XX. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2017. május 18–21.

Agilis szoftverfejlesztés a felsőoktatásban: tartalom vagy módszer?

Kivonat

Az agilis módszerek népszerűsége az elmúlt évek során megnőtt a szoftverfejlesztő csapatok körében, így azok egyre gyakrabban kerülnek a figyelem középpontjába. Az egyetemek alap- és mesterképzésen egyaránt változatos módszerekkel igyekeznek felzárkózni e módszerek tanításában. Jelen dolgozat módszertani javaslat az agilis módszerek és a scrum keretrendszer alapképzésen való oktatására, valamint egy kutatás eredményeit elemzi, amelyet harmadéves alapképzéses egyetemistákkal végeztünk a Babeș-Bolyai Tudományegyetemen, a javasolt módszer alkalmasságának ellenőrzésére. Az eredményeket kérdőívvel, illetve egy olyan szoftverrel ellenőrizzük, amely az egyetemisták félévi tevékenységét – verziókövető rendszerek és projekt menedzsment eszközök használatát – monitorizálja.

Szerző:
Deák Anna, Babeș-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika kar, Vállalati szoftvertervezés és -fejlesztés szak, mesterképzés

Témavezetők:
dr. Barabás László egyetemi adjunktus, Babeș-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika kar
drd. Sulyok Csaba doktorandusz, Babeș-Bolyai Tudományegyetem, Kolozsvár, Matematika és Informatika kar

Tartalomjegyzék

1. Bevezetés

Az agilis módszerek jelentős teret hódítottak az elmúlt években, ez minden bizonnyal annak köszönhető, hogy alternatívát jelentenek a hagyományos metódusokkal szemben. [BBB+] Az angol szó (*agile*, magyarul *fürge, gyors*) szemléletesen foglalja össze mindazt, amit az egész mozgalom jelent: nyitottságot a változó igényekre, hajlandóságot az alkalmazkodásra, gyors és világos fejlesztést.

Ahogy azt a *Szakirodalmi áttekintés* című fejezetben kifejtjük, számos külföldi egyetem kutatóközössége jutott arra a következtetésre, hogy időszerű az agile/scrum oktatását bevezetni, akár már alapképzésen. A szerző személyes tapasztalatai is azt mutatják, hogy az egyetemi oktatás nem helyez nagy hangsúlyt az agile ismertetésére. Az alapképzés alatt, illetve annak elvégzése után munkába álló hallgatóktól már elvárják e módszerek ismeretét, begyakorlásukra viszont ritkábban nyílik lehetőség a működő projektek futamideje alatt.

Az agilis módszerek és a scrum keretrendszer elméleti háttere c. fejezet összefoglalja mindazon információkat, amelyek szükségesek ahhoz, hogy az általunk bemutatott témák és módszerek érthetőek legyenek, illetve, hogy ezek fontossága és hasznossága mellett érvelhessünk. A szakirodalom részletesen foglalkozik a különböző módszerek összehasonlításával, erre mi külön nem térünk ki.

A *Módszertani javaslat a scrum keretrendszer gyakorlati oktatására* című részben felvázoljuk azt a tervet, amelyet a 2016/2017-es tanév során, alapképzésen tanuló harmadéves egyetemistákkal ki is próbáltunk, hogy a módszer gyakorlati alkalmazhatóságát ellenőrizzük.

A hallgatókkal közösen végzett kísérlet eredményeit négyféle metrikával vizsgáltuk, ezeket az *Alkalmazott metrikák a kutatás elemzésében* című fejezetben mutatjuk be. *A mérésekhez használt webalkalmazás bemutatása* részletesebben szemlélteti a szoftvertermékét, amelyet az adatok begyűjtésére használtunk. Az *Eredmények elemzése* során levonjuk a következtetéseket a módszer alkalmazhatóságáról.

Ahogy a fejezetcímek is jelzik, jelen dolgozat célja egy olyan módszertani javaslat összeállítása, amely már alapképzésen betekintést nyújt a diákoknak az agile elméletébe és a scrum keretrendszer használatába, a lehetőségekhez mérten gyakorlati eszközökkel. A kutatást megelőző évek során a Babeș-Bolyai Tudományegyetem Matematika-Informatika karán belül két tantárgy nyújtott lehetőséget arra, hogy az itt tanulók megismerkedhessenek az agilis módszerekkel és a scrummal; az egyik (jelenleg is oktatott Software technológia [DB]) az alapképzés harmadévének első félévében kapott helyet, a diákok ekkor ismerik meg a különböző szoftverfejlesztési keretrendszerek és módszerek elméleti hátterét. A másik tantárgy (Agilis módszerek és vállalatfejlesztési stratégiák) mesterképzésen volt hallgatható, a Vállalati szoftvertervezés és -fejlesztés nevű képzés keretén belül. Annak érdekében, hogy az alapképzésről az iparba kerülő egyetemisták már használható ismeretekkel rendelkezzenek az agile-ról és a scrumról, a jelen módszertant az ők igényeihez és lehetőségeihez igazítottuk.

2. Szakirodalmi áttekintés

Az agilis módszerek a szoftvertervezés és -fejlesztés gyakorlatának szerves részét képezik már 2001 óta [BBB+], folyamatosan egyre nagyobb teret hódítva maguknak. A képzett munkaerő iránti igény növekedésével az egyetemi oktatók egyre gyakrabban kezdeményezték az agile bevezetését az egyetemi oktatásba, mint ahogy az a szakirodalom tüzetesebb tanulmányozása során kiderült. A következőkben témánk kutatóinak legfontosabb megállapításait és következtetéseit foglaljuk össze.

A legrészletesebb összefoglalót Viljan Mahnić 2015-ben írt dolgozatában olvashatjuk [Mah15]. Ez a munka tartalmazza a szakirodalom legfontosabb eredményeit az agile és a scrum egyetemi oktatáson belüli alkalmazásáról. Mahnić saját kutatásait végül a scrumra szűkítette le, így részletesen csak húsz, ebben a témában releváns cikket elemez. A bemutatott kutatások többsége a scrum oktatását gyakorlati eszközökkel látja megvalósíthatónak, féléves projektek keretén belül. Egyes tanulmányok oktató játékok alkalmazását tárgyalják, mások a diákok visszajelzéseire reflektálnak, míg a dolgozatok egy harmadik csoportja a pedagógiai kérdéseket taglalja. Ugyanúgy változó a hozzáállás a Product Owner (termékgazda) és a Scrum Master (scrum mester) személyét illetően. Míg az esetek többségében (hét esetben a tizenegyből) a Product Owner egy az iparban jártas szakember, a Scrum Master általában a diákokból álló fejlesztő csapat tagja (hét esetben a tizenegyből).

Mahnić kutatásokat végzett a scrum csoportos munkán belül való oktatásáról is [Mah10] [Mah12]. Következtetéseit az egyetemisták visszajelzéseire és a projekt kimenetelére alapozta. Kísérletének második, javított formájában tizenhárom csapattal dolgozott. Ezek mindegyike ugyanazt a terméket készítette el, négy sprint (futam) alatt. Az oktató játszotta a Product Owner és a Scrum Master szerepét is. A félév utáni megfigyelések alapján a csapatok könnyen alkalmazkodtak a munkamódszerhez, a harmadik sprint végére mindegyikük fejlődött az esztimálási gyakorlatokban, és a résztvevők megtanulták annak a fontosságát, hogy nyíltan és gyakran kommunikáljanak a termékgazdával. A kérdőívek azt jelzik, hogy a módszer népszerű a hallgatók körében. A tapasztalat szerint a scrumot leginkább gyakorlatban lehet megtanulni, illetve a kommunikáció kulcsszerepet játszik a projektek sikerességében. Túl a *scrum az oktatásban* témakörön, Mahnić számos más tanulmányra is utal, amelyekben az egyetemen tanított Extreme Programming-ről vagy a scrum/XP keverékéről olvashatunk.

Orit Hazzan és Yael Dubinsky az interperszonális kapcsolatok és az ipari gyakorlat alapján tíz érvvel támasztja alá, hogy időszerű lenne bevezetni az agilis módszereket a mérnöki programok tanterveibe. [HD07] A két szerző később e tíz érv alapján dolgozott ki egy keretrendszert, amely az extrém programozáson keresztül oktatja az agile-t. Kilenc félév, harmincegy projekt és több mint háromszáz egyetemista együttesére volt szükség a módszer tökéletesítésére, amely így már biztosan alapoz az extrém programozás értékeire: kommunikáció, visszajelzés, egyszerűség és bátorság. [DH05]

A Swiss Agile Study eredményeiből indulnak ki Martin Kropp és Andreas Meier kutatásai.

A svájci felmérés célja az volt, hogy felmérje és elemezze az agile használatát a belföldi vállalatok és projektek mindennapjainak szintjén. A megkérdezettek a képzett munkaerő hiányát nevezték meg a legnagyobb blokkoló tényezőnek az agile alkalmazásának területén. Ennek orvoslására a szerzők egy olyan képzési programot állítottak össze az alapképzésen tanulók számára, amely az agilis kompetenciák három szintjére épít: mérnöki és ipari gyakorlat, agilis menedzsment gyakorlat és agilis értékek. A Scrum és XP kombinációjával sikeresen lefedték mind a menedzsment-beli, mind az ipari gyakorlatokat. A diákok visszajelzései túlnyomóan pozitívak voltak. [KM14] A kidolgozott képzési javaslatot 2013-ban alkalmazták először. Ennek alapját egy olyan gyakorlatsor képezi, amelynek következetes és tudatos ismétlése hozzájárul a berögződött szokások kialakulásához. [KM13] Egy későbbi kutatásban Kropp, Meier, Mateescu and Zahn kiemelték a kommunikációs és kollaborációs készségek fontosságát az agilitás oktatásában, illetve egy olyan javaslattal álltak elő, amely egyéni, csapat- és szervezet-szinten kezeli ezeket. Az általuk kidolgozott módszer egy olyan játék, amely a mindennapi szoftverfejlesztési feladatoktól elvonatkoztatva csak az agilis értékekre összpontosít. [KMMZ14] A két előző cikk eredményeit figyelembe véve Kropp és Meier kombinálja a két módszert. A *Scrum City Game* és egy tizenhat hétig tartó csoportos projekt kettőse nagy sikernek örvendett az egyetemi hallgatók körében. [KM16] [KMB16]

Peter Maher az XP segítségével tanítja az agilitást mesterképzéses hallgatók számára, egy féléves előadás keretein belül [Mah09]. Tapasztalatai közül kiemeli, hogy a hallgatók nehezen becsülik fel az eltervezett munkához szükséges időt, nem tudják megfelelően kihasználni a tesztelési keretrendszerek nyújtotta lehetőségeket, illetve hajlamosak arra, hogy túl sok időt szenteljenek a projekt megtervezésére. A szerző végül javasolja egyes témakörök alapképzésen való oktatását.

