Gradus Vol 1, No 1 (2014) 15-24
ISSN 2064-8014

BIZTONSÁGOS SCRUM VÁLTOZATOK ÁTTEKINTÉSE

SURVEY ON SECURE SCRUM VARIANTS

Johanyák Zs. Cs. [*], *Bolla K., Alvarez Gil R.P. and Halczman Sz.L.*

Informatika Tanszék, Gépipari és Automatizálási Műszaki Főiskolai Kar, Kecskeméti Főiskola, Magyarország

Kulcsszavak:
scrum,
agilis,
biztonságos,
szoftverfejlesztés

Keywords:
Scrum,
agile,
secure,
software development

Történet:
Beérkezett: 2015. október 10.
Átdolgozva: 2015. október 31.
Elfogadva: 2015. november 7.

Összefoglalás

A hatékony szoftverfejlesztés egyik alapfeltétele a megfelelő módszertan kiválasztása és alkalmazása. Napjaink egyik legelterjedtebb módszertana a Scrum, amit sok ezer projekt során alkalmaztak sikeresen. Számos előnyös tulajdonsága ellenére a Scrum gyenge pontja az, hogy nem kínál teljes értékű megoldást biztonságkritikus szoftverek fejlesztésére, pl. nem tartalmaz explicit módon biztonsági elemzést és tervezést. Cikkünkben két olyan módszertant (S-Scrum és Secure Scrum) is áttekintünk és értékelünk, ami az eredeti eljárást alkalmassá teszi biztonságkritikus szoftverek fejlesztésére újabb lépések és komponensek beépítésével.

Abstract

Selection and application of the proper methodology are basic requirements of efficient software development. Scrum is one of the most spread methodologies being applied in thousands of projects. Despite its numerous advantages its weak point is its lack in giving techniques for the development of security critical applications. In this paper, after presenting the original technique, we present and evaluate two methods (S-Scrum and Secure Scrum) that by defining new steps and components enhance the original Scrum to be able to fulfil the security demands as well.

1. Bevezetés

Életünk szinte minden területén jelen vannak az informatikai eszközök. Elektronikusan kommunikálunk, banki és egyéb ügyeket intézünk, fizetünk, és a szórakozástól a hétköznapi munkáig szinte mindenhol használunk szoftvereket. Az alkalmazott programok hozzáférnek számos bizalmas adatunkhoz, értékeinkhez, ami különös jelentőséget ad a biztonság iránti igénynek. Ezen szoftverek jelentős része nagy bonyolultságú, és együtt kell működjön, adatot kell cseréljen más rendszerekkel. Ezt a kockázati tényezőt még az a tény is erősíti, hogy rohanó világunkban mindent azonnal szeretnénk megkapni, ami a szoftverfejlesztő cégekre nehezedő erős határidőnyomást eredményez.

A hagyományos tervezés és dokumentálás alapú módszertani megközelítések nehezen vagy egyáltalán nem képesek alkalmazkodni a gyorsan változó felhasználói követelményekhez, a megrendelő bizonytalanságához és a rövid fejlesztési határidőkhöz. Ennek felismerése az agilis módszertanok (pl. [2][3][4][5][6]) megjelenését és elterjedését eredményezte a 90-es évek második felétől kezdődően. Bár már az ezredforduló környékén történtek kezdeményezések az egységesítés érdekében (pl. agilis kiáltvány [1]) de ezek akkor csak részben bizonyultak sikeresnek. Napjainkra a Scrum [6] vált legjobban elterjedt agilis szoftverfejlesztési módszertanná.

* Kapcsolattartó szerző. Tel.: +36 76 516 413; fax: +36 76 516 399
E-mail address: johanyak.csaba@gamf.kefo.hu

A Sutherland és Schwaber nevével fémjelzett keretrendszer alapgondolata abból a megfigyelésből indul ki, hogy új komplex termékek fejlesztése során jobb teljesítmény érhető el kis létszámú, önszerveződő, önálló csapatok felállításával úgy, hogy nem feladatokat, hanem célokat határoznak meg számukra.

Több mint húsz éves története során a Scrum keretrendszert sikeresen alkalmazták számos szoftver projektben, több mint ezer könyv témájául szolgált, valamint alkalmazást nyert olyan területeken is, mint a gyártás, marketing és oktatás [7]. Számos előnyös tulajdonsága és széleskörű elterjedtsége ellenére ennek a módszertannak gyenge pontja az, hogy nem kínál teljes értékű megoldást biztonságkritikus szoftverek fejlesztésére. Biztonsági szempontból jelentős hiányosság például, hogy nem tartalmaz explicit módon biztonsági elemzést és tervezést, valamint a fejlesztési tevékenységekhez nem kapcsolódik pontos dokumentálás [9]. Ezen hiányosságok felismerése olyan új Scrum változatok megjelenését eredményezte, amelyek lehetősé teszik a biztonsági kihívásoknak történő megfelelést is.

