HADMÉRNÖK
KÖRNYEZETBIZTONSÁG

15. évfolyam (2020) 3. szám 93–110.
DOI: 10.32567/hm.2020.3.5

Mészáros Gergely[1]

Nyílt fejlesztői közösségek hatása az informatikai biztonságra

The Impact of Open Source Development on IT Security

Napjainkban a nyílt forrású fejlesztési modell termékeinek felhasználása széles körben elfogadottá, sokak szerint egyenesen megkerülhetetlenné vált. A közösség, amely ezeket a szoftvereket és komponenseket létrehozza, azonban más normákat és struktúrát követ, mint a hagyományos fejlesztőcsapatok. Fontos kérdés, hogy ezeknek az eltéréseknek milyen biztonsági vonzatai lehetnek rendszerelemet felhasználó információs rendszer biztonságát illetően, illetve hogyan védekezhetünk ezek ellen. A cikkben a terület tudományos eredményének átfogó és szisztematikus elemzése alapján bemutatom a lehetséges kockázati tényezőket és a javasolt védelmi lehetőségeket.

Kulcsszavak: **nyílt forrás, FLOSS, informatikai biztonság, kiberháború**

Products of the Open Source Development Model are widely accepted if not imperative nowadays. The community which creates these components is following different norms and structure than the conventional developer teams. It is important to understand what implications for the information infrastructure might be caused by these differences, and what are the possibilities for protection. In this paper I conduct a systematic analysis of the relevant literature to identify the possible risks and defence mechanisms.

Keywords: **Open Source Community, FLOSS, IT security, Cyberwarfare**

[1] Szent István Egyetem Ybl Miklós Építéstudományi Intézet, mérnöktanár, e-mail: meszaros.gergely@gmail.com; ORCID: https://orcid.org/0000-0002-8390-5627

Problémafelvetés

A nyílt forrású közösségek jellegzetes szervezeti struktúrát mutatnak, és a zárt modelltől jelentősen eltérő projektvezetési, módszertani megoldásokat alkalmaznak. Tekintve, hogy napjainkban egyre több nyílt forrású komponenst használunk fel közvetve vagy közvetlenül, joggal merül fel a kérdés, vajon ezeknek az eltéréseknek van-e valamilyen kimutatható hatása a végterméket felhasználó szervezet biztonságára.

A kérdés egyáltalán nem érdektelen, hiszen a felhasznált komponenseken keresztül akár olyan komoly sérülékenységek is bekerülhetnek a késztermékbe, mint a hírhedt Heartbleed, amely lényegében globális biztonsági válságot okozott. A kiberháborús törekvések egyre komolyabb fenyegetést jelentenek, különös tekintettel a létfontosságú rendszerelemekre, amelyek közvetve vagy közvetlenül szintén felhasználnak nyílt forrású komponenseket.

A nyílt forrás felhasználása érezhetően egyre nő. Ennek részben üzleti, részben technikai okai vannak. Üzleti cél lehet a versenytársak hegemóniájának megtörése, üzleti növekedés segítése vagy alternatív piaci lehetőségek megnyitása. A technikai okok közt említhetjük a piacra kerülési idő lerövidülését, a karbantartás egyszerűsödését és az innovációs képesség növelését.[2] Felhasználóként vagy integrátorként tehát fontos minél pontosabban megérteni milyen kockázatokkal kell szembenéznünk az egyre elkerülhetetlenebbnek látszó nyílt forrású komponensintegráció során.

A fejlesztői közösség szervezeti, módszertani eltéréseiből fakadó biztonsági problémák nem egyértelműek, így pontos megértésük segít a lehetséges kockázatok mérséklésében.

Kutatási módszer

A probléma megértéséhez és elemzéséhez négy nagy digitális könyvtár (IEEE Digital Library, ACM Digital Library, ScienceDirect és SpringerLink) publikációit használtam fel. A forrásirodalomból szisztematikus keresés és irodalomfeldolgozás segítségével gyűjtöttem biztonságot befolyásoló tényezőket.

A szűréshez használt kiválasztási protokoll a következők szerint alakult: csak olyan dokumentumok, cikkek folyóiratok és konferenciapublikációk kerültek az elemzésbe, amelyek a nyílt forrás vagy annak fejlesztési módszertana valamely egyedi jellemzőjét, biztonsági vonzatát elemzik, illetve összehasonlítják azt a zárt forrású fejlesztési módszertan eredményeivel. A publikáció 2005 után jelent meg, kísérlet, esettanulmány, összehasonlító elemzés, vélemény vagy tapasztalati beszámoló. Továbbá, témáját tekintve valamilyen számítástudományi, szoftvermérnöki, illetve szoftverbiztonság-profilú folyóiratban jelent meg.

Nem szerepelnek az elemzésben a következő kihagyási kritériumok szerinti művek: angoltól eltérő idegen nyelven íródott; hálózaton, digitális könyvtáron keresztül

[2] Carl-Erik Mols – Krzystof Wnuk: *Charting the Market Disruptive Nature of Open Source: Experiences from Sony Mobile.* Proceedings of the 39th International Conference on Software Engineering Companion. Buenos Aires, Argentina, IEEE Press, 2017. 175–176.

nem elérhető; szabványok, workshopjelentések, recenzió kategóriákba eső nem teljes értékű dokumentumok, vázlatok, prezentációdiák és -kivonatok; másodlagos és harmadlagos tanulmányok és metaanalízisek; nem szoftvertervezéssel foglalkozó számítástudomány témájú cikkek; nyílt forrásra csak további, tervezett feladatként hivatkozó munkák; amennyiben a publikált eredmény nem köthető a nyílt forráshoz, pusztán a bemutatott vagy felhasznált szoftver(ek) nyílt forrású(ak); illetve ha csak azért foglalkozik nyílt forrással, mert könnyen elérhető, de a vizsgált kritériumnak nincs köze a biztonsághoz, és nem vizsgálja a zárt forrástól való eltérést; esettanulmány, ahol egy adott nyílt forrású szoftver olyan tulajdonságát elemzik, amely nem általánosítható; végül, ha egy szűkebb nyílt forráskategória elemeit hasonlítja össze. A cél elérése érdekében az alábbi keresőkifejezést állítottam össze:

```
(  „open source software" OR „libre software" OR
   „free software" OR „FOSS" OR „F/OSS" OR „F/OSSD" OR
   „FOSSD" OR „FLOSS" OR „F/LOSS" OR „OSSD"
) AND (
   (
      („closed source" OR traditional OR proprietary) AND
      (comparison OR evaluation OR difference)
   ) OR vulnerability OR (security AND (implication* OR problem* OR weakness*
OR issue*))
)
```

A keresőkifejezésben nem szerepelnek a közösségre vonatkozó kulcsszavak, mivel az elemzést szélesebb körű kutatás részeként végeztem, a közösséggel kapcsolatos műveket egyedileg azonosítottam be, ami pontosabb eredményt biztosít, mint a kulcsszavas keresés. A kiválasztott publikációk elemzése során kilenc nyílt forrású jellemzőt és azok biztonsági hatásait kategorizáltam. A kategóriák közül a cikk csak egyetlen kategóriával, a közösség hatásaival foglalkozik. Ennek megfelelően a más kategóriák alá sorolható jellemzők, például a forrás nyíltságából adódó eltérések, a támogatási rendszer eltérései vagy a konkrét terméktulajdonságok nem szerepelnek a jelen elemzésben.

