiválasztott termék Backlog elemek és a megvalósításhoz szükséges feladatok is meghatározásra és dokumentálásra kerültek így a megbeszélés végére létrejön a Sprint Backlog. A Sprint Planninget lezárják és elindul a munka.

Sprint Review

» *A Sprint Review célja, hogy megvizsgálja a Sprint eredményeit és meghatározza a jövőbeni korrekciókat. A Scrum Team a legfontosabb érdekelt*

feleknek mutatja be munkája eredményeit, továbbá megvitatják a haladást a Product Goal irányában.

Az esemény során a Scrum Team és az érdekelt felek áttekintik, hogy mit értek el a Sprintben, és mi változott a környezetükben. Ezen információk alapján működnek együtt a résztvevők a további teendőket illetően. «

Péntek 10:00 - A Sprint második hetének utolsó napja. A Fejlesztők és Product Owner meghívták a szervezet azon tagjait akik érdekeltek a termék fejlesztésében, hogy megnézzék mi az az inkrementum amit a csapat a Sprint végén leszállított.

- Köszönöm, hogy eljöttetek! Szeretnénk megmutatni, hogy ebben a Sprintben milyen értéket adunk az ügyfelek számára, mi az amit a csapat létrehozott. A megbeszélésünk célja az, hogy megvizsgáljuk a létrejött inkrementumot és a látottak alapján kérdéseket, javaslatokat fogalmazzunk meg, amelyek kijelölik a fejlesztés és a termék jövőbeli irányát. - kezdi a megbeszélést a Product Owner.

A bevezetés után a fejlesztők elkezdik bemutatni a termék elkészült új funkcióit. A csapat az útközben felmerülő akadályok ellenére minden tervezett feladatot sikeresen megoldott.

- … és ez a regisztrációs folyamat vége. - fejezi be a demonstrációt az egyik fejlesztő.
- Nagyon jól néz ki a felület - veszi át a szót a marketing vezető. Ha jól látom akkor jelenleg csak a keresztnevet és az e-mail címet kérjük el a felhasználótól. Mit gondoltok lehetséges lenne, hogy ezt kiegészítsük lakcím adatokkal?
- Technikailag nem látom akadályát. - feleli a fejlesztő.
- Megkérdezhetem mi lenne a cél a lakcímmel? - kérdezi meg egy másik fejlesztő.

- Egyrészt a számlák kiállításához szükség lesz rá, másrészt ha a későbbiekben szeretnénk mondjuk fülhallgatókat is értékesíteni kiegészítő termékként akkor tudnunk kell, hová szállítsuk ki őket. - érkezik a válasz.
- Mindkettő jó ötletnek hangzik, le is jegyeztem őket - mondja a Product Owner. Két kérdésem merült fel ezzel kapcsolatban: Biztosan szükség van ezekre az adatokra már a regisztráció során? Mikorra tervezzük a kiegészítő termékek forgalmazását?
- A digitális termékekkel ebben a negyedévben, a kiegészítőkkel pedig inkább az év végén szeretnénk megjelenni.
- Ebben az esetben talán az lenne a legjobb, hogy az első vásárlásnál kérjük be ezeket az adatokat, hogy gördülékeny maradjon a regisztrációs folyamat. Utána ezeket elmentve már automatikusan be tudjuk tölteni egy későbbi vásárlásnál.

A javasolt megoldást elfogadják a Product Owner pedig a visszajelzések alapján több termék Backlog elemet is hozzáad az eddigiekhez. A hangoskönyvek vásárlásánál a számlázáshoz szükség lesz az ügyfelek lakcímére, ezeket a későbbi tárolásnál automatikusan be kell majd tölteni, a cég fizikai termékeket is szeretne értékesíteni.

A fenti példa egy olyan esetet mutat be, ahol a Scrum Csapat és az érdekelt felek a Scrum alapelvei alapján megvizsgálják a kész inkrementumot és ezek alapján döntéseket hoznak a jövőbeli állapotról. Ahhoz, hogy az igényeknek leginkább megfelelő terméket hozhassunk létre, folyamatosan látnunk kell működés közben a jelenlegi állapotot.

- Az email és a keresztnév is érzékeny adatnak minősül, viszont nem láttam, hogy hozzájárulást kértünk volna ezek használatához a leendő ügyfelektől. Ha ezt nem tesszük meg akkor komoly büntetésre számíthatunk! - jegyzi meg a cég jogi képviselője.

- Még nem nyilvános az oldalunk, tehát az ügyfelek adatai nincsenek nálunk. Ezzel azt is mondod, hogy amíg ez nincs meg nem indulhatunk el a piacon? Szeretnénk legalább 100 regisztrációt a következő Sprint alatt. - kérdez vissza a Product Owner.
- Ha a következő Sprintben meg tudjuk oldani, hogy utólagosan hozzájárulást kérjünk az akkor regisztráló ügyfelektől *és* megoldjuk, hogy az azt követő időszakban ez kötelező legyen az új ügyfeleknek akkor vállalható a kockázat.

A Product Owner újabb két elemmel bővíti a termék Backlogot a beszélgetés után: a már regisztrált ügyfelek járuljanak hozzá adataik kezeléséhez. Az új regisztrációk során legyen kötelező ugyanez a hozzájárulás. Mindkettő magas prioritást is kap, hiszen enélkül kockázatos a termék piaci indulása.

A Sprint Review során a fentiekhez hasonló párbeszédek zajlanak le és ez alapján bővül a termék Backlog és változik az elemeinek a sorrendje.

Elképzelhető, hogy a gyakorlatban “Sprint Demo” néven találkoztok ezzel az eseménnyel. Ha a folyamatok egyébként hasonlóak akkor nem az elnevezésen fog múlni a siker. Ha azonban ez tényleg egy egyirányú dolog, ahol nem kap a csapat visszajelzést, akkor az egy intő jel lehet arra, hogy valamilyen probléma van a szervezetben.

Ehhez hasonló hiba lehet a Sprint Review kapcsán, ha nem működő terméket látunk az esemény során, hanem egy prezentációt arról, hogyan kellene (majd) működnie a terméknek, vagy ennél is rosszabb esetben egy beszámolót arról mit csinált a csapat a Sprint során. Ez szintén figyelmeztető jel kell, hogy legyen, mivel a csapatnak nem sikerült értéket előállítani és vagy ezt próbálják meg elfedni a státusz jelentéssel vagy a szervezeti elvárásoknak próbálnak meg megfelelni. Bármelyik magyarázat is az igaz a csapatnak felül kell vizsgálnia a saját működését, amire kiváló alkalom nyílik a Sprint Retrospective során.

Időkeretek

A Sprint maga is egy időben korlátos esemény. Az általánosan használt gyakorlat, hogy két hetes Sprintekben dolgozik a csapat. A Scrum Guide a maximális időkeretet egy hónapban határozza meg. Függetlenül attól, hogy mennyi feladatot zárt le a csapat az adott Sprint lezárul és az új Sprint elindul amikor a csapat által meghatározott időkeret végéhez érkezünk.

A Sprintet meg lehet szakítani, de csak abban az esetben, ha a Product Owner úgy dönt, hogy ha a Sprint Goal elavul. Egy egyszerű példa lehet erre, ha például egy fizetési megoldás megszűnik a piacon miközben azt integrálja a csapat a termékbe. Mint a példából is látszik ez tényleg csak extrém esetekben fordul elő.

A Sprint tervezés időkeretét egy hónapos Sprintre vonatkoztatva 8 órában határozza meg a Scrum Guide. Mivel a Backlog Refinement nem része a keretrendszernek ezért a Sprint tervezésre akkor érdemes hosszabb időt fordítani, ha kevésbé jól határolt backlog elemek alapján kell Sprintet tervezni a csapatnak.

A Sprint Review 4 órás esemény egy egy hónapos Sprint esetén. A résztvevők számától és az inkrementum komplexitásától függően ez változhat, de érdemes időt szánni, hiszen az érdekelt felek ezen az eseményen tudnak érdemben értesülni és beavatkozni a termék jövőjébe.

Agilis metrikák

> *Nem tudjuk menedzselni amit nem tudunk mérni.*

Talán ismerjük ezt az idézetet amit eredetileg W. Edwards Demingtől származik. Egy aprócska probléma van vele, mégpedig hogy kiragadták a szövegkörnyezetből. Az eredeti idézet így hangzik:

» *Hibás lenne azt állítani, hogy nem tudjuk menedzselni amit nem tudunk mérni. - egy költséges mítosz.* «

Az agilis metrikák területén sincs ez másképpen. Rengeteg segítséget nyújthatnak abban hogy fejlesszük a termékünket vagy a folyamatainkat, de hogy újra Deminget idézzük:

» *A Problémák 85%-ért ti feleltek.* «

Hogy hogyan lehet egy mérőszámot rosszul használni? Íme egy kiváló példa a XIX. századi Angol gyarmatbirodalomból.

A Kobra hatás

1832-ben járunk a távoli Indiában, ami ekkor a brit korona fennhatósága alá tartozott. A királyi helytartóknak és katonáknak nem csak a felkelésekkel és lázadásokkal, addig soha nem látott betegségekkel és a monszun okozta fülledt, esős, szinte kibírhatatlan időjárással kellett szembenézniük hódítóként, hanem a dzsungelben élő vadállatok jelentette veszélyekkel is. Ezek közül is kiemelkedett egy rettenetes fenevad: a kobra. Aki megérezte a kobra halálos marását egy fél órán belül kivétel nélkül a siralomházban végezte. Lord William Bentinck aki ekkor az ország kormányzójaként szolgálta a királyi udvart nem nézhette tétlenül, hogy emberei úgy hullanak mint a legyek. Kihirdette hát, hogy bárki aki egy kobra lenyúzott bőrét beszolgáltatja bármely kormányzói kirendeltségen 1 shilling 2 penny üti a markát.

Az intézkedés először hatalmas sikernek bizonyult: nem csak a kobra bőrök kezdtek el hatalmas halmokban gyűlni a Kelet-Indiai Társaság szolgálatában álló vargák és bőrdíszművesek legnagyobb örömére, hanem a kígyómarásokról szóló beszámolók és az ennek tudható halálesetek száma is meredek esésnek indult. Néhány hónap elteltével, mielőtt Lord Bentinck hátradőlhetett volna azonban valami történt. A

helyőrségek új és újabb tagjait érte halálos csók. Az őrjáratoknak újra kiemelt óvatossággal kellett masírozni a városokban. Az adószedők külön kísérettel mertek csak kivonulni, hogy behajtsák ami a királynak járt. Egy alkalommal egy helyi földműves házát ellenőrizték és nem hittek a szemüknek.

A ház egyik hátsó eldugott zugában százasával tekeregtek a halálosztó hüllők. Ahol ugyanis kereslet van valamire - jelen esetben a kígyók lenyúzott bőrére - ott előbb-utóbb megjelenik a kínálat is. A helyiek közül sokan rájöttek, hogy felesleges megvárni míg előbújnak a kobrák a dzsungelből, ha néhány példányt befogva otthon is kiváló tenyészetet lehet belőlük létrehozni és a bőrüket pedig jó pénzért eladni a briteknek. Ahogy erre fény derült a kormányzó azonnal beszüntette a kobrákért járó jutalmak kifizetését. Ahol azonban megszűnik a kereslet ott hamarosan a kínálat is követni fogja azt: a földművesek tömegesen engedték szabadon a mit sem érő kobraténysészetüket. Lord William terve nem csak hogy nem vált be, de a tenyésztett kígyókkal növelt állomány még nagyobb veszélyt jelentett a britekre.

A fenti anekdota szereplői persze nem valósak, de jól példázzák, hogy abban a pillanatban ahogyan egy kvantitatív mérőszámot (az élő veszélyes kobrák száma) döntési eszközként használnak (fizetnek a kobra bőrért) szinte biztosan manipulálni fogják ezt a mutatót (elkezdenek kobrát tenyészteni otthon) és ez torzító tényezőként működik, torzítva ezzel azt a folyamatot amit figyelemmel kell kísérni (csökkentsük a kobramarás esélyét).

A fenti törvényt Donald T. Campbell pszichológus és társadalomtudós fogalmazta meg és a gyarmati kobrák számán kívül az élet számos más területén is visszaigazolható volt.

Az agilis metrikák mindegyikénél igyekszem majd kitérni arra, hogyan torzíthatja a folyamatot, ha nem megfelelően használjuk ezeket.

A Fejleszési folyamat metrikái

A következő mérőszámok a Fejlesztők számára nyújthatnak információt arról, hogy hogyan tervezzék meg a következő Sprinteket, illetve a Product Owner számára nyújthatnak segítséget az termék Backlog elemeinek sorba rendezésében.

Sprint velocity

A Sprint velocity azt mutatja meg, hogy egy átlagos Sprintben mennyi munkát tudnak elvégezni a Fejlesztők. A Sprint velocity kiszámításához két változó ismeretére van szükségünk: mennyi munkát végeztek el a fejlesztők és mennyi időt vett igénybe a munka elvégzése.

A kobra hatás elkerülésének érdekében fontos leszögezni, hogy sprint velocity egy leíró mérőszám és nem a csapat sikerességének vagy hatékonyságának a mutatója. A csapat törekedhet arra, hogy növelje ezt a mérőszámot, ha úgy érzik, hogy egy adott területen kellő tapasztalatot szereztek, de ezt semmiképpen nem szabad elvárásként megfogalmazni feléjük.

A Sprint velocity arra ad rálátást a Fejlesztőknek, hogy egy Sprint terv készítésekor mennyi az a feladat mennyiség amelyet nagy valószínűséggel inkrementummá tudnak alakítani.

A Sprint velocity kiszámítása egy egyszerű hányados:

$$Sprint\ velocity\ = \frac{Sprint\ során\ elvégzett\ munka}{Sprint\ ideje\ napokban\ kifejezve}$$

Ha egyszerűen a termék backlog elemek számában határozzuk meg az elvégzett munka mennyiségét, akkor egy két hetes sprintben (10 munkanap) ha a csapat 30 backlog elemet alakított inkrementummá a következő egyenletet kapjuk:

$$Sprint\ velocity\ =\ \frac{30\ db}{10\ nap}\ =\ 3\ db/nap$$

Ha egy átlagos Sprint következik akkor a csapat tudni fogja, hogy ~30 backlog elemet érdemes a Sprint tervezés során a Sprint Backlogba helyezni. Mivel a darabszám nem sokat árul el az egyes elemek komplexitásáról ezért a gyakorlatban inkább a becslésekkor használt story pontokkal kalkulálják a Sprint velocityt. Így ha az adott sprintben 24 story pont volt a backlog elemek becsült értékének összege akkor az egyenlet így változik:

$$Sprint\ velocity\ =\ \frac{24\ SP}{10\ nap}\ =\ 2,4\ SP/nap$$

Ha a következő Sprint 9 munkanapból áll (például egy munkaszüneti nap miatt) és a termék Backlog első hat helyén álló elemeket a Fejlesztők 8,5,3,5,2,2 pontra becsülték akkor az első négy elemet nagy valószínűséggel be tudják fejezni a Sprint során (21 SP / 9 nap = 2,33 SP/nap < 2,4 SP/nap) és talán az ötödik elemet is be tudják fejezni (23 SP / 9 nap = 2,55 SP/nap > 2,4 SP/nap), de nagy valószínűséggel már nem fog beleférni a hatodik elem.

Ha csak a számokat nézzük akkor több csapdába is beleeshetünk. Egyrészről a story pontok egy becslés eredményeként születnek. Egy tapasztalatlan, kevésbé érett csapat esetén megkérdőjelezhető hogy mennyire jó becsléseket tudnak adni az egyes backlog elemekre. Másrészt a story pontok nem abszolút mérőszámok, hanem az egyes backlog elemek egymáshoz viszonyított bonyolultságát hivatottak kifejezni.

Egy újabb csapda, ha a csapat teljesítményét a “szállított” story pontok mennyiségében kezdik el mérni. A kobra hatásnak megfelelően erre a csapat egyre nagyobb és nagyobb becsléseket fog adni az egyes backlog elemekre és ugyan a számok jól fognak mutatni a ténylegesen előállított érték azonban minimális lesz.

Súlyosbító tényező lehet, ha egyéni szinten kezdi el egy szervezet mérni a velocityt *és* teljesítmény mutatóként is használják azt vagy amikor két csapat teljesítményét a velocity alapján akarák összehasonlítani.

A számos negatív példa után nézzük meg azt, hogyan érdemes használni ezt a mérőszámot. Egyrészt soha ne hagyatkozzunk kizárólag a velocityre egy Sprint megtervezésekor. Indikátorként jól működhet, de a Fejlesztők látják a legjobban azt, hogy egy-egy backlog elem milyen komplexitású és mi az amit magabiztosan inkrementummá tudnak alakítani. Másrészt érdemes több már befejezett Sprint átlagát venni amikor a jövőbeni Sprintek várható értékeit becsüljük. Harmadrészt pedig a Sprint Retrospectivek alkalmával érdemes felülvizsgálni a becslések pontosságát abban a tekintetben, hogy sikerült-e az összes technikai feladatot azonosítani és azok összetettségét jól megállapítani előre. Ha a velocity rendszeresen egy adott konstans érték körül mozog akkor az is jó indikátor lehet a folyamatok felülvizsgálatára, ha ettől jelentős eltérést tapasztalunk akár felfelé akár lefelé. Ha a sprint során módosítani kell a Sprint backlogon (a Sprint céljának megtartásával) akkor szintén hagyatkozhatunk a velocityre egy új elem hozzáadásánál.