Cikkünkben két ilyen módszertan, az S-Scrum [8] és a Secure Scrum [10], fontosabb jellemzőit tekintjük át és értékeljük. A cikk további részének felépítése a következő. A 2. szakaszban az alap Scrum módszertan lényegi elemeit ismertetjük. A 3. és a 4. szakaszban az S-Scrum és a Secure Scrum változattal foglalkozunk, majd a következtetésekre és összegzésre az 5. szakaszban kerül sor.

2. Scrum

Ebben a fejezetben összefoglaljuk a Scrum módszer [6] meghatározó elemeit és röviden bemutatjuk annak működését a gyakorlatban. A Scrum kilenc elemből épül fel. Ezek a következők: három szerepkör, három értekezlet típus, és három dokumentum. A szerepkörök termékgazdára (Product Owner - PO), Scrum Master-re (SM) és a csapatra (Team - T) bontódnak szét. A Scrum-ban a szerepkörök úgy vannak kialakítva, hogy nincsen közöttük alá- és fölérendeltség, így aki részt vesz az adott projektben - a fejlesztőtől egészen az ügyfél kapcsolattartóig - mindenki egyenrangú. Az alábbiakban részletezzük az egyes szerepköröket betöltő emberek feladatait.

A termékgazda (PO) feladata az, hogy megtestesítse az ügyfél hangját, mivel az ügyfél a PO-val tartja a kapcsolatot. Erre a feladatra egy olyan személy a legmegfelelőbb, aki precíz, valamint képes szót érteni és megegyezni az ügyféllel. A PO tisztában kell legyen a fejlesztéshez szükséges időkerettel, és a fejlesztés által termelt bevétel mennyiségével. Egyik legfontosabb feladata, hogy megfelelően átlássa az üzleti és pénzügyi folyamatokat. Másik fontos feladata, hogy az elvégzendő munkát a fejlesztőkkel ismertesse, és elkészítse a fejlesztési igényeket tartalmazó táblázatot (Product Backlog - PB). A PB elkészítése során a teljes feladatot kisebb részfeladatokká alakítja, és fontossági sorrendet állapít meg. Amennyiben a feladatok szétbontásához nem rendelkezik elegendő információval, kérheti a Team segítségét. Mivel a PO és a Team egyenrangú, a fejlesztők is tehetnek javaslatokat a részfeladatok meghatározására.

A Scrum Master egyfajta ellensúlyt képez a PO-val szemben. Elsődleges feladata az, hogy képviselje a csapatot. Biztosítania kell azt, hogy a csapat előtt ne legyenek akadályok, és zavartalan legyen a munkavégzés folyamata. A SM feladata az is, hogy a projektben elengedhetetlen, a fejlesztéshez és teszteléshez szükséges eszközöket beszerezze, az ő felelőssége a projekt zavartalan előrehaladása. Mindezek mellett az SM-nek az értekezletek (meeting-ek) megtartásában fontos szerepe van, valamint azok hatékonyságának ellenőrzésében, mely alatt azt értjük, hogy amennyiben egy megbeszélésen nem a termékről van szó, az SM-nek minden esetben vissza kell terelnie a szót az eredeti feladatra. Egy jó Scrum Master céltudatos személyiség, aki kiáll véleménye mellett, és jól képviseli csapata érdekeit. Az SM akkor dolgozik jól, ha a csapat ebből nem vesz észre semmit, és mindenki számára úgy tűnik, hogy nincs is igazán munkája. Egy SM több csapat munkáját is segítheti egyszerre.

Az utolsó ismertetésre kerülő szerepkör a fejlesztő csapat. A tapasztalatok azt mutatják, hogy igazán hatékonyan 4 főtől egészen 8 főig működik egy csapat. Egy szoftverfejlesztési projektben a csapat tagjai között általában tesztelőt, designer-t, adatbázis specialistát és a különböző alkalmazásrétegek megvalósításért felelős szoftverfejlesztők vesznek részt. A Scrum módszertan hangsúlyt helyez arra, hogy minden tudást és információt meg kell osztani a

csapattagok között, így nem lesznek pótolhatatlan személyek, és a projekt akkor sem válik sikertelenné, ha egy vagy több személy távozik a cégtől.