A University of Maryland University College mesterképzésébe szerette volna bevezetni a hagyományostól eltérő agilis módszertan oktatását David F. Rico és Hasan H. Sayani [RS09]. A résztvevőket három csapatra osztották, mindegyik csoportnak ugyanazt a követelményrendszert adták ki megvalósításra. Mivel a csapattagok már rendelkeztek programozási tapasztalattal, a tantárgy keretén belül lehetőség nyílt az agile oktatására koncentrálni. A tapasztalatok azt mutatják, hogy egy ilyen felállás esetén hasznos lett volna az agile elméletének előzetes ismerete, illetve hogy a mentorok szerepe kulcsfontosságú. A lehető legjobb eredmények elérésének érdekében megfelelő személyeknek kellene ellátniuk ezt a feladatot.

Baochuan Lu és Tim DeClue egyszerre tanítják a hagyományos és az agilis módszereket egy féléves projekt keretén belül [LD11]. A pozitív visszajelzések mellett kiemelnek pár, a tesztelésben és a mérföldkövek felállításában felmerült gondot. Ezek mélyebb okai leginkább a berögződött hagyományos módszerekben keresendőek.

A legnagyobb akadályt az agile oktatási alkalmazásában a szoftverfejlesztési gyakorlat hiánya jelenti Tucker Smith, Kendra M.L. Cooper és C. Shaun Longstreet tapasztalata szerint is [SCL11]. Kiemelik, hogy az igazi agilitás megéléséhez dedikált emberekre lenne szükség, ez viszont nehezen kivitelezhető a túlterhelt egyetemisták esetében. Megfigyeléseik szerint kárba

vész a jó munkamegosztás, amennyiben a csapattagok nem képesek egymással kommunikálni.

Egy három kontinensen átívelő projekten alapul Christelle Scharff, Olly Gotel és Vidya Kulkarni kutatása, amelynek öt tagja kilenc hétig dolgozott egy adott szoftverterméken [SGK10]. Ezen felállásban a kommunikációs akadályokat és az időzóna-eltolódás okozta problémákat virtuális eszközök figyelmes kiválasztásával próbálták megoldani. A projekt sikere megkérdőjelezhető, ugyanis a scrum eseményeket nem lehetett teljes mértékben kivitelezni, és a munkára szánt idő is nehezen volt megbecsülhető.

Chuan-Hoo Tan, Wee-Kek Tan és Hock-Hai Teo célja egy olyan egyetemi tantárgy kidolgozása volt, amely nagy hangsúlyt fektet az alkalmazkodási készségre, a rugalmasságra és az agilitásra [TTT08]. A tantárgyat négy féléven keresztül oktatták. A visszajelzések alapján egy olyan egyedi rendszert dolgoztak ki, amely ötvözi a scrumot, az extrém programozást és a tesztvezérelt fejlesztést. Tudatosan iktattak be olyan változó igényeket, amelyekre a projekt során derült fény, ezekkel próbálták javítani a hallgatók alkalmazkodási készségét.

Craig Anslow és Frank Maurer szerint az agilis szoftverfejlesztés oktatásában a *cselekedve tanulás* a legcélravezetőbb [AM15]. Az általuk kidolgozott képzésben az agile állt a figyelem középpontjában, az elkészítendő szoftver csak eszköz maradt. Szerintük a programozási gyakorlatot és az agilitást nem kellene együtt tanítani; a csapatokat visszahúzó tényezők közül a lassú visszajelzést és az egyenlőtlenül elosztott munkát emelik ki.

Adarsh Kumar Kakar azon a véleményen van, hogy az agilitás elméletének alapos ismerete nélkül nem beszélhetünk gyakorlati alkalmazásról [Kak14]. Az elmélet mögötti érvek nemcsak didaktikai haszonnal bírnak, hanem segítenek a tudást átültetni a gyakorlatba, projekttől függetlenül. Hogy az agile jelentőségét megérthessük, azt is tudatosítanunk kell, hogy milyen problémák megoldására jött létre eredetileg.

A szoftvertervezés és az agilis fejlesztés terén szerzett nyolc éves tanítási tapasztalatát foglalja össze Vladan Devedžić és Saša R. Milenković [DM11]. Pár bevett gyakorlat segíthet abban, hogy a diákok sikerélményként könyveljék el a féléves munkát: ilyenek a rövid iterációk, az akadályok időben való felismerése és elhárítása. A megtanult eszközök ismételt alkalmazása is segíthet a tudás elmélyítésében. Kicsi, önszerveződő csapatok alkalmazásával a hallgatók nyitottabban kommunikálnak és felelősségteljesebben állnak hozzá a várt feladathoz.

A Jonas Schild, Robert Walter és Maic Masuch által oktatott tárgy keretén belül játékfejlesztést tanulnak a hallgatók, a meglévő elméleti és gyakorlati tudást egy szintre hozva [SWM10]. A scrum keretrendszer bevezetésével az volt a céljuk, hogy javítsák a csapatmunkát, a rugalmasságot, a produktivitást és az időben elkészülő, használható prototípusokat, illetve hogy megkönnyítsék az elméleti és gyakorlati tudás együttes alkalmazását. A hallgatók a vártnál is jobban teljesítettek. Miután itt a többszintű tudás együttes alkalmazása volt a vizsgálat tárgya, a tanárok nem pontozták a programozást, ám a csapatok mégis nagy hangsúlyt fektettek rá. A jövőre való tekintettel a kutatók kiemelik az interaktív és gyakori visszajelzés fontosságát.

Ahogy az irodalomból vett számos példa igazolja, léteznek már kísérletek az agilis módszerek és a scrum egyetemi oktatására. A fentebb említett tapasztalatok számunkra is hasznosak.

Noha ugyanazokról a problémákról beszélhetünk mi is, mint az előttünk szólók, az átadandó tudásanyagot mégis más körülményekhez kell igazítanunk. Kutatásunk egyik egyedisége az, hogy más helyi realitáshoz alkalmazkodik, nem vezet be új tantárgyat, hanem a meglévők keretén belül keresi meg a lehetőséget. Egy másik jellegzetessége az, ahogy az eredményeket begyűjti és elemzi: az általunk olvasott szakirodalomban nem találtunk példát saját fejlesztésű alkalmazás használatára az adatgyűjtésre, általában kérdőív és vizsgajegyek alapján mérik a munka sikerességét. Célunk megkeresni a módot arra, hogy a meglévő körülményekben miként lehet az agile/scrum témákkal érdemben is foglalkozni.

3. Az agilis módszerek és a scrum keretrendszer elméleti háttere

Az agile-t és a scrumot gyakran említik ugyanabban a kontextusban. Ez nem meglepő, hiszen kialakulásuk, történetük összefonódik. Az agilitást mégis egy tágabb gyűjtőcsoportnak tekintjük, amely a scrumot is magába foglalja. A scrum mint szoftverfejlesztési keretrendszer, az agilitás megfogalmazódása előtt jött létre.

2001 februárjában a különböző szoftverfejlesztési keretrendszerek tizenhét képviselője találkozott, hogy megvitassák ötleteiket. Az eredményt az *Agile Manifesto* nevű dokumentumban rögzítették. A kiáltvány, amelyet minden jelen lévő aláírt, megfogalmazza az agile alapjait. Véleményük szerint a következőket előnyös értékelni:

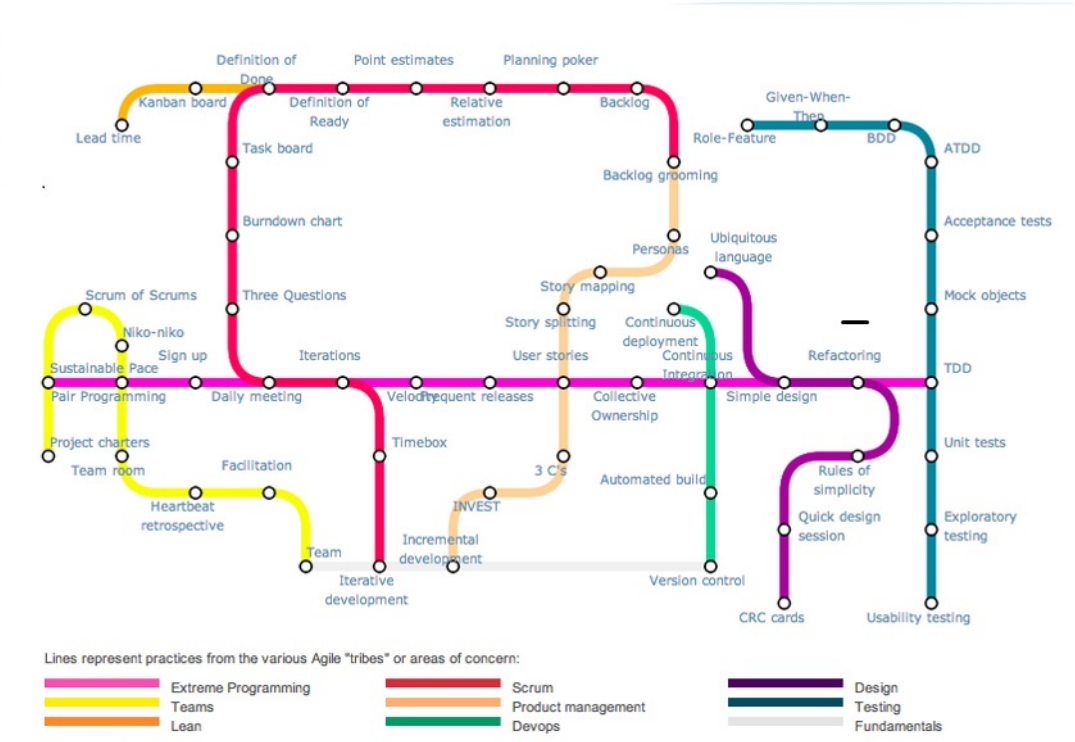
- "Az egyéneket és a személyes kommunikációt a módszertanokkal és eszközökkel szemben
- A működő szoftvert az átfogó dokumentációval szemben
- A megrendelővel történő együttműködést a szerződéses egyeztetéssel szemben
- A változás iránti készséget a tervek szolgai követésével szemben."

Az aláírók tizenkét pontban részletezik a fentieket. Kiemelik a csapatok önszerveződésének fontosságát, a projektben érintett személyek közti szoros együttműködés és a mindennapi kommunikáció szerepét, a működő szoftver mihamarabbi és folyamatos szállítását a kliens felé, a reakciókészséget, az egyszerűséget.[BBB+]

Ahhoz, hogy a fent említett elveket kivitelezzék, a szoftverfejlesztés során számos agilisnak nevezett olyan gyakorlatot alkalmaznak. Ilyen például a Backlog Grooming (termék kívánságlista napirenden tartása), a Continuous Integration (folyamatos integráció), a Definition of Done (“kész" meghatározása), az Iteration (iteráció), a Kanban tábla (Kanban Board), a tervező póker (Planning Poker), a kód refaktorálása (Refactoring), az egyszerű Simple Design (egyszerű tervezés), a Story Splitting (feladat felbontása) és a Test Driven Development (tesztvezérelt fejlesztés) [All]. Noha ezek mind agilis gyakorlatok, használatuk nem feltétlenül része minden egyes szoftverfejlesztési keretrendszernek (lásd az 1. ábrán).