A duplumok eltávolítása után a kihagyási kritériumokat az absztrakt alapján alkalmaztam. Ezt követően 938 publikációt elemeztem gyorsolvasás segítségével. A nyílt forrású közösséggel kapcsolatba hozható témájú írások száma 238-ra csökkent, amelyből 152 mű kifejezetten a közösséggel foglalkozott.

Eredmények

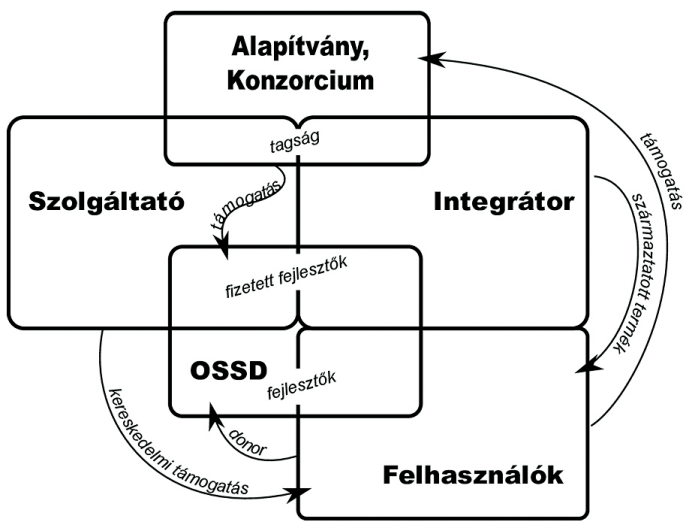
A nyílt modell munkaerejét adó közösség mind szerkezetében, mind motivációjában, vezetési struktúrájában gyökeresen eltér a zárt modell szociális struktúráitól. Az utóbbi években azonban bizonyos konvergencia figyelhető meg, egyre terjednek a nyílt modell módszereit követő kisebb startupok, a nagyobb IT-vállalatok pedig elemeket vesznek át a nyílt módszertanból, sőt aktívan támaszkodnak a nyílt modell közösségeire.

A közösség képességei, szervezettsége jelentős hatást gyakorolhat a végtermékre, így közvetve vagy közvetlenül az azt felhasználó szervezet biztonsági szintjére. Emiatt – bár első pillantásra úgy tűnhet, hogy az információ- és informatikai biztonsághoz nem sok köze van – fontosnak tartom elemezni a klasszikus nyílt közösség felépítését és szociális jellegét. Az eltérések halmaza két csoportra, a szervezeti és a személyi (közösség szereplőinek eltéréseiből adódó) kérdéskörre bontható. A szervezeti jellemzők tovább bonthatók a felépítésből, szociális struktúrából, átláthatóságból, szerveződési módszertanból és az irányítás eltéréseiből adódó eltérésekre. Az alábbiakban az eredményeket ilyen bontásban ismertetem.

Szervezet

A klasszikus nyílt közösség önszerveződő, viszonylag gyorsan változó laza, moduláris hálózatot alkot, amelyben a résztvevők döntési mechanizmusai, szociális kapcsolatai eltérhetnek az üzleti modellben megszokottól.

Korábban nyílt közösség alatt magánszemélyek egy csoportját – elsősorban a fejlesztőket és tesztelőket – értették, mostanra azonban ez az elképzelés megváltozott. A gazdasági szereplők belépésével a nyílt közösség különféle csoportok komplex függőségi viszonyban álló halmazává vált, amelyben jelentős szerepet töltenek be a terméket felhasználó integrátorok, a termékkel kapcsolatos szolgáltatásokat végző cégek és a jogi, társadalmi hátteret adó támogató szervezetek (alapítványok, konzorciumok). A nyílt közösséggel kapcsolatos csoportok szerepét és függőségeit az 1. ábra mutatja be.



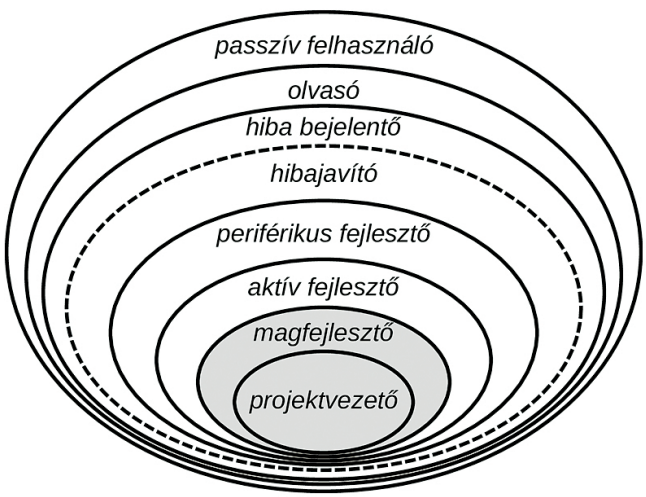
1. ábra
OSSD szervezeti környezete.
Forrás: a szerző szerkesztése

E csoportok együttes hatása alakítja ki a nyílt fejlesztési modell működési hátterét, befolyásolva ezzel a végső termék használhatóságát és biztonsági szintjét. A nyílt közösség által fejlesztett (FLOSS) termékből közvetlenül hasznot húzó szolgáltatók és integrátorok fizetett fejlesztőkön keresztül igyekeznek biztosítani a projekt számukra kedvező irányát, míg a felhasználók – akik gyakran fejlesztők is egyben – élvezhetik a két csoport nyújtotta termékeket és szolgáltatásokat.

A támogatások és irányítás gyakran formális kereteket ölt konzorcium vagy alapítvány formájában, amelyet minden érdekelt fél támogathat, jogi, pénzügyi és társadalmi hátteret biztosítva a közös fejlesztésnek.

Fejlesztőközösség felépítése

Az OSSD-közösségek sokkal inkább hasonlítanak egy szociális hálózatra vagy a megoszló hálózatokra mint hagyományos fejlesztőcsapatra. Bárki részt vehet bennük, bonyolult, határokon átívelő rendszert alkotnak, és a kapcsolattartás döntő részben a hálózaton keresztül zajlik.



2. ábra

OSSD hagymamodell.

Forrás: a szerző szerkesztése (saját változata Di Bella nyomán)[3]

A közösség rugalmas, könnyen változik, szerkezetét tekintve pedig általában egy központi mag részből, és az azt körülvevő további külső rétegekből áll, hagymaszerű struktúrát hozva létre (lásd 2. ábra). A magrész tagjai hozzák a legfontosabb döntéseket, általában csak nekik van joguk módosítani a központi forrástárakat és jobbára hosszú

[3] Enrico Di Bella – Alberto Sillitti – Giancarlo Succi: A multivariate classification of open source developers. *Information Sciences,* 221. (2013), 72–83.

időn keresztül a projektben tartózkodnak.[4] A magot körülvevő fejlesztői holdudvar a magtól távolodva egyre ritkábban adományoz kódot, és egyre kisebb szerepet vállal a fejlesztésből. A teljes fejlesztőtábor lehet egészen nagy, több ezer fős, de a közösség méretének növekedésével már egyre kevesebb a hozzáadott fejlesztőerő.[5]

A magfejlesztők kis csoportja adja a projekt gerincét, az érdemi munka döntő hányadát is gyakran ők végzik[6] az átlagnál magasabb minőségben (a hagyma „magja" tehát kissé eltérő). A mag mérete változhat, de a legtöbb projektben 15 fő alatt marad.[7] A magfejlesztők között lehetnek pótolhatatlan személyek, ami termelékenységi szempontból hasznos ám egyben kockázati tényező is, ugyanis a centrális emberek kiesése komolyan sértheti a csoport teljesítményét.[8] Az akadémiai projektként indult GIMP fejlesztése például több mint egy évre leállt, mert vezető fejlesztői befejezték az iskolát és elmentek dolgozni. Ennyi ideig tartott, míg valaki más felvette a stafétabotot.[9] Kimutatható, hogy a közepesnél nagyobb nyílt projektek döntő része „hősprojekt", ahol a fejlesztés 80%-át a fejlesztők 20%-a végzi.[10]

Az aktív fejlesztők jó rálátással rendelkeznek a projektre, idővel beléphetnek a magfejlesztők közé, bevonják őket a kulcsfontosságú döntésekbe, de szerepük marginális. Az alkalmi fejlesztők csoportja a legnagyobb, ők egyetlen funkcióra vagy hibajavításra koncentrálnak. E csoport külső határán helyezkednek el az egyszeri fejlesztők, akik valamikor hozzájárultak a projekthez, ám végül különféle okok miatt (időhiány, érdektelenség) végül eltávolodtak attól.

