Agilis módszertan bevezetése az egyetemi projektlaborban

Kolozsvári Szilvia[1], Kiss Attila[2], Molnár Bálint[3]

[1] szkolozsvari@gmail.com
[2] kiss@inf.elte.hu
[3] molnarba@gmail.com
ELTE IK

Absztrakt. Az Eötvös Loránd Tudományegyetem Informatikai Karának Információs rendszerek tanszéke 2011 óta működtet egy ún. projekt labort, amelynek elsődleges célja a BSc-t végző, és MSc-s hallgatók számára szakmai gyakorlat lehetőségének biztosítása. A laborban a hallgatók egy szemeszteren keresztül valós (azaz megrendelés formájában külső partnertől érkező) projektfeladaton, vagy ennek hiányában, az egyetemi oktatók által kijelölt kutatás-fejlesztés feladaton dolgoznak a valós kisvállalati projektkörnyezetet szimuláló keretek között. Ez azt jelenti, hogy a labormunkára jelentkezettek, és kiválasztottak a szemeszter elején projekt csapatot alkotnak (projektenként 5 – 15 fővel). A csapat célja a szemeszter végére a kijelölt informatikai feladat (projekt) megvalósítása, egyetemi oktatók és doktoranduszok vezetésével, a meghatározott projektvezetési módszertan szerint.
A labormunka eddig a klasszikusnak mondható vízesés modell-szerű fejlesztési módszertant alkalmazta, azaz a szemeszter elején a feladat ismertetése után a diákok foglalkoztak egy kicsit a megvalósítási lehetőségek feltárásával, majd megtervezték a feladatot, aztán megvalósították, tesztelték és a szemeszter végén bemutatták. Mivel a projekt labor fontos célja, hogy a diákok megismerkedjenek a valóság-közeli projektmunkával, fontos, hogy olyan környezetet teremtsünk számukra a laboron belül, amely nagymértékben hasonlít a vállalati működéshez. Napjainkban azonban egyre kevesebb azon vállalatok száma, amelyek a klasszikus vízesés modellt alkalmazzák informatikai fejlesztési projektjeik vezetésére, ugyanakkor egyre divatosabbá válik az ún. agilitás alapú fejlesztés.

Ebben a tanulmányban kidolgoztunk egy agilis elveken alapuló fejlesztési módszertant a tanszéki projektlaborunk számára. A módszertan általános ismertetését követően megvizsgáljuk, hogy milyen feltételekkel tudjuk ezt bevezetni a tanszéki projektlaborba, majd összefoglaljuk vélt előnyeit, hátrányait, használhatósági szempontjait a jelenleg alkalmazott módszertannal szemben. A jövőben tervezzük az ebben a tanulmányban leírt módszer bevezetését a gyakorlati oktatásba, és ez által gyakorlati tapasztalatok gyűjtését a módszertan hatékonyság-elemzéséhez.

Bevezetés

A tanulmány első fejezetében részletezzük, hogy mi az egyetemi projekt labor létrejöttének és működésének a célkitűzése, miért tartjuk szükségesnek, hogy az egyetem rendelkezzen ilyen fajta laborgyakorlati lehetőségekkel.

A második fejezetben a labor feladatát tárgyaljuk, azaz a célkitűzés(ek) elérése érdekében milyen feladatokkal kell megbirkóznunk. A laborszervezők fő feladata a projektötlet megtalálása, azaz annak a feladatnak a meghatározása, amin a laboron résztvevő diákok a félév során dolgozni fognak. A projektötlet kiválasztásának szempontjait részletezzük ebben a fejezetben.

Miután már tudjuk, hogy milyen céllal akarjuk üzemeltetni a labort és ennek a célnak az elérésére mi a kiválasztott projektfeladat, meg kell néznünk, hogy milyen más feltételek teljesülése szükséges még a labor gyakorlatban való (folyamatos) üzemeltetéséhez. A 3. fejezet arra ad választ, hogy hogyan találjuk meg az ideális csapatot a labormunkához.

Már van célunk, feladatunk, csapatunk. Az a kérdés maradt hátra, hogy a csapat hogyan fogja elvégezni a feladatát a cél elérése érdekében, azaz másként fogalmazva, mi legyen a csapat működésének módszertana. Ehhez választottuk az agilis fejlesztési módszertanok közül a SCRUM módszert, aminek elméleti bemutatását részleteztük a 4. fejezetben.

Az 5. fejezet szól arról, hogy az elméleti SCRUM módszert hogyan lehet alkalmazni a laborgyakorlatban. Megvizsgáltuk a szerepkörök leosztását, az alkalmazandó eszközöket, a feladatokat és felelőségi köröket.

A konkrétumok vizsgálatához feltételeztünk egy olyan környezetet, amely az egyetemi mindennapok gyakorlatában a lehető legnagyobb valószínűséggel következik be. A 6. fejezetben azonban kitérünk néhány gondolattal a feltételezett világon kívüli világra, és megvizsgáljuk, hogy milyen hatással lenne ez a leírt módszertanra vonatkozóan.

Végül a 7. fejezet a jövőbeli kutatási terveinkről szól, aminek célja megvizsgálni és következtetéseket levonni a gyakorlatban alkalmazott módszer hatékonyságáról, aminek alapja a laborban jelenleg alkalmazott fejlesztési módszerrel történő összehasonlítás.

1. A labor célja

Az egyetemi képzések a hangsúlyt leginkább az elméleti oktatásra helyezik, ami alatt a diákok elsajátíthatják a szakmához szükséges tudásanyag elméleti alapjait. Bár az előadásokon kívül a legtöbb tantárgy rendelkezik gyakorlati órákkal is, ám ezek a gyakorlatok a konkrét elmélet jobb megértését szolgáló, célirányos és rendszerint példa jellegű, mikro-feladatok megoldására összpontosul. Az egyetemi oktatásokban a gyakorlati órák tehát elengedhetetlenül fontosak az elméleti tananyag elmélyítésében, viszont nem szolgálják azt a célt, hogy a diák betekintést kaphasson a munka világába, azaz megtapasztalhassa, hogy az iskolapadból kikerülve milyen elvárásokkal, és milyen kihívásokkal fog szembesülni, amennyiben elhelyezkedik egy szoftverfejlesztő cégnél / vállalatnál.

A labor létrejöttének elsődleges célja az volt, hogy lehetőséget biztosítson a Bsc-s, Msc-s hallgatóknak, hogy megtapasztalhassák, milyen elvárásokkal fognak szembesülni a leendő munkahelyeiken. Ennek a célnak az elérése egy másik szempontból is nagyon fontos, mégpedig az, hogy a cégek munkaerő felvételnél rendszerint előnyben részesítik azokat a jelentkezőket, akiknek már van valamennyi projekt-tapasztalatuk, azokkal szemben, akik még soha sem vettek részt projekt munkában.

A labor működése a doktorandusz hallgatók szempontjából is egy nagyon ígéretes tapasztalatszerzési lehetőség. Az esetek többségében doktorandusz hallgatókká az Msc-s képzést közvetlenül azt megelőzően elvégző, kiemelkedő képességű diákok válnak, akik további ismeretekre szándékoznak szert tenni. Az esetek nagy százalékában a doktori képzést nem előzi meg munkavállalói múlt, a doktori hallgatók így továbbra is gyakorlati tapasztalat nélküli elméleti szakemberekké válnának, akiknek a képzés utáni érvényesülése válna nehézkessé. A labortevékenységeken belül általában projektvezetéssel, illetve szervezési, koordinálási, segítési (coaching) feladatokkal foglalkoznak, ami által csapat vezetői, szervezői gyakorlatokra tesznek szert.

Nem utolsó sorban fontos szempont az egyetem számára, hogy folyamatos piaci kapcsolatokat ápoljon kis-közép és nagy informatikai vállalatok köréből. Az elméleti kutatásokon túlmenően egyre nagyobb hangsúly fektetődik az alkalmazott kutatásokra, továbbá a szakember-utánpótlás szempontjából is ösztönző hatással bír egy ilyen labor működése a piaci szereplőkre nézve, valamint egyfajta természetes evolúciós kapcsolatot teremt az egyetem elméleti kutatói-oktatói és a gyakorlati alkalmazás között. [1]

Összefoglalva a labor célja, hogy a hallgatók számára céges működést szimuláló projekt-környezetet teremtsen, amin belül gyakorlat-orientált projekt-tapasztalattal gazdagodhatnak a résztvevők. A cél elérése érdekében a labor feladata aktív piaci kapcsolatok feltárásán keresztül konkrét megrendelői projekt-feladatok keresése.

Természetesen érezhető, hogy a megfogalmazott feladat kulcsfontosságú a labor aktív és eredményes működéséhez, ám a folyamatos működtetés érdekében fel kell készülni olyan esetre is, amikor éppen nincs konkrét piaci megrendelés. A labor működési modellje lényegében ki kell, hogy térjen arra az esetre is, amikor nem piaci megrendelés alapú projekt-munkán dolgozik a labor-csapat. Minekután ez külső feltételektől nem függő működési modell, ezért jelen módszertani leírásban ezt részesítettük előnyben. A tanulmány végén kitérünk azonban azokra a különbségekre, amelyek konkrét piaci megrendelésen alapuló projektek esetében lépnek fel.

2. A labor feladata

Az előző fejezetben részletesen áttekintettük, hogy milyen célkitűzések mentén jött létre az egyetemi projektlabor. A labor céljainak elérése érdekében definiálnunk kell azokat a feladatokat, amelyek elvégzése elengedhetetlen a meghatározott célok eléréséhez. Láttuk továbbá, hogy a labor elsődleges feladata a projekt kiválasztása, amin a labor résztvevői tevékenykedni fognak. Vizsgáljuk azt az esetet, amikor nem konkrét piaci megrendelésen alapuló projektet választunk a labormunka számára. Ebben az esetben a labort szervező egyetemi oktatók feladata megtalálni és kiválasztani a projektet, amin az adott időszak alatt a labor résztvevői dolgozni fognak.

2.1. Milyen típusú legyen a projektfeladat?

A projektötlet kiválasztásánál érdemes figyelembe venni azt a szempontot, hogy a projektmunka milyen jellegű tevékenységre irányuljon. Gyakorlatorientáltság szempontból három csoportba sorolhatóak a projektötletek:

- az alapkutatás jellegű ötletek egy adott problémára keresnek újfajta, innovációs megoldást. Egy alapkutatás időtartama és kimenetele nem meghatározható és ezáltal nem becsülhető, továbbá olyan széleskörű ismereteket kíván, amelyek a hallgatóktól nem várhatóak el, és a labormunkára szánt időtartam alatt nem szerezhetők meg. Ez azt jelenti, hogy alapkutatás jellegű projektötletek nem alkalmazhatóak a laborkörnyezetben
- alkalmazott kutatás és kutatás fejlesztés jellegű ötletek helyzete lényegében eltér az alapkutatásétól. Ebben az esetben egy már meglévő elméleti tudást kellene alkalmazni a gyakorlatban. Többnyire ez azt jelenti, hogy az adott tudás gyakorlati alkalmazásában ez a projekt kellene, hogy élen járjon, azaz kitaposatlan ösvényen keresztül kellene a diákoknak haladnia. Az ilyen fajta projektek kivitelezése nem lehetetlen, de meglehetősen sok kockázattal jár, és sikere nem garantált. Bsc-s hallgatók esetében nem célszerű az alkalmazása, mert számukra az alaptudás megszerzése is még csak folyamatban van, nem várható el, hogy az alaptudást meghaladó, arra szorosan épülő elméleti innováció gyakorlati hasznosításában élenjáróak legyenek. Msc-s vagy Phd hallgatók körében már kevesebb kockázattal alkalmazható, de mindenképpen célszerű a feladatot körültekintően a csapat képességeihez, elvárásaihoz és céljaihoz igazítani.
- alkalmazott fejlesztési projektek a legkevesebb kockázattal, legcélszerűbb módon alkalmazhatóak laborkörnyezetben. Ezek olyan feladatok, amelyek megvalósítása nem jelent újdonságot a szakemberek számára, azaz nagy biztonsággal előfordul majd olyan csapattag vagy környezet, aki a téma szakértőjének tekinthető, és problémás

kérdésekben döntéshozóként vagy tanácsadóként tud működni. Ezeknek a projekteknek a sikeresség-kockázata a legalacsonyabb, így nagyobb biztonsággal alkalmazhatóak laborkörnyezetben a labor céljainak elérése érdekében.

2.2. Mekkora legyen a projektfeladat?

A laborfeladat kiválasztásánál további fontos szempont az átfutási idő becslése. Mivel a labor célja az, hogy megismertesse a diákokkal a projektek működését a gyakorlatban, ezért fontos olyan feladatot választani, aminek teljesítési ideje nem hosszabb egy szemeszter alatt a labormunkára szánt időnél. A gyakorlatban projekteken dolgozó szakemberek tudják, hogy a projektmunka talán legnehezebb feladata megfelelő idő- és erőforrásbecslést készíteni a projekt elején, amikor még a feladat sincs részletesen definiálva, és a csapat teljesítménye sem ismert.