A Sprint velocityt leginkább oszlopdiagrammal és az értékek alapján számított trendvonallal szokták megjeleníteni.

Burndown

A burndown chart egy adott Sprintre vonatkozóan a hátralévő feladatok mennyiségét (y-tengely) veti össze a még hátralévő idővel (x-tengely). Grafikonon ábrázolva egy lineárisan csökkenő egyenes jelzi az ideális állapotot, egy lépcsőzetesen csökkenő görbe pedig a ténylegesen elvégzett feladatok ütemét mutatja meg.

Ha minden a tervek szerint halad a befejezett feladatok görbéje követni fogja a lineárisan csökkenő ideális görbét. Előfordulhat, hogy néha időarányosan több vagy kevesebb feladat van még hátra, de az eltérés nem jelentős és a Sprint végén pedig az összes tervezett feladat elkészül. Nem minden Sprint végkifejlete ilyen ideális és az anomáliák feltárásában segítségünkre lehet a burndown chart.

Ha azt tapasztaljuk, hogy a befejezett feladatok görbéje sokkal nagyobb ütemben csökken mint az ideális és akár a Sprint zárása előtt eléri a 0-t az x-tengelyen, akkor a csapat nagy valószínűséggel túlbecsülte a feladatok komplexitását, azok jóval egyszerűbbek voltak és gyorsabban be tudták fejezni őket. Ilyenkor két dolgot érdemes megfontolni: egyrészt megvizsgálni, hogy mik azok a backlog elemek amik a Sprint célját támogatják és hozzáadni őket a Sprint backloghoz, másrészt a Sprint Retrospective alkalmával felülvizsgálni az eredeti becsléseket és azt, hogy mi okozta az alacsonyabb komplexitást majd ezt az információt felhasználni a későbbi becslések pontosítására.

Az ezzel ellentétes működés amikot az elvégzett feladatok görbéje ugyan csökken de nem az ideális mértékben. Ezt a helyzetet két alesetre oszthatjuk: amikor sikerül befejezni a tervezett feladatokat, de az utolsó pillanatban csökken drasztikusan a feladatmennyiség és éri el a görbe a nullát az x-tengelyen és amikor a hátralévő feladatok mennyisége lassan csökken és el sem éri a végén a nullát.

Az első esetben a folyamatokat érdemes felülvizsgálni, mivel az utolsó pillanatban sikerült befejezni mindent. Elképzelhető, hogy a csapat vízesés szerű modellben dolgozik, vagy több feladatot végeznek párhuzamosan és ezeket egyszerre próbálják meg lezárni. Előfordulhat az is, hogy valamelyik fejlesztési területen szűk keresztmetszet alakult ki. Ahogy látjuk a számok önmagukban csak indikátorok a probléma jelenlétére, de a tényleges okot a csapatnak kell feltárnia és megoldania.

A második esetben a csapat vagy alulbecsülte a feladatok komplexitását vagy túlvállata magát. Mindkét esetet érdemes a Sprint Retrospectiven megvizsgálni és a tapasztalatok alapján felülvizsgálni a korábban már megbecsült, de még meg nem valósult termék Backlog elemek komplexitását vagy a Sprint Backlogba kerülő elemek mennyiségét.

Ismét ki kell emelni, hogy a burndown nem azt mutatja meg, hogy a Fejlesztők jól vagy rosszul dolgoznak-e, hanem a becslések- és az azok alapján készített tervek pontosságáról ad egy pillanatképet.

Burnup

A burnup chart a burdown-al ellentétben nem a még hátralévő munka mennyiségét mutatja meg, hanem egy előrejelzést ad arra vonatkozóan, hogy milyen jövőbeli időpontban fejeződik be egy nagyobb egység fejlesztése. A másik különbség, hogy amíg a burndown a Sprinten belüli haladást mutatja meg addig a burnup esetén az idő alapegysége a Sprint (x-tengely) és az elvégzett feladatok kumulatív értékét vizsgáljuk (y-tengely).

A példa kedvéért vegyünk egy csapatot aki az elmúlt három Sprintben egy egészségügyi applikáción dolgoztak és az orvosok számára fejlesztettek egy felületet ahol a vizsgálatok után fel tudják vinni az eredményeket a rendszerbe. A Sprintek velocityje rendre 34,36,35 SP/Sprint voltak (az egyszerűség kedvéért most a Sprintet vesszük alapegységnek nem a napokat a velocity számításához). Azokat az elemeket amelyeket még nem alakítottak inkrementummá a Fejlesztők összesen 115 SP-ra becsülték. Ha továbbra is stabilan tartják a Sprintenkénti 35 SP-os velocityt akkor előreláthatólag még 4 Sprintre van szükségük ahhoz, hogy teljes egészében befejezzék ezt a nagyobb egységet.

$$35 * 3 = 105 < 115 < 140 = 4 * 35$$

Két hetes Sprintekkel számolva ez 2 hónapos időtáv. Érdemes azonban újra megjegyezni, hogy nem ismerjük a jövőt és nem tudjuk előre, hogy egyrészt tartható-e a velocity vagy nem találkozunk olyan új igénnyel vagy technológiával ami ezt jelentősen befolyásolja. A Burnup semmiképpen nem ígéret, vagy elköteleződés hanem egy indikátor. A gyakorlatban előfordul, hogy a ±5% és a ±10%-os trendeket is figyelembe veszik ezzel ábrázolva az idő előrehaladtával egyre növekvő bizonytalanságot.

A Burnupot leginkább arra érdemes használni, hogy a csapat lássa egy-egy nagyobb egység várható befejezési idejét és átgondolja, hogy mindenképpen szükséges-e az összes ezt érintő termék Backlog elemet inkrementummá alakítani vagy esetleg érdemes az elemek sorrendjén módosítani.

A termék metrikái

A termékkel kapcsolatos metrikák megismerésével olyan mérőszámokat tudunk használni a napi gyakorlatban amelyek megmutatják az ügyfeleink viszonyát a termékhez illetve segítenek abban, hogy számszerűsítsük az egyes termékfunkciók értékét.

Net Promotes Score

Mi az a végső kérdés amely megmondja nekünk hogyan látnak minket az ügyfelek? A Net Promoter Score (NPS) ezt a kérdést fogalmazza meg a következőképpen:

Mennyire ajánlaná ismerőseinek, barátainak a {terméket}?

A válaszadóknak egy 0-tól 10-ig terjedő skálán kell választ adni a kérdésre, ahol a 0 azt jelenti, hogy semmi esetre sem ajánlaná, míg a 10-es érték jelentése a mindenképpen ajánlá.

A kérdésre adott válaszok alapján a válaszadókat három csoportra osztjuk: 0-6 értéket választók az ellenzők (detractors) akik kifejezetten ellenszenvesek a termékünk vagy szolgáltatásunk iránt. A 7-8 értéket választók a passzív ügyfelek, akik használják ugyan a termékünket vagy szolgáltatásunkat, de még nyitottak lehetnek a konkurencia ajánlatára. A 9-10 értéket adó ügyfelek a rajongók (promoters) akik teljesen elkötelezettek irántunk és nem csak lelkesednek, de aktívan meg is osztják az élményeiket másokkal.

A Net Promoter Score számításához a következő képletet használjuk:

$$NPS = Promoters\% - Detractors\%$$

Azaz a rajongók összes ügyfélhez viszonyított arányából kivonjuk az ellenzők összes ügyfélhez viszonyított arányát.

Az NPS értékét célszerű rendszeres időközönként mérni és nem csak a végső eredményt elemezni, hanem az egyes kategóriákon belüli változásokat is megvizsgálni. A mérést nem csak a termék egészére, hanem egyes termékfunkciókra vagy egy kapcsolódó szolgáltatásra (például ügyfélszolgálat) is elvégezhetjük.

Ez a mérőszám sem kivétel az alól, hogy ha teljesítmény elvárásokkal párosul akkor torz képet ad a valóságról. Példaként vegyünk egy ügyfélszolgálatot ahol a cég képviselői közvetlen kapcsolatban állnak az ügyfelekkel. Minden kapcsolatfelvétel után az ügyfelek értékelik az ügyfélszolgálatot amely befolyásolja az ott dolgozók bérét. Ha az ügyfélszolgálaton dolgozók az értékelés előtt elmondhatják, hogy kérem értékelje a munkámat 10-esre, akkor az esetek nagy részében ezt az ügyfelek szívesen adják ezt az értéket még akkor is, ha egyébként nem ajánlják a szolgáltatást másoknak.

Value Score Card

A korábban megismertek alapján a Product Owner feladata, hogy maximalizálja a Scrum csapat által előállított értéket. Azonban az, hogy pontosan mi az érték és mekkora az előállított érték nem is olyan egyszerű meghatározni. Ha a vállalat oldaláról nézzük akkor a bevétel növekedés egyértelműen értéket képvisel. Az ügyfelek számára az jelenti az értéket, ha minden nap időt takaríthatnak meg a használatával. Ha az ilyen és ehhez hasonló értékeket egy mérőszámban szeretnénk összefoglalni, akkor nem lesz könnyű feladatunk. Tegyünk rá egy kísérletet.

A termék értéke különböző fogalmat takar az érdekelt felek számára. Ha egyszerűen pénzügyi oldalról vizsgáljuk meg a termék értékét, akkor az általa termelt bevétel és a fejlesztése során felmerült költségek különbözeteként értelmezhetjük. Egy előfizetéses zenei szolgáltatást nyújtó applikáció esetében az ingyenes csomagot igénybe vevő felhasználók, a fizetős csomagra váltók és a lemondást választók száma a vállalat oldalról jó mérőszámnak tűnhet, de igen kis mértékben mond bármit is az ügyfélélményről. Egy híroldal látogatottsági adatai szintén árulkodóak lehetnek, de nem mutatnak teljes képet.

Ha megfelelő mutatószámot akarunk találni akkor egy olyan átfogó értéket kell keresnünk amely megmutatja, hogy amit fejlesztünk hogyan szolgálja az üzlet érdekeit és elégíti ki az ügyféligényeket.

Sajnos olyan mérőszám amely ezt önmagában képes megmutatni és egységesen alkalmazható minden szervezetre nem létezik. Ez azonban korántsem jelenti azt, hogy a megfelelő módszertannal ne tudnánk egy adott szervezet számára specifikus és hasznos mérőszámot alkotni amely kifejezheti az előállított értéket.

Ennek a mérőszámnak az előállításához szükség van az összes olyan érdekelt fél azonosítására akik számára a termék értéket teremt, bármilyen formában is jelenjen

meg ez az érték. Ezek után meg kell határoznunk, hogy az egyes felek számára a termék milyen aspektusai jelentenek értéket. Ezeket egy egységes rendszerbe rendezve pedig megkapjuk a Value Score Card-ot.

Nézzük meg egy példán keresztül, hogy használhatja ezt egy Product Owner az egyes jövőbeni termék funkciók értékének becslésére. Miután elvégezte az előkészítő munkát úgy vélte, hogy három különböző dimenzió mentén fogja a termék Backlog elemeit értékelni, mindegyiket egy hetes skálán. A kialakított Value Score Card-on értékeli: az ügyfelek számát akik számára értéket jelent majd ez a funkció, a várakozások mértékét, azaz mennyire fontos ez azok számára akiknek készül, az üzleti célok támogatását, azaz a vállalat vagy a termék céljaival mennyire mutat egy irányba a funkció. Ezeken túl a becsléséhez párosít egy bizonytalansági együtthatót is.

Az új termék Backlog elem amelyet értékel egy közösségi médiás megosztás lehetősége egy rövid videók megjelenítésére szakosodott platformon. A Product Owner az ügyfelek számára 7, a fontosságra, 6, az üzleti célokra pedig 5 pontot ad. A becslése pontosságára pedig 85%-ot. A termék Backlog elem Value Score Card értéke így

$$(7 + 6 + 5) * 0,85 = 15,3$$

lesz. Ugyanúgy mint a story pontok alkalmazásánál ez is csak egy becsléseken alapuló szubjektív érték, arra viszont jól alkalmazható, hogy a megvalósult funkciókat megvizsgáljuk és összevessük az előzetes becslésünkkel, valamint szükség esetén finomítsuk a Value Score Card modellünket több paraméter hozzáadásával.

A Product Ownernek közösen kell dolgoznia azon az érdekelt felekkel, hogy elégséges mértékű dimenziót adjon a saját Value Score Card rendszerének és egyre pontosabb becsléseket rendeljen az egyes termék Backlog elemekhez.

Objectives and Key Results

Az eddigiek során olyan mérőszámokat ismertünk meg amelyek relatív vagy szubjektív elemeket tartalmaztak és ezért leginkább a becslést és annak finomítását segítették elő. Az Objects and Key Results (OKR) egy olyan rendszer amely segít a szervezet egységeinek a megfelelő célokat kitűzni és kellő felhatalmazást ad a kezükbe, hogy el is érjék azokat. Az egységek kitűzött céljai a szervezet stratégiai célkitűzéseit támogatják ezzel növelve a összhatékonyságot. Az OKR modell két részből tevődik össze a célkitűzésből (Objective) és az ehhez kapcsolódó kulcsfontosságú eredményekből (Key Results).

A célkitűzés egy tömör, cselekvésre buzdító, és ambiciózus állítás amely leírja a jövőbeni kívánt állapotot. A kulcsfontosságú eredmények pedig néhány számszerűsíthető, mérhető és konkrét cselekvésből állnak amellyel nyomon követhető a haladás a célkitűzés felé. Általánosságban egy OKR a következő formátumban írható le:

A *{célkitűzést}* valósítjuk meg és a *{kulcsfontosságú eredményekkel}* mérjük a megvalósulását.

Ha hatékony OKR rendszert szeretnénk létrehozni, akkor éles határokat kell húznunk a célkitűzés és a kapcsolódó kulcsfontosságú eredmények között. Amíg az előbbi azt írja le “mi az amit szeretnénk elérni?” addig az utóbbi azt, hogy “hogyan mérjük, hogy elértük-e a célt?”.

Ha a megfelelő kérdéseket tesszük fel akkor jól körül tudjuk határolni a célkitűzéseket:

Mi a szervezet vagy a csapat számára az elsődleges?

Mik azok a célok amelyek a szervezet hosszú távú vízióját támogatják?

Milyen startégiai, csapat szintű vagy egyéni célokat határozzunk meg?

A kulcsfontosságú eredmények meghatározásához pedig ezekhez hasonló kérdéseken keresztül vezet az út:

Hogyan tudjuk számszerűsíteni a célkitűzést?

Hogyan kövessük nyomon a haladást?

Mikor mondhatjuk azt, hogy elértük a célkitűzést?

Az OKR rendszer az egy célkitűzéshez tartozó kulcsfontosságú eredmények számát ötben maximálja, de a gyakorlatban a legtöbbször két-három eredmény is elegendő szokott lenni. Az első OKR megalkotásakor nem garantált a siker és elképzelhető, hogy újra kell írni, vagy módosítani kell időközben, de a gyakorlattal egyre jobb minőségben állítjuk majd ezeket elő és egyre hatékonyabban használjuk őket.

Nézzük meg néhány példán keresztül, hogyan is épul fel egy OKR.

Célkitűzés: Az applikációnk a legfelhasználóbarátabb felülettel rendelkezzen a piacon

Kulcsfontosságú eredmények:

- A következő negyedévben legalább 4 szemmozgás követéses ügyfélinterjút végzünk el és dokumentálunk
- Az NPS kérdőív kitöltésének arányát megnöveljük 60%-ra
- Az NPS eredményünket megnöveljük 73%-ról 81%-ra

Ahogyan a fenti példából is látszik az OKR nyitva hagyja a lehetőséget a csapat vagy a szervezet számára, hogy hogyan éri el a célokat - jelen esetben a felhasználói felület minőségének növelését - azonban egyértelmű mérőszámokat rendel a cél mellé, hogy a változások hatását a célra nyomon lehessen követni.

Vizsgáljuk meg egy másik példában egy B2B termék bevezetéséhez kapcsolódó OKRt:

> Célkitűzés: Az új ügyfelek integrációja során első osztályú ügyfélélményt biztosítunk
>
> Kulcsfontosságú eredmények:
>
> - Az integráció alatti kapcsolatfelvételek számát 5-re emeljük
> - A bevezetés utáni kapcsolatfelvételek számát 15%-al növeljük
> - A bevezetéshez kapcsolódó ügyfél elégedettségi kérdőív eredményét “jó”-ról “kiváló”-ra emeljük

Ebben az esetben is tág teret kap a csapat arra, hogy automatizálja-e a folyamatot, vagy az ügyfélszolgálati rendszer továbbfejlesztésével éri el a célt.

Az OKR-ek kiértékelését - azaz annak vizsgálatát, hogy sikerült-e megvalósítani a célkitűzést - többféle módszerrel is végezhetjük. Ha egyszerűen akarjuk megoldani, akkor egyszerűen egy igen/nem válasszal is lezárhatjuk a kérdést, de ezzel elég sok információt elvesztünk a vizsgálat során.