A nagyobb projektekben általában cégek is adományoznak kódot, illetve saját fejlesztőkön keresztül igyekeznek a mag közelébe kerülni.

A klasszikus homogén hagymamodell azonban kissé megtévesztő, ugyanis a magot körülvevő közösség általában kis csoportokra, modulokra bomlik, akik egymással sűrűbben kommunikálnak, esetleg saját központi tagjaik vannak. A szponzorált fejlesztők jelenléte tovább növeli a modularitást, az egy céghez tartozók között szorosabb a kapcsolat. Ez a struktúra segíti, hogy a fejlesztőszám növekedésével a kommunikáció ne váljon kezelhetetlenné. A decentralizált projektek jellemzően modulárisabbak mint a centralizáltak, a kisebbek pedig többnyire a hierarchikus típusból indulva érik el

[4] S. Toral – M. Martínez-Torres – F. Barrero: Analysis of virtual communities supporting OSS projects using social network analysis. *Information and Software Technology,* 52. (2010), 3. 296–303.

[5] Ingo Scholtes – Pavlin Mavrodiev – Frank Schweitzer: From Aristotle to Ringelmann: A large-scale analysis of team productivity and coordination in Open Source Software projects. *Empirical Software Engineering,* 21. (2016), 2. 642–683.

[6] Mathias Müller: *Managing the Open Cathedral.* Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Tallinn, Estonia, ACM, 2019. 1176–1179.

[7] Kazuhiro Yamashita et alii: *Revisiting the Applicability of the Pareto Principle to Core Development Teams in Open Source Software Projects.* Proceedings of the 14th International Workshop on Principles of Software Evolution, ACM, 2015. 46–55.

[8] David Garcia – Marcelo Serrano Zanetti – Frank Schweitzer: *The Role of Emotions in Contributors Activity: A Case Study on the GENTOO Community.* International Conference on Cloud and Green Computing, 2013. 410–417.

[9] Akinori Ihara – Akito Monden – Ken-Ichi Matsumoto: *Industry Questions about Open Source Software in Business: Research Directions and Potential Answers.* 6th International Workshop on Empirical Software Engineering in Practice, 2014. 55–59.

[10] Amritanshu Agrawal et alii: *We Don't Need Another Hero?: The Impact of "Heroes" on Software Development.* Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2018. 245–253.

a decentralizált állapotot.[11] A viszonylag kis mag kialakulása nem hiányosság, hanem szükségszerű következmény, ugyanis egy nagyobb létszámú központi csoport tagjaira kezelhetetlenül nagy kommunikációs teher nehezedne.

A hibabejelentők és a fejlesztők egymás között sokat kommunikálnak, ám a legjelentősebb fejlesztők a hibabejelentőkkel keveset kommunikálnak,[12] ami arra enged következtetni, hogy a gyors és hatékony hibajavítások érdekében a FLOSS felhasználó szervezetnek csatlakoznia kell a fejlesztéshez.

Egyes projektek saját biztonsági csapattal is rendelkeznek, akik a biztonsággal kapcsolatos figyelmeztetéseket figyelik és értékelik, illetve folyamatosan hibákat keresnek az alkalmazásban. Ez a jelenség azonban nem általános és formális követelmények nélkül hatékonysága is megkérdőjelezhető. Ruohonen et alii CVE sérülékenység jelentések analízise segítségével arra a megállapításra jutott, hogy a sérülékenységek jelentése általában kis számú magfejlesztőhöz fűződik.[13]

Az irányított és szponzorált nyílt közösségek szignifikánsan több fizetett fejlesztőt alkalmaznak, és jóval magasabb a módosításra jogosult fejlesztők száma. A magot körbevevő holdudvar végzi a tesztelés, hibakeresés nagy részét, a végső döntést és a kiadási ütemezést általában az irányító ipari szereplő határozza meg.[14] Az irányított közösségekben érdemes megkülönböztetni a kettős licencelést alkalmazó, egyetlen cég köré épült projekteket, illetve a közös cél érdekében egyesülő cégcsoport által vezetett projekteket (technológiai konzorcium).

A projekttel kapcsolatos kockázatbecslés elvégzéséhez elengedhetetlen a közösség típusának és struktúrájának ismerete. A magfejlesztők számának és stabil jelenlétének komoly következményei lehetnek a projekt stabilitására. A nyílt szerkezet ellenére csak a maggal szorosabban kommunikáló tagoknak van jó esélye a hibajegyek hatékony kezelésére, ezért a kockázatok mérséklése érdekében érdemes kapcsolattartókat, fejlesztőket delegálni a közösségbe.

Szociális struktúra

A közösség szociális struktúrája és ideológiai beállítottsága direkt hatást gyakorol a jelentkezésekre és a döntéshozatalra, így a teljes projekt teljesítményére.[15] A társakról alkotott vélemény és kapcsolattartás különösen fontos egy közösség esetén. A felek

[11] Andrea Capiluppi – Martin Michlmayr: From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects. In Joseph Feller – Brian Fitzgerald – Walt Scacchi – Alberto Sillitti (szerk.): *Open Source Development, Adoption and Innovation.* Springer, 2007. 31–44.

[12] Darren Forrest et alii: Exploring the Role of Outside Organizations in Free/Open Source Software Projects. In Imed Hammouda – Björn Lundell – Tommi Mikkonen – Walt Scacchi (szerk.): *Open Source Systems: Long-Term Sustainability.* Springer, 2012. 201–215.

[13] Jukka Ruohonen et alii: *Mining Social Networks of Open Source CVE Coordination.* Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement. Gothenburg, Sweden, ACM, 2017. 176–188.

[14] Mario Schaarschmidt – Gianfranco Walsh – Harald F. Von Kortzfleisch: How do firms influence open source software communities? A framework and empirical analysis of different governance modes. *Information and Organization,* 25. (2015), 2. 99–114.

[15] M. Martínez-Torres: A genetic search of patterns of behaviour in OSS communities. *Expert Systems with Applications,* 39. (2012), 18. 13182–13192.

gyakran csak egymás digitális valóját ismerik, gyakori a köszönetnyilvánítás, üdvözlés, a pozitív atmoszféra fenntartása. Az együttműködés alapja elsősorban a bizalom és nem a tekintély. A bizalom elvesztésének fő oka technikai (pl. kritikus hibák bevezetése, tervezési egység megsértése) kisebb részben szociális jellegű. A közösségekben régebb óta részt vevő felek hamarabb közös nevezőre tudnak jutni, vagyis a közösségi kommunikáció tanulható, fejleszthető. A prominens fejlesztők jelentős része gyakran személyesen is ismeri egymást.[16]

Rendszeres személyes kapcsolat híján a szoftverfejlesztés egyes folyamatai, például a kódellenőrzés segít a munkatársakról formált kép kialakításában, ugyanakkor – vagy épp ezért – a kódellenőrzés vagy belső kommunikáció során használt udvarias szociális forma nagyobb jelentőséggel bír.[17]

A szociális normák betartása – amely minimális kockázati tényező üzleti viszony esetén – komoly biztonsági következményekkel járhat nyílt közösség esetén. Az udvariatlanul kért javítást egyszerűen figyelmen kívül hagyhatják, a bizalom elvesztése jelentősen lassíthatja a szervezet számára kritikus folyamatokat. A közösség által elvárt technikai és szociális normák betartása vagy be nem tartása tehát sokkal inkább kockázati tényező, mint zárt forrás esetében, ahol a motiváció alapvetően financiális természetű.

