HADMÉRNÖK

IX. Évfolyam 3. szám - 2014. szeptember

Mészáros Gergely
meszaros.gergely@gmail.com

ELOSZTOTT VERZIÓKEZELÉS A KÖZIGAZGATÁSBAN

Absztrakt

Napjainkban a szoftverfejlesztés nehezen elképzelhető verziókövető rendszer segítsége nélkül. Szinte minden nagyobb szervezet alkalmaz valamilyen a verziókövető megoldást, a fejlesztés vagy általában a dokumentum-kezelés területén. Ugyanakkor megfigyelhető, hogy egyre nagyobb népszerűségnek örvendenek az eredetileg nyílt forrású fejlesztéseket támogatására kifejlesztett elosztott verziókövető rendszerek. Cikkünkben arra keressük a választ, milyen vonzó lehetőségeket nyújthat ez a feltörekvő megoldás a közigazgatási információs rendszerek vagy a kapcsolódó létfontosságú rendszerelemek információs rendszereinek területén. Bemutatjuk az elosztott verziókövetők potenciálisan hasznosítható képességeit és feltárjuk a közigazgatási rendszerek azon tevékenységeit, ahol az ilyen eszközök használata előrelépést jelenthet.

Nowadays it is hard to imagine software development without help of version control systems. Almost any mayor organizations are using some kind of version control in the field of development or document management. In the same time one can observe the increasing popularity of distributed version control systems that were originally created for supporting open source development. In this paper we are seeking for opportunities that can be exploited in the area of governmental information systems and related critical information infrastructures. We are specifying the potentially usable features of distributed version control systems and exploring government activities where application of these tools may introduce improvements.

Kulcsszavak: *DVCS, közigazgatás, verziókövetés ~ DVCS, government, version control*

BEVEZETÉS

A nyílt forrású programfejlesztés támogatására kifejlesztett verziókövető rendszerek valóságos virágzásnak indultak az utóbbi évtizedben. Tucatnyi különféle változat versenyez az első helyért [1], és ma már szinte elképzelhetetlen olyan komolyabb fejlesztés, amelynek forráskód változásait ne lehetne legalább valamelyik elterjedtebb verziókövető megoldás segítségével megfigyelni.

Ezeket a rendszereket ugyan elsősorban a programkód változásainak követésére tervezték, valójában bármilyen szöveges, illetve bizonyos korlátok között bináris adatok verziókövetésére is alkalmasak. Tekintettel arra, hogy a demokratikus társadalomban az állami szektor dokumentációinak és informatikai megoldásainak könnyen kereshető archiválása, adott esetben publikálása kiemelt fontossággal bír, joggal merül fel a kérdés hogy hasznosíthatóak-e az elterjedt verziókövető rendszerek képességei ezen a területen.

Mint látni fogjuk, ma már nem példa nélkül való, hogy az állami szféra nyílt forrású verziókövető rendszert használjon fel saját céljaira, sőt, a nyíltság fontosságát hangsúlyozó, és a közösség erőteljesebb bevonását szorgalmazó demokratikus irányvonal kifejezetten előnyben részesíti ezeket a megoldásokat.

A modern elosztott verziókövető rendszerek felhasználhatósága kapcsán két fontos kérdésre kell választ kapnunk. Egyrészt melyek azok a közigazgatási felhasználási területek, amelyek működése hatékonyabbá tehető a verziókövetés integrálásával, másrészt melyek azok a kritériumok amelyek teljesülése ezeken a területeken elvárható illetve alapkövetelmény. E két alapvető kérdés megválaszolása után a szerencsésen széles választékból már ki lehet keresni a céljainknak leginkább megfelelő variánst, illetve meghatározhatóak az esetlegesen fejlesztésre szoruló területek.

Ez a cikk a közigazgatással kapcsolatos munkafolyamatokban felhasználható elosztott verziókezelő rendszerek vizsgálati kritériumainak meghatározására koncentrál. Az első fejezetben röviden összefoglalom a verziókövetés céljait és bemutatom a vizsgálat tárgyát képező elosztott verziókezelő rendszereket (továbbiakban DVCS[1]). A második fejezetben néhány pozitív példán keresztül szemléltetem a verziókövetők jelenlegi alkalmazási lehetőségeit a közigazgatásban, és meghatározom azokat a közigazgatási felhasználási területeket és feladatokat, ahol a DVCS alkalmazása feltehetően előnyt jelenthet. A harmadik fejezetben meghatározom a DVCS értékelésének lehetséges szempontjait valamint javaslatot adok a második fejezetben meghatározott felhasználási területek esetében célszerűen alkalmazható prioritási sorrend felállítására.

1. VERZIÓKÖVETÉS HÁTTERE

A modern verziókövető rendszerek elsődleges felhasználási területe a szoftverfejlesztés, különösen a közösségi, elosztott szoftverfejlesztés támogatása. A ténylegesen kezelhető dokumentumok köre azonban ennél jóval bővebb; bizonyos korlátok között bármilyen dokumentumtípus kezelhető ezekkel a rendszerekkel.

A közigazgatás folyamataiban felhasznált dokumentumok esetében alapvető fontossággal bír a változások pontos, esetenként nyílt nyomon követhetősége. Az elterjedt verziókövető rendszerek használata tehát előnyt jelenthet. Az egyre inkább előtérbe kerülő, és döntő részben valamilyen verziókövető megoldást alkalmazó nyílt forrású szoftverekkel való kapcsolattartás igénye szintén azt sugallja, hogy kormányzati, közigazgatási szinten érdemes ezzel a területtel behatóbban foglalkozni.

1 distributed vagy decentralized version control system, röviden DVCS

1.1. Verziókövetésről röviden

A mérnöki tudományokban és a szoftverfejlesztés területén különösen elterjed gyakorlat a tervek, dokumentumok vagy forráskód változásainak rögzítése, azaz a verziókövetés. Az egyes változatokat valamilyen jelöléssel, számmal vagy betűjellel, más néven verziókóddal ellátva és tárolva a dokumentum korábbi állapota visszakereshetővé és összehasonlíthatóvá válik. Amennyiben a változtatást végző személy azonosítóját valamint a változtatás időpontját is rögzítjük, a dokumentum módosításainak teljes naplója rendelkezésünkre áll. A technológia fejlődésével a dokumentumok száma jelentősen megnövekedett, és egyre nagyobb igény mutatkozott a különféle verziókon párhuzamosan végzett munka esetleges konfliktusainak kezelésére is.

A verziókövető rendszerek ezt a két a feladatot végzik el automatikus vagy félautomatikus módon. Használatukkal jelentős mértékben leegyszerűsödik az összehasonlítás és keresés, bizonyos dokumentumtípusok esetén lehetőség nyílik az eltérő verziók különbségeinek kezelésére is, emellett követhető naplót kapunk a dokumentumok változásairól.

A verziókövető rendszer (az irodalomban általában VCS[2]) lehet önálló alkalmazás, illetve valamilyen dokumentumkezelő szoftverbe ágyazott képesség is. Ez utóbbira példa a Google Docs, a Wikipedia Page history vagy a LibreOffice és Microsoft Word beépített verziókezelő képessége [2]. Az alkalmazásba ágyazott verziókövetés értelemszerűen az adott dokumentumtípus és formátum kezelésére korlátozódik. Ezekkel a rendszerekkel itt nem foglalkozunk, a vizsgálatunk tárgyát az első csoportba tartozó általános célú verziókezelő rendszerek képezik.

