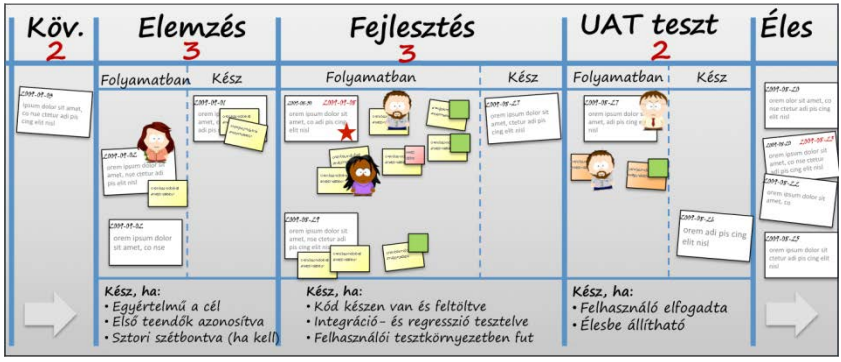


Kanban és Scrum

mindkettőből a legjobbat

Henrik Kniberg és Mattias Skarin

Fordította: Csutorás Zoltán és Marhefka István

Előszó: Mary Poppendieck és David Anderson

C4Media, Publisher of InfoQ.com.

Ez a könyv az InfoQ Enterprise Software Development könyvsorozat része.

A könyv angol változatának megrendeléséhez lépjen kapcsolatba a kiadóval a books@c4media.com címen.

Felelős szerkesztő: Diana Plesa
Borító terv: Bistrian Iosip
Composition: Accurance

ISBN: 978-0-557-13832-6

Fordította:

Csutorás Zoltán (www.adaptiveconsulting.hu)
Marhefka István (http://infokukac.com)

A fordítás nyelvi lektorálását a Factory Creative Studio támogatta.

Tartalom

Előszó a magyar fordításhoz

Évekkel ezelőtt egy kritikus állapotba került projektbe csöppentünk válságkezelőként és új vezetőfejlesztőként. Gyorsan világossá vált számunkra, hogy az eddig alkalmazott vízesés módszerekkel nem jutunk messzire. A projekt közel egy évvel korábban indult, a követelményspecifikációt pedig még mindig nem sikerült lezárni. A megrendelő türelme elfogyott, látható eredményt várt a csapattól, méghozzá nagyon gyorsan. Ekkor kezdtünk el új projektvezetési módszerek után kutatni.

Az agilis és XP irányzatokról hallottunk már, néhány elemüket alkalmaztuk is, de emellett szükségünk volt egy hatékony és dokumentált menedzsmentmódszerre. Az interneten sok információt lehetett találni a Scrum „ideológiáról", de nekünk konkrét, gyakorlati tanácsokra volt szükségünk, viszont nem ismertünk hazai, jelentős tapasztalattal rendelkező embert. Ekkor jelent meg Henrik Kniberg *Scrum and XP from the trenches* című könyve az InfoQ-n. Pont ilyet kerestünk!

A projekt végül sikeres lett, a terméket éles használatba vették, mi is külön utakra tévedtünk. A Scrumról időközben rengeteget tanultunk, már magabiztosan alkalmaztuk és bevezettük új helyeken is. Jóval később mindketten – bár más projekten, más cégnél – újra közös problémába botlottunk. A már átadott termékek éles üzemi támogatása, és ezzel egy időben továbbfejlesztése új kihívások elé állított minket. A háromhetes, megszakíthatatlan sprintek valahogy nem működtek ebben a helyzetben.

Ekkor már ismertük a Lean-/Kanban-módszert, ami jó választásnak tűnt az adott helyzetben. Nagyobb rugalmasságot, erősebb folyamatszemléletet ígért. Mindketten alkalmazni kezdtük. A sors úgy hozta, hogy ebben a témában is megjelent egy könyv Henrik Kniberg és Mattias Skarin tollából. Mielőtt letöltöttük volna, tudtuk, hogy mire számíthatunk. Nem csalódtunk.

Úgy éreztük, itt az ideje, hogy használható, gyakorlatias könyv formájában magyar nyelven is ismertté váljon ez a téma. Az angol szöveg nagyon személyes hangvételén kicsit ugyan változtattunk, de meggyőződésünk szerint a fordítás híven tükrözi az eredeti tartalmat.

A könyvet azoknak ajánljuk, akik az agilis projektvezetési módszertanok iránt érdeklődnek. Célközönségünk mindenki, aki szoftverfejlesztéssel foglalkozik: legyen az megrendelő vagy szoftverfejlesztő cég, ezen belül is üzleti elemző, projektmenedzser, fejlesztő vagy akár tesztelő.

Az Olvasó ezzel a kiadvánnyal egy csapásra két legyet is üthet: megismerheti mind a Scrum, mind a Kanban lehetőségeit.

A manapság oly divatos Scrum új megvilágításba kerül. A könyv elrugaszkodik a tananyagként oktatott dogmáktól, megmutatja, mik a Scrum korlátai, és azokon hogyan lehet változtatni úgy, hogy közben ne veszítsük el az irányítást.

Számos ötletet, tippet kaphatunk arról, hogyan szervezhető meg több csapat vagy akár egy teljes szervezet munkája, hogyan lehet kezelni egyszerre több fejlesztés alatt álló terméket, amelyeken egy közös csapat dolgozik.

A könyv fordítása abban a reményben készült, hogy a benne szereplő ötleteket az Olvasó a valóságban is kipróbálja, és eredményesen alkalmazza.

Sok sikert kívánunk!

Csutorás Zoltán
http://www.adaptiveconsulting.hu

Marhefka István
http://infokukac.com

Mary Poppendieck előszava

Henrik Kniberg egyike azon a keveseknek, akik képesek egy bonyolult helyzet lényegét megragadni, kiemelni a fontos gondolatokat a lényegtelenek közül, és kristálytisztán, könnyen érthető formában közölni azokat. Ebben a könyvben Henrik briliánsan magyarázza el a Scrum- és a Kanban-rendszer közötti különbségeket. Világossá teszi, hogy ezek csak eszközök, és arra van igazán szükségünk, hogy megértsük, hogyan használjuk őket gyengeségeik ellenére és erősségeik kiaknázásával.

Ebből a könyvből megérthető, miről is szól a Kanban, mik az erősségei és korlátai, illetve mikor érdemes alkalmazni. Szintén jó leckét mutat arról, hogyan és mikor fejleszthetjük vele a Scrumot vagy bármely más módszert, amit éppen alkalmazunk. Henrik világossá teszi, hogy nem az a lényeg, milyen eszközök alkalmazásával kezdünk, hanem az a mód, ahogyan időről-időre folyamatosan fejlesztjük és bővítjük a rendelkezésünkre álló lehetőségeket.

A Mattias Skarin által írt második fejezet még hasznosabbá teszi a könyvet azáltal, hogy – egy valós életből vett példán keresztül – végigvezet minket a Scrum és a Kanban alkalmazásán. Ebben a részben követhetjük végig, hogy ezen eszközök hogyan segítették együtt és külön-külön is egy szoftverfejlesztési folyamat tökéletesítését. Észrevehetjük, hogy a sikernek nincs egyetlen igaz útja, hanem – a saját, pillanatnyi helyzetünk függvényében – magunknak kell kitalálnunk a következő lépést a jobb szoftverfejlesztési folyamat felé.

Mary Poppendieck

David Anderson előszava

A Kanban nagyon egyszerű gondolaton alapszik. A végrehajtás alatt álló feladatok (angolul work-in-progress, rövidítése WIP) számát korlátozni kell, és új teendőt csak abban az esetben szabad elkezdeni, ha egy „munkadarab” elkészült, vagy azt egy következő feldolgozási folyamat átvette. A Kanban (vagy vezérlő-/jelzőkártya) olyan vizuális jel, ami arra utal, hogy megkezdődhet egy új feladat végrehajtása, mivel a folyamatban lévők száma nem éri el a megengedett maximális értéket. Mindez nem hangzik túl forradalmi gondolatnak, sem olyasvalaminek, ami alapvetően megváltoztatná egy csapat és az őt körülvevő szervezet teljesítményét, kultúráját, érettségét vagy alapvető képességeit. Pedig pontosan ezt teszi, és ez az igazán bámulatos! A Kanban bevezetése apró dolognak tűnik, mégis mindent megváltoztat a szervezet működésében.

Nekünk változáskezelési megközelítése tűnt fel igazán. Maga a Kanban nem szoftverfejlesztési vagy projektmenedzsment életciklusmodell vagy -folyamat. A Kanban-szemlélet, mely a meglévő projektmenedzsment vagy szoftverfejlesztési módszereink részévé teszi a változást. Alapelve, hogy abból indulunk ki, ahogyan aktuálisan működünk. Megértjük a jelenlegi folyamatainkat azáltal, hogy feltérképezzük az értékteremtés teljes menetét, és annak minden lépésére meghatározunk egy végrehajtás alatt álló feladatszám- (WIP-) korlátot. Ezek után az elvégzendő munkát végigáramoltatjuk ezen a rendszeren úgy, hogy akkor kezdünk bele egy folyamatlépés végrehajtásába, ha megjelent egy Kanban-jelzés.

A Kanban-módszer hasznosnak bizonyult az agilis szoftverfejlesztést követő csapatoknál, de egyre gyakrabban társul tradicionálisabb megközelítést alkalmazó csoportok munkájához is. A Kanban a Lean-kezdeményezésben bukkant fel először, hogy segítse a vállalati kultúra átalakítását és a folyamatos fejlődést.

Mivel a végrehajtás alatt álló feladatok száma korlátozott a Kanban-rendszerben, bármi, ami megakad a folyamatban, a teljes rendszer bedugulását eredményezi: ha sok feladat blokkolódik, az a teljes rendszert leállásra kényszerítheti. Ez arra készteti a szervezetet, hogy a megakadt munkafázisra figyeljen, és elhárítsa az akadályt, hogy újra megindulhasson a folyamat.

A Kanban vizuális visszajelzési mechanizmusokat alkalmaz a feladatok nyomon követésére, ahogy azok végigáramlanak az értékteremtő folyamat különböző állomásain. Ehhez jellemzően fehér táblát és post-it cetliket vagy elektronikus rendszereket használnak. A legjobb megoldás valószínűleg mindkét módszer együttes alkalmazása. A Kanban-rendszerből fakadó magas szintű átláthatóság nagyban hozzájárul a szervezet kulturális változásához. Az agilis módszerek jónak bizonyultak a folyamatban lévő és elvégzett feladatok számának, valamint az olyan mérőszámok mint a sebesség (azaz az egy iteráció alatt elvégzett feladatok mennyiségének) átláthatóvá tételében. A Kanban ennél egy lépéssel továbbmegy – láthatóvá téve a folyamatokat és az értékáramot. Felszínre hozza a szűk keresztmetszeteket, a sorban álló feladatokat, az inkonzisztenciákat és a veszteséget is; az elvégzett értékes munka mennyiségében és az ehhez szükséges ciklusidőben kifejezve minden olyan tényezőt láthatóvá tesz, ami befolyásolja a szervezet teljesítményét. A Kanban lehetővé teszi a csapat résztvevői és a külső érintettek számára, hogy gyors visszajelzést kaphassanak minden olyan cselekedetük hatásáról, amit végrehajtottak vagy éppen nem hajtottak végre. Mindezeknek köszönhetően az esettanulmányok azt mutatják, hogy a Kanban az emberek viselkedését a nagyobb együttműködés irányába tereli. Az inkonzisztenciák, a szűk keresztmetszetek, a veszteségek és ezek hatásainak felszínre hozatala ösztönzi a problémák okainak feltárását, a továbbfejlesztési lehetőségekről folyó eszmecseréket, ezért a csapatok gyorsan hozzálátnak a folyamataik javításához.

Ennek eredményeként a Kanban ösztönzi a meglévő folyamatok állandó fejlesztését, mely alapvetően összhangban van az agilis és Lean értékekkel. A Kanban nem várja el, hogy mindenestül felforgassuk azt, ahogyan az emberek dolgoznak, hanem a fokozatos és folyamatos változás feltételeit teremti meg, a szervezet minden szintjének egyetértésével és támogatásával.

A húzórendszer sajátosságainak köszönhetően a Kanban elősegíti a visszafordíthatatlan kötelezettségvállalások és döntések lehető legkésőbbi időpontra történő halasztását. A csapatok és a vezetők általában kialakítják a rendszeres priorizációs találkozók ütemezését, ritmusát, melyeken meghatározzák a következő elvégzendő feladatokat. Ezek a találkozók meglehetősen gyakoriak is lehetnek, mivel az időtartamuk rövid. Ezeken a megbeszéléseken – leegyszerűsítve – az alábbihoz hasonló kérdéseket tárgyalnak meg: „Az utolsó egyeztetés óta két teendő számára szabadult fel hely. A jelenlegi ciklusidőnk a feladatok leszállítására hat hét. Melyik az a két feladat, aminek leszállítására a leginkább szükség van mához hat hétre?” Ennek a módszernek kettős hatása van. Az egyszerű kérdések feltétele általában gyors és lényegre törő választ eredményez, ami segít a megbeszélést mederben tartani. A mechanizmus az utolsó pillanatig támogatja a döntési lehetőségek nyitva tartását, ami az elvárások kordában tartásával növeli a hatékonyságot, valamint csökkenti a feladat vállalása és megoldása között eltelt ciklusidőt, így annak valószínűségét is, hogy a prioritások a feladat végrehajtása közben megváltoznak.

A Kanban melletti végső érv, hogy a végrehajtás alatt lévő teendők korlátozása lehetővé teszi a ciklusidő előrejelzését, és megbízhatóbbá teszi a feladatok leszállítását. Az akadályokhoz és hibákhoz történő „állítsd meg a folyamatot!” (stop the line) hozzáállás nagyon magas minőségi színvonalhoz vezet, és csökkenti a módosítások, javítások számát.

Bár mindezek magától értetődővé válnak a könyv nagyszerűen megírt, tiszta leírásaiból, az azonban nem kerül ismertetésre, hogyan is jutottunk el idáig. A Kanban nem egy délután alatt született meg valami csodálatos ihlet eredményeként, hanem évek alatt fejlődött azzá, ami. Azon mély pszichológiai és szociológiai hatások közül, melyek megváltoztatják a szervezetek képességeit és érettségét, sokat soha nem terveztek meg, inkább felfedeztek. A Kanban eredményei közül több nehezen megmagyarázható. Az, ami első látásra technikai megközelítésnek tűnik – korlátozzuk a végrehajtás alatt lévő feladatok számát, és alkalmazzunk húzórendszert – végső soron alapvető hatással van arra, ahogy az emberek együttműködnek egymással. Sem én, sem más azok közül, akik a Kanban korai időszakában részt vettek, nem látták előre mindezt.

Azt kerestem, amivé a Kanban lett: úgy közelíti meg a változásokat, hogy csak minimális ellenállást eredményez; ez már 2003-ban világossá vált előttem. A technikai megoldásai miatt is érdeklődtem iránta. Ahogyan a gyakorlatban megismertem a Lean-eszközöket észrevettem, hogyha van értelme foglalkozni a végrehajtás alatt álló feladatok számával, akkor pláne van értelme korlátozni azokat, hiszen ez csökkentette a menedzsmentterheket. 2004-ben úgy döntöttem, hogy a legfontosabb Lean-alapelvek közül megkezdjük a „húzórendszer" bevezetését. Erre akkor nyílt lehetőségem, amikor a Microsoft egyik vezetője felkért, hogy segítsek egy belső IT-rendszerek karbantartását végző csoport működésének megújításában. Az első megvalósítást a Theory of Constraints elméletben[1] Dob–Puffer–Kötél néven ismert húzórendszer alkalmazásával végeztük. Az eredmény igazi sikertörténet lett: a ciklusidő 92 százalékkal csökkent, a teljesítmény több, mint háromszorosára nőtt, az előrejelzési pontosság pedig 98 százalék lett.

2005-ben Donald Reinertsen beszélt rá, hogy valósítsunk meg egy teljes Kanban-rendszert. Erre 2006-ban nyílt lehetőségem, amikor a Corbisnál megbízást kaptam a szoftverfejlesztési részleg vezetésére, Seattle-ben. Az első eredményekről 2007-ben kezdtem beszámolni, előadást először az 2007 májusában tartottam a Lean Termékfejlesztési Konferencián, Chicagóban. Ezt követte egy OpenSpace beszélgetés még ebben az évben Washingtonban, az Agile 2007 konferencián, huszonöt résztvevővel, közülük hárman a Yahoo!-tól voltak (Aaron Sanders, Karl Scotland és Joe Arnold). Ezt követően hazamentek Kaliforniába, Indiába, illetve az Egyesült Királyságba, majd bevezették a Kanban-rendszert saját, Scrummal bajlódó csapataiknál. Szintén ők indítottak egy Yahoo!-vitacsoportot is, melyben – e sorok írásakor – közel nyolcszáz tagot számlálnak. A Kanban-rendszer megkezdte térhódítását, a korai alkalmazók elkezdtek beszámolni a tapasztalataikról.

Most, 2009-ben, a Kanban-rendszer alkalmazása igazán terjedőben van, és egyre több és több tapasztalati beszámoló kerül napvilágra. Az elmúlt öt évben sokat tanultunk a Kanbanról, és ez minden nap valami újjal egészül ki. A saját munkámat annak szentelem, hogy alkalmazzam a

[1] A *Theory of Constraints* (*A kényszerek elmélete*) dr. Eliyahu M. Goldratt által megfogalmazott megközelítés, mely a húzórendszerek egyfajta leírását adja (a ford.).

Kanbant, írjak, beszéljek és gondolkodjak róla, hogy jobban megértsem, és másoknak is segítsek megérteni. Szándékosan kerülöm, hogy a Kanbant más agilis módszerekhez hasonlítgassam, bár 2008-ban tettem erőfeszítéseket, hogy megértessem, miért van méltó helye az agilis módszertanok között.

Az olyan kérdések megválaszolását, hogy „Milyen a Kanban a Scrumhoz képest?” azokra hagyom, akik nálam több tapasztalattal rendelkeznek. Éppen ezért nagyon örülök, hogy Henrik Kniberg és Mattias Skarin e kérdés élére állt. Önnek a tudás világában történő munkához információkra van szüksége, hogy megalapozott döntéseket hozhasson és magasabb szintre léphessen. Henrik és Mattias úgy elégíti ki ezeket az igényeket, ahogyan én sosem tudnám. Engem különösen megfogott az a tényszerű, bölcs és elfogulatlan mód, ahogy Henrik összehasonlítja a két módszert. Rajzai és illusztrációi különösen találóak, és gyakran több oldalnyi szöveg elolvasását spórolják meg. Mattias esettanulmánya azért fontos, mert jól példázza, hogy a Kanban lényegesen több mint egy elmélet, és példákon keresztül mutatja be, hogyan lehet hasznos az Ön projektjeiben is.

Bízom benne, hogy élvezni fogja ezt a könyvet a Kanban és a Scrum összehasonlításáról, mely mélyebb bepillantást enged általában az agilitás világába, és különösképpen a Kanban és Scrum módszereibe. Ha többet szeretne megtudni a Kanbanról, látogasson el a Limited WIP Society közösségi oldalára a http://www.limitedwipsociety.org/ címen.

David J. Anderson

Sequim, Washington, USA
2009. július 8.

Bevezetés

Nem szoktunk könyvet írni. Jobban szeretjük az időnket éles környezetben tölteni, és az ügyfeleinket segíteni abban, hogy jobban megértsék, optimalizálják és átszervezzék folyamataikat és szervezetüket. Újabban észrevettünk néhány irányvonalat, amelyekkel kapcsolatban szeretnénk gondolatainkat megosztani az Olvasóval. Kezdjük egy tipikus szituációval!

- **Jim:** „Végre túl vagyunk a Scrum bevezetésen!”
- **Fred:** „Na, és milyen?”
- **Jim:** „Tulajdonképpen sokkal jobb, mint ahogy korábban működtünk...”
- **Fred:** „...de?”
- **Jim:** „... csak, tudod, mi egy támogató és karbantartó csoport vagyunk.”
- **Fred:** „Igen, és?”
- **Jim:** „Nos, hát imádjuk ezt a dolgot a product backloggal, a prioritások rendezgetésével, az önszerveződő csoportokkal, a napi Scrum egyeztetésekkel, kiértékelésekkel stb...”
- **Fred:** „Akkor, mi a gond?”
- **Jim:** „Hát az, hogy rendre elbukjuk a sprintjeinket.”
- **Fred:** „De miért?”
- **Jim:** „Mert nehéz kéthetes terveket bevállalnunk. Az iterációknak nincs túl sok értelme a mi esetünkben, általában csak az aznapra éppen legfontosabb feladatokon dolgozunk. Szervezzünk esetleg egyhetes iterációkat?”

- **Fred:** „Be tudnátok vállalni egy hétre való feladatot? Meg tudnátok oldani, hogy csak az egy hétre előre betervezett feladatokon dolgozzatok?”
- **Jim:** „Nem igazán, naponta bukkannak fel újak. Esetleg, ha egynapos sprintekben dolgoznánk...”
- **Fred:** „A feladataitok megoldhatók kevesebb, mint egy nap alatt?”
- **Jim:** „Nem, néha néhány napig is eltartanak.”
- **Fred:** „Tehát az egynapos sprintek sem működnének nálatok. Nem gondoltatok arra, hogy teljesen megszabaduljatok a sprintektől?”
- **Jim:** „Nos, tulajdonképpen nem lenne rossz. De ez nem Scrum-ellenes?”
- **Fred:** „A Scrum csak egy eszköz. Te döntöd el, hol és hogyan használod. Ne légy a szolgája!”
- **Jim:** „Tehát, mit kellene tennünk?”
- **Fred:** „Hallottatok már a Kanbanról?”
- **Jim:** „Az meg micsoda? Miben különbözik az a Scrumtól?”
- **Fred:** „Tessék, olvasd el ezt a könyvet!”
- **Jim:** „De én egyébként szeretem a Scrumot, most akkor váltsak?”
- **Fred:** „Nem, kombinálhatod is a kettőt!”
- **Jim:** „Micsoda? Hogy?”
- **Fred:** „Csak olvasd el...”