A fejlesztés 2-3 hetes időszakokra, ún. futamokra (Sprint) bontottan történik. Egy projekt indítása mindig a futamtervezéssel (Sprint Planning) kezdődik, amely a három értekezlet típus közül az első. Ezen a megbeszélésen az összes résztvevőnek (PO, SM, T) jelen kell lennie. A PO feladata a résztvevők tájékoztatása a projekt részleteiről, illetve annak bemutatása, hogy a teljes projektet milyen elvégzendő részfeladatokra, ún. történetekre (Story) bontotta. Előfordulhat olyan eset, hogy a PO nem tud mindenkinek kielégítő magyarázatot adni egy történethez, ilyenkor a Story-hoz felírják a fontosabb kérdéseket és a PO feladata megszerezni az ügyféltől a szükséges információt. A PO feladata, hogy ellássa a csapatot az összes fontos információval, míg a csapat mindent olyan tudnivalót át kell adjon a PO-nak, ami a megrendelővel történő tárgyaláshoz szükséges a fejlesztés szempontjából. A futamtervezés során az SM kérdéseket tehet fel, közvetítő szerepet láthat el a csapat és a PO között. Szükség esetén a PO a történeteket felbonthatja, és kisebb feladatokat alakíthat ki. A PO és a csapat közösen becsülik meg a történetek elvégzéséhez szükséges időmennyiséget. Mivel az egy-egy feladatra szánt idő hozzávetőleges megállapítása egy tapasztaltabb fejlesztőnek is fejtörést okozhat, ezért az óra és ember nap alapú becslés mellett történet pontokat (Story Point) is használhatnak. Ebben az esetben komplexitási pontokat határoznak meg mindenki által jól ismert mértékekkel dolgozva (pl. 1. táblázat). Ilyen pl. a póló méretsor (XS, S, M, L, XL, stb.), a kettő hatványai (1,2,4,8, stb.) vagy a Fibonacci számsor (1, 2, 3, 5, 8, stb.) kiválasztott tartománya [11]. A történeteket úgy helyezik el, hogy minden számhoz csak egy Story-t rendelnek. A kisebb számok jelentik a könnyen végrehajtható feladatokat, a nagyobbak pedig a komplexebb Story-kat. A nagyobb értékekkel akár azt is kifejezhetik, hogy az adott feladathoz nem áll rendelkezésre megfelelő kompetencia az adott pillanatban. A sorba rendezési technika azon az alapvető emberi tulajdonságon alapszik, hogy könnyebben meg tudjuk becsülni egy feladat bonyolultságát egy korábbi feladathoz viszonyítva.

1. táblázat. Scrum történet pontozás [12]

Fibonacci	*Póló*	*Starbucks Coffee*
1	XS	espresso shot
2	S	short
3	M	tall
5	L	grande
8	XL	venti
13	XXL	trenta

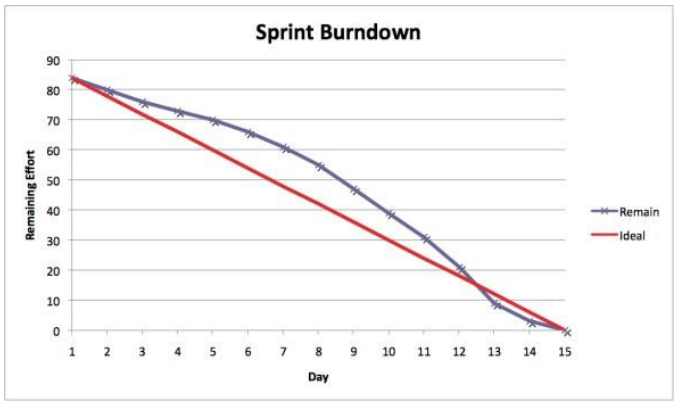
A projekt indulása után napi megbeszéléseken (Daily Standup) mutatják be a fejlesztők, hogy hogyan haladnak a munkával. Személyenként másfél-két perc áll rendelkezésre, amely során röviden elmondják, hogy mit csináltak az előző napon és mit fognak csinálni aznap. Egy ilyen megbeszélésen a fejlesztők egyesével kiállnak a csapat, a PO (neki nem kötelező jelen lennie) és a SM elé, röviden elmondják, hogy melyik történeten dolgoznak, hátráltatja-e a munkájukat valami (pl. információ- vagy eszközhiány). Mivel minden felhasználói történet esetén ismert, hogy ki dolgozott rajta, ezért a felelősök mindig jól beazonosíthatóak, és hiba esetén a feladat felelősét kötelezik a javításra. Ez a meeting akkor működik jól, ha minden nap ugyanabban az időpontban tartják meg, és mindenki részt vesz az eseményen. Egy jól működő, összeszokott csapatnál akár 15 perc alatt 15 főt is meg tudnak hallgatni.

Minden futam végén egy ún. futamáttekintés (Sprint Review) keretében a csapat bemutatja az elkészített produktumot. A megbeszélés során két lépcsőben, demóval és értékeléssel foglalják össze a futam eredményeit. A csapat bemutatja az elvégzett tevékenységet az ügyfél és a PO számára nagyjából 30-60 percben, majd ezt követően az ügyfél és a PO minden egyes történetet értékel abból a szempontból, hogy elfogadja vagy elutasítja azt. Elutasítás esetén alapos indoklás szükséges. Amennyiben egy feladatot az ügyfél elfogadott, a későbbiekben hivatkozhat arra, hogy ő másként gondolta, minden ilyen esetben új történetet kell felvenni a módosítások elvégzéséhez.