Felhasználási területét tekintve napjainkban a VCS rendszereket elsősorban a forráskód nyilvántartására használják, de találhatunk alkalmazási példát az eredeti cél, a konfiguráció menedzsment valamint az általános értelemben vett dokumentum-kezelés területén is. A modern, olvasható XML struktúrájú dokumentum állományok változásai jól kezelhetők az általános célú verziókövetőkkel.

A verziókezelés számos változata között kialakult egy közös képesség készlet, egységes nyelvezet, amely kifejezések alatt minden rendszerben hasonló elveket értünk. Mivel az irodalom gyakran használja ezeket a kifejezéseket, röviden összefoglalom a legfontosabb fogalmakat:

- *Commit:* Érvényesítés, átvezetés. Változások mentése a rendszerbe.
- Atomi műveletek: A modern verziókövetők tulajdonsága. A rendszer a műveletek megszakítása esetén sem kerülhet inkonzisztens állapotba. A legfontosabb ilyen művelet az érvényesítés (commit).
- *Merge:* Összefésülés. Két eltérő verzió különbségeit a rendszer automatikusan vagy emberi segítséggel képes összeolvasztani.
- *Tag, label:* Címke. A legtöbb verziókövető képes egyes állapotokat egyedi címkével megjelölni, amellyel utólag az adott állapotra hivatkozni lehet.
- *Branch:* Elágazás. A rendszer képes egy adott állapotnál elágazást képezni és több eltérő változásfolyamot párhuzamosan vezetni. A két változat függetlenül fejlődhet, az eltérések később általában összefésülhetők.

A verziókezelők részletes képességeiről további információk találhatóak a Wikipédia oldalain [1].

2 Version Control System

1.2. Elosztott verziókövető rendszerek

A párhuzamos munkavégzés során kialakuló ellentmondások feloldására két módszer terjedt el: a kérdéses adathalmaz zárolása illetve a változatok összefésülése. A CAP[3] tétel alapján egy mai modern hálózatos rendszerben, ahol a particionálódás teljes kivédése gyakorlatilag lehetetlen, választani kell a rendelkezésre állás vagy a konzisztencia előtérbe helyezése között. [3] A zárolás lényegében a konzisztencia előtérbe helyezése a rendelkezésre állás rovására, az utólagos összefésülés pedig az azonnali rendelkezésre állás biztosítása a feltétel nélküli konzisztencia feláldozása árán. A hatékony párhuzamos munkavégzést támogató verziókezelő rendszerek általában ez utóbbi utat követik, igaz, néhányuk esetében lehetőség van az állományok zárolására is. A rendelkezésre állás biztosítása tekintetében az elosztott rendszerek általában sokkal jobban teljesítenek, mint a központosított rendszerek [4], az egyre összetettebb fejlesztési folyamatok pedig robusztus megoldást igényeltek, így az elosztott verziókezelők megjelenése és térhódítása csupán idő kérdése volt.

1997-ben jelent meg a Bitkeeper, az elosztott irányvonal egyik úttörője [5]. A Linux kernel fejlesztése során is használt rendszer azonban nem volt nyílt, így a kernelfejlesztők saját céljaikra megalkották az azóta is töretlenül növekvő népszerűségű Gitet, amelyet hamarosan számos újabb elosztott verziókezelő megjelenése követett.[6] Napjaink három legismertebb nyílt forrású elosztott verziókezelő rendszere a Git, a Mercurial és a Bazaar. Kevésbé elterjedt, de figyelemre méltó projekt a Darcs és a Fossil. A jövőben minden bizonnyal további versenytársak is megjelennek majd. Az Ohloh nyilvántartásában szereplő nyílt forrású projektek elemzésével készült statisztika szerint 2014 elején az elosztott verziókezelők részaránya körülbelül 40%, amelynek döntő hányadát a Git felhasználás adta [7]. Az elosztott verziókezelők alapvető céljai nem különböznek a központosított megoldásétól, viszont az elosztott kialakítás néhány igen előnyös tulajdonsággal jár. Az elosztott verziókezelők gyakran említett előnyei az alábbiak [8, 9]:

- minden csomópont tartalmazza az alapadathalmazt (egyenrangúság);
- a felhasználók központi adminisztráció nélkül is együttműködhetnek;
- nincs egyetlen gyenge pont4;
- lehetőség van gyors, helyi (offline) commit végrehajtására;
- az összefésülés művelete egyszerű;
- a munkafolyamat rugalmasan alakítható ki;
- tesztelés nélkül egy helyen bejegyzett változtatás nincs hatással fejlesztés egészére.

Ez utóbbi szempont elsősorban programfejlesztés során érdekes és létező problémát enyhít, ugyanis a verziókövető rendszerekbe a forráskód egy része valóban tesztelés nélkül kerül be: Negara et. al. megfigyelései szerint a változtatások 24%-a ilyen volt [10]. A központi adminisztráció nélküli együttműködés természetesen nem előfeltétel, pusztán csak lehetőség. Megfelelő munkafolyamatot választva szükség szerint elosztott, hierarchikus vagy akár központosított fejlesztési rend is kialakítható [11]. Az egyenrangú csomópontok közül ki lehet jelölni egyet, amelyen a hivatalos változatot tartjuk nyilván, így megfelelő szervezéssel szimulálni tudjuk a központosított verziókövetést. Ezek alapján az elosztott verziókövetést tekinthetjük bővebb képességkészletnek, azaz központosított verziókövető rendszerben funkcionális előnyt nem fogunk találni. Ennek ellenére van néhány olyan tulajdonság, amely bizonyos körülmények közt a DVCS alkalmazása ellen szólhat [5]:

- Nagyobb helyi méret. Az elosztott rendszer minden csomóponton minden adatot tárol, így több erőforrást igényel a csomópontokon.

3 Consistency, Availability, Partition tolerability
4 single point of failure

– Meredekebb tanulási görbe. A több funkció hatékony kihasználása több ismeretet igényel a felhasználótól.
– Nehezebb karbantartás. A fejlesztési struktúra kialakítása és tudatosítása időigényesebb, az elosztott tárolóhelyek karbantartása pedig nehezebb.
– A helyi commitok miatt a csomópontok valójában nem teljesen egyenrangúak. A helyi csomópont meghibásodása esetén a lokális változási információk elveszhetnek.

Mint látható elosztott verziókezelők működése néhány elv tekintetében hasonlít a felhő alapú rendszerek működéséhez. Ugyanakkor fontos látni, hogy a hagyományosan felhő alapú megoldáson alapuló verziókezelő rendszerek éppúgy lehetnek elosztottak mint központosítottak, ahogy napjainkban valóban láthatunk mindkét esetre példákat. Vizsgálatunk szempontjából azonban ez a fajta felhőalapúság pusztán implementációs részletnek tekinthető.

1.3. Verziókezelés mint szolgáltatás

A nyílt forrású programok fejlesztése esetében nem mindig áll rendelkezésre központi tárhelyet biztosító vállalati infrastruktúra. Ezen a problémán azonban könnyen segíthetnek az egyre népszerűbb közösségi szoftvertárhelyek. Ilyen szolgáltatások közel két évtizede léteznek, és az utóbbi években jelentősen bővült a választék és a szolgáltatások minősége. A tucatnyi lehetőségből a kiszolgált projektek száma alapján a legjelentősebb szereplők: GitHub, SourceForge, GoogleCode, Bitbucket, Assembla, CodePlex, Launchpad, Gitorious[5]. Használatuk olyan széles körű, hogy szinte minden nyílt szoftver forrása megtalálható legalább az egyik ismertebb szolgáltatón. A közös kommunikációs felülettel segített együttműködésre és fejlesztésre új kifejezés is született: a "közösségi programozás[6]".