A becslés pontossága nagymértékben függ a csapat teljesítőképességén, ami viszont – új csapat lévén – nehezen becsülhető. Ezért célszerű bizonyos irányelveket megfogalmazni a csapat kiválasztásával kapcsolatosan, amihez ha sikerül igazodni, csökkenthető a becslés pontatlansága. Például érdemes meghatározni egy várt irányszámot a csapat létszámára vonatkozóan, továbbá a csapattagok kompetenciájára és azon belül tudásmélységére vonatkozóan is érdemes irányelveket megfogalmazni, és a csapattagokat ezen irányelvek alapján kiválasztani. Ez természetesen nem jelenti azt, hogy a gyengébb tanulók nem vehetők fel labormunkára, azt viszont jelenti, hogy a csapat vagy csapattagok összeállításánál érdemes figyelni arra, hogy különböző kompetenciákkal, és különböző tudásszintekkel rendelkező tagok kerüljenek be.

A becslés készítéséhez ismerjük továbbá a labormunkára szánt idő mennyiségét viszonylag nagy pontossággal. Ez az a dimenzió, ami – speciálisan a labor esetében – teljesen fixnek vehető, ugyanis tantárgyként meghirdetett munka lévén nincs rá lehetőség, hogy a projekt futamideje alatt a ráfordítandó időt (túlórázással) megnöveljük.

A legrugalmasabban állítható paraméter ebben az esetben a projektfeladat mennyisége, főleg ha nem piaci megrendelésről van szó. A munkamennyiség futamidő alatti változtatásával tudjuk esetlegesen majd kompenzálni a becslésünk pontatlanságából eredő csúszást. Ez az egyik oka annak, hogy célszerűnek láttuk a laboron belüli projektmunkát agilis alapokra átszervezni.

A feladat nagyságának kiválasztásához a fent ismertetett fix és bizonytalan paraméterek ismerete mellett alkalmazhatjuk a funkciópont analízis módszert. [2]

Egy információrendszer méretét funkciópont index formájában a következő módon lehet megállapítani: a rendszer összes logikai tranzakciójára össze kell számolni a bemenő, a kimenő adatokat és a feldolgozás során érintett entitásokat (adatcsoportokat). Ezután az első lépésben matematikai módszerekkel korrigálatlan funkciópontot számítunk, a másodikban pedig a műszaki bonyolultságnak megfelelően korrigáljuk az eredményt, figyelembe véve a technológiai tényezőt. [2]

A rendszer funkciópont méretét az 0,4-ik hatványra emelve a fejlesztésre fordítandó naptári hónapok közelítő értékét kapjuk. (A megfigyelt értékek 0,32 és 0,45 között szórnak.)

Az alkalmazás elkészítéséhez szükséges személyzet létszámát úgy becsülhetjük meg, hogy az alkalmazás funkciópont értékét elosztjuk 150-nel.

Nézzünk példaként egy 100 funkciópontos projektet. A funkciópont analízis módszer azt mondja, hogy a funkciópontok mértékét 0,4-ik hatványra emelve megkapjuk a projektre fordítandó idő becslését. Ez esetünkben 6,3 hónap. Továbbá ha a funkciópontok számát elosztjuk 150-nel, becslést kapunk a szükséges fejlesztői erőforrás számára. Ez esetünkben 1 fő. Mivel a laborban alapfeltétel, hogy egy szemeszter alatt, nem teljes munkaidőben, csapatnak kell dolgoznia a projektfeladaton, így vizsgáljuk meg a továbbiakban, hogy mekkora funkciópontos

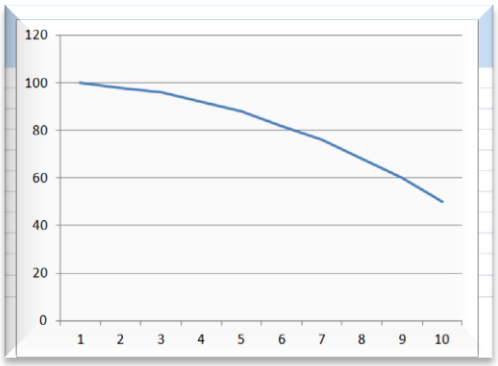
projektet, hány fős csapattal érdemes elvégeztetni ahhoz, hogy optimális eredményt kapjunk. Feltételezzük, hogy a labor projektmunkára heti 2 x 3 órát szánunk.

Az alábbi táblázat mutatja, hogy a fejlesztők számának növelésével hogyan változik a becsült ráfordítás mennyisége.

Funkciópont	Naptári hónapok száma (heti 5*8 óra)	Egyetemi hónapok száma (heti 2*3 óra)	Munkaórák száma	Fejlesztők száma	Fejlesztői egység
100	6,3	42	1008	1	100
	3,2	21,4	514	2	196
	2,2	14,6	350	3	288
	1,7	11,4	274	4	368
	1,4	9,5	229	5	440
	1,3	8,5	205	6	492
	1,2	7,9	189	7	532
	1,2	7,7	185	8	544
	1,2	7,8	187	9	540
	1,3	8,4	202	10	500

1. ábra Funkciópont elemzés

A táblázat utolsó oszlopában szereplő fejlesztői egység azt jelenti, hogy a csapat hány egységnyi feladatot tud ellátni adott (fix) idő alatt. Ha növeljük a csapat méretét, a csapat által elvégezhető feladategységek száma nem lineárisan fog növekedni. Ezt szemléltetik az utolsó oszlop adatai, valamint grafikus nézetben az alábbi diagram.



2. ábra Létszám szerinti elmozdulás

A táblázatból kiolvasható továbbá, hogy 7-9 fejlesztő esetében érjük el az optimumot, azaz ha efölé emelnénk a csoport létszámát, akkor a csoport produktivitása csökkenne. Látható viszont, hogy heti 2*3 órás ráfordítással számolva optimális esetben is közel 8 hónapra lenne szükség, hogy a feladatot elvégezze a csapat. Ez a munkamennyiség a laborunk esetében egy szemeszternyi ráfordított időbe nem férne bele. Mit tudunk ilyenkor tenni? Segítségül hívhatjuk ilyenkor a projekt menedzsment módszertanok klasszikusnak mondható scope- ütemezés – erőforrás kölcsönhatást vizsgáló, ún. minőség háromszögét. [4]

3. ábra Minőség háromszög

A minőség-háromszög lényege, hogy egy azonos szintű minőség megtartásához a háromszög oldalai mentén mozdulhatunk el, azaz vagy az erőforrást növeljük, vagy az ütemezést toljuk ki, vagy a scope-ot csökkentjük. Az előbbiekben láttuk, hogy a laborunk tekintetében az ütemezés nem módosítható, az erőforrás szempontjából meghatároztuk az optimumot, azaz erőforrás növeléssel nem tudnánk elérni a célunkat, és természetesen a minőség rovására sem akarjuk vezetni a projektünket. Nem marad más mozgásterünk, csak a scope. Kimondható tehát, hogy egy szemeszternyi labormunka alatt 100 funkciópontos feladat nem teljesíthető. Csökkentsük tehát a feladatot, és nézzük meg, hogy mekkora csökkentést kell végeznünk ahhoz, hogy az adott erőforrásból, adott ütemezés szerint, meghatározott minőségben teljesíteni tudjuk a feladatot.

Számításunkat figyelembe véve látható, hogy 35 – 55 közötti funkcióponttal rendelkező feladat végezhető el a laborban.

Funkciópont	Naptári hónapok száma	Egyetemi hónapok száma (heti 2*3 óra)	Munkaórák száma	Fejlesztők száma	Fejlesztői egység
45	4,6	30,7	736	1	100
	2,4	15,7	376	2	196
	1,6	10,7	256	3	288
	1,3	8,3	200	4	368
	1	7	167	5	440
	0,9	6,3	150	6	492
	0,9	5,8	138	7	532
	0,8	5,6	135	8	544
	0,9	5,7	136	9	540
	0,9	6,1	147	10	500

4. ábra Funkciópont elemzés

Összefoglalva: a labor projektcsapatának nagysága optimálisan 7-9 fő, akik egy szemeszterben heti 2x 3 óra ráfordítással átlagosan 35 – 55 funkciópont közötti feladatot tudnak megvalósítani.

3. Hogyan indítsuk el a labort?

Azzal a céllal hoztuk létre a projektlaborunkat, hogy tevékenykedése a körülményekhez képest a lehető legjobban hasonlítson egy kis fejlesztőcég működéséhez. Nézzük meg tehát, hogyan indít projektet egy fejlesztőcég. Természetesen a projektindítás körülményei cégenként, projektenként, tevékenységenként, projekt méretenként nagyon eltérőek is lehetnek, a

körülmények, a szokás és az emberi szubjektivitás vezet sok esetben az indító döntéshez. Abban azonban közösek, hogy valamilyen módon minden fejlesztőcég

- keresi a megrendelőket, majd
- bekéri a koncepciót,
- megtérülés becslést végez (értékel) és
- dönt, hogy elvállalja-e a megvalósítást.

E négy alapfunkció minden projekt indítása előtt fellelhető, a különbség az alapfunkciók kivitelezésének módjából adódik. A felsorolt egyes alapfunkciók kivitelezésére további módszertanok találhatók például marketing, kommunikáció területén a megrendelők felkutatására vagy éppen pénzügyi oldalról a megtérülés számítás elvégzésére. Az alkalmazott módszerek továbbá függnek a cég éppen aktuális vállalati stratégiájától is, és a cég életében, akár projektenként, akár az eltérő körülmények hatására dinamikusan változtathatók. Összefoglalva a projekt indítása előtti tevékenységsorozat választása cég specifikus, a cég aktuális stratégiájához illeszkedő, de akár időben és térben a körülmények hatására változtatható módszertan.

Laborunk esetében tehát a projektindítást megelőző négy alaplépést el kell végeznünk, viszont az alaplépések elvégzésének módszertanát úgy kell kialakítanunk, hogy az illeszkedjen az egyetem által biztosított és megkövetelt környezeti tényezőkhöz. A következőkben nézzük meg részletesen, hogyan lehetne kivitelezni egyetemi környezetben a cég alapfunkciót.

3.1. „Megrendelők” keresése

E tanulmány korábbi fejezetében részleteztük és kiválasztottuk azt az irányt, hogy megrendelőknek belső (azaz egyetemen belüli tanárok vagy hallgatók) erőforrásokat fogunk választani, azaz nem piaci megrendelés indíttatású projekteket fogunk első körben vizsgálni.

A megrendelő az a természetes vagy nem természetes személy vagy csoportosulás, aki valamilyen jól meghatározott cél elérése érdekében valamilyen tevékenységsorozatot kíván elvégeztetni a szállítóval, és ezért valamilyen pénzbeli vagy természetbeli juttatást biztosít.

Jelen esetünkben a labor lesz a szállító, és meg kell keresni azokat a belső (egyetemen belüli) személyeket vagy személyek csoportosulását, akik a megrendelő szerepét fogják betölteni a projektmunkában. A megrendelő lehet tehát tanár vagy diák / diákok csoportja.