A könyv célja

Ha Ön érdeklődik az agilis szoftverfejlesztés iránt, bizonyára ismeri már a Scrumot, és hallhatott már a Kanbanról is. Egyre gyakrabban találkozunk a kérdéssel: Mi ez a Kanban? Milyen a Scrummal összehasonlítva? Hogyan egészítik ki egymást? Van közöttük potenciális ellentét?

E könyv célja a köd eloszlatása, hogy az Olvasó is megítélhesse, hogyan hasznosíthatná a Kanbant és a Scrumot saját környezetében.

Ha sikerrel jártunk, értesítsen róla!

Első rész – összehasonlítás

A könyv ezen első része tárgyilagos és gyakorlati összehasonlítást kíván adni a Scrumról és Kanbanról. Ez a 2009 áprilisában Kanban vs. Scrum címmel írt cikk kissé módosított változata. Az írás népszerűvé vált, ezért úgy döntöttem, hogy könyv formájában is kiadom, és megkérem Mattias nevű kollégámat, hogy tuningolja fel egy kis esettanulmánnyal. Szerintem jól sikerült. Ez az első rész akár egy az egyben kihagyható, és elkezdhető a második, esettanulmányt tartalmazó fejezettel.

Henrik Kniberg

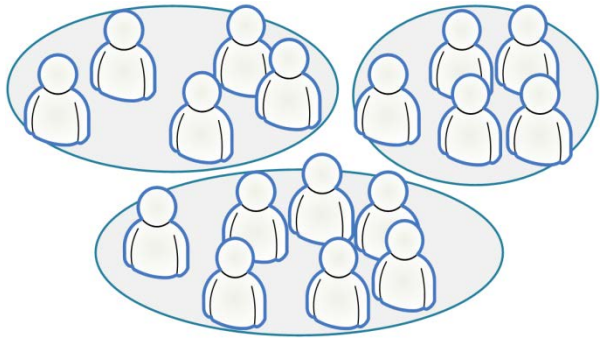
1

Miről is szól a Scrum és a Kanban?

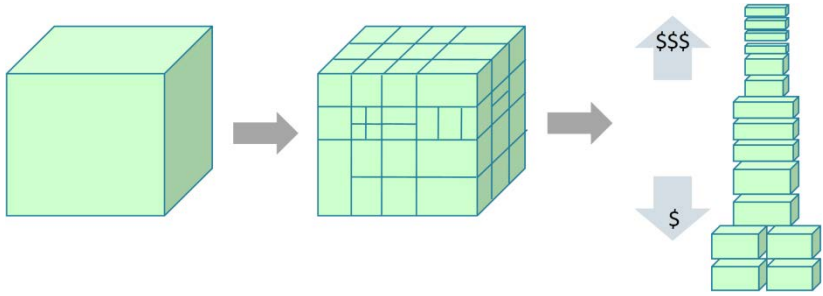
Ok, próbáljuk meg összefoglalni mindkét módszert kevesebb, mint száz szóban.

A Scrumról dióhéjban

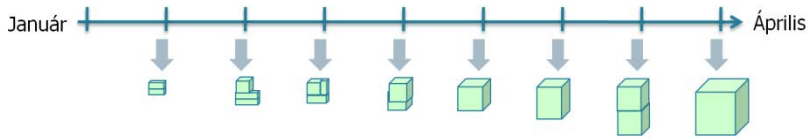
- **Osszuk fel a szervezetet** kisméretű, keresztfunkcionális (cross-functional) önszerveződő csapatokra!



- **A munkánkat szintén daraboljuk** fel kis, konkrét, szállítandó elemekre! Ezt a listát rendezzük prioritás szerint, és becsüljük meg mindegyik elem relatív ráfordítását!



- **Osszuk a rendelkezésre álló időt** rövid, fix-hosszúságú (általában egy-négyhetes) iterációkra (azaz „sprintekre”, ahogy a Scrum hívja)! Minden egyes iteráció végén egy potenciálisan szállítható kódot demonstrálunk.



- **Finomítsuk a releasetervünket** a vevővel együttműködve és frissítsük az abban szereplő prioritásokat az iterációk során szerzett tapasztalatokra alapozva.

- **Állandóan javítsuk a folyamatainkat** úgy, hogy minden iteráció után kiértékelést tartunk!

Ezzel a módszerrel ahelyett, hogy **nagy csoport**tal **sok idő**t töltenénk **nagyméretű munka** elvégzésével, **kisméretű csapat**tal **rövid idő** alatt **kisméretű munkát** teljesítünk. Ezeket azonban rendszeresen integráljuk, hogy az egész kép látható maradjon.

A Scrum részletes ismertetése megtalálható a *Scrum and XP from the Trenches* című kiadványban, amely ingyen elérhető:

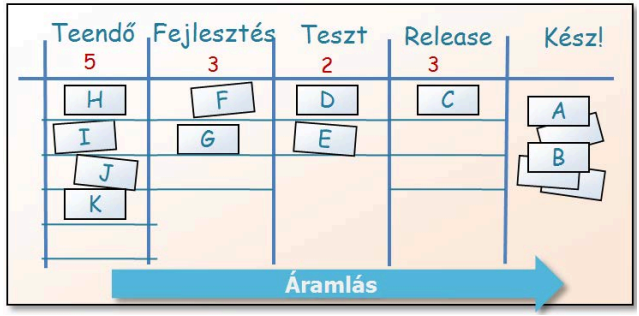
http://www.crisp.se/ScrumAndXpFromTheTrenches.html

További scrumos linkek találhatók a http://www.crisp.se/scrum oldalon.

A Kanban dióhéjban

- **Tegyük láthatóvá a munkafolyamatot!**
    - Bontsuk fel a munkát kisebb részekre, mindegyiket írjuk fel egy kártyára, és helyezzük a falra!
    - Vezessünk be találóan elnevezett oszlopokat, ezzel illusztrálhatjuk, hogy az egyes kártyák hol tartanak a folyamatban!
- **Korlátozzuk a WIP-et (**WIP = work in progress, folyamatban lévő munkák**) –** rendeljünk egyértelmű korlátokat mindenegyes folyamatlépésekhez, amik azt jelzik, hogy az egyes oszlopok hány kártyát tartalmazhatnak!

- **Mérjük az átfutási időt** (egy kártya átlagos teljesítési idejét, ezt gyakran nevezik ciklusidőnek is), és optimalizáljuk a folyamatot, hogy ez az idő a lehető legkisebb és legtervezhetőbb legyen!



Kanbannal kapcsolatos hasznos cikkek linkgyűjteménye itt található: http://www.crisp.se/kanban

2

Hogyan viszonyul egymáshoz a Scrum és a Kanban?

A Scrum és a Kanban egyaránt módszertani eszközök

Eszköz = bármi, amit arra használunk, hogy végrehajtsunk vele egy feladatot, vagy elérjünk egy célt.

Módszer = ahogyan dolgozunk.

A Scrum és a Kanban módszertani eszközök, azaz a hatékonyabb munkavégzésben támogatnak bennünket azáltal, hogy – bizonyos mértékig – meghatározzák mit, hogyan tegyünk. A Java is eszköz, mely a számítógép programozásának egyszerűbb módját biztosítja; a fogkefe is eszköz, ami a fogaink tisztításában segít bennünket.

Azért hasonlítsunk össze eszközöket, hogy megértsük őket, ne azért, hogy ítélkezzünk felettük!

Kés, vagy villa – melyikük jobb?

Meglehetősen értelmetlen kérdés, nem? A válasz a körülményeken múlik. Kicsi fasírtok elfogyasztásához a villa feltehetően jobb eszköz, gombaszeleteléséhez inkább a kés tűnik megfelelőnek. Az asztalon

történő doboláshoz bármelyik megfelelő. Egy steak elfogyasztásához valószínűleg mindkettőt egyszerre fogjuk alkalmazni. Rizsevéshez..., nos..., van, aki a villát kedveli, mások viszont evőpálcikát választanának.

Ha tehát eszközöket hasonlítunk össze, érdemes óvatosnak lennünk. Arra használjuk a hasonlítgatást, hogy megértsük a működésüket, ne arra, hogy bíráljuk őket.

Nem létezik kész vagy tökéletes eszköz

Mint bármely más eszköz, sem a Scrum, sem a Kanban nincs tökélyre fejlesztve. Nem mondanak el mindent arról, hogy mit tegyünk, mindössze bizonyos iránymutatásokat és szabályokat nyújtanak. A Scrum például azt írja elő, hogy keresztfunkcionális csoportokat hozzunk létre és időkeretek közé szorított iterációkat alkalmazzunk. A Kanban szabályai közé tartozik, hogy vizualizáljuk a folyamatot, és korlátozzuk a sorban álló feladatok számát.

Meglehetősen érdekes dolog, hogy egy eszköz értéke éppen abban rejlik, hogy *korlátozza a lehetőségeink számát*. Egy módszertani eszköz, ami szerint bármi megtehető, nem különösebben hasznos; elnevezhetnénk „Csinálj bármit”, vagy akár „Tedd a megfelelőt” módszernek. Ez utóbbi garantáltan működik, ez a csodafegyver! Ha ugyanis nem működött, akkor nyilvánvalóan nem követtük a módszert. ☺

A megfelelő eszközök alkalmazása segít a siker elérésében, de nem garantálja azt. Egyszerű zavart kelteni a projektek sikere/bukása és az eszközök sikere/bukása összekeverésével.

- Egy projekt sikeres lehet egy nagyszerű eszköznek köszönhetően.
- Egy projekt sikeres lehet annak ellenére, hogy gyenge eszközt használtunk.
- Egy projekt elbukhat egy gyenge eszköz miatt.
- Egy projekt elbukhat egy nagyszerű eszköz ellenére is.

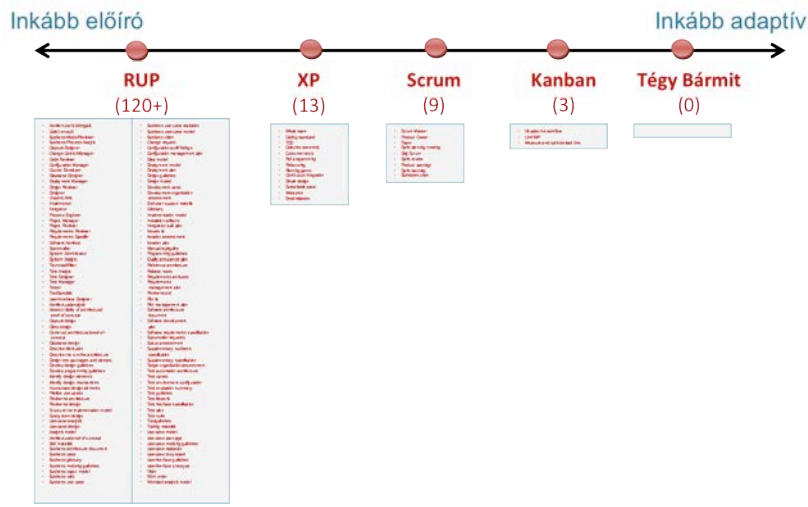
A Scrum előíróbb, mint a Kanban

Összehasonlíthatunk eszközöket aszerint, hogy mennyi szabályt határoznak meg. Az *előíró* szerint „kövess több szabályt”, míg az adaptív azt jelenti, hogy „kövess kevesebb szabályt”. A teljes mértékig előíró módszer nem engedi, hogy használjuk az agyunkat, míg az abszolút adaptív azt jelenti, hogy azt teszünk, amit akarunk, egyáltalán nincsenek megkötések és szabályok. Könnyen belátható, hogy mindkét véglet képtelenség...

Az agilis módszereket néha „könnyednek” nevezik elsősorban azért, mert kevésbé előíróak, mint a tradicionális módszertanok. Tény, hogy az Agilis Kiáltvány első tétele ez: „az egyén és a személyes kommunikáció az eszközök és folyamatok felett”.

A Scrum és a Kanban meglehetősen adaptív módszerek, de a Scrum szigorúbb a Kanbannál: több megkötést tartalmaz, ennélfogva kevesebb lehetőséget hagy nyitva. Előírja például az időkeretek közé szorított iterációkat, amit a Kanban nem.

Hasonlítsunk össze néhány módszertani eszközt az előíró-adaptív-skálán:



A RUP meglehetősen szabályozott – több mint harminc szerepkört, húsz feletti tevékenységet és hetvennél több terméket határoz meg. Iszonyúan sok tanulnivaló... Nem feltétlenül szükséges ugyanakkor mindegyik elem használata, az alkalmazóra van bízva, hogy kiválogassa az adott projektben relevánsakat. Sajnálatos módon ez a gyakorlatban eléggé bonyolultnak bizonyult. „Hmmmm... szükségünk lesz vajon *Konfiguráció*

audit jelentés termékre? Kell-e nekünk Változásmenedzser szerepkör? Nem feltétlenül, de biztos, ami biztos alapon tartsuk meg őket.” Ez lehet az oka annak, hogy a RUP-megvalósítások meglehetősen „nehézsúlyúra” sikerednek a végén, összehasonlítva a Scrum- vagy XP-módszerekkel.

Az XP (eXtreme Programming) meglehetősen előíró a Scrumhoz képest. Majdnem mindent tartalmaz amit a Scrum, kiegészítve egy halom speciális fejlesztési módszerrel, mint például a test-driven development vagy a páros programozás. A Scrum kevésbé előíró, mint az XP – tekintettel arra, hogy egyetlen specifikus fejlesztési technikát sem határoz meg –, ugyanakkor szigorúbb, mint a Kanban, mivel olyan dolgokat ír elő, mint az iterációk vagy a keresztfunkcionális csoportok.

A Scrum és a RUP közötti egyik nagy különbség, hogy a RUP túl sokat ad és a felhasználójára van bízva, hogy megszabadítsa azoktól az elemektől, amikre nincs szüksége. A Scrum kevesebbet ír elő és az alkalmazójára hárul a feladat, hogy hozzátegye azokat az elemeket, amikre még szüksége van.

A Kanban majdnem mindent nyitva hagy. Szinte annyi az összes előírása, hogy „Jelenítsd meg a munkafolyamatot” és „Korlátozd a végrehajtás alatt lévő feladatok (WIP) számát”. Csak egy hajszálnyira van a Tégy bármit módszertől, de mégis meglepően hatékony.

Ne korlátozd magad egyetlen eszközre!

Vegyítsük az eszközöket igényeink szerint! Nehezen tudok elképzelni sikeres Scrum-csoportot anélkül, hogy ne használná például az XP elemeinek többségét. Sok Kanban-csapat tart napi megbeszéléseket (ami Scrum-módszer). Néhány Scrum-csapat a backlog-bejegyzések egy részét használati esetekbe (use case; RUP-elem) szervezve írja meg, vagy korlátozza a WIP-et (Kanban-módszer). Akármelyik módszer működhet.

Musashi (XVII. századi szamuráj, aki híres volt két kardos harci technikájáról) fogalmazta meg találóan:

Soha ne korlátozd magad egyetlen fegyverre vagy harcművészetre!

- Miyamoto Musashi

Tartsuk ugyanakkor tiszteletben minden módszer szabályait! Ha például Scrumot alkalmazunk, és úgy döntünk, hogy nem ragaszkodunk tovább a szigorú időkeretek közé szorított iterációkhoz (vagy a Scrum bármely más szabályához), ne nevezzük többé Scrumnak. A Scrum elég minimalista önmagában is ahhoz, hogyha bármitől megfosztjuk és továbbra is Scrumnak nevezzük, a név értelmét veszítse. Nevezzük valami másnak, például „Scrum-alapú” módszernek, „Scrum-töredéknek”, vagy mit szólnánk a „scrumos” elnevezéshez? ☺

3

A Scrum szerepköröket ír elő

A Scrum három szerepkört ír elő: product owner (meghatározza a termékkel kapcsolatos víziót és a prioritásokat), team (megvalósítja a terméket) és Scrum master (problémák elhárítása és a fejlesztési folyamat vezetése).

A Kanban egyáltalán nem határoz meg semmilyen szerepkört.

Ez persze nem jelenti azt, hogy ne lehetne vagy kellene betölteni a product owner szerepkörét a Kanbanban. Ez mindössze annyit jelent, hogy nem *kötelező*! A Scrum és a Kanban egyaránt lehetővé teszi, hogy bármilyen további szükséges szerepkörrel kiegészítsük őket.

A továbbiak meghatározásával ugyanakkor bánjunk óvatosan! Győződjünk meg arról, hogy az általunk kialakított szerepek hozzáadott értéket képviselnek, és nem ütköznek a módszer más szerepköreivel. Biztos, hogy szükségünk van projektmenedzserre? Egy nagy munkában hasznos lehet, ha ő hangolja össze több csapat és product owner munkáját; de kis projektekben feleslegessé válhat, vagy ami még rosszabb, mikromenedzsmenthez és konfliktusokhoz vezethet.

A Scrum és a Kanban szerint is az az alapelv, hogy „a kevesebb jobb”! Ha bizonytalanok vagyunk, kezdjünk a kevesebbel!

A továbbiakban a product owner kifejezést a tárgyalt módszertől függetlenül arra a szereplőre alkalmazzuk, aki meghatározza a team feladatainak prioritásait.

4

A Scrum időkeretek közé szorított iterációkat ír elő

A Scrum az időkeretek közé szorított iterációkon alapszik. Bármilyen hosszúságút választhatunk, de az általános vélekedés szerint érdemes egy időszakon belül ugyanolyan hosszúságokat alkalmazni, hogy kialakulhasson a fejlesztés megszokott *üteme*.

- **Az iteráció kezdetén** erre vonatkozó tervet hozunk létre: a product owner prioritásai alapján a csapat annyi elemet választ ki a product backlogból, amennyiről úgy gondolja, hogy teljesíteni tudja az adott iterációban (tervezés és vállalás).

- **Az iteráció közben** a csapat a bevállalt elemek befejezésére koncentrál. Az iteráció szkópja rögzített.

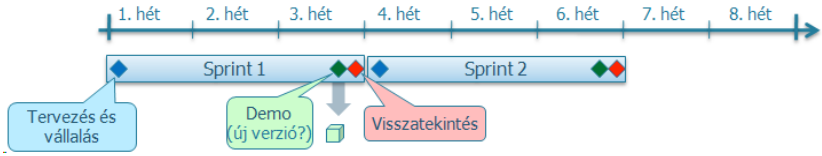
- **Az iteráció végén** a csapat bemutatja a legfontosabb résztvevőknek a működő kódot (demo), amelynek elméletileg potenciálisan szállíthatónak kellene lennie (azaz leteszteltnek és indulásra késznek). A csapat azután értékelést tart (retrospective), és megvitatja, hogyan javíthat a folyamatain.

Egy Scrum-iteráció tehát nem más, mint egyetlen – időkeretek közé szorított – ütem, amely az alábbi három tevékenységet kombinálja: a tervezést, a folyamatfejlesztést és – ideális esetben – a kibocsátást.

A Kanban nem írja elő az időkeretek közé szorított iterációkat. Mi választhatjuk ki, hogy mikor tervezünk, mikor javítunk a folyamatunkon, és mikor bocsátunk ki verziót. Dönthetünk úgy, hogy ezen tevékenységeket bizonyos rendszerességgel végezzük (pl. minden hétfőn verziót adunk), de úgy is dönthetünk, hogy szükség esetén végezzük el a feladatokat (pl. akkor adunk release-t, ha sikerült valami hasznosat előállítanunk).

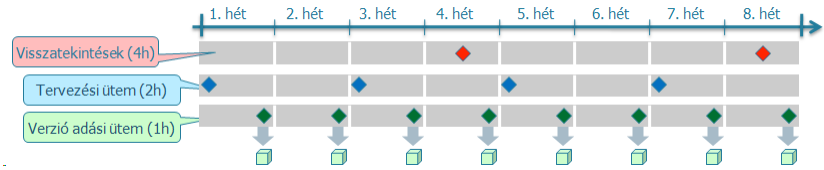
Egyes számú csapat (egyetlen ütem)

“Scrum-iterációkban dolgozunk”



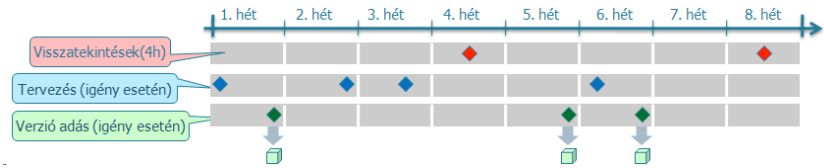
Kettes számú csapat (három ütem)

„Három különböző ütemet alkalmazunk. Minden héten kibocsátjuk, amivel elkészültünk. Minden második héten tervezünk, valamint frissítjük a prioritásokat és a release-terveket. Minden negyedik héten visszatekintést tartunk, hogy tovább fejlesszük a folyamatunkat.”



Hármas számú csapat (főként eseményvezérelt)

„Minden alkalommal, amikor kifogyunk a tennivalókból, tervezést tartunk. Akkor bocsátunk ki egy verziót, amikor már van piacképes funkció. Mindenegyes alkalommal egyeztetünk, ha másodszor futunk bele ugyanabba a problémába, és minden negyedik héten tartunk egy teljesen átfogó visszatekintést is.”

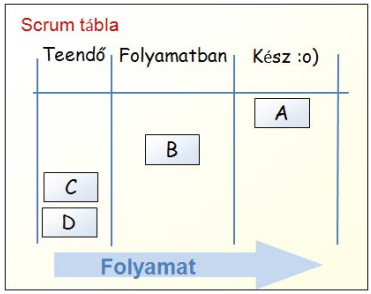


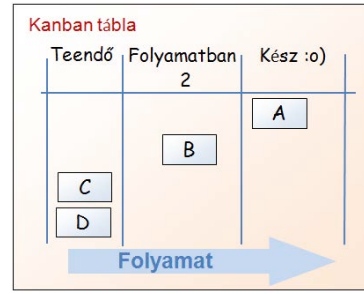
5

A Kanban munkafolyamat lépésenként, a Scrum iterációnként korlátozza a WIP-et

A Scrumban a sprint backlog tartalmazza azokat a feladatokat, amiket az adott iterációban el kell végezni. Ezeket a feladatokat általában a falon lévő kártyák jelenítik meg, amit Scrum- vagy Feladattáblának nevezünk.