Az előzőekben ismertetett három megbeszélés mellett nem kötelező jelleggel lehet még egy úgynevezett visszatekintő értekezletet (Retrospective Meeting) is tartani a projekt befejeztével. A résztvevők megbeszélhetik mivel voltak elégedettek és mivel nem, áttekinthetik hogyan teljesítettek a résztvevők, objektív és szubjektív szempontokból egyaránt. A projekt során felmerült hiányosságokra, problémákra megoldásokat, akciókat definiálnak, amelyek a későbbi projekteknél hasonló helyzetekben segítséget nyújthatnak.

A Scrum alkalmazása során három dokumentum keletkezik. Az első a termék teendőlistája (Product Backlog), amely a termék fejlesztésével kapcsolatos feladatokat (történetek) tartalmazza fontossági sorrend meghatározásával. A projekt indulásakor nem kell a teljes feladatot történetekre bontani, általában két futamra való fejlesztést helyeznek el a termék teendőlistájában. Az elvégzendő feladatok tisztázása után a PO kialakítja a soron következő futam teendőlistáját (Sprint Backlog). A Sprint Backlog a Scrum második dokumentuma. Ebben is fontossági sorrendben jelennek meg a történetek. Ezeket a csapat tagjai egyesével, a fontossági sorrend figyelembevételével választják ki. A futam lezárultát követően a PO újból értékeli a történetek fontosságát, esetenként új Story-kat hoz létre, majd ezekből kialakítja a következő fejlesztési időszak teendőlistáját.

A fejlesztési folyamat harmadik dokumentuma az úgynevezett eredmény kimutatás diagram (Burn Down Chart), amelyből van futamra (1. ábra) és teljes fejlesztésre vonatkozó változat is. A diagramban a vízszintes tengely az időt jelképezi, míg a függőleges tengelyen a még hátra levő feladatokhoz szükséges tevékenységek és erőforrások nagysága jelenik meg. Minden elvégzett feladattal a további fejlesztéshez szükséges munka mennyisége csökken. A cél az, hogy a futam végére elfogyjanak a megvalósítandó feladatok. Az ideális haladást egy kontrolvonal jelzi. A PO feladata olyan nagyságú részfeladatok kialakítása, hogy a fennmaradó fejlesztési munka mindig a kontrol vonal alatt legyen. Túl nagy történeteknél ez a módszer nem használható, ezért fontos a kisebb feladatok definiálása. A dokumentumot a Daily Standup-ok során frissítik. A teljes projektre vonatkozó kimutatás diagram a futamhoz készített változathoz hasonló felépítésű, de ebben az esetben a vízszintes tengely egységei futamok.



1. ábra. Futam eredmények kimutatása [13]