A verziókezelés, mint webszolgáltatás mindamellett jóval többet ad egyszerű közös tárhelynél. A legtöbb szolgáltatás komplett infrastruktúrát nyújt: közös kommunikációs csatornaként működik, egyszerűen kezelhető felhasználói felületet biztosít a projektkezeléshez, kódvizsgálati és hibakövető rendszert szolgáltat, valamint átjárást biztosít az több különféle verziókövető megoldás között. [13]

A GitHub egyik általánosan használt DVCS stratégiája a beolvasztási kérelmen (pull request) alapuló együttműködés. Ennél az együttműködési formánál a központi kódtárat nem osztják meg minden fejlesztő között. Ehelyett a fejlesztők klónokat (fork) készítenek róla, és egymástól függetlenül végzik a módosításaikat. Amikor a változást elküldésre érdemesnek ítélik, a GitHub felületén létrehoznak egy pull request kérelmet, amely meghatározza melyik ágat kívánják a központi tárolóba olvasztani. A projekt vezetőségének egyik tagja átnézi ezeket a változtatásokat, majd esetleges egyeztetés és javítási körök után, beolvasztja vagy elutasítja a kérelmet. G. Gousios et al. kutatásaiból kiderül, hogy pull request alapú fejlesztési stratégia használata egyelőre nem túl magas, körülbelül 14% körüli értékre tehető [14], azonban a vonzóan egyszerű kezelőfelület, a kötetlen csatlakozás lehetősége és könnyen ellenőrizhető változások a jövőben akár uralkodó formává is tehetik ezt a fajta fejlesztési modellt.

2. VERZIÓKÖVETÉS A KÖZIGAZGATÁSBAN

Amint arról korábban szó volt, a verziókövető rendszerek képességei hagyományosan három területen hasznosíthatók: a konfiguráció menedzsment, a dokumentáció nyilvántartás és fejlesztés, valamint a programfejlesztés területein. Információs technológiákkal sűrűn átszőtt világunk minden nagyobb szervezetének munkájában, így a közigazgatás szervezeteinek feladatai közt is felfedezhető mindhárom terület.

5 https://github.com/, http://sourceforge.net/, https://code.google.com/, https://bitbucket.org/, https://www.assembla.com/, https://www.codeplex.com/, https://launchpad.net/, https://gitorious.org/
6 social coding

Ugyanakkor, bár a feladatok és célok hasonlóak, a prioritások eltérőek lehetnek egy nyílt forrású közösség és a közigazgatás területén. A verziókezelők, különösen a elosztott verziókezelők újszerű képességeinek kihasználása iránt egyértelműen érezhető a kormányzati szféra érdeklődése, amit az alábbiakban néhány példán keresztül igyekszem illusztrálni. Az érdeklődés egyik oka az lehet, hogy az állami szektor dokumentumkezelési eljárásai nem mindig felelnek meg korunk igényeinek. A Coleman Parkes Research által 2012-ben végzett felmérés szerint az állami szektorhoz közel álló európai vállalatok kevesebb, mint felében találták az alkalmazottak fejlettnek a dokumentum-kezelési stratégiát [15]. Emellett a nyílt kormányzat[7] digitális megvalósítása kapcsán szintén felmerülhet a DVCS rendszerek lehetőségeinek kiaknázása.

A kormányzati szervezetek sok szempontból erősen hasonlítanak a nagyvállalatok működéséhez, a nagyobb szervezetek működtetéséhez pedig hozzá tartozik a folyamatos fejlesztés. Ez a fejlesztés hagyományosan erősen központosított, ugyanakkor egyre gyakrabban támaszkodik nyílt forrású elemekre. A DVCS alkalmazása ilyenformán két területen kerülhet előtérbe. Egyrészt, mint a belső központosított fejlesztés új eszköze, másrészt, mint a felhasznált nyílt forrású komponensek ellenőrzésére és támogatására használt kapcsolattartó eszköz.

A konfiguráció-menedzsment céljaira ma már többnyire integrált célrendszereket alkalmaznak, valamint alapvetően az üzemeltetés témakörébe esik, ezért – bár néhány helyen alkalmazzák – ezzel a területtel itt most nem foglalkozunk. Vizsgálatunk tárgyát képező közigazgatási DVCS felhasználási területek tehát az alábbiak lesznek:

1. dokumentumkezelés, jogszabályok;
2. belső fejlesztések;
3. F/LOSS8 biztonsági audit és kódkövetés.

Az alábbiakban ezekkel a területekkel kapcsolatos példákat mutatunk be, felhívva a figyelmet néhány érdekes lehetőségre és körülményre.

2.1. Példák a DVCS kormányzati alkalmazására

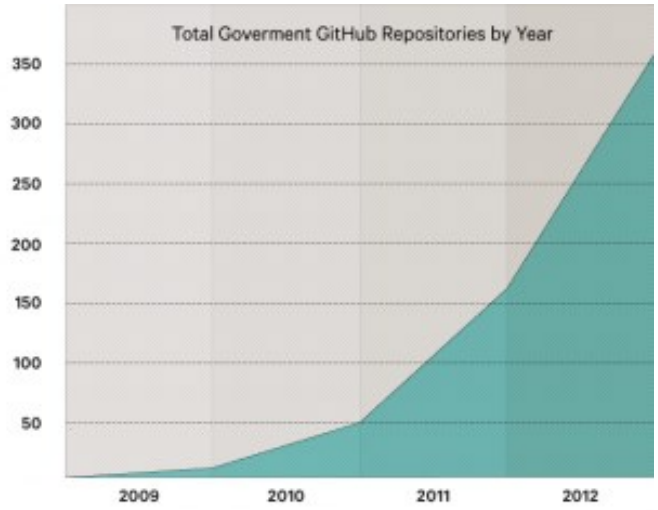
Az utóbbi években láthatóan felerősödött az érdeklődés a DVCS nyújtotta képességek iránt az közigazgatási szféra részéről. Bár használatuk messze nem széles körű, több olyan projekt is ismert, amelyek a nyíltság vagy egyszerűen csak technikai előnyök miatt a DVCS használata mellett tették le a voksukat. Az alábbiakban néhány ilyen példát mutatok be.

A Govcode[9] portálon összegyűjtött, DVCS alapokon elérhető számos nyílt forrású kormányzati projektek tanúsága szerint az Államokban már a valóságban is érezhető az Obama kormányzat fémjelezte Nyílt Kormányzás[10] hatása. Ágazatok széles skáláját felölelő projektek közt válogathatunk. Többek közt megtalálható itt számos API definíció, például a Fehér Ház weblap adatelérési szabványa[11], petíció keretrendszer, kézikönyvek, földrengés-esemény előrejelző rendszer, NASA küldetés irányító technológiák[12], rákkutatással kapcsolatos rendszerek, rádiótorony helymeghatározó rendszer és számos más projekt. Témánkat tekintve feltétlenül érdekes és tanulságok lista, amit azonban túlértékelni sem szabad. A Govcode egyéni kezdeményezés és nem állami portál, ráadásul számos projekt igencsak kezdetleges állapotban van, sokszor egyetlen commitot tartalmaz. A projektek elérhetőségének technikai hátterét sem mindenhol a kormányzati biztosítja, a legtöbb projekt a GitHub oldalain érhető el.