- ha úgy döntünk, hogy a megrendelő tanár legyen, annak előnye, hogy a projektfeladat volumenét és minőségét tekintve jól definiált, előre mutató, összeszedett lesz. Hátrányai viszont számottevőek:
    - a megrendelő tanár a labor teljes működése alatt rendelkezésre kell, hogy álljon a megrendelő szerepében, azaz viszonylag nagyfokú terheltséget jelent a tanárnak
    - a tanár-diák viszony miatt egyfajta alá-fölérendelt viszony jön létre a megrendelő és a szállítók között, ami a valós céges projektekre kevésbé jellemző, ami távolítana attól a céltól, hogy a labor meglehetősen jól szimulálja a céges projektmunkát.
    - továbbá ha több projekten (több féléven) keresztül ugyanaz a tanár választ feladatot, akkor előbb-utóbb tipizálttá válik a labor, mert bizonyára hasonló típusú feladatok fognak félévről félévre visszatérni.
- ha diákot választunk erre a feladatra, annak előnyei:

  - fel tudjuk mérni a diákok érdeklődési körét, ötletelésre, gondolkodásra tudjuk őket bírni, amely akár a későbbi szakdolgozati témaválasztáshoz is segítséget nyújt majd nekik
  - olyan diákok is teret kaphatnak a labormunkában, akiknek elsődlegesen nem a kódolás a céljuk és / vagy erősségük, inkább tervezőként, mintsem kivitelezőként szeretnének részt venni
  - az egyetemi oktatóknak megmarad a tanácsadói, segítői szerep, a projekt csapatot nem kényszerítjük bele az alá-fölérendelt viszonyba. Az oktatók leterheltsége ezáltal nem nő drasztikusan.
- Hátrányai a következők:
  - a projekt feladat kiválasztása hosszabb időt vesz igénybe, mert össze kell gyűjteni a diákok ötleteit, értelmezni és minősíteni kell azokat, és kiválasztani belőlük a megvalósítandót. Mindezt nem lehet előkészíteni a félév megkezdése előtt, tehát a projekt előkészítő munkái csökkentik a félévből a tényleges projektmunkára szánt időt.
  - előző fejezetekben láttuk a félév során a megvalósításra szánt idő viszonylag szűkös, ami a projekt terjedelmét jelentősen befolyásolta (mondhatjuk, minimálisra szorította), ez azt jelenti, hogy a projekt előkészítő feladatokat célszerű lenne a megvalósítás félévétől külön választani, azaz a megelőző félévben elvégezni. Fontos továbbá, hogy a kiválasztott megrendelő a projekt időtartama alatt aktív szereplője legyen a projekt csapatnak, tehát a kiválasztást úgy kell időzíteni, hogy a megvalósítás félévében is jelen legyen a kiválasztott megrendelő. Emiatt célszerű a kiválasztást és projekt előkészítő munkákat az első félévre, míg a megvalósítást a második félévre időzíteni.
  - továbbá jelentős kockázati tényezőt jelent a „megrendelő” szerepkörbe választott diák motivációjának fenntartása, mert a megrendelő kulcsszerepet tölt be a projekt életében. Megrendelő nélkül tulajdonképpen nincs projekt. Ha elveszítjük a megrendelőnek választott diák motivációját, akkor a labor működését úgy tudjuk továbbra is fenntartani az adott projekt tekintetében, ha ezt a szerepkört valaki más (esetünkben leginkább oktató vagy phd hallgató, kisebb eséllyel más diák) veszi át.

Összefoglalva, ha diákot szeretnénk választani megrendelői szerepkörbe, annak hátrányai és kockázata másmilyen, de nem kisebb, mintha ezt a szerepet oktató töltené be. Előnyeinek ismeretében az egyetemi oktatóknak kell mérlegelniük, hogy vállalják-e ezeket a kockázatokat vagy nem. A továbbiakban a tanulmányban azt feltételezzük, hogy az oktatók úgy döntenek, hogy a diákok közül választanak megrendelőt.

3.2. Projektötletek bekérése

Nézzük meg, hogyan történhet a megrendelő kiválasztása. A kiválasztáshoz a következőket kell végrehajtani:

- az első (őszi) félév elején, miután már stabilizálódtak a tantárgyfelvételek, írjunk ki egy felhívást a labormunka során megvalósítandó projektötletek gyűjtésére. A felhívásban ki kell térni a beadandó anyag tartalmi követelményeire, valamint nyertes pályázat esetén a projektmunkában való részvétel feltételeire. Fontos, hogy csak olyan diákok pályázzanak ötlettel, akik nyerés esetén vállalják a következő félévben a projektben a megrendelő szerepének betöltését. A szerepkörökhöz tartozó feladatkörökről a

tanulmány későbbi fejezeteiben lesz szó. A pályázati szakmai anyag 3-5 oldalban a megvalósítandó ötlet koncepcionális leírását kell, hogy tartalmazza. Nem kell kitérnie a megvalósítás menetére, eszközeire, környezetére, mert azt majd a projektcsapat fogja kidolgozni a projekt indítása után.

3.3. A beérkezett koncepciók értékelése

A labor „ötletbörze" pályázatára beérkezett koncepciókat a következőképpen értékeljük:

- az összegyűlt projektötleteket az oktatók és a labor szervezői értékelik, és kiválasztják azt a néhány munkát, amelyeket továbbengednek a kiválasztás második körére.
- a kiválasztás második körében a néhány kiválasztott ötlet gazdáját meg kell hívni egy személyes találkozóra, aminek résztvevői a labormunkában érintett oktatók, és szervezők. A személyes elbeszélgetés a lehető legszűkebb körben kell, hogy történjen, hogy megőrizze a bizalmat ébresztő, támogató és motivációt erősítő hangulatát. Az ötletadó diákot ugyanis nem számon kérni kell, hanem megerősíteni benne a hitet és az elszántságot, hogy merje akarni vállalni az ötlenének kivitelezésében való aktív részvételt.

  A személyes találkozó további célja, hogy a labor szervezői felmérjék és megbizonyosodjanak arról, hogy az ötletgazda kellő határozottsággal, komolysággal és elszántsággal áll az ötlete mögött, és szeretné vállalni a megvalósítás kihívásait. (az egyetem lehetőségeit figyelembe véve sikeres projekt esetében megfelelő anyagi és / vagy természetbeni juttatással is lehet erősíteni az ötletadó elszántságát (pl. érdemjegy megajánlása adott tantárgyból, vagy szakdolgozati témának való elfogadás, stb.)).

3.4. Döntés a nyertes pályázat kiválasztásáról

Végül az interjúk lezajlása után a laborszervezők együttesen kiválasztják a nyertes ötletet. Amennyiben az egyetemnek van erőforrás kapacitása és infrastruktúrája ahhoz, hogy a laboron belül több, párhuzamosan futó projektet is vezessen, úgy több nyertes pályázatot is kiválaszthat. A nyertes pályázatok projektötleteit fogja a labor a következő (tavaszi) félévben megvalósítani.

Előfordulhat olyan eset, hogy több nyerésre esélyes ötlet érkezett be, mint amennyi megvalósítási projektet el lehet indítani. Ilyen esetben a következőképpen lehet eljárni:

- fejlesztői projekt csapat toborzása során (amit részletesen a következő pontban tárgyalunk) meghirdetünk több témát, mint amennyi projektet majd ténylegesen indítunk, és amelyikre a legtöbben jelentkeznek, az lesz ténylegesen megvalósítva.
- az elutasított projektötletek gazdáit arra ösztönözzük, hogy következő évben szintén pályázzák meg ugyanezzel az ötlettel a labor „ötletbörze" pályázatát, mert nagy eséllyel jövőre az ő projektötletük lehet a kiválasztott.
- a szűkített listán szereplő, nyerésesélyes projektötletek gazdáinak is fel lehet ajánlani bizonyos juttatásokat (pl. érdemjegy egy adott tantárgyból), hogy ne veszítsék el lelkesedésüket.

A nem nyertes pályázatok ötletgazdáinak is megvan továbbra is a lehetősége arra, hogy a labormunkára csapattagként jelentkezzenek és részt vegyenek a kiválasztott ötlet megvalósítási projektjében.

A nyertes projektötlet gazdája fogja betölteni a megvalósítás során a scrum módszertan szerinti „product owner" szerepét. (a szerepkörről a későbbi fejezetekben szólunk részletesen)

Projekt-csapat összeállítása

Térjünk vissza egy gondolattal a 4. fejezet elején tárgyalt négy alapfunkcióhoz, amely a legtöbb esetben piaci projektek esetében is szükséges a projekt indításához. Megnéztük, hogy hogyan értelmezhető e négy funkció az egyetemen működtetett projektlabor esetében. Döntöttünk tehát arról, hogy milyen feladat megvalósítását vállaljuk el, ami azt jelenti, hogy ha cég lennénk, nem maradna más hátra, mint megállapodni a megrendelővel a szerződés feltételeiben, megkötni a szerződést és elkezdeni a projektet. Szerencsére laborkörnyezetben nincs szükség ilyen jellegű megállapodásra és szerződéskötésre, viszont a laborkörnyezetnek van egy másik megoldandó specifikuma, amivel a cégek (jó esetben) nem kell, hogy számoljanak: fejlesztő csapatot kell toborozni.

A fejlesztőcsapat toborzást még a projektmunka megkezdése előtt, azaz még az első (őszi) félévben célszerű megtenni. A fejlesztő csapat toborzáshoz a következőket kell tennünk:

- egy felhívás keretében néhány mondatban ismertetni kell a diákok körében a megvalósítandó projektötletet, a labormunka céljait és juttatásait, majd meg kell határozni a jelentkezés határidejét.

  A felhívás ismertetéseként több különböző eszközt is választhat az egyetem, amelyeket szabadon kombinálhat: pl. elektronikus terjesztés eszközei (lev.listák, honlapok, fórumok) plakátok, vagy élőszavas tájékoztatók. A beadási határidő lejárata előtt szervezett vagy szervezetlen formában a labor szervezőinek lehetőséget kell biztosítani az érdeklődők számára konzultációs időpontra.
- a felhívásnak tartalmaznia kell, hogy pontosan milyen feladatok elvégzésére várjuk a diákok jelentkezését (azaz milyen szerepkörökre jelentkezhetnek, és szerepkörönként milyen elvárásokat támasztunk majd a jelentkezőkkel szemben). A megvalósítás során érintett szerepkörökre, a projektben résztvevők szerepkörönkénti számosságára és a szerepkörökhöz tartozó feladat- és felelőségi körök meghatározására a későbbi fejezetekben térünk ki részletesen.
- a jelentkezés során érdemes bekérni a jelentkezőktől egy néhány mondatban (fél oldalban) összefoglalt motivációs leírást, mert – főleg túljelentkezés esetében – ez nagymértékben segítséget jelenthet a csapat tagjainak kiválasztásában.
- a jelentkezési határidő lejáratát követően a labor szervezőknek ki kell választaniuk a jelentkezők közül a leendő projektcsapat tagjait. A kiválasztási döntés meghozatalát alátámasztó szempontrendszer helyi és eseti jellegzetességekkel rendelkezik.

  A kiválasztásnál figyelnünk kell arra, hogy a projektcsapat nagysága ne haladja meg az optimális létszámot (az optimális létszám kalkulációt a tanulmány korábbi fejezetében részleteztük). Ha túljelentkezés van, akkor a motivációs leírások alapján a labort szervező oktatók választanak csapattagokat. Aluljelentkezés esetén a szervezők eldönthetik, hogy elindítják-e a labort nem optimális létszámmal, vagy újraindítják a felhívást.

4. A fejlesztési módszertan bemutatása

4.1. A SCRUM módszer alapelvei

A laborműködtetés elsődleges céljának elérése érdekében, miszerint piaci körülményekhez legjobban hasonlító gyakorló környezetet szeretnénk kiépíteni diákjaink számára, úgy döntöttünk, hogy a labormunkára az ún. agilis fejlesztési módszert fogjuk bevezetni.

Az agilitás szó jelentése: fürgeség. Az agilitás olyan tulajdonság, amely képes változásokat létrehozni, illetve reagálni azokra. Az agilis szervezetek egyrészt fürgék és rugalmasak, másrészt képesek megtalálni a káosz és a rend közötti egyensúlyt. [5] Ezt a tulajdonságot felismerve napjainkban egyre több szervezet (főleg a kis és középméretű szoftverfejlesztő cégek) választja szoftverfejlesztési módszertanaként az agilis módszertanokat.

Az agilis szoftverfejlesztésre elsősorban mint értékrendszerre érdemes tekinteni. Ennek az értékrendszernek a lényege a gyorsaság, a változásra való reagálási képesség, az egyének és a csapat képességeibe és motivációjába vetett bizalom, a működő terméknek, mint a siker egyetlen mércéjének az elismerése. Az agilis szemlélet nem más, mint az értékrendbeli hangsúlyok erős megváltoztatása a vízesés szemlélethez képest. Amíg vízesés szemlélet kiindulási pontja, hogy az a csapat, amelyik jól kidolgozott eljárásokat, szabályokat és szervezeti felépítést követ, hatékony lesz, addig az agilis szemlélet abból indul ki, hogy ha a megfelelő emberekből összeállított team elé világos célokat tűzünk ki, és világos kereteket jelölünk ki számukra, akkor azok hatékony eljárásokat, szabályokat és szervezeti felépítést fognak kialakítani. A különbség tehát a kultúra és a csapat kialakításának sorrendjében van, nem pedig abban, hogy szükség van-e szabályokra. Az agilis szemlélethez igazodó modellek egy olyan keretrendszert definiálnak, amelyek azt írják elő a megvalósító csapatok számára, hogy tudatosan, és megállás nélkül vizsgálják felül saját működésüket, és a termékkel párhuzamosan saját szabályaikat és eljárásaikat is folyamatosan fejlesszék. Ezek a keretrendszerek nem a termék megvalósítására vonatkozó módszertanok, hanem olyan szabályok és szervezeti keretek, melyek az egyedi problémákra testreszabott eljárások kialakítására késztetik a megvalósító csapatot. [6]

Az agilis szoftverfejlesztési alapelvek mentén számos módszertani keretrendszer alakult ki. Ezek közül a legismertebb a Scrum, de vannak más érdekes irányzatok is pl. az eXtreme Programming, a DSDM, vagy a Kanban System for Lean Software Development. Tanulmányunkban a SCRUM módszer bevezetésének a lehetőségét vizsgáljuk.