Mi a különbség tehát egy Scrum- és egy Kanban-tábla között? Kezdjük egy végtelenül leegyszerűsített példával, és hasonlítsuk össze a kettőt:





Mindkét esetben egy halom elem útját követjük nyomon a munkafolyamaton keresztül. Három állapotot határoztunk meg: Teendő, Folyamatban és Kész. Bármilyen állapotot választhatnánk, ami szimpatikus – néhány csapat továbbiakat ad hozzá, például Integráció, Teszt, Kibocsátás stb. Sose feledkezzünk meg ugyanakkor a *kevesebb több* alapelvről!

Mi a különbség a példában mutatott két tábla között? Igen, a kicsi kettes szám a Kanban-tábla középső oszlopában. Mindössze ennyi. Ez a kettes azt jelenti, hogy „ez az oszlop egy pillanatban sem tartalmazhat kettőnél több elemet”.

A Scrumban nincs olyan szabály, ami megakadályozná a csapatot abban, hogy az összes elemet egyszerre helyezze a Folyamatban oszlopba! Létezik ugyanakkor egy implicit korlát, hiszen az iterációnak önmagában meghatározott terjedelme van. Ebben az esetben az egy oszlopban elhelyezhető feladatok korlátja négy, hiszen mindössze ennyi szerepel a

táblán. A Scrum tehát indirekt módon korlátozza a WIP-et, míg a Kanban direkt.

A legtöbb Scrum-csapat végül is belátja, hogy nem jó ötlet túl sok feladatot folyamatban tartani, és kialakítanak egy rendet arra, hogy mielőtt nekilátnak egy új feladat elvégzésének, a folyamatban lévőt lezárják. Néhány csapat ráadásul rá is szánja magát arra, hogy explicit korlátozza a folyamatban lévő feladatok számát – és lám! – a Scrum-tábla Kanban-táblává alakult.

Tehát mind a Scrum, mind a Kanban korlátozza a feldolgozás alatt álló feladatok számát, csak különböző módon. A Scrum-csapatok általában mérik a sebességet – mennyi feladatot (vagy munkamennyiség mértékét mérő egységet, pl. „sztoripontot") képesek elvégezni egy iteráció alatt. Amint egy csapat ismeri saját sebességét, az lesz az egy iterációra vonatkozó WIP-korlátjuk (vagy legalábbis egy iránymutatás arra). Ha egy csapat átlagos sebessége például tíz elem (vagy sztoripont), akkor ennél többet általában nem tervez egy sprintre.

A Scrumban tehát a *WIP időegységre vonatkoztatva korlátozott*.

A Kanbanban a *WIP munkafolyamat-lépésekre korlátozott*.

A fenti Kanban-példában bármely pillanatban maximum két elem lehet a Folyamatban munkafázisban, függetlenül a fejlesztés ütemezésétől. Magunknak kell megválasztanunk, hogy mely munkafázishoz milyen WIP-korlátot határozunk meg, de az általános szabály az, hogy az értékáram lehető legkorábbi és legkésőbbi lépése között mindenegyes fázishoz rendeljünk ilyen korlátot. A fenti példában tehát törekedni kell arra, hogy a Teendő (vagy akárhogy nevezzük az első fázist) oszlop elemszámát is korlátozzuk. Ha már egyszer meghatároztuk a WIP-korlátokat, megkezdhetjük mérni és előrejelezni az átfutási időt, azaz egy elem összes munkafázison történő áthaladásához összesen szükséges átlagos időt. Az előrejelezhető átfutási idő teszi lehetővé számunkra a kötelezettségvállalást és a reális tervek készítését.

Ha az egyes feladatok mérete nagymértékben különböző, akkor érdemes megfontolni a WIP-korlátok sztoripontban vagy bármely más feladatméretet kifejező mértékegységben történő meghatározását. Néhány csapat külön energiát fektet abba, hogy a teendőket közel azonos méretű elemekre bontsa, hogy elkerülje a különböző feladatméretek kezeléséből adódó komplikációkat és csökkentse a becslések idejére fordított időt. Könnyebb zökkenőmentes áramlást megvalósítani, ha a feladatok nagyjából azonos méretűek.

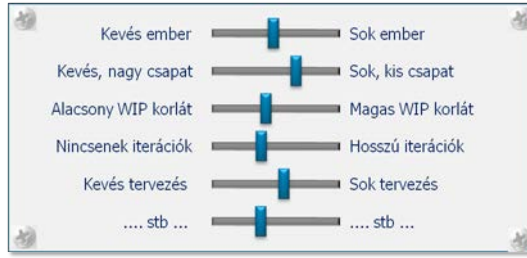
6

Mindkettő empirikus

Képzeljük el, hogy ezen mutatók szabályozásával szabadon konfigurálhatjuk folyamatainkat. „Nagy fejlesztési sebességet akarok, alacsony átfutási idővel, magas minőséggel és nagyfokú tervezhetőséggel. Tehát a mutatókat rendre 10, 1, 10, 10 értékekre állítom be.”

Ez nagyszerű lenne, ugye? Sajnos nem léteznek ilyen direkt szabályozók, legalábbis én nem ismerek ilyeneket.

Helyette van egy halom *indirekt* szabályozási eszközünk.



A Scrum és a Kanban empirikus rendszerek abban az értelemben, hogy kísérletezésen keresztül a felhasználónak kell saját körülményeinek megfelelőre hangolnia őket. Sem a Scrum, sem a Kanban nem adja meg a választ mindenre – mindössze az alapvető szabályokat határozzák meg ahhoz, hogyan fejleszthetjük saját folyamatainkat.

- A Scrum azt javasolja, hogy állítsunk fel keresztfunkcionális csapatokat. Kik legyenek egy csapatban? Nem tudjuk, kísérletezni kell.

- A Scrum szerint a csapatnak kell meghatároznia, hogy mennyi munkát emeljen be egy sprint terjedelmébe, tehát pontosan mennyit vállaljon. De mennyi legyen az? Nem tudjuk, kísérletezni kell.

- A Kanban szerint korlátozni kell a WIP-et. Mekkorára állítsuk be a WIP-korlátot? Nem tudjuk, kísérletezni kell.

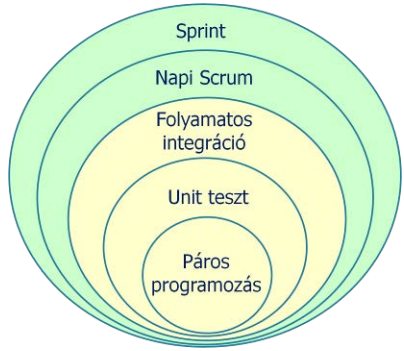
Ahogyan korábban említettük, a Kanban kevesebb szabályt ír elő, mint a Scrum. Ez azt jelenti, hogy több paraméter szabályozásáról kell magunknak gondoskodni. Ez, a körülményektől függően, egyszerre lehet előny vagy hátrány. Ha például megnyitjuk egy szoftver paraméterező felületét, melyik esetet preferáljuk: ha három paramétert állíthatunk be, vagy ha százat? Feltehetően valahol a kettő között. Attól függ, hogy mekkora rugalmasságra van szükségünk, és hogy mennyire ismerjük az eszközt.

Tegyük fel tehát, hogy csökkentjük a WIP-korlátot arra a feltételezésre alapozva, hogy ez hatékonyabbá teszi folyamatainkat. Ezek után megfigyeljük, hogy milyen irányba alakultak a sebességet, átfutási időt, minőséget és tervezhetőséget jelző mutatóink. Az eredmények alapján következtetéseket vonunk le, és további módosításokat végzünk, hogy folyamatosan javítsuk a folyamatainkat.

Erre több elnevezést is használnak. *Kaizen* (folyamatos fejlődés a Lean-szóhasználatban), *vizsgál és adaptál* (Scrum-terminológia), *empirikus folyamatirányítás*, vagy miért ne lehetne *tudományos módszer*?

Mindezek legfontosabb eleme a ***visszacsatolási hurok***. Változtass valamin => vizsgáld meg mi lett az eredménye => tanulj belőle => ismét változtass valamin. Általában véve olyan rövid visszacsatolási hurkot akarunk, amennyire csak lehet, így gyorsan javíthatjuk a folyamatainkat.

A Scrumban az alap visszacsatolási hurok a sprint. Ennél lényegében több is van, különösen, ha az XP-technikáival együtt alkalmazzuk:



Helyesen alkalmazva a Scrum és az XP egy sor rendkívül értékes visszacsatolási ciklussal szolgál számunkra.

A legbelső visszacsatolási mechanizmus a páros programozás, mindössze néhány másodpercen belül biztosítja a visszajelzést. A hibák megtalálása és kijavítása mindössze néhány pillanatot vesz igénybe („Ennek a változónak nem háromnak kellene lennie?”). Ez a „jól fejlesztjük a terméket?” visszacsatolási ciklus.

A külső visszacsatolási hurok, a sprint, néhány héten belül biztosít visszajelzést. Ez biztosítja a „jó terméket fejlesztünk?” visszacsatolási ciklust.

Mi a helyzet a Kanbannal? Nos, az előbb említett visszacsatolási hurkok mindegyikét alkalmazhatjuk (sőt, javasolt alkalmaznunk), akár Kanbant használunk, akár nem. A Kanban kevés, de nagyon hasznos, valós idejű mérőszámot kínál.

- Átlagos átfutási idő. Minden alkalommal frissítendő, amint egy elem a Kész (vagy akárhogy is nevezzük a jobb szélső oszlopot) állapotba kerül.

- Szűk keresztmetszetek. Jellemző tünet, hogy az X oszlop zsúfolt feladatokkal, míg az X+1 oszlop üres. Keressük a „légbuborékokat” a táblán!

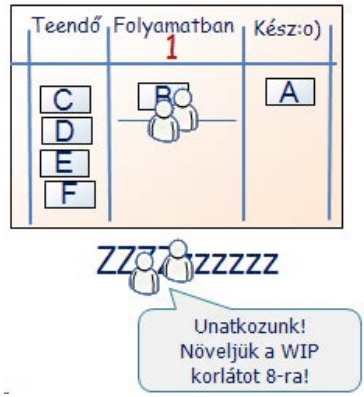
A valós idejű mérőszámok szépsége abban rejlik, hogy magunk választhatjuk meg a visszacsatolási ciklus hosszát annak függvényében, milyen gyakran szánjuk rá magunkat az eredmények kiértékelésére és a változtatások bevezetésére. A túl hosszú visszacsatolási ciklus azt eredményezi, hogy folyamatfejlesztésünk üteme lassú lesz. A túl rövid visszacsatolási idő eredményeként előfordulhat, hogy folyamataink nem tudnak stabilizálódni a változtatások között, ami káoszhoz vezethet.

Valójában a visszacsatolási ciklus hossza az egyik olyan tényező, amivel kísérletezhetünk... egyfajta meta-visszacsatolásként. Na jó, itt befejeztük :-)

Példa: Kísérletezés a Kanban WIP-korláttal

A Kanban egyik tipikus kulcskérdése a WIP-korlát. Honnan tudjuk, hogy jól határoztuk meg?

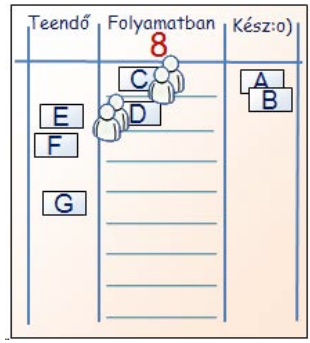
Tegyük fel, hogy egy négyfős csapattal dolgozunk, és a WIP-korlát értékét egyben határozzuk meg.



Valahányszor hozzálátunk egy feladat megoldásához, nem indíthatunk újat addig, amíg az elsőt le nem zártuk, így nagyon gyorsan Kész státuszba kerül.

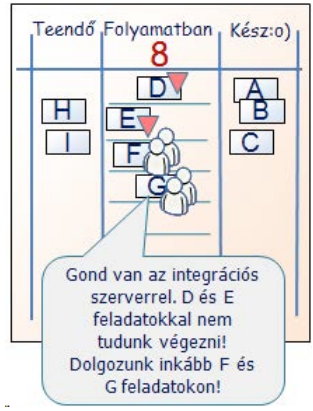
Nagyszerű! De hamar kiderül, hogy általában nem túl kifizetődő, hogy mind a négy ember ugyanazon a feladaton dolgozzon (ebben a példában), tehát két emberünk tétlen marad. Ha ez hosszú idő alatt csak egyszer fordul elő, nem nagy probléma, de ha rendszeresen, annak következményeként az átlagos átfutási időnk növekedni fog. Végső soron az egy WIP-korlát azt jelenti, hogy a feladat gyorsan továbblép a Folyamatban státuszból, ha egyszer oda kerül, viszont a szükségesnél tovább marad a Teendő állapotban, tehát a teljes folyamaton való átfutási idő szükségtelenül hosszú lesz.

Ha tehát az egy WIP-korlát túl alacsony volt, mi lenne, ha felemelnénk nyolcra?

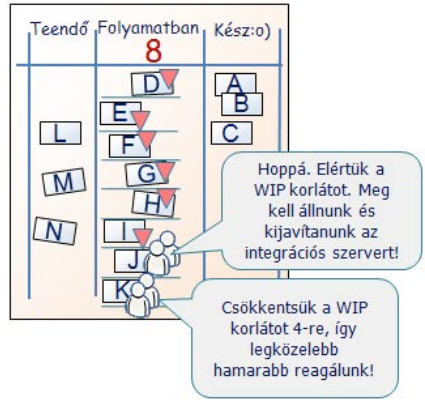


Ez egy darabig jobban működik. Észrevesszük, hogy a párokban történő munkavégzés gyorsabb feladatmegoldást tesz lehetővé, egy négyfős csapattal tehát bármely adott időpontban általában két folyamatban lévő feladatunk lesz. A nyolcas WIP-korlát csak felső határ, tehát ha ennél kevesebb van folyamatban, az még jó.

Most képzeljük el, hogy az integrációs szerverrel egy problémába futottunk bele, ezért egy feladatot sem tudunk teljes egészében lezárni (a Kész definíciója nálunk tartalmazza az integrációt is). Ilyesmi előfordul néha, ugye?

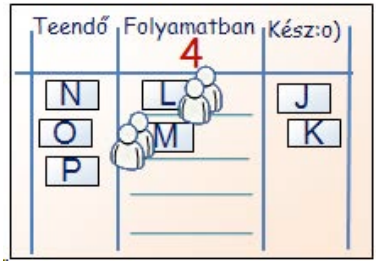


Mivel nem tudjuk lezárni a D és E elemeket, dolgozni kezdünk az F feladaton. Mivel azt sem tudjuk integrálni, kiveszünk egy új elemet, a G-t. Egy idő után a Folyamatban státuszban elérjük a nyolcas WIP-korlátot.



Ezen a ponton már nem tudunk több elemen dolgozni, ezért jobban járunk, ha kijavítjuk azt az integrációs szervert! A WIP-korlát figyelmeztetett bennünket arra, hogy reagáljunk és szüntessük meg a szűk keresztmetszet okozóját ahelyett, hogy egy rakás befejezetlen feladatot halmoznánk fel.

Ez jó. De ha a WIP-korlát négy lett volna, akkor sokkal hamarabb reagálunk, és jobb átlagos átfutási időt érhetünk el – ez az egyensúly. Mérjük az átlagos átfutási időt, és addig finomítjuk a WIP-korlátot, amíg el nem érjük a legjobb időértéket.



Egy idő után előfordulhat, hogy a Teendő oszlopban feltorlódnak a feladatok. Ilyenkor itt az ideje, hogy növeljük a WIP-korlátot.

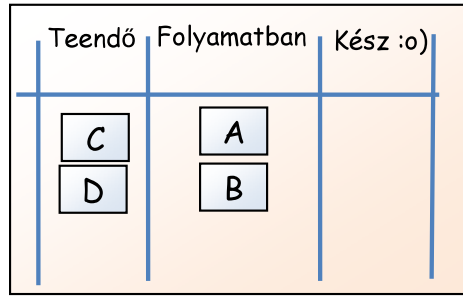
Miért van egyáltalán szükségünk a Teendő oszlopra? Nos, ha a megrendelő mindig elérhető lenne a fejlesztőcsapat számára, hogy bármikor megkérhessék, hogy megmondja, mi legyen a következő elvégzendő feladat, akkor a Teendő oszlopra nem lenne szükség. De ebben az esetben a megrendelő nem érhető el bármikor, tehát a Teendő oszlop egy kis puffert képez a csoport számára a köztes időben.

Kísérletezzünk! Vagy ahogy a „Scrumológus” mondaná: vizsgálódj és adaptálj!

7

A Scrum ellenáll az iterációkon belüli változtatásoknak

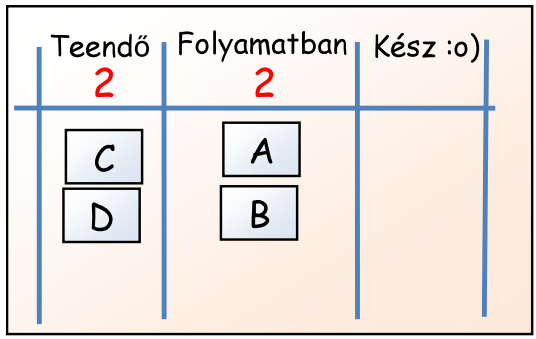
Tegyük fel, hogy a Scrum-táblánk a következő módon néz ki:



Mi van, ha valaki felbukkan, és az E feladatot szeretné elhelyezni a táblán?

A Scrum-csapat reakciója feltehetően valami ilyesmi lesz: „Sajnáljuk, de nem lehet, mi ebben a sprintben az A, B, C és D feladatokat vállaltuk, viszont az E feladatot nyugodtan be lehet tenni a product backlogba. Ha a product owner elég magas prioritásúnak találja, beemeljük a következő sprintbe.” A megfelelő hosszúságú sprintek éppen elegendő időt hagynak a fejlesztői csapatnak, hogy valamit elkészítsenek, valamint lehetőséget biztosítanak a product owner számára a prioritások rendszeres meghatározására.

Hogyan kezeli ezt a helyzetet egy Kanban-team?



A Kanban-csapat azt mondhatja, hogy „Nyugodtan helyezd az E feladatot a Teendő oszlopba. Mivel azonban a WIP-korlátunk erre az oszlopra kettő, ezért a C és D elemek közül valamelyiket ki kell onnan venni. Jelenleg az A és B feladatokon dolgozunk, de amint lesz szabad kapacitásunk, megkezdjük a Teendő oszlop legfontosabb feladatának végrehajtását.”

A Kanban-team válaszideje (azaz amennyi idő alatt a prioritások változására reagál) tehát annyira hosszú, amennyi idő alatt szabad kapacitása szabadul fel az „egy elem ki = egy elem be” szabályt követve (amit a WIP-korlát vezérel).

A Scrumban az átlagos válaszidő a sprint hosszának fele.

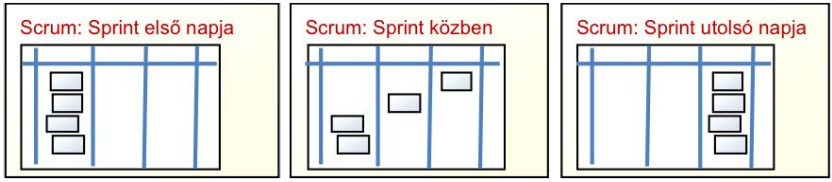
A Scrumban a product owner nem nyúlhat hozzá a Scrum-táblához, mivel a team egy iterációban az előre meghatározott feladatokra vállalt kötelezettséget. A Kanbanban magunknak kell kialakítanunk a saját szabályainkat arra nézve, hogy ki és mit változtathat a táblán. Jellemzően a product owner számára fenntartanak egy Teendő, Előkészített, Backlog vagy Előterjesztett nevű oszlopot a bal szélen, amelyben szabadon variálhatja a feladatokat.

Ez a két megközelítés egyébként nem zárja ki egymást. Egy Scrum-csapat is *dönthet* úgy, hogy lehetővé teszi a product owner számára, hogy a sprint közben is változtathasson a prioritásokon (igaz, ezeket kivételes eseteknek kell tekinteni). A Kanban-team szintén *dönthet* úgy, hogy korlátozza a prioritások változtatásának szabadságát, vagy akár úgy is, hogy, hasonlóan a Scrumhoz, időkorlátok közé szorított, rögzített szkópú iterációkat alkalmaz.

8

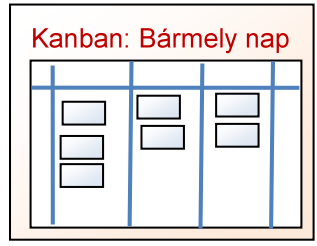
A Scrum-tábla minden iteráció után alaphelyzetbe kerül

A Scrum-tábla tipikusan a következőképpen néz ki a sprint egyes fázisaiban:



Amikor a sprint véget ér, a tábla alaphelyzetbe kerül, azaz minden elemet eltávolítanak róla. A következő sprint elkezdődik, és a tervezés után új Scrum-tábla vár ránk, amelynek a bal oldalán sorakoznak az új elemek. Technikailag úgy tűnhet, ez időpazarlás, de egy gyakorlott Scrum-csapat számára ez a művelet általában nem tart sokáig, és a tábla alaphelyzetbe állítása, a befejezés és lezárás egyfajta kellemes érzést ad. Olyan, mintha elmosogatnánk vacsora után: nincs kedvünk hozzá, de utána jól érezzük magunkat.