A scrum szoftverfejlesztési keretrendszer alapjait Jeff Sutherland és Ken Schwaber fektették le 1995-ben. Szerintük a scrum egy olyan keretrendszer, amelyen belül lehetőség nyílik összetett problémák leküzdésére, és a lehető legjobb minőségű eredmények kreatív és hatékony szállítására. Hangsúlyozzák, hogy a scrum nem egy munkafolyamat, vagy gyártási gyakorlat, hanem egy keretrendszer, amely teret ad számos folyamat és gyakorlat alkalmazására [SS16].

A keretrendszer alkotóelemei a *scrum csapat tagjai*, azok *szerepkörei*, az *események*, a *scrum termékek*, illetve azok a *szabályok*, amelyek mindezeket összefogják. Mindez alapját



1. ábra. Adott agilis keretrendszerek különféle módszerekkel dolgoznak, noha vannak átfedések is. A különböző praktikák és keretrendszerek közti kapcsolatot térképszerűen is lehet szemléltetni. [1]

az empirikus tapasztalat képezi, a keretrendszer arra bátorítja a csapatok tagjait, hogy a meglévő tudásuk alapján hozzanak döntéseket. A scrum iteratív, inkrementum alapú megközelítése beláthatóbbá teszi a munkafolyamatot és segít a kockázatok kézben tartásában [SS16]. Az empirikus alapú munka azt feltételezi, hogy a projekt teljes időtartama alatt következetesen alkalmazzuk az *átláthatóság, felülvizsgálás és alkalmazkodás* elvét. Erre a scrum események nyújtanak megfelelő keretet. Ahhoz, hogy egy csapat ezen elvek alapján hatékonyan dolgozzon, ideális esetben öt értéket kell magáénak valljon. Ezek a következők: "elköteleződés, bátorság, összpontosítás, nyitottság és tisztelet" [SS16].

3.1. A scrum csapat és tagjainak szerepkörei

A scrum csapatok egyik fontos jellemzője a kölcsönös funkcionalitás (cross-functionality). A csapat eleve rendelkezik minden szükséges tudással ahhoz, hogy függetlenül dolgozhasson egy feladaton. Nincsenek dedikált szerepkörök, amelyek esetleges kiesése vagy elérhetetlensége akadályozná a csapatot a munka befejezésében. Egy másik jellemző az önszerveződés. Nincs szükség egy külső személyre, aki leosztja a munkát a csapaton belül, a csapattagok döntik el, hogy ki mihez lát hozzá. Ez a két jellemző a "rugalmasságot, kreativitást és a hatékonyságot"

[1] Kép forrása: https://www.agilealliance.org/agile101/subway-map-to-agile-practices/, letöltési dátum: 2017. április 9.

hivatott növelni.

3.1.1. A Product Owner (termékgazda)

A Product Owner feladata a termék és a csapat által nyújtott munka értékének maximalizálása. Ő felel a Product Backlog (termék kívánságlista) napirenden tartásáért. Gyakran ő az összekötő a kliens és a fejlesztő csapat között.

3.1.2. A Development Team (fejlesztő csapat)

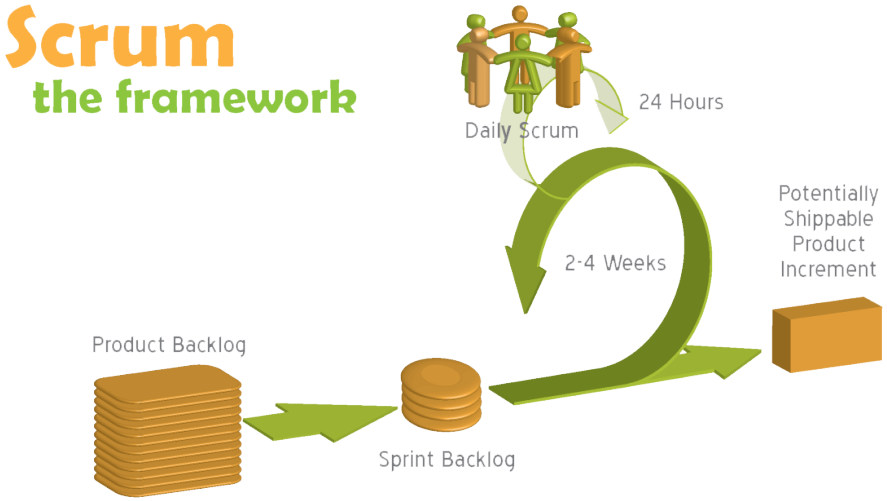
Ideális esetben 3-9 tagból áll. Olyan szakemberek közössége, akiknek fő feladata a termék kivitelezése és szállítása minden sprint (futam) végén. A kölcsönös funkcionalitáson és az önszerveződésen túl megemlítjük, hogy az egyetlen cím, amit a csapattagok viselhetnek, az a *developer* (*fejlesztő*). A scrum keretén belül nem beszélünk alcsapatokról, és noha előfordulhat, hogy egy-egy csapattag szélesebb tudáskörrel rendelkezik egy adott témakörben, a sprint eredményességéért a teljes csapat felelős marad.

3.1.3. A Scrum Master (scrum mester)

A Scrum Mastert gyakran a csapatot szolgáló vezetőnek nevezik. Ő felel azért, hogy a scrum elmélete, szabályai és bevett gyakorlatai tiszták legyenek mindenki számára, és hogy a csapat betartsa azokat. Nem csak a csapat irányába vannak feladatai, hanem a Product Owner és a vállalat felé is. A Product Ownert tekintve segít a product backlog hatékonyabb menedzselésében, tisztázza a csapat felé annak bemeneteire vonatkozó szabályokat. Tisztában van a termék megtervezésének folyamataival egy empirikus alapú környezetben, segít a Product Ownernek fontossági sorrendbe helyezni a backlog elemeit. Érti az agilitás elveit és segít azokat gyakorlatba ültetni, illetve igény szerint moderálja a scrum eseményeket. Segíti a csapatot az önszerveződésben és a kölcsönös funkcionalitás megvalósításában, igyekszik elgördíteni az előttük levő akadályokat, irányítja a csapatot olyan munkakörnyezetben, ahol a scrumot még nem ismerik vagy nem használják. Végül, de nem utolsó sorban a Scrum Master a vállalat szemében is felel a scrum nyújtotta lehetőségek helyes módon való megértéséért és alkalmazásáért, valamint más Scrum Masterekkel együttműködve egyengeti a vállalatot a keretrendszer bevezetésének útján.[SS16]

3.2. Events (események, ceremóniák)

A scrum események célja, hogy a résztvevők lényegretörően és hatékonyan szervezzék meg saját munkájukat, valamint hogy könnyebben tanuljanak saját tapasztalataikból. Az események során a csapatoknak lehetősége nyílik az átláthatóság növelésére és az önmagukra való reflektálásra. Fontos, hogy felismerjék a gátló tényezőket és növeljék saját teljesítményüket, alkalmazkodásukat.



2. ábra. A scrum munkafolyamatának legfontosabb lépései és eszközei. [2]

3.2.1. A Sprint (futam)

A sprint rögzített hosszúságú, legfeljebb egy hónapig tarthat és minden egyéb scrum esemény keretét képezi. Minden sprintet egy új követ, a bemutató és a visszatekintés után. A csapat célja az, hogy a sprint végén egy bemutatható és szállítható termékegységet mutasson fel, a sprint planningen (futamtervezés során) megbeszéltek alapján.

3.2.2. A Sprint Planning (futamtervezés)

Egyhónapos sprint esetén legfeljebb nyolc órát tartó esemény, amely az új sprint első mozzanataként egybegyűjti a csapatot. Ilyenkor döntik el a tagok, hogy milyen munkaegységeket fognek elvégezni az előttük álló sprint alatt. A bemeneti adatokat a product backlog, a múlt sprint eredménye és a csapat teljesítménye képezik, ezek alapján készül el az új sprint célja és a sprint backlog (futam feladatlista). Ez utóbbi mindazt a munkát tartalmazza, amelyek szükségesek ahhoz, hogy egy sprint alatt a csapat elérje a kitűzött célt [SS16].

3.2.3. A Daily Scrum (napi scrum)

A sprint hosszától függetlenül a napi scrum legfeljebb 15 percet tarthat. Naponta ismétlődő esemény, amelynek keretén belül minden csapattag választ ad a következő kérdésekre: "Mit tettem a tegnapi nap annak érdekében, hogy a csapat elérje a sprint célját? Mit fogok ma tenni ennek érdekében? Áll-e valamilyen akadály előttem?" [SS16] A napi scrum célja az, hogy kövesse, miként halad a csapat a kitűzött cél felé.

[2] Kép forrása: https://27gen.com/2012/01/24/introducing-the-scrum-to-churchworld/, letöltési dátum: 2017. április 24.

3.2.4. A Sprint Review (futam bemutató)

Egyhónapos sprint esetén maximum négy óra hosszú esemény. A csapat ezen belül mutatja meg az elmúlt sprint eredményét, nemcsak a termékgazdának, hanem valamennyi, a projektben érdekelt és az eredményre kíváncsi személynek. A bemutató után frissül a product backlog, aszerint, hogy a bemutatott munka el lett-e fogadva vagy sem.

3.2.5. A Sprint Retrospective (futam visszatekintés)

Egy legfeljebb három óra hosszú esemény, amelynek keretén belül a csapatnak lehetősége nyílik önmagára reflektálni, az elmúlt sprint fényében. Az észrevételek alapján a csapat egy olyan tervet készít, amely a következő sprinttől kezdődően segít elkerülni ugyanazon hibák megismétlődését. A scrum master felel azért, hogy a csapat ne csak a technikai akadályokra figyeljen, hanem az emberközi kapcsolatokra, folyamatokra és más problémákra is.

3.3. Artifacts (termékek, eszközök)

A scrum artifact-ek olyan egységek, amelyek a projekt során értéket hordoznak magukban.

3.3.1. A Product Backlog (termék kívánságlista)

A product backlog mindazokat a feladatokat tartalmazza, amelyek a termék kivitelezéséhez szükségesek, fontossági sorrendbe rendezve. Ez az egyetlen hely, ahol a termékre vonatkozó változási igényeket jelezni lehet. Minden listaelem rendelkezik leírással, fontossági sorrenddel, munkaidő-becsléssel és értékkel [SS16]. Kezdetben itt található minden előrelátható elvárás a projektet illetőleg, de ez nem jelenti azt, hogy az elvárások nem változhatnak a projekt futamideje során: a csapat készen kell álljon arra, hogy az esetleges változásokra reagáljon. Amennyiben több scrum csapat ugyanazon a terméken dolgozik, a product backlogon közösen osztoznak.

3.3.2. A Sprint Backlog (futam feladatlista)

A sprint planning (futamtervezés) során a product backlogból kiválasztott elemek a sprint backlogba kerülnek. A konkrét feladatokon kívül a sprint backlog tartalmazza a vállalt munka megvalósítására vonatkozó tervet is. Ide kerül be minden olyan munka, amit a csapat szükségesnek ítél a sprint céljának elérése érdekében. Ez egy valós idejű kép arról, hogy a csapat min dolgozik, illetve hogy mit tervez megvalósítani. Ez a termék kizárólag a csapat tulajdonát képezi [SS16].