A SCRUM egyik legfontosabb alapelve az, hogy felismeri és elfogadja, hogy a megrendelő a fejlesztés során meggondolhatja magát a követelményekkel kapcsolatban, és a váratlan változások nem kezelhetők könnyen a hagyományos, előzetes tervezési fázison alapuló módszerekkel. Ezért a scrum gyakorlati megközelítést választ, és elfogadja, hogy nincs lehetőség a probléma teljes megértésére és definiálására. Inkább azt próbálja maximálisan elősegíteni, hogy a csapat gyorsan meg tudja valósítani a funkciókat és gyorsan tudjon reagálni a változó követelményekre.

A SCRUM egy keretrendszer, amely magában foglal bizonyos tevékenységeket és meghatározott szerepeket. A SCRUM főbb szerepkörei a „Scrum Master”, aki a folyamatot felügyeli és munkája hasonlít a projektmenedzseréhez, a „Product Owner” aki a projektben érdekelt döntéshozókat képviseli, és a „Csapat” (Team), ami a nagyjából 7 főből áll és lefedi az összes munkafolyamatot.

Minden „futam” (sprint) során - amely 2 és 4 hét közötti időtartamot jelent (a csapat döntésétől függően) - a csapat egy működő szoftveregységet hoz létre. A futam során megvalósítandó funkciók a „Product Backlog”-ból (termék teendőlistája) kerülnek ki, ami az elvégzendő munka magas szintű követelményeiből álló, fontossági sorrendbe állított lista. Hogy a futam során a lista melyik elemei kerülnek megvalósításra, azt a futam elején tartott „futamtervező” megbeszélés során választják ki. A megbeszélés során a „Product Owner” közli a csapattal, hogy a teendők listájából melyek azok, amiket leghamarabb akar, hogy elkészüljenek. Ezután a csapat eldönti, hogy ezek közül melyek azok, amelyeket a következő futam során meg tud valósítani, és ezek megvalósítására ígéretet tesz. A futam folyamán a „futam teendőlistáját” nem lehet

megváltoztatni, a futam során elvégzett tevékenységek rögzítettek. Amint a futam a végéhez ért, a csapat bemutatja az elkészült funkciókat (demó). [7]

4.2. Kommunikáció a csapaton belül

A SCRUM módszer mozgatórugója a hatékony kommunikáció. Kis létszámú csapattal dolgozik, a csapatra egy összefüggő, kollaboráló egységként tekint, azaz nem az egyéneket, hanem a csapat egészét részesíti előnyben. Fontos eleme a csapattagok közötti folyamatos, és hatékony kommunikáció. A kommunikáció hatékonyságának növelése érdekében a módszer ajánlásokat fogalmaz meg a szervezett megbeszélések lebonyolításával kapcsolatosan, amelyek a következők:

Napi „stand up” megbeszélés

A futam során naponta tartott megbeszélés, lebonyolításából adódó sajátossága miatt stand-up megbeszélésnek (álló megbeszélésnek) is nevezik, és a következők vonatkoznak rá:

- mindig pontosan kezdődik, előre tervezetten, lehetőség szerint minden nap ugyanazon a helyen és ugyanabban az időben célszerű megtartani. Ennek oka, hogy a környezetváltozás nem vonja el a csapattagok figyelmét, koncentráltan és hatékonyan a megbeszélésre tudnak figyelni, nincs szükség akklimatizálódási időszakra. Továbbá a fix időpont tervezhetővé teszi a megbeszélést, ami egy idő után rutin tevékenységé válik.
- a megbeszélésen bárki részt vehet (megfigyelőként akár külső tag is), de csak a csapat operatív tagjai beszélnek.
- a megbeszélés ideje 15-20 percre korlátozott. Az időkorlát figyelése ösztönzi a csapattagokat arra, hogy csak a lényegről beszéljenek, néhány mondatba összefoglalva. (Ami nem fér bele ebbe a néhány mondatba, az bizonyára már nem is tartozik a lényeghez).
- a résztvevők általában állnak a megbeszélés során (innen ered a megbeszélés elnevezése). Ez elősegíti, hogy a megbeszélés ne húzódjon el.
- a megbeszélés során minden résztvevő, minden nap ugyanazt a három kérdést válaszolja meg:
    - Mit csináltam a tegnapi megbeszélés óta?
    - Mit fogok ma csinálni?
    - Van-e bármilyen akadály, ami hátráltat a napi célom elérésében? (a megoldáskeresés már nem ennek a megbeszélésnek a tárgya)

A napi megbeszélés célja, hogy a csapat tagjai összehangolják tevékenységüket. A megbeszélés időpontjára nincs ajánlott szabály, lehet napindító megbeszélésként tartani, ami abban az esetben nem szerencsés, ha a csapattagok nem ugyanabban az időpontban kezdik a munkát. Népszerű időpont még a státuszmegbeszélésre az ebéd utáni időszak. Egyes scrum-szakértők úgy vélik, hogy az emberek gyakran elpillednek az ebédtől, tehát egy élénk állómeeting felfrissítheti őket. Továbbá úgy gondolják, hogy mindenki dolgozott már ebéd előtt, ezért az emberek nem a privát dolgaikon gondolkoznak éppen, hanem a munkájukon.

Futamtervező megbeszélés (Sprint planning)

Minden futam (sprint) előtt futamtervező megbeszélést tart a csapat, amelynek célja:

- az elvégzendő feladatok kijelölése a termék teendőlistájáról (product backlog) a product owner (terméktulajdonos) közreműködésével,
- a sprint teendőlistájának előkészítése, amely a teljes csapat figyelembevételével részletezi az egyes részfeladatok időszükségleteit,
- annak meghatározása és kommunikálása, hogy mennyi feladat elvégzése várható el az aktuális sprint során,

A futamtervező megbeszélés 4 hetes futam esetén maximum 8 óra hosszúságú, rövidebb futam esetén ez az esemény is rövidebb.

Futamáttekintés (Demó vagy Sprint Review)

Sprint végén tartott megbeszélés, amelynek fő célja, hogy a csapat bemutassa az elkészült funkció(ka)t. A futamáttekintő megbeszélés jellemzői:

- a csapat áttekinti, hogy a sprint elején tartott sprint-tervező megbeszélésen bevállalt feladatok közül melyek készültek el, és melyek nem,
- az elkészült munkát bemutatják a terméktulajdonos és a fejlesztésben érdekeltek részére (demó).
- be nem fejezett munkát nem lehet bemutatni.

A futamáttekintő megbeszélés ajánlott hossza 4 hetes futam esetén maximum 4 óra hosszúságú, rövidebb futam esetén ez az esemény is rövidebb.

Visszatekintés (Retrospective)

A visszatekintés időszakonként (nem feltétlenül minden sprint után) tartott megbeszélés arra vonatkozóan, hogy a csapattagok megfogalmazhassák véleményüket a közös munkával kapcsolatosan. A visszatekintő megbeszélésen

- a csapattagok véleményt alkotnak az elmúlt sprint(ek)ről. A vélemény lehet egy puszta benyomás is, nem kell kidolgozott, szilárd álláspontnak lennie.
- javaslatokat tesznek a folyamatok továbbfejlesztésére. A javaslatoknak nem kell kiérleltnek lenniük, a kidolgozás nem a visszatekintés része.
- A megbeszélésen minden csapattag a következő három kérdést válaszolja meg:
    - Mit kell megtartani és alkalmazni a következő sprintekben is
    - Mit kell – lehetőség szerint – elhagyni a következő sprintekben
    - Mit lehetne esetleg kipróbálni a következő sprintekben a jobb működés elérése érdekében

4 hetes sprint esetén maximum 3 óra hosszúságú, rövidebb futam esetén ez az esemény is rövidebb.

4.3. Adatok és eszközök

Az agilis fejlesztés a működő szoftvert előnyben részesíti az átfogó dokumentációval szemben. [8] A fejlesztésben az információátadás azonban nem tud hatékonyan működni megfelelő dokumentáció nélkül. A módszer azt javasolja, hogy az optimális információátadáshoz és áttekintő szervezéshez szükséges minimális dokumentációt kell elkészíteni, mindig az optimális időpontban. Az optimális időpont azt jelenti, hogy nem utólag, asztalfióknak készítünk dokumentációt, amit már senki semmire sem fog használni, és nem is jóval korábban, mert

ekkor nagyobb a kockázata annak, hogy a folyamatosan változó körülmények miatt a felhasználás pillanatában már nem aktuális információkat tartalmazna, hanem pontosan jó időben: kivárjuk az időpontot, amikor már biztonsággal eléggé sokat tudunk a környezetről és a dokumentálandó funkcióról, de figyelünk arra is, hogy a dokumentáció elkészítésének idejét is belekalkuláljuk, hogy a dokumentáció pontosan akkorra készüljön el, amikor a felhasználása megkezdődne. [9]

A SCRUM módszer három típusú dokumentációt tart hasznosnak. Ezek a következők:

- Termék teendőlistája (Product Backlog)

  A Product Backlog az egész termékre vonatkozóan tartalmaz magas szintű követelmény-leírásokat. A lista elemei lehetnek üzleti funkciók leírásai, kívánságok, ötletek, stb., amelyek prioritás szerint vannak rendezve. Tulajdonképpen azt tartalmazza, hogy mi a fejlesztés célja. A lista szabadon szerkeszthető bárki által, és becsléseket tartalmaz a bejegyzések üzleti értékére és ráfordításigényére vonatkozóan. A becslések abban segítik a terméktulajdonost, hogy meghatározza a bejegyzések sorrendjét és bizonyos mértékig a prioritásukat. Ha két funkciónak azonos az üzleti értéke, akkor a kevesebb fejlesztési ráfordítást igénylő funkció fog magasabb prioritást kapni, hiszen annak jobb a megtérülése. A termék teendőlistája a terméktulajdonos kezelése alatt áll. Az üzleti értéket a terméktulajdonos, míg a fejlesztési ráfordításigényt a fejlesztők határozzák meg.

- Futam teendőlistája (Sprint Backlog)

  A Sprint Backlog olyan dokumentum, amely azt tartalmazza, hogy a csapat hogyan fogja elkészíteni a sprint során megvalósítandó funkciókat. Ez egyes funkciókat részfeladatokra bontják. A felbontást célszerű úgy elvégezni, hogy egy részfeladat 4-16 óráig tartson. Ilyen szintű részletezettség mellett a csapat összes tagja pontosan érti, hogy mit kell elvégezni, és mindenki kiválaszthatja a neki legmegfelelőbb részfeladatot. A futam teendőlistájában levő részfeladatokat nem rendelik személyhez, inkább a csapattagok választják ki azokat a meghatározott prioritások, szükségletek és a csapattag képességeinek megfelelően.

  A futam teendőlistája a csapat kezelése alatt áll. A becsléseket a csapattagok adják meg. Az áttekinthetőség kedvéért érdemes a sprint során valamilyen feladatkövető (ún. ticketing) rendszert alkalmazni, amely arra használható, hogy a feladatokat, azok prioritását, erőforrásigényét és státuszát („elvégzendő”, „folyamatban”, „elvégzett”, stb.) megterhelő erőfeszítés nélkül a csapat naprakészen vezetni tudja.

- Burn down chart (egyfajta napi eredmény-kimutatás)

  Mindenki számára elérhető grafikon, amely mutatja a sprint teendőinek a listájából hátralevő munka mennyiségét. Minden nap frissítik, egyszerű módon jeleníti meg a futam állapotát.

4.4. Szerepkörök a csapatban

A SCRUM módszer az alábbi szereplőket különbözteti meg a fejlesztés során [8]:

- Terméktulajdonos (Product Owner)

  A Product Owner a megrendelőt képviseli a fejlesztési csapatban. Biztosítja, hogy a csapat az üzleti szempontból fontos dolgokkal foglalkozzon. Fő feladata a termék-teendőlista bővítése a megrendelő szempontjából megfogalmazott bejegyzésekkel,

majd prioritásokkal való ellátása, továbbá a sprint backlog-ban szereplő üzleti funkciók előkészítése fejlesztésre.

- Scrum Master

  Elsődleges feladata hogy elhárítsa az akadályokat, amelyek gátolják a csapatot abban, hogy a sprint célját megvalósítsa. A scrum master nem a csapat vezetője, (a csapat önszerveződő), hanem a csapat és a zavaró tényezők közötti lökhárító. Ügyel arra, hogy a scrum folyamatot megfelelően alkalmazzák. Ő tartatja be a scrum szabályait. Kulcsfontosságú feladatának számít a csapat védelme és annak biztosítása, hogy a csapat az elvégzendő feladatokra koncentráljon. A scrum master tartja mozgásban a csapatot (facilitator), jelentős szerepet vállal az események lebonyolításában.

- Csapat (Scrum Team)

  A csapat azért felelős hogy a termék elkészüljön. Általában 5-9 főből áll. A csapattagok különféle képességei lehetővé teszik, hogy a feladatot közösen megoldják (fejlesztő, dizájner, tesztelő, stb.)