A Google által is használt módszer alapján a kulcsfontosságú eredményeket kifejezetten ambiciózus módon határozzák meg. Az elért eredményt pedig egy 0.0 és 1.0 közötti skálára helyezik el. A 0.0 és 0.3 közötti értékek azt jelentik, hogy nem sikerült megvalósítani a célt. 0.4 és 0.6 közötti érték jelentése, hogy történt haladás a cél irányába, de nem teljesült maradéktalanul. 0.7 és 1.0 között pedig teljes értékűnek tekinthető a cél elérése. A gyakorlatban a 0.6 és 0.7 közötti értéket tekintik ideálisnak a 0.7 felettit pedig kiemelkedőnek.

Habár az OKR sokkal inkább objektív mérőszámokra épül, a kiértékelésnél sokkal fontosabb az okok feltárása akár siker akár kudarc esetén, mintsem az értékek

merev követése. Ne essünk itt sem abba a csapdába, hogy kobra bőrt nyúzatunk a csapatunkkal.

Bizonyítékokon Alapuló Menedzsment (Evidence Based Management)

> *» Az Evidence Based Management (EBM) egy tapasztalatiságon alapuló megközelítés amely segít a szervezeteknek abban, hogy növeljék az ügyfélélményt, fejlesszék a teljesítőképességet és üzleti eredményeket bizonytalan környezetben. Keretet biztosít a szervezetek számára, hogy fejlesszék a képességüket, hogy értéket teremtsenek egy bizonytalan világban, keresve a stratégiai célok felé vezető utat. «*

Az EBM azt a korábban már tárgyalt problémát próbálja meg megoldani, hogy a szervezet által előállított érték mérése során rendszeresen nehézségekbe ütközünk, ugyanakkor ha nem megfelelő vagy nem elégséges mértékű értéket állítunk elő akkor nem leszünk sikeresek a piacon.

A Bizonyítékokon Alapuló Menedzsment négy kulcsfontosságú értékterületen (key value areas) ad választ erre a dilemmára. Ezek a területek vizsgálják:

- A szervezet céljait (Meg Nem Valósult Érték / Unrealized Value)
- A szervezet jelenlegi állapotát a célokkal összevetve (Jelenlegi Érték / Current Value)
- A szervezet válaszidejét az érték szállításban (Piacra Lépési Idő / Time-to-Market)
- A szervezet hatékonyságát az érték szállításban (Innovációs Képesség / Ability-to-Innovate)

A fenti négy érték területre fókuszálva a szervezet meg tudja határozni jelenlegi helyzetét és haladásának irányát. Mindegyik értékterület a szervezetben előállított

értékre vagy az érték előállításának képességére fókuszál. Az érték szállítása fontos tényező (Jelenlegi Érték), de a szervezetnek reagálnia kell a piaci- és technológiai környezet változásaira (Piacra Lépési Idő) úgy hogy közben képes legyen a folyamatos megújulásra (Innovációs Képesség), továbbá megvalósítsa a hosszú távú céljait (Meg Nem Valósult Érték).

Jelenlegi Érték (Current Value)

A Jelenlegi Érték a szervezet pillanatnyi állapotát vizsgálja: milyen értéket nyújt a szervezet az ügyfeleinek és az érdekelt feleknek. Nem veszi figyelembe mi lehetséges a jövőben. A Jelenlegi érték vizsgálatához olyan kérdésekre kell választ adnia a szervezetnek, mint:

> Mennyire elégedettek az ügyfeleink? Növekszik vagy csökken az elégedettségük?
>
> Boldogok a szervezet tagjai? Növekszik vagy csökken a boldogságuk?
>
> Mennyire derűlátóak a befektetők? Nő vagy csökken a derűlátásuk?

A Jelenlegi értékhez kapcsolódó mutatók amelyek segíthetnek kialakítani a megfelelő képet erről az érték területről (a lista néhány példát tartalmaz, de nem leíró jellegű):

> Alkalmazottankénti bevétel - a hatékonyság jó indikátora lehet az összes bevétel aránya az összes alkalmazotthoz. Vegyük azonban figyelembe, hogy az érték iparáganként eltérő lehet.
>
> Termék költség arány - A termék vagy szolgáltatás létrehozásakor és üzemeltetéskor felmerülő összes költség aránya a termék vagy szolgáltatás által termelt bevétellel.

Alkalmazottak elégedettsége - Általában kérdőíves formában készülő felmérés amely az alkalmazottak elkötelezettségét, energiaszintjét és lelkesedését méri

Ügyfél elégedettség - számos formában mérhető az ügyfelek viszony a termékhez, a korábban ismertetett NPS egy lehetséges mérőszám

Használati statisztikák - A termék vagy szolgáltatás vagy annak egyes területeinek használati gyakorisága vagy a használat hossza

Meg Nem Valósult Érték (Unrealized Value)

A Meg Nem Valósult Érték magába foglal minden olyan értéket amely előáll amikor a szervezet teljesíti az összes ügyfél vagy felhasználói igényt. Ennek az értékterületnek a vizsgálata lehetőséget biztosít a szervezetnek arra, hogy azonosítsa az ügyfelek elvárásai és tapasztalatai közötti különbségeket. Ennek áthidalásával újabb érték teremtésére nyílik lehetőség. Ez a Meg Nem Valósult érték. Ahhoz hogy a szervezet elérje a Meg Nem Valósult Értéket a következő kérdésekre adott válaszokat kell értékelnie:

Képes-e a szervezet új értéket teremteni ezen a piacon vagy más piacokon?

Megtérül-e ha ezeket a kiaknázatlan lehetőségeket megvalósítjuk?

Szükség van-e további befektetésre a Meg Nem Valósult érték eléréséhez?

Hasonlóan a Jelenlegi Értékhez, a Meg Nem Valósult Érték is behatárolható a megfelelő mutatókkal (a lista itt sem teljeskörű):

Piaci részesedés - Mekkora az a rész a piacból amely jelenleg nem a terméké. Mekkora növekedési lehetőség van még a piacon, ha a termék jobban megfelel az ügyféligényeknek.

Ügyfél elégedettségi rés - Az ügyfelek jelenlegi tapasztalatai és elvárásai közötti különbség

Elvárt ügyfélélmény - Az ügyfelek által támasztott elvárások a jövőbeli ügyfél élménnyel kapcsolatban

Piacra Lépési Idő (Time-to-Market)

A Piacra Lépési Idő a szervezet képessége arra, hogy terméket, szolgáltatást vagy új fejlesztést vigyen a piacra.

A Piacra Lépési Idő vizsgálatára azért van szükség, hogy a lehető legrövidebb legyen az az idő amíg a szervezet értéket szállít. Ha ezt nem tudja folyamatosan alacsonyan tartani, a szervezetnek azzal kell majd szembenéznie, hogy bizonytalanná válik a fenntartható értékszállítási képessége a jövőben. Ha a Piacra Lépési Időre szerente a szervezet hatást gyakorolni, akkor a következő kérdésekre kell választ találnia:

Milyen gyorsan képes a szervezet tanulni az új információk és tapasztalatok alapján?

Milyen gyorsan képes a szervezet alkalmazkodni a megszerzett információ alapján?

Milyen gyorsan képes a szervezet új ötleteket tesztelni valódi ügyfelekkel?

A Piacra Lépési Idő is leírható bizonyos metrikákkal:

Az összeállítás és integráció gyakorisága - Az adott idő alatt összeállított, tesztelt és integrált szoftver egységek száma.

A kiadás gyakorisága - Milyen rendszerességgel jelenik meg a szervezet által készített szoftver a piacon. Naponta? Hetente? Havonta? Negyedévente? Ez

az érték megmutatja, hogy milyen időközönként tud a szervezet reagálni az ügyféligényekre.

A kiadás stabilizálásának hossza - Mennyi idő telik el (pl. hibajavítással) aközött, hogy a fejlesztők kiadásra késznek minősítik a terméket és amíg ténylegesen kiadásra kerül a felhasználóknak. Ha ez az érték túlságodan megnő az a fejlesztés beli hiányosságokra hívhatja fel a figyelmet

Hibajavítás átlagos ideje - Az az időtartam ami egy termékben feltárt hiba észlelése és javítása között telik el. A szervezet reakció képességét mutatja meg a hibák javításában.

Fejlesztési idő - Azt az időintervallumot írja le amíg az ötlettől olyan állapotig jut a termék ahol már értéket szállít a felhasználóknak.

A szolgáltatás helyreállításának ideje - Egy szolgáltatás kiesése és az újabb teljes rendelkezésre állás között eltelt idő.

Tanulási idő - Egy termék ötletének felvázolása, fejlesztése és szállítása között eltelt idő. Akkor mérjük a periódus végét amikor már érkezik visszajelzés a termékről a felhasználóktól.

A Piacra Lépési Idő javítása megnöveli annak gyakoriságát amilyen időközönként a szervezet képes növelni a Jelenlegi Értéket.

Innovációs Képesség (Ability-to-Innovate)

Milyen hatékonyan tud a szervezet új képességeket létrehozni amelyekkel jobban ki tudja elégíteni az ügyféligényeket. Az Innovációs Képesség létrehozásával és fenntartásával a szervezet folyamatosan új megoldásokat tud kidolgozni, hogy növelje a hatékonyságát és elégedetté tegye az ügyfeleit. Az Innovációs Képesség vizsgálatakor a következő kérdéseket teszi fel a szervezet:

Mi akadályozza a szervezetet, hogy új értéket hozzon létre?

Mi akadályozza meg a felhasználókat vagy ügyfeleket abban, hogy részesüljenek az innováció hasznából?

Az Innovációs Képesség feltérképezéséhez is segítségünkre lehetnek mérőszámok:

Az innováció arány - Az összes ráfordításból mennyi azoknak az aránya amelyeket kifejezetten új képességek kialakítására fordít a szervezet.

Terméken töltött idő - Az összes rendelkezésre álló időből mennyit tölt a szervezet a termékkel és az érték előállítással.

Termékverziók száma - Hány különböző verziót tart karban és üzemeltet a szervezet egyszerre.

Technikai adósság - Mennyi többlet időt fordít a szervezet arra, hogy teszteljen és hibát javítson korábbi kapkodva vagy nem megfelelő minőségben létrehozott termék egységek miatt.

Kontextus váltások gyakorisága - Milyen gyakran kell a fejlesztési folyamatot megszakítani hívásokkal, megbeszélésekkel vagy feladatok közötti gyors és gyakori ugrálással.

Az Innovációs Képesség növelése segít abban a szervezetnek, hogy minél hatékonyabban működjön és garantálja, hogy az elvégzett munka eredményeként a létrehozott termék minél magasabb értéket képviseljen az ügyfelek számára.

A célok elérése kis lépésekben

Az első lépés egy szervezet stratégiai céljának eléréséhez a Jelenlegi Érték meghatározása. Ha a szervezet célkeresztjében a Meg Nem Valósult Értékkel kapcsolatos stratégiai cél elérése van, akkor a Jelenlegi Érték fogja megmutatni, hogy

honnan indulunk. Természetesen egy új terméknél a Jelenlegi Érték 0. A fejlődési területek azonosításához meg kell érteni az Innovációs Képességet és a Piacra Lépési Időt is.

A Kísérleti Ciklus segít a szervezetnek abban, hogy elérje a következő célját és végül megvalósítsa a stratégiai célkitűzését is, úgy hogy kísérletnek nevezett apró és folyamatosan mért lépésekben halad előre. Ezek a lépések a következők:

Hipotézis megalkotása a fejlődéshez. A tapasztalatok alapján egy ötlet létrehozása amely segít a szervezetnek a következő cél elérésében, úgy hogy meghatározzuk honnan fogjuk tudni, hogy mérhető módon elértük a célt.

A kísérletek lefolytatása. A változtatások elvégzése amelytől a fejlődést reméljük és adatok gyűjtése amely alátámasztja vagy megcáfolja a hipotézisünket.

Az eredmények megfigyelése. A mérések alapján javítja-e az eredményeket a változtatás? Nem minden változtatásnak lesz ez az eredménye. Lesznek olyan változtatások amelyek rontanak a helyzeten.

A célok vagy a megközelítés kiigazítása a tanultak alapján. Mind a célok, mind a fejlesztést szolgáló kísérletek fejlődni fognak ahogy a szervezet egyre jobban megismeri az ügyfeleket, a versenytársakat és a saját képességeit. A célok változhatnak mert változik a környezet és a megvalósításhoz szükséges megoldásokat folyamatosan újra kell gondolnia a szervezetnek. A köztes célok megfelelőek voltak? A Stratégiai Cél még mindig releváns? Egy köztes cél elérésekor ki kell tűzni az új köztes célt. Ha nem sikerült elérni, akkor el kell dönteni, hogy a szervezet tartja-e tovább az irányt, megálljon vagy megváltoztassa azt. Ha a Stratégiai Cél már nem releváns alkalmazkodnia kell a szervezetnek vagy új Startégiai Célt kell kitűznie.

A hipotézisek felállítása, a megfigyelés és mérés, valamint a tapasztalatok alapján a célok ellenőrzése és módosítása az agilis működés alapvető elemei. Az Evidence Based Management abban segít a szervezetnek, hogy a munkafolyamatokat átláthatóvá és egyértelművé teszi.

Agilitás a szervezetben

Az hogy egy termék mennyire lesz sikeres. Mennyi értéket állít elő nem is olyan egyértelmű kérdés ahogyan azt az előző alkalommal megismerhettük. Az értékteremtés nem csak az ügyfél felé irányuló folyamat, hanem a szervezeten belülre is irányul. A sikerben számos szereplő érdekelt a szervezetben, de az ő elképzeléseik nem feltétlenül egyeznek egymással vagy éppen a Product Owner elképzeléseivel. A hatékony működéshez szükséges, hogy a kiváló Product Owner egyeztetni tudja a különböző érdekeket, kezelni a felé érkező megkereséseket és összességében eredményesen menedzselje az érdekelt feleket.

Az érdekelt felek menedzselése (stakeholder management)

Ha nem szeretnénk, hogy a termékünk a süllyesztőben végezze, mert nem teljesíti az üzleti célokat vagy az ügyfelek igényeit akkor Product Ownerként nem engedhetjük meg magunknak, hogy ne működjünk szorosan együtt a szervezetben azokkal akik érdekeltek a termék fejlesztésében.

Ahhoz hogy ez az együttműködés hosszú távon is sikeres legyen először is azonosítanunk kell, kik is ezek az egyének a szervezetben és milyen szerepkörben dolgoznak, majd ezek után egy hatékony kommunikációs és menedzsment tervet kell kidolgozni amelyben részletezzük, hogy milyen kapcsolatot építünk ki velük és hogyan kezeljük őket.

Az együttműködés minősége jelentősen befolyásolja a termék sikerét.

Nézzük meg hogyan alakítsunk ki hatékony együttműködést velük.

Érdekelt felek agilis környezetben - kik ők?

Érdekelt fél lehet egy személy, egy csoport, egy másik szervezet akire hatással van a termék sikere. Érdekeltek tehát abban, hogy sikeres legyen a termék és folyamatosan magas értéket állítson elő.

Az érdekelt feleket két csoportra oszthatjuk: szervezeten belüli érdekeltek felekre és szervezeten kívüli érdekeltek felekre.

A szervezeten belüli felek érdekeltsége a közvetlen kapcsolatból fakad. A külső felek nem feltétlenül tudnak arról, hogy a szervezet egy terméken dolgozik, de a jövőben kapcsolatba kerülhetnek vele.

Íme néhány példa a belső érdekelt felekre:

- Üzletági vezetők
- Értékesítési vezetők
- Fejlesztők, például a mérnökök, dizájnerek és ügyfélélmény tervezők
- A szervezet részlegei
- Marketing vezetők
- C szintű vezetők
- Jogi szakemberek
- Más Product Ownerek

És néhány példa a külső érdekelt felekre:

- Felhasználók
- Ügyfelek
- Beszállítók
- Részvényesek

- A termék köré épült közösség

Az érdekelt felek menedzselésében a fő hangsúly a belső felekre irányul. A külső felek közül a legfontosabbak természetesen a (leendő) ügyfelek.

Amikor döntéseket hozunk a termék fejlesztésével kapcsolatban akkor az érdekelt felek hangja lesz a legfontosabb számunkra, mivel ők a fejlesztéshez kapcsolódó igények fő forrásai. Ha nem alakítunk ki megfelelő kapcsolatot velük és nem tartjuk azt folyamatosan fent, akkor azzal azt kockáztatjuk, hogy elpazaroljuk a rendelkezésre álló erőforrásokat, mivel nem a megfelelő terméket fejlesztjük és ezáltal nem érjük el a stratégiai céljainkat. A megfelelő kapcsolat kialakítása ezzel szemben erősíti az átláthatóságot és segíti a közös megértést.

A megfelelő együttműködés kialakítása az érdekelt felekkel komoly mértékben csökkenti a kockázatot is. Ha aktívan bevonjuk őket a termék fejlesztésébe és rendszeresen adnak visszajelzést az aktuális állapotról, akkor akkor ezzel az értékteremtés részeseivé válnak és a fejlesztés hatékonyságát is növelhetik.