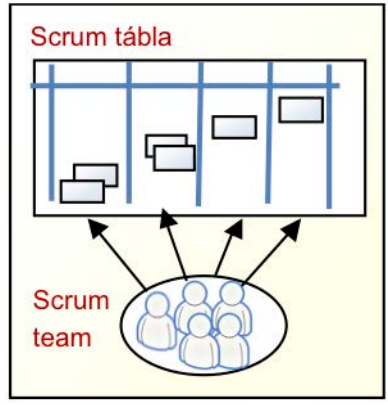
A Kanbanban a tábla általában folyamatos dolog, nem szükséges alaphelyzetbe állítani és újra indítani.



9

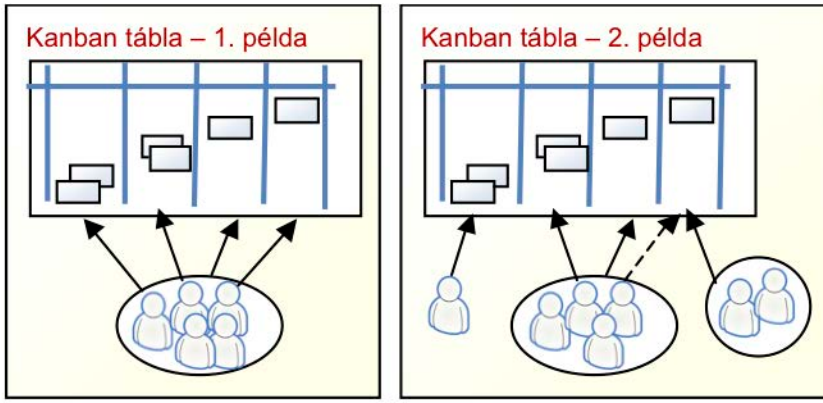
A Scrum keresztfunkcionális csapatokat ír elő

Egy Scrum-táblát csak egy csapat használhat. A csapatnak keresztfunkcionálisnak kell lennie, azaz tartalmaznia kell mindazon szaktudást és képességet, amellyel az iteráció összes elemét teljesíteni tudja. A táblát általában bárki láthatja, akit érdekel, de csupán a csapat változtathat rajta. Ez az ő eszközük, amelyen az iteráció vállalásait kezelik.



A Kanbanban opcionálisak a keresztfunkcionális csapatok, és az sem szükséges, hogy a táblát egyetlen csapat birtokolja. A tábla egy munkafolyamathoz tartozik, nem feltétlenül egy csapathoz.

Íme két példa:



1. példa: Az egész tábla egyetlen keresztfunkcionális csapatot szolgál ki csakúgy, mint a Scrumban.

2. péda: A product owner az 1. oszlopban állítja fel a prioritásokat. Egy keresztfunkcionális csoport fejleszt (2. oszlop), illetve tesztel (3. oszlop), a kibocsátást (4. oszlop) egy specialista csapat végzi. Valamennyi átlapolódás a kompetenciákban van, ezért ha a kibocsátó csapat válik a szűk keresztmetszetté, akkor valamelyik fejlesztő besegít.

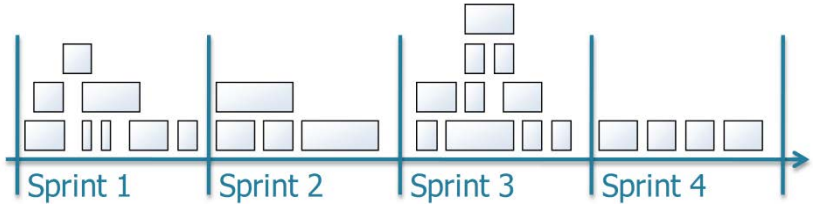
A Kanbanban tehát pár alapszabályt kell felállítani, hogy kik és hogyan használhatják a táblát. Kísérletezzünk azután a szabályokkal, hogy optimalizálhassuk a folyamatot!

10

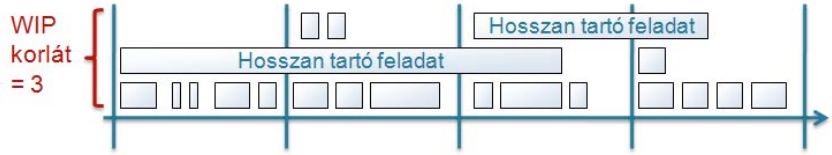
A Scrum backlog elemeinek bele kell férniük egy sprintbe

Mind a Scrum, mind a Kanban az inkrementális fejlesztésen alapszik, azaz a munkát kisebb részekre bontják.

Egy Scrum-csapat csak olyan elemeket vállal el, amelyekről azt gondolja, hogy – a Kész definíciója alapján – egy iteráción belül képes teljesíteni. Ha egy elem túl nagy ahhoz, hogy beférjen a sprintbe, a csapat és a product owner együttesen próbálja kitalálni, hogyan lehetne kisebb részekre bontani. Ha az elemek rendszerint nagyméretűek, az iteráció is hosszabb lesz (de általában négy hétnél nem hosszabb).



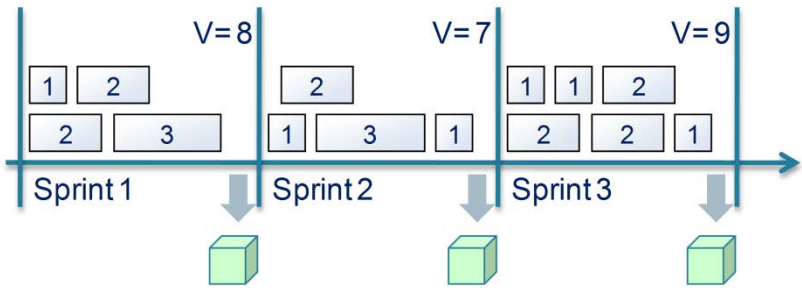
A Kanban-csapatok megpróbálják minimalizálni az átfutási időt és szintekre bontani a folyamatot, amelynek közvetett hatása, hogy az elemeket viszonylag kisméretű részekre bontják. Arra azonban nincs kimondott szabály, hogy az elemeknek bele kellene férniük valamely időkorlátba. Ugyanazon táblán lehet egy hónapig, de akár egy napig tartó feladat is.



11

A Scrum előírja a tervezést és a sebesség mérését

A Scrum elvárja a csapatoktól, hogy megbecsüljék a bevállalt feladatok egymáshoz viszonyított méretét. Azzal, hogy minden, a sprint végéig elvégzett feladat méretét összeadjuk, megkapjuk a csapat sebességét (v). A sebesség a csapat kapacitásának mérését szolgálja – mennyi feladatot tud elvégezni sprintenként. Íme egy nyolcas átlagsebességű csapat példája.



Azért hasznos tudni, hogy a csapat sebessége nyolc, mert így reális előrejelzést adhatunk arról, hogy melyik feladatokat tudjuk teljesíteni a következő sprintben, ami végül reális release-tervek készítését teszi lehetővé.

A Kanban nem írja elő a tervezést. Ha tehát vállalásokat kell tennünk, ki kell találnunk, hogyan teremtsük meg az előrejelezhetőség feltételeit.

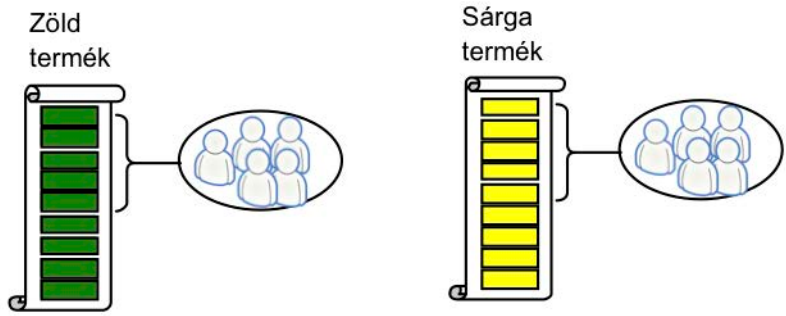
Néhány csapat azt választja, hogy – a Scrumhoz hasonlóan – becsül, és méri a sebességet. Mások úgy döntenek, hogy kihagyják a becslést, de megpróbálják közel azonos méretű részekre szétbontani a feladatot – ezáltal egyszerűen az adott időegység (pl. hetente) alatt elvégzett feladatok számával mérik a sebességet. Vannak, akik a feladatokat úgynevezett MMF-ekbe (minimum marketable feature, minimális hasznos funkció) csoportosítják, az MMF átlagos átfutási idejét mérik, és ez alapján állapítják meg saját szolgáltatási szintjüket – pl. „ha egy MMF fejlesztését bevállaljuk, akkor az minden esetben tizenöt napon belül leszállítható lesz”.

Mindenféle érdekes technika létezik a Kanban-stílusú release-tervezésre és menedzsmentre, de nincs semmilyen kötelező módszer sem előírva – szóval gyerünk és keressünk az interneten, amíg meg nem találjuk azt, ami illeszkedik a helyzetünkhöz. Idővel feltehetően láthatunk majd kifejlődni néhány „legjobb gyakorlatot”.

12

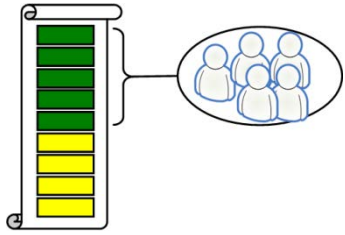
Mindkettő lehetővé teszi, hogy több terméken dolgozzunk párhuzamosan

A Scrumban alkalmazott product backlog elnevezés meglehetősen szerencsétlen, mivel azt sugallja, hogy minden elem ugyanahhoz a termékhez tartozik. Alább látható két termék a zöld és sárga, mindkettő a saját product backlogjával és fejlesztő csapatával:

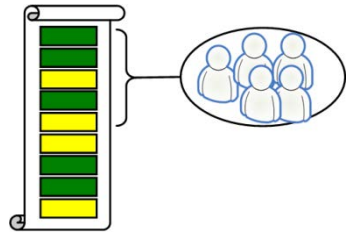


Mi a helyzet akkor, ha csak egyetlen csapatunk van? Gondoljunk inkább a product backlogra, mint egy team backlogra, ami egy adott csapat (vagy csapatok) következő iterációinak priorizált feladatait tartalmazza. Ha tehát a csapat több terméken dolgozik, egyesítsük az összes feladatait egyetlen listába. Ez rákényszerít bennünket arra, hogy rangsoroljunk a termékek között, ami néhány esetben hasznos lehet. A gyakorlatban több módszer is kínálkozik erre.

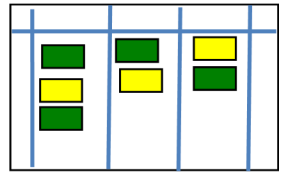
Egy lehetséges megközelítés, hogy a csapat figyelmét egy sprintben egy termékre koncentráljuk:



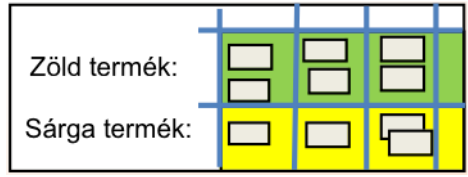
Másik megközelítés lehet, hogy a csapat egy sprinten belül mindkét termék funkcióin egyszerre dolgozik:



Ugyanez a helyzet a Kanban estén. Egyszerre több termékünk is végighaladhat ugyanazon tábla folyamatain. Megkülönböztethetjük őket különböző színű kártyákkal:



Vagy alkalmazhatunk vízszintes sávokat:



13

Mindkettő Lean és agilis

Most nem fogjuk részletesen elemezni a *Lean szemlélet* című könyvet és az agilis kiáltványt, de általában véve mind a Scrum, mind a Kanban jól illeszkedik ezekhez az értékekhez és alapelvekhez. Például:

- Mind a Scrum, mind a Kanban húzórendszert valósít meg, ami illeszkedik a Lean JIT (just in time) készletkezelési elveihez. Ez azt jelenti, hogy a csapat választja meg, mikor és mennyi munkát vállal fel, azaz amint végeztek egy feladattal, egy újat „húznak ki" ahelyett, hogy valaki kívülről „tolná" rájuk azokat. Ahogyan egy nyomtató is csak akkor húzza ki a következő lapot, ha az előzővel végzett (jóllehet egy korlátozott méretű köteget előre betöltenek a tárolóba).

- A Scrum és a Kanban egyaránt tapasztalati alapon történő állandó folyamatoptimalizálást valósít meg, ami illeszkedik a Lean Kaizen (folyamatos javítás) alapelvéhez.

- A Scrum és a Kanban is a változásra történő reagálást hangsúlyozza a tervek rigorózus követése helyett (jóllehet a Kanban tipikusan gyorsabb reakcióidőt tesz lehetővé, mint a Scrum), ami egyike az agilis kiáltványban megfogalmazott négy értékének

... és folytathatnánk.

Bizonyos szempontból a Scrum kevésbé Leannek tűnhet, mivel a feladatok időkeretek közé szorított iterációkba történő összegyűjtését írja elő. De mindez csak az iterációk hosszúságától, és attól függ, hogy mihez hasonlítjuk. Egy tradicionálisabb módszerhez hasonlítva – ahol talán évente kettő–négy alkalommal szállítunk le valamit – egy kéthetente új funkciót kibocsátó Scrum-csapat kifejezetten Lean.

Ha viszont folyamatosan rövidebb és rövidebb iterációkat kezdünk bevezetni, alapvetően közelíteni kezdünk a Kanbanhoz. Amint arról kezdünk beszélgetni, hogy az iterációk hosszát egy hét alá csökkentsük, érdemes megfontolnunk az időkeretes iterációk teljes elhagyását.

Eddig is mondtuk, és továbbra is javasoljuk: kísérletezzünk, amíg nem találunk valamit, ami jól működik nálunk! Aztán folytassuk a kísérletezést…

14

Apró különbségek

Van néhány különbség, ami kevésbé látszik jelentősnek, mint a fent bemutatottak. Ennek ellenére érdemes tudatában lenni ezeknek is.

A Scrum priorizált product backlogot ír elő

A Scrumban a priorizálás minden esetben a product backlog elemeinek sorba rendezésén keresztül történik és hatását a következő sprintre fejti ki (nem pedig a folyamatban lévőre). A Kanbanban tetszőleges priorizációs mechanizmust alakíthatunk ki (akár el is hagyhatjuk), amely hatását azonnal kifejti, amint a csapatnak kapacitása szabadul fel (nem pedig fix időközönként). A product backlog vagy létezik, vagy nem és vagy priorizált, vagy nem.

A gyakorlatban ez kevés különbséget jelent. A Kanbanban jellemzően a bal szélső oszlop tölti be a Scrum product backlogjának szerepét. A feladatlista akár priorizált, akár nem, a csapatnak döntési szabályokra van szüksége, amik alapján meghatározza, hogy melyik legyen a következő elvégzendő feladat. Példák döntési szabályokra:

- Mindig a legfelső elemet vesszük ki.
- Mindig a legrégebbi elemet vesszük ki (tehát minden elem időbélyeggel rendelkezik).
- Bármelyik elemet kivehetjük.
- Az időnk kb. húsz százalékát töltsük támogatási feladatokon, és nyolcvan százalékát új funkciók fejlesztésén.
- A csapat kapacitását egyenlő arányban osszuk el A és B termékek között.
- Ha van piros elem, akkor azt vegyük ki elsőnek.

A product backlogot a Scrumban is kezelhetjük Kanban-szerűen; azaz korlátozhatjuk az elemszámot és felállíthatunk a priorizálás módjára vonatkozó szabályokat.

A Scrum napi megbeszéléseket ír elő

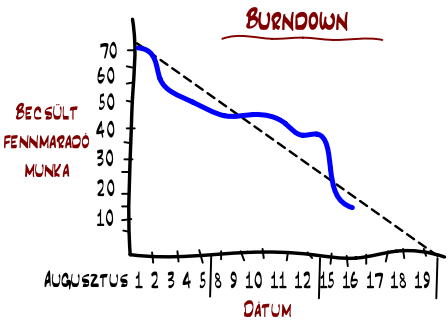
A Scrum-csapat minden nap rövid (maximum tizenöt perc hosszúságú) megbeszélést tart ugyanabban az időpontban, ugyanazon a helyen. Ennek célja az információk megosztása arról, hogy mi történik, mik az aznapra tervezett feladatok, és mik a munkát akadályozó legnagyobb problémák. Ezt gyakran napi standupnak (felállásnak) nevezik, mivel a résztvevők általában állnak a megbeszélés alatt (azért, hogy röviden és aktívan tarthassák).

A Kanban nem írja elő a napi megbeszéléseket, de a legtöbb team ennek ellenére alkalmazza. Ez nagyon jó technika függetlenül attól, hogy melyik módszert alkalmazzuk.

A Scrumban a megbeszélés személyközpontú – minden tag egyenként számol be. Sok Kanban-csapat inkább táblaorientált értekezletet tart, a szűk keresztmetszetekre és az egyéb látható problémákra koncentrálva. Ez a megközelítés jobban skálázható. Ha például négy csapat osztozik ugyanazon a táblán közös standup megbeszélésükön, rövidebb ideig tart, ha csak a tábla problémákat tartalmazó részeire fókuszálunk.

A Scrum előírja a burndown-grafikon vezetését

A sprint burndown chart (lefutási grafikon) napi szinten mutatja, hogy mennyi munka van hátra az adott iterációban vállalt összes feladat befejezéséig.



Az Y tengely mértékegysége megegyezik a sprintfeladatok méretezésénél használt egységekkel, jellemzően órák vagy napok (ha a team feladatokká bontja szét a backlog elemeit) vagy sztoripontok. Rengeteg variáció létezik erre.

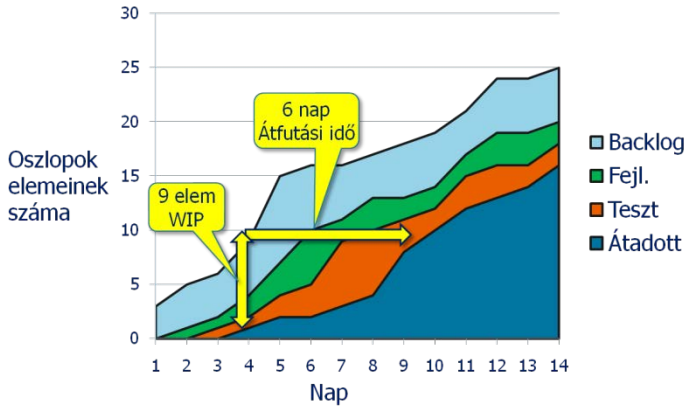
A Scrumban az iteráció előrehaladásának elsőszámú követő eszközeként a sprint lefutási grafikonokat használják.

Néhány csapat release lefutási grafikonokat is használ, amik ugyanígy néznek ki, csak release szintet jelenítenek meg – jellemzően azt mutatják, hogy egy-egy sprint után mennyi sztoripont maradt hátra a release befejezéséig.

A lefutási grafikon elsődleges célja, hogy egyszerűen követhető legyen, hogy a csapat az ütemtervhez képest előrébb jár, vagy elmaradt attól azért, hogy gyorsan reagálhasson a kialakult helyzetre.

A Kanban nem írja elő a burndown chart alkalmazását, tulajdonképpen semmilyen grafikon alkalmazása nincs előírva; ugyanakkor természetesen bármilyen grafikont (beleértve a burndown chartot is) alkalmazhatunk, ha akarunk.

Nézzünk egy példát a kumulatív flow diagramra, amely azt mutatja meg, hogy mennyire kiegyensúlyozott a fejlesztési folyamatunk és a WIP-korlát hogyan hat az átfutási időnkre.



Ez a következők szerint működik. Minden nap adjuk össze a Kanban-tábla minden oszlopában lévő elemek számát, és jelenítsük meg ezeket az Y tengelyen. Ezek szerint a negyedik napon kilenc elem volt a Kanban-táblán. A jobb szélső oszloptól kezdve egy elem volt az átadott oszlopban, egy a tesztelésben, kettő a fejlesztésben és öt a backlogban. Ha minden nap megjelöljük és összekötjük ezeket a pontokat, a fenti diagramhoz hasonlót kapunk. A függőleges és vízszintes nyilak mutatják az összefüggést a WIP-korlát és az átfutási idő között.

A vízszintes nyíl azt mutatja meg nekünk, hogy a negyedik napon a backlogba tett elem hat nap alatt jutott el az átadásig, a tesztelés körülbelül ez idő fele volt. Láthatjuk, hogyha a tesztelés és backlog oszlopok WIP-korlátját csökkentenénk, jelentősen rövidíthetnénk az átfutási időn.

A sötétkék terület lejtése mutatja a sebességet (értsd naponta átadott elemek száma). Idővel láthatjuk, hogy a nagyobb sebesség hogyan csökkenti az átfutási időt, míg a magasabb WIP-korlát növeli azt.

A legtöbb szervezet gyorsabban akarja elkészíteni a dolgait (= csökkenteni az átfutási időt). Sajnos sokan esnek abba a hibába, hogy ezt több ember több túlóráztatásán keresztül akarják elérni. Általában a dolgok elkészítésének gyorsításához vezető legjobb módszer a folyamat kiegyensúlyozása és a munkamennyiség illesztése a csapat kapacitásához, nem pedig a még több ember még keményebb dolgoztatása. Ez a diagram megmutatja hogy miért, így növelve az esélyét annak, hogy a csapat és a menedzsment együttműködése hatékony lesz.

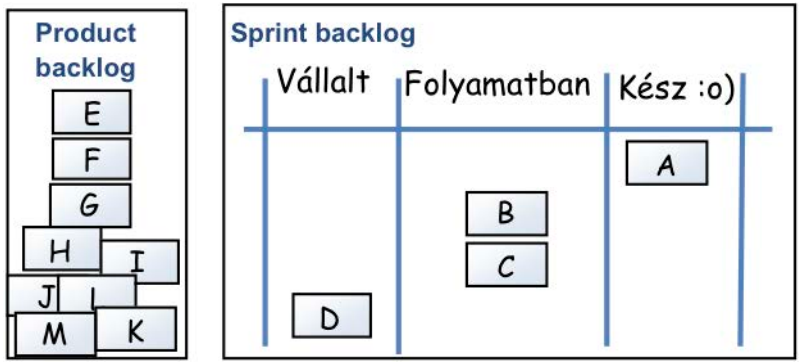
Ez még világosabbá válik, ha különbséget teszünk a sorban állás (pl. tesztelésre várakozó) és végrehajtás (pl. tesztelés alatt) állapotok között. Teljes mértékben minimalizálni akarjuk a sorban várakozó elemek számát, amihez a kumulatív flow diagram mutatja a jó irányokat.