Átláthatóság

A közösség működése és felépítése szinte minden esetben átlátható. Igaz ez az információ nem közvetlenül publikált (nincs szervezeti diagram), de a metainformációk alapján felmérhető és elemezhető. Az OSSD-projektek nyíltan működnek, és minden érdeklődő közeledését szívesen veszik. Szabadon elérhető a szervezettel kapcsolatos (bár bizonyos területeken szegényes) minden dokumentáció, napló, beszélgetés.

Az átláthatóság segíthet felmérni a nemzetbiztonsági kockázatokat annak ellenére, hogy a szervezet általában nemzetközi. Egy határon túli vállalat üzletmenetébe meglehetősen nehézkes belelátni, nehezen átvilágítható a szervezeti struktúrája és azok a nem szabályozási jellegű kényszerek, amelyeknek a szervezetnek meg kell felelnie. Elég az elmúlt évek nagyobb adatgyűjtési botrányaira gondolni a Facebook vagy a Twitter kapcsán. Felmerül a kérdés, hogy vajon mekkora nyomást tud gyakorolni egy vállalatra a működési környezetét biztosító nemzetállam. A klasszikus nyílt közösségekre messzemenően nehezebb nyomást gyakorolni, különösképpen elrejteni annak nyomait.

A közösségre élénk kommunikáció jellemző és a kommunikációs adat szinte minden esetben könnyen hozzáférhető. Általában megállapítható az egyes szereplők munkaköre, szerepe, a munkaerő kihasználásának hatékonysága és ideje. A fejlesztő–fejlesztő és fejlesztő–szoftvertermék közötti interakció analízise révén értékes információ gyűjthető, amely felhasználható a termék minősítése és a kockázatbecslés

[16] Bogdan Vasilescu – Vladimir Filkov – Alexander Serebrenik: *Perceptions of Diversity on Git Hub: A User Survey.* IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, 2015. 50–56.

[17] Kangning Wei et alii: Roles and politeness behavior in community-based free/libre open source software development. *Information & Management,* 54. (2017), 5. 573–582.

során. Ezt a megfigyelést és elemzést érdemes folyamatosan végezni, azaz a közösség állapotát folyamatosan monitorozni.

A fejlesztők beazonosíthatósága komoly előnyt jelenthet komponens integráció során, hiszen közvetlenül azzal az emberrel lehet felvenni a kapcsolatot, aki az adott komponenst készítette, amire IP, üzleti titok és szervezési okokból kifolyólag vajmi kevés esély van az üzleti modell esetén. A vezető fejlesztőket a kommunikációban elfoglalt centralitásuk alapján automatikusan be lehet azonosítani.[18]

Hasonlóképpen elemezhető a projekt fejlesztői dinamikája is, azaz projektfejlesztő-megtartó és fejlesztővonzó képessége.[19] Ennek ismeretében az egyes fejlesztők kilépésének valószínűsége megjósolható, ami segíthet a beszállítói lánccal kapcsolatos kockázatok felmérésben.

Az átláthatóság tehát segít a beszállítói lánccal kapcsolatos kockázatbecslésekben, kritikus alkalmazás esetén, a határokon átívelő biztonsági átvilágítás elvégzésében, továbbá kifejezetten előnyös lehet vészhelyzet esetén, ugyanis a kulcsemberek közvetlen elérése és a többlépcsős ügyfélszolgálat kihagyása jelentősen lerövidítheti a reakcióidőt.

Önszerveződés

A közösség valamilyen célkitűzés, ideológia és/vagy kulcsemberek köré szerveződik. A csoport sikere alapvetően a közösségtől, annak szerveződőképességétől függ, hiszen ha nem sikerül kellő számú hatékony fejlesztőt vonzania, hosszabb távon nem számíthat sikerre.

A tagok általában maguk választanak maguknak feladatot, azaz a feladatok kiosztása is önszerveződő. Nem lehet valakit utasítani vagy elbocsátani, nincs fejlesztő felvétel, sem alkalmassági vizsga, ezeket a folyamatokat a „fejlesztőbevonzás" helyettesíti.[20] A humánerőforrás-menedzsment nem nagyon hasonlít az üzleti világban megszokotthoz. A kevésbé népszerű feladatok elvégzése kapcsán a belépő gazdasági szervezetek szerepe felértékelődik, hiszen megfelelő (általában anyagi) motivációval kiegyensúlyozhatja az egyenlőtlenségeket.

Gyakran megfigyelhető valamilyen mentori szisztéma, amelynek során egy régebbi tapasztalt fejlesztő segíti az újonnan beszállni kívánót. A mentorált csatlakozók mérhetően hatékonyabbak, mint az önállók és a szociális akadályok leküzdésében is sokat segíthet egy jó mentor.[21]

Az önszerveződés ugyanakkor nem jelenti azt, hogy nincs szükség menedzsmentre. A fejlesztői hálózat szervezését elhanyagoló projektek sokkal rosszabbul teljesítenek, gyakran meg is szűnnek. A kisebb, önjelölt vezérrel rendelkező projektek legfeljebb

[18] Xin Yang: *Social Network Analysis in Open Source Software Peer Review.* Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, New York, NY, ACM, 2014. 820–822.

[19] Kazuhiro Yamashita, et alii: *Magnet or Sticky? An Oss Project-by-Project Typology.* Proceedings of the 11th Working Conference on Mining Software Repositories, ACM, 2014. 344–347.

[20] Ioannis Stamelos: Management and Coordination of Free/Open Source Projects. Günther Ruhe – Claes Wohlin (szerk.): *Software Project Management in a Changing World.* Berlin–Heidelberg, Springer, 2014. 321–341.

[21] Fabian Fagerholm, et alii: *The Role of Mentoring and Project Characteristics for Onboarding in Open Source Software Projects.* Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, New York, NY, ACM, 2014. 55:1–55:10.

akkor tudnak fejlődni, ha valamilyen módon fel tudják kelteni a figyelmet; a nagyra növő, rosszul szervezett projektekben pedig a nehézkes kommunikáció jelentősen megnöveli a feladatok lezárásához szükséges időt.

Érdekes megfigyelés, hogy a résztvevők döntő többsége csak kevés (gyakran egyetlen) projektben vesz részt, és csak nagyon kevesen ismerik az egész projektet.[22] Ez arra enged következtetni, hogy a jó fejlesztőkért komoly verseny folyik, a nyílt közösségben szerepet vállalni kívánók száma és a fejlesztésre szánható idő is véges. Következésképpen a nyílt közösség megtartása és vonzása érdekében érdemes bizonyos marketingtevékenységet végezni.

A projekt szerveződésének felmérése segít kockázatbecslés esetén megítélni a projekt várható életképességét. Amennyiben a szervezet új fejlesztőt kíván a fejlesztésbe delegálni, érdemes mentort keresni az átmeneti időszak minimalizálása érdekében.