Az aktív kapcsolat segítségével a fejlesztés útjában álló akadályok is könnyebben elmozdíthatóak, így csökkenhet a piacra lépési idő is. Ha egyenesen kommunikálunk a szervezetben az érdekelt felekkel, akkor pedig új szemszögből ismerhetjük meg a termékünk működését és az általa előállított értéket is.

Hogyan építsük ki ezt a kapcsolatot az érdekelt felekkel?

A megfelelő kapcsolatok kiépítéséhez először az lesz a feladatunk, hogy azonosítsuk az érdekelt feleket a szervezetben, majd meghatározzuk, hogy milyen mértékű az érintettségük a termék sikerében, mennyire aktívan szeretnénk a fejlesztési folyamatban belefolyni valamint, hogy mennyire befolyásosak és ezen tulajdonságok alapján csoportokba rendezzük őket.

A kulcsfontosságú felek azonosítása

Gyűjtsük össze az összes olyan szereplőt a szervezetünkből akikre hatással lehet a termékünk állapota és teljesítménye a piacon, akinek bármilyen hatalma van a termék fejlesztés befolyásolására és aki érdekelt abban, hogy sikeres legyen a termék. Az azonosításhoz segítségünkre lehetnek a következő kérdések:

> Ki az akire hatással lehetnek a termékkel kapcsolatos döntések?
>
> Ki az akinek személyes érdekes is fűződik ahhoz, hogy sikeres legyen a termék?
>
> Ki az aki segíthet nekünk abban, hogy jobb termékkel jelenhessünk meg a piacon?
>
> Kell-e egyeztetni a jogi részleggel a termék kapcsán?
>
> Ki az aki lassíthatja a fejlesztési folyamatot?

Akár a kérdések mentén akár más módszerekkel azonosítottuk az érdekelt feleket, létezik egy listánk arról, hogy kik ők a szervezeten belül.

Vizsgálat és megértés

Ha azonosítottuk az érdekelt feleket, akkor meg kell határozni, hogy mik azok az elvárások, célok és igények amelyekkel rendelkeznek, hogy jobban megértsük milyen hajtóerők mozgatják őket. Ezek alapján pedig kialakíthatjuk a fontossági sorrendet közöttük. Ehhez meg kell értenünk, hogy a termék milyen hatással van a mindennapi működésükre. Milyen elvárásokkal rendelkeznek? Mi az ami motiválja őket? És azzal sem árt tisztában lennünk, mik azok a dolgok amelyek kifejezetten zavarják őket a termékkel kapcsolatban. Előfordulhat az is, hogy bizonyos érdekek mentén konfliktusok is kialakultak vagy kialakulhatnak az érdekelt felek között. Készüljünk fel ezek kezelésére.

A Stakeholder Canvas egy olyan eszköz, amellyel megkönnyíthetjük a fenti vizsgálatokat és jobban megérthetjük vele az egyes érdekelt felek viselkedését. A következőkben egy példán keresztül megnézhetjük, hogy milyen elemekből épül fel egy stakeholder canvas. Ezt a saját tapasztalatok alapján természetesen nyugodtan lehet bővíteni és még részletesebben leírni az érdekelt feleket a szervezetn belül.

A vizsgált stakeholder: közösségi média menedzser

Ki ő és mit csinál? A szervezeten belül dolgozik és az a feladata, hogy minél több leendő ügyféllel megismertesse a terméket

Mik a céljai? Szeretne minél több információt megtudni az ügyfelekről, hogy hatékonyabban tudja célba juttatni a termékkel kapcsolatos üzeneteket a megfelelő csatornákon keresztül.

Mi a legnagyobb akadály azelőtt, hogy ezt a célt elérje? Nincs kellő ismerete arról, hogy kik a jelenlegi ügyfelek és kik azok akik a jövőben használni fogják a terméket. Nem ismeri, hogy az ügyfelek milyen igényeket, elvárásokat támasztanak a termékkel kapcsolatban. Ahhoz, hogy megtalálja az ügyfelek által használt csatornákat és ott célba juttassa a termékkel kapcsolatos üzenetet több információra van szüksége az ügyfelekről.

A Stakeholder Canvas létrehozása minden érdekelt félről egy átfogó képet biztosít arról, hogy kikkel kell kommunikálnunk a jövőben a termék kapcsán és ők milyen hajtóerők mentén reagálnak majd az elképzeléseinkre.

Osztályozás - befolyás mátrix

Tudjuk, hogy kik az termék fejlesztésében érdekelt felek és már azt is, hogy mi mozgatja őket. Ha megfelelő prioritást alakítunk ki közöttük, akkor összhangba tudjuk hozni a termék stratégiánkat az egyes érdekelt felek stratégiai céljaival. Az azonosított érdekelt felek között lesz olyan aki kellő hatalommal rendelkezik, hogy

akár akadályozza a fejlesztés haladását, lesz aki egyszerűen érdeklődő a fejlesztéssel kapcsolatban, és olyanok is lesznek akik szinte érdektelenek. Ahhoz hogy osztályozni tudjuk őket ismernünk kell a befolyásukat, a szervezetben betöltött szerepüket, érdekeiket és azt is, hogy mennyire tudnak a rendelkezésünkre állni.

Ehhez az osztályozáshoz a Stakeholder Mapet használhatjuk eszközként. Az eszköz az egyes érdekelt feleket a szervezeten belüli befolyásuk és a termék iránt tanúsított érdeklődésük alapján értékeli. A Stakeholder Map egyrészt segít ráirányítani a fókuszt a legfontosabb érdekelt felekre másrészt az alapja a feléjük irányuló kommunikációs tervnek. A Stakeholder Map kialakításánál fontos, hogy minden érdekelt felet abba a kategóriába helyezzünk el amelybe az összegyűjtött információk alapján ténylegesen tartozik és ne pedig abba, ahol szívesen látnánk.

Nézzük meg néhány példán keresztül hogyan épül fel a Stakeholder Map. A két dimenzió amely segíti az osztályozást a befolyás és az érdekeltség.

Magas befolyás, magas érdekeltség - Ezek azok az érdekelt felek akik a kulcsfontosságú döntéseket hozzák meg a szervezetben. A termékre is komoly hatást tudnak gyakorolni, akár azért mert támogatják a sikerét, akár azért mert akadályozzák a fejlesztés haladását. Ezek miatt szoros együttműködésre van velük szükség: részletesen kell ismernünk az elvárásaikat és igényeiket és ezzel párhuzamosan kell a termékkel kapcsolatos döntéseket meghozni.

Tipikusan ilyen érdekelt fél lehet egy tulajdonos vagy egy nagyobb szervezetben egy igazgató aki azt a részleget vezeti ahol a terméket fejlesztjük.

Magas befolyás, Alacsony érdekeltség - Ugyan az ebbe a kategóriába eső érdekelt felek, nem fognak aktívan részt venni vagy beavatkozni a termék fejlesztésébe, de egyébként nagy befolyással rendelkeznek a szervezeten

belül, ezért érdemes képben tartani őket és törekedni arra, hogy elégedettek legyenek.

Az ügyfélszolgálat vezetője például egy ilyen szereplő lehet. Közvetlenül nem fog beavatkozni a termék fejlesztésébe, de ha megnövekszik a terhelés az ügyfélszolgálaton egy új termék funkció működése miatt akkor élni fog a hatalmával, hogy minél hamarabb változást érjen el a termék működésével kapcsolatban.

Alacsony befolyás, magas érdekeltség - a termék fejlesztésével kapcsolatos döntésekre nincs vagy alacsony a ráhatásuk, de hatással van rájuk a termék működése vagy teljesítménye. Emiatt érdemes rendszeres tájékoztatást nyújtani számukra a termék aktuális állapotáról és várható fejlesztési irányairól emailben vagy prezentációk formájában.

A jogi osztály GDPR specialistája nem rendelkezik túl nagy befolyással a szervezeten belül, de egy módosítás a regisztrációs felületen jelentős mennyiségű munkát generálhat számára. Érdemes ezért időben jeleznünk számára, ha ilyesmit tervezünk.

Alacsony befolyás, alacsony érdekeltség - ha egy érdekelt fél egy időpillanatban ebbe a kategóriába esik, akkor az nem jelenti automatikusan azt, hogy végig itt is marad. Ennek megfelelően a szervezet ezen tagjait figyelemmel kell követnünk, hogy változik-e a helyzetük: akár azért mert érdekeltek lesznek a termék sikerében, akár azért mert megnövekszik a befolyásuk.

Az osztályozás négy kategóriát különböztet meg, de az egyes kategóriákon belül eltérhet az, hogy hogyan kezeljük az érdekelt feleket. Az elégedettség egy magas befolyású, de alacsony érdekeltségű fél esetén jelentheti azt is, hogy időben piacra

tudunk lépni egy termék funkcióval, de akár azt is, hogy minden backlog refinementen részt tudjon venni.

Újraértékelés

Végre elkészült a Stakeholder Map! Itt az ideje, hogy átgondolt újra! Az idő előrehaladtával az egyes szereplők helyzete változhat a térképen, új szereplőket azonosíthatunk, vagy akár egy meglévő pozícióban lévő szereplő személye változik meg. Egy termék indulásakor sem feltétlenül tudjuk azonosítani az összes érdekelt felet. Ennek megfelelően rendszeres időközönként felül kell vizsgálnunk a Stakeholder Mapünket és ha szükséges el kell végeznünk a megfelelő kiigazításokat.

Egy agilis szervezetben a Product Owner feladata az, hogy kezelje az érdekelt feleket mind a szervezeten kívül, mind a szervezeten belül. Delegálhat feladatokat másoknak, de a felelősség mindvégig az övé marad.

Koordináció az érdekelt felekkel

A koordináció során jó kapcsolatokat építünk ki, elmélyítjük ezeket a kapcsolatokat és folyamatosan magas minőségben tartjuk fenn a közös viszonyunkat azokkal akik érdekeltek a termék sikerében és a legnagyobb befolyással bírnak a szervezeten belül. A megfelelő kommunikációs csatornák kialakítása alapvető fontosságú az érdekelt felek igényeinek-, céljainak- és siker kritériumainak megértéséhez. Egy megfelelő kommunikációs terv kialakításával még az előtt meg tudjuk válaszolni a felmerülő kérdéseket, hogy azokat az érdekelt felek fogalmazzák meg. Első lépésként érdemes azt megtudnunk, ki milyen formát részesít előnyben a kommunikáció kapcsán.

Kommunikációs terv

Az érdekelt felek kommunikációs igényeinek megismerésével lerakhatjuk egy hatékony kommunikációs terv alapjait. A korábban elkészített Stakeholder Map alapján meg tudjuk vizsgálni, hogy melyik érdekelt féllel milyen kommunikációs csatornát építsünk ki, milyen a terméket érintő témákban kapjanak információt és hogy milyen gyakran van szükségük az információkra.

A kommunikációs tervünk tartalmazza azt is, hogy milyen formában adjuk át az információkat az egyes érdekelt feleknek. Meghívhatjuk őket a sprint reviewra, megírhatjuk a termékkel kapcsolatos fejleményeket emailben, készíthetünk videót a termék új funkcióiról vagy akár bemutathatjuk az aktuális piaci teljesítményt egy interaktív táblázatban.

Amikor az érdekelt felekkel kommunikálunk, akkor vegyünk figyelembe néhány ökölszabályt. Először is, a kommunikáció nyújtson értéket az érdekelt felek számára. Azt kommunikáljuk ami az adott fél számára fontos, vagy amivel kapcsolatban korábban elvárásokat fogalmazott meg. Készüljünk arra, hogy az idő előrehaladtával ezek az igények és elvárások változhatnak a piaci változások hatására vagy azért mert egyre jobban megismerik a felek a terméket.

A szükséges mennyiségű információval lássuk el az érdekelt feleket. A közösségi média marketinggel foglalkozó kollégánknak fontos lesz az az információ, hogy mi lesz a következő termékverzió tartalma és mikor tervezzük a kiadását, de kevésbé fogja érdekelni az infrastruktúra költsége ellentétben a kontrollerrel aki a büdzsét tartja kézben.

Használjunk megfelelő nyelvezetet amikor az érdekelt felekkel kommunikálunk. Kerüljük a szakmai zsargont, fogalmazzunk egyszerűen a technikai részletek kapcsán, ha egy új csapattag felvételéről beszélünk a kiválasztást végző szakemberrel, de ismertessük részletesen a javasolt megoldások részleteit ha

szeretnénk a fejlesztési részleg vezetőjét megnyerni szponzorként egy új termék piaci bevezetéséhez.

Minden érdekelt féllel olyan kommunikációs csatornát alakítsunk ki ami számára is megfelel és elősegíti a hatékony információ átadást. A magas befolyással és érdekeltséggel rendelkező feleket hívjuk meg minden sprint reviewra, míg az alacsony befolyással és érdekeltséggel rendelkezőeket elég emailben tájékoztatnunk, ha elértünk egy komoly mérföldkövet.

Legyen egy transzparens módon elérhető dokumentum amely bármely érdekelt fél számára rendelkezésre áll és átláthatóvá teszi a fejlesztési folyamatot valamint be tudja mutatni a termék aktuális állapotát.

Hogyan vonjuk be a Scrum Csapat működésébe az érdekelt feleket?

Főleg a termék induló fázisában kulcsfontosságú, hogy a kiemelt érdekelt feleket bevonjuk, hiszen javaslataikkal vagy akár kérdéseikkel jelentősen elősegíthetik, hogy jobban megismerjük a termékkel kapcsolatban felmerülő igényeket.

A sprint reviewk során biztassuk arra az érdekelt feleket, hogy adjanak visszajelzést, mondják el a látottak alapján milyen irányokat látnak a fejlesztés jövője tekintetében.

Szervezzünk rendszeres megbeszéléseket az érdekelt felekkel, hogy új lehetőségeket tárjunk fel a termék jövőjével kapcsolatban.

Akadályozó tényezők

Vizsgáljuk meg azokat a tényezőket amelyek akadályozhatják, hogy megfelelő kapcsolatot alakítsunk ki az érdekelt felekkel.

Alakítsunk ki olyan csatornákat ahol azok az érdekelt felek is elérik a számukra szükséges információkat akikkel kevésbé aktív kapcsolatot ápolunk. Ez lehet egy

tábla ahol vezetjük a termék backlogot vagy egy szoftveres megoldás is. Ennek hiányában az átláthatóság csorbát szenved és emiatt különböző elképzelések alakulhatnak ki a termékkel és aktuális állapotával kapcsolatban valamint megnövekszik azoknak a találkozóknak a száma, amelyek a kialakult helyzetet próbálják meg tisztázni.

Főleg nagyobb szervezetekben felmerülhet az eltérő kulturális háttérből adódó különbség problémája. Az ázsiai kultúrákban tipikusan arc vesztéssel jár az, ha nemet mondunk valamire. Ennek könnyen az lehet az eredménye, hogy olyan ígéretet kapunk amely később mégsem teljesül vagy olyan fejlesztés indul el amelyre valójában nem volt szükség. A kulturális különbségek megismerésével és alkalmazkodással ez azonban kiküszöbölhető.

Az eltérő kultúra gyakran együtt jár az eltérő nyelvvel is. A nyelvi akadályok, pedig félreértésekre adhatnak okot. Ha lehetséges használjunk közös nyelvet a terméken dolgozók között és ha nem vagyunk biztosak, hogy a pontosan értettünk egy igényt inkább kérdezzünk vissza többször vagy kérjünk gyakoribb visszajelzést.

A szervezet méretétől függően kialakulhatnak információs silók. Ha ezek létrejönnek nem vagy csak költségesen áll majd rendelkezésre az a megfelelő minőségű információ mennyiség amely elengedhetetlen a termék sikeréhez. A silók legjobb ellenszere, ha rendszeresen egy asztalhoz ültetjük a különböző érdekelt feleket és segítjük az információ megfelelő áramlását a szervezeten belül.

Az érdekelt felek kezelésének alapja a nyílt és átlátható kommunikáció. Ha figyelembe vesszük az igényeiket akkor értékes információval lehetünk gazdagabbak és ez segítségünkre lehet, hogy jobb döntéseket hozzunk a termékkel kapcsolatban.

Veszélyes állatok a szervezetben

Az érdekelt felek néhány nem is olyan ritkán előforduló típusát Dean Peters rendezte kategóriákba és nevezte el az egyes kategóriákat vadállatokról. Az egyes állatok nevei angol betűszavakat takarnak, amelyek a kategória egy-egy jellemző tulajdonságát írják le. A következőkben ezeket a típusokat vizsgáljuk meg példákon keresztül és megismerhetjük azt is, hogyan tudjuk megszelidíteni a dzsungelek vagy a szavannák fenevadjait.

A Farkas - WolF (Works on Latest Fire)

Ha a Farkas a termék környékén ólálkodik biztosak lehetünk benne, hogy hamarosan mindenkinek mindent el kell dobnia és át kell váltanunk tűzoltó üzemmódba.

A termék fejlesztése során még akkor is kialakulhat technikai adósság, ha folyamatosan javítjuk a folyamatokat és finomítjuk a Definition of Donet. Így egy idő után ki kell alakítanunk azt az egyensúlyt amivel a Sprinteken belül teret adunk a hibák javításának az új funkciók fejlesztése mellett. Egy apró hiba először csak szikra vagy izzó parázs lesz, de ha elhanyagoljuk ezek kezelését akkor egyre nagyobb tüzek gyúlnak a termék körül amelyek a hosszú távú értékteremtést veszélyeztetik.