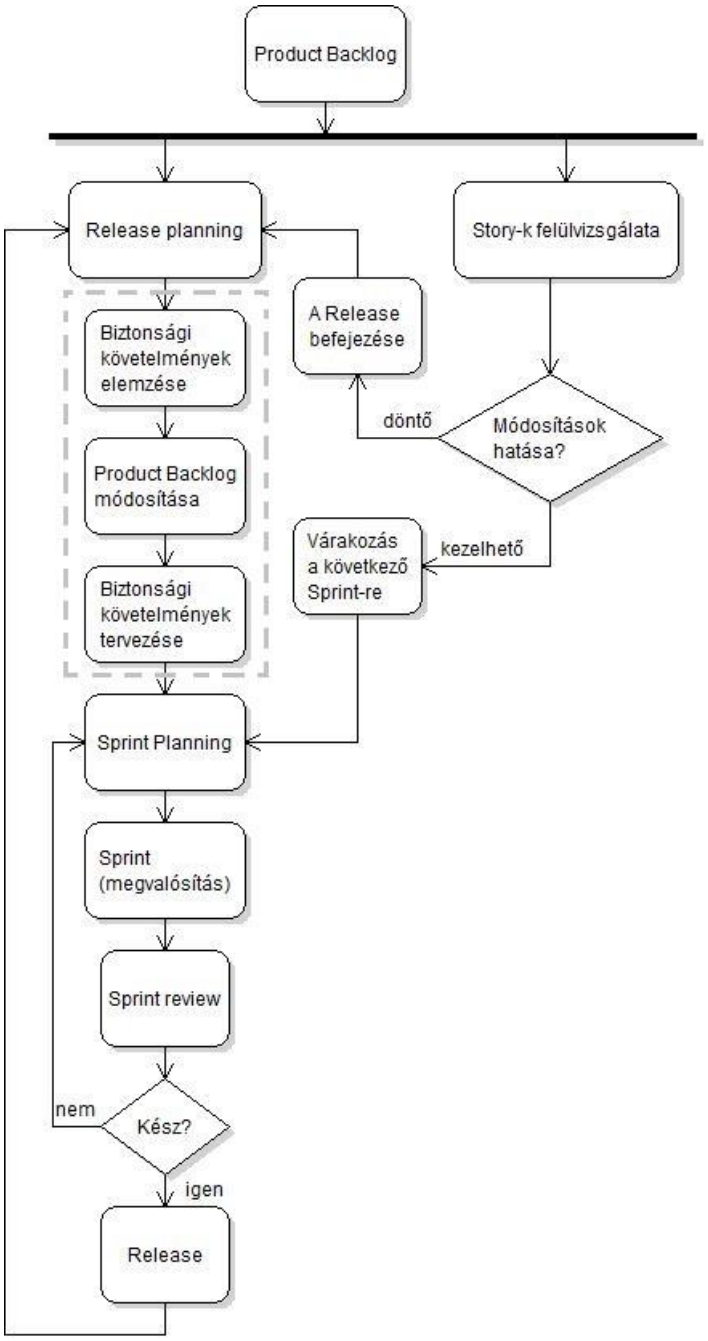
3. Biztonságos Scrum

A Scrum módszert sikeresen alkalmazták számos szoftver létrehozása során jelentősen javítva ezzel a fejlesztési munka hatékonyságát. Megjelenése óta a módszer sokat csiszolódott és fejlődött, valamint néhány hiányosságára is fény derült, ami újabb specifikus változatok kidolgozását eredményezte. Egy ilyen hiányosságként értékelhetjük azt, hogy a módszertan nem ír elő biztonsággal foglalkozó tevékenységeket. Számos területen (pl. web alapú szolgáltatások, hálózati kommunikációt alkalmazó egyéb alkalmazások, stb.) a fejlesztés során kiemelt figyelmet szükséges fordítani a biztonság kérdésére, sőt egyes szoftverek esetében gyakran a biztonságot alapvető minőségi jellemzőnek tekinti a piac. Ezen igény felismerése a biztonsági lépéseket is

integráló Scrum változatok megjelenését eredményezte. Az alábbiakban két ilyen módszert, az S-Scrum-ot és a Secure Scrum-ot tekintjük át.

3.1. S-Scrum

Az S-Scrum-ot (Secure Scrum) [8] eredetileg webes alkalmazások fejlesztéséhez dolgozták ki. A módszer alapgondolata a Scrum bővítése biztonság elemzésével és tervezésével foglalkozó tevékenységekkel. Az S-Scrum folyamatát a 2. ábra mutatja be. Első lépésként itt is a termék teendőlista (Product Backlog) elkészítésével találkozunk, majd ezt követően a folyamat két ágban folytatódik.



2. ábra. Az S-Scrum folyamata

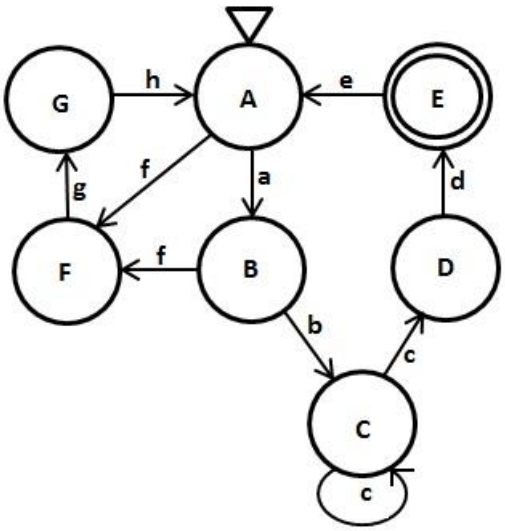
A jobb oldali ág a történetek felülvizsgálatával kezdődik, amely lépés követi a rendszerkövetelmények lehetséges változásait is. Amennyiben a változások hatása oly mértékű,

hogy az aktuális kiadás (Release) nem folytatható, akkor a kiadás befejeződik, és új kiadás tervezését (Release Planning) indítják. Kismértékű változások esetén a módosítások a következő futamtervbe (Sprint Plan) kerülnek.

A bal oldali ág a kiadástervezéssel indul és terméke a kiadásterv (Release Plan - RP), amely egy több futamot magába foglaló magas szintű terv [14]. Ez a terv tükrözi a megvalósítandó funkciókkal kapcsolatos elvárásokat, és segíti a projekt előrehaladásának követését. Kiadás történhet a projekt során, amikor a tervezett részfeladatok elkészülnek, vagy a projekt végén. A kiadásterv elkészítését a biztonsági követelmények elemzése követi. Ez a lépés maga után vonhatja a termék teendőlista módosítását. A biztonsággal kapcsolatos (szaggatott vonallal jelölt rész) utolsó lépése a biztonsági követelmények tervezése, azaz a biztonsági modell kialakítása. Itt épülnek be a kiadásba a biztonsági követelmények elemzésének eredményei.