---

7 Open Government
8 Free / Libre and Open Source Software
9 Government Open Source Projects, http://www.govcode.org
10 Open Government Initiative
11 White House Web API Standards
12 NASA Mission Control Technologies, MCT.

Az Egyesült Királyság kormányzati portáljának forráskódja[13] megtalálható a Githubon. Kanada és Finnország kormányzata szintén tart fenn tárhelyet a Github oldalain. [16] Összességében a Githubon tárolt kormányzati tárhelyek száma jelentősen emelkedett az utóbbi években, 2013-ban már elérte a 350-et. [17]



1. ábra. Kormányzati tárhelyek a Githubon. (Forrás: Brian Ross/Wired)

Stefan Wehrmeyer aktivista és fejlesztő jóvoltából a teljes Deutche Bundesgesetze megtalálható GitHub portálon[14] [18]. Az írás születése pillanatában 126 fork létezik, ami jól mutatja az érdeklődést. A DVCS változat ugyanakkor nem hivatalos. A Git tárolóban lévő anyag a német kormányzat által elérhetővé tett XML formátumú állományokból generált Markdown[15] formátumú szövegeket tartalmaz.

2.2. Törvényalkotás

A törvények és jogszabályok rendszere néhány tekintetben igen hasonló a programok forráskódjához. A jogszabályok között függőségi viszony áll fenn, akárcsak a programrészek között. A módosításokat több időpontban több entitás végzi, és a tervezés során több alternatív vázlat létezhet egy időben hasonlóan az ágakhoz. A korábbi változatok ismerete és a változások kereshetősége éppúgy fontos, mint a forráskód esetében. A jogszabályokat azonban jelenleg a társadalom igen szűk rétege alkotja, valahogy úgy, ahogy az üzleti szoftverek forrását is egy igen szűk programozócsapat készíti.

Manapság a nyílt forrású szoftverek dinamikus térhódítását tapasztalhatjuk [19]. Az átláthatóság, nyíltság, és nem utolsósorban a DVCS rendszereknek köszönhető hatékony közösségi munka nagymértékben megnövelte a nyílt fejlesztési modell népszerűségét. Mint láthattuk néhány ország már meg is tette a az első lépéseket egy nyílt, áttekinthető törvényi rendszer felé.

Clay Shirky TED talks előadásában érdekes kérdést feszeget. Elképzelhető-e vajon, hogy egy napon a törvényalkotás folyamata a jelenlegi nyílt forrású fejlesztési modellhez hasonló szerkezetűvé váljon? [20] Az előnyök nyilvánvalóak: bárki felvethet változtatásokat, egyes csoportok önállóan és szabadon alkothatnak jogszabály variánsokat, a változtatások könnyen és gyorsan áttekinthetőek, a változások beolvasztása a hivatalos verzióba pedig egyszerű és követhető. Arra is lehetőség lenne, hogy az egyes változások szerzőségi információit nyilvántartsuk.

13 http://gov.uk
14 https://github.com/bundestag/gesetze
15 XHTML formátumba konvertálható, könnyen olvasható és írható strukturált szöveges állomány.

Természetesen rengeteg a nyitott kérdés ezen a területen. Bár a technológia adott és a programfejlesztésben évek óta hatékonyan használják, a nyílt, elosztott törvényalkotásra valószínűleg a fejlett demokráciákban is jó ideig várni kell, még ha egyébként életképesnek is bizonyulna a módszer. Egyrészt a felhasználók tábora a jogi területen lényegesen kevésbé technikai beállítottságú, másrészről a hagyományosan erősen központosított törvényalkotás folyamatának elosztottá alakításához a teljes társadalom részéről alapvető koncepcióváltás szükséges. Mivel ezek nem technológiai feltételek, a tényleges megvalósításhoz megfelelő környezetet valószínűleg csak egy generáció váltás teremtheti meg. A jövő lehetőségei és jelenleg is kihasználható előnyök viszont azt mutatják, hogy igenis érdemes foglalkozni a DVCS-szerű képességek kihasználásának gondolatával a törvényalkotás területén.

2.3. Belső fejlesztés és együttműködés a FLOSS közösséggel

Szinte minden nyílt forrású projekt használ valamilyen VCS rendszert, egyre gyakrabban DVCS rendszert a forráskód követésére. Pontos statisztikát készíteni a nyílt forrású VCS felhasználásról igen nehézkes, de a népszerű verziókövető szolgáltatások felhasználóinak és projektjeinek száma azt sugallja, hogy napjainkban már ez a nyílt forrású fejlesztések domináns, majdhogynem kizárólagos formája. Referenciaként említhető, hogy a GitHub népszerűbb közösségi fejlesztői oldal 2014-ben 5.7 millió felhasználót és 12.1 millió projektet tart nyilván[16]. Természetesen a legnagyobb nyílt forrású projektek, mint a LibreOffice, OpenStack, Hadoop, Android, Webkit, Firefox, Apache vagy a Linux forráskódja az interneten keresztül néhány kattintással legalább egy VCS rendszeren keresztül elérhető.

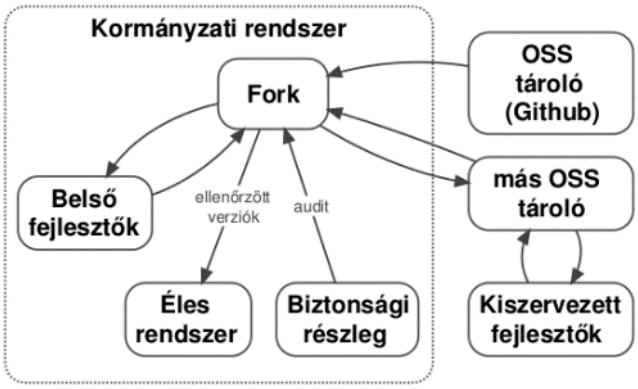
Általában véve a nagyvállalati szférában és a kormányzati környezetben is egyre gyakrabban találkozhatunk nyílt forrású fejlesztésekkel. A nyílt forrás felbukkanhat beágyazott rendszerként, mint az Android vagy routerek szoftvere, használhatjuk önálló alkalmazásként, ahogy az Apache webszervert vagy a Chrome böngészőt, és megjelenhet, mint valamely üzleti szoftver felhasznált komponense, mint a Python, az OpenStack, MySQL vagy egyéb FLOSS[17] szoftverkönyvtár. A közigazgatási és kormányzati rendszerek esetében különösen fontos szempont a gyártófüggetlenség és a biztonság, amelyet nyílt forrás segítségével biztosítani lehet. Ugyanakkor a nyílt fejlesztési modell sajátosságai miatt pusztán felhasználóként viszonyulni a nyílt fejlesztésekhez kockázatos. A nyílt projektek, különösen a kisebbek, elhagyatottá válhatnak és folyamatos kapcsolat nélkül a fejlesztés koordináló személyi állományra vagy a fejlesztés minőségére éppúgy nincs rálátásunk mint a üzleti szoftver esetében. Míg azonban az üzleti szoftver esetében bizonyos garanciát jelenthet a fejlesztők egzisztenciális függése a projekttől valamint az esetleges külső audit, a nyílt forrás esetében erről gyakran le kell mondanunk.