3.3.3. Az Increment (kivitelezett érték, növekedés)

Az increment az a termékegység, amely önmagában (szállítható) értéket képvisel az ügyfél számára. Egy sprint alatt kivitelezett backlog-elemek megvalósítása, az előző sprintek eredmé-

nyeivel együttesen. Fő tulajdonsága a használhatóság [SS16].

Nem része a Scrum Guide-nak, hanem inkább egy agilis gyakorlat, de ennek ellenére fontos ismerni a *user story* fogalmát. Ezek rövid, lényegretörő leírásai a funkcionalitásnak úgy, ahogy azt a felhasználó szemszögéből látni lehet; nem a szigorú követelménylistát, hanem a *user* nézőpontját helyezi előtérbe. Amennyiben egy csapat sikeresen megvalósít egy *user story*-t, akkor elvileg egy olyan működő termékegységet tud nyújtani a felhasználónak, mely önmagában megállja a helyét; elvonatkoztat a termék vízszintes tagolásától (adatbázis, backend, felhasználói felület). A *user story* egy nagyobb funkcionalitás (*epic*) felbontása révén születik. Megfogalmazásában tartalmazza a szerepkört, az elvégzett cselekményt és a várt eredményt (például *Én, mint felhasználó, meg szeretném jeleníteni egy buszjárat összes adatát úgy, hogy megadom a kiindulási és érkezési pontot*). A *user story* megvalósításához szükségesek az elfogadási feltételek (*acceptance criteria*), a csapat ezeket tartja szem előtt akkor, amikor a feladat kivitelezésén dolgozik.

A scrum egy olyan keretrendszer, amelyet könnyű megérteni, viszont nehéz mesterszintre emelni. Manapság az egyik legnépszerűbb szoftverfejlesztési keretrendszerek egyike. A szerepeket, eseményeket és termékeket egyszerű, de szigorú szabályok kapcsolják össze, kedvező környezetet teremtve ahhoz, hogy a csapattagok kreatívan dolgozhassanak és hatékonyan működhessenek együtt.

4. Módszertani javaslat a scrum keretrendszer gyakorlati oktatására

A számítástechnika és a szoftverfejlesztés fejlődésével lépést tartva a vezető számítástechnikai közösségek olyan ajánlásokkal reagáltak, amelyek arra vonatkoznak, hogy mit lenne érdemes tanítani a felsőoktatásban. Az ACM ajánlásai a legrészletesebbek[fCM]. Az agilis módszereket 2009-ben említik először [AAA+09], több figyelmet viszont csak a 2014-es kiadásban kapnak [DIC+15]. A dokumentum negyedik fejezetében az agile-t a folyamatok kivitelezéséhez, az életciklus-modellekhez sorolják, és alapvető tudáskövetelményként jelölik meg.

Mindezt szem előtt tartva áttekintettük a Babeș-Bolyai Tudományegyetem informatika szakának tantervét. Az alapképzés leírásában megfogalmazott célok egyike az, hogy megismertesse a hallgatókkal a szoftverfejlesztési technológiákat, a mögöttük rejlő ötleteket, illetve az azokon belül alkalmazott módszereket [Depa]. Az általunk összeállított módszertani javaslat végig ezt a célt tartja szem előtt. Az elméleti tartalom és az alkalmazott módszerek kiválasztásánál számos tényezőt figyelembe vettünk. Megvizsgálva az előző és az aktuális tanévek tantárgyait [Depb] azt láttuk, hogy a Software technológia [DB] és a Csoportos projekt [DCS] tantárgyak a legalkalmasabbak az agile és a scrum gyakorlati megközelítésére. Míg az előző tantárgy az elméleti tartalmat tárgyalja, a második lehetőséget nyújt a gyakorlatra. A hallgatókkal végzett kutatás során és a félév végén kitöltetett kérdőív eredményeiben visszatér az a gondolat, amely az agilitás/scrum bevezetésének időszerűségét igazolja vissza: a résztvevők gyakran jelzik, hogy a nyári szakmai gyakorlat alatt szükségük lett volna erre a tudásra, és hogy jó lenne ezekkel a témákkal már másodéven foglalkozni.

A következő lépés a fent említett tantárgyakért felelős tanárokkal való egyeztetés volt, együtt határoztuk meg a hallgatókkal végzett gyakorlat hosszát és gyakoriságát.

4.1. A gyakorlat célja

Az általunk végzett gyakorlat elsődleges célja az agile és a scrum elméleti és gyakorlati ismertetése a Közös projekt című tantárgy keretein belül. Hosszabb távon megvizsgáltuk az általunk javasolt módszer hatékonyságát és fenntarthatóságát.

4.2. A gyakorlat tartalma

A Software technológia és a Közös projekt tantárgyak szerkezetéhez igazodva a gyakorlat tartalmát az Agile Manifesto és a scrum elmélete alapján állítottuk össze. Az egyes elemek a következők: az agilitás kulcsfogalmai, scrum szerepkörök, eventek és artifactek, ezek beazonosítása a hallgatók projektjeiben, az eventek eljátszása a csapatokkal közösen, a csapatok eredményeinek optimalizálása a gyakorlat ismételt alkalmazásával.

4.3. Használt pedagógiai módszerek

Ahhoz, hogy a találkozásokat dinamikusabbá tegyük, számos pedagógiai módszert alkalmaztunk. Lehetőségekhez mérten az *előadást beszélgetéssel* és *interakcióval* helyettesítettük. *Hasonlatokkal* emeltük ki a scrum erősségeit a hagyományos módszerekkel szemben. Az elméletet *metaforákkal* és *analógiákkal* tettük szemléletesebbé, illetve ezek révén kötöttük őket össze gyakorlati témákkal. A visszatekintések (retrospektívák) során *játékokat* alkalmaztunk, hogy elősegítsük a csapatok önreflekcióját. *Ismétlés* és *gyakorlati alkalmazás* révén mélyítettük el a scrum események menetét. A csapatokat *viták* révén biztattuk arra, hogy megoldásokat találjanak a felmerült problémákra.

Egyike a gyakran alkalmazott analógiáknak a user story-k felbontására vonatkozott. A hallgatók hajlamosak voltak a feladatokat komponensekben elképzelni: adatbázis réteg létrehozása, felhasználási felület megtervezése stb. Ennek elkerülése végett a projektet egy tortához hasonlítottuk, melynek legalább három rétege van; amennyiben a tortából ki szeretnénk venni egy szeletet, függőlegesen vágjuk fel, nem vízszintesen, lapjaira bontva. Ezt a példát alkalmaztuk a hallgatók projektjeire is, melyekben az adatbázis réteg nem jelent önálló funkcionalitást, a felhasználó számára használható, különálló értéket. Fontos ugyan a projekt egészének működéséhez, a jó user story viszont más szemszögből közelíti meg a projekt felbontását.

Az általunk alkalmazott retrospektíva-játékok [Ker] [DL06] mellett egy saját gyakorlatot is kifejlesztettünk, mely alkalmazkodik az egyetemi élet kereteihez. Egy űrhajót rajzoltunk fel, mely a csapatot képviselte; a projekt által elérhető jegyeket az űrbeli elemek jelentették: az atmoszféra az ötös, a Föld körül keringés a hatos, a csillagok a hetes-nyolcas-kilences jegyeket jelentették, míg a tízes a Holdra került. A csapat tagjai meg kellett mondják, hogy szerintük a projektjük az aktuális pillanatban meddig jutna fel, illetve mit adhatnának még az üzemanyaghoz ahhoz, hogy magasabbra szállhasson. Ezzel a feladattal azonosítottuk be azon lépéseket, melyek az utolsó száz méteren még értéket adhattak a projekthez, a bemutató hetében.

4.4. A találkozások száma és gyakorisága

Mivel egy újonnan bevezetett kísérleti gyakorlatról volt szó, igyekeztünk a diákok már meglévő órarendjéhez igazodni. A tanárokkal való előzetes egyeztetés után leszögeztük, hogy a Közös projekt tantárgy laboralkalmain kerül sor a találkozásokra. Ahhoz, hogy minden scrum eseményt szemléltethessünk, a félévet háromhetes futamokra osztottuk, a hallgatókkal való találkozások pedig a következőképp alakultak:

- 4. hét: első találkozó, hossza 1 óra.
- 7. hét: második találkozó, hossza 45 perc.
- 10. hét: harmadik találkozó, hossza 45 perc.
- 13. hét: negyedik (és egyben utolsó) találkozó, hossza 45 perc.

4.5. Előfeltételek

A gyakorlat előfeltételei nem különböznek más tantárgyakétól. Csapatonként egy laptop szükséges ahhoz, hogy futam végén a kész terméket bemutassák, illetve hogy a termék kívánságlistát napirendre hozzák. Amennyiben különleges eszközök szükségesek a visszatekintés lebonyolításához, azokról a scrum mester gondoskodik.

4.6. A hallgatók értékelése, a gyakorlat eredményeinek mérése

A gyakorlat eredményeit különféle metrikákkal vizsgáljuk, ezekre a következő fejezetben térünk ki. A gyakorlaton részt vevő hallgatók félév végén egy kérdőívet töltöttek ki. A tőlük begyűjtött visszajelzések csakis az általunk alkalmazott módszer hatékonyságát voltak hivatottak vizsgálni. Az agileról és a scrumról szerzett elméleti tudást a Software technológia tantárgy egyik elméleti vizsgája mérte fel.

4.7. Fejlesztett kompetenciák

A gyakorlat figyelmet helyez mind a szakmai, mind a transzverzális kompetenciák fejlesztésére, a Software technológiák és a Csoportos projekt tantárgyakkal szinkronban. Ezek: a szoftver technológiák módszerei és gyakorlatai mögött rejlő kapcsolatok és érvek megértése, kritikus gondolkozás alkalmazása egy szoftvertermékre vonatkoztatva, a tanult módszerek alkalmazása problémamegoldás céljából, olyan eredmény felmutatása, amely csapatmunka eredménye, határidők és minőségi megkötések betartásta, valamint problémák felismerése és megoldása mind egyéni, mind pedig csapatmunkán keresztül. [DCS] [DB]

4.8. A gyakorlat tartalma

A gyakorlat során átadott tartalom a négy találkozási alkalom között oszlik el.

4.8.1. Első találkozás: bevezetés és az első sprint planning (4. hét)

A bemutatkozást egy rövid érvelés követte, amelyben elmagyaráztuk a hallgatóknak, hogy mi a célunk a velük közösen végzett gyakorlattal. Felvázoltuk a félév menetét és az irányukba való elvárásainkat. A kérdések megválaszolása utáni kötetlen beszélgetésben megfigyeltük, hogy az adott csapat mennyire ismeri a projektet, amelyen a Csoportos projekt tantárgy keretén belül dolgozni fog. Felmértük, hogy hányan találkoztak már az agile és a scrum terminusaival, ezt követően pedig megtörtént a bevezetés az agilis módszerek és a scrum elméleti fogalmaiba. Miután letisztáztuk a kulcsfogalmakat, megpróbáltuk beazonosítani az első feladatokat, amelyeket a csapatok bevezettek a product backlogba. A találkozás zárásaként a csapatok megfogalmazták, hogy három hét múlva mit szeretnének készen látni a projektből.