- Felhasználók

  A felhasználó a szoftver leendő használói. (A sok egyedi vásárló számára készülő szoftverek esetén a felhasználó hangját megtestesítő személy.)

- Üzleti szereplők

  Azok az emberek, akik lehetővé teszik a projekt létrehozását, és akiknek a termék hasznot fog hozni. Közvetlenül csak a futam áttekintő megbeszélésen (demó) vesznek részt a folyamatban. Ez az adminisztrátorokat és az igazgatókat is magába foglalja.

- Menedzserek

  A fejlesztésben résztvevő szervezeti egységek munkakörnyezetét teremtik meg. (Nem csak a funkcionális vezetőket jelenti.)

5. Kezdődjön a munka

Az előző fejezetben áttekintettük a SCRUM módszer alapelemeit, meghatároztunk azokat a követelményeket, amelyek szükségesek a módszer gyakorlatban való használatához. Összefoglalva szükségünk van egy megrendelésre (másként projektötletre), egy csapatra (amely különböző szerepköröket betöltő személyekből áll) és fejlesztőkörnyezetre (azaz térben, időben és eszközökben megfelelő környezetre).

A megrendelő, a projektötlet és a csapat kiválasztásának folyamatát áttekintettük az előző fejezetekben. Nem tértünk ki azonban a csapat összetételére, azaz a SCRUM által javasolt szerepkörök megosztására a csapaton belül.

5.1. A csapat összetétele

A SCRUM módszer három szerepkört határoz meg, ami szorosan a csapattagokra vonatkozik: a product owner, scrum master és a csapat. A további három javasolt szerepkör a projekt külső, befolyásoló környezetére vonatkozik, azaz azt határozza meg, hogy milyen szerepkörök lehetnek hatással a külső környezetből a csapat működésére.

Tekintsük át a csapatunk tagjait:

- szükség van egy ún. product owner-re (termékgazdára). Ezt a szerepkört – az előző fejezetekben részletezett – megrendelő tölti be. A termékgazda feladata, hogy képviselje üzletileg a termékkel kapcsolatos funkciókat a fejlesztők felé. Ő a megrendelő és a leendő felhasználók szóvivője a fejlesztők felé. Meghatározza a

fejlesztendő funkciókat (részletesen, logikai specifikáció szintig), tervet készít a fejlesztők számára a fejlesztendő funkciókról, priorizálja a fejlesztési igényeket, majd az elkészült terméket üzletileg teszteli. A product owner tesztelésének nem a hibák felfedése a célja, hanem annak meghatározása, hogy ténylegesen az a termék / termékfunkció készült-e el, amit a megrendelő várt, azaz a logikai összefüggéseket nézi meg az elkészült terméken.

A laborkörnyezetre visszatérve: a product owner egy nagyon speciális kulcsszerepet tölt be a fejlesztés során, mert értenie kell a fejlesztők nyelvén, specifikálnia kell tudni, de ugyanekkora gondossággal és alapossággal kell értenie a leendő felhasználókat. Sokkal ügyfél-orientáltabbnak kell lennie, mint a többi csapattagnak. Ez volt az oka annak, hogy megfelelően nagy gondot kell fordítani a megfelelő product owner (ötletgazda) kiválasztásának, mert nagymértékben rajta múlik a projekt sikere.

- scrum master feladatot – a módszertan szerint - elláthat bárki a csapatból, sőt akár nem is szükséges, hogy mindig ugyanaz az ember lássa el. Mielőtt ezt a szerepkört levetítjük a laborunkra, nézzük meg, hogy milyen feladatokkal és milyen tulajdonságokkal kell, hogy rendelkezzen egy scrum master. A scrum master alapvető feladata a csapattagok segítése, a felmerülő akadályok elhárítása, a zavartalan munka biztosítása. Továbbá feladata a scrum kereteinek betartatása a csapaton belül, ő figyel arra, hogy a működéshez szükséges kommunikációs és egyéb eszközök rendeltetésszerűen legyenek alkalmazva. Ő a csapat mozgatórugója, valamint a kereteken belüli tartója. Nagyfokú szervezőképességgel, jártassággal, tapasztalattal és jó kommunikációs képességgel kell, hogy rendelkezzen. Ezeket a feladatokat figyelembe véve nem célszerű, hogy a scrum master szerepet diákra osszuk ki, sőt még kevésbé sem célszerű Bsc-s, Msc-s diákot megbízni ezzel a feladattal. Javasoljuk, hogy a laborprojektekben a scrum master szerepét a labort szervező oktató lássa el.
- a csapatot a scrum módszertan egybefüggő, egyenrangú, homogén egészként tekinti. Ám nyilvánvaló, hogy a különböző fejlesztési feladatok ellátására különböző szakemberekre van szükség, pl. web-fejlesztésre, hálózatfejlesztésre, back-end és adatbázis fejlesztésre vagy akár adattárház feladatokra nem szerencsés ugyanazt a programozót megkérni, mivel az egyes emberek tudása és érdeklődési köre eltér egymástól. Továbbá a csapat szerves része a tesztelő is. Nézzük, hogyan tudjuk ezt alkalmazni a laborunkban: miután kiválasztottuk a projektötletet, a labort szervező oktatóknak van már valamilyen elképzelése arról, hogy a megvalósítás milyen szakterületeket érint. A csapattagok toborzását tehát szakterület szerint célirányosan kell elvégezni, azaz a toborzási felhívásban meg kell határozni, hogy milyen érdeklődési körökkel rendelkező diákok jelentkezését várjuk.

  Az érdeklődési terület mellett fontos azonban a tudásmélység is, azaz nem mindegy, hogy egy adott szakterületen kezdő, haladó vagy profi tudással rendelkezik az adott csapattag. A csapat összeállításánál figyelni kell arra, hogy mindenképpen különböző tudásmélységű szakembereket válasszunk. Nem vezet a projekt sikerére az, ha csak kezdőkkel próbálunk dolgoztatni, de az sem, ha csak profikkal. Kell néhány szakmai vezető profi, de természetesen kellenek a kezdők is, hiszen pontosan ez is célja a labornak, hogy szakmai tudást is mélyítsen el. Javasoljuk, hogy szakmai vezetőket Phd hallgatók köréből, vagy hiányukban mélyebb tudással rendelkező Msc-s hallgatók köréből jelöljünk ki, és egy projektcsapat legalább – szakterületenként- egy szakmai vezetővel rendelkezzen.

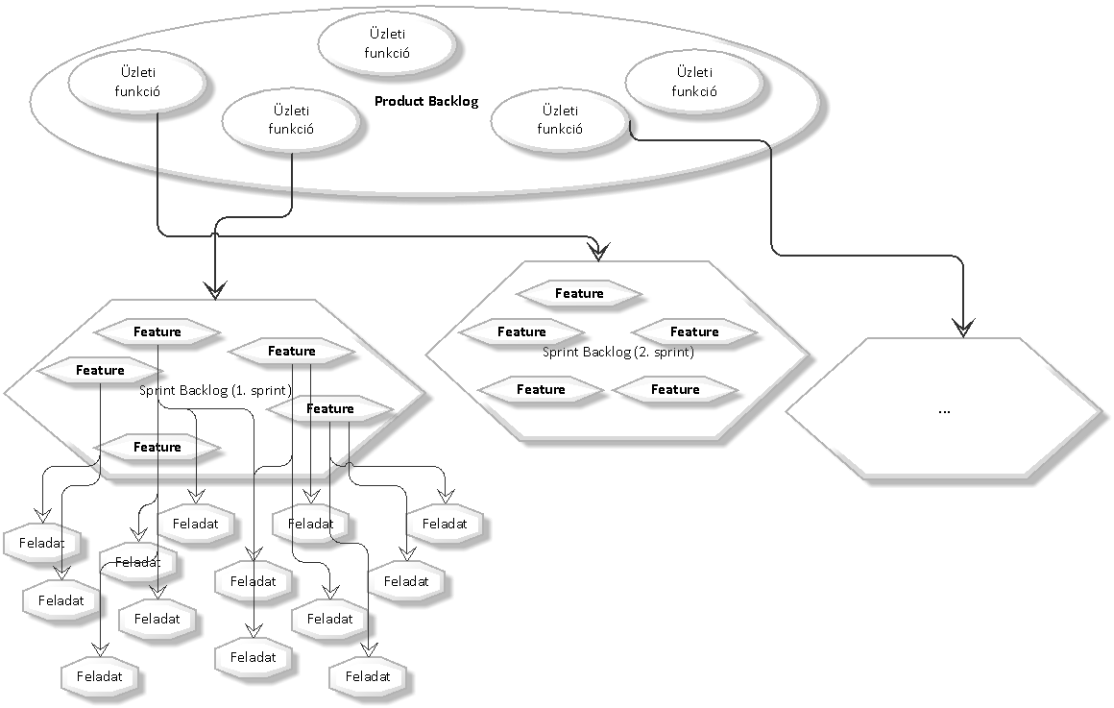
  Fontos kérdés a csapaton belül a tesztelő szerepkörének kijelölése. A toborzási felhívásban külön kitérhetünk a tesztelőkre, azaz fogadhatunk olyan diákok jelentkezését is a laborra, akik leginkább a tesztelés, azon belül az automatizált tesztelés

felé orientálódik. Az automatizált tesztelés a SCRUM fontos mérföldköve, szükséges, de nem elégséges eszköze a tesztelésnek. Ha nem akarunk ennyire diverzifikálni csapaton belül, akkor mondhatjuk azt, hogy minden fejlesztő egyben tesztelő is. Fontos azonban, hogy ilyenkor a több szem többet lát elven alapulva a fejlesztők egymás kódrészleteit is teszteljék, ne csak a sajátjukat. Saját kódjuk hatékony tesztelésére alkalmazhatjuk a TDD (Test Driven Development) technikát, amely a gyakorlatban nagyon hatékony eszköznek bizonyult scrum-ot használó fejlesztési környezetben.

5.2. Projektadminisztráció

A hatékony munkavégzés alappillére a strukturált és jól definiált rend. A SCRUM módszer ugyan nem a dokumentáltságot helyezi előtérbe, a módszer sokkal inkább a jó minőségben működő szoftverre teszi a hangsúlyt. Ahhoz azonban, hogy megfelelő minőségű szoftvert tudjunk hatékonyan előállítani, fontos rendszerezni, priorizálni a fejlesztési feladatokat. Ezeket célirányos, hatékonyan kezelhető, és könnyen naprakészen tartható tervdokumentumban kell definiálni.

A SCRUM szerint a tervezés felülről lefelé építkezik. Ez azt jelenti, hogy a projekt elején meghatározzuk a Product Backlog-ot, azaz a főbb üzleti funkciókat, amelyeket szükséges megvalósítani a projekt során. Ezeket az üzleti funkciókat fontossági sorrendbe helyezzük, és kezdjük a munkát a prioritási sorrend szerinti elsővel.



5. ábra Feladatok lebontása

A sprint során megvalósítani kívánt üzleti funkciót ún. feature-kre bontjuk. Egy feature egy olyan minimális üzleti egységfeladatot jelent, amelyet már nem lehet tovább bontani ahhoz, hogy továbbra is önállóan életképes üzleti funkció szerepet ellásson. A sprint backlog feature-k halmazát jelenti, amelyeket meg kell valósítani a kiválasztott üzleti funkció megvalósításához.

Természetesen a feature-k között is vannak elengedhetetlenül fontosak, és kevésbé fontosak is, ezért a sprint elején célszerű a feature-ket priorizálni, a fejlesztést pedig a prioritási sorrend alapján a legnagyobb prioritásúakkal kezdve lefolytatni. Ez azért fontos, mert ha a sprint végén esetleges időhiány miatt kimaradnak funkciók a sprintből, akkor lehetőleg a kevésbé fontos funkciók maradjanak le. Két egyformán fontos funkció között a megvalósítás időbecslése dönt: amelyiket rövidebb idő alatt meg lehet valósítani, azt kell előnyben részesíteni, mert annak az ár-érték aránya lesz magasabb.

A feature-ket a fejlesztők rendszerint tovább bontják, fejlesztési feladatokat készítenek belőle. Egy fejlesztési feladat egy jól definiált, max. 4-16 óra alatt elvégezhető feladat, amely egy feature építőkövének tekinthető. A fejlesztési feladatok rendszerint üzletileg nem alkotnak önálló egységeket, hanem az üzleti funkciók tégláinak, építőelemeinek tekinthetők. Törekedni kell arra, hogy a szükséges összes téglát megfelelő gondossággal, és megfelelő időben építsük be, mert különben instabil lesz az építmény.