Döntéshozatal, irányítás, befolyás

A klasszikus modellben általában nincs világos vezetői lánc, a döntések sok vitával járnak, ami megnöveli a koordinációs erőfeszítéseket, a nagyobb projektek (talán épp emiatt) centralizáltabb döntéshozatalt alkalmaznak, ugyanakkor a döntések és indoklások szinte mindig átláthatók maradnak, máskülönben elidegenítik a fejlesztőket és leépítik a közösséget. Nincs alkalmazható kényszer, így bizonyos népszerűtlen feladatok elvégzetlenül maradnak, illetve a nagyobb, szponzorált projektekben az ilyen feladatokat fizetett fejlesztők végzik el.

A közösség irányítása lehet decentralizált (ún. „Bazár" stílusú) vagy centralizált, esetleg hierarchikus felépítésű. A előbbiek általában a független, nem rutinszerű nagy bizonytalanságú feladatokban teljesítenek jól, míg a rutinszerű, erősen összefüggő, kis bizonytalanságú feladatokhoz a centralizált felépítés a megfelelőbb.[23]

A vezetőket – klasszikus esetben – kis részben hagyomány, nagyobb részben alkalmasságuk alapján választják ki, többnyire demokratikus formában, így a klasszikus nyílt közösség meritokráciának vagy technokráciának tekinthető. A bizalom kiépítésénél a legfontosabb tényezők a fejlesztői tudás, a reputáció, valamint a formális és informális tevékenység a közösségen belül.[24] A vezetők szerepe és kiléte meghatározó, egy-egy vezéralak véleménye nagy súllyal eshet latba a döntéseknél. A hatásgyakorlás inkább csak belülről lehetséges, ezért a szoftvercégek gyakran szponzorálnak fejlesztőket, akik képviselik az érdekeiket. Az iparági szereplők befolyása a népszerűbb projekteknél igen jelentős is lehet. A Linux kernel esetében például mára a vállalati hozzájárulás mértéke meghaladja a független hozzájárulások mértékét.[25]

[22] Hironori Hayashi et alii: *Why is collaboration needed in OSS projects? A case study of eclipse project.* Proceedings of the 2013 International Workshop on Social Software Engineering. ACM Press, 2013. 17–20.

[23] Mohammad AlMarzouq – Varun Grover – Jason Bennett Thatcher: Taxing the development structure of open source communities: An information processing view. *Decision Support Systems,* 80. (2015), 27–41.

[24] Yuanfeng Cai – Dan Zhu: Reputation in an Open Source Software Community: Antecedents and Impacts. *Decision Support Systems,* 91. (2016), 103–112.

[25] Iftekar Ahmed – Darren Forrest – Carlos Jensen: *A Case Study of Motivations for Corporate Contribution to FOSS.* 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2017. 223–231.

A közösség fontos döntéseiben a magcsoport tagjainak sokkal nagyobb szerepe van, emiatt fontos lehet e szereplők beazonosítása, hiszen segítségükkel lehet leginkább hatni a csoportra, illetve ezeket a szereplőket érdemes a szervezetnek támogatni. A döntéshozatal általában nem teljesen demokratikus, a magrésztől távolabb esőknek kevesebb beleszólásuk van a döntésekbe, illetve a mag élén állók gyakran fenntartják maguknak a jogot a végső döntésre. A vezérek sem tehetnek meg bármit, hiszen ha szembe mennek a közösséggel, akkor annak elvesztését vagy a projekt másolását (*fork*) kockáztatnák. Legtöbb projekt esetében tehát a végső döntés közösségi nyomásra jön létre, nagyon sokat számít, hogy a változtatni kívánó fél mekkora lobbierőt tud felvonultatni, hány embert tud maga mellé állítani.[26] A változásokat egyáltalán nem olyan egyszerű keresztülvinni, mint az ember elsőre gondolná, hiszen a szoftver óriásira duzzadna, ha bárki beletehetné kedvenc funkcióját (ez az úgynevezett „*feature bloat*"), ami ellen a fejlesztői közösség általában keményen fellép.[27]

Az ipari szereplőknek ugyanakkor gyakran megéri a közösséghez való csatlakozás vagy saját közösség létrehozása, hiszen a közösség által fejlesztőkhöz, informális tesztelőkhöz és rengeteg visszajelzéshez jutnak, valamint növelhetik befolyásukat a projektben. Következésképpen a szervezet felépítése gyakorta bővül fizetett fejlesztőkkel és iparági kapcsolatokkal. A klasszikus nyílt fejlesztői modell nyitottsága természetszerűleg támogatja a belépést, a meritokráciajelleg miatt a cégeknek gyakran nincs is más lehetőségük az irányításra. A belépés – mint láttuk – általában meg is éri, ugyanakkor adaptálódni kell az adott projekt kultúrájához, ugyanis nagyon nehéz megváltoztatni azt.[28] Amennyiben a cég jelenléte nagy, akvizíciója komoly veszélyt jelenthet a projekt jövőjére nézve, és súlyos zavart kelthet a közösségen belül még akkor is, ha a terméket érintően semmilyen változtatás nem történik. Hasonlóan súlyos lehet a helyzet, ha a cég elhagyja a korábban vezetett projektet, még akkor is, ha korábban az nélküle is működőképes volt.[29]

Amennyiben egy gazdasági szereplő hatást szeretne gyakorolni a közösségre, azt a vezetők befolyásolásával vagy erőforrás-bevitellel – elsősorban saját fejlesztőket integrálva – teheti meg. Az erőforrás-ráfordítás formája lehet *integráció,* ahol a szervezet beépül a közösségbe; lehet *hatalomátvétel,* amely esetben a szervezet egy létező közösség felett veszi át az irányítást; *új közösség létrehozása,* amikor a szervezet maga hozza létre a közösséget, és saját üzleti stratégiájához illeszkedő erőforrásként kezeli azt; *másolás* (*fork*), ahol a szervezet saját független változatot indít a FLOSS termékből; végül *kiadás,* ahol a szervezet nyílt forrásúként kiadja valamely terméket, de nem foglalkozik vele, hogy épül-e köré közösség, vagy sem. Ez az utóbbi stratégia

[26] Roshanak Zilouchian Moghaddam – Michael Twidale – Kora Bongen: *Open Source Interface Politics: Identity, Acceptance, Trust, and Lobbying.* Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems, ACM, 2011. 1723–1728.

[27] Paula M. Bach – Robert DeLine – John M. Carroll: ***Designers Wanted: Participation and the User Experience in Open Source Software Development.*** Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2009. 985–994.

[28] Mikko Rajanen – Netta Iivari: Examining Usability Work and Culture in OSS. In Ernesto Damiani – Fulvio Frati – Dirk Riehle – Anthony I. Wasserman (szerk.): *Open Source Systems: Adoption and Impact.* Cham, Springer International Publishing, 2015. 58–67.