15

Scrum-tábla vs. Kanban-tábla - egy kevésbé triviális példa

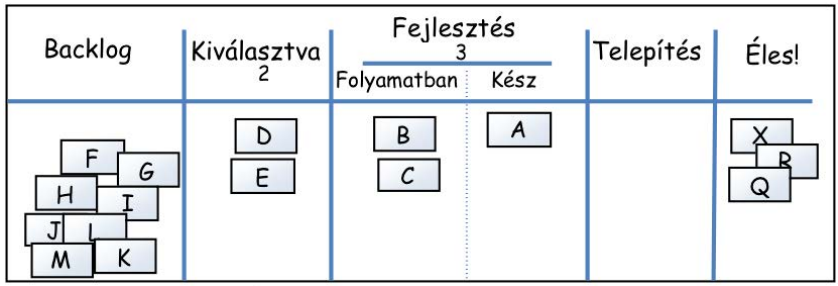
A Scrumban a sprint backlog a teljes képnek csak egy része – az, amely megmutatja, hogy a csapat min dolgozik az adott sprintben. A másik része a product backlog, azon dolgok listája, amiket a product owner a távolabbi sprintek során akar megvalósítani.

A product owner láthatja, de nem érintheti a sprint backlogot. A product backlogot bármikor kedve szerint módosíthatja, de ennek nem lesz hatása (legalábbis az elvégzett feladatokra) egészen a következő sprintig.



Amikor a sprintnek vége, a csapat „potenciálisan szállítható” kódot állított elő a product owner számára; lezárja a sprintet, elvégzi a kiértékelést és büszkén bemutatja a product owner számára az elkészült A, B, C és D funkciókat. A product owner ekkor eldöntheti, hogy ezek a funkciók élesbe álljanak-e, vagy sem. Ez az utolsó mozzanat – a termék tulajdonképpeni élesbe állítása – általában nem része a sprintnek, ezért nem is látható a product backlogban.

E forgatókönyv szerint a Kanban-tábla inkább valahogy így nézne ki:



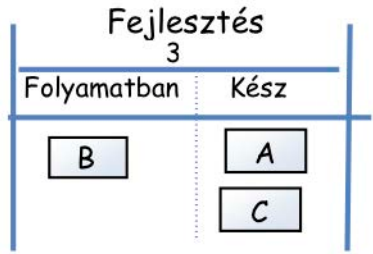
Most az egész workflow ugyanazon a táblán van – nem csak azt látjuk, amit egy Scrum-csapat csinál egy iterációban.

A fenti példában a Backlog oszlop mindössze általános kívánságlista, különösebb sorrend nélkül. A Kiválasztva oszlop tartalmazza a magas prioritású elemeket, kételemű Kanban-korláttal. Tehát bármely adott pillanatban összesen két magas prioritású feladat lehet. Amint a csapat kész egy új feladat megvalósítására, azt a Kiválasztva oszlopból veszi ki. A product owner bármikor változtathat a Backlog és a Kiválasztva oszlopokban, de a többiben nem.

A Fejlesztés oszlop (két továbbira bontva) mutatja, hogy mi áll fejlesztés alatt, hármas Kanban-korláttal. Hálózati terminológiával élve a Kanban-korlát felel meg a „sávszélességnek”, míg az átfutási idő a „pingnek” (válaszidőnek).

Miért osztottuk ketté a Fejlesztés oszlopot egy Folyamatban és egy Kész oszlopra? Azért, hogy a telepítőcsapat láthassa, mely elemek állíthatók élesbe.

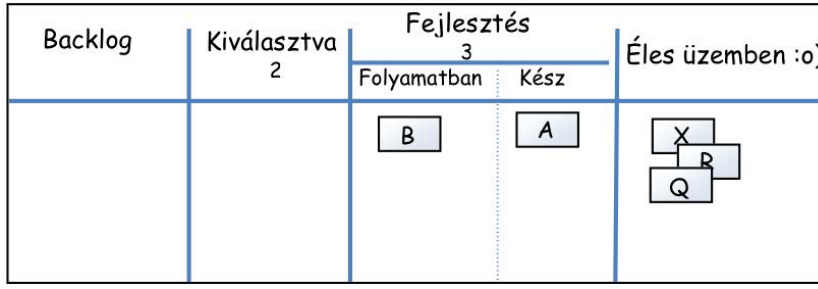
A Fejlesztés hármas korlátján osztozik a két aloszlop. Miért? Tegyük fel, hogy két elem van a Készben:



Ez azt jelenti, hogy összesen egy elem lehet Folyamatban. Így többletkapacitásunk lesz: fejlesztők, akik elkezdhetnének egy új tételen dolgozni, de nem tudnak a Kanban-korlát miatt. Ez erős ösztönző lesz számukra, hogy kapacitásaikat a telepítésre koncentrálják, hogy a Kész oszlopot kiüríthessék és gyorsítsák az áramlást. Ez hasznos és folyamatos hatás, hiszen minél több dolog van a Készben, annál kevesebb megengedett a Folyamatban, ami segíti a csapat figyelmét a megfelelő dolgokra irányítani.

Egydarabos áramlás

Az egydarabos áramlás „tökéletes áramlás”, ahol egy elem halad végig a folyamaton anélkül, hogy bárhol beragadna; azaz minden pillanatban valaki éppen dolgozik rajta. Ez valahogy így nézne ki a táblán:

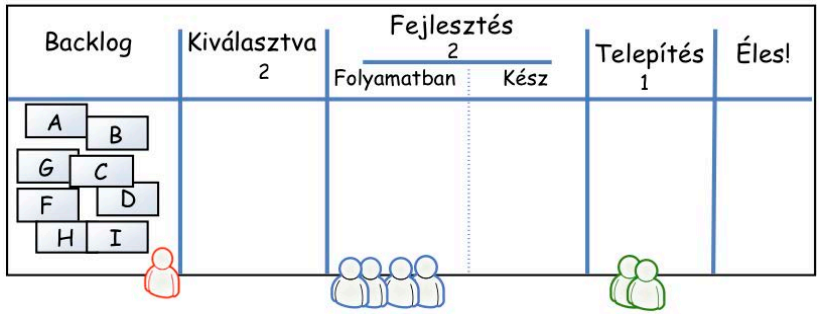


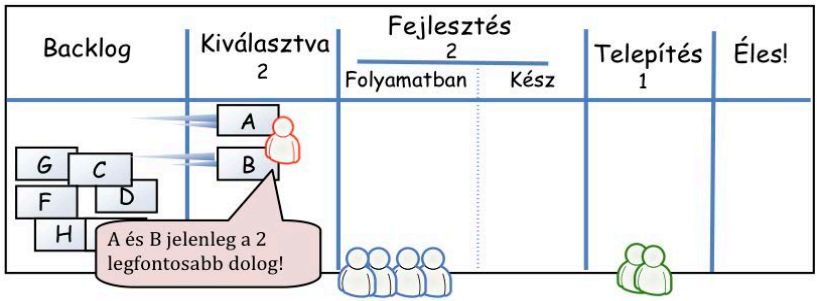
Az adott pillanatban B fejlesztés alatt van, A-t éppen élesbe állítják. Bármikor, amikor a csapat kész egy következő elem feldolgozására, megkérdezik a product ownert, hogy melyik a legfontosabb feladat, és azonnali választ kapnak tőle. Ha ez az ideális állapot hosszú ideig tartható, megszabadulhatunk a Backlog és Kiválasztva oszlopoktól, és igazán rövid átfutási időt kaphatunk.

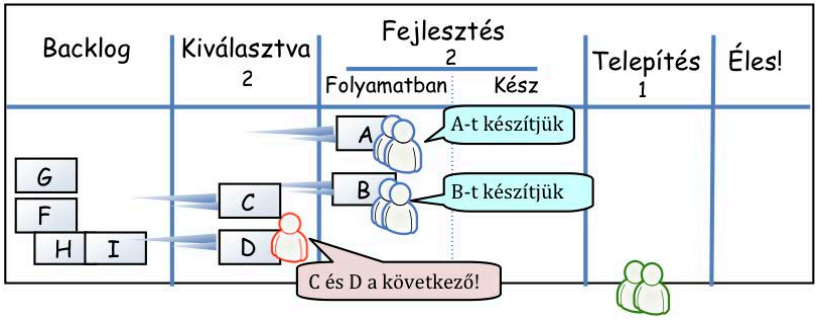
Cory Ladas így fogalmazott: „Az ideális munkatervezési folyamat mindig a legjobb elemet jelöli ki a csapat következő feladatának, nem többet és nem kevesebbet.”

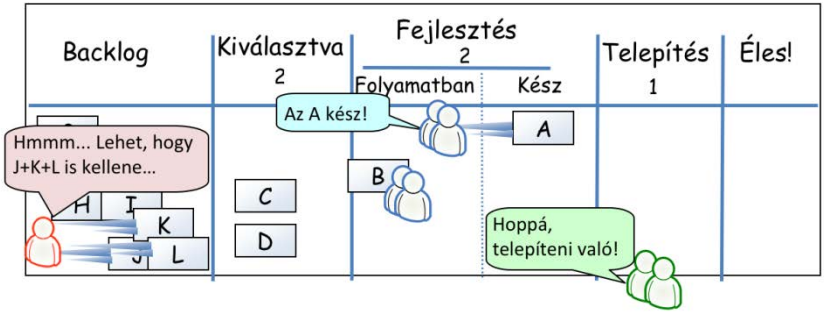
A WIP-korlátok célja megakadályozni, hogy a problémák kicsússzanak a kezeink közül, tehát ha a munkafolyamat zökkenőmentesen zajlik, nincs is igazán szükség rájuk.

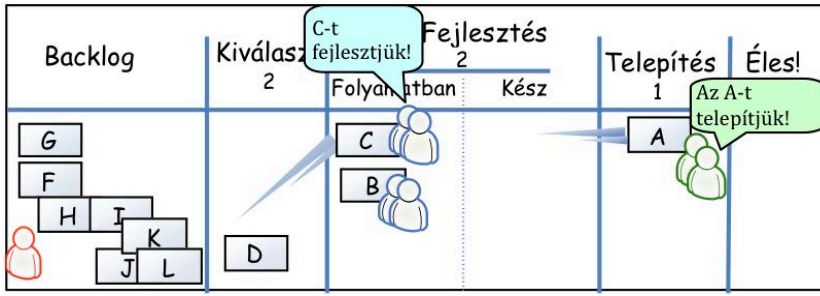
Egy nap a Kanban világában

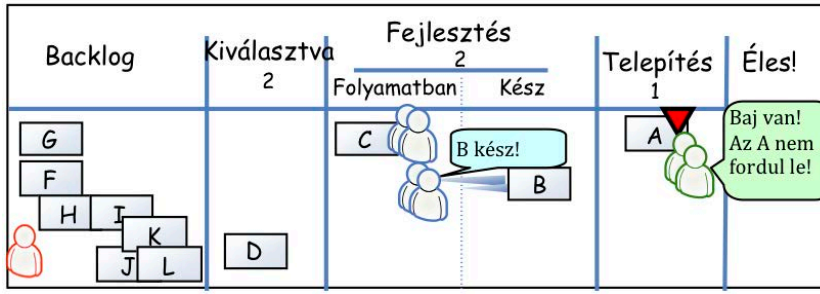


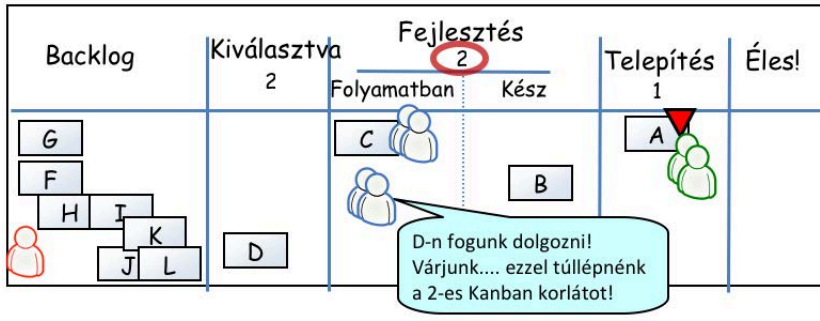


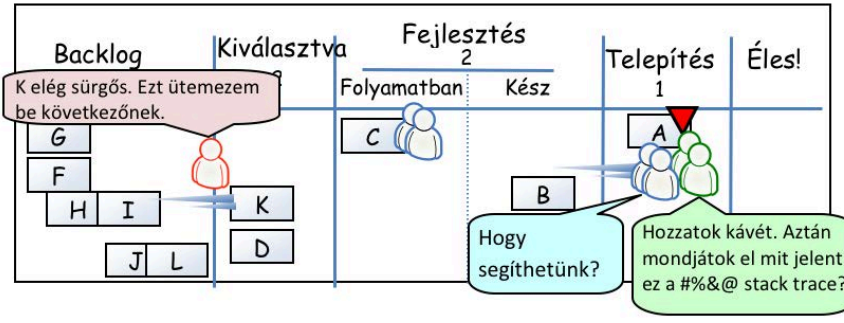


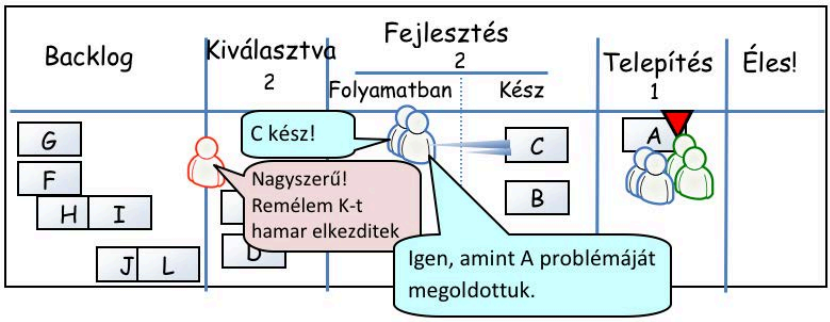


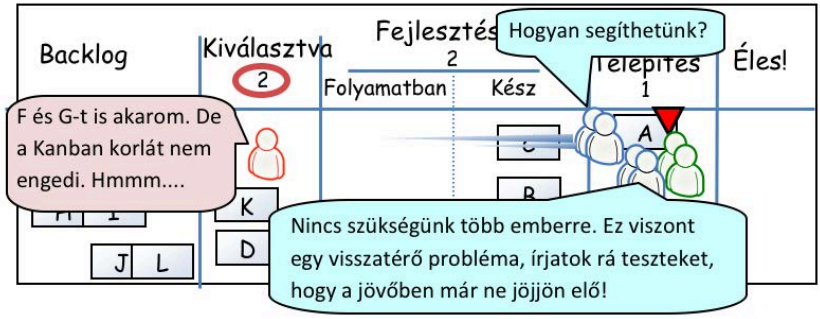

Backlog
Kiválasz
2
C-t fejlesztjük!
Fejlesztés
2
Folyamatban
Kész
Telepítés
1
Éles!
Az A-t telepítjük!
G
F
H
I
K
J
L
D
C
B
A

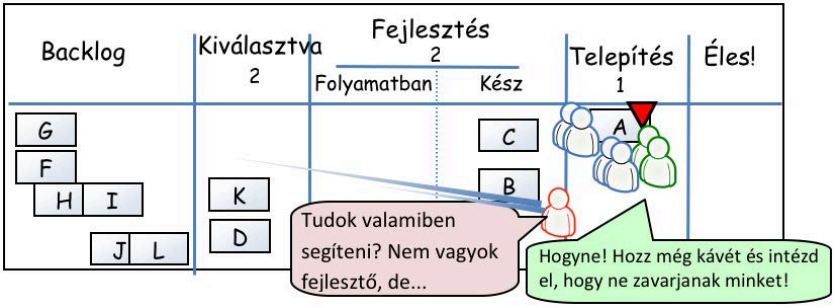

Backlog
Kiválasztva
2
Fejlesztés
2
Folyamatban
Kész
Telepítés
1
Éles!
B kész!
Baj van! Az A nem fordul le!
G
F
H
I
K
J
L
D
C
B
A

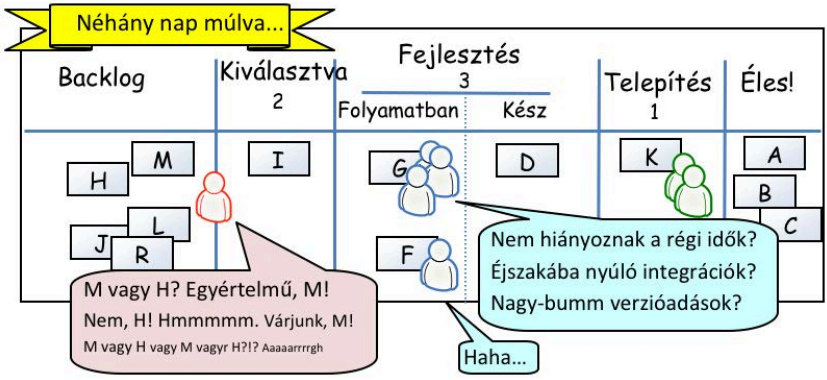

Backlog
Kiválasztva
2
Fejlesztés
2
Folyamatban
Kész
Telepítés
1
Éles!
D-n fogunk dolgozni! Várjunk.... ezzel túllépnénk a 2-es Kanban korlátot!
G
F
H
I
K
J
L
D
C
B
A


Backlog
Kiválasztva
Fejlesztés
2
Folyamatban
Kész
Telepítés
1
Éles!
K elég sürgős. Ezt ütemezem be következőnek.
Hogy segíthetünk?
Hozzatok kávét. Aztán mondjátok el mit jelent ez a #%&@ stack trace?
G
F
H
I
K
J
L
D
C
B
A


Backlog
Kiválasztva
2
Fejlesztés
2
Folyamatban
Kész
Telepítés
1
Éles!
G
F
H
I
J
L
C kész!
Nagyszerű! Remélem K-t hamar elkezditek
C
B
A
Igen, amint A problémáját megoldottuk.


Backlog
Kiválasztva
2
Fejlesztés
2
Folyamatban
Kész
Telepítés
1
Éles!
Hogyan segíthetünk?
F és G-t is akarom. De a Kanban korlát nem engedi. Hmmm....
A
K
D
J
L
Nincs szükségünk több emberre. Ez viszont egy visszatérő probléma, írjatok rá teszteket, hogy a jövőben már ne jöjjön elő!


Backlog
Kiválasztva
2
Fejlesztés
2
Folyamatban
Kész
Telepítés
1
Éles!
G
F
H
I
J
L
K
D
C
B
A
Tudok valamiben segíteni? Nem vagyok fejlesztő, de...
Hogyne! Hozz még kávét és intézd el, hogy ne zavarjanak minket!


Néhány nap múlva...
Backlog
Kiválasztva
2
Fejlesztés
3
Folyamatban
Kész
Telepítés
1
Éles!
H
M
I
J
L
R
G
D
F
K
A
B
C
M vagy H? Egyértelmű, M!
Nem, H! Hmmmmm. Várjunk, M!
M vagy H vagy M vagyr H?!? Aaaaarrrrgh
Nem hiányoznak a régi idők?
Éjszakába nyúló integrációk?
Nagy-bumm verzióadások?
Haha...

Pontosan így kell kinéznie egy Kanban-táblának?

Nem, a fenti tábla csak egy példa volt!

Az egyetlen dolog, amit a Kanban előír, hogy a folyamatot láthatóvá kell tenni, és a feldolgozás alatt lévő elemek számát korlátozni kell. A cél, hogy zökkenőmentes áramlást alakítsunk ki, és minimalizáljuk az átfutási időt; tehát rendszeresen fel kell tennünk a kérdést:

Milyen oszlopaink legyenek?
Minden oszlop egy munkafolyamat-lépést, vagy a két munkafázis között várakozó sort jelent. Kezdjünk kevés oszloppal, és csak akkor adjunk hozzá újat, ha kell.

Mekkora legyen a Kanban-korlát?
Ha elértük a „saját" oszlopunk Kanban-korlátját, és nincs feladat, amit elkezdhetnénk, nézzünk utána a „dugulás" okának (a tábla jobb oldalán sorakozó elemek), és segítsünk elhárítani. Ha nincs „dugulás", lehet, hogy túl alacsony a Kanban-korlátunk, mivel éppen azért vezettük be, hogy csökkentsük a szűk keresztmetszetek „továbbtömésének" kockázatát.

Ha azt tapasztaljuk, hogy sok elem sokáig áll anélkül, hogy valamilyen munkát végeznénk rajta, meglehet, hogy Kanban-korlátunk túl magas.

- Túl alacsony Kanban-korlát => várakozó emberek => gyenge termelékenység

- Túl magas Kanban-korlát => várakozó feladatok => hosszú átfutási idő

Mennyire szigorúak a Kanban-korlátok?
Néhány csapat szigorú szabályként kezeli őket (azaz a korlát nem léphető túl), mások irányelvekként vagy vitaindítókként tekintenek rájuk (azaz a Kanban-limit átléphető, de csak megalapozott és egyeztetett indokkal). Ez tehát rajtunk múlik. Szóltunk előre, hogy a Kanban nem túl előíró, igaz?

16

Scrum vs. Kanban összefoglaló

Hasonlóságok

- Mindkettő Lean és agilis.
- Mindkettő húzórendszeren alapul.
- Mindkettő korlátozza a végrehajtás alatt lévő feladatok számát.
- Mindkettő az átláthatóságot használja a folyamatok fejlesztéséhez.
- Mindkettő a korai és gyakori szoftverkibocsátásra koncentrál.
- Mindkettő önszervező csapatokra alapul.
- Mindkettőnél szükséges a munka részfeladatokra bontása.
- Mindkettő esetében folyamatosan finomítjuk a kibocsátási tervet a tapasztalati adatok alapján (sebesség/átfutási idő).