Az előző ábra is jól szemlélteti, hogy a felülről lefelé való tervezés, majd alulról felfelé való építkezés egy idő után nagy mennyiségű feladat adminisztrációval jár. Minden egyes feladatról számon kell tartanunk minimum a következőket:

- mi a feladat (az előbbi hasonlatunkkal élve: milyen téglára van szükségünk)
- mi a feladat célja, miért és hol van rá szükség (hova kell beépíteni a téglát)
- mennyire fontos a feladat (ha esetleg kimaradna a tégla, mekkora instabilitáshoz vezetne)
- mennyi idő alatt készíthető el
- ki dolgozik rajta
- mi az állapota (új, fejlesztése folyamatba, tesztelhető, teszt sikeres vagy teszt sikertelen)
- melyik sprintbe van besorolva

Továbbá az elengedhetetlenül szükséges információkon kívül felmerülnek még egyéb információk is, amelyeket bizonyos körülmények között szeretnénk ismerni az adott feladatról. Pl.:

- ki és mikor hozta létre, ki és mikor kezdeményezte a feladatot
- milyen típusú a feladat (pl. működési probléma, vagy esetleg új igény)
- milyen kategóriába sorolható (pl. riport, vagy front-end fejlesztés, stb.)
- van-e csatolmánya (pl. hibaüzenet, vagy pillanatkép a felületről, vagy riport terv, stb.)
- ki és mikor végezte el a feladatot, mennyi időbe tellett elvégezni
- ki és mikor tesztelte
- melyik verzióval került élesbe

Láthatóan ez rengeteg információ, amit a hatékony működés érdekében naponta kell frissíteni, és karbantartani. Továbbá ezekből az információkból nyerhetők ki a haladást jól szemléltető termelékenységi mutatók, másként a burndown diagramok. A csapatnak minden nap tisztában kell lennie ezeknek a mutatóknak az alakulásával, mert ez alapján követhető az előrehaladás üteme, illetve befolyásolható a napi feladat kiosztása és vállalása.

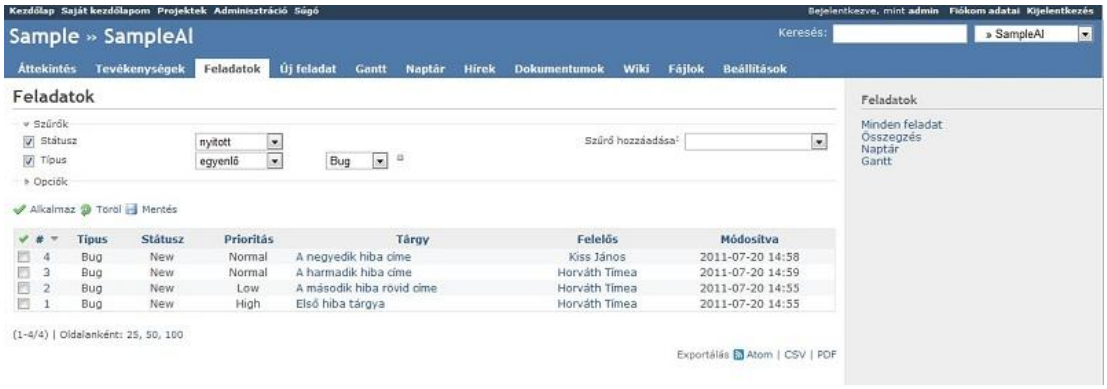
Fontos azonban, hogy a projektadminisztrációra fordított idő a szükséges minimumra legyen csökkentve, hiszen egyébként értékes fejlesztési órákat venne el a szakértőktől. Ezért nagy

gondot kell fordítani arra, hogy megválasszuk azt a legalkalmasabb eszközt, amiben könnyen és gyorsan, naprakészen vezetni tudjuk majd a feladataink alakulását.

SCRUM-ot használó fejlesztőcégek körében elterjedt a Redmine [10] nevezetű rendszer használata.

A Redmine egy open source, Ruby nyelvet használó, web alapú projektmenedzsment eszköz (http://www.redmine.org/). Egyszerű, könnyen átlátható, jól testre szabható rendszer. A Redmine párhuzamosan több, egymástól elszeparált projektet képes kezelni. A projektek hierarchiába szervezhetőek tetszőleges mélységben, tehát van lehetőség alprojektek kialakítására. A projektekhez rendelhetünk felhasználókat, meghatározott szerepkörrel. A szerepkörök határozzák meg, hogy mihez van létrehozási, szerkesztési, módosítási, törlési, hozzászólási jog. Az alap szerepköröket bővíthetjük, módosíthatjuk (Adminisztráció/Szerepkörök és jogosultságok). A szerepkörök beállítása jogosultsági mátrixszal történik, azaz modul szinten állíthatók (ki-be kapcsolhatók) a hozzáférési jogok (pl. Projekt modulhoz: projektek szerkesztése, projekt kezelése, tagok kezelése, verziók kezelése).

Publikus vagy privát projekteket hozhatunk létre. Publikus projektre bárkinek rálátása van, a privát projektekhez csak a projektre felvett felhasználóknak. A rendszerben állítható, hogy bárki regisztrálhat-e, vagy csak az adminisztrátornak legyen joga létrehozni felhasználókat.



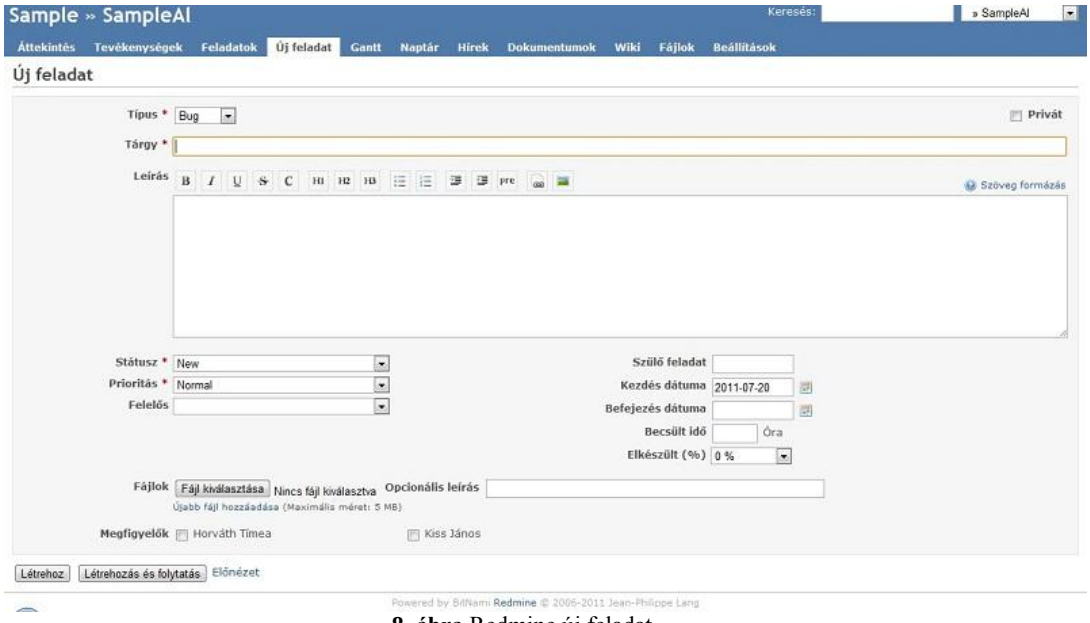
6. ábra Redmine feladatok

A workflow-val szabályozhatjuk, hogy a különböző típusú bejegyzéseink (hiba, módosítási kérelem, support, task...) milyen életutakat járhatnak be. Minden bejegyzés típushoz szerepkörönként adhatjuk meg, melyik státuszból melyik státuszba állítható egy bejegyzés. A workflow meghatározásánál, szerepkörön belül megkülönböztethetjük a "szerző" és a "hozzárendelt" felhasználókat is.

7. ábra Redmine workflow

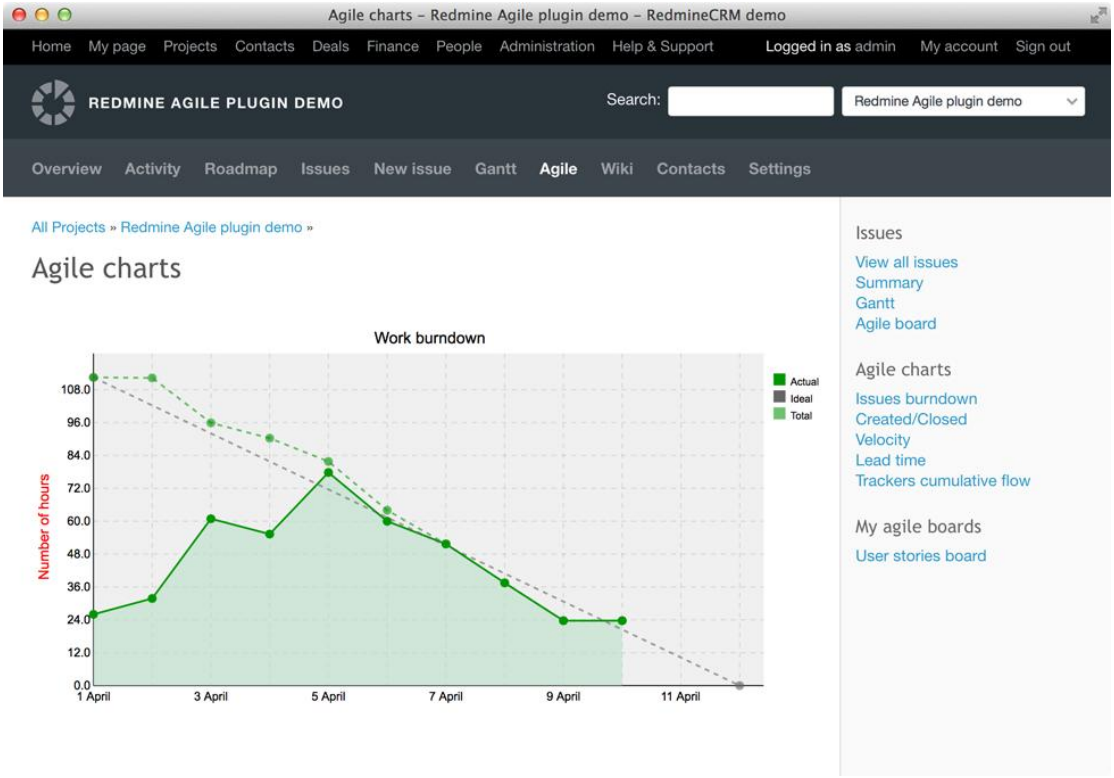
A Redmine-ba tehát egyszerűen adminisztrálhatjuk a Feature-inket, feladatainkat, sprintekbe (release-kbe) szervezhetjük őket, és – amint az alábbi képen is látszik – tárolhatjuk benne a számunkra fontos paramétereket, mint pl. a bejegyzés típusa, megnevezése, leírása, státusza, prioritása, felelőse, stb. Csatolmányokat is feltölthetünk, és nem utolsó sorban a lekérdező felületen keresztül bármilyen megadott szempont alapján (csoportosított, szűrt, rendezett) lekérdezéseket definiálhatunk, amiket el is menthetünk saját vagy a projektcsapat felhasználása érdekében.

A rendszer továbbá alkalmas összesítő erőforrás-lekérdezések listázására is, azaz egyszerűen képet kaphatunk arról, hogy a projektünkre összesen, vagy fejlesztőnként hány munkaórát fordítottak már. A találati listákat exportálhatjuk pdf vagy csv formátumba. [11]



8. ábra Redmine új feladat

A Redmine-hoz különböző bővítményeket (plugin) tölthetünk le, amelyek közül a SCRUM módszer használóinak talán a legfontosabb a diagram-megjelenítő nézet, közöttük is a burndown diagram (részletes bemutatását ld. előző fejezetekben). [12]



9. ábra Redmine diagram

5.3. Az elmélet alkalmazása a laborgyakorlatban

A SCRUM módszer azt javasolja, hogy 2-3 hetes időintervallumokra tervezzünk sprinteket, azaz egy sprint ne haladja meg a 3 hetet. A sprint alatt minimum egy önálló üzleti funkciónak kell elkészülnie. A tanulmány elején láttuk, hogy várhatóan 50 funkciópont körüli projektméret fér bele a labor egy szemeszteres munkájába, ami azt jelenti, hogy 4-5 üzleti funkcióval tudunk kalkulálni. Ha 3 hetes sprintekben gondolkozunk (ami 3 x 6 munkaórát jelent a csapatnak), akkor egy szemeszternyi idő alatt sprintenként egy funkcióval reálisan el tudjuk végezni a munkát.