Gyakran előfordul, hogy egy széles körben alkalmazott OSS projekt karbantartói financiális problémákkal küzdenek. Az OpenBSD alapítvány például pénzügyi források híján 2014-ben közel állt a megszűnéshez. A probléma súlyosságát a közelmúlt nagy publicitást kapott sérülékenysége, a „Heartbleed bug” példája mutatja meg a legjobban. Az igen népszerű, számos alkalmazásban használt OpenSSL projekt egyik moduljában 2014 áprilisában súlyos sérülékenységet fedeztek fel, amellyel jelszavak illetve a privát kulcs is megszerezhető. A sérülékenységet hordozó változat 2012 márciusában került az éles változatba, azaz két éven át felderítetlen maradt. [21] A kódot számos vállalat és kormányzat használta, a probléma weblapok, VPN[18] hálózatok, levelezőszerverek és webalkalmazások tömegét, egyes becslések szerint a titkosított weblapok kétharmadát [22] érintette, ugyanakkor az igen összetett kód ellenőrzésére az erőteljesen alulfinanszírozott alapítványnak nem volt erőforrása. Steve Marquess az alapítvány elnöke szerint: “bár az OpenSSL az embereké, nem realisztikus és nem

16 https://github.com/about/press
17 Free Libre and Open Source, azaz szabad (nyílt forráskódú) szoftver
18 Virtual Private Network, virtuális magánhálózat.

is helyénvaló elvárni, hogy néhány száz vagy akár néhány ezer magánember biztosítsa a financiális támogatást". [23] Vizsgálódásunk tárgyát illetően különösen fontos, hogy a jelek szerint a kormányzatok kifejezetten lassan reagálhatnak egy ilyen problémára. A hiba publikálását követő napon a kanadai adóhatóságtól[19] 900 társadalombiztosítási számot tulajdonítottak el a lassú reagálás miatt [24]. Az események tükrében szakmai körökben felmerült a kormányzat felelősségének kérdése, illetve hogy szükséges lehet a kritikus nyílt forrású alkalmazások biztonsági ellenőrzését központi forrásból finanszírozni [22].

A fentiek alapján, véleményem szerint a nyílt forrású szoftverek biztonságos és hatékony felhasználása csak aktív állami együttműködés mellett képzelhető el a közigazgatás területén. Az állami szerepvállalás formája lehet például a visszacsatolt fejlesztés (aktív fejlesztés), az egységes kódfelügyeleti információ biztosítása (információs adatbázis, portál), financiális és jogi támogatás illetve a kormányzati eseménykezelő központok kifejezetten nyílt forrásra specializálódott részlege.



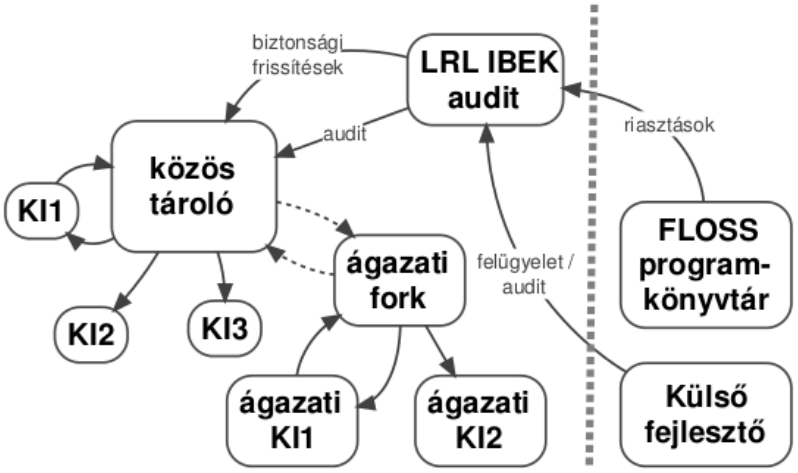
2. ábra. Példa az OSS-kormányzati fejlesztés kapcsolatára

Elosztott verziókövetők segítségével fejlesztés igen sokrétűen szervezhető. Illusztrációként bemutatok két önkényesen választott vázlatos lehetőséget, amelyeken keresztül talán jobban érzékelhető a rugalmasság és változatosság.

A 2. ábrán valamely nyílt forrású komponenseket hasznosító kormányzati rendszer lehetséges fejlesztési struktúráját mutatjuk be. A felhasznált szoftver Github tárolójáról helyi fork[20] készül, amelyben a belső fejlesztéshez külön ágat hozunk létre. A belső audit (kódminőség, biztonság, minőségbiztosítás) ezt az ágat ellenőrzi, a problémákat a belső hibakövető rendszeren keresztül jelenti. Az egyes fejlesztők saját másolataikon végezve a munkát, ami így lehet távmunka is, rendszeres időközönként beolvasztják a módosításokat a belső ágba. A OSS kapcsolattartó részleg ezek közül a változtatások közül bizonyosakat visszaolvaszt a közösségi változatba. Amennyiben a belső fejlesztői kapacitás elégtelennek tűnik egy bizonyos feladat elvégzéséhez, a kiszervezett fejlesztőcsapatnak nem feltétlenül szükséges hozzáférést adni a belső változathoz, a szükséges változtatásokat a közösségi változat megfelelő ágán elvégezve, azok folyamatosan visszaolvaszthatók belső rendszerbe. Mindeközben a ténylegesen használt éles rendszer kizárólag a biztonsági részleg által jóváhagyott folyamatosan ellenőrzött változatokból álló kóddal fut.

19 Canada Revenue Agency

20 A szoftvercsomag teljes forráskódjának lemásolásával induló független fejlesztés.



3. ábra. Egy lehetséges audit konfiguráció

A 3. ábrán a DVCS egy másik lehetséges alkalmazási módját mutatom be a kritikus infrastruktúrák területén. Az széles körben használt szoftver (amely lehet OSS vagy zárt belső fejlesztés) központi korlátozott hozzáférésű DVCS tároló készül. Az egyes szervezetek minden verziót innen szereznek be, amennyiben egyedi fejlesztéseket végeznek külön ágba szervezve ide csatolják vissza. Az egyes ágazatok speciális igényeihez saját ágazati forkot vagy ágat hoznak létre. Mindeközben a változtatások cseréje egyszerű marad. A kritikus infrastruktúrákkal kapcsolatos hálózatbiztonsági feladatok elvégzéséért és sérülékenységek publikálásáért felelős Létfontosságú Rendszerek és Létesítmények Informatikai Biztonsági Eseménykezelő Központja (LRLIBEK) folyamatos ellenőrzi a tárolók változásait, valamint elvégzi a szükséges biztonsági frissítéseket a külső forrásból felhasznált elemeken. Az egyes szervezetek közötti együttműködés egészen magas szinten valósítható meg, azonos funkciókat nem szükséges többször implementálni, a külső forrásból származó kódellenőrzést egyetlen szervezet végzi egy alkalommal és nem pusztán riasztásokat szolgáltat, hanem aktívan azonnali hatállyal javítja azokat. A kritikus infrastruktúra fejlesztőinek pusztán a megfelelő funkció ágat (feature branch) kell az stabil és a fejlesztői ágba olvasztaniuk.

A hatékony együttműködés és kódvizsgálat előfeltétele, hogy ismerjük és hatékonyan használjuk azokat az eszközöket, amellyel maga a fejlesztőközösség is dolgozik. A fenti példákon keresztül talán sikerült rámutatni, hogy e hatékony eszköz alkalmazása az egyszerű együttműködésen jóval túlmutató előnyökkel is járhat. Az elosztott verziókövetők használata az ellenőrzési folyamatokban elengedhetetlen, ugyanakkor érdemes megjegyezni, hogy az a kódot vagy a fejlesztés módszertanát elemző vizsgálat, amely kizárólag a verziókövetők adataira támaszkodik nem feltétlen nyújt átfogó képet [10].