4.8.2. Második találkozás: első review és retrospektíva, második sprint planning (7. hét)

A második találkozót ismétléssel indítottuk, amelyben felmértük, hogy a hallgatók mennyire jegyezték meg az elméleti részt. A homályos fogalmak esetében átvettük azok helyes jelentését, majd visszacsatoltunk a megelőző találkozás utolsó mozzanatához: mit szerettek volna készen látni akkor a következő alkalomra. A következő lépésben megtartottuk a daily scrumot, amely során mindegyik csapattag válaszolt az előírt kérdésekre, csak nem egy nap, hanem három hét spektrumában. Az elkészült munka bemutatása következett, ezután pedig az első visszatekintésre került sor. Négy kérdés segítségével (*Mi ment jól? Mi ment kevésbé jól? Mit tanultam? Mi az, ami még kérdéses számomra?*) begyűjtöttük az elmúlt sprint során felmerült gondokat, majd megpróbáltunk megoldási javaslatokat keresni. Az utolsó pont a termék kívánságlista aktualizálása volt, a csapattagok újból megfogalmazták, hogy mit szeretnének készen látni három hét múlva.

4.8.3. Harmadik találkozás: második review és retrospektíva, harmadik sprint planning (10. hét)

Az előző találkozáshoz hasonlóan épült fel, a retrospektívához használt játék kivételével. Első lépésben minden csapattag jegyet adott az elmúlt futamnak, 1-től 10-ig, majd ezeket átlagolva megfigyeltük a csapat morálját. Ezt követően minden csapattag papírlapokat kapott, amelyekre pontszerűen leírták, hogy mi ment jól az elmúlt sprintben, illetve mit tehettek volna másképp. Miután mindenki elkészült, két oszlopba rendeztük az eredményt, majd pedig a csapat megszavazta a legfontosabb témákat, ezekre próbáltunk utána megoldást keresni.

4.8.4. Negyedik találkozás: harmadik review és retrospektíva, végszó (13. hét)

A daily scrum és a review után három kérdésre kerestük a választ: *Mit tanultam ebből a projektből? Mit tennék másképp, ha újrakezdhetném? Mi az, amit a jövőben is felhasználhatok?* Végül pedig, egyben a tantárgy végi bemutatóra is készülődve egy saját fejlesztésű játékkal zártunk. A játék végén a csapatok megfogalmazták, hogy min tudnának még javítani a végső vizsgabemutató előtt. Megköszöntük nekik a segítséget, emlékeztetve őket, hogy a gyakorlat utolsó mozzanata a kérdőív kitöltése lesz.

5. Alkalmazott metrikák a kutatás elemzésében

A hallgatókkal végzett gyakorlat során és az azt követő időszakban számos szempontból vizsgáltuk az általunk kidolgozott módszer alkalmasságát. Az esetek többségében a vizsgált egység a csapat, hiszen a scrum is azt helyezi a figyelem középpontjába. Időnként előfordul azonban, hogy az egyéni vélemény is előtérbe kerül.

5.1. A találkozások alatt végzett mérések

Néhány szempontot már a négy találkozási alkalom alatt is vizsgáltunk, annak érdekében, hogy megfigyelhessük a változásokat bizonyos scrummal kapcsolatos vonatkozásokban.

1. Minden alkalommal megfigyeltük: csapatonkénti résztvevők száma.
2. Minden alkalommal megfigyeltük: részt vesznek-e aktívan a gyakorlat menetében?
3. Minden alkalommal megfigyeltük: a sprint megtervezése végén elkötelezték-e magukat egy adott cél mellett?
4. Minden alkalommal megfigyeltük: a sprint review-ra sikerült-e teljesíteni azt, amit célként tűztek ki maguknak?
5. Minden alkalommal megfigyeltük: a sprint review-kor van-e működő alkalmazásuk?
6. Minden alkalommal megfigyeltük: a retrospektíva után sikerül megfogalmazzanak action item-eket (értéknövelő teendőket)?
7. Minden alkalommal megfigyeltük: a daily scrumkor sikerült megfogalmazniuk, behatárolniuk az egyéni munkájukat?
8. Első héten figyeltük meg: tudják-e, hogy miről szól a projektjük?
9. Első héten figyeltük meg: hallottak-e a scrumról?
10. Első héten figyeltük meg: létezik-e egy verziója a product backlognak?
11. Második héten figyeltük meg: melyek a leggyakrabban megjegyzett szavak a scrummal kapcsolatban?
12. Harmadik héten figyeltük meg: milyen jegyet adtak az elmúlt sprintnek?
13. Negyedik héten figyeltük meg: milyen jegyet adtak az elmúlt sprintnek?
14. Negyedik héten figyeltük meg: milyen osztályzatot adnának a saját munkájukra, ha most kellene bemutassák?

15. Negyedik héten figyeltük meg: mit csinálnának másképp, ha újrakezdhetnék?

16. Negyedik héten figyeltük meg: mit jegyeztek meg tanulságképpen maguknak a projekt végén?

17. Negyedik héten figyeltük meg: mi az a tudás, amit majd máshol is felhasználhatnak?

18. Negyedik héten figyeltük meg: csapatonkénti átlag részvételi arány a négy találkozás után.

A scrum keretén belül fontos értéket jelent a csapaton belüli kommunikáció és az egyéni visszajelzés. Ahhoz, hogy ezeket elősegítsük, a visszatekintési alkalmakra retrospektíva-játékokkal készültünk, amelyek előhozzák a csapatok gondolatait saját erősségeikről és gyengeségeikről. [DL06] [Ker] Ezen játékok kimenetele, ha nem is mérhető, de értékes információkat hordoz.

5.2. Saját fejlesztésű webalkalmazás segítségével begyűjtött adatok

A hallgatók által csapatban kivitelezett projektek a web alapú Github-ot [Gitb] és Gitlab-ot [ser] használták forráskód-menedzsmentre, verziókövetésre és a dokumentáció elkészítésére. A product backlogot is ezeken keresztül kezelték, a megfelelő vizuális felület segítségével (Waffle.io [Gitc], illetve Redmine [Red]). Az általunk fejlesztett web alkalmazás REST API hívások segítségével [Fie00] statisztikai adatokat kér le a verziókövető rendszerektől, amelyeket aztán grafikus módon jelenít meg. A rendszer részletes leírása a hatodik fejezetben olvasható, alább a begyűjtött adatokat részletezzük.

1. **Megfogalmazott user storyk száma.** Az agilis módszertanok egyik bevált gyakorlata, hogy az elvégzendő munkát a lehető legapróbb funkcionalitásokra bontja le. A hallgatók irányában is ez volt az elvárás a *Software technológiák* tantárgy keretén belül: az első user story-kat már a követelmények megismerése után bevezették a verziókövetőbe, majd ezt követően a review és planning alkalmakon újraértékelték ezeket, néhány esetben újakat vezettek be. A diákokat arra biztatták, hogy kövessék és monitorizálják a user story-k állapotát, hogy azok mindig valós képet nyújtsanak a projekt aktuális állapotáról.

   A user story-k állapotát Github és Gitlub issue-k/feladatok formájában követték. Az issue-k teljes száma projektenként segít megérteni, hogy a hallgatók mennyire ültették gyakorlatba a feladatok atomi részekre való bontását. Megfontoltuk a mérföldkövek és a verziókiadások követését is, de mivel ez nem volt követelmény a *Software technológiák* keretén belül, eltekintettünk ezek megfigyelésétől.

2. **Commitok száma sprintenként.** Ennek a metrikának a segítségével felmérjük, hogy a hallgatók mekkora mértékben vették figyelembe a sprint adta időkeretet a munka elvégzésénél, illetve mennyire próbálkoztak elkészülni valamilyen működő funkcionalitással a sprint review-ra. Ideális esetben már az első sprintben látunk commitokat, hiszen a scrum

keretén belül a csapat már az első sprint végére működő terméket igyekszik felmutatni. [SS16]

3. **Commitok dátuma sprintenként.** A fent említett szám továbbelemzésével megfigyelhetjük, hogy miképp oszlanak el a commitok az egyes sprinteken belül. Arra számítunk, hogy a commitok száma megnő, ahogy közeledünk a sprint review dátumához.

4. **Fejenkénti commitok száma a projekt teljes hossza alatt.** A scrum keretén belül elméletileg nem nyílik lehetőség dedikált szerepek kiosztására. [SS16]. A csapattagokat arra biztattuk, hogy user story-kban gondolkozzanak, ne vízszintesen tagolt komponenseken, illetve hogy próbáljanak meg a saját munkájukra is reflektálni, ne mindig a csapat állapotára. A daily scrum és a sprint planning adták meg a keretet ahhoz, hogy ezekről a témákról beszéljünk. Ezzel a metrikával azt szeretnénk mérni, hogy egy adott csapaton belül az egyes tagok mennyire vették ki a részüket a fejlesztési feladatból, noha előfordulhat, hogy az eredmény mélyebb analízist von maga után.

5. **Fejenkénti commitok száma az egyes sprintek keretén belül.** Az előző esethez hasonlóan azt próbáljuk megfigyelni, hogy mennyire igyekeztek az egyes hallgatók hozzáadni munkájukat az egészhez, a csapat céljának elérése érdekében, különösen a sprintek végéhez közeledve.

6. **Sprintenként bezárt issue-k száma.** Ennek a metrikának a segítségével két dolgot figyelünk: mennyire igyekeztek a csapatok befejezni egy munkát a sprint végefelé közeledve, illetve mennyire volt hatásosan alkalmazva az atomi feladatokra bontás elve. Részletesebb elemzés során összevethetjük a betervezett feladatokat a valóságban bezártakkal.

7. **Feladatok állása a projekt végén.** Megfigyelhetjük, hogy az összesen létrehozott issuek közül hány került bezárásra és hány maradt nyitva projekt végén. Ez az információ összevethető a projekt végleges bemutatóján kapott jeggyel.

5.3. Kérdőív

A hallgatók számára összeállított kérdőívvel az volt a célunk, hogy felmérjük az általános hozzáállást az agile/scrum témakörökhöz és az általunk végzett gyakorlathoz. Kérdéseinket négy témakörbe csoportosítottuk:

- **Tanult fogalmak**. Kíváncsiak voltunk arra, hogy a Csoportos projekt és a velünk végzett gyakorlat együtteseként sikerült-e olyan új tudást átadni, amely nem technikai, mennyire volt új számukra mindaz, amit az agile/scrum kapcsán hallottak, illetve amennyiben cég által mentorált projekten dolgoztak, fektettek-e ott külön hangsúlyt az agile és scrum elemeire.