A továbbiakban a folyamat a Scrum módszertannál megismert módon folytatódik, azaz a futamtervezés (Sprint Planning), a futam és a futamáttekintés következik. A futamáttekintés után, amennyiben elkészült és el lett fogadva az összes történet, akkor kiadásra kerül sor. Egyéb esetekben újabb futamokra lesz szükség. Mivel az S-Scrum középpontjában a biztonság áll, a futamtervezés során a kialakított történetekben figyelembe veszik a biztonsági modellben az adott történetre vonatkozó követelményeket, majd a futamáttekintés során elvégzik a szükséges biztonsági teszteket.

Az S-Scrum módszertan formális modellje a 3. ábrán bemutatott nem determinisztikus véges automatával is leírható. Az ábrán követhetőek a módszer lépései és átmenetei. Az egyes állapotok leírását a 2. táblázat ismerteti, míg az átmeneteket értelmezése a 3. táblázatban szerepel.



3. ábra. S-Scrum automata modellje

2. táblázat. S-Scrum automata modelljének állapotai

Állapot	*Leírás*
A	Biztonsági követelmények elemzése
B	Biztonsági követelmények tervezése
C	Futam
D	Utolsó futam
E	Kiadás
F	Új követelmények
G	Teendőlista módosítása

3. táblázat 2. S-Scrum automata modelljének átmenetei

Átmenet	*Leírás*
a	Biztonsági követelmények elemzése kész
b	Biztonsági követelmények tervezése kész
c	Történetek megvalósítva
d	Kész
e	Új kiadás
f	Új követelmények
g	Új követelmények prioritás értékekkel
h	Teendőlista módosítva

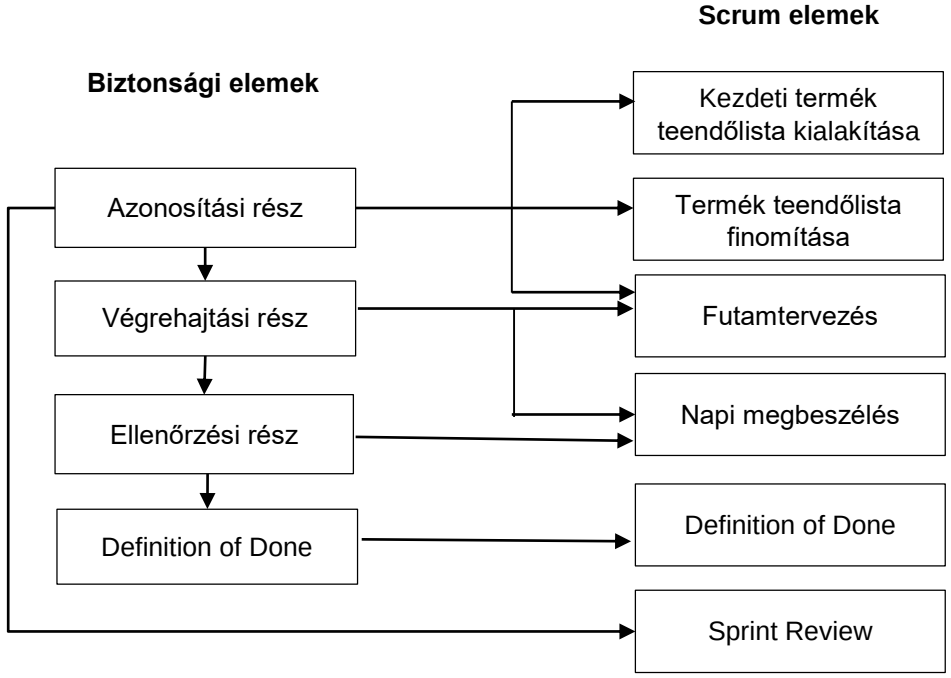
Az S-Scrum nem nevez meg ajánlott technikákat a biztonság elemzésére és tervezésére, a gyakorlatban bármelyik erre a célra kifejlesztett eljárás használható a módszertan alkalmazása során.

3.2. Secure Scrum

A Secure Scrum (SS) [10] a Scrum keretrendszer egy olyan változata, amely kiemelt figyelmet fordít a biztonsággal kapcsolatos problémák kezelésére a teljes fejlesztési folyamat során. A Secure Scrum lehetővé teszi még a nem biztonsági szakemberek számára is, hogy észrevegyék a biztonsági problémákat, bevezessenek biztonsági funkciókat és ellenőrizzék a végrehatást. A Secure Scrum négy komponensből áll:

- Azonosító komponens,
- Implementációs komponens,
- Ellenőrzési komponens,
- Megvalósulási (Definition of Done) komponens.