3. DVCS ÉRTÉKELÉSI SZEMPONTRENDSZERE

Az eddigiekben felvázoltuk azokat a lehetséges kormányzati felhasználási területeket, ahol a DVCS alkalmazása előnyös lehet. A következő kérdés, hogy mi alapján választhatjuk ki a leginkább megfelelő variánst, illetve a jelenleg használatban lévő megoldások valóban teljes mértékben biztosítják-e a szükséges képességeket. Ennek érdekében első lépésként meghatározom az elosztott verziókezelők gyakran felmerülő értékelési szempontjait, majd kiemelem közülük azokat, amelyeket az adott területen a legfontosabbnak tartok. Természetesen a pontos eredmény érdekében minden esetet egyedileg kell megvizsgálni, itt mindössze felhasználási területenként egy-egy lehetséges prioritási sorrendet mutatok be.

A kiemelt tulajdonságok alapján viszonylag gyorsan eldönthető, hogy az esetleges előnyök arányban állnak-e a várható befektetéssel, létezik-e minden igényt kielégítő megoldás, illetve milyen szempontokra érdemes odafigyelni a szoftverválasztás során.

3.1. Elosztott verziókövetők értékelési szempontjai

Az általános verziókövető rendszerek azonos célokat valósítanak meg, közel azonos vagy hasonló képességkészlettel rendelkeznek, de néhány jellemzőjük eltérő. A vizsgálathoz használható értékelési szempontok gyűjtésénél két módszert használtam. A népszerű verziókövető rendszerek dokumentációjában előnyként említett tulajdonságokat vettem alapul, ezt egészítettem ki internetes fórumokon és portálokon (Stack Exchange különféle portáljai) jellemzően előforduló észrevételekkel, azaz olyan szempontokkal, amelyeket a kérdezők vagy válaszolók saját területükön fontosnak tartottak. Ezek alapján az elosztott verziókezelők legfontosabb paraméterei az alábbiak:

- Tanulhatóság (learning curve). a hatékony használati szint elsajátításához szükséges erőforrás mennyisége. Esetükben a képzés idejét és költségét befolyásolja.
- Adatbiztonság.
- Konzisztencia. Azonos állapot megőrzése valamennyi csomóponton. Magas elérhetőségű elosztott rendszereknél nem kivitelezhető, tehát az inkonzisztencia elfogadható mértékét és jellegét értjük alatta.
- Rendelkezésre állás és megbízhatóság. Mennyire stabilan és biztosan tudja a rendszer biztosítani az adatok elérhetőségét és az elvárt működést. Esetünkben kritikus rendszereknél nagy jelentősége lehet.
- Karbantarthatóság. Az infrastruktúra működtetésének erőforrás igénye.
- Átjárhatóság. Az adatok konvertálhatósága más verziókezelő rendszerek alá.
- Használhatóság. Az aktív használat egyszerűsége és gyorsasága. Felhasználó interfész és sebesség, azaz a felhasználói élmény megfelelősége.
- Elterjedtség. A szoftverfejlesztés területén általános értelemben véve.
- Sértetlenség védelme (data integrity). A változási napló sértetlenségének biztosítása.
- Commit szerzőjének azonosíthatósága.
- Naplómódosítási képességek. A napló egyszerűsítési lehetőségei (rebase, squash) az átláthatóság érdekében.
- Jogosultságrendszer. Az adathalmaz objektumainak, esetleg korábbi verzióinak hozzáférési jogosultság-ellenőrzése.
- Ágak egyszerű kezelése.
- Skálázhatóság.
- Egyszerű offline munka. A központ elérése nélkül is egyszerű munkavégzés.

Az utóbbi két kritériumot az elosztott kialakításból kifolyóan valamennyi DVCS jól teljesíti, nagy eltérések nem várhatók, de a teljesség kedvéért helye van a felsorolásban. A népszerű verziókezelők teljes körű értékelése a fenti szempontrendszer alapján jócskán túlmutat írásunk lehetőségein. A jelen cikk célja éppen egy ilyen jellegű kutatás előkészítése, ezért itt most csak néhány kiragadott, jellemző esetet mutatunk be. Az igen népszerű git verziókezelő viszonylag magas tanulási igényű, ugyanakkor igen rugalmas, ellenőrző összegek és GPG[21] kulcsok segítségével képes az adatintegritás megőrzésére, lehetővé teszi a napló módosítását, és szinte minden belső művelete vezérelhető parancssorból. Az egyre növekvő népszerűségű mercurial verziókezelő ezzel szemben a könnyű kezelhetőséget tartja szem előtt, parancsai hasonlítanak

21 Gnu Privacy Guard. A PGP kriptográfiai szoftver GPL licenszű implementációja.

a még mindig vezet SVN utasításaihoz de nem ad lehetőséget a napló módosítására. Beépített jogosultságrendszert egyik megoldás sem tartalmaz, ezt inkább külső megoldásra bízzák.

3.2. Dokumentum kezelés

A közigazgatási szféra dokumentum-kezelési feltételei alapvetően az adott jogi környezettől függenek, ezért az alábbiakban a hazánkban alkalmazható megoldásokra koncentrálunk. A közfeladatot ellátó szerveknél alkalmazható iratkezelési szoftverekkel szemben támasztott követelményeket Magyarországon a 24/2006. (IV. 29.) BM-IHM-NKÖM együttes rendelet szabályozza. A hazai szervezetek ennek megfelelő ügyiratkezelő rendszert működtetnek, tehát DVCS alkalmazása csak akkor lehetséges amennyiben a szabályozásnak megfelelő ügyiratkezelő rendszerbe integrálható. A jogszabály előírja az elektronikus ügyiratokkal kapcsolatos összes változtatásra irányuló művelet és változtatás dátum szerinti naplózását, a tartalmi módosítások rögzítését ugyanakkor nem szabályozza. A dokumentum integritásának védelmét a jogszabály kiemeli, de az elektronikus aláírás használatát nem teszi kötelezővé.

Véleményem szerint egy dokumentum-kezelő rendszer csak akkor teljes értékű, ha megfelelő jogosultságok kikényszerítése, integritás védelem és az eseménynaplózás mellett képes az információ változását is gyorsan és hatékonyan megjeleníteni. A fájlonként tárolt verziókból ugyan elvben kinyerhető a változás, de amellett, hogy a tárolás így feleslegesen redundáns, a két eltérő verziójú állomány összehasonlítása is nehézkes. Különösen igaz ez ha két verzió közt formátum vagy platformváltás történt. Az irodai szoftverek beépített változáskövető funkciója bizonyos szintig megoldást jelenthet, ugyanakkor a redundáns tárolás és a platformváltás problémája továbbra is fennáll.

Gyártófüggetlen, hosszú távú megoldást csak a nyílt szabványok alkalmazása jelenthet. A European Interoperability Framework for Pan-European e-Government Services ajánlásai közt szerepel a nyílt szabványok, nyílt forráskód használata, az információbiztonság megőrzése úgy, hogy minél szélesebb rétegek nyerjenek hozzáférést, továbbá hangsúlyozza a központilag egyeztetett XML séma alkalmazásának fontosságát [25]. A tömörítetlen XML alapú állományok jól kezelhetők az általános célú VCS rendszerekkel, így alkalmazásuk a dokumentum-kezelés terén indokolt lehet.