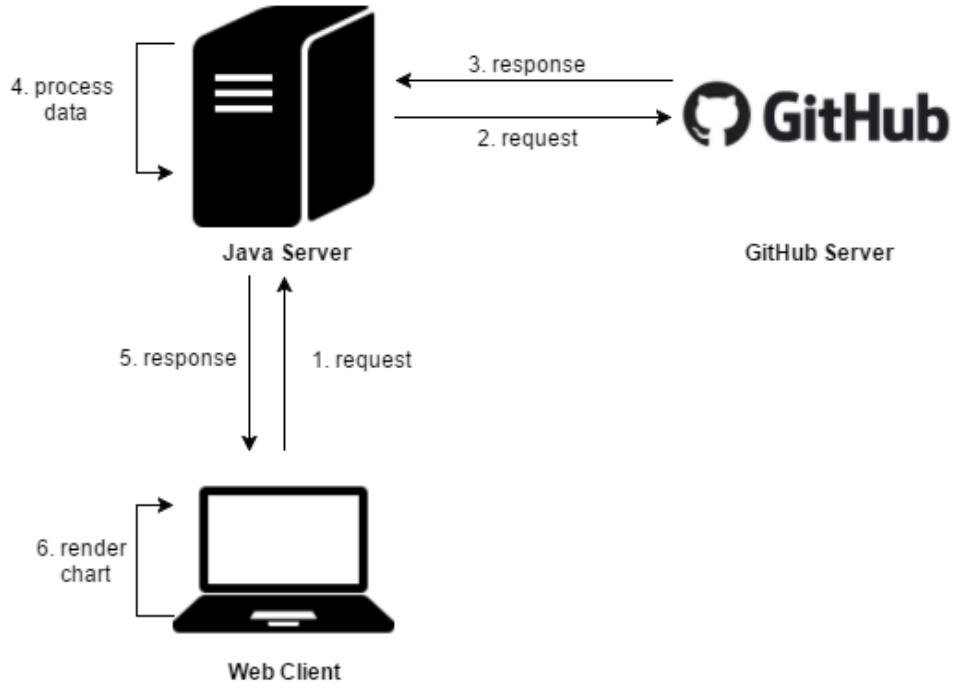
- **Az egyetemi gyakorlat felépítése**. Az alkalmak hosszáról, gyakoriságáról és időpontjairól, a használt szerepkörökről, illetve arról kérdeztünk, hogy szerintük mikor lenne időszerű az agile/scrum kérdéskörével foglalkozni.

- **Teljesítmény értékelése**. Ebben a részben a csapatra, mint egységre, a közös munkára és a scrum által nyújtott fejlődési lehetőségekre összpontosítottunk.

- **Megjegyzések, üzennivalók**. Itt nyílt lehetőség olyan visszajelzés küldésére, amelyet a fenti kérdések nem fedtek le, szabad formátumú szövegben.

5.4. Rokon tantárgyakon elért eredmények

A negyedik fejezet során említett tantárgyak (Csoportos projekt és Software technológiák) saját követelmény- és értékelési rendszerrel rendelkezne, mindkét tantárgy kezel agilitás- és scrum-közeli témaköröket. Míg az első tantárgy végén a projektek gyakorlati bemutatására került sor, az utóbbi egy elméleti vizsga segítségével mérte fel a hallgatókat. A Csoportos projekt tantárgy nem csak a kész terméket pontozta, hanem azt is, hogy a csapat mennyire hatékonyan dolgozott együtt, mennyire sikerült megoldaniuk a csapaton belül felmerülő gondokat. A két tantárgy során szerzett jegyek összevethetőek az általunk végzett megfigyelésekkel, ami a csapatmunkát és a kommunikációt illeti.



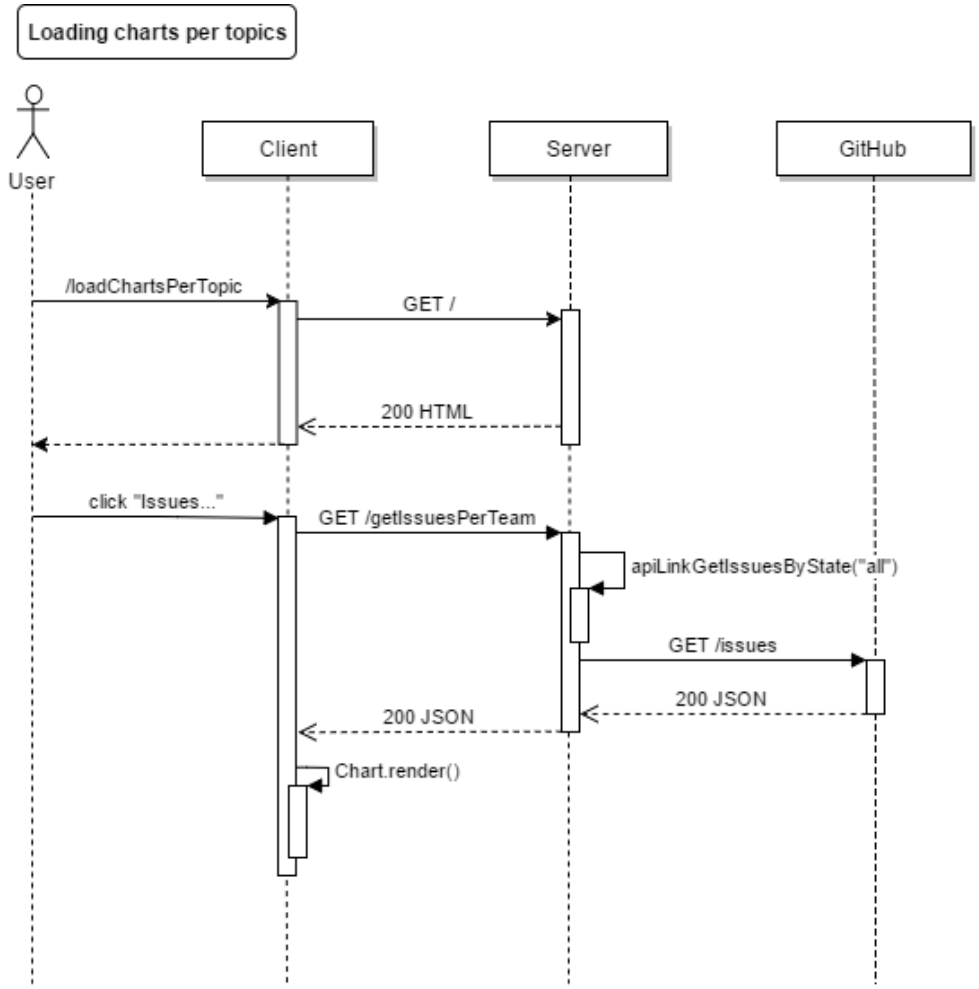
3. ábra. Kommunikációs diagram a TheAgileExperiment projekthez

6. A mérésekhez használt webalkalmazás bemutatása

A találkozási alkalmakon, kérdőíveken és vizsgaeredményeken kívül értékes információ rejlik azokban a verziókövető rendszerekben, amelyeket a csapatok a Csoportos projekt tantárgy teljes futamideje alatt használtak. Ideális esetben minden munka ide kerül feltöltésre, gyakran, rendszeresen és lehetőleg arányosan megosztva a csapat tagjai között. Az 5.2-es alfejezetben kifejtettük, hogy melyik adat miért hasznos számunkra. Ahhoz, hogy ezeket gyorsan begyűjthessük és látványosan szemléltethessük, web alkalmazás fejlesztése mellett döntöttünk.

A projekt alapját a **Spring Boot** szolgáltatta [STa]. Konfigurációigénye alacsony, használata egyszerű, nagyon gyorsan futtatható alkalmazáshoz jutunk általa. Beépített szerverének köszönhetően gyorsan indul és nincs szükség külön telepítési folyamatokra. A Spring Boot lehetőséget ad a **Spring MVC** keretrendszer automatikus konfigurációjára [STb], amit mi ki is használunk. A Controller osztályok részén történik az adatok lekérése és a feldolgozás. A View részen, a megjelenítéshez **JSP oldalakat**[Ora] alkalmazunk és a **CanvasJS** grafikon-rajzoló, Javascript alapú függvénykönyvtárat [Can] használjuk. A RESTful webszolgáltatás [Fie00] lehetőséget ad arra, hogy bizonyos rendszerektől szöveges adatokat kérjünk le, HTTP protokollra alapuló hívások révén. A számunkra értékes adatokat a verziókövetők **REST API**-ján keresztül kérjük le [Gita].

Az adatok lekérésének folyamatát a 3. ábra szemlélteti. Noha a kliens oldalon a webes felület statikus része első híváskor betöltődik, a kérés a verziókövető rendszer felé csak a felhasználó explicit kérésére lesz továbbítva. A verziókövetőtől megkapott és a szerver oldalon feldolgozott válasz megjelenítésre készen visszakerül a kliens oldalra, ahol a felület aszinkron

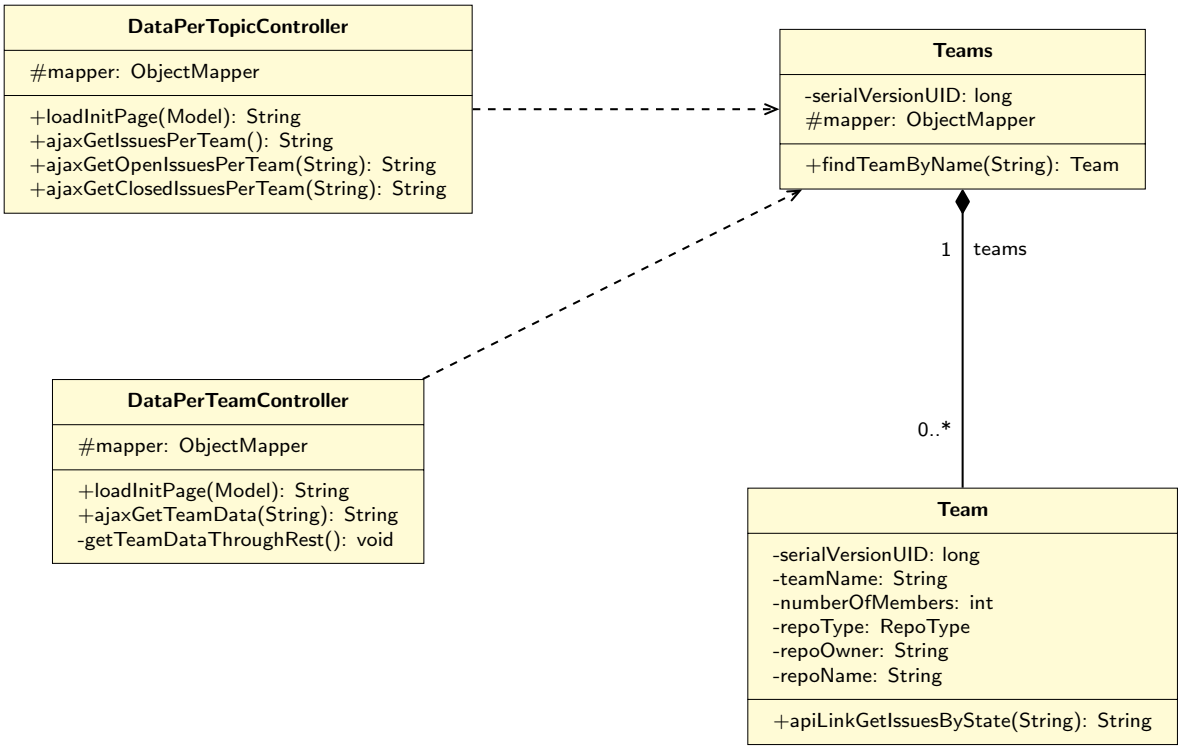


4. ábra. Szekvenciadiagram a TheAgileExperiment projekthez

módon frissül, és kirajzolódik a megfelelő grafikon.