[29] Andrea Capiluppi – Klaas-Jan Stol – Cornelia Boldyreff: Exploring the Role of Commercial Stakeholders in Open Source Software Evolution. In Imed Hammouda – Björn Lundell – Tommi Mikkonen – Walt Scacchi (szerk.): *Open Source Systems: Long-Term Sustainability.* Springer, 2012. 178–200.

figyelhető meg többek között a kormányzati szektorban, ahol a kormányzati forrásokból készült terméket OSI-kompatibilis licenccel adják ki.[30]

A fizetett fejlesztők vélhetően nagyobb hatást képesek kifejteni a közösségre, de legalábbis több emberrel kommunikálnak, velük is több ember keresi az interakciót, és a centralitási értékeik is magasabbak,[31] emiatt hasznos lehet a fizetett fejlesztők (automatizált) beazonosítása.[32]

A vállalatok általában dedikált módszertant alkalmaznak, hogy minősített információk ne szivároghassanak ki a nyílt együttműködésen keresztül. A megosztott adatokat üzleti szempontok szerint szűrik és felelősöket, úgynevezett „Gatekeeper"-eket jelölnek ki, akik a szervezet határán állva ellenőrzik az információáramlást.[33]

A nagy, nyílt projektek irányítása sok tekintetben hasonlít a politikára, egzakt módon nehezen megfogható szociális vonatkozásai vannak, több szereplő küzd különféle módszerekkel a befolyásért, és néha ütköznek az érdekek. Integrátorként a nyílt forrású közösség feletti irányítás képessége fontos stratégiai tényező lehet. Amennyiben a projekt elkanyarodik a szervezet által képviselt iránytól, elvben ugyan lehetőség van a projekt forkolására, új közösség alakítására, ám ezek a megoldások rendkívül erőforrásigényesek. Amennyiben a szervezet a projekt irányításában nem érdekelt, akkor is érdemes kockázatértékelésnél figyelembe venni a tényleges irányítók szerepét és céljait, hiszen egy-egy fontosabb szereplő kilépése vagy a projekt forkolása jelentős zavart okozhat a közösségben.

Résztvevők

Mint kiderült, a kulcsszereplők beazonosítása igen fontos, ez azonban nehézségekbe ütközhet. Nemcsak a konkrét személy identitását lehet nehéz megállapítani, hanem az egyes pszeudoanonim identitások (e-mail, nick, álnév, cím stb.) azonosságát is. Néha a személyazonosság pár perces kereséssel kideríthető, más esetben – különösen, ha a fejlesztő szándékosan rejtőzködik – a feladat közel lehetetlen. Jó példa erre Satoshi Nakamoto, a Bitcoin-hálózat tervezője, akinek kilétét az ez irányú jelentős erőfeszítések ellenére is mind a mai napig homály fedi.

Az azonosítás nehézsége miatt problematikus lehet a javítás és hibajegyek szerzői kilétének ellenőrzése. Ezen a problémán segíthet valamelyest a csomaghoz csatolt vagy verziókezelő rendszerben tárolt digitális aláírás. Sajnos sok esetben a fejlesztők több ilyen aláírást is használnak (elvesztés, frissítés miatt), így sem 100%-os, és gyakran csak a pszeudoidentitáshoz rendelés lehetséges. További probléma, hogy a fejlesztők által használt digitális aláírás nem az iparban elterjedtebb Public Key Infrastructure

[30] Schaarschmidt–Walsh–Kortzfleisch i. m. (14. lj.)

[31] Anh Nguyen Duc et alii: Impact of Stakeholder Type and Collaboration on Issue Resolution Time in OSS Projects. In Scott A. Hissam – Barbara Russo – Manoel G. de Mendonça – Neto Fabio Kon (szerk.): *Open Source Systems: Grounding Research*. Springer, 2011. 1–16.

[32] Maëlick Claes et alii: *Towards Automatically Identifying Paid Open Source Developers.* IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), 2018. 437–441.

[33] Anh Nguyen Duc, et alii: Coopetition of Software Firms in Open Source Software Ecosystems. In Arto Ojala – Helena Holmström Olsson – Karl Werder (szerk.): *Software Business.* Cham, Springer, 2017. 146–160.

(PKI) rendszerén alapul, hanem a közösségi GPG kriptográfiai szoftver Web of Trust (WoT-) eljárását használja.

A résztvevők sokféle környezetből érkeznek, több közülük kiváló szakember, hátterük pedig igen eltérő mind életkorra, származásra és képzettségre való tekintettel. Általában jó csapatjátékosok és alapvetően implementációorientáltak.[34]

Az üzleti fejlesztés profitvezérelt (külső motiváción alapul), a nyílt fejlesztést (elsősorban belső) motivációk hatásegyüttese jellemzi. Belső indíttatásból fejleszthet valaki azért, mert tanulni akar, vagy hiányzik egy rég kívánt képesség a kedvenc szoftveréből, a közösségi élmény miatt, egyszerűen az alkotás örömeért vagy a társadalom iránti elkötelezettségből, altruizmusból.[35]

A fizetett fejlesztők motivációja típusát tekintve lehet szabad szponzorálás, amely esetben nincs konkrét instrukció az alkalmazótól; lehet alkalmazotti viszony világos feladatokkal; részlegesen kötött, ahol a fejlesztő ideje egy részével rendelkezhet; valamint egy adott cél eléréséért kapott díj, megbízási viszony.[36]

A belső motiváció előnye, hogy hatékonyabb, mint a külső motiváció, hátránya viszont az irányíthatóság hiánya, így például a belső indíttatásból fejlesztők nem igazán kedvelik az adminisztrációt, az „unalmas" feladatokat, kedvelik viszont a nyíltságot, kényesek a kommunikáció minőségére, a termék nyíltságára, valamint a licencre, könnyen előfordulhat, hogy kifejezetten elutasítóak a zárt forrású rendszerekkel szemben.[37]

A motiváció ismerete fontos, ugyanis ha a szervezet befolyást akar gyakorolni a közösségre saját delegált emberei által, a delegáltaknak tisztában kell lenniük a nyílt fejlesztők motivációival, amennyiben hatni akarnak rájuk. Mint korábban kiderült, a szervezet számára sok esetben fontos lehet a kulcsemberek beazonosítása és aktivizálása. A megfelelő belső motiváció ismerete és külső motivációs forma megtalálása fontos lehet a hatékony hibajavítás, támogatás és a beszállítói lánc stabilitásának szempontjából.

Következtetések

A nyílt és zárt modell szervezeti felépítése és szociális struktúrája jelentősen eltér. A fejlesztőközösség szerkezete közösségihálózat-szerű, irányítása pedig magas technikai felkészültséget, ugyanakkor jó szociális érzékenységet, illetve némi politikusi vénát igényel. A nyílt közösség nehezen befolyásolható, működése viszont teljesen átlátható, így az esetleges kockázat könnyebben becsülhető.

[34] K.Y. Sharif et alii: An empirically-based characterization and quantification of information seeking through mailing lists during Open Source developers' software evolution. *Information and Software Technology,* 57. (2015), 77–94.

[35] Jailton Coelho et alii: *Why We Engage in FLOSS: Answers from Core Developers.* IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2018. 114–121.

[36] Evangelia Berdou: Insiders and Outsiders: Paid Contributors and the Dynamics of Cooperation in Community Led F/OS Projects. In Ernesto Damiani – Brian Fitzgerald – Walt Scacchi – Marco Scotto – Giancarlo Succi (szerk.): *Open Source Systems.* Springer, 2006. 201–208.

[37] Adam Alami – Yvonne Dittrich – Andrzej Wasowski: *Influencers of Quality Assurance in an Open Source Community.* Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2018. 61–68.

A FLOSS felhasználására vonatkozó biztonsági hatások csak közvetetten, a közösség működésén keresztül érzékelhetők. A közösség hatékony működése általában feltétele a hosszú távú működésnek, azaz a közösség segítése és irányítása, de legalább elemzése valamilyen módon szerepet kell kapjon a beszállítói lánccal foglalkozó kockázatelemzésünkben.

Nagyobb volumenű felhasználás esetén elengedhetetlen a projekt befolyásolása, amely csak a közösségben való részvétel által érhető el. A részvétel történhet közvetlenül szervezeti keretek közt, állami szinten vagy alapítványok formájában.