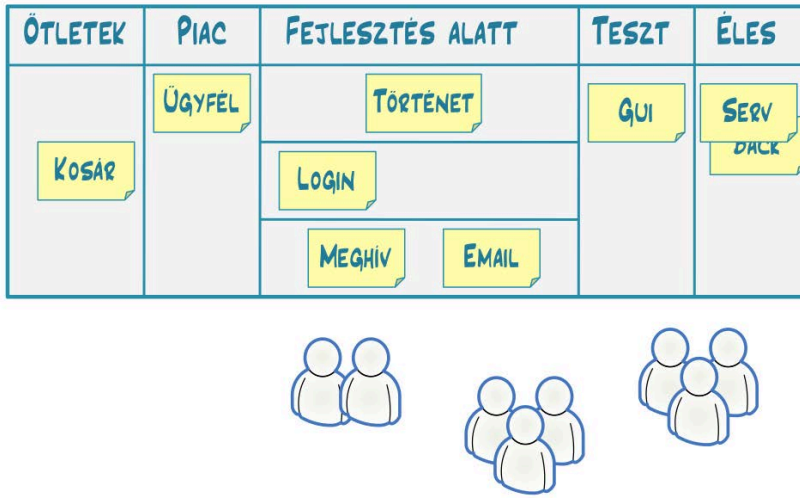
Különbségek

Scrum	Kanban
Időkeretek közé szorított iterációt ír elő.	**Az időkeretek meghatározása opcionális.** A release-tervezés és a folyamat javítás ritmusa elválhat egymástól. Időkeretek helyett eseményvezérelt is lehet.
A csapat elkötelezi magát egy adott mennyiségű feladat elvégzésére az iteráció során.	**A kötelezettségvállalás opcionális.**
A tervezés és folyamatfejlesztés alapértelmezett mérőszáma a **sebesség**.	A tervezés és folyamatfejlesztés alapértelmezett mérőszáma az **átfutási idő**.
Keresztfunkcionális csapatokat ír elő.	A keresztfunkcionális csapatok opcionálisak. **Megengedi a specializált csoportokat.**
A feladatokat olyan méretűre kell bontani, hogy azok egy sprint alatt megvalósíthatóak legyenek.	Nincs előírás a feladatok méretére vonatkozóan.
Előírja a lefutási diagram alkalmazását.	Nincs megkötés bármilyen konkrét diagram alkalmazására.
A **WIP-korlát indirekt módon** (sprintenként) meghatározott.	**A WIP közvetlenül** (folyamat-állapotonként) **korlátozott.**
Előírja a becslést.	**A becslés opcionális.**
Folyamatban lévő iterációhoz nem adható újabb feladat.	**Bármikor hozzáadható újabb feladat, amikor rendelkezésre áll a végrehajtási kapacitás.**
A **sprint backlog egyetlen meghatározott csoporthoz tartozik.**	A **Kanban-tábla megosztható** több team vagy önálló személy között.
Három szerepkört ír elő (PO/SM/Team).	Nem ír elő semmilyen szerepkört.
A **Scrum-táblát** a sprintek között „**újraindítják**”.	**A Kanban-tábla állandó.**
Priorizált product backlog vezetését írja elő.	**A priorizálás opcionális.**

Íme, ennyi. Most már ismerjük a különbségeket.

De ezzel még nincs vége, most jön a legjobb rész! Húzzuk fel a csizmáinkat, itt az ideje, hogy Mattiasszal belevessük magunkat a lövészárkokba, és megnézzük hogyan is működik ez a gyakorlatban!

Kanban a gyakorlatban



Egy történet arról, hogyan tanultunk meg fejlődni a Kanban segítségével. Amikor elkezdtük, alig találtunk információt a Kanbanról és még dr. Google is faképnél hagyott bennünket. Ma a Kanban sikeres, fejlődik, és egyre több tudás áll rendelkezésünkre. Bátran ajánlom David Anderson munkáit, például a „szolgáltatások osztályai" témában. Íme az első (és egyben utolsó) felelősséghárítás. Bármilyen megoldást is alkalmazunk, bizonyosodjunk meg róla, hogy az a konkrét problémák megoldását szolgálja. Ennyi volt. Vágjunk bele, ez a történetünk…

Mattias Skarin

17

Az üzemeltetési munka természete

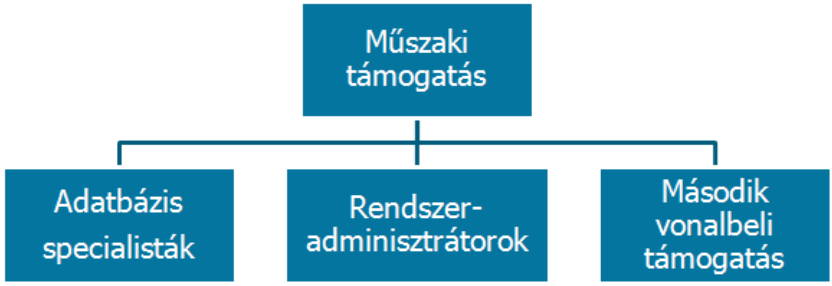
Aki már volt 7/24 órás ügyeletben, annak van fogalma arról, hogy mit jelent egy éles üzemben lévő rendszer üzemeltetésével járó felelősség. A probléma megoldását várják tőlünk – akár az éjszaka közepén is –, függetlenül attól, hogy a probléma forrása nálunk van, vagy nem. Senki nem tudja, hogy miért éppen minket hívnak. Elég nagy kihívás, mivel soha nem gyártottunk hardvert, drivereket, operációs rendszert vagy egyedi szoftvert. A lehetőségeink gyakran arra korlátozódnak, hogy leszűkítsük a probléma lehetséges okait, csökkentsük a következményt, adatokat gyűjtsünk a probléma reprodukálhatóságához, és várjuk, hogy a megfelelő emberek megoldják a problémát, aminek tanúi lettünk.

A kulcs: reakcióképesség és problémamegoldás, gyorsasággal és pontossággal párosítva.

18

Miért változtassunk?

2008-ban egyik ügyfelünk – egy skandináv játékfejlesztő cég – több folyamatfejlesztésen ment keresztül. Az egyik ezek közül a Scrum vállalati szintre történő kiterjesztése volt, amely során egyenként számolták fel a fejlesztési munkát akadályozó tényezőket. Ahogy a munkamódszerek fejlődtek és a szoftverfejlesztés felgyorsult, a nyomás egyre fokozódott az őket kiszolgáló műszaki támogató csapaton. Korábban a műszaki támogatás többnyire csak kívülről figyelte az eseményeket, most viszont egyre inkább a fejlesztési folyamat aktív részeseivé vált.



1. ábra. A műszaki támogatócsoport három csapatból állt: az adatbázis-specialistákból (DBA), a rendszer-adminisztrátorokból és a második vonalbeli támogató csapatból.

A fejlesztőcsapatok működésének javítása többé nem volt elég. Ha továbbra is csak rájuk koncentrálunk, az még több lemaradást okozott volna a kritikus fontosságú műszaki támogatási feladatokat ellátó csapatoknál. Mindkettő fejlesztésére szükség volt.

A fejlesztőcsapatok munkamódszereinek átalakulása az új ötletek értékelésére és elemzésére folyamatosan több energiát igényelt a menedzsmenttől, ennek következtében a vezetőknek egyre kevesebb ideje maradt a feladatok prioritásainak meghatározására és az operatív problémamegoldásra. A vezetőség gyorsan felismerte, hogy cselekednie kell, mielőtt a helyzet kezelhetetlenné válik.

19

Hol kezdjük el?

Jó kiindulási pont volt a fejlesztők – a műszaki terület vevői – megkérdezése.

A fejlesztők véleménye a műszaki támogatásról

Megkérdeztem, hogy mi az első három dolog, ami eszükbe jut a műszaki támogató csapatról? A leggyakoribb válaszok ezek voltak:

“Változó szakértelem”

“Értik a dolgukat, ha infrastruktúráról van szó”

“Segíteni akarnak, de igazi segítséget kapni nehéz tőlük”

“A projektek túl sokáig tartanak”

“A hibajegy rendszerük szívás”

“Mivel foglalkoznak azok a srácok?”

“Sok email-be kerül, mire elvégeznek egy egyszerű dolgot”

“Nehéz elérni őket”

Röviden ez volt a fejlesztők véleménye a műszaki támogató területről. Most hasonlítsuk össze ezt a műszakiak véleményével a fejlesztőkről…

A műszakiak véleménye a fejlesztőkről

„Miért nem használják ki a platform lehetőségeit?”

„Ne legyen a release-adás ekkora ügy!”

„Mi tartjuk a hátunkat a gyenge minőségű munkátok miatt!”

„Változniuk kellene” – a két oldal véleményében ez volt a közös. Nyilvánvalónak tűnt, hogy ezen a hozzáálláson kell változtatni, ha ki akarunk jutni a kátyúból. A bíztató jelekből – „értik a dolgukat, ha infrastruktúráról van szó” (annak a jele, hogy bíznak a szakmai hozzáértésükben) – arra lehetett következtetni, hogy az „ők és mi” mentalitás megszüntethető, ha sikerül megfelelő munkakörnyezetet kialakítani. A túlórák csökkentése és a minőség előtérbe helyezése jó kiindulási pontnak tűnt.

20

Az indulás

Tehát valahogy el kell indulnunk. De hol kezdjük? Az egyetlen dolog, amit tudtunk, hogy nem ott fogunk végezni, ahol indultunk.

Fejlesztői múlttal rendelkeztem, tehát meglehetősen keveset tudtam az üzemeltetésről. Nem az volt a célom, hogy „berobbanjak, és mindent megváltoztassak". Sokkal kevésbé konfrontatív megközelítésre volt szükségem, ami rávilágít a fontos dolgokra, elkerüli a lényegteleneket, és könnyű alkalmazásba venni.

A következő jelöltjeim voltak:

1. Scrum – ez már bevált a fejlesztő csapatoknál.

2. Kanban – új és kipróbálatlan, de jól passzol a Lean-alapelvekhez, amiknek híján voltunk.

Egyeztetve a vezetőkkel, a Kanban és Lean-alapelvek látszódtak megoldást kínálni a megcélzott problémákra. Meglátásuk szerint a sprintekbe szervezés nem tűnt jó alternatívának, mivel a feladatok prioritásainak meghatározása napi szinten történt. Tehát a Kanban látszódott jó kiindulási pontnak, annak ellenére, hogy mindannyiunknak új állatfajta volt.

21

A csapatok elindítása

Hogyan indítsuk el a csapatokat? Nem volt fellelhető semmilyen kézikönyv arról, hogyan induljunk, rosszul csinálni nagyon kockázatos lett volna. Az, hogy esetleg nem sikerül fejlődést elérni, csak egy dolog. Nagyon speciális szaktudású embereket igénylő területen dolgoztunk, akiket nehéz lett volna pótolni, elidegeníteni pedig nagyon rossz ötletnek tűnt.

- Vágjunk egyszerűen bele? Lesz, ami lesz?
- Vagy kezdjünk egy workshoppal?

Számunkra egyértelmű volt: kezdjünk egy workshoppal, igaz? De hogyan? Komoly kihívás volt a teljes üzemeltető csapatot egyszerre beterelni egy workshopra (ki veszi fel a telefont, ha valami történik?). Végül úgy döntöttünk, hogy egyszerű, feladatorientált, félnapos workshopot szervezünk.

A workshop

A workshop egyik előnye, hogy gyorsan felszínre hozza a problémákat, emellett bizalmas közeget teremt, ahol a különböző szempontokat a team tagjai gyorsan és közvetlenül átbeszélhetik. Az igazság az – nézzünk szembe vele –, nem volt mindenki túl lelkes a munkamódszerek változásáért, de a csapat legtöbb tagja nyitott volt a próbálkozásra. Lebonyolítottunk egy workshopot, ahol bemutattuk a legfontosabb alapelveket, és eljátszottunk egy leegyszerűsített Kanban-gyakorlatot.

A workshop végén ötujjas szavazást rendeztünk, hogy lássuk a csapat hajlandóságát mindezek élesben történő kipróbálására. Nem volt ellenvetés, tehát belevágtunk. *(Az ötujjas szavazás [angolul „Fist of Five”] egy konszenzusmérő módszer, melynek során minden résztvevő a felmutatott ujjaival fejezi ki egyetértésének mértékét. Egy ujj felmutatása erős ellenvetést jelent, öt erős egyetértést. A három ujj egyetértést jelent, de fenntartásokkal. A konszenzust akkor értük el, ha mindenki legalább három ujját felmutatta.)*

22

Az érdekeltek bevonása

Megjósolható volt, hogy a csapat környezetét is érinteni fogja a Kanban bevezetése. Ezek a változások az ő szempontjukból pozitív irányúak – többek között elkezdenek nemet mondani azokra a feladatokra, amiket nem tudnak elvégezni, kiállnak a minőségért, és elkezdik kitakarítani a backlogból a kevésbé fontos feladatokat. Mindezek ellenére a változások előtti egyeztetés mindig jó ötlet.

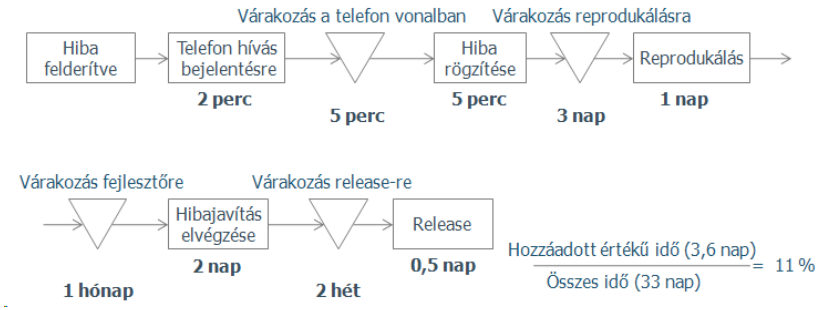
A legközelebbi érintettek az első vonalbeli támogatás és a csoportvezetők voltak. Mivel részt vettek a workshopon, már támogatták a változást; ugyanez állt a fejlesztői csoportokra (akik többé-kevésbé egyébként is fejlődést vártak). A supportcsapat ugyanakkor más volt, hiszen az ő legnagyobb problémájuk a túlterheltség volt, ők kezelték a felhasználói bejelentéseket, és a cég elkötelezte magát amellett, hogy minden felhasználói bejelentésre reagál. A Kanban bevezetésével és a WIP-korlátok betartatásával ez a helyzet jó eséllyel változás előtt állt.

Körutat tettünk a legfontosabb érdekelteknél, és bemutattuk az elképzeléseinket, az elvárt eredményeket és a lehetséges következményeket. Megkönnyebbülésemre elképzeléseink alapvetően kedvező fogadtatásra találtak, néha a „nagyszerű lesz, ha végre megoldjuk ezeket az ügyeket!” megjegyzéssel.

23

Az első tábla megalkotása

A Kanban-tábla elkészítésének jó kiindulópontja az érték–folyamat–térkép felrajzolása. Ez alapvetően az értékfolyamat megjelenítése, ami betekintést nyújt a munkafázisokba, a folyamatba, és az ezekre fordított időbe (ciklusidő).



De mi ennél lényegesen egyszerűbben kezdtük: ami nem más, mint egy, a vezetővel közösen papírra rajzolt minta Kanban-táblával. Néhányszor felülvizsgáltuk, majd belevágtunk. Ebben a fázisban többek között a következő kérdések merültek fel:

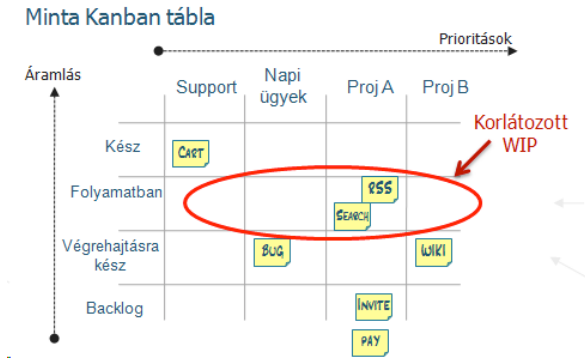
- Milyen típusú feladataink vannak?
- Ki végzi el őket?
- Megosszuk a felelősséget a különböző feladattípusok között?
- Hogyan kezeljük a megosztott felelősséget az adott speciális szakterületek között?

Mivel a különböző munkatípusokhoz különböző szolgáltatási szintmegállapodások (SLA) tartoztak, természetesnek tűnt, hogy minden csoport maga alakítsa ki a saját tábláját. Maguk határozták meg a sorokat és az oszlopokat.

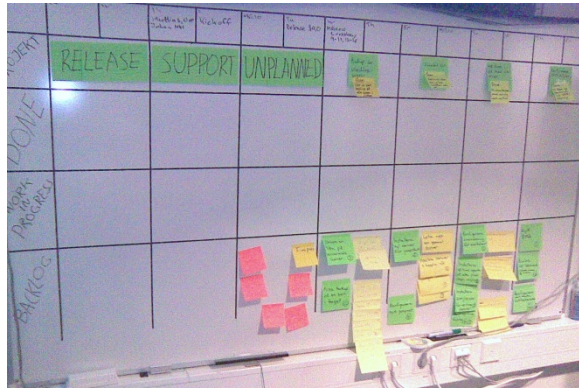
A következő nagy döntés az volt, hogy megosztott felelősséget alkalmazzunk-e a különböző feladattípusok között. „Hagyjuk-e, hogy a csapat egy állandó része csak a supportfeladatokkal foglalkozzon (reaktív munka), míg a másik része a projektfeladatokra koncentráljon (proaktív munka)?” Elsőre úgy döntöttünk, hogy próbáljuk ki a megosztott felelősséget. A legfőbb érv emellett az volt, hogy a csapatok önszerveződése és a csapattagok egyéni fejlődése kulcsfontosságú a folyamatos alakulás szempontjából. A döntés hátránya, hogy a supportfeladatok potenciálisan bárki időbeosztását megzavarhatják, de elinduláskor még ezzel együtt is ez tűnt a legjobb megoldásnak. Egy megjegyzés: amikor a workshopot tartottuk, a csapat tulajdonképpen maga alakította ki ezt a megoldást. Egy embert az azonnali feladatok megoldásával bíztak meg, míg a többiek a nagyobb ügyekkel foglalkoztak.

Az első Kanban-modell

Lent látható az egyszerű Kanban-modell. Megjegyezzük, hogy a team döntötte el, hogy a feladatok áramlása fölfelé haladjon (mint a buborékok a vízben) – szemben a tipikus, balról jobbra iránnyal.



2. ábra. Ez az első Kanban-modell. A prioritások balról jobbra növekednek, a folyamat felfelé halad. A feldolgozás alatt lévő feladatok számítása a Folyamatban sorban lévő feladatok számának összege alapján történik (pirossal bekarikázva). A modell kialakítását a Linda Cook által leírt tapasztalatok befolyásolták.



3. ábra. A rendszeradminisztrációs csapat első Kanban-táblája.

Az alkalmazott sorok

Workflow állapot (sor)	Meghatározás
Backlog	Azok a sztorik, amelyeket a vezető szükségesnek tart.
Végrehajtásra kész	Felbecsült és maximum nyolc órás feladatokra bontott sztorik.
Folyamatban	A WIP-korláttal rendelkező sor. A kezdeti érték a 2×[team létszám]–1 volt. (A –1 az együttműködés miatt.) Egy négyfős csapat WIP–korlátja tehát hét.
Kész	A felhasználó által futtatható.

Alkalmazott oszlopok

Feladattípus	Meghatározás
Release	Fejlesztő csapat támogatása a release kiadásában.
Support	Más csapatok kisebb kérései.
Nem tervezett	Váratlan feladatok, melyeket el kellett végezni, de nem volt egyértelmű gazdája, pl. kisebb infrastruktúrajavítások.
Projekt A	Nagyobb infrastruktúraprojekt, pl. egy staging környezet teljes hardver-költöztetése.
Projekt B	Egy másik nagyobb projekt.

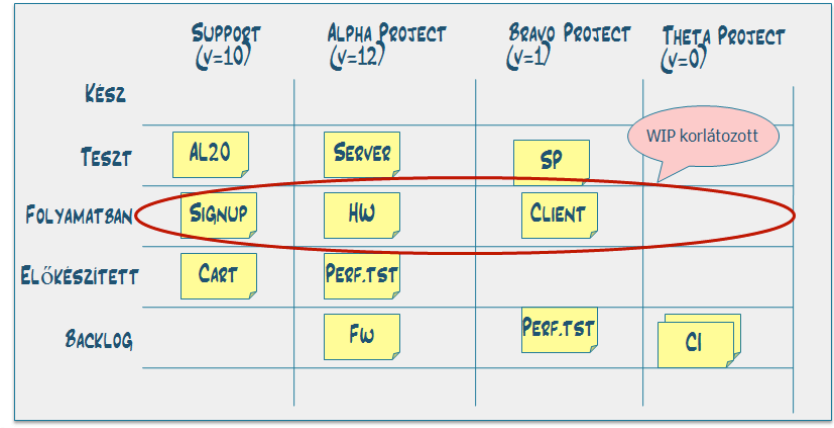
Nem minden Kanban-tábla volt ugyanolyan; mindegyik egyszerű vázlattal kezdődött, majd az idők során fejlődött.

24

Az első WIP-korlát beállítása

Az első feldolgozás alatt lévő feladatszám- (WIP-) korlátunk meglehetősen „nagyvonalú” volt. Ezt azzal indokoltuk, hogy előbb tegyük láthatóvá a folyamatot és nézzük meg, hogy mi történik. Valószínűtlennek tartottuk, hogy elsőre eltaláljuk a legjobb értéket. Ahogy az idő múlik, majd minden alkalommal finomítjuk a WIP-korlátot, ha jó okot látunk erre.