A Farkas azért lehet veszélyes, mert egyetlen dologra figyel: hogy oltsuk ezeket a tüzeket, függetlenül attól, hogy mennyire veszélyesek. Egyik égető sürgősségű dologról a másikra ugrál, még akkor is, ha ennek az az ára, hogy nem hozunk létre értéket. Ez nem csak az ügyfelek szempontjából problémás, hanem a Scrum Csapat azon képességét is redukálja, hogy új és innovatív megoldásokat hozzanak létre.

A tűzoltás célja a Farkas szemében az, hogy a legújabb problémával foglalkozzon mindenki. Így érzésre a problémák megoldódnak, de valójában a priorizálás hiánya csak tovább fokozza az égető problémák súlyosságát.

2003-ban a Knight Capital nevű tőzsdei kereskedéssel foglalkozó cég egy fordított algoritmust fejlesztett ami magas árfolyamon vásárolt és alacsony árfolyamon adott el. Mivel csak egy teszthez szerették volna használni ezt a funkciót egyszeri alkalommal ezért a néhány dolláros veszteséget vállalható kockázatnak tekintették. A funkciót más fontos dolgok miatt végül elfelejtették kivenni a kód bázisból így 9 évig része volt a terméknek, anélkül, hogy használták volna. Egészen addig amíg egy frissítés alkalmával az éles rendszernek is része lett, ahol automatikusan elindult az algoritmus és 45 perc alatt 460 millió dolláros veszteséget generált.

A problémát az okozta, hogy a felügyelet nélkül hagyott funkcióra más termékfunkciók is elkezdtek ráépülni így az éles kódbázis részévé vált. A folyamatos tűzoltásban nem jutott idő kellő tesztelésre és arra sem voltak felkészülve, hogy bizonyos mennyiségű veszteség után automatikusan leálljon a kereskedés. Az elnevezések konvenciója nem volt a Definition of Done része ezért a hiba kijavítása is rendkívül sokáig tartott. Nem volt napi gyakorlat a refaktorálás amelyben ez a kódrészlet eltávolításra kerülhetett volna.

Ha elhanyagoljuk a technikai adósságot, mert egy Farkas típusú érdekelt fél igényeinek próbálunk meg megfelelni akkor akár ekkora mértékű veszteséget is szenvedhet a szervezet.

A Farkas megszelidítésére két módszer javasolt. Elsőként, meg kell akadályozni, hogy az óhatatlanul előforduló tűzesetek komoly károkat okozhassanak. A vadonban tűzzárakat használnak a futótüzek megállítására: körbeássák például a burjánzó tűz területét, hogy az ne terjedjen tovább. Szoftveres környezetben ha nem minden felhasználó számára egyszerre lesz elérhető egy új funkció, vagy olyan időintervallumban végezzük el a frissítést amikor kevés felhasználó aktív esetleg

bevonjuk a felhasználók egy körét a tesztelésbe béta verziókkal minimalizálhatjuk az esetleges hibák által okozott károkat.

Másrészről folyamatosan időt kell szakítanunk arra, hogy dolgozzunk a technikai adósságokon és refaktoráljuk a meglévő kódot. A Product Owner szerepe ebben az, hogy megfelelő prioritást adjon az ehhez kapcsolódó feladatoknak.

Az Orrszarvú - RHiNO (Really High-value New Opportunity)

Dübörgést hallunk a távolból, majd berobog a Scrum Csapat mellé az Orrszarvú egy új, fantasztikus ötlettel, amit ha megvalósítunk biztosan megszerezzük ezt a nagy ügyfelet. Általában ezzel érkezik meg hozzánk a legtöbbször a marketing- vagy értékesítési oldalon dolgozó érdekelt fél. Még ha végül be is igazolódik az új funkció eredményessége ez a viselkedés nagyon rövidlátó és zavaró a csapat számára.

Miért veszélyes az Orrszarvú? Ha engedünk a nyomásának, akkor szembesülni fogunk azzal, hogy legtöbbször csak “egyszerű” kérései vannak. Szerinte. Ezekről rövid időn belül ki fog derülni, hogy egyáltalán nem egyszerűen megoldható felvetések. Másrészről egy új nem tervezett funkció beépítése a termékbe többlet feladatokkal jár a tesztelés és integráció területén is. Ha túlságosan egyedi az igény akkor pedig akár a párhuzamosan futó verziók számát és ezek fenntartására fordított erőforrások mennyiségét is megnöveli.

Az Orrszarvú kérései szinte soha nem arra irányulnak, hogy egy az ügyfelek széles körében meglévő problémára keresünk és adunk választ, hanem rendszerint konkrét megoldás szállítását várja el egy konkrét ügyfélre fókuszálva. Ha több ilyen funkciót toldozunk egymáshoz akkor egy Frankenstein Szörnyéhez hasonlító terméket fogunk előállítani ami egyszerre akar mindent megoldani, de végül semmit sem fog.

Azok az ígéretek amellyel az Orrszarvú megpróbálja befolyásolni a fejlesztés irányát legtöbbször mindenféle tervezést vagy garanciát nélkülöznek és az elpazarolt erőforrásokon felül bevételt sem termelnek.

> A ScaleFactor nevű vállalat azt ígérte a részvényesei számára, hogy kifinomult mesterséges intelligenciával támogatott könyvelői szolgáltatásokat nyújtanak leendő ügyfeleiknek. A megoldás fejlesztése helyett azonban a tőke összegyűjtésére helyezték a hangsúlyt és mivel fel kellett valami mutatni ezért titokban könyvelőket kezdtek el alkalmazni akik elvégezték a feladatot az ügyfelek számára. Mivel a koordináció hiányzott, illetve a fejlesztés a könyvelők munkáját kezdte el támogatni ezért rengeteg hiba került a rendszerbe mint szoftveres mind emberi oldalról. Az ügyfelek ezt tapasztalva hamar elpártoltak a cégtől a befektetőket pedig a hazugság leleplezése riasztotta el. 2020-ra cég elbocsátotta az alkalmazottait és a megmaradt tőkét visszafizette a befektetőknek majd megszűnt.

Jó hír, hogy az Orrszarvú is szelidíthető. A dübörgés ellenére meg kell értetnünk a szervezeten belül a termék vízióját és nem egyedileg felmerült igényekre fordítani a meglévő szűkös erőforrásokat, hanem arra az ügyfél csoportra akik számára az új funkció a legnagyobb értéket teremti.

Szánjunk arra időt, hogy megismertessük a technikai részleteket az Orrszarvú típusú érdekelt felekkel, így csökkenthetjük annak a kockázatát, hogy egyszerűbbnek lássák a megvalósítást a tényleges komplexitásnál és ezt már a leendő ügyfeleknek is így adják el.

Az új ötleteket nem kell azonnal elvetni. Kísérletezéssel igazolhatjuk azok sikerességét a piacon vagy gyorsan rájöhetünk, hogy valójában egy szűk kör számára tudunk csak kis mennyiségű értéket teremteni.

Product Ownerként ha határozott víziónk van és a kitűzött cél felé haladunk akkor bátran szembe állhatunk az Orrszarvúval.

A Víziló - HiPPO (Highest Paid Person's Opinion)

A Product Ownernek úgy kell meggyőzni az érdekelt feleket, hogy támogassák a termék vízió megvalósulását, hogy nincs pozícióból fakadó hatalma. Az üzleti döntéseket amely befolyásolja a termék fejlesztésére fordítható költségvetés nagyságát is igazgatói szinttől felfelé hozzák meg. Ha minden rendben megy, akkor ezek az érdekelt felek komoly segítséget adhatnak a termék értékteremtő képességéhez, hacsak nem változnak Vízilóvá.

A Vízilovak meg vannak arról győződve, hogy sem mások gondolataira sem adatokra nincs ahhoz szükség, hogy megerősítsék saját ösztönös elképzeléseiket a termékkel kapcsolatban. Saját tapasztalataikat többre tartják a szervezet többi tagjának szakértelménél ezért hajlamosak lesöpörni a tőlük érkező javaslatokat az asztalról.

A Vízilovak az üzleti környezetben sem veszélytelenek. Mivel nekik komoly pozícióból fakadó hatalmuk van - megszűntethetnek a munkaviszonyokat vagy dönthetnek a költségvetés egyes elemeiről - ezért nehéz nekik ellentmondani.

Mivel képesek átgázolni a szervezeten a saját ötletük kivitelezése érdekében a mellettük dolgozók elvesztik a lelkesedésüket az innováció iránt. Ezen felül olyan funkciók kerülhetnek a termékbe amelyek piaci létjogosultsága megkérdőjelezhető. Szélsőséges esetben az egész szervezet működése egy funkció gyárrá alakul - azaz csak a létrehozott funkciók mennyisége számít nem pedig az azok által teremtett érték.

> A Víziló típusú működésre kiváló példa Ron Johnson a JC Penney nevű amerikai kereskedelmi cég vezetője. Johnson korábban komoly sikereket ért el az Apple és a Target vezetői pozícióiban. A JC Penney élén a megérzéseire

hagyatkozva egy bonyolult rögzített árazási rendszert vezetett be. Hiába jelezték a szervezetből - fókuszcsoportos kutatásokkal alátámasztva - hogy az ügyfelek nagyon árérzékenyek és fogékonyak a leárazásokra, nem módosított az elképzelésein. Az eredmény: 25%-os forgalomcsökkenés és 17 hónap alatt 1 milliárd dolláros veszteség.

A Vízilovakat felnyergelni ugyan nem fogjuk, de a háziasításukra tehetünk kísérletet. Álljunk az oldalukra. Ha új ötlettel érkeznek akkor javasoljuk, hogy segítünk megerősíteni az elképzeléseiket a piacon. Ha a kísérletünk végül nem igazolja a Víziló ötletét akkor is azt fogja látni, hogy az ő érdekében cselekedtünk és jó szándékkal végeztük el a validációt.

Gyűtsünk hatalmas mennyiségű tényt és visszajelzést valós ügyfelektől. Ahogy a JC Penney példájából láttuk ez nem garancia a sikerre, de ezek nélkül lehetetlen lesz meggyőzni a Víziló típusú vezetőket.

Határozzuk meg közösen mit tekintünk sikernek. Formalizáljuk a Víziló ötleteit és rendeljünk mellé jük számszerűsíthető eredményeket amely alapján el tudjuk dönteni, hogy bevált-e a javaslatuk.

Zebra (Zero Evidence, but Really Arrogant)

Miközben a többi veszélyes állatról olvastunk, könnyen gondolhattunk arra, hogy azokban a pozíciókban mi biztosan nem viselkednénk így. Annál jobban ismerjük a terméket, hogy Orrszarvúként rontsunk rá. Adatok alapján hozzuk meg az alaposan megfontolt döntéseinket, nem úgy mint egy Víziló. Azonban a Zebra olyan állat amellyé akár egy Product Owner is könnyen válhat. Mivel a saját termékünket és területünket jól ismerjük előfordulhat, hogy piaci validáció nélkül tisztán szívből hozunk meg döntéseket. Ezek vagy igazolódnak később vagy nem, de mindkét esetben erőforrások felhasználásával járnak, így pazarlást is okozhatnak.

A Vízilóhoz hasonlóan a Zebra viselkedésével is az a probléma, hogy megalapozatlan döntéseket hozunk a termékkel kapcsolatban és így akár komoly lehetőségeket is elszalaszthatunk, mert éppen egy másik kétséges sikerű funkció fejlesztésével foglalkozunk.

> A Blockbuster nevű amerikai videókölcsönző cég vezérigazgatója John Antioco 2000-ben lehetőséget kapott, hogy megvásároljon egy másik hasonló profilú, de a digitalizáció felé nyitó céget 50 millió dolláros áron. Antioco azonban azt mondta, hogy ez a dotkom hisztéria csak egy múló szeszély és elutasította az üzleti ajánlatot. A másik céget Netflixnek hívták. Ma a Blockbuster már nem létezik.

Ha házi Zebrát akarunk tartani akkor érdemes konfrontálódni velük. Ehhez viszont szükségünk lesz arra, hogy megfelelő mennyiségű adattal tudjuk alátámasztani a Zebra által bedobott javaslat érvénytelenségét.

Gyűjtsünk szövetségeseket, akik hasonlóan hozzánk tényszerű alapokra szeretnék helyezni a termék jövőjét. A Zebra meg van győződve arról, hogy az ő véleménye a legjobb, de ha kellően sok hang igazolja ennek az ellenkezőjét akkor meghátrál.

Keretezzük át a Zebra javaslatait úgy, hogy az beilleszthető legyen a termékünk stratégiai irányvonalába. Ha a Zebra egója az akadály akkor adjuk meg neki azt, hogy nem sértjük meg, hanem támogatólag, de kicsit módosítva beépítjük a javaslatot a kijelölt irányvonalba.

Az arrogancia el tudja ködösíteni a döntéshozatali készségeinket. Product Ownerként még akkor is adjuk meg a lehetőséget egy ötlet tesztelésére, ha az első hallásra teljesen eszementnek tűnik. A legrosszabb ami történhet, hogy rövid időn belül kiderül, hogy tényleg nem működik. És ne feledjük mi is könnyen Zebrává válhatunk ha nem figyelünk oda.

A Sirály

A Sirály nem egy betűszóról kapta a nevét. Azok az érdekelt felek akik a Sirályhoz hasonlítanak, pontosan ugyanazt csinálják mint a tengeri madarak: keringenek a termék fölött különösebb kapcsolat nélkül, majd egyszer csak leszállnak hatalmas zajt és felfordulást csapnak maguk körül és továbbrepülnek, hogy aztán takaríthassanak utánuk.

Habár elsőnek nem mindig tűnhet így, de a Sirályok alapvetően jó szándékkal érkeznek. Szeretnék ha gyorsabban működnének a folyamatok, vagy felrázódna a csapat az ötleteiktől. Ezzel azonban az a baj, hogy nem ismerik a részleteket és elveszik a csapattól a felhatalmazást valamint a lehetőséget, hogy önállóan találjanak megoldást a problémákra.

Ha a Sirály típusú fejlesztési vezető egy napon azzal állít be a Fejlesztőkhöz, hogy “gondolkoztam a problémán amiről a múlt héten beszéltünk és hoztam egy megoldást rá”, akkor valójában segíteni akar, de mivel csak távolról ismeri a működést ezért egyáltalán nem biztos, hogy az általa írt kódot egyáltalán be lehet illeszteni a termékbe.

Egy Sirály típusú menedzsereket vizsgáló kutatás során azt tapasztalták, hogy a beosztottjaik 30%-al nagyobb eséllyel szenvednek szív- és érrendszeri betegségekben valamint nagyobb arányban demotiváltak és jutnak el akár a felmondásig. A jóindulat ellenére pontosan az ellenkező hatást váltják ki a szervezetben.

Ha egy Sirály köröz a fejünk felett két dolgot tehetünk. Egyrészről értsük meg a javaslatát és igyekezzünk minél hamarabb megmérni a piacon. A Sirályok általában tényleg szakértői a területüknek, ezért nem légből kapottak az ötleteik, csak nem ismerik a teljes kontextust. Másrészt ha megfelelő kommunikációs tervet építünk ki

és átláthatóvá tesszük a folyamatokat akkor csökkenthetjük a Sirály leszállásainak számát.

Product Ownerként a gyakori kommunikációval és az áttekinthetőség növelésével tudjuk megakadályozni a Sirályt abban, hogy újra és újra lecsapjon.

Az állatidomár eszköztára

Nincs egyszerű dolgunk, ha az a feladatunk, hogy az érdekelt felek igényeit figyelembe véve haladjunk a termék víziónk megvalósítása felé. Van azonban néhány olyan eszköz amellyel általánosságban sikeresek lehetünk állatidomárként.

Empátia

Könnyebben megérthetjük az érdekelt felek viselkedését, ha egy pillanatra átülünk az ő székükbe és megpróbáljuk az ő szemszögükből megvizsgálni a helyzetet. A motivációjuk megismerésével és megértésével könnyebben tudjuk összeegyeztetni azt a saját termék víziónk irányával.

Átláthatóság

Sokkal könnyebb úgy bevonni egy érdekelt felet a döntéshozatalba, hogy nem csak a termékkel kapcsolatos döntéseink miértjét, hanem a döntések mögött álló okokat is megérti. Éppen ezért érdemes ezeket kellő részletességgel dokumentálni és elérhetővé tenni a szervezet egésze számára. Legyen szó ügyfél interjúkról, mérési adatokról vagy konkrét döntésekről. Ezek segítségével elérheted, hogy az érdekelt felek a megfelelő információk birtokában tegyék meg a javaslataikat.

Technológiai ismeretek

Az érdekelt felek általában nem rendelkeznek kellő technológiai tudással a termék kapcsán. Különösen igaz ez a technikai adósság mértékére és arra, hogy az új

ötletek mekkora mértékben növelhetik azt. Ha Product Ownerként nem is szükséges ezt részletekbe menően értenünk, keressünk olyan Fejlesztőt aki segíthet nekünk abban, hogy az érdekelt felek ismereteit bővítsük, hiszen a későbbi fejlesztések hatékonyságát tudjuk ezzel növelni.