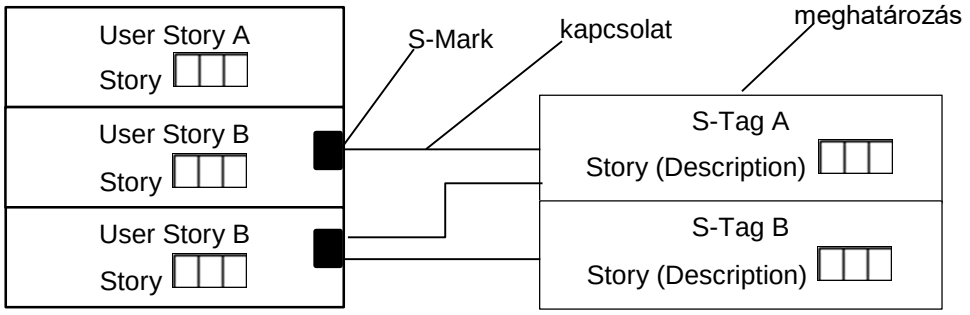
A Secure Scrum által definiált komponensek a standard Scrum folyamat hat szakaszát befolyásolják a 4. ábrán bemutatott módon. Az azonosítási komponens a biztonsági problémák meghatározására, valamint a biztonságot érintő felhasználói történetek azonosítására és jelölésére szolgál.



4. ábra. A Secure Scrum komponensek integrálása a standard Scrum-ba [10]

Ez a komponens a kezdő termék teendőlista kialakítása, a futamtervezés és a Product Backlog finomítása során használatos. Először a megrendelő és a csapattagok rangsorolják a történeteket aszerint, hogy a feldolgozandó adatokhoz való illetéktelen hozzáférés esetén mekkora a veszteségértéke pénzben kifejezve. Ezután értékelik a visszaélési lehetőségeket és rangsorolják azokat kockázatuk szerint. Ebben a fázisban hasznos lehet külső biztonsági szakértő bevonása annak érdekében, hogy a megfelelő kérdések feltevésével és biztonságos felhasználói történetek ajánlásával segítse a csapat munkáját.

Az azonosítást követően a megrendelő és a csapattagok a termék teendőlistában rögzítik a közösen megállapított biztonsági kockázatokat. Ezek dokumentálására a Secure Scrum úgynevezett S-Tag-eket (biztonsági címkéket) használ, amelyek egy-egy biztonsági kockázatot írnak le. Az S-Tag S-Mark-kal (biztonsági jel) jelöli meg azokat a teendőlista elemeket, amelyeknek biztonsági szempontból kockázatot jelentenek. Az S-Tag-eket és S-Mark-okat ún. kapcsolatok kötik össze (ld. 5. ábra). A biztonsági probléma részletes leírása az S-Tag-en segíti a csapatot a biztonsági kockázat megértésében. A biztonsági aggályok külön teendőlista elemben például felhasználói történetben, helytelen használat/visszaélés-történetben (misuse/abuse story) is meghatározhatók. Egy biztonsági kockázat több teendőlista elemhez is kapcsolódhat. Az S-Mark-ok és S-Tag-ek használatának fontos célja, hogy tudatosítsák a fejlesztőkben az egyes részfeladatokhoz (felhasználói történetekhez) kapcsolódó biztonsági kockázatokat.



5. ábra. S-Tag-ek és S-Mark-ok [10]

Az S-Tagek nemcsak a fejlesztés, hanem az ellenőrzés szakaszában is elősegítik a fejlesztői tudatosságot. Pontosan azonosítják a szoftver azon részeit, ahol biztonsági ellenőrzés szükséges, és hasznosak az ellenőrzés erőforrás-igényeinek megbecsüléséhez is.

A Definition of Done (DoD) [15] egy ellenőrzési lista, ami rögzíti azokat a feltételeket, amelyek teljesülése esetén egy felhasználói történetet megoldottnak tekinthetünk. A Secure Scrum az ellenőrzésre és a DoD-ra két megoldást is kínál. Abban az esetben, ha az ellenőrzési folyamat teljesíthető ugyanabban a futamban, ugyanazon fejlesztő által, akkor az a DoD része kell, hogy legyen. Amennyiben azonban a fejlesztőnek nincs meg a szükséges tudása az ellenőrzéshez, vagy külső erőforrást, extra tesztelési időt igényel, akkor az ellenőrzés nem lehet része a DoD-nak. Ekkor az ellenőrzésre egy új feladatot hoznak létre, és S-Mark-kal megjelölik azt. Ezen feladatnak kapcsolódnia kell az eredeti S-Tag-hez is. Így a fejlesztő a saját DoD-ját meg tudja határozni az ellenőrzés nélkül, és a jelölésnek köszönhetően a biztonsági kockázat tesztelésére is sor kerül.