Az első WIP–korlát, amit használtunk ez volt: 2n–1 (n = a team tagjainak száma, –1 azért, hogy elősegítsük a kommunikációt). Miért? Egyszerűen azért, mert nem volt jobb ötletünk, és mert védhetőnek tűnt ezzel kezdeni. A képlet egyszerű és logikus érvvel szolgált bárki számára, aki rá akart erőltetni egy feladatot a teamre: „… tehát ha minden teamtag egyszerre két feladattal tud foglalkozni – egy aktívval és egy passzívval –, miért várnád el tőle, hogy belekezdjen még egybe?” Visszanézve, kezdetnek bármilyen bő WIP-limit megteszi, hiszen a Kanban-tábla folyamatos ellenőrzésével könnyű menet közben megtalálni a megfelelő korlátot.



4. ábra. Így alkalmaztuk a WIP-korlátot az adatbázis rendszeradminisztrációs csoportban: munkatípusonként egy feladat.

Egyik tapasztalatunk, hogy értelmetlen a WIP-korlátot sztoripontokban meghatározni, mivel túl bonyolult a követése. Az egyetlen könnyen

kezelhető módszer egyszerűen a feladatok számának követése (= párhuzamos feladatok).

A supportcsapat esetében a WIP-et oszlopokra határoztuk meg, mert gyorsabb reakcióra volt szükség, ha a korlát kezdett megtelni.

25

A WIP-korlát tisztelete

A WIP-korlát betartása elméletben egyszerűnek hangzik, a gyakorlatban ugyanakkor komoly elkötelezettséget igényel. Ez azt jelenti, hogy adott esetben ki kell tudni mondani, hogy „nem”! Mi több megközelítést is kipróbáltunk.

Megvitatni a táblánál

Ha a WIP-korlát túllépését tapasztaltuk, odahívtuk az érintetteket a táblához, és megkérdeztük mit akartak elérni. Kezdetben a korlát túllépésének leggyakoribb oka egyszerűen a tapasztalatlanság volt. Néha a prioritások különböző szempontok szerinti értelmezésével találkoztunk, aminek tipikus esete, hogy egy specialista team tagja egy speciális feladaton dolgozott. Egyedül ilyenkor jelentkeztek súrlódások, de az esetek többségében ott és akkor, a tábla előtt el tudtuk simítani a problémákat.

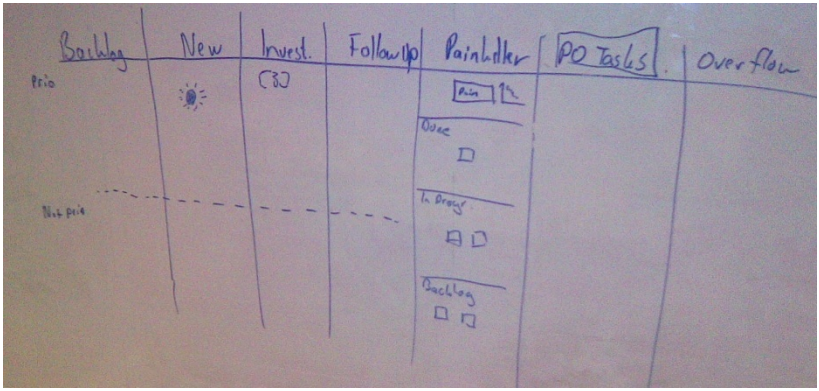
Túlfolyó terület kijelölése

Ha nemet mondani túlzottan konfrontatív, és valamelyik feladatról lemondani nehéz volt, az alacsony prioritású teendőket egy „túlfolyó” területre helyeztük, amennyiben túlléptük a WIP-korlátot. A „túlfolyó” területre két szabályt alkalmaztunk:

1. Nincsenek elfelejtve, amint kapacitásunk szabadul fel, foglalkozunk velük.

2. Ha kidobjuk, értesítünk róla.

Mindössze két hét alatt bebizonyosodott, hogy a túlcsordult elemekkel soha nem fogunk foglalkozni, tehát a csapat vezetőjének támogatásával végül megszabadultunk tőlük.

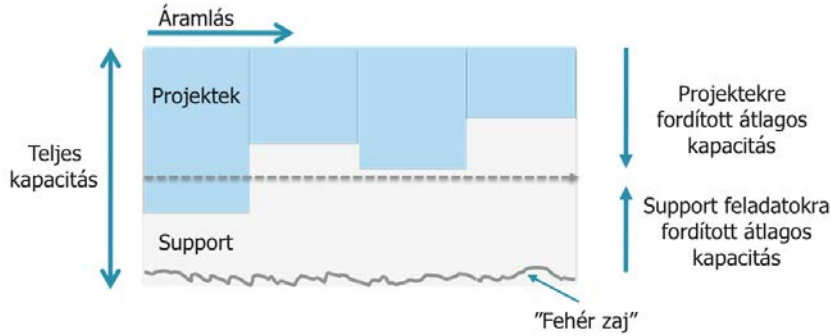


5. ábra. Vázlat a támogatócsapat Kanban-táblájáról. A jobb szélső a túlfolyó szakasz.

26

Melyik feladatok kerüljenek a táblára?

Hamar arra az elhatározásra jutottunk, hogy nem érdemes *minden* teendőt a táblára aggatni. A Kanbant az olyan feladatok követése, mint pl. egy telefonhívás vagy kávézás, adminisztrációs szörnyeteggé változtatja. Mi azért voltunk itt, mert problémákat kellett *megoldani*, nem pedig újabbakat generálni. Úgy döntöttünk, csak azok kerülnek fel a táblára, amelyek egy óránál tovább tartanak, minden ennél kisebb feladatot „fehér zajnak" tekintettünk. Ez az egyórás limit aztán végül elég jól bejött, és egyike lett azon kevés dolognak, amelyek a későbbiekben változatlanul maradtak.



6. ábra. Kezdetben azt feltételeztük, hogy a teljes kapacitásunkat főleg két típusú munkavégzés köti le: a nagyobbak (projektmunkák), illetve a kisebbek (supportjellegű feladatok). A projektekre vonatkoztatott sebesség követése szükség esetén jelezte számunkra a várható szállítási dátumot. A „fehér zaj" jelenlétét állandó jelenségnek tekintettük (egy óránál rövidebb supportjellegű tevékenységek, meetingek, kávé, kollégák segítése).

27

Hogyan becsüljünk?

Ez állandó téma, és természetesen több válasz is felmerül ennek kapcsán:

- Becsüljünk rendszeresen!
- Csak akkor becsüljünk, ha szükség van rá!
- Számoljunk ideális napokban/sztoripontokban!
- A becslések bizonytalanok, ezért becsüljünk inkább pólóméretekkel (kicsi, közepes, nagy)!
- Egyáltalán ne becsüljünk, vagy csak akkor, ha a csúszás költsége indokolja!

Scrumos befolyásoltságunk eredményeképpen úgy döntöttünk (mégiscsak ezt ismertük), hogy sztoripontokkal kezdjük a becsléseket. A gyakorlatban azonban a csapatok a sztoripontokat ekvivalensnek tekintették az emberórákkal (sokkal természetesebbnek érezték). Kezdetben minden sztorit megbecsültünk, idővel a menedzserek megtanulták, hogyha a párhuzamos projektek számát alacsonyan tartják, kevésbe várakoztatják meg az ügyfelet. Arra is rájöttek, hogyha nem várt változás következik be, bármikor újrapriorizálhatnak, és így kezelhetik a problémát.

A szállítási határidő szükségessége már nem volt igazán téma többé. A menedzserek leszoktak arról, hogy állandóan a becslésekről kérdezzenek, csak akkor tették ezt, ha attól féltek, hogy megvárakoztatják az ügyfelet.

Valamikor az elején az egyik menedzser, bestresszelve egy telefonhívástól, megígérte, hogy a projektet a hét végéig leszállítjuk. Mivel a projekt fenn volt a Kanban-táblán, az elvégzett sztorik alapján könnyen meg lehetett becsülni a haladás folyamatát, és így arra a következtetésre jutottunk, hogy egy hét alatt huszonöt százalékot teljesítünk, így további három hétre van szükség a befejezéshez. Ezzel a ténnyel szembesülve a menedzser azonnal megváltoztatta a projekt prioritásait, leállította a párhuzamos munkákat, így lehetővé tette az ígért szállítást. Mindig figyeljük a táblát!

Mit jelent a becsült méret? Átfutási időt vagy munkaórát?

Sztoripontjaink a munkaórákat tükrözték, azaz azt, hogy hány megszakítás nélküli óraráfordítást várunk egy sztori befejezéséhez, és nem pedig az átfutási időt (naptári időintervallumot vagy eltelt órát). Azzal, hogy mértük, hány sztoripontot teljesítünk egy hét alatt (sebesség), következtetni tudtunk az átfutási időre.

Mindenegyes új sztorit csak egyszer becsültünk, és a megvalósítás során soha nem korrigáltuk a becslésünket, ezzel csökkentettük a csapat által becslésekre fordított időt.

28

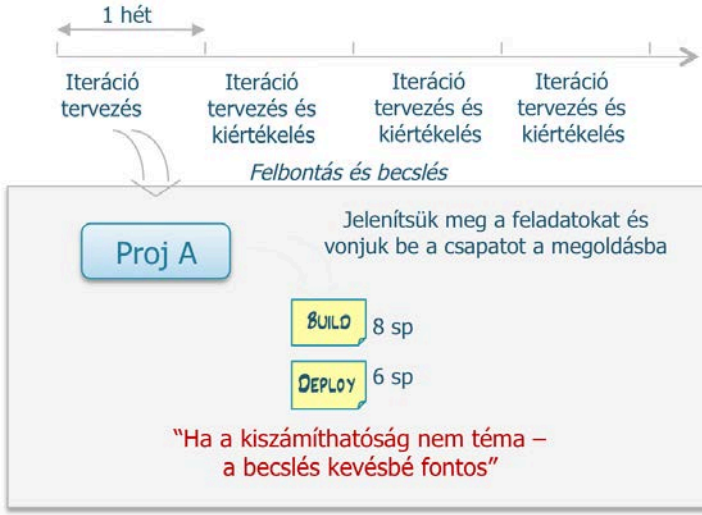
Hogyan dolgoztunk a valóságban?

A Kanban tényleg nagy szabadságot biztosít, ezért bármilyen módon alkalmazhatjuk. A csapat dolgozhat idő szerinti ütemezésben vagy eseményvezérelten, például arra reagálva, ha adott mennyiségű feladat összegyűlt.

7. ábra. Ha három elem összegyűlt a backlogban, tervezés-/becslésmegbeszélést tartunk.

Úgy döntöttünk, két ismétlődő eseményt tartunk rendszeresen:

- Napi standup – a csapattal a tábla előtt azért, hogy felszínre hozzuk a problémákat és átláthatóvá tegyük egymás feladatait.
- Heti iterációtervezés – a tervezés és folyamatos javítás fenntartása érdekében.



Ez nálunk bevált.

Napi standup

A napi standup a Scrumhoz hasonlóan zajlott. Ezt a napi „Scrum of Scrums" megbeszélés után tartottuk, ahol minden team képviseltette magát (fejlesztés, tesztelés, üzemeltetés). A „Scrum of Scrums" egyeztetések értékes információval szolgáltak a Kanban-csapatok számára arról, hogy mely problémákat kell elsőnek megoldani, melyik team van a legnagyobb bajban. Kezdetben a menedzsment rendszeresen látogatta ezeket a megbeszéléseket, megoldási javaslatokat tettek és priorizációs döntéseket hoztak. Idővel, ahogy a csapatok egyre tapasztaltabbak és önállóbbak lettek, ritkábban jelentek meg (de szükség esetére mindig a közelben voltak).

Iterációtervezés

Hetente egyszer iterációtervezést tartottunk, minden héten ugyanabban az időpontban, mert ha nem volt előre betervezve, valami fontosabb mindig elvitte az időt, nekünk viszont több kommunikációra volt szükségünk. A tipikus menetrend a következő volt:

- Grafikonok és a tábla aktualizálása. (A befejezett projektek a Készek falára kerültek.)
- Visszatekintés az előző hétre: Mi történt? Miért így történt? Mit tehetünk, hogy javítsunk ezen?

- A WIP-korlátok hangolása (ha szükséges).
- Feladatlebontás és új projektek becslése (ha szükség volt rá).

Az iterációtervezés tulajdonképpen becslési és folyamatfejlesztési impulzusok keveréke volt. A kicsi és közepes ügyeket az első vonalbeli vezetők támogatásával helyben orvosoltuk, a komplexebb infrastruktúraproblémák megoldásának követése már lényegesen nagyobb megpróbáltatást jelentett. Éppen ezért bevezettük, hogy minden csapat maximum kettő „akadályozó tényező” megoldását oszthatta ki a vezetője számára.

A szabályok ezek voltak:

1. A vezetők egyszerre két problémán tudnak dolgozni.
2. Ha mindkét hely betelt, új feladatot akkor lehet kihelyezni, ha a kevésbé fontosat levettük.
3. A csapat dönti el, mikor tekinti megoldottnak a problémát.

Ennek nagyon pozitív hatása volt, mivel a csapatok láthatták, hogy a vezetők mellettük állnak, még a kemény problémák megoldásában is. Rámutathattak a problémára, és megkérdezhették „hogy halad?”. Nem merültek feledésbe, és nem írták őket felül az új, fontosabb ügyek.

Az egyik ilyen súlyos akadály például az volt, hogy az üzemeltetés nem kapott kellő támogatást a fejlesztőktől, ha egy fejlesztési hibára gyanakodtak. Szükségük lett volna segítségére, hogy kideríthessék a rendszer melyik részében van a hiba, de mivel a fejlesztők el voltak foglalva a sprint feladataival, a problémák csak halmozódtak. Nem meglepő módon az üzemeltetés úgy érezte, hogy fejlesztők nem törődnek a minőséggel.

Amikor ez a probléma felszínre került, első körben az alsó vezetés felé, majd a részlegvezető szintjére eszkalálták, aki megbeszélést szervezett a fejlesztési részleg vezetőjével. Ekkor állapodtak meg abban, hogy a minőségi kérdések elsőbbséget élveznek, és kitaláltak egy vetésforgó rendszert: minden sprint alatt az egyik team ügyeletet ad, és azonnal elérhető lesz az üzemeltetők számára. Miután főnökei támogatását

megszerezte, a fejlesztési részleg vezetője egy ügyeleti listát adott át az üzemeltetés vezetőjének. Bár nem voltak biztosak benne, hogy a megoldás működni fog, azonnal cselekedtek, és az elképzelést késlekedés nélkül gyakorlatba ültették és tesztelni kezdték. Ez önmagában megkönnyebbülést hozott az üzemeltetés számára.

29

Egy működő tervezési módszer keresése

Egy történet

Emlékszem az egyik csapat áttörésére; a második tervezési megbeszélésükön ültem. A csapat elakadt egy projekttel, amelyről nem tudta, hogyan becsülje meg a méretét, mivel túl sok ismeretlen volt benne, és a megbeszélés befulladt. Ahelyett, hogy részt vállaltam volna a tervezésben, arra kértem őket, hogy gondolják újra a tervezési folyamatot, és találjanak egy jobb módszert. Vezetőjük irányításával szembenéztek a kihívással, és nekiláttak saját megoldásuk kialakításának. Ez az esemény fontos fordulópont volt, „győzelem", amely során magabiztos csapattá értek. Ez után olyan sebességgel kezdtek fejlődni, hogy el kellett állnunk az útjukból.

Két hónappal később vezetőjük egy visszatekintő megbeszélés után odajött hozzám, és a csapat Kanban-táblájára mutatva ezt mondta: „Van egy problémám: nincsenek igazi problémáink. Mit tegyünk?"

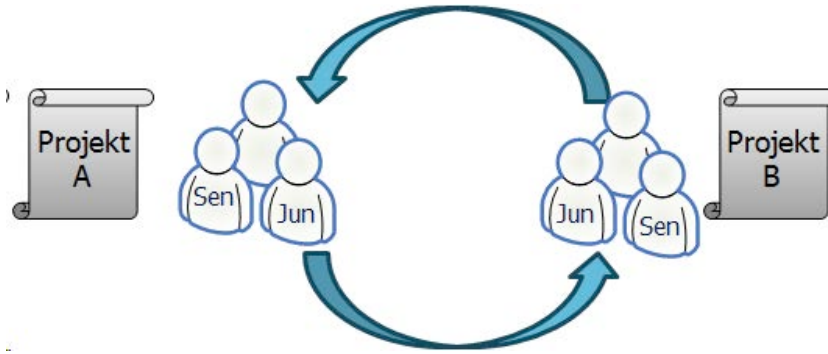
A tervezés újrafelfedezése

A team minden tagjának részvételével zajló tervező póker (planning poker) becslési módszer nem működött jól egyik üzemeltetői csapatnál sem. Többek között ezért:

1. A team tagjainak tudása nem volt egyenletes.
2. Legtöbbször egyetlen ember beszélt.
3. A team tagjai saját égető ügyeiket akarták előtérbe helyezni.

A kísérletezés során viszont a csapatok egymástól függetlenül két különböző becslési mechanizmussal álltak elő; az adott csapatnál mindkettő jól működött.

Első módszer – csere és felülvizsgálat



- Minden projektet/sztorit egy junior és egy senior csapattagból álló párhoz rendelünk becslésre (azaz egy emberhez, aki részletesen ismeri a problémát és egyhez, aki nem). Ez segíti a tudás elterjedését.

- A csapat többi tagja eldöntheti, hogy mely projektek becslésében akar részt venni (a hatékonyság megtartása érdekében minden becslésben maximum négy ember vehet részt).

- Minden becslést végző team feladatokra bontja a projektjét – ha szükséges –, és elvégzi ezek becslését.

- Ezután a teamek felcserélik a projekteket, és szemlézik egymás munkáját (minden teamből egy ember a feladattal marad, hogy megossza a tervezési munka mikéntjét a szemlézőkkel).

- Kész!

Jellemzően az egész iterációtervezés nagyjából negyvenöt percig tartott, a csapat energiaszintje a megbeszélés alatt végig magas maradt. A felülvizsgálat során általában egy-két finomítást hajtottak végre.

Második módszer – senior elemzés, majd becslés

Két tapasztalt csapattag elvégezi a projekt magasszintű elemzését a tervezés előtt. Értékelik a lehetséges megoldásokat, és kiválasztanak egyet. Ha ez megtörtént, a csapat a kiválasztott megoldás mentén feladatokra bontja a projektet.

8. ábra. Feladat lebontás másik csapat általi felülvizsgálattal.

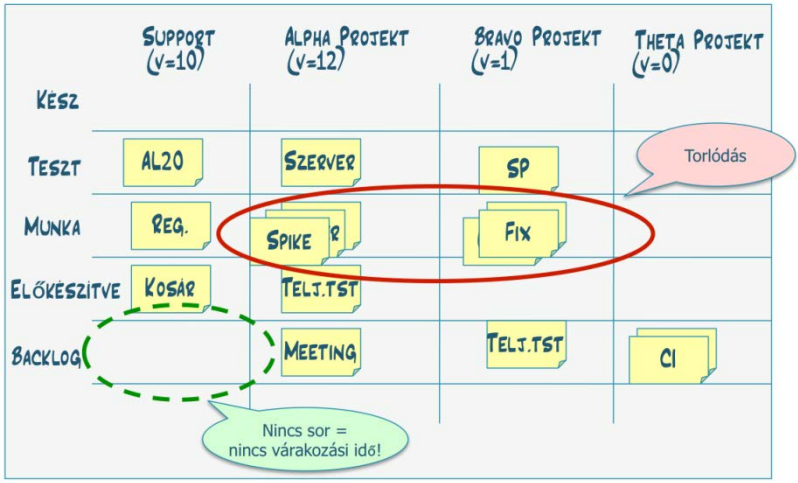
30

Mit mérjünk?

Sok mindent mérhetnénk: a ciklusidőt (az igény felbukkanása és megvalósítása között eltelt időt), a sebességet, a várakozási sorok méretét vagy akár a lefutás sebességét… A legfontosabb kérdés azonban az, hogy melyik metrika használható folyamatunk fejlesztéséhez? A tanácsom az, hogy kísérletezzünk, és figyeljük meg, hogy melyik segít leginkább számunkra. Korábbi tapasztalataink azt mutatják, hogy a burndown chartok minden, egy–négy hétnél rövidebb projektet megfojtanak. A haladás továbbra is figyelemmel kísérhető egy Kanban-táblán. (Hány sztori volt a backlogban, és hány lett kész.)

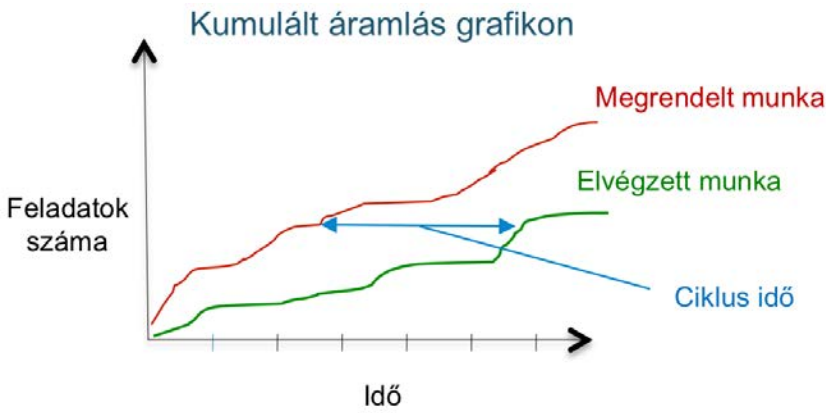
Lehetséges metrika	**Pro**	**Kontra**
Ciklusidő	Könnyű mérni. Nem kell becsülni. Megrendelőnél kezdődik, és nála végződik.	Nem veszi figyelembe a feladatok méretét.
Összesített sebesség (figyelembe véve az összes munkavégzést)	A folyamat lehetséges továbbfejlesztésének és változtatásának nagyvonalú és egyszerű indikátora.	Nem segít előrejelezni a szállítási időpontokat az egyes specifikus munkatípusokra.
Munkatípuson kénti sebesség	Pontosabb, mint az összesített sebesség.	Ahhoz, hogy használható legyen, a megrendelői igénytől a szállításig kell figyelnünk a folyamatot. Hosszabb időt visz el, mint az összesített sebesség meghatározása.
Várakozási sorok mérete	Gyors indikátor az igények fluktuációjához. Könnyen vizualizálható.	Nem jelzi, hogy egyenetlen igény vagy egyenetlen kapacitás a kiváltó ok. A nullaméretű sor valójában túlzott kapacitást jelez.