Kísérletezés

Mielőtt belevágnánk valami hatalmas világmegváltó ötlet megvalósításába érdemes tesztelnünk, hogy valójában akkora potenciált rejt-e és, hogy valóban akkora érdeklődést vált-e ki az ügyfelekből. Ha először az ügyfelek kisebb csoportján tesztelünk egy új funkciót (például béta- vagy A/B tesztekkel) akkor sokkal pontosabb képet kapunk arról, hogy a jövőben érdemes-e a terméket ebbe az irányba bővíteni. A piaci validáció kis lépésekben nagyszerű eszköz az érdekelt felek meggyőzésére.

Ügyfélközpontúság

Ha valójában szeretnénk, hogy az érdekelt felek az ügyfél szemüvegén keresztül nézzenek a termékre, akkor mutassuk meg, hogy Product Ownerként hogyan helyezzük az ügyfeleket a középpontba. Vegyünk egy valós problémát és vonjuk be őket egy lehetséges megoldás kidolgozásába. A műhelymunka során bemutathatjuk azt is, hogy nem csak egy megoldás létezhet egy problémára és mélységében megismertethetjük az ügyféligények mozgatórugóit.

Vízió

Akár a saját érveinket támasztjuk alá adatokkal, akár egy érdekelt fél által hozott bizonyítékokat vizsgálunk meg ne tévesszük szem elől, hogy azért gyűjtjük és elemezzük a tényeket, hogy a termék víziót támogató döntéseket hozzunk. Kétség esetén mindig ez legyen a legutolsó szempont amit megvizsgálunk. Ha az ötlet nem támogatja a víziót akkor nem lesz része a terméknek.

Product Ownerként nincs pozícióból adódó hatalmunk a szervezet más tagjai felett, azonban ez nem jelenti azt, hogy ne tudnánk olyan kapcsolatokat kialakítani amelyek segíthetnek abban, hogy kiváló terméket hozzunk létre. Alkalmazzuk a megismert eszközöket és értékes szövetségesekre tehetünk szert.

Agilitás a gyakorlatban

Az előző alkalmak során megismerkedtünk az agilis gondolkodásmóddal és a Scrum keretrendszerrel. Használható tudást szereztünk arról, hogy hogyan alkossuk meg egy termék vízióját, hogyan bontsuk le ezt kisebb elemekre és rendezzük sorba az elemeket a környezeti változásoknak megfelelően. Virtuálisan részt vettünk egy Sprint eseményein, ahol láthattuk milyen döntésekkel szembesülnek a Scrum Csapat tagjai. Eszközöket kaptunk a kezünkbe, hogy mérni tudjuk a hatékony működést a szervezeten belül. Megfelelő kapcsolatokat tudunk kiépíteni a termékünk sikerében érdekelt felekkel.

Ezzel az eszköztárral felvértezve készen állunk arra, hogy akár egy új termék fejlesztésében, akár egy már piacon lévő termék további sikereiben aktívan és hatékonyan részt vállaljunk.

Útravalóul nézzük még meg azt, hogy egy nagy méretű, a saját területén piacvezető vállalat hogyan ültette át a gyakorlatba az agilis működést. Lássunk néhány veszélyt, lehetséges csapdát amibe sok cég beleesik amikor megpróbál agilisan működni.

Hogyan csinálja a Spotify?

A Spotify céges filozófiája, hogy az egész szervezet magáénak érzi az agilis gondolkodásmódot. Nem csak a termékfejlesztéssel foglalkozó területek működnek agilisan, hanem a vállalat egésze. Mivel az agilis működés a vállalati kultúra szerves

része, ezért nincs is külön hangsúlyozva a szervezeten belül, egyszerűen csak megnyilvánul a mindennapok gyakorlatában.

A Spotify 2008-as indulásakor a Scrum keretrendszert használta a termékfejlesztéshez. A szervezet azonban drasztikus növekedésnek indult és azt tapasztalták, hogy a Scrum keretei amelyek leginkább a kis csapatos működést segítik akadályként jelentkeznek a skálázás során.

Érdemes megjegyezni, hogy több skálázott agilis keretrendszer is működik a gyakorlatban (SAFe, LeSS, Nexus) amelyek megismerése nem része a jelenlegi képzésnek. A Spotify nem egy saját keretrendszert alkotott - tehát a Spotify modell nem egy agilis keretrendszer. Ugyanakkor a vállalat működése jól példázza azt, hogy a keretek- és folyamatok gépies követése helyett hogyan lehet ténylegesen a megfigyelésekre és tapasztalatokra építve, kísérletezve továbbra is alkalmazkodóképes szervezeti struktúrát és gyakorlatokat kialakítani.

A Spotify csapatai amikor megtapasztalták a működés során a keretrendszer korlátait, úgy döntöttek, hogy nem vetik azt el teljesen, de opcionálissá teszik azok használatát. Az agilis működést fontosabbnak tartották, mint egy specifikus keretrendszert.

Itt tekintsünk ki egy kicsit újra a történetből, mert egy fontos tanulsága van a Spotify döntésének. Több vállalatnál - főleg olyanoknál amelyek a hagyományos működési modelljüket igyekeznek lecserélni az agilis működés valamilyen formájára - megfigyelhető, hogy a Scrum azon tulajdonsága amely képes gyorsan a felszínre hozni a működés hiányosságait szervezeti problémákat generál. Erre reakcióként a szervezet elkezdi a keretrendszert a saját ízlésének megfelelően átalakítani. A Spotify azért tudta ezt jól végrehajtani, mert egy még agilisabb működés volt a kitűzött cél, nem pedig a devolúció egy olyan működés irányába amely kevésbé alkalmazkodóképes. Ha olyan döntési helyzetbe kerülünk, hogy el kell hagynunk

vagy át kell alakítanunk egy agilis keretrendszer egyik elemét mindig tartsuk szem előtt az Agilis Kiálltványt és a mögötte húzódó alapelveket. Ha továbbra is megfelel a működésünk ezeknek akkor jó irányba tettünk lépést.

A Scrum Master szerepkört a Spotify a működési modell váltás után Agile Coach szerepkörre nevezte át. A döntés oka az volt, hogy a Scrum Master szerepben egyfajta módszertani működésért felelős személyt láttak és ezt szerették volna egy szolgáló vezető szerepkörré alakítani. Személyes véleményem az, hogy a Scrum keretrendszerben a Scrum Master a neve ellenére ugyanígy szolgáló vezetői szerepet tölt be, tehát ezzel a lépéssel a Spotify inkább egy anomáliát javított, mintsem valami újat alkotott.

A Fejlesztők helyét a Squadok (raj, osztag) vették át. A Squadok működésének alapja az önállóság volt. Egy squad számos tekintetben hasonlít a Scrum keretrendszerből megismert Fejlesztői csapatra. A Squadok is keresztfunkcionálisak - azaz képesek önállóan elvégezni minden olyan feladatot amely az ötlettől a megvalósításig tart és ezért felelősséget is vállalnak. Önszerveződőek - azaz a saját munkafolyamataikat úgy alakítják, hogy folyamatosan növelik a hatékonyságukat. Egy Squad létszáma általában 8 főnél kevesebb.

Az eltérés ott jelentkezik, hogy az egyes Squadok önálló hosszú távú misszióval rendelkeznek: például “A Spotify legyen az elsődleges választás ha zenét keres egy felhasználó” vagy a belső működést segítő “Folyamatosan álljon rendelkezésre az infrastruktúra az A/B teszteléshez”. Az önállóan működő Squadok ezek mentén határozzák meg, hogy mit és hogyan fejlesszenek és hogyan működjenek együtt a megvalósítás közben. Természetesen itt is vannak keretek: összhangban kell lenni a fejlesztésnek a hosszú távú termékstratégiával és a negyedéves rövid távú célokkal.

A felhatalmazás amelyet a Squadok kapnak rendkívül motiválóan hat az egyéni teljesítményre és a motivált csapattagok jobb terméket tudnak fejleszteni. Az

önállóság azt is jelenti, hogy a végrehajtáshoz a lehető legközelebb születnek meg a döntések: a Squadon belül. Minimális a szervezeten belül a döntéshozatal időköltsége, hiszen nincs szükség hosszas egyeztetésekre menedzserekkel vagy bizottságokkal. Ezzel párhuzamosan csökken annak az ideje is, hogy a Squadok egymásra várnak mivel függőségek alakulnak ki a fejlesztési területek között. Ez utóbbi nem csak a hatékonyságra, hanem a skálázhatóságra is jótékony hatást gyakorol. A Squadok önállóan hoznak ugyan döntéseket, de ezeket párhuzamba állítják a vállalat céljaival. Ha már Spotify akkor ezt leginkább úgy lehet leírni, mint egy szimfonikus zenekart: az egyes zenészek önállóan, saját kottából játszanak, de az összhangzást egy karmester irányítja így az egyes hangszerek összhangja egymást erősítve áll össze a darabbá.

A Spotify vállalati kultúráját úgy érthetjük meg jobban, ha két dimenzió mentén vizsgáljuk a működést: az összehangoltság és az önállóság tekintetében.

Azokban a szervezetekben, ahol **alacsony** mértékű az **összehangoltság** és **alacsony** mértékű az **önállóság** is a mikromenedzsment kultúra a jellemző. Nincs magasabb szintű elérendő cél, mindenki szinte vakon követi a magasabbról kapott utasításokat. Talán nem kell külön hangsúlyozni, hogy ez nem éppen agilis működésre utal, magunk sem szívesen dolgoznánk ilyen szervezetben.

Ha az **összehangoltság magas** , de az **önállóság alacsony** akkor a vállalat vezetésének ugyan van egy határozott elképzelése, hogy mi az a cél amit szeretne elérni, de azt is kőbe vési, hogy hogyan kell a problémát megoldani. Ez elképzelhető, hogy működőképes olyan környezetben ahol minden folyamat könnyen leírható és számon kérhető, de kevésbé alkalmas komplex problémák megoldására.

Alacsony összehangoltság és **magas önállóság** mellet a feladatokon dolgozó csapatok szabad kezet kapnak és motiváltak is, de semmi sem garantálja azt, hogy

az egyes csapatok munkájának összessége jó megoldást ad az ügyfelek igényeire, vagy hogy a szervezet hosszú távú céljai megvalósulnak.

Az a működési modell amelyre a Spotify törekszik **magas összehangoltság** mellett ad **magas önállóságot** a szervezeti egységeknek. Csak a hosszú távú célokat határozzák meg, de az ezt támogató megoldások kidolgozása az egyes csapatok feladata. Így a problémák komplexitásától függetlenül tudnak a stratégiai célokkal összhangban dolgozni.

Az önállóság magas szintje azt is eredményezi, hogy nagyon kevés sztenderd megoldást használ a Spotify a szervezeten belül. Legyen szó akár arról, hogy milyen szoftvert használ egy Squad az általuk írt kód szerkesztésére vagy tesztelésére vagy éppen arról, hogy milyen agilis keretrendszerben dolgoznak együtt. Ahelyett, hogy ezeket előírnák a csapatoknak a vállalati kultúra része az, hogy a jól működő módszereket önállóan átveszik egymástól a Squadok, ha látják, hogy valahol jól működik és így válik egy eszköz vagy módszer széles körben elterjedtté a vállalaton belül. Ennek a tapasztalatokon alapuló organikus átalakulásnak egy olyan egyensúlyi helyzet az eredménye amely egyszerre ad stabil alapot és rugalmasságot a mindennapi működésnek.

A Spotify architektúra több száz különböző alrendszerből áll össze amelyeket egymástól függetlenül fejlesztenek és telepítenek. Ezek a rendszerek természetesen kapcsolatban vannak egymással, de mindegyik egy specifikus igényre fókuszálva működik: van amelyik a keresési lehetőséget teszi elérhetővé az ügyfeleknek és van amelyik a kiszolgáló infrastruktúra megfigyelésére fókuszál.

Egy-egy ilyen rendszer általában egy Squad felelősségi körébe tartozik, de a Spotify egy “belső nyílt forráskódú” rendszert használ. Inkább a rendszerek megosztása a cél mintsem az, hogy szigorúan egy Squad legyen egy rendszer tulajdonosa. Ez a gyakorlatban úgy néz ki, hogy ha egy Squadnak szüksége van egy új funkcióra egy

másik Squad által fejlesztett rendszerben akkor első körben megkérik a másik csapatot, hogy be tudják-e ütemezni ezt a fejlesztést a saját rendszerükbe. Ugyanakkor, ha a másik csapatnak nincs elég kapacitása akkor az első csapat nem fog várni, hanem elvégzi a szükséges módosításokat a másik csapat kódjában amit aztán a másik csapat jóváhagy. Ez a működés nem csak az önállóságot és a felelősségvállalást növeli a szervezeten belül, hanem a rendelkezésre álló tudást is széles körben képes elterjeszteni.

Az ilyen típusú működés alapja, hogy rendkívül szerves része a vállalati kultúrának a kölcsönös tiszteletadás. A kollégák rendszeresen adnak pozitív visszajelzést a többiek munkájára és csak ritkán fordul elő, hogy a saját eredményekkel büszkélkednek. Összességében mindenki az ajtón kívül hagyja a saját egóját mikor belép a kapukon. Azok akik frissen csatlakoznak a szervezethez először szokatlan ez a szinte korlátlan szabadság: senki nem mondja meg nekik mit és hogyan kell csinálni, de ha segítséget kérnek egy területen akkor biztosak lehetnek abban, hogy sorban fognak állni a tapasztalt kollégák, hogy közösen találjanak megoldást a problémára.

A szervezet tagjainak egyéni motivációjára különös hangsúlyt fektetnek a Spotifynál. Egy exponenciálisan növekvő szervezetnél ritka az ha 91% azoknak a munkavállalóknak az aránya akik elégedettek a munkahelyükkel és a körülményekkel és csak 4% azoké akik nem. Egy ilyen felmérés után bármely felsővezető elégedetten dőlhet hátra, nem így a Spotifynál, ahol ezt nem találták kielégítőnek és arra biztatták a 4%-ba tartozó munkavállalókat, hogy lépjenek kapcsolatba a felsővezetéssel, hogy megoldják az általuk tapasztalt problémákat. A felmérést követő évben egyébként az elégedettek aránya 94%-ra nőtt.

A rohamos növekedést a szervezeti egységek skálázásával is sikeresen követte le a Spotify. Az Squadokat nagyobb egységekbe Tribeokban (törzsekbe) szervezték, amely egy laza mátrix struktúra. Minden munkavállaló egy Squad tagja amely a

termék egy meghatározott részéért felel, de ugyanakkor tagja egy Chapternek (szakasznak) is. A Chapterekben több különböző Squadból a hasonló kompetenciákkal rendelkező tagok egyesülnek. Így létezik például Chapter a front-end fejlesztők, az agile coachok és a tesztelők számára is. A Chaptereken belül létezik hierarchia: minden Chapternek van egy vezetője aki a többi csapattag szakmai mentorálásáért felel. Így a csapattagok úgy tudnak Squadot váltani, hogy a közvetlen szakmai vezetőjük nem változik.

A Squadokokn és Chaptereken kívül a közös szakmai érdeklődésű tagok Guildekben (céhekben) gyűlnek össze, ahol egy adott területen összegyűjtött tudás megosztása a cél, legyen szó vezetői készségekről vagy webfejlesztésről. A Guildek elég laza kapcsolatot tartanak fenn, bárki bármikor csatlakozhat vagy elhagyhatja őket. A kommunikáció formái is változatosak: a levelezési listától egészen a belső vállalati konferenciákig terjednek.

A szervezeti kultúra a Spotfiynál nem a papíron megrajzolt struktúrától lesz olyan amilyen, hanem azért mert az önálló közösségek működését többre értékelik mint a szigorú rendszereket - emlékezzünk mennyire összhangban áll ez az Agilis Kiáltványban megfogalmazottakkal.

Release management

Az önállóság egyik alapja az, hogy mennyire könnyen tudja egy Squad az éles környezetbe telepíteni az új fejlesztéseket. Ha ez a folyamat túlságosan nehéz, akkor az azt fogja eredményezni, hogy folyamatosan ritkul az új verziók megjelenése. Ez azt fogja eredményezni, hogy az egyes verziók egyre több újdonságot tartalmaznak ami egyre nehezebbé teszi a megjelenést és egy ördögi kör alakul ki ami egyre lassabb működést eredményez. Ha ezt megfordítjuk és megkönnyítjük az új megjelenéseket, akkor viszont egyre gyakrabban egyre kisebb egységek jelennek meg az éles környezetben így fel tudjuk gyorsítani a működést. Ennek alapjaként a

Spotify nagy hangsúlyt fektet a tesztek automatizálására és a Continous Integration / Continous Delivery (CI/CD) infrastruktúra fejlesztésére.

A Spotify asztali kliensének indulásakor a teljes alkalmazás egy monolitikus struktúrát alkotott. Ez egészen addig jól működött amíg egy Squad dolgozott ezen a terméken, de ahogy nőttek a felhasználóktól érkező igények és ezzel párhuzamosan a terméken dolgozó Squadok száma, úgy nőtt meg az igény arra is, hogy ne legyen szükség a Squadok munkájának folyamatos összehangolására. Ezt felismerve úgy alakították át a termék architektúráját, hogy annak egyes részeit külön-külön is lehessen frissíteni.