Mivel magas szintű biztonságtechnikai ismeretek nem állnak mindig rendelkezésre, ezért külső tanácsadó bevonása is szükségessé válhat. Ez alapvetően három úton lehetséges. A meghívott szakértő tarthat biztonsági témájú képzést, a tanácsadó megoldhatja az érintett feladatrészt, vagy a tanácsadó felkért támadóként teszteli a rendszert.

4. Következtetések

A Scrum széleskörű elterjedése a hatékony erőforrás-kihasználásnak, egyszerű szerkezetnek és a kommunikációra helyezett erős hangsúlynak köszönhető. Biztonságkritikus feladatokat megoldó és folyamatokat támogató szoftverek fejlesztése során az eredményesség és a megkövetelt minőség biztosítása érdekében az alap Scrum folyamatot olyan elemekkel és lépésekkel szükséges kiegészíteni, amelyek a szoftverfejlesztési folyamat szintjén biztosítják azt, hogy a fejlesztő csapat az elvárható legnagyobb gondossági szinten kezelje és értékelje a biztonsági kockázatokat.

Az elsőként bemutatott S-Scrum módszer egyszerű, és az alap módszertanba könnyen beilleszthető megoldást kínál a biztonsági kérdések kezelésére három lépésben összefoglalva a biztonsági kockázatokkal kapcsolatos teendőket. Hátránya, hogy nem határozza meg részletesen, hogy a kapcsolódó elemzési és követelmény tervezési lépéseket milyen technikákkal és hogyan célszerű végrehajtani. Emellett nem fedi le a teljes fejlesztési folyamatot. Ezzel szemben a másodikként ismertetett Secure Scrum sokkal részletesebb útmutatást ad a biztonsági kérdésekkel kapcsolatos teendők végrehajtására, és négy komponensével lefedi a teljes Scrum folyamatot.

Irodalom

[1] Manifesto for Agile Software Development - 2001. http://www.agilemanifesto.org/. [Hozzáférés: 2015.09.12.].
[2] Highsmith, J.: Messy, exciting, and anxiety-ridden: Adaptive software development, American Programmer, Vol. 10, No. 4 (April), 1997, pp. 23-29.
[3] Beck, K.: Embracing Change with Extreme Programming, IEEE Computer 32(10), 1999, pp. 70-77.
[4] Palmer, S.R., Felsing, J.M.: A Practical Guide to Feature-Driven Development. Prentice Hall. 2002.

[5] Poppendieck, M., Poppendieck, T.: Lean Software Development: An Agile Toolkit, Addison-Wesley Professional, 2003.
[6] Sutherland, J.V.; Schwaber, K.: Business object design and implementation. In OOPSLA '95 workshop proceedings, The University of Michigan, 1995, p. 118.
[7] The History of Scrum http://www.scrumguides.org/history.html. [Hozzáférés: 2015.09.12.]
[8] Mougouei, D., Sani, N.F.M., Almasi, M.M.: S-Scrum: a Secure Methodology for Agile Development of Web Services, World of Computer Science and Information Technology Journal, Vol. 3, No. 1, 2013, pp. 15-19.
[9] Wäyrynen, J., Bodén, M., Boström, G.: Security Engineering and eXtreme Programming: An Impossible Marriage?, in Extreme Programming and Agile Methods - XP/Agile Universe 2004 vol. 3134, C. Zannier, H. Erdogmus, and L. Lindstrom, Eds., ed: Springer Berlin / Heidelberg, 2004, pp. 152-195.
[10] Pohl, C., Hof, H.J.: Secure Scrum: Development of Secure Software with Scrum, arXiv:1507.02992v1 [cs.CR] 10 Jul 2015, http://arxiv.org/pdf/1507.02992.pdf. [Hozzáférés: 2015.09.12.].
[11] Scrum Effort Estimation and Story Points http://scrummethodology.com/scrum-effort-estimation-and-story-points/ [Hozzáférés: 2015.10.26.].
[12] Scrum Story Points http://www.powerhouse360.com/2012/03/story-points/ [Hozzáférés: 2015.10.26.].
[13] Sprint Burndown Reports / Charts http://www.scrum-institute.org/Sprint_Burndown_Reports.php [Hozzáférés: 2015.10.26.].
[14] Scrum Release Planning http://www.scrum-institute.org/Release_Planning.php [Hozzáférés: 2015.09.28.].
[15] Definition Of Done http://guide.agilealliance.org/guide/definition-of-done.html [Hozzáférés: 2015.10.26.].