Mi először a „munkatípusonkénti sebességgel” és a „várakozási sorok méretének” mérésével kezdtünk. A munkatípusonkénti sebességet könnyű mérni, és kezdetben jól használható; a várakozási sorok mérete pedig jó irányadó indikátor és szembeötlő is (amennyiben tudjuk, hogy mit kell figyelni).



9. ábra. Szűk keresztmetszetek és lehetőségek. A pirossal jelölt terület azt mutatja, hogy a torlódó várakozási sorok hogyan alakítottak ki szűk keresztmetszetet a tesztelésben. A supportoszlopban található backlog mezőhöz nem tartozik várakozási sor, így nincs várakozási idő az újonnan bekerülő supportfeladatoknál. Ez azt jelzi, hogy magas a szolgáltatás színvonala.

Mi nem alkalmaztuk a kumulált áramlás grafikont, bár érdekes lett volna.



Azért nem készítettük el a diagramot, mert a Kanban-tábla és a sebességgrafikon elegendő információt szolgáltatott számunkra, legalábbis a kezdeti időszakban. A szűk keresztmetszeteket, az egyenetlenséget és a túlterhelést mind könnyen azonosíthattuk, ezen problémák orvoslása elegendő feladatot adott nekünk az elkövetkezendő hat hónapban.

31

Hogyan indult be a változás

Három hónappal a Kanban bevezetése után a rendszeradminisztráció csoport kapta az IT terület „legjobban teljesítő csapata” elismerést. Ezzel egy időben a céges kiértékelésen a rendszeradminisztráció csoportot választották a három „legjobb tapasztalat” egyikének. A céges kiértékelést hathetente tartják meg a teljes vállalatra, és ez volt az első alkalom, hogy egy csapat jutott a top hármas listára! Mindössze három hónappal korábban ugyanez a csapat a cég szűk keresztmetszetének számított, amire sokan panaszkodtak.

A szolgáltatás minősége egyértelműen nőtt. Hogyan érték el mindezt?

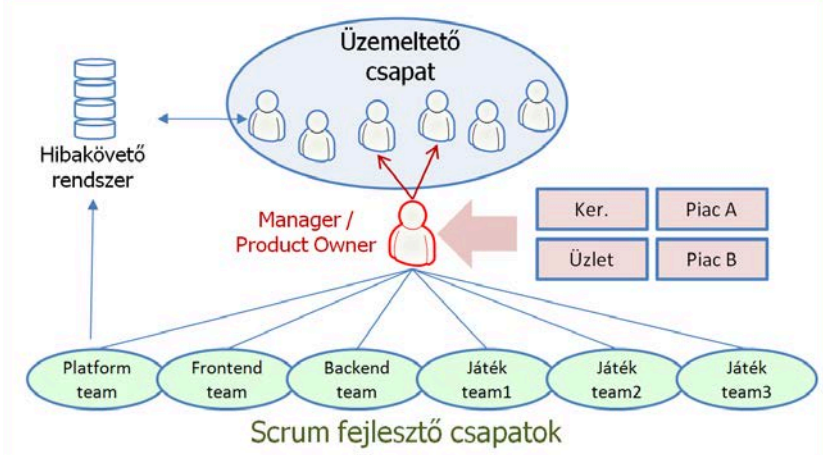
Az áttörést az a pillanat hozta meg, amikor mindenki egy irányba kezdett húzni. A vezetők világos célokat jelöltek ki, és védeni kezdték a csapatot azoktól a feladatoktól, amik nem hozzájuk tartoztak, ők pedig felelősséget vállaltak a minőségért és a határidőkért. Az átállás hozzávetőleg három–négy hónap közötti időt vett igénybe, onnantól viszont minden gördülékenyen ment. Ez nem jelenti azt, hogy a világ összes problémáját megoldottuk (akkor nem lenne munkánk), de új kihívások jelentek meg, például „hogyan tartsuk fenn a csapat lelkesedését a további fejlődésre (ha már többé nem ők a szűk keresztmetszet)?”.

Az önszerveződő team kialakulásának egyik fontos építőeleme a „minden csapathoz egy üzemeltetési kapcsolattartó” elv volt. Ez azt jelentette, hogy minden fejlesztői csapat kapott egy dedikált kapcsolattartót az üzemeltetői csoportból. A Kanban lehetővé tette, hogy az üzemeltető csapat a feladatainak megfelelően maga szervezze meg a munkáját, megakadályozta a túlterheltséget, és ösztönözte a folyamatos fejlődést. Korábban véletlenszerűen kijelöltek valakit a sorból, aki legjobb tudása szerint megoldja a problémát, majd veszi a következő feladatot. Bármilyen félreértés, kommunikációs hiba az egész folyamat új support bejegyzéssel történő újraindulását eredményezte. Amikor az egy az egyhez összerendelési elvet bevezették, lehetővé vált a gyors visszajelzés, ha minőségi problémák veszélyeztették a rendszert.

Idővel egyedi kommunikációs módszerek alakultak ki. Az üzemeltető csapat tagjai instant messaging eszközöket kezdtek használni azokkal a

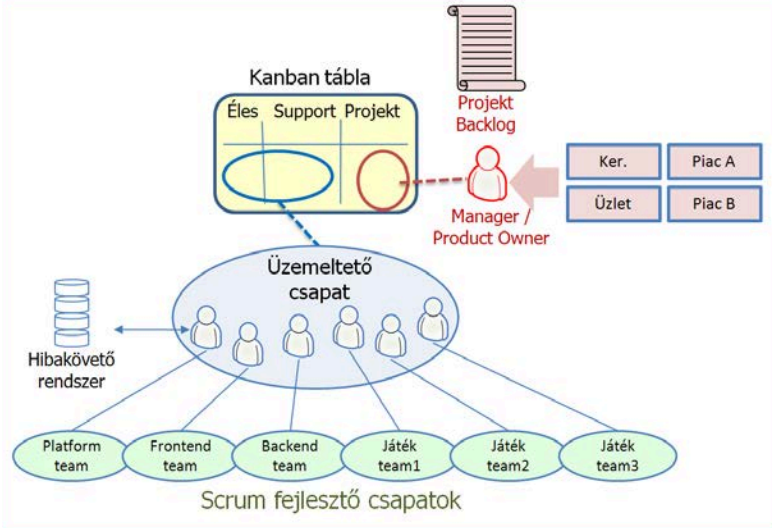
fejlesztőkkel, akiket jól ismertek, e-mailt azoknál, akik jobban írtak, mint beszéltek és telefonáltak, ha ez volt a probléma megoldásának leghatékonyabb módja.

Előtte



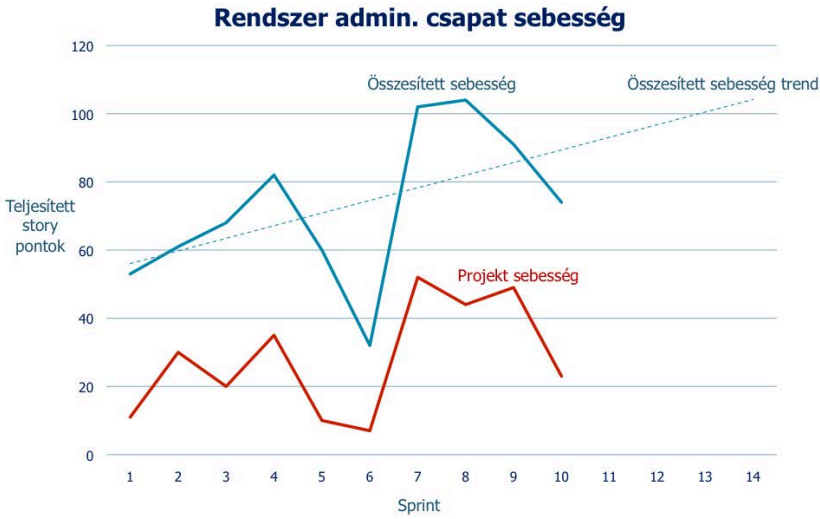
10. ábra. Előtte: a közvetlen vezető a csapat elsődleges kapcsolattartója. Bármi, amit el kell végezni, rajta keresztül kerül a csapathoz. Kisebb problémák – tipikusan a fejlesztőktől – a hibakövető rendszerbe kerülnek. Kevés közvetlen személyes kommunikáció.

Utána



11. ábra. Utána: „teamenként egy üzemeltetési kapcsolattartó” bevezetése. A fejlesztőcsapatok közvetlenül beszélnek az üzemeltetésen kijelölt kapcsolattartóval. Sok közvetlen, személyes kommunikáció. Az üzemeltető csapat tagjai maguk szervezik a munkát a Kanban-tábla segítségével. A vezető a nagyobb projektfeladatokra és komplexebb problémákra fókuszál.

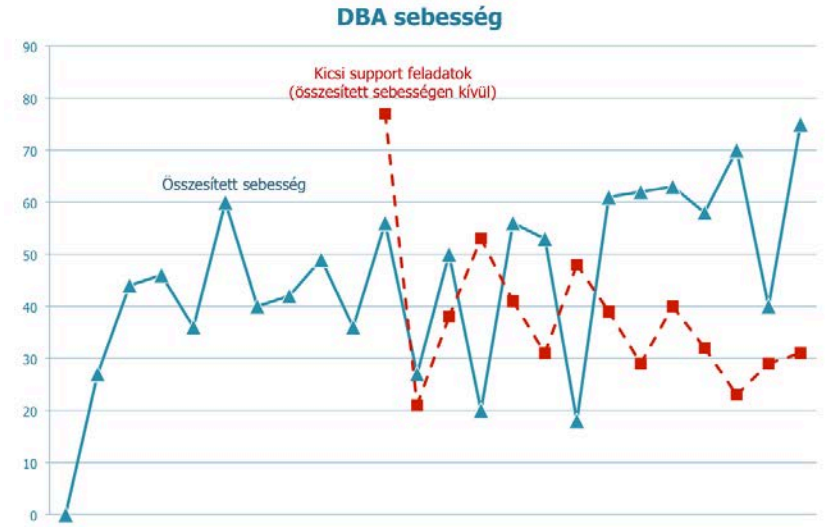
Hogyan hatott ez a csapat teljesítményére?



12. ábra. Összesített és projektsebesség, az egy hét alatt teljesített sztoripontokban kifejezve. Az összesített sebesség az összes oszlop összege, a projektsebesség az a rész, amit a projektek (nagyobb feladatok, pl. hardver platform upgrade) érdekében hajtottak végre. A két zuhanás magyarázata: 1.) olyan hét, amikor a team majdnem minden tagja elutazott; 2.) a fejlesztésről kiadott nagyobb release.

Összegezve, a csapat teljesítménye pozitív trendet mutat. Ezzel egy időben a csapat a páros programozás bevezetésével jelentős erőforrást fektetett a tudásmegosztásba.

Vessünk egy pillantást a DBA csapat teljesítményére is!



13. ábra. Összesített sebesség és kis supportfeladatok. A közepén látható zuhanás a karácsonyhoz köthető.

Az összesített sebesség trendje emelkedik, bár az ingadozás nagy. Az ingadozás nagysága arra ösztönözte a csapatot, hogy vizsgálja a kicsi supportfeladatokat (azok, amik túl kicsik ahhoz, hogy a Kanban-táblára kerüljenek). Mint láthatjuk, a grafikon jól mutatja a kis feladatok száma és az összesített sebesség közötti kedvezőtlen kapcsolatot.

Az üzemeltető csapat később kezdte el a Kanban alkalmazását, mint a másik két team, ezért nincsen még elegendő megbízható adatunk.

Érettség-növekedés

Kezdetben egyszerű volt azonosítani a problémákat, a legnagyobb fejlődési lehetőséget rejtő területek kiválasztása viszont nehéznek bizonyult. A Kanban-tábla az átláthatóság egészen új szintjét tette lehetővé számunkra. Nemcsak a problémák feltárását tette egyszerűbbé, de fontos kérdéseket vetett fel a folyamatos áramlásról, az ingadozásról és a feladatok sorban állásáról. A sorban állás jelenségét a problémák felismerésére kezdtük használni. Négy hónappal a Kanban bevezetése után a vezetők vadászni kezdtek a csapatokat akadályozó ingadozások okaira.

Ahogy a csapatok egyénekből önszerveződő csoportokká alakultak, a vezetők új menedzsmentkihívásokkal találták szembe magukat. Többet kellett foglalkozniuk az emberi tényezővel – panaszok kezelése, közös

célok meghatározása, konfliktusok feloldása és megállapodások kialakítása. Nem volt fájdalommentes átalakulás – nyíltan bevallották, hogy mindez jártasságot és erőfeszítést igényel, de vállalták a kihívást, és végül jobb vezetőkké váltak.

32

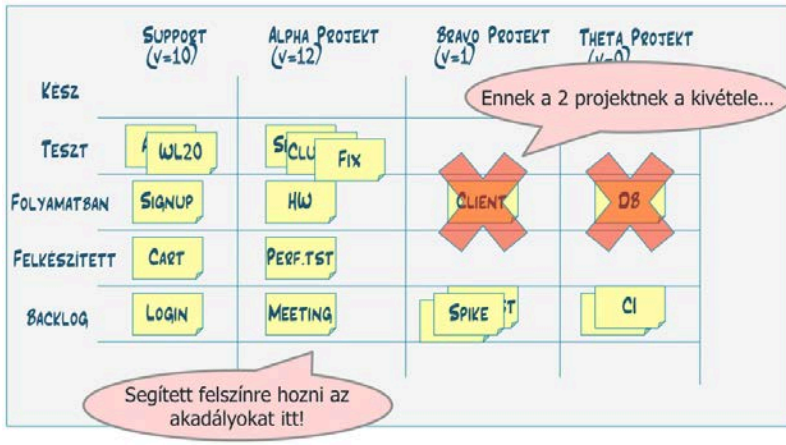
Általános tanulságok

A párhuzamos feladatok csökkenésével korlátok merülnek fel

Az összes csapat meglehetősen nagyvonalú WIP-korláttal indult. Ebben az időben a legtöbb energia az áramlás kialakítására, és annak biztosítására ment el, hogy a szervezet megkapja a szükséges támogatást.

Az elején a vezetők több párhuzamosan futó projektet akartak, de néhány héten belül világossá vált, hogy nincs elég kapacitás az alacsonyabb prioritású projektekre. Mindössze a táblára vetett egyetlen rövid pillantás megmutatta, hogy soha nem kezdődött meg a munka egyetlen alacsony fontosságú ügyön sem. Ez arra késztette a vezetést, hogy csökkentse az egy csapatra jutó projektek számát.

Idővel, ahogy a fontos feladatok végrehajtása egyenletesebbé vált, csökkenteni kezdtük a WIP–korlátokat; a folyamatban lévő projektek (az oszlopok) számának háromról kettőre, majd egyre csökkentésével tettük, melynek hatására elkezdtek felszínre kerülni a csapaton kívüli korlátok. A csoportok tagjai jelezni kezdték, hogy nem kapnak időben segítséget másoktól, a vezetők figyelme tehát erre a problémára irányult.



Felszínre került továbbá, hogy a más csapatoktól kapott gyenge minőségű eredményeknek milyen negatív hatása van a teljesítményre. A folyamatos áramlás fenntartása nehéz, ha a beérkező inputok állandó javítást igényelnek.

Ezek a problémák korábban sem voltak ismeretlenek. A kérdés inkább az volt, hogyan állapodjunk meg abban, hogy „melyik problémával foglalkozzunk elsőként”? A Kanban-táblán mindenki láthatta, hogy egy bizonyos problémának milyen hatása van az áramlásra. Ez segített lendületet gyűjteni a szervezeti határokon átnyúló problémák leküzdéséhez.

A tábla idővel változni fog, ne véssük kőbe

Idővel az összes Kanban-tábla megváltozott. Átlagban kétszer-háromszor újra kellett tervezni, mire a csapat rátalált a megfelelőre. Az első tábla elrendezésére valószínűleg felesleges sok időt fordítani. Ügyeljünk arra, hogy a tábla könnyen átalakítható legyen! A sávok megrajzolására mi fekete ragasztószalagot használtunk, amit egyszerű volt mozgatni, és alkalmazhattuk fehér táblán és falon is. Láttam olyat, hogy vastag filccel rajzolták meg a rácsokat (de ügyeljünk arra, hogy törölhető legyen!)

Alább látható egy tipikus példa az elrendezés optimalizálására. A prioritások eleinte gyakran változtak, ezért – elkerülendő az egész oszlopnyi cetlik oda-vissza mozgatását – a csapat az oszlopok fölé ragasztott papírokra rögzítette a prioritásokat.

14. ábra. Kezdeti Kanban-tábla, ragadós cetlikre rögzített prioritásokkal

Ne féljünk a kísérletezéstől és a kudarcoktól!

A tanulság, amit ebből a kalandból levontam, hogy nem létezik végállomás. Abban a pillanatban elbuktunk, amint azt gondoljuk, hogy mégis van. Csak a vég nélküli kísérletezés és tanulás létezik. Kudarcok nélkül nincs fejlődés. Mi is több alkalommal hibáztunk az út során (rossz táblaelrendezés, becslések, redundáns grafikonok stb.) – de minden alkalommal valami újat és fontosat tanultunk. Ha leálltunk volna a kísérletezéssel, hogy tanulhattunk volna?

A Kanban sikere mára arra ösztönözi a vezetőket és a Scrum-csapatokat, hogy kísérletezzenek a Kanban-táblákkal is. Reméljük, ez a könyv segít ebben!

Végső tanulságok

Kezdjük a kiértékelésekkel!

Sok az alternatíva és az átgondolandó kérdés, ugye? Reméljük, ez a könyv segített kicsit a köd eloszlatásában. Nekünk legalábbis igen.

Ha a folyamatok változtatása és javítása a cél, hadd segítsünk a döntésben! Ha eddig nem tartottunk rendszeres időközönként kiértékelést, akkor kezdjük ezzel! Gondoskodjunk róla, hogy ezek valódi változásokhoz vezessenek! Vegyünk igénybe külső moderátort, ha szükséges.

Ha már hatékony kiértékeléseket tartunk, máris elindultunk a megfelelő folyamatok felé vezető utunkon, alapuljon ez akár Scrum, XP, Kanban, ezek kombinációján, vagy bármi más módszeren.

Soha ne álljunk le a kísérletezéssel!

Nem a Scrum- vagy a Kanban-rendszer a cél, hanem a folyamatos tanulás! A szoftverfejlesztés egyik nagyszerű tulajdonsága éppen a gyors visszacsatolás lehetősége, ami pedig a tanulás kulcsa. Használjuk ki a visszacsatolásban rejlő lehetőségeket! Kérdőjelezzünk meg mindent, kísérletezzünk, hibázzunk, tanuljunk és próbálkozzunk újra! Ne akarjunk elsőre tökéleteset, mert úgysem lesz az! Egyszerűen kezdjük el valahol és fejlődjünk onnan tovább.

Az egyetlen *igazi* hiba nem tanulni a hibákból.

De végül is ebből is tanulhatunk...

Sok sikert, és jó utat!

Henrik és Mattias

Stockholm 2009.06.24

H: Ez minden?

M: Úgy gondolom, itt álljunk meg.

H: Talán bemutatkozhatnánk…

M: Jó ötlet. Ha jó fejeknek látszunk, talán szerzünk tanácsadói megbízásokat.

H: Akkor rajta!

M: Igen, nekünk is és az olvasónak is van más dolgunk.

H: Éppenséggel nekem mostanában kezdődik a szabadságom :o)

M: Hé, ne bosszants!

A szerzőkről

Henrik Kniberg és Mattias Skarin a stockholmi Crisp cég tanácsadói, akik örömmel segítik a cégek szoftverfejlesztési gyakorlatának emberi és technikai oldalának fejlesztését. Vállalatok tucatjait támogatták a Lean- és Agilis-elvek gyakorlati bevezetésében.

Henrik Kniberg

Az elmúlt évtizedben Henrik három svéd informatikai vállalat IT igazgatója volt, és sok másikat segített folyamatainak javításában. Certified Scrum Trainer, rendszeresen dolgozik együtt a Lean- és Agilis-fejlesztés úttörőivel, mint Jeff Sutherland, Mary Poppendieck és David Anderson.

Henrik előző könyvét a Scrum and XP from the Trenches-t több mint 150 000-en olvasták, és a téma egyik legnépszerűbb könyve. Számos alkalommal nyerte el nemzetközi konferenciák legjobb előadójának járó címét.

Tokióban nőtt fel, most Stockholmban él feleségével, Sophiával, és három gyermekével. Szabadidejében aktív zenész és zeneszerző, basszusgitáron és billentyűs hangszereken játszik.

henrik.kniberg<at>crisp.se

http://blog.crisp.se/henrikkniberg

http://www.crisp.se/henrik.kniberg

Mattias Skarin

Mattias Lean-tanácsadóként segíti a vállalatokat a Lean- és Agilis-módszerek előnyeinek kiaknázásában; a fejlesztőktől a menedzsmentig minden szintet mentorál. Részt vett egy játékfejlesztő cég fejlesztési idejének huszonnégyről négy hónapra csökkentésében, visszaállította a bizalmat egy egész fejlesztési részleg iránt, a Kanban egyik úttörője.

Vállalkozóként két cég társalapítója és irányítója.

Mattias minőségbiztosításból szerzett egyetemi diplomát, és tíz évig dolgozott fejlesztőként kritikus feladatokat ellátó rendszerek fejlesztésén.

Stockholmban él, szabadidejében rock and roll táncos, versenyző, síelő.

mattias.skarin<at>crisp.se

http://blog.crisp.se/mattiasskarin

http://www.crisp.se/mattias.skarin