Később felismerte a szervezet azt is, hogy ezt több termék esetében is sikeresen tudja alkalmazni, ezért a Squadokat is specializálták. Három különböző területen működnek Squadok: a funkciók fejlesztésén, a kliens applikációk területén és az infrastruktúra fejlesztés témakörében. A funkciók fejlesztésén dolgozó Squadok a termék egy-egy funkcióján dolgoznak, mint amilyen például a keresés. Az általuk fejlesztett funkciók minden platformon elérhetőek legyen szó iOS vagy Android applikációról vagy egy webes kliensről. A kliens applikációkon dolgozó Squadok célja, hogy egy meghatározott platformon elősegítse és megkönnyítse a funkciókon dolgozó csapatok számára a frissítéseket és az új fejlesztések szállítását. Az infrastruktúra fejlesztésére szakosodott Squadok pedig olyan eszközöket adnak a többieknek amelyekkel megkönnyítik a folyamatos szállítást vagy optimalizálhatják az erőforrások felhasználását.

Ez az ökoszisztéma úgy működik mint egy önkiszolgáló étterem. Egy infrastruktúráért felelős Squad nem fogja élesbe állítani egy funkciókkal foglalkozó Squad kódját, de mindent megtesz azért, hogy rendelkezésre álljanak azok az eszközök amelyek megkönnyítik ezt. Mindenki azt tesz a tálcájára amire éppen szüksége van.

Release Trains, Feature Toggles

Habár a fenti modell célja, hogy minden Squad a lehető legnagyobb önállóság mellett tudja a munkáját végezni, szükség van arra, hogy időközönként ezt összehangolja a szervezet. Erre a Spotify a Release Trainek és a Feature Togglek bevezetésével talált megoldást.

Minden kliens alkalmazás egy rendszeres időközönkénti kiadási sorozattal rendelkezik, amelyet Release Trainnek nevez. Ezek a nevüket arról kapták, hogy egy előre meghatározott - a menetrendben leírt - időben indulnak el. Nincs szükség tehát folyamatos tervezésre a kiadások kapcsán. Minden héten vagy minden második héten ugyanabban az időpontban indul a kiadási vonat. Aki ott van felszáll rá és ki tudja adni az új frissítést vagy funkciót.

Ha egy Squad elkészült egy új funkcióval akkor ezt hozzá tudja adni a következő Release Trainhez és az elérhetővé válik az ügyfelek számára is. Mi történik akkor, ha még nincs olyan állapotban egy funkció, hogy azt megosszuk a felhasználókkal? A Spotify válasza a kérdésre a Feature Toggle. Ez gyakorlatilag egy kapcsoló amivel azt szabályozzák, hogy a Release Trainhez adott funkció elérhető legyen-e az éles környezetben. Elsőre nem hangzik logikusan, hogy félkész dolgokat adjunk hozzá az éles kódbázishoz, de ez jelentősen csökkenhet két dolgot. Egyrészt azt az időt amíg egy integrációhoz kapcsolódó hiba feltárásra kerül, másrészt a párhuzamosan fenntartott elágazások számát a kódbázisban. Hosszú távon mind a hibák feltárásának késése mind az elágazások számának növekedése a technikai adósság növekedéséhez vezet. A Feature Toggle a fentiekentúl még arra is használható, hogy szelektíven alkalmazva egy-egy funkciót csak a felhasználók kis csoportjának tegyünk elérhetővé béta- és A/B tesztek keretében.

Egy ennyire az önállóságot támogató modell elsőre rendkívül félelmetesnek tűnhet és a Spotify sem működik hiba nélkül, ugyanakkor sokkal értékesebbnek tartják a

bizalmat mint a közvetlen ellenőrzést. Miért alkalmazna egy szervezet bárkit akiben nem bízik meg? A skálázott agilis működés a bizalom skálázásával is együtt jár, ezért tudatosan az irodai politikai játszmák ellen dolgozik a szervezet. A szervezet tagjait arra bátorítják, hogy merjenek hibázni. Ha ugyanis a hibákat büntetik egy szervezetben akkor félelem alakul ki a hibázással kapcsolatban ami azonnal megöli a kísérletező kedvet így az innovációs készséget is.

A hibázással kapcsolatban a Spotify filozófiája az, hogy mindenkinél gyorsabban szeretné a hibákat elkövetni. A gyors hibázás, gyors tanulást eredményez a szervezetben. A gyors tanulás pedig a fejlődés gyorsaságát növeli meg. Ennek megfelelően a Spotifyon belül a hibákat követő helyreállítási folyamatot többre értékelik, mint a hibák elkerülését.

Ezt a kulturális tényezőt erősítendő a Spotifyban találhatunk olyan falakat ahol a fejlesztők nyilvánosan kiírják milyen területeken hibáztak, hogy mások is tanulhassanak belőle. Ha valami komolyabb hiba után szeretnék elindítani a tanulási folyamatot úgynevezett “post mortem” összejövetelt tartanak, ahol elemzik a kialakult helyzet okait, a megoldásokat és azokat a lépéseket amelyeket a hiba javításakor elvégeztek. Soha nem az a kérdés, hogy “kinek a hibája volt?” az adott helyzet. Egy egyszerű hibajegy sem akkor zárul le a szervezetben ha elhárításra került a probléma, hanem akkor amikor dokumentálásra és megosztásra kerültek a kiváltó okok és a tanulságok. Az egyes Squadok ezen túl a Scrum keretrendszerből ismert retrospektív megbeszélésekkel is ösztönzik azt, hogy folyamatosan fejlődjön a szervezet a legkisebb egységtől kezdve.

A hibázás és a hibák javításán keresztüli tanulás csak akkor működhet, ha egyik hiba sem képes akkor katasztrófát okozni amely végzetes lehet a szervezet számára. A Spotify a korábban megismert, az egyes funkciókat egymástól elkülönülő módon

kezelő architektúrális felépítése biztosítja azt, hogy ha valamelyik komponens hibásan működik akkor az csak lokálisan az adott komponensen belül jelent problémát, de a termék egészét tekintve nem okoz szolgáltatás kiesést. Mivel egy ilyen területért egy Squad felelős ezért hiba esetén gyorsan tudnak reagálni és megoldani a problémát.

A másik biztosíték a rendszerben, hogy az új funkciókat a szintén korábban megismert Feature Togglek segítségével mindig csak a felhasználók egy kis csoportja számára teszik elérhetővé. Így ha valami nem megfelelően működik akkor nem a teljes ügyfélkör érzi meg ennek negatív hatásait.

Mivel a hibázás kockázatát sikerül alacsonyan tartani ezért a Squadok sokkal bátrabban kísérletezhetnek új dolgokkal, ami növeli a szervezet innovációs képességét.

Termékfejlesztési folyamat

A Spotify termékfejlesztési folyamata a Lean Startup alapelveire építkezik. Röviden összefoglalva ez azt jelenti, hogy gondoljuk ki, építsük meg, szállítsuk le és finomhangoljuk a terméket.

Egy termék fejlesztése során a legnagyobb kockázat mindig az, ha nem a megfelelő dolgot adjuk az ügyfelek kezébe. Egy új ötlet megvalósítása a Spotifyban mindig a narratíva megalkotásával kezdődik. Ez egy rövid, egy mondatos leírás arról mivel szeretnék bővíteni a termék funkcionalitását. A narratíva létrehozása nagyban hasonlít ahhoz amit a termék vízió megalkotásánál ismertünk meg, de ez a termék egésze helyett csak egy kisebb egységre vonatkozik. Gyakran társul a narratíva egy prototípussal is, amivel a Service Design megismerése során találkozhattunk. A prototípusokat már meg lehet mutatni valódi felhasználóknak is, így jobban megérthető, hogy van-e igény az adott új funkcióra. Ha innen érdemes továbblépni akkor először egy MVP készül el amit a Spotifynál Minimum Lovable Productként

ismernek. Az igazi visszajelzést természetesen akkor kapja meg a szervezet amikor éles környezetben is elérhetővé válik egy funkció. Ez a korábban megismert elvek mentén csak a felhasználók szűk körében lesz elérhető. A már valós ügyfelek számára elérhető funkciókat folyamatosan méri a Squad és azt figyelik, hogy elérte-e a korábban kitűzött célt. Amennyiben nem akkor addig finomhangolják és adnak ki új verziókat amíg ez be nem következik. Ha szűk körben elérték a célt akkor lépésenként egyre szélesebb körben válik elérhetővé a funkció, úgy hogy közben az üzemeltetés- és skálázás során felmerülő kérdéseket is kezeli a Squad. Ezzel a módszerrel mire mindenki számára elérhető a funkció a siker már garantált, mivel ha nem így lenne akkor nem is készülne olyan verzió ami senkit sem érdekel.

A fejlesztési folyamat során a hatás vagyis a termék által teremtett érték sokkal fontosabb hangsúlyt kap mint a fejlesztési idő hossza. Egy funkciót akkor tekint késznek a Squad, ha az elérte a meghatározott célokat.

Ha tervezhetőség szempontjából vizsgáljuk meg ezt a folyamatot akkor azt láthatjuk, hogy a Squadok nem meghatározott határidőkkel dolgoznak. Sokkal többre értékelik az innovációt mint a kiszámíthatóságot. Ha egy marketing esemény vagy egy partner integráció miatt mégis konkrét szállítási időpontot kell megadnia a Squadnak akkor használják a már megismert burnup chartot, de a határidő véglegesítését a lehető legkésőbbre halasztják. Az értékteremtés sokkal nagyobbra értékelt a szervezetben mint a tervek szolgai követése, csakúgy mint ahogyan azt az Agilis Kiáltványban láthattuk.

Innováció és kultúra

Az már az eddigiekből is tisztán látható, hogy a Spotify különösen nagy hangsúlyt fektet az innovációra és különösen nagy teret ad neki. Minden munkavállaló a munkaideje 10%-át “hack time”-ként használhatja fel, azaz kísérletezhet és alkothat úgy, hogy annak nem kell feltétlenül szorosan kapcsolódni a stratégiai célokhoz. Ez

akár lehet egy olyan elvetemült ötlet, hogy egy tárcsás telefonon egy adott számot felhívva az lejátszik egy zeneszámot. Nem számít, hogy egy ilyen ötlet mennyire haszontalan a cég szempontjából. Az elv az, hogy kellően sok ötletet megvalósítva néha kimosható egy aranyrög a patakból. Ha ez nem is fordul elő gyakran a kísérletezés során megszerzett tudás mindenképpen gyarapítja a szervezetet.

Évente kétszer ez egy "Hack Week"-ben csúcsosodik ki, ahol egy egész héten keresztül "menő dolgokon" dolgoznak az alkalmazottak. Azokkal akiket ők választanak, azt csinálnak amit szeretnének úgy ahogyan szeretnék. A hetet pedig egy nagyszabású bemutatóval és bulival zárják.

Az innováció a Spotify DNS-ébe van kódolva. Ha két fejlesztési eszköz közül kell választani akkor kipróbálják mindkettőt és a tapasztalatok alapján döntenek. A sarokba kerüljön egy új gomba vagy középre? Kipróbálják mindkettőt A/B teszttel és az eredmények alapján helyezik el a gombot.

Ahelyett, hogy megbeszélések sorát pazarolnák arra, hogy egy lehetséges fejlesztést jóváhagyjanak meghatároznak egy hipotézist és tesztelik azt, majd levonják a következtetéseket és kezdik elölről. A döntések így adatok és visszajelzések alapján születenek a vélemény-, ego- vagy hatalmi alapú döntési mechanika helyett.

Habár a szervezetben rengeteg kísérletezés zajlik nagy hangsúlyt fektetnek a pazarlás elkerülésére. Ha valami működik akkor megtartják, ha nem akkor a kukában landol és nem foglalkoznak vele tovább. Ez vonatkozik a termékre csakúgy mint a termék létrehozásához használt folyamatokra.

A pazarlás egyik lehetséges forrása, ha nagy méretű belső projekteken kell dolgoznia egyszerre több csapatnak. A szervezet ha lehet ezeket több kisebb, önállóan is megvalósítható elemre bontja és így juttatja el az ügyfeleknek. Ha ez végképp elkerülhetetlen akkor mindenképpen láthatóvá teszik a haladást (akár falra ragasztott cetlikkel), napi szinten egyeztetnek egymással a projektben részt vevő

csapatok, rendszeresek a bemutatók ahol az összes érdekelt fél jelen van és meg tudják vizsgálni a termék aktuális állapotát. A gyakori egyeztetés és az együttműködés segít abban, hogy minimálisra csökkenjen a pazarlás lehetősége.

A folyamatosan növekvő szervezetnek komoly kihívást jelent a növekedési fájdalom: mekkora legyen az a bürokrácia mennyiség amely még nem akadályozza az agilis működést, de már nem a teljes kontrollvesztést eredményezi. A Spotify többre értékeli a kötetlenséget a rögzített struktúráknál és csak annyi merev struktúrát enged meg a szervezetben ami a teljes káoszt elkerüli.

Folyamatos fejlődés

A megfigyelés és alkalmazkodás azaz a folyamatos fejlődés minden téren jelen van a Spotify szervezeti egységein belül. Több Squad alkalmaz fejlődési táblákat, ahol feltüntetik a legnagyobb akadályokat a munkájukban és a következő lépéseket amellyel megpróbálják azokat elhárítani.

Egy sajátos koncepció a “Definition of Awesome”. A fantasztikus működést Squadonként máshogy kezelik, de egy kiváló példa az amikor a Toyotától emelt át az egyik Squad egy Fejlődési Kata nevű módszertant. A módszer lényege, hogy a fejlesztendő területen először meghatározzuk az aktuális helyzetet, majd azt az ideális állapotot amikor a probléma nem létezik többé. A következő lépésben azt az állapotot kell meghatározni ami belátható időn belül reálisan megvalósítható és a fantasztikus jövőbeli állapot irányába tett lépés. Végül pedig a következő három olyan konkrét feladatot kell meghatározni amellyel elérhető a reális célkitűzés.

Mivel a folyamatos fejlődés minden területen jelen van a vállalat életében - a vállalati kultúra része - ezért bármilyen a változások által felmerült problémára képesek gyors választ adni. Az egészséges kultúra képes gyógyítani a rossz folyamatok okozta sebeket.

Mivel a vállalati kultúra kiemelt szerepet játszik a szervezet agilis működésében ezért nagy hangsúlyt fektetnek arra, hogy megtartsák és még jobbá tegyék. Legyen szó arról, hogy az új csatlakozók egy "kiképző táborban" önálló Squaddá formálódnak és úgy ismerkednek meg egymással és a vállalattal valamint a termékkel, hogy közben egy valós problémán dolgoznak. Gyakran úgy, hogy a folyamat végén a létrehozott kódjuk része lesz az éles rendszernek.

Mivel rendkívül büszkék erre a kultúrára kifejezetten szeretik megosztani mind a sikereket mind a bukásokat, akár egy konferencián akár egy blog posztban.

Ahogy láthatjuk ha a szervezeti kultúra minden elemében az agilitást tükrözi és ezt a szervezet tagjai nap mint nap megélik és megosztják akkor függetlenül a keretrendszertől sikeresen tud reagálni a vállalat a környezeti változásokra.

Zombi Scrum és rakomány kultuszok

Az előzőekben azt láthattuk, hogy egy szervezet az agilis keretrendszerek egyes elemeit megtartva másokat elhagyva hogyan lehet sikeres. A Spotify példája azért kiváló mert ezeket a módosításokat úgy hajtották végre, hogy az agilis alapelveket sokkal többre értékelték a keretrendszerek néha merev határainál. Sajnos számtalan szervezet megpróbálja úgy testre szabni a keretrendszereket - így a Scrumot is - hogy a cél egyáltalán nem az, hogy még rugalmasabb, még alkalmazkodóképesebb legyen a szervezet. Sokkal inkább úgy, hogy elkerülje azt a kényszert, hogy az agilis működés során felszínre kerülő szervezeti- és működési problémákat meg kelljen oldania. Ilyenkor beszélhetünk rakomány kultusz Scrumról vagy Zombi Scrumról.

A Zombi Scrum először olyannak tűnhet mint a Scrum, csak hiányzik belőle a dobogó szív. Minden Scrum eseményt megtart a Scrum Csapat, csak éppen a Sprint végén nagyon ritkán jön létre inkrementum. A Scrum Csapat jól láthatóan nem

törekszik arra, hogy változtasson ezen a kialakult helyzeten. Sőt, akár az is előfordulhat, hogy már az érdekelt felek sem foglalkoznak azzal, hogy mi történik a Scrum Csapattal. A Zombi Scrum ugyanolyan mint a Scrum, kivéve, hogy nagyon ritkán jön létre működő szoftver általa.

A Zombi Scrum tünetei

Honnan ismerhetjük fel, ha a szervezetben felüti a fejét a Zombi Scrum? Egy idő után jól látható tüneteket tapasztalhatunk amelyek arra utalnak, hogy nem jó irányba tartunk.