A jogszabály előírásait és a DVCS rendszerek képességeit összehasonlítva egyértelműen látszik, hogy önmagukban ezek a rendszerek elégtelenek a nyilvántartás vezetésére. A részletes, objektum szintű jogosultságrendszer, nagy mennyiségű metaadat tárolási lehetőség nemigen szerepel egy átlagos DVCS rendszer kitűzött céljai között. Ugyanakkor figyelembe véve, hogy a nyílt rendszerek egyik nagy előnye éppen a könnyű módosíthatóság és jó integrálhatóság, ez a probléma megfelelő ráfordítással áthidalható.

A jogszabályi előírásokat figyelembe véve a közigazgatásban dokumentumkezelés céljára alkalmazott DVCS rendszer legértékesebb tulajdonságai az adatbiztonság, megbízhatóság, napló sértetlenségének biztosítása és a szerző azonosíthatósága valamint a jogosultságrendszer. Ez utóbbit jelenleg egyetlen DVCS rendszer sem valósítja meg az elvárt mértékben. Nagy mennyiségű adathalmaz esetén a skálázhatóság szintén fontos szempont lehet, bár ebben az esetben valószínűleg nem járható út, hogy minden csomópont a teljes adathalmazt tárolja, tehát hierarchikus, részekre osztott felépítés kialakítása tűnik célravezetőnek. Tekintve, hogy esetünkben a végfelhasználók szakmáját figyelembe véve a parancssorból való működtetés lehetőségét nyugodtan kizárhatjuk, a tanulhatósági és karbantartási szempontok kisebb súllyal esnek latba, hiszen egy esetleges integráció során a szükséges grafikus felületeket amúgy is ki kell alakítani.

Úgy vélem a alkalmazhatóság legkomolyabb akadálya a megfelelő jogosultság rendszer hiánya. A DVCS rendszerek elsődleges alkalmazási területe a nyílt forrású programfejlesztés, ahol a teljes adathalmaz definíció szerint hozzáférhető. A közigazgatás dokumentumainak esetében ez sokszor szintén kívánatos, más esetben, például minősített adatok vonatkozásában

viszont jogszabály által tiltott. Néhány verziókövető rendszer lehetővé teszi az információ elérésének korlátozását, másokhoz található ilyen célú kiegészítés, egyes üzleti DVCS változatok pedig kifejezetten támogatják az ACL[22] listák alkalmazását [26], tehát a probléma nem megoldhatatlan, de további vizsgálatokat igényel.

Milyen előnyöket lehet szembeállítani a várható nehézségekkel? A DVCS rendszerek alkalmazásától várható legfontosabb előnyök: robosztusság, gyártófüggetlenség, nyíltság, precíz és gyors eltérésvizsgálat valamint a távoli (otthoni vagy kiszervezett) munkavégzés esetén elérhető nagyobb hatékonyság. Hazánkban ugyan a távmunka még nem túl elterjedt gyakorlat, a világban tapasztalható trendek mindenképpen efelé mutatnak.

3.3. Belső fejlesztés

A kor követelményeinek megfelelni akaró szoftverprojekt ma már nehezen képzelhető el valamilyen verziókezelő rendszer alkalmazása nélkül. Ez természetesen éppúgy igaz a közigazgatás szervezeteinek belső projektjeire is, különösen akkor, ha az adott területen nagyobb létszámú fejlesztőcsoport dolgozik. A megfelelő rendszer kiválasztásának szempontjai jobbára egybeesnek a szoftverfejlesztés általános szempontjaival, de esetünkben kicsit erősebb hangsúlyt kapnak az együttműködést, dokumentum- és gyártófüggetlenséget célzó elvek. Az európai együttműködést segítő JoinUp portál[23] például több száz közigazgatással kapcsolatos projektet fog össze. A fejlesztések eredménye alatt nem feltétlenül forráskódot kell érteni. Az együttműködés keretében megosztandó információ nagy része API definíció, XML séma, szótár és taxonómia.

A fent említett adatok modern formátumokban kiválóan kezelhetők verziókezelő rendszerekkel, tehát a DVCS bővebb képességkészletének kihasználása itt is előnyökkel járhat. A bizonyítottan sikeres együttműködést megvalósító nyílt fejlesztői modellek akár mintaként is szolgálhatnak egy ágazatközi, nyílt fejlesztési-információ, dokumentum, API és forráskódmegosztó rendszer kialakításakor. A fent említett JoinUp portál alkalmazásai például valószínűleg szorosabb együttműködést tehetnének lehetővé, ha egyszerű tömörített fájlok helyett DVCS alapokon osztanák meg a forrásokat. Az együttműködési platformok szervezett DVCS támogatása nagyban ösztönözheti a technológia aktív belső használatát.

A belső fejlesztések esetén általános elvárás a jó tanulhatóság, a magas rendelkezésre állás, használhatóság és a karbantarthatóság. A közigazgatás területén nem ritka kritikus rendszerek esetén viszont a sértetlenség védelme lesz az egyik legfontosabb szempont.

Amennyiben belső fejlesztést segítő együttműködési hálózatban gondolkodunk, előtérbe kerül a jogosultságrendszer, valamint a változások szerzői információinak nyilvántarthatósága. A nagyfokú skálázhatóság igénye a várhatóan hierarchikus kialakítás miatt csak különlegesen nagy projektek esetében merülhet fel. Feltételezve a intraneten meglévő illetve szervezetközi dedikált, jó hálózati kapcsolatot, az egyszerű offline munka nem jelent nagy előnyt. Amennyiben viszont a vezetés kiszervezett munkavégzésben illetve távmunkában gondolkodik, az offline munkavégzés képessége jelentősen felértékelődik.

3.4. Szoftverellenőrzés

A nyílt forrású szoftverek kiválasztási és ellenőrzési folyamata az egyetlen, ahol elkerülhetetlenül kapcsolatba kerülünk a DVCS rendszerekkel. Maga az ellenőrzés is csak így valósítható meg, továbbá, mint korábban felmerült, a tartós, hatékony és biztonságos nyílt forrás felhasználás előfeltétele a közvetett vagy közvetlen részvétel a fejlesztésben.

Felhozható érvként, hogy nem sok értelme van a megfelelő DVCS rendszer kiválasztásával bajlódni, hiszen ha a fejlesztés irányítása nincs a kezünkben, legfeljebb alkalmazkodni tudunk a meglévő megoldáshoz. Ez azonban nem feltétlenül igaz. Egyrészt sok elosztott verziókövető

22 Access Control List, jogosultságlista
23 https://joinup.ec.europa.eu

szolgáltató több eltérő verziókövető formátumban is elérhetővé teszi ugyanazt a tárhelyet, tehát a választás lehetősége adott. Másrészről, a kormányzat irányából érkező pénzügyi vagy szakmai támogatás igen kedvező feltételeket teremthet egy nyílt projekt számára, így jó eséllyel meggyőzhető a közigazgatás számára inkább megfelelőbb megoldás használatáról. Mindamellett az sem példa nélkül való, hogy egy nyílt forrású fejlesztés központi irányítását állami szerv végezze[24]. Ennek fényében talán mégis van értelme a szoftverellenőrzés céljaira leginkább alkalmas DVCS platformról beszélni.