A fejlesztők pontos kiléte nem mindig állapítható meg, ugyanakkor a pszeudo, identitások katalogizálása megoldható, még ha nem is szokványos gyakorlat.

Általános, direkt vagy indirekt felhasználás esetén az alábbi biztonsági hatásokat azonosítottam:

Amennyiben a fejlesztők és a hibabejelentő (felhasználó szervezet) közötti kommunikáció nem felel meg a közösség normáinak, a hiba vagy sérülékenység javítás nélkül maradhat. Ha a közösség vezető szereplője távozik (cég akvizíciója, fejlesztő kilépése), a támogatás folytonossága veszélybe kerülhet. A felhasználó szervezet nem feltétlen alkalmazza a FLOSS-fejlesztésben megszokott, de az üzleti világban ritkán használt kriptográfiai eljárásokat a sértetlenség ellenőrzésére, ami az ellenőrzés elhanyagolásához vezethet. Ha a közösség által képviselt fejlesztési irány jelentősen megváltozik, és a felhasználó szervezet nem tudja befolyásolni azt, a megszokott pénzügyi eszközök (pl. akvizíció) hatástalanok.

Amennyiben a szervezet részt tud vállalni a fejlesztésben, a következő kockázatokkal kell számolni: A delegált fejlesztők nyílt kommunikációja érzékeny adatokat tehet publikussá; az erős cégirányítás miatt a fejlesztők elhagyhatják vagy forkolják a projektet, veszélyeztetve a támogatást; rossz vagy az elvárásoknak nem megfelelő kód küldése a bizalom, ezáltal az közösség feletti irányítás elvesztését okozza.

A kockázatok egy része szerencsére mérsékelhető, az alábbi szempontok betartásával:

1. A vezetőkben tudatosítani kell, hogy a nyílt közösség eltérő módszertant igényel, fejlesztőit más módon kell motiválni.
2. Érdemes folyamatosan monitorozni a közösséget.
3. A hibabejelentések során oda kell figyelni az elvárt szociális és technikai normák betartására.
4. Elemezni kell a belső szerkezetet, meghatározva a kulcsszereplőket. Az adatok alapján kockázatelemzés végezhető, és szükség esetén a megfelelő személy elérési adatai rendelkezésre állnak.
5. A digitális aláírásokat használó fejlesztők nyílt kulcsait (rendszerint GPG) be kell szerezni, biztonságos adatbázisban tárolni, hozzáférhetővé tenni és szükség esetén ellenőrizni.
6. Szükség esetén a közösségbe fejlesztőket kell delegálni, akiknek lehetőség szerint mentort kell keresni.
7. Megfelelő lobbierőt kell kiépíteni az érdekérvényesítő képesség érdekében.
8. A közösségbe delegált fejlesztők kommunikációja ne legyen elkülönített, de a közösségi csatornán folyó kommunikáció információtartalmát ellenőrizni kell. Erre a célra érdemes felelőst (*gatekeeper*) kijelölni, aki a közösséggel való kapcsolattartást és az oda áramló információt ellenőrzi.

Mint látható, a nyílt forrású fejlesztések metodikája és szervezése valóban jelentősen eltérhet az üzleti világban megszokottól (bár mindkét irányból megfigyelhető bizonyos konvergencia). Ezek a különbségek azonban nem egyszerűen belső technikai és adminisztratív eltérések, hanem egyedi felhasználói (integrátori) magatartást is igényelnek. A dobozos vagy szerződéses alapon fejlesztett szoftverek esetében megszoktunk bizonyos garanciális feltételeket és technikai megoldásokat, amelyek a nyílt forrás esetében teljességgel hiányoznak. Léteznek ellenben alternatív megoldások, amelyekkel a korábbiak esetében nem élhetünk.

Az egyik – egyszerű – lehetőség, hogy szervezeti szinten közvetlen nyílt forrású fejlesztésből származó szoftvert egyáltalán nem veszünk igénybe. A nyílt forrású komponensek rendkívüli elterjedtségét figyelembe véve azonban szinte bizonyos, hogy közvetett módon akkor is számtalan FLOSS-komponenssel kerülünk kapcsolatba. Ebben az esetben tehát arról kell meggyőződnünk, hogy a továbbértékesítést vagy fejlesztést végző partner valóban képes a kockázatok mérséklésére, és figyelembe veszi-e a nyílt közösség „játékszabályait".

Összetettebb rendszer huzamosabb ideig történő felhasználása esetén viszont tényleges biztonsági előnyt jelenthet, ha a megszokottnál közelebbről követjük a fejlesztést, ideális esetben saját fejlesztőket delegálva a közösségbe.

Felhasznált irodalom

Agrawal, Amritanshu – Akond Rahman – Rahul Krishna – Alexander Sobran – Tim Menzies: *We Don't Need Another Hero?: The Impact of "Heroes" on Software Development.* Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), 2018. 245–253. DOI: https://doi.org/10.1145/3183519.3183549

Ahmed, Iftekar – Darren Forrest – Carlos Jensen: *A Case Study of Motivations for Corporate Contribution to FOSS.* IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2017. 223–231. DOI: https://doi.org/10.1109/vlhcc.2017.8103471

Alami, Adam – Yvonne Dittrich – Andrzej Wasowski: *Influencers of Quality Assurance in an Open Source Community.* Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2018. 61–68. DOI: https://doi.org/10.1145/3195836.3195853

AlMarzouq, Mohammad – Varun Grover – Jason Bennett Thatcher: Taxing the development structure of open source communities: An information processing view. *Decision Support Systems,* 80. (2015), 27–41. DOI: https://doi.org/10.1016/j.dss.2015.09.004

Bach, Paula M. – Robert DeLine – John M. Carroll: *Designers Wanted: Participation and the User Experience in Open Source Software Development.* Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, 2009. 985–994. DOI: https://doi.org/10.1145/1518701.1518852

Berdou, Evangelia: Insiders and Outsiders: Paid Contributors and the Dynamics of Cooperation in Community Led F/OS Projects. In Ernesto Damiani – Brian Fitzgerald – Walt Scacchi – Marco Scotto – Giancarlo Succi (szerk.): *Open Source Systems.* Springer, 2006. 201–208. DOI: https://doi.org/10.1007/0-387-34226-5_20

Cai, Yuanfeng – Dan Zhu: Reputation in an Open Source Software Community: Antecedents and Impacts. *Decision Support Systems,* 91. (2016), 103–112. DOI: https://doi.org/10.1016/j.dss.2016.08.004

Capiluppi, Andrea – Klaas-Jan Stol – Cornelia Boldyreff: Exploring the Role of Commercial Stakeholders in Open Source Software Evolution. In Imed Hammouda – Björn Lundell – Tommi Mikkonen – Walt Scacchi (szerk.): *Open Source Systems: Long-Term Sustainability.* Springer, 2012. 178–200. DOI: https://doi.org/10.1007/978-3-642-33442-9_12

Capiluppi, Andrea – Martin Michlmayr: From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects. In Joseph Feller – Brian Fitzgerald – Walt Scacchi – Alberto Sillitti (szerk.): *Open Source Development, Adoption and Innovation.* Springer, 2007. 31–44. DOI: https://doi.org/10.1007/978-0-387-72486-7_3

Claes, Maëlick – Mika Mäntilä – Miika Kuutila – Umar Farooq: *Towards Automatically Identifying Paid Open Source Developers.* IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), 2018. 437–441. DOI: https://doi.org/10.1145/3196398.3196447