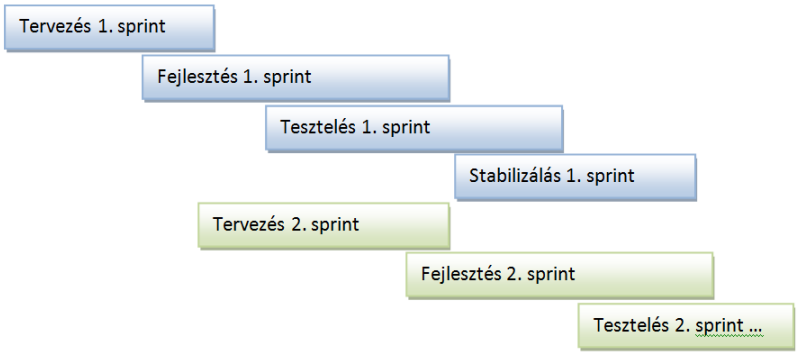
Első feladatunk tehát meghatározni azt az 5-6 funkciót, amiknek megvalósítását célul tűzzük ki. A projekt funkciókra bontását a termékgazda a labor oktatóival és a labor vezető fejlesztő(i)vel közösen végzik.

Miután meghatároztuk az elvégzendő funkciók listáját, prior0izálni kell a funkciókat. Ez a termékfelelős feladata. Amennyiben két funkció is ugyanazt a prioritást kapta, úgy a vezető fejlesztő(k) dönt(enek) annak alapján, hogy melyik funkciónak a megvalósítása kerül várhatóan kevesebb időbe. Az elkészült egyértelműen rangsorolt listából válasszuk ki a legnagyobb prioritással rendelkezőt. Minden sprint elején ugyanígy fogunk eljárni: a maradék – még el nem

végzett – funkciók listájából kiválasztjuk a legnagyobb prioritással rendelkezőt. Természetesen, mivel a módszertan a változáskezelés magas szintű támogatásáról híres, megengedhető, hogy két sprint között a maradék funkciólista prioritási sorrendjén változtasson a termékfelelős.

Minden sprint egy 3 hétig tartó, mini vízesés modellnek feleltethető meg. Azaz a három héten belül egyértelműen elkülönülnek a tervezés, fejlesztés, tesztelés, stabilizálás fázisok. A tervezés és a stabilizálás a SCRUM szerint nem tartoznak szorosan a sprinthez, viszont szükségszerűen elvégzendő, így emiatt ütemezendő feladatok ezek is, tehát meg kell találni ezek helyét is a fejlesztésben. A legszerencsésebb, ha humán erőforrását tekintve a tervezés és a stabilizálás leválasztható a sprintről, azaz ezeket a funkciókat más személyek látják el. A tervezés rendszerint a termékfelelős és a vezető fejlesztő közös munkája, ami egyértelműbben leválasztható és párhuzamosítható a sprinttel, mint a stabilizálás, amely rendszerint a tesztelő, a termékfelelős és legalább egy fejlesztő közös munkája.

Célszerű a sprinteket úgy szervezni, hogy a sprinten belüli fázisok egymáshoz képest a következőképpen helyezkedjenek el:



10. ábra Sprintek ütemezése

Tervezés

Felelős: product owner (termékfelelős)

Résztvevők: product owner; vezető fejlesztő(k); megrendelő(k), ha vannak

Elkészült termék: az első sprintben megvalósítani kívánt funkció logikai specifikációja

Tipp, jó tanács: a kényelmes munkatempó megtartása miatt célszerű, ha a product owner 1-2 sprinttel meghaladja a fejlesztőket, azaz előre dolgozik. Fontos, hogy a túlzottan előre dolgozás nem előnyös, mert nem szabad elfelejtenünk, hogy változó környezetben vagyunk, ahol ha túlzottan előre elkészítjük a terveket, mire a megvalósításhoz érünk, a környezet, az elvárások már annyira megváltozhattak, hogy a tervek aktualitásukat vesztik, és kezdhetjük elölről a tervezést.

Ha viszont egy kicsit sem dolgozunk előre, fennáll a veszélye annak, hogy a programozók utolérik a tervezőt, azaz nem lesznek előkészített tervek, amin a fejlesztők dolgozhatnak, ami miatt időt vesztünk. A hatékony és termelékeny munka alapelve, hogy a munkamennyiség lehetőség szerint egyenletes legyen.

Teendők: a nagyrészt elkészült tervek alapján megtartható a sprintnyitó megbeszélés. Ezen a megbeszélésen a product owner élőszóban ismerteti a fejlesztőcsapattal az adott sprintre tervezett feladatokat. A fejlesztőknek lehetőségük van kérdezni, tisztázni a termékfelelőssel

a feladat elvégzéséhez szükséges üzleti (háttér)információkat. Cél, hogy a fejlesztők megértsék a feladatot, illetve ahol bizonytalanságot, vagy további mélyebb specifikációs igényt éreznek, jelezzék azt a tervezőnek. Nem célja a megbeszélésnek a megvalósítás technikai részleteinek az átbeszélése, illetve a feladatkiosztás sem.

A sprintindító megbeszélés következő mérföldpontja a megtervezett funkció feature-kre, majd a feature-k fejlesztési feladatokra való bontása. Ezeket a feature-ket, és feladatokat célszerű valamilyen adminisztrációs rendszerbe felvezetni (ld. 5. fejezet).

A fejlesztők, a vezető fejlesztővel az élen becslést mondanak a feladatok megvalósítási időtartamára vonatkozóan. Ez történhet úgy is, hogy minden fejlesztő becsül, majd az így kapott becslések súlyozott átlagát kiszámítják, és az így kapott érték lesz a feladat becsült megvalósítási ideje. Ezt viszonylag időigényes feladat, viszont cserében minden fejlesztő véleményét figyelembe veszi (a tapasztaltabbakét nagyobb súllyal). A becslés elvégezhető úgy is, hogy a csapat tapasztaltabb tagjai (rendszerint a vezető fejlesztő(k)) becsülnek. Ez várhatóan pontatlanabb, de gyorsabb eredményhez vezet.

A feladatok megbecsült átfutási időit feature-k szintjére kell összegezni, majd a feature-ket üzleti hasznosság szempontok alapján rangsorolni, priorizálni kell.

Fejlesztés

Felelős: vezető fejlesztő (ha több, nagyobb tapasztalattal rendelkező fejlesztő is van a csapatban, akkor valakit – a labor szervező oktatói segítségével – vezető fejlesztőnek kell titulálni. Bár a SCRUM módszertan nem követeli meg az ilyen fajta diverzifikációt, a gyakorlatban hasznosnak bizonyul, ha egy feladatnak egyszemélyes felelőse van. Ez természetesen nem azt jelenti, hogy a felelősnek kell elvégeznie a feladatot, hanem azt, hogy az ő felelősége mozgósítani úgy a csapatot, hogy a fázistermék a megadott határidőre elkészüljön.

Résztvevők: vezető fejlesztő, fejlesztők, product owner, scrum master

Elkészült termék: a sprint scope-ját alkotó üzleti funkció bemutatható és éles használatra (legrosszabb esetben felhasználói átvételi tesztelésre) átadható változata

Tipp, jó tanács: a gyakorlatban számtalanszor feltevődik a kérdés, hogy melyik fejlesztő melyik feladattal foglalkozzon. A feladatkiosztás kétféleképpen is történhet: a SCRUM módszer szerint önszerveződő módon történik, azaz minden fejlesztő választ egy feladatot magának aszerint, hogy mi a soron következő legnagyobb prioritású feladat, amit meg tud (hatékonyan) oldani. Ez azt jelenti, hogy mivel a fejlesztők különböző érdeklődési körrel rendelkező, más-más szakterület képviselői, így lehet, hogy nem a soron következő legnagyobb prioritású feladat lesz az, amit ő hatékonyan meg tud oldani, hanem mondjuk a harmadik legfontosabb. Ilyenkor neki ezt kell választania, a többit – hiába azok magasabb prioritásúak – meg kell hagynia más fejlesztőnek, aki jobban ért az adott feladat megoldásához szükséges szakterülethez.

A feladatkiosztás történhet úgy is, hogy a vezető fejlesztő minden munkakezdéskor áttekinti, hogy ki hogyan áll a saját feladatával, és akinek elfogyott, vagy hamarosan el fog fogyni a dolga, annak kijelöl egy soron következő feladatot az üzleti prioritások és a fejlesztő szakértelmének figyelembevételével. Ez a centralizált feladatkiosztás abban az esetben működik jól, ha a vezető fejlesztő megfelelően tapasztalt a feladatok szakmai hozzáértésigényét tekintve és ismeri jól a csapatát, vagy ha a fejlesztők még teljesen kezdők és nem tudják eldönteni, hogy melyek azok a feladattípusok, amelyekben ők hatékonyan részt tudnak majd venni.

Fontos, hogy a feladatokat ne osszuk szét maradéktalanul a sprint elején, hanem mindig egyesével „adagoljuk”. Csak abban az esetben delegáljunk egynél több feladatot egyszerre egy fejlesztőre, ha azok olyan feladatok, amelyeket senki más a csapatból nem tudna megoldani.

Fontos továbbá, hogy a fejlesztők egy légtérben, egymáshoz fizikailag közel helyezkedjenek el. A gyakorlatban bizonyított tényező, hogy az egy légtérben tartózkodó személyek között sokkal hatékonyabb a kommunikáció. A SCRUM fejlesztés mozgatórugója a kommunikáció, ezért fontos, hogy olyan közeget teremtsünk, amelyikben elősegítjük hatékonyságát. A product owner jelenléte is elengedhetetlenül fontos a fejlesztés alatt is. Bár látszólag neki ilyenkor nincs semmi dolga a sprinten, ez nem így van. Minekután a SCRUM nem a dokumentációkészítés alaposságáról híres, így nagyon fontos, hogy amikor a fejlesztők esetlegesen üzleti kérdés miatt elakadnak, kéznél legyen a válasz. Sokkal rosszabb minőségű végtermék keletkezne, ha ilyenkor magára hagyjuk a fejlesztőket, mert ahhoz, hogy tovább tudjanak haladni, hajlamosak ők maguk megválaszolni ezeket a kérdéseket, amelyek nem minden esetben egyeznek a megrendelő gondolataival (sőt rendszerint nem egyeznek).

A SCRUM módszer legjellemzőbb eleme a napi stand-up megbeszélések. A gyakorlatban nagyon hasznos ezt a kb. negyed órás megbeszélést megtartani, mert ezáltal a csapattagok folyamatos naprakész információt kapnak egymásról, problémáikat, a munkában esetleges akadályozó tényezőiket elmondhatják, tapasztalataikat megoszthatják egymással. Ez élteti a csapatműködést, és hatékonyan viszi előre a projektet. A scrum megbeszéléseken kulcsszerep jut a scrum masternek, ugyani az ő feladata a megbeszélést úgy irányítani, hogy mindig célorientált, lényegre törő legyen. A problémákat nem ezen a megbeszélésen kell megoldani, ez csak arra való, hogy felvessük őket, és a megoldásukon ezt követően már nemcsak egyedül, hanem a csapat erejével dolgozzunk a következőkben.

Tesztelés

Felelős: a tesztelési fázis felelőse a tesztelő. Ha a laborban úgy válogattuk össze a résztvevőket, hogy nem jelöltünk ki különálló tesztelőt, hanem a tesztelés feladata minden projekttagnak, azaz minden fejlesztőnek és a termékgazdának is, akkor is célszerű kijelölni egy tesztelés felelőst a csapatból. A tesztelés felelős feladata és felelősége, hogy az elkészült funkcióknak a tesztelését megszervezze, felügyelje és dokumentálja. A tesztelési jegyzőkönyv projektdokumentumtól eltekintve (ami formális, leadandó dokumentációja a tesztfázisnak), a gyakorlatban a leginkább az előző fejezetben említett ticketing rendszer a legalkalmasabb. Ennek oka, hogy egyszerűen riportozható, könnyen karbantartható, átlátható és strukturált formában tartalmazza a tesztelendő feladatokat, valamint a tesztelés végeredményét. sikerességének vagy sikertelenségének tényét is egyszerűen nyomon lehet követni benne.

Résztvevők: tesztelő(k) vagy tesztelés felelős és a tesztelési feladatok elvégzésére kijelölt projekttagok, továbbá a termékfelelős

Elkészült termék: a tesztelők feature szinten nézik meg az elkészült terméket, továbbá ha a feladat jellege megköveteli, akkor performancia teszteket végeznek, valamint regressziós tesztet végeznek az új verzió kiadása előtt. A termékfelelős felhasználói elfogadó teszteket (user acceptance tesztek) végez a rendszeren. A tesztek végeredményét Redmine-ban adminisztrálják kétféleképpen: ha a felmerült hiba egy feature-rel kapcsolatban keletkezett, akkor az adott feature-t teszt sikertelenre kell állítani, és visszaadni fejlesztésre, valamint ha nem egyértelműen feature-höz köthető, akkor hibaként (bug-ként) kell felvenni és eljuttatni a fejlesztőkhöz javításra.