Nincs működő szoftver

Azokban a csapatokban ahol elindul a Zombi Scrum fertőzés jó darabig ugyanúgy működnek mint egy Scrum Csapat. Megtartják a Scrum eseményeket és feladatokon dolgoznak, de a Sprintek végén az Inkrementum, a működő szoftver létrejötte egy választható lehetőség. Ha éppen nem sikerül befejezni valamit az adott Sprintben, hát akkor majd kész lesz a következőben. Ha valami működik is a Sprint végén az csak egy nem túl részletes Definition of Donenak felel meg, amit a Fejlesztők ha nem muszáj nem is nagyon bolygatnak. Egy egészséges Scrum Csapat életében a működő szoftver nem valamiféle hasznos melléktermék, hanem a működés fő célja. Ha a Zombi Scrum megjelenik a csapatban akkor nincs igény az inkrementum létrehozására, a visszajelzések összegyűjtésére és a folyamatok fejlesztésére sem.

Nincs kapcsolat a külvilággal

A Zombi Scrum alapján működő csapatok nem, hogy nem igénylik az ügyfelektől vagy az érdekelt felektől érkező visszajelzéseket, hanem egyenesen igyekeznek elrejtőzni valamilyen sötét, észrevehetetlen helyre a külső hatások elől. Nem felel meg az ügyfelek számára a szoftver amit létrehoztak? A Fejlesztők csak azért vannak a csapatban, hogy “kódot írjanak!”. A Zombi Scrum csapat tagjai csak úgy tekintenek

a munkájukra, mintha egy fogaskerékként dolgoznának a gépezetben: Nem tudnak és nem is akarnak változtatni a helyzetükön, nincs semmire valódi ráhatásuk.

Nincs érzelmi válasz sem a sikerre sem a kudarcra

Ha nincs meg a kapcsolat a külvilággal akkor az gyakran vezethet oda, hogy a csapat nem kap visszajelzést így nem tudja vagy nem érdekli, hogy sikerült-e elérni a célokat vagy sem. Az eredményesség iránti érzéketlenség kialakulhat a Zombi Scrum csapatoknál akkor is ha nincs visszajelzés: ha nincsenek célkitűzések akkor nincs is mit elérni. Ha sikertelen a Sprint, akár azért mert nem valósult meg minden tervezett feladat, akár azért mert mert nincs működő szoftver a Zombi Scrum csapat tagjainak a szeme sem fog megrebbenni. A csapat morál eleve alacsony. “Ha most nem sikerült valamit megcsinálni, majd átvisszük a következő Sprintre. Végül is miért ne? Mindig van egy következő Sprint. És egyébként is ez az egész Sprint dolog csak egy művi valami, nem?”

Nincs igény a fejlődésre

Ha egy csapat elkapta a Zombi Scrum fertőzést akkor sem igényünk nincs a változásra és fejlődésre, sem örömüket nem lelik benne. Ami a legrosszabb, még csak senkit nem is érdekel az egész. Ez nem csak a Fejlesztők hibájából eredhet. Ha a Product Owner nincs jelen a Scrum eseményeken vagy csak utasításokat osztogatni jelenik meg, ha a csapat tagjait rendszeresen más csapatokhoz rendelik át mert ott van szükség a speciális tudásukra és még egy Scrum Master sincs aki segítene ezt felismerni és javítani akkor nincs min csodálkozni mikor felüti a fejét a Zombi Scrum. A csapat kezéből kikerül az irányítás, vagy legalább is ezt érzik. A Sprint Retrospectivek unalmasak vagy méginkább panaszkodásba fulladnak. A csapat nem vágyik a fejlődésre.

A Zombi Scrum okai

Ha az előbb megismert tüneteket tapasztaljuk egy agilis szervezetben akkor biztosra vehetjük, hogy terjedni kezdett a Zombi Scrum. Ahhoz, hogy meg tudjuk akadályozni a terjedését vagy akár megszüntessük a Zombi inváziót meg kell ismernünk a kiváltó okokat.

A rakomány kultusz Scrum

1945. szeptember 2. Japán hivatalosan is letette a fegyvert, így az emberiség történetének eddigi legkiterjedtebb fegyveres konfliktusa - a második nagy világégés - hivatalosan is lezárult, immár a Csendes-óceáni hadszíntéren is. A háború lezárása azonban valami új kezdetét is jelentette a szigetvilágban. A véres csatározások során mind a japán mind az amerikai flotta- és légierő az óceánon található apró szigeteket használta előretolt helyőrségként. A szigetvilágot jellemzően szinte ősközösségi körülmények között élő törzsek lakták. Amikor a hadsereg megérkezett egy-egy ilyen szigetre akkor azonnal elkezdték kiépíteni azokat a feltételeket amivel biztosítani tudták a folyamatos utánpótlást a frontokon. Ez a gyakorlatban azt jelentette, hogy a partokon nagy forgalmat fogadni képes kikötők épültek, az erdők helyét pedig igencsak gyorsan leszállópályák vették át. Miután az ellátási hálózat kiépült megérkeztek az első légi- és vízi szállítmányok is a katonák számára. A szigetek lakó hirtelen évszázadokat ugrottak előre az időben és olyan civilizációs vívmányokhoz jutottak hozzá a katonai szállítmányokon keresztül, mint a csokoládé, a cigaretta, az alkohol vagy a sorozatgyártott ruhaneműk. Ezek a javak rendszerint ejtőernyővel hullotak alá az égből vagy hatalmas ládákban érkeztek a nagy ezüst színű hajók fedélzetén. A jelentősen fejletlen helyiek számára valóságos csoda volt ahogy varázsütés szerűen belecsöppent az életükbe a XX. század.

A háború végeztével azonban a sergek amilyen gyorsan jöttek, olyan gyorsan távoztak is. A katonákkal együtt pedig az ellátmányok utánpótlása is rendkívüli

gyorsasággal illant el. Az őslakosok tanácstalanul álltak a jelenség fölött. Mit lehet tenni, hogy az életüket olyan jólétben folytathassák, mint azt a távoli harcok időszakában megszokhatták? Teljesen egyértelmű volt, hogy a rakomány vagy az égből vagy a tengerről érkezett. Az idegenek magas tornyokból vették fel a kapcsolatot az égi- és vízi istenségekkel. Rendszeresen rituálékat mutattak be, hogy elnyerjék a kegyeiket. Hamarosan megszületett a megoldás. Bambuszból fülhallgatók készültek, cölöpökből fa irányító tornyok nőttek ki a földből az elhagyatott kifutópályák mellett. Egyes helyeken még a repülők mását is megalkották. A hátramaradt használhatatlan fegyvereket újra hadrendbe állították és botokkal kiegészítve minden napi alaki gyakorlatokat mutattak be.

Az elképzelés az volt, hogy ha teljes mértékben lemásolják a katonai bázisok külsőségeit és követik a korábban látott rituálékat - mint a menetelés - akkor a halhatatlanok újra kegyesek lesznek és ládák szállnak alá az égből csokoládéval, cigarettával, alkohollal és pamut ruhákkal. A mana természetesen nem jött. A repülőket és a hajókat nagyon is halandó emberek vezették akiknek addigra már semmi dolguk nem akadt arra. A civilizáció szépen lassan másodjára is betette a lábát a szigetekre, bár ennek ellenére a mai napig léteznek a rakomány kultuszok.

Gondolhatnánk, hogy ez a viselkedés csak egy érdekes lábjegyzet az embertan egyre bővülő tárában, de nem is tévedhetnek nagyobbat. A rakomány kultuszok betették a lábukat a vállalati struktúrákba is. Számos olyan szervezet létezik, ahol hiányos információk alapján pusztán azért mert “mindenki más is agilis” vezetik be a Scrumot vagy más agilis keretrendszert. Mi sem egyszerűbb igaz? Aki eddig projektmenedzser volt, mostantól Product Owner, a vezető fejlesztőt átnevezzük Scrum Masterre. Elkezdünk Sprintekben dolgozni, de a fejlesztést és a tesztelést külön csapatok végzik. A határidők rögzítettek, a fejlesztés tartalmi elemei rögzítettek, a minőségi elvárásokat előre dokumentálva adjuk át a fejlesztő csapatnak. Kívülről ez egészen úgy tűnhet mintha, a szervezet Scrum keretrendszert

használna, de közelebbről megnézve ez nem más mint egy rakomány kultusz, ahol a külsőségeket tartalom nélkül kezdi el használni a szervezet.

A másik tipikus rakomány kultusz működési mód, amikor csak a Scrum Csapat tekinti magáénak az agilis értékeket. Becsülettel küzdenek, de amikor egy Sprint Retrospective során felszínre kerül, hogy a marketing részleg napi szinten dobja át a csapatnak az "azonnal megoldandó" feladatokat akkor nem kapnak rá kellő felhatalmazást, hogy ezeket visszautasítsák.

A mérőszámokról szóló alkalom kapcsán szintén számos olyan példát megismerhettünk amikor azok nem megfelelő használata rakomány kultuszt eredményez.

A Spotify példája jól tükrözi azt, hogy lehet saját agilis működést kialakítani úgy is, hogy házi szabályokkal játszunk, de ha ezeknek nem az a célja, hogy még agilisabb legyen a működés akkor az a legtöbb esetben egy rakomány kultusz kialakulásához fog vezetni.

Nincs vágy a gyorsaságra

Ha a szervezet számára nem egyértelmű, hogy mi jelent értéket az ügyfelei számára, akkor az jelentősen megnehezíti a célok kitűzését és ezáltal azt is, hogy a szervezet tagjai azt érezzék, hogy ezt az értéket minél hamarabb át szeretnék adni az ügyfeleknek. Ha egy ilyen szervezetben kezd el működni egy Scrum Csapat, akkor nem fogják tudni meghatározni a saját céljaikat, akár még egy Sprint célt sem. Ha pedig nincs célja a Sprintnek mi baj lenne abból, ha nincsenek készen a feladatok? Ha nincsenek jól körülhatárolt és mindenki számára egyértelmű célok egy szervezetben az potenciális Zombi Scrum járvány gócnak számít.

Értékütközés

Szintén veszélyt jelenthet a szervezet számára, ha az értékei ellentmondásba kerülnek az agilis alapértékekkel. Ha egy-egy vezető elképzelése a termék fejlesztéséről teljesen más, mint amit a Scrum keretrendszer értéknek tekint akkor könnyen kialakulthat a Zombi Scrum működés.

A Zombi Scrum ugyanis úgy tekint a keretrendszerre mint egy módszertanra. Fontosabb a Scrum Útmutató betűje mint a szelleme. Hiába ad lehetőséget például a Scrum a gyors bukásra - és ezáltal a tapasztalati tanulásra - ha a hibázást bünteti a szervezet.m Hiába alapvető az értékteremtés a Scrumban az Inkrementum által, ha a szervezet a megírt kód mennyiségét értékeli. És hiába az idő és a minőség a rögzített tényező a szállítandó funkciók pedig változhatnak a Scrumban, ha a szervezetben mindig mindent megadott határidőre kell szállítani az előzetes elvárások alapján.

Mazsolázgatás

Bizonyos dolgok amelyek személyesen nem tetszenek a szervezet tagjainak tetszőlegesen elhagyásra kerülnek a Scrumból. Mi baj lehet belőle, igaz? Nézzük meg néhány példán keresztül, hogy a mazsolázgatás hogyan vezethet Zombi Scrum járványhoz.

A Scrum Master drága mulatság. Amíg az egyik Fejlesztő nem teszteket ír addig ő is el tudja küldeni a menedzsment számára a státusz jelentéseket heti négy órában.

> *A Scrum Master feladata nem az, hogy riportokat készítsen, hanem hogy segítse a folyamatos fejlődést.*

A Sprint véget ért pénteken, de még nincs minden kész. Egy kis túlóra simán belefér a csapatnak a hétvégén. Még kis is fizetjük és rendelünk pizzát is nekik.

A Sprintnek akkor van vége amikor letelik az előre meghatározott ideje. Ha sikertelen volt az ugyan kudarc, de ennek az okait kell megvizsgálni és a következő Sprint során kijavítani a hibákat.

A Sprint Reviewt a héten inkább nem tartjuk meg, mert semmi nincs amit bemutathatunk. Ami ennél is szörnyűbb: bemutatjuk egy Powerpoint prezentációban, hogy min dolgoztunk.

A Sprint Review kiváló alkalom, hogy visszajelzést kapjon a csapat a termékről és ezt beépítse a következő iterációba. Ha nincs mit bemutatni akkor nincs sok értelme a működésnek sem. Hogy ennek mi az oka az sokkal izgalmasabb és kiváló téma egy Sprint Retrospectiven.

Annyi munka van még hátra, inkább maradjon el a Sprint Retrospective.

Ha a csapat nem vizsgálja meg időről-időre a saját működését, akkor képtelen lesz a fejlődésre. Ha elmarad a Sprint Retrospective - akár rendszeresen - akkor az esélyt sem adják meg a fejlődésnek.

A Backlog Refinement során elég ha a Product Owner és a “vezető fejlesztő” vesz részt.

A Fejleszők között nincs hierarchia. És ahogyan azt a Scrum esemenyékeről szóló alkalom során láthattuk igenis van hozzáadott értéke minden Fejlesztőnek egy Backlog Refinement során.

A Zombi Scrum gyógyítható!

Ha látjuk a tüneteket és ismerjük az okokat, akkor jó hír, hogy a Zombi Scrum gyógyítható és Product Ownerként te is aktívan tehetsz érte, ha ezt tapasztalod a szervezetben.

Legyél zombi suttogó

Ha egy csapat demotivált, érdektelen és érzéketlen emberrel kell együtt dolgozni, akkor nem biztos, hogy túl sok elvárás lehet feléjük, de ha egyszerűen csak beszélsz velük akkor akár az is csodákat tehet. Ha sikerül megérteni az egyéni motivációkat és sikerkritériumokat vagy csak átadni, hogy hogyan képes az agilitás átalakítani a működést úgy hogy az értéket teremtsen, akkor sikerülhet kimozdítani az egyéneket a fásult állapotukból. A Zombi Scrum nem egy csapaton belüli probléma, akkor jelenik meg a színen, ha a Scrum és az agilitás alapértékei ellentétesek azokkal az értékekkel amelyet a szervezet képvisel. Ezt csakis a szervezeti kultúra átalakításával lehet áthidalni ami csak akkor működik, ha a szervezet minden tagja képes és hajlandó is a változásra.

Scrum vérátömlesztés

A legtöbb szervezet, ha észleli is, hogy probléma van akkor nem igazán tudják, hogyan is kellene megoldani. Ilyenkor jöhet jól valaki aki ismeri az agilis működést vagy legalább is közel érzi magához azokat az értékeket amelyeket szükségesek az agilis működéshez. A jó példa ragadós, főleg akkor ha eredményekkel is párosul. Ha nem kívülről érkezik valaki a szervezetbe akkor érdemes lehet más vállalatok működését megvizsgálni, például egy Safari keretein belül (amilyen a Budapest Startup Safari). Nem szégyen ha külső segítséget vesz igénybe egy szervezet az induláshoz vagy a problémák megoldásához, de ilyen esetben fontos, hogy ne egyszeri alkalomról legyen szó.

Rázzuk fel a dolgokat

Elképzelhető, hogy néhány kisebb módosítással is nagy hatást érhetünk el, ha megfigyeljük hogyan használja a Scrum keretrendszert a szervezet.

A Scrum Útmutató a Sprint hosszát legfeljebb egy hónapban határozza meg. Sokszor ennek csökkentése három- vagy két hétre meghozhatja a kellő eredményt.

A Sprint Planning során adjuk a Fejlesztők kezébe a gyeplőt és kérdezzük meg őket, hogy milyen célt szeretnének elérni a következő Sprint során.

A Daily Scrum során vizsgáljuk meg a kitűzött célt és nézzük meg a csapattal közösen, hogy mi történt ami ebbe az irányba haladt.

Hívjunk meg valódi ügyfeleket a Sprint Reviewra. A szervezeten belüli érdekelt felek is jobban el tudnak köteleződni, ha egy hús-vér felhasználó mondja el a tapasztalatait a termékről.

A Sprint Retrospective során rugaszkodjunk el a csapattal a valóságtól. Egy nagyobb, hosszú távú cél elérése motiválólag hat mint ha apró módosításokat végeznénk a folyamatokon.

Bármit is teszünk lesz olyan helyzet ami menthetetlen. Akár azért mert a szervezet bizonyos szereplői nem hajlandóak a változásra, akár azért mert túl mélyen vert gyökeret a Zombi Scrum. Ha minden kötél szakad akkor lássuk ezt be és lépjünk tovább.

Meríts egy nagyot a Scrum közösségből

Sokan és egyre többen használják jól a Scrumot és az agilitást a saját szervezetükben. Közülük rengetegen büszkék is erre és örömmel osztják meg a tapasztalataikat a közösség többi tagjával. Csatlakozz te is szakmai csoportokhoz, menj el meetupokra, írd meg egy blog posztban ha valami jól működik a saját szervezetedben. Ha te is tartasz betétet a szívesség bankban akkor könnyen tudsz hitelt felvenni onnan.

A Zombi Scrum és a rakomány kultuszok sok szervezetben jelen vannak, de ahogy kiderült nem lehetetlen felszámolni őket. Alkalmazd a megismert módszereket és legyél te is Zombi Scrum Irtó.