A 4. ábrán látható szekvenciadiagram ugyanezt a kommunikációs folyamatot hivatott ábrázolni, egy konkrét esetben. Jelenlegi állás szerint a kliens két oldalhoz fér hozzá. Az egyiken (`/loadChartsPerTeam`) a csapatok közül választhat egyet; a felületen lévő gombra kattintva a rendszer előkészíti a kiválasztott csapat projektjének megfelelő REST API hívást, elküldi, majd visszatéréskor az adott csapatra vonatkozó információkat megjeleníti a felületen. A második oldal (`/loadChartsPerTopic`) adott témákra vonatkozó adatokat jelenít meg. Ha a felhasználó kéri egy ilyen grafikon megjelenítését, a háttérben minden csapat számára felépül a REST API híváshoz szükséges link, majd ezek rendre meghívódnak. Az összesített adatokból készül el a válasz, ami végül megjelenik a felületen is. Ezt szemlélteti részletesebben az 5. ábra, ahol az `apiLinkGetIssuesByState("all")` felelős a linkek felépítéséért.

Az 5. ábra betekintést nyújt az osztálystruktúrába. Mindkét Controller osztály egy `Teams` objektumot tartalmaz, amely inicializáláskor fel lesz töltve a projektünkben aktuális csapatokkal, egy szöveges állományból. Mivel esetünkben konkrétan tudjuk, hogy milyen csapatokkal dolgoztunk, amellett döntöttünk, hogy a csapatok adatait statikusan tároljuk, JSON formátumú szöveges állományokban. A `Teams` osztályra azért is volt szükség, mert a szöveges állomány-



5. ábra. Osztálydiagram a TheAgileExperiment projekthez

ban tárolt, JSON formátumú csapatlistát nem lehet a Java generikus lista típusába átültetni. A `Team` osztály nem csak a statikus adatok betöltésekor hasznos, ezek teljes élettartama alatt lehetőségünk nyílik a REST API hívások során visszakapott adatok csapatokhoz kötéséhez. A továbbiakban megfontolandó új osztályok létrehozása aszerint, hogy a verziókövető rendszerektől milyen adatokat kérünk le (pl. Issue, User, Label, Commit stb.). Ezeknek az absztrahálásával is számolunk abban az esetben, ha a jelenleg implementált Github-os megoldást kiterjesztenénk GitLab-ra és más verziókövetőkre, amelyek publikus REST API-val rendelkeznek.

7. Eredmények elemzése

A négy találkozási alkalmon lejegyzett észrevételekből számos dolgot megfigyelhetünk. Az első találkozóra a negyedik és ötödik egyetemi heteken került sor. Az akkor feltett kérdésekből kiderült, hogy a tizenhárom megkérdezett csapatból csak hat vélte úgy, hogy ismeri a saját projektjét, és csak ötnek volt kezdeti verziójú product backlog-ja. Hat csapatban voltak olyan tagok, akik ismerték a scrumot vagy valamilyen más agilis módszert. A találkozó végére csak hat csapat fogalmazott meg magának arra vonatkozó célt, hogy mit szeretne készen látni majd három hét múlva. Azonban ezeknek a megfogalmazott céloknak is egy része tanulási folyamatokra vonatkozott.

A második találkozóra már elkészültek a backlogok. Hét csapat rendelkezett működő incrementtel, a planning végén viszont már mindenkinek volt megfogalmazott célja a következő három hétre. A leggyakrabban megjegyzett, scrummal kapcsolatos szavak a *sprint, planning, tervezés, spike, backlog, user story, mérföldkövek, daily standup*. Az előző alkalomhoz képest a hallgatók sokkal aktívabban vettek részt a beszélgetésben.

A harmadik találkozókor egyetlen olyan csapat volt, amely a teljes kitűzött munkaadaggal elkészült. A többi csapat, egy kivételével, rendelkezett működő funkcionalitással, még ha nem is azzal, amelyet eredetileg kitűztek maguknak. Két csapattal nem sikerült találkozni. A standup során előfordult, hogy egy-egy csapat nem tudta elválasztani az egyéni munkát a csapat haladásától. Az elmúlt sprintet értékelő kérdésre csapatonként eltérő válaszokat kaptunk: egy esetben az átlag 9-es felett volt, a legborúlátóbbak átlaga pedig 4-est eredményezett. Mindenki tűzött ki célt maga elé, ez általában már a projekt végső állapota felé konvergált.

Az utolsó alkalomra a téli vakáció után került sor. Noha mindegyik projektnek volt működő változata, egyik sem felelt meg teljes mértékben annak, amit a csapatok az elmúlt planning alkalmával célként kitűztek. A jól haladó csapatok hangulatát tükrözték a sprint-értékelő pontok, a lassabban haladók jegyein meglátszott az aggodalom. Arra a kérdésre, hogy mit tanultak a projektből, a technikai tudás mellett a csapatmunka fontosságát, a munka megtervezését, annak időben való elkezdését, az alapos tesztelés elengedhetetlenségét, a fontosabb dolgok prioritását emelték ki, valamint azt, hogy az ismeretlen faktort nem szabad alábecsülni. Ha valamit a tapasztalatok alapján másképp csinálhatnának, akkor korábban letisztáznák a követelményeket, hamarabb nekifognának az effektív munkának és alaposabban tesztelnének. A retrospektíván a projektjüket többnyire 5-6-os jegyre értékelték, két csapat magabiztosabban 8-9-esre számított.

A négy találkozóra visszatekintve megfigyeltük, hogy a hallgatók egyre nyitottabban kommunikálnak, mernek beszélni a munkájukról, felismerik erősségeiket és gyengeségeiket. Sajnos a zsúfolt egyetemi órarend és a vizsgaidőszak közelsége miatt az utolsó találkozón alacsony volt a részvétel.

A kérdőívek elemzése során kiderült, hogy a kitöltők 83,5%-a számára a projekt nyújtott olyan tudást is, amely nem feltétlenül technikai jellegű. A leggyakrabban említett kulcssza-

vak: időbeosztás, csapatmunka, együttműködés és kommunikáció. A megkérdezettek 62%-a hasznosnak, vagy nagyon hasznosnak tartotta a találkozókon elhangzott elméleti információkat. Csak 18% válaszolta azt, hogy nem volt számára új az agile/scrum-témakör: nyári szakgyakorlaton, vagy munkahelyen már találkozott vele. A gyakorlat felépítését tekintve a válaszadók 75%-a megfelelőnek találta az alkalmak időpontját, hosszát és a három hetes sprint terjedelmét. A scrum master személyét illetőleg megoszlanak a vélemények: 58,2% maradna a külsős megoldás mellett, 27,8% egy állandó embert szeretne a csapat tagjai közül, míg a többiek számára az jelentené a megoldást, ha sprintenként változna a scrum master személye.

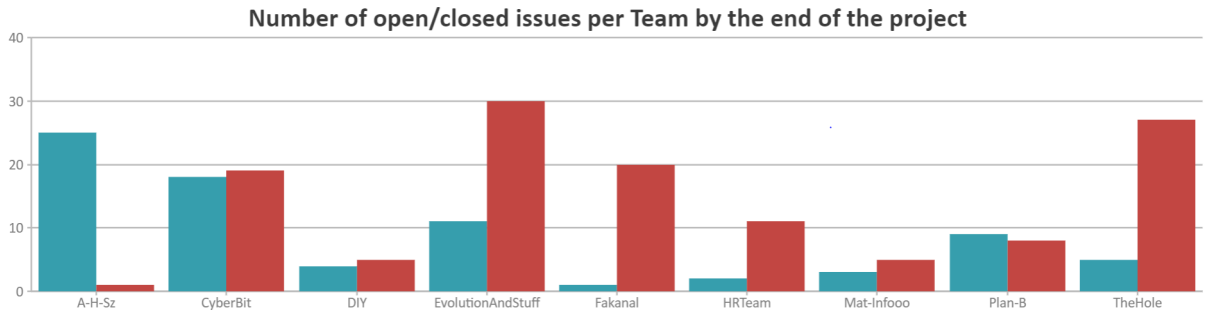
Ami a csapat teljesítményét illeti, a válaszadók 88,6%-a sikeresnek érzi a csapatuk által megvalósított projektet. A csapat teljesítményét 68,4% tartja jónak, vagy nagyon jónak, míg az egyéni teljesítményt sokkal kritikusabban szemlélik. A retrospektíva és a planning hasznosságát tekintve a válaszadók 62%-a jelez jót vagy nagyon jót, a semleges válaszok aránya 24%, és 14% nem látta hasznát ezeknek. A szabad szöveges visszajelzések során többen is kihangsúlyozták, hogy a gyakorlat hasznos volt és hogy érdemes lenne az elkövetkezőkben is folytatni azt. Olyan válasz is érkezett, amely szerint a gyakorlat nem kivitelezhető a hallgatók túlzsúfolt órarendje és számos kötelezettsége miatt.

Az általunk fejlesztett webalkalmazás jelenlegi állapotában olyan adatok megjelenítésére alkalmas, amelyek a csapatok statisztikáiból lettek összesítve. Kiemeljük, hogy jelenleg csak a GitHub-bal dolgozó csapatok adataihoz férünk hozzá, a GitLab REST API-ja más jellegű megközelítést feltételez.

A 7. ábrán megfigyelhető, hogy a csapatok eltérő számú issue-kkal dolgoztak. Míg a Mat-Infooo csapatnak csak 8 issue-ja volt a projekt teljes futamideje alatt, addig a TheHole, az EvolutionAndStuff és a CyberBit csapatok 30-at hoztak létre.

Ha az issue-k összességét állapot szerint lebontjuk (*open/closed*), megfigyelhetjük, hogy hány feladatot zártak be a csapatok, illetve mennyi maradt nyitva a projekt végéig. A kék oszlop jelzi a nyitvahagyott, a vörös pedig a bezárt issue-k számát.

Ha összevetjük az ábrát és a rendelkezésünkre álló félévközi megfigyeléseket, néhány összefüggést vélünk felfedezni. Az **A-H-Sz** csapat, amely 26 issue-val dolgozott, csak egyet zárt be a projekt végéig. Jól haladtak, viszont a találkozásokkor nem azzal készültek el, amit azelőtt elterveztek; cég által mentorált projekten dolgoztak, melynek gyakran változtak a követelményei a félév során. A **CyberBit** bezárta a létrehozott issue-k szinte felét, de ha a találkozók során bemutatott projektrészleteket figyeljük, akkor feltűnik, hogy a munkájuk csak a félév végére kezdett működő egésszé összeállni. A **DIY**-vel elmaradt egy találkozásunk, a meglévő alkalmakon pedig gyakran panaszkodtak arra, hogy időhiány miatt nem tudtak haladni, és hogy egy adott pontban sokáig el voltak akadva. Látványosabb a bezárt/nyitott feladatok aránya az **EvolutionAndStuff**, a **Fakanál**, a **HRTeam** és a **TheHole** csapatok esetében. Ők jobban tartották magukat az eltervezett célokhoz, noha teljes mértékben egyik csapatnak sem sikerült betartania őket. Ahogy azt a létrehozott és bezárt feladatok is tükrözik, a **Plan-B** és a **Mat-Infooo** csapa-



6. ábra. A projekt során létrehozott issue-k száma aszerint, hogy hány maradt nyitva és hány lett bezárva, csapatonként leosztva

tok nehezebben lendültek bele a fejlesztésbe. A találkozókon is gyakran előjött az a gond, hogy egyes csapattagok dolgoztak valamin, de nem hozták azt a többiekkel egységes szintre.