Tipp, jó tanács: a tesztelés gyakorlatilag folyamatos kell, hogy legyen a sprint teljes időtartama alatt. A tesztelő naponta (akár többször is) megkapja az elkészült funkciókat tesztre, és amíg a fejlesztő csapat egy másik funkciót fejleszt, addig a tesztelő leteszteli a készre jelentett funkciókat. A fejlesztők – tesztelők arányát annak megfelelően kell megválasztani, hogy a keletkezett munkamennyiség ne legyen túl sok vagy túl kevés a számunkra, és ezzel a folytonosság biztosítható.

Említést kell tennünk az automatizált tesztek fontosságáról. Az automatikus tesztek írása nagy munka és lényegében sose mondható, hogy elkészült, mert folyamatos karbantartást igényel. Egy jól működő automatikus teszt viszont nagy hasznot tud hozni főleg a regressziós tesztelések terén. Agilis környezetben, ahol a folyamatos változáson van a hangsúly, nagyon nehéz időálló automatikus teszteket írni, mert hétről hétre, napról napra megváltozhatnak korábban már készre jelentett elemek. A változás beborítja az automatikus teszteket, gyakorlatilag újra kell írni az érintett részeit. Feladat- és projektfüggő tehát, hogy megéri-e a teszteket automatizálni, a megírásuk és karbantartásuk erőforrásigényét kell összevetni a várható hasznukkal.

Az automatizált tesztek egy speciális fajtája a tesztelés vezérelt fejlesztés (TDD – Test Driven Development). A TDD lényege röviden összefoglalva, hogy a fejlesztő minden leírt programkód sorához kell, hogy tesztet is készítsen és futtasson. A gyakorlatban hatékonyan működő modell, de programozói szemléletváltást igényel. Továbbá munkaigénye nem kisebb, de kevesebb utólagos karbantartást igényel, viszont a programozót terheli (nem a tesztelőt).

A hibajavítás szintén folyamatos feladat a sprint során. A tesztelő (vagy a termékfelelős) priorizálja a megtalált hibák súlyosságát, ezáltal elhelyezi a hibákat a fejlesztendő feature-k közé, így egy homogén, priorizált listát kap a csapat, ami alapján az előzőekben leírtak szerint történik a fejlesztés. A priorizált listából való választás esetén a fejlesztőknek nem kell különbséget tenniük bug és feature között, sőt a teszt sikertelen feature-k is visszakerülnek a listára.

Sprintzáró demó

Felelős: a sprintzáró demó szervezése a product owner felelősége, hiszen ő – mint a megrendelő és a fejlesztőcsapat között álló kapocs - érdekelt a leginkább abban, hogy a megrendelőnek be legyen mutatva az elkészült (rész)termék.

Résztvevők: a demó-t rendszerint a vezető fejlesztő, vagy az a fejlesztő, akit a csapat kiválaszt erre a feladatra, szokta tartani. A demón részt vesz(nek) a megrendelő(k), illetve a menedzserek is, és bárki, aki valamilyen kapcsolatban áll a projekttel.

Elkészült termék: a demó egy élőszóban megtartott bemutató, ahol a fejlesztők megmutatják azokat a funkciókat, amelyek elkészültek a sprint során. A nem elkészült funkciókat nem lehet bemutatni, de fontos őket azonosítani, mert fontos a megrendelőt tájékoztatni arról, hogy pontosan mit is fog kapni ebben a fázisban. Az iteratív fejlesztés „elbírja” azt, hogy egyes funkciók később, egy következő sprintben készülnek el, vagy hogy átadáskor a rendszer még nem teljesen hibamentes (de vannak már részei, amik használhatóak akár élesben is), viszont arra kell vigyázni, hogy a projekt vége hogyan van leszabályozva. A megrendelő nyilván nem lesz boldog, ha az idő lejártával félmunkát adunk neki át, mert csak ennyi fért bele. A fejlesztő sem lenne boldog attól, hogy „ingyen” garanciában még meg kell valósítania az elmaradt funkciókat, amelyek egészen hosszú időre is kitolódhatnak. Összefoglalva az agilis módszertanok szerinti fejlesztés speciális, körültekintő projektszerződést igényel, amire jobb, ha a projekt megkezdése előtt gondolunk.

A laborkörnyezetben nyilván ez akkor jönne elő, ha külső megrendelőnek készítenénk az alkalmazást.

6. Általános gyakorlati irányelvek az agilis fejlesztéshez

Az alábbiakban összefoglaljuk a SCRUM módszertan gyakorlati alkalmazásának legfontosabb tanulságait, és egyben tanácsoljuk a labormunka alkalmával ezek betartását:

- egy sprint 2-3 hetes legyen (az ennél hosszabb időtartalmúak már nem hatékonyak, a rövidebbek pedig túl rövidek ahhoz, hogy egy funkció lefejlesztésére elegendőek legyenek)
- annyi feladatot kell vállalni a sprintre, ami a becslések alapján belefér a fejlesztési időbe figyelembe véve a fejlesztők labormunkával töltött óraszámát
- figyeljünk a csapatmunka hatékonyságára (hatékonyság növelő eszközök lehetnek az egy légtérben való elhelyezés, a napi stand-up meetingek, a ticketing rendszer, a határozott és tapasztalt scrum master jelenléte).

6.1. Külső megrendelők esete

A tanulmányban azt az esetet vizsgáltuk, amikor a projektötlet gazdája belső személy (diák vagy oktató). A labornak másodlagos célja olyan projektötleteket találni, amelyek mögött valóságos megrendelő, azaz piaci szereplő áll. Nézzük meg, hogy ilyen esetben miben változik a folyamat:

- a projekt előkészítő fázisa változik olyan szempontból, hogy nem kell projektötleteket esetleges pályáztatással gyűjteni a diákok köréből, vagy nem az oktatók találják ki az ötleteket, hanem a feladat adott.
- kockázata, hogy a feladat, amire a külső fél szerződni akar, nagyobb (több funkciópontból álló), mint amit egy csapat egy szemeszter alatt el tud végezni. Ebben az esetben meg kell oldani a folyamatosságot, akár a nyári szünet alatti dolgozást, vagy az egymást követő szemeszterek miatti személyi változásokat a projektcsapatban.
- termékfelelőst (product owner) továbbra is kell választani, mert lennie kell egy olyan (célszerűen) belső személynek, aki megvalósítja a fejlesztők és a megrendelők közötti kapcsolatot, mindig elérhető a fejlesztők számára, de üzletileg a megrendelőt képviseli a SCRUM csapatban. Piaci megrendelés esetében célszerű, hogy a product owner szerepét oktató vagy tapasztaltabb phd hallgató töltse be, mert kifelé (egyetemen kívülre) is kell tudnia kommunikálni és kapcsolatot tartani.

6.2. A projektmenedzser szerepe

Láttuk, hogy a tanulmányban lényegében nem került elő a projekt menedzser szerepkör. Ez nem véletlenül van így. A projekt menedzser a projekt adminisztratív vezetője, az agilis módszertan viszont elsősorban nem projektek vezetésére vonatkozik, hanem csak a szoftverfejlesztésre korlátozódik. Egy projekten belül a szoftverfejlesztés feladaton kívül sok más teendő is van, pl. szerződés, jogi környezet bevonása, beszerzés, üzemeltetés, support tevékenység, dokumentálás, felhasználók oktatása, stb. Ezek a feladatok a projekthez tartoznak, de nem a SCRUM-hoz. A SCRUM kizárólag a szoftver előállításának szakaszára ad módszertani tanácsokat.

A gyakorlatban a SCRUM alatt általában a projektmenedzser szokta betölteni a scrum master szerepkört.

7. Leendő kutatási feladatok

A tanulmányban bemutattuk, hogy hogyan képzeljük el az agilis módszertan alkalmazását az egyetemi projektlaborban. Kollégáim korábban a „Szakmai gyakorlat egyetemen belül" című esettanulmányukban [1] bemutatták, hogy milyen módszerekkel működött és működik a labor a létrehozását követően napjainkban is. Az első fejezetben kitértünk rá, hogy mi miatt merült fel az igény az agilis módszertan bevezetésének. Nyilvánvalóan az a gyakorlati célunk, hogy a labor minél hatékonyabban működjön, és minél több tapasztalatszerzési lehetőséget biztosítson diákjaink számára, továbbá nem utolsó sorban működése piaci szereplők figyelmét is felkeltse, hogy megrendelésekkel bízzák meg a labort.

A jövőben célunk az itt leírt feltételezett működési modellt a labor gyakorlatában is kipróbálni, és összehasonlító elemzéseket végezni a hatékonyságával kapcsolatosan. A kísérletet úgy tervezzük végrehajtani, hogy két - nagyjából méret és tudásszint szempontjából hasonló – csapatot terjedelmét és jellegét tekintve hasonló feladattal bízunk meg, de az egyik csapat a hagyományos módon, a másik csapat az itt leírt agilis technikát alkalmazva kell, hogy a félév során dolgozzon a feladaton.

A félév végén a következő összehasonlításokat tervezzük elvégezni:

- a feladatot a két csapat mennyire tudta teljesíteni?

  Melyek azok a funkciók, amelyek megvalósultak, és melyek nem. Keressük a választ arra, hogy miért pontosan azok a funkciók nem valósultak meg, illetve a nem megvalósult funkciók milyen hatással vannak az átadott termékre.
- a megvalósult funkciók milyen minőségben valósultak meg?

  Felfedezhető-e valamelyik módszernél típushiba?
- a csapat erőforrás ráfordítási igénye mekkora volt?

  Ki mennyit dolgozott a feladaton, milyen részfeladatokat vállalt a teljes projektből?
- mekkora volt a tervezés – fejlesztés – tesztelés – dokumentálás fázisok erőforrás ráfordítási ideje, ez milyen kapcsolatban van az elkészült termékek minőségével
- a projekttagok szubjektív véleménye, hangulata milyen volt a projekt alatt? Milyen problémák merültek fel, azokat milyen hatékonysággal lehetett elhárítani?

Az elemzésekhez szükséges információkat kétféle módon tervezzük begyűjteni: egyrészt a tényadatokon alapulva a ticketing rendszerben adminisztrált bejegyzéseket szeretnénk elemezni, valamint a tényleges elkészült termékeket (szoftvert és dokumentációit) szeretnénk megvizsgálni, másrészt a szubjektív személyes véleményeket tervezzük felhasználni, amiket egyrészt a projekt végén a tagokkal készült interjúk alapján gyűjtünk be, másrészt a projektet felügyelő oktatók, laborszervezők heti státuszjelentéseiből dolgozunk ki.

A begyűjtött nyers adatokon különféle összehasonlító, benchmarking elemzéseket tervezünk végrehajtani a funkciópont elemzés módszerét felhasználva, az agilis projektekre testre szabva, esetleg automatizált eszközöket használva, hogy csökkentsük a munkaráfordítást.

Továbbá megkeressük majd a megfelelő vizsgálatok elvégzésére alkalmas modelleket a statisztikai modellezés, neuronhálók és / vagy adatbányászati módszerek területén, majd az elvégzett vizsgálatok eredményét kiértékelve következtetéseket vonunk le az alkalmazott modellek hatékonyság szempontból történő javítására vonatkozóan.

Irodalom

[1] Gombos Gergő, Matuszka Tamás, Pinczel Balázs, Rácz Gábor, Kiss Attila. „*Szakmai gyakorlat Egyetemen belül – Esettanulmány*", InfoDidact, (2012).

[2] Molnár Bálint. „*Funkciópont elemzés a gyakorlatban*", MTA Információtechnológiai Alapítvány, (2003).

[3] *MKII funkciópont elemzés súlytényezői* http://194.143.167.33/library/Resources/MkIIr131.pdf (2001)

[4] Mike Cohn. „*Succeeding with Agile*", (2012).

[5] Sereg Ákos, Varga Balázs. „*Agilis szoftverfejlesztés és scrum*", Miskolc, (2008).

[6] „*Projektmenedzsment és az agilis fejlesztés"* http://akocsis.blog.hu/2012/03/06/projektmenedzsment_es_az_agilis_szoftverfejlesztes

[7] Schwaber, Ken, and Jeff Sutherland. „*The scrum guide*." Scrum. org, October (2011).

Rising, Linda, and Norman S. Janoff. „*The Scrum software development process for small teams"* IEEE software 17.4 (2000): 26-32.

„*Scrum*.", http://hu.wikipedia.org/wiki/Scrum

[8] Schwaber, Ken. „*Scrum development process"* Business Object Design and Implementation. Springer London, (1997). 117-134.

[9] Schwaber, Ken, and Jeff Sutherland. „*What is Scrum*." URL: http://www. scrumalliance. org/system/resource/file/275/whatIsScrum. pdf,[Sta nd: 03.03. 2008] (2007).

[10] *Redmine hivatalos oldala*, www.redmine.org

[11] „*Tesztelés a gyakorlatban*", http://www.tesztelesagyakorlatban.hu/

[12] Redmine CRM, „*Redmine premium plugins*" http://redminecrm.com/