Coelho, Jailton – Marco Tulio Valente – Luciana L. Silva – André Hora: *Why We Engage in FLOSS: Answers from Core Developers.* IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2018. 114–121. DOI: https://doi.org/10.1145/3195836.3195848

Di Bella, Enrico – Alberto Sillitti – Giancarlo Succi: A multivariate classification of open source developers. *Information Sciences,* 221. (2013), 72–83. DOI: https://doi.org/10.1016/j.ins.2012.09.031

Duc, Anh Nguyen – Daniela S. Cruzes – Geir K. Hanssen – Terje Snarby – Pekka Abrahamsson: Coopetition of Software Firms in Open Source Software Ecosystems. In Arto Ojala – Helena Holmström Olsson – Karl Werder (szerk.): *Software Business.* Cham, Springer, 2017. 146–160. DOI: https://doi.org/10.1007/978-3-319-69191-6_10

Duc, Anh Nguyen – Daniela S. Cruzes – Claudia Ayala – Reidar Conradi: Impact of Stakeholder Type and Collaboration on Issue Resolution Time in OSS Projects. In Scott A. Hissam – Barbara Russo – Manoel G. de Mendonça – Neto Fabio Kon (szerk.): *Open Source Systems: Grounding Research*. Springer, 2011. 1–16. DOI: https://doi.org/10.1007/978-3-642-24418-6_1

Fagerholm, Fabian – Alejandro S. Guinea – Jürgen Münch – Jay Borenstein: *The Role of Mentoring and Project Characteristics for Onboarding in Open Source Software Projects.* Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, New York, NY, ACM, 2014. 55:1–55:10. DOI: https://doi.org/10.1145/2652524.2652540

Forrest, Darren – Carlos Jensen – Nitin Mohan – Jennifer Davidson: Exploring the Role of Outside Organizations in Free/Open Source Software Projects. In Imed Hammouda – Björn Lundell – Tommi Mikkonen – Walt Scacchi (szerk.): *Open Source Systems: Long-Term Sustainability.* Springer, 2012. 01–215. DOI: https://doi.org/10.1007/978-3-642-33442-9_13

Garcia, David – Marcelo Serrano Zanetti – Frank Schweitzer: *The Role of Emotions in Contributors Activity: A Case Study on the GENTOO Community.* International Conference on Cloud and Green Computing. 2013. 410–417. DOI: https://doi.org/10.1109/cgc.2013.71

Hayashi, Hironori – Akinori Ihara – Akito Monden – Ken-ichi Matsumoto: *Why is collaboration needed in OSS projects? A case study of eclipse project.* Proceedings of the 2013 International Workshop on Social Software Engineering. ACM Press, 2013. 17–20. DOI: https://doi.org/10.1145/2501535.2501539

Ihara, Akinori – Akito Monden – Ken-Ichi Matsumoto: *Industry Questions about Open Source Software in Business: Research Directions and Potential Answers.* 6th International Workshop on Empirical Software Engineering in Practice, 2014. 55–59. DOI: https://doi.org/10.1109/iwesep.2014.12

Martínez-Torres, M.: A genetic search of patterns of behaviour in OSS communities. *Expert Systems with Applications,* 39. (2012), 18. 13182–13192. DOI: https://doi.org/10.1016/j.eswa.2012.05.083

Mols, Carl Erik – Krzysztof Wnuk: *Charting the Market Disruptive Nature of Open Source: Experiences from Sony Mobile.* Proceedings of the 39th International Conference on Software Engineering Companion, Buenos Aires, Argentina, IEEE Press, 2017. 175–176. DOI: https://doi.org/10.1109/icse-c.2017.110

Müller, Mathias: *Managing the Open Cathedral.* Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Tallinn, Estonia, ACM, 2019. 1176–1179. DOI: https://doi.org/10.1145/3338906.3341461

Rajanen, Mikko – Netta Iivari: Examining Usability Work and Culture in OSS. In Ernesto Damiani – Fulvio Frati – Dirk Riehle – Anthony I. Wasserman (szerk.): *Open Source Systems: Adoption and Impact.* Cham, Springer International Publishing, 2015. 58–67. DOI: https://doi.org/10.1007/978-3-319-17837-0_6

Ruohonen, Jukka – Sami Hyrynsalmi – Sampsa Rauti – Ville Leppänen: *Mining Social Networks of Open Source CVE Coordination.* Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, Gothenburg, Sweden, ACM, 2017. 176–188. DOI: https://doi.org/10.1145/3143434.3143458

Schaarschmidt, Mario – Gianfranco Walsh – Harald F. Von Kortzfleisch: How do firms influence open source software communities? A framework and empirical analysis of different governance modes. *Information and Organization,* 25. (2015), 2. 99–114. DOI: https://doi.org/10.1016/j.infoandorg.2015.03.001

Scholtes, Ingo – Pavlin Mavrodiev – Frank Schweitzer: From Aristotle to Ringelmann: A large-scale analysis of team productivity and coordination in Open Source Software projects. *Empirical Software Engineering,* 21. (2016), 2. 642–683. DOI: https://doi.org/10.1007/s10664-015-9406-4

Sharif, Khaironi Y. – Michael English – Nour Ali – Chris Exton – J.J. Collins – Jim Buckley: An empirically-based characterization and quantification of information seeking through mailing lists during Open Source developers' software evolution. *Information and Software Technology,* 57. (2015), 77–94. DOI: https://doi.org/10.1016/j.infsof.2014.09.003

Stamelos, Ioannis: Management and Coordination of Free/Open Source Projects. Günther Ruhe – Claes Wohlin (szerk.): *Software Project Management in a Changing World.* Berlin–Heidelberg, Springer, 2014. 321–341. DOI: https://doi.org/10.1007/978-3-642-55035-5_13

Toral, S. – M. Martínez-Torres, F. Barrero: Analysis of virtual communities supporting OSS projects using social network analysis. *Information and Software Technology,* 52. (2010), 3. 296–303. DOI: https://doi.org/10.1016/j.infsof.2009.10.007

Vasilescu, Bogdan – Vladimir Filkov – Alexander Serebrenik: *Perceptions of Diversity on Git Hub: A User Survey.* IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering. 50–56. DOI: https://doi.org/10.1109/chase.2015.14

Wei, Kangning – Kevin Crowston – U.Yeliz Eseryel – Robert Heckman: Roles and politeness behavior in community-based free/libre open source software development. *Information & Management,* 54. (2017), 5. 573–582. DOI: https://doi.org/10.1016/j.im.2016.11.006

Yamashita, Kazuhiro – Shane McIntosh – Yasutaka Kamei – Ahmed E. Hassan – Naoyasu Ubayashi: *Revisiting the Applicability of the Pareto Principle to Core Development Teams in Open Source Software Projects.* Proceedings of the 14th International Workshop on Principles of Software Evolution, ACM, 2015. 46–55. DOI: https://doi.org/10.1145/2804360.2804366

Yamashita, Kazuhiro – Shane McIntosh – Yasutaka Kamei – Naoyasu Ubayashi: *Magnet or Sticky? An Oss Project-by-Project Typology.* Proceedings of the 11th Working Conference on Mining Software Repositories, ACM, 2014. 344–347. DOI: https://doi.org/10.1145/2597073.2597116

Yang, Xin: *Social Network Analysis in Open Source Software Peer Review.* Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, New York, NY, ACM, 2014. 820–822. DOI: https://doi.org/10.1145/2635868.2661682

Zilouchian Moghaddam, Roshanak – Michael Twidale – Kora Bongen: *Open Source Interface Politics: Identity, Acceptance, Trust, and Lobbying.* Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems, ACM, 2011. 1723–1728. DOI: https://doi.org/10.1145/1979742.1979835