Figyelembe véve, hogy elsődleges célunk a folyamatos biztonsági audit és a projektkövetés, a legfontosabb képességek a napló sértetlenségének védelme valamint a commit szerzőjének azonosíthatósága. Fontos szempont természetesen az elterjedtség illetve átjárhatóság, hiszen annál kevesebb esetben kell a konverzióval járó esetleges információveszteséggel számolni. A naplómodosítási képesség szempontunkból kétélű fegyvernek számít. Részben tisztább és egyértelműbb projekttörténetet kapunk, hiszen ez az elsődleges célja, ugyanakkor információt veszítünk, ami egy esetleges fejlesztői profil felállításakor vagy statisztikai adatok képzése során hiányozhat.

Aktív részvétel esetén a belső fejlesztések szempontjai köszönnek vissza, leszámítva a karbantarthatóságot, melynek terhét ez esetben valószínűleg amúgy sem a szervezet viseli.

4. ÖSSZEFOGLALÁS

A cikkben röviden összefoglaltam a jellemző DVCS képességeket majd ismert példák és következtetések alapján a beazonosítottam azokat a felhasználási területeket a közigazgatásban ahol ezek a képességek előnyösen kihasználhatók. Felhívom a figyelmet a megfelelő szabványos szöveges formátumok használatának fontosságára, amely az általános célú verziókezelők hatékonyságát nagyban megnöveli. Az általános értékelési szempontok meghatározása után röviden kiemeltem néhány, véleményem szerint fontos képességet minden egyes területen. A fentiek alapján látható, hogy a DVCS rendszerek változatlan formában történő felhasználásának egyik legnagyobb akadálya, kivált a dokumentum-kezelés és belső fejlesztések terén, a megfelelő részletességgel szabályozható jogosultságrendszer hiánya. A DVCS alkalmazása a közigazgatásban ugyanakkor kivitelezhető megoldás, sőt, egyes esetekben, mint például a nyílt rendszerek biztonsági ellenőrzése, tulajdonképpen elkerülhetetlen.

A bemutatott eredmények értékelésekor szem előtt kell tartani, hogy csak a kutatási területemhez kapcsolódó nyílt forrású verziókövető rendszerek képességeit vizsgáltam, az üzleti verziókövető rendszerek vagy hasonló feladatkört is ellátó integrált rendszerekre, mint például a Sharepoint szerver, nem tértem ki. A cikk csak az elemzési szempontokat és a potenciális felhasználási területekkel foglalkozik, az egyes változatok konkrét összehasonlító vizsgálata további kutatás feladata lehet. Az írás célja éppen egy ilyen elemzés megalapozása volt, de az eredmények önmagukban is felhasználhatók az állami szféra informatikai rendszereinek korszerűsítése során. Az bemutatott képességek és szempontrendszer annak eldöntésében nyújthat segítséget, hogy egy korszerűsítés keretében érdemes-e az adott rendszer esetében megfontolni egy modern DVCS képességeinek integrálását, hol várhatók előnyök illetve milyen akadályokkal kell számolni.

Felhasznált irodalom

[1] http://en.wikipedia.org/wiki/Comparison_of_revision_control_software

[2] http://en.wikipedia.org/wiki/Revision_control

24 https://github.com/WhiteHouse

[3] Nancy Lynch, Seth Gilbert: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. ACM SIGACT News. 2002, Vol. Volume 33, no. Issue 2, pp. 51–59.

[4] Andrew S. Tanenbaum, Maarten Van Steen: Distributed Systems: Principles and Paradigms. 2nd edition. S.l.: Prentice Hall, 2007. ISBN 0132392275.

[5] D.M.B.D. Kuhn: Distributed Version Control Systems. [online]. 2010, http://www.sonicwaves.de/downloads/publications/Distributed-Version-Control-Systems.pdf [2013-06-28]

[6] P. Mukherjee: A fully Decentralized, Peer-to-Peer Based Version Control System [online]. Ph.D. Thesis. TU Darmstadt, 2011. <http://tuprints.ulb.tu-darmstadt.de/id/eprint/2488> [2013-06-28]

[7] http://www.ohloh.net/repositories/compare [2014-05-08]

[8] C.F. Malmsten: Evolution of Version Control Systems-Comparing CENTRALIZED against DISTRIBUTED Version Control models [online]. Bachelor of Applied Information Technology Thesis. Gothenburg, Sweden: University of Gothenburg, 2010. <http://gupea.ub.gu.se/handle/2077/23474> [2013-06-28]

[9] Ian Clatworthy: Distributed Version Control Systems - Why and How. OSDC 2007. 2007.

[10] S. Negara, M. Vakilian, N. Chen, R.E. Johnson, D. Dig: Is it dangerous to use version control histories to study source code evolution? ECOOP 2012–Object-Oriented Programming [onlinel] http://courses.cs.washington.edu/courses/cse590n/12sp/danny.pdf [2014-05-08].

[11] Vincent Driessen: A successful Git branching model [online] http://nvie.com/posts/a-successful-git-branching-model/ [2014-03-10]

[12] D.M.B.D. Kuhn: Distributed Version Control Systems. [online]. 2010, http://www.sonicwaves.de/downloads/publications/Distributed-Version-Control-Systems.pdf [2013-06-28]

[13] Comparison of open-source software hosting facilities http://en.wikipedia.org/wiki/Comparison_of_open-source_software_hosting_facilities [2014-03-28]

[14] Georgios Gousios, Martin Pinzger, Arie van Deursen: An Exploratory Study of the Pull-based Software Development Model. Delft University of Technology Software Engineering Research Group Technical Report Series. 2013, ISSN 1872-5392.

[15] Coleman Parkes Research: Ricoh Document Governance Index 2012. http://www.ricoh.hu/about-ricoh/news/2013/ricoh_allam_dokumentum_felho.aspx [2014-03-10]

[16] Nico Adams: Version control for open government [online] http://scimantica.com/blog/2013/3/10/version-control-for-open-government [2014-03-10]

[17] Robert Mcmillan: How GitHub Helps You Hack the Government [online] http://www.wired.com/2013/01/hack-the-government/ [2014-04-16]

[18] Robert Mcmillan: German Federal Law Now on GitHub http://www.wired.com/2012/08/bundestag/ [2014-02-02]

[19] Amit Deshpande, Dirk Riehle: The Total Growth of Open Source. Proceedings of the Fourth Conference on Open Source Systems (OSS 2008) [online]. Springer Verlag, 2008. pp. 197–209. http://dirkriehle.com/publications/2008-2/the-total-growth-of-open-source/ [2014-03-26]

[20] Clay Shirky: How the Internet will (one day) transform government [online]. TED talk. 2012. https://www.ted.com/talks/clay_shirky_how_the_internet_will_one_day_transform_government

[21] Codenomicon: The Heartbleed Bug http://heartbleed.com/ [2014-04-10]

[22] Heartbleed Shows Government Must Lead on Internet Security http://www.scientificamerican.com/article/heartbleed-shows-government-must-lead-on-internet-security/ [2014-04-16]

[23] OpenSSL foundation president asks for more financial support in the wake of heartbleed http://www.digitaltrends.com/computing/openssl-foundation-president-asks-financial-support-wake-heartbleed/ []

[24] Heartbleed bug shows governments slow to react https://ca.finance.yahoo.com/news/heartbleed-bug-shows-governments-slow-react-090000961.html [2014-04-16]

[25] Budai Balázs Benjámin: E-Közigazgatás Axiomatikus megközelítésben. Doktori értekezés. Pécs: Pécsi Tudományegyetem Állam- és Jogtudományi Kar, 2008.

[26] Pablo Santos: Distributed Version Control Systems in the Enterprise. InfoQ [online]. 2012 http://www.infoq.com/articles/DVCS-Enterprise [2014-04-18]