8. Következtetések és további lépések

A legegyszerűbben értelmezhető visszajelzést a hallgatók megjegyzéseiből kaptuk: van haszna annak, hogy az agile/scrum témakörökkel gyakorlati módon foglalkozzunk. Az az ötlet sem elhanyagolandó, mely szerint érdemes lenne már másodéven beszélni ezekről a témákról.

Az egyik eredeti célunk az volt, hogy megkeressük az agile/scrum tanítására nyíló lehetőségeket a meglévő kereteken belül. Az egyszerű tény, hogy kiviteleztük eredeti tervünket, azt bizonyítja, hogy lehetséges. Kérdés marad, hogy miben változtassunk az eredeti módszertani javaslaton a hallgatók és a tanárok visszajelzései alapján. Ami világos már mostantól, hogy szükség van egy olyan személyre, aki összehangolja a scrum gyakorlat menetét, illetve folyamatosan kapcsolatban áll a termékgazdákkal. Az általunk alkalmazott felállás egyetlen dedikált külsős személyt helyezett a scrum master szerepébe. A hallgatók véleménye megoszlott erről; a többség ugyan továbbra is a külsős scrum master megoldást javasolta, számottevő arány szavazott arra, hogy a feladatot betöltő személy a csapatból kerüljön ki, és annak a változatnak is akadtak pártolói, mely szerint a feladat felelőse sprintenként változzon.

Ahhoz, hogy egy számszerűbb eredményt kapjunk, a jövőben szükséges részletesebben beleásnunk magunkat a rendelkezésünkre álló adatokba, illetve kihasználnunk a verziókövetők nyújtotta lehetőségeket. A következő lépésekben tervezzük a GitHub API felé küldött kérések kibővítését, illetve, hogy a megoldást implementáljuk a GitLab API-jának szintjén is. Továbbá egy OAuth klienset szeretnénk integrálni kliens oldalon annak érdekében, hogy globálisan beléphessünk a GitHub, GitLab, Redmine oldalakra. Mivel minden egyes grafikon-generálás több kérést is küld a verziókövetők felé, egy olyan cache-elési mechanizmust tervezünk, melynek révén a szerverünk eltárolja a statisztikákat, és csak a kliens kérésére számolja őket újra. Jelenleg az alkalmazás a kutatásunkkal végzett csapatok tanulmányozására korlátozódik, de a jövőbeni felhasználhatóság érdekében olyan kiterjesztési lehetőséget is számon tartunk, mely végén tetszőleges Git repository-ról lekérhetjük majd az adatokat.

Előttünk áll még a tantárgyak során kapott jegyek összehasonlítása a félévi megfigyelésekkel és a hallgatók önértékelésével. Ugyancsak fontosnak tartjuk a kérdőív minden árnyalatát kielemezni. Mindezek után tervezzük a módszertan következő változatának elkészítését, melyet igény szerint következő tanévekben alkalmazni lehet.

Hivatkozások

[AAA+09] Rick Adcock, Edward Alef, Bruce Amato, Mark Ardis, Larry Bernstein, Barry Boehm, Pierre Bourque, John Brackett, Murray Cantor, Lillian Cassel, Robert Edson, Richard Fairley, Dennis Frailey, Gary Hafen, Thomas Hilburn, Greg Hislop, David Klappholz, Philippe Kruchten, Phil Laplante, Qiaoyun (Liz) Li, Scott Lucero, John McDermid, James McDonald, Ernest McDuffie, Bret Michael, William Milam, Ken Nidiffer, Art Pyster, Paul Robitaille, Mary Shaw, Sarah Sheard, Robert Suritis, Massood Towhidnejad, Richard Thayer, J. Barrie Thompson, Guilherme Travassos, Richard Turner, Joseph Urban, Ricardo Valerdi, Osmo Vikman, David Weiss, and Mary Jane Willshire. Curriculum guidelines for graduate degree programs in software engineering. Technical report, New York, NY, USA, 2009.

[All] Agile Alliance. Agile Glossary. https://www.agilealliance.org/agile101/agile-glossary/.

[AM15] Craig Anslow and Frank Maurer. An experience report at teaching a group based agile software development project course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 500–505, New York, NY, USA, 2015. ACM.

[BBB+] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. Manifesto for Agile Software Development. http://agilemanifesto.org/.

[Can] CanvasJS. CanvasJS Official Page. http://canvasjs.com/.

[DB] Babes-Bolyai University's Computer Science Department and László Barabás. Software technológia - a tantárgy adatlapja. http://www.cs.ubbcluj.ro/files/curricula/2016/syllabus/IM_sem5_MLM5011_hu_lbarabas_2016_1979.pdf.

[DCS] Babes-Bolyai University's Computer Science Department, Lehel Csató, and Károly Simon. Közös projekt - a tantárgy adatlapja. http://www.cs.ubbcluj.ro/files/curricula/2016/syllabus/IM_sem5_MLM5012_hu_csatol_2016_1984.pdf.

[Depa] Babes-Bolyai University's Computer Science Department. Computer science programme profile. http://www.cs.ubbcluj.ro/education/academic-programmes/undergraduate-programmes/computer-science-programme-profile/.

[Depb] Babes-Bolyai University's Computer Science Department. Tantervek a 2016–2017-es egyetemi tanévre. http://www.cs.ubbcluj.ro/tantervek-a-2016-2017-es-egyetemi-tanevre/.

[DH05] Yael Dubinsky and Orit Hazzan. A framework for teaching software development methods. *Computer Science Education*, 15(4):275–296, 2005.

[DIC+15] E. Durant, J. Impagliazzo, S. Conry, R. Reese, H. Lam, V. Nelson, J. Hughes, W. Liu, J. Lu, and A. McGettrick. Ce2016: Updated computer engineering curriculum guidelines. In *2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–2, Oct 2015.

[DL06] Esther Derby and Diana Larsen. *Agile Retrospectives: Making Good Teams Great*. Pragmatic Bookshelf, 2006.

[DM11] V. Devedzic and S. R. Milenkovic. Teaching agile software development: A case study. *IEEE Transactions on Education*, 54(2):273–278, May 2011.

[fCM] Association for Computing Machinery. Curricula recommendations. http://www.acm.org/education/curricula-recommendations.

[Fie00] Roy Fielding. Chapter 5: Representational state transfer (REST). http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, 2000.

[Gita] GitHub. Official GitHub API page. https://developer.github.com/v3/.

[Gitb] GitHub. Official GitHub page. https://github.com/.

[Gitc] GitHub. WaffleIO page for GitHub. https://waffle.io/.

[HD07] Orit Hazzan and Yael Dubinsky. Why software engineering programs should teach agile software development. *SIGSOFT Softw. Eng. Notes*, 32(2):1–3, March 2007.

[Kak14] A. K. Kakar. Teaching theories underlying agile methods in a systems development course. In *2014 47th Hawaii International Conference on System Sciences*, pages 4970–4978, Jan 2014.

[Ker] Norman Kerth. The key questions to be answered during a retrospective. http://www.retrospectives.com/pages/RetrospectiveKeyQuestions.html.

[KM13] M. Kropp and A. Meier. Teaching agile software development at university level: Values, management, and craftsmanship. In *2013 26th International Conference*

on Software Engineering Education and Training (CSEE T), pages 179–188, May 2013.

[KM14] M. Kropp and A. Meier. New sustainable teaching approaches in software engineering education. In *2014 IEEE Global Engineering Education Conference (EDUCON)*, pages 1019–1022, April 2014.

[KM16] M. Kropp and A. Meier. Collaboration and human factors in software development: Teaching agile methodologies based on industrial insight. In *2016 IEEE Global Engineering Education Conference (EDUCON)*, pages 1003–1011, April 2016.

[KMB16] M. Kropp, A. Meier, and R. Biddle. Teaching agile collaboration skills in the classroom. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 118–127, April 2016.

[KMMZ14] M. Kropp, A. Meier, M. Mateescu, and C. Zahn. Teaching and learning agile collaboration. In *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE T)*, pages 139–148, April 2014.

[LD11] Baochuan Lu and Tim DeClue. Teaching agile methodology in a software engineering capstone course. *J. Comput. Sci. Coll.*, 26(5):293–299, May 2011.

[Mah09] P. Maher. Weaving agile software development techniques into a traditional computer science curriculum. In *2009 Sixth International Conference on Information Technology: New Generations*, pages 1687–1688, April 2009.

[Mah10] Viljan Mahnic. Teaching scrum through team-project work: Students' perceptions and teacher's observations. *International Journal of Engineering Education*, 26(1):96–110, Jan 2010.

[Mah12] V. Mahnic. A capstone course on agile software development using scrum. *IEEE Transactions on Education*, 55(1):99–106, Feb 2012.

[Mah15] Viljan Mahnic. Scrum in software engineering courses: an outline of the literature. *Global Journal of Engineering Education*, 17(2):77–83, 2015.

[Ora] Oracle. JSP documentation. http://docs.oracle.com/javaee/5/tutorial/doc/bnagy.html.

[Red] Babeș-Bolyai University's Redmine. http://pdae.cs.ubbcluj.ro/redmine/.

[RS09] D. F. Rico and H. H. Sayani. Use of agile methods in software engineering education. In *2009 Agile Conference*, pages 174–179, Aug 2009.

[SCL11] Tucker Smith, Kendra M.L. Cooper, and C. Shaun Longstreet. Software engineering senior design course: Experiences with agile game development in a capstone project. In *Proceedings of the 1st International Workshop on Games and Software Engineering*, GAS '11, pages 9–12, New York, NY, USA, 2011. ACM.

[ser] Babeș-Bolyai University's GitLab server. http://pdae.cs.ubbcluj.ro/git/.

[SGK10] C. Scharff, O. Gotel, and V. Kulkarni. Transitioning to distributed development in students' global software development projects: The role of agile methodologies and end-to-end tooling. In *2010 Fifth International Conference on Software Engineering Advances*, pages 388–394, Aug 2010.

[SS16] Ken Schwaber and Jeff Sutherland. The Scrum Guide. http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf, 2016.

[STa] The Spring Team. Spring Boot Project. https://projects.spring.io/spring-boot/.

[STb] The Spring Team. Spring MVC auto-configuration. http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#boot-features-spring-mvchttp://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/#boot-features-spring-mvc.

[SWM10] Jonas Schild, Robert Walter, and Maic Masuch. Abc-sprints: Adapting scrum to academic game development courses. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 187–194, New York, NY, USA, 2010. ACM.

[TTT08] Chuan-Hoo Tan, Wee-Kek Tan, and Hock-Hai Teo. Training students to be agile information systems developers: A pedagogical approach. In *Proceedings of the 2008 ACM SIGMIS CPR Conference on Computer Personnel Doctoral Consortium and Research*, SIGMIS CPR '08, pages 88–96, New York, NY, USA, 2008. ACM.