TARTU ÜLIKOOL
Arvutiteaduse instituut
Infotehnoloogia eriala

Irina Ivanova

Versiooniuuenduse automatiseerimine kasutades skriptimiskeelt Bash

Bakalaureusetöö (6 EAP)

Juhendajad: dotsent Helle Hein
Polina Morozova, MSc (Nortal AS)

Tartu 2016

Versiooniuuenduse automatiseerimine kasutades skriptimiskeelt Bash

Lühikokkuvõte:
Modulaarset süsteemi kasutavas projektis tehakse rakendusse palju uuendusi, eriti kui seda rakendust kasutavad mitu klienti. Manuaalne versioonivahetuse protsess võtab palju aega ja suurendab inimliku vea tekkimise tõenäosust. Bakalaureusetöö teoreetilise osa eesmärk on leida võimalikud lahendused uuendamisprotsessi automatiseerimiseks ja valida neist välja parim. Töö praktilise osa eesmärk on valitud lahenduse loomine ja selle rakendamine projektis.

Võtmesõnad:
Bash skript, versiooniuuendus, Apache Tomcat, Java, modulaarne süsteem

CERCS:
CERCS P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Version Update Automation Using Scripting Language Bash

Abstract:
There are a lot of application updates in a project, where modular system is in use, especially if the application is used by many clients. Manual version update process takes a lot of time and increases the possibility of human errors. The aim of the theoretical part of this Bachelor Thesis is to find possible solutions for the application update process automation and to choose the best one. The aim of the practical part is to create the solution and apply it to the project.

Keywords:
Bash script, version update, Apache Tomcat, Java, modular system

CERCS:
CERCS P170 Computer science, numerical analysis, systems, control

Sisukord

Sissejuhatus

Tarkvara versiooniuuendamine on protsess, mille korral veebiserverisse paigaldatakse mooduli uuemat versiooninumbrit, et rakendada selles olemas olevaid koodi parandusi või täiendusi. Tavaliselt manuaalne uuendus koosneb neljast põhi-sammust: uue versiooni allalaadimine, selle paigaldamine, vana versiooni eemaldamine ja uue versiooni staatuse kontrollimine. Mida rohkem on mooduleid ja veebiservereid, seda ajamahukamaks muutub versiooniuuendamise protsess, kuna neid samme on vaja korrata mitu korda.

Käesolevas lõputöös keskendutakse rakendusliku probleemi lahendamisele toimivas tarkvaraprojektis. Projekti olemuse tõttu võtab rakenduse versiooni uuendamine kaua aega ja eeldab palju manuaalset sekkumist. See kulutab ressursse ja suurendab inimliku vea tekkimise tõenäosust. Bakalaureusetöö teoreetilise osa eesmärk on leida võimalikud lahendused uuendamisprotsessi automatiseerimiseks ja valida neist välja parim. Praktilise osa eesmärk on valitud lahenduse loomine ja selle rakendamine projektis.

Töö koosneb viiest osast. Esimeses osas püstitatakse probleem: kirjeldatakse projekti arhitektuuri, manuaalset uuendamisprotsessi, senise protsessiga kaasnevaid probleeme ja nõudeid uuele lahendusele. Peamine raskus seisneb selles, et rakendus koosneb paarikümnest moodulist (täpne arv sõltub kliendi nõuetest), mis tuleb paigaldada 31 rakendust. Selleks, et paigaldada rakenduse uus versioon kõigisse veebiserveritesse, on vaja teha umbes 600 uuendust (arvestamata testrakenduste vaheuuendusi). Lisaks on projektis kasutusel kaks erinevat veebiserverit: Oracle WebLogic 12[1] ja Apache Tomcat 8[2]. Uuendamisprotsessi automatiseerimine peab arvestama kõigi nende nõuetega.

Töö teises osas uuritakse võimalikke lahendusi: kas kasutada turul olemasolevat toodet (töös uuritakse nelja toodet: Atlassian Bamboo, Chef, Jenkins ja Ansible Tower) või kirjutada ise tarkvara. Tuuakse välja iga lahenduse eeliseid ja puuduseid ning selgitatakse, miks langetati otsus Bash skripti kirjutamise kasuks. Valmistoodete peamised puudused on nende maksumus, võimetus hallata protsesse ja parandada vigu, vajadus kulutada palju aega toote tundmaõppimiseks ja rakendamiseks.

Kolmandas osas kirjeldatakse valitud lahendust: milline on skripti tööpõhimõte ja arhitektuur ning kuidas skripti saab paigaldada ja kasutada. Skripti põhifunktsioonid on moodulifaili allalaadimine, vana versiooni mahavõtmine, uue versiooni paigaldamine ja vajalike inimeste teavitamine toimunud versiooniuuendusest. Lisaks on oluliseks funktsiooniks erinevad kontrollid, mis vähendavad vale versiooni paigaldamise tõenäosust, samuti logimisfunktsioon, mis võimaldab hankida and-

[1] `https://www.oracle.com/middleware/weblogic`

[2] `http://tomcat.apache.org`

meid iga uuenduse kohta.

Neljandas osas uuritakse, kuidas valitud lahendus mõjutas projekti: võrreldakse versiooniuuendusele kulunud aega manuaalse ja automatiseeritud protsessi puhul ning vahendatakse kolleegidelt uue uuendamisprotsessi kasutamise kohta saadud tagasisidet.

Viimases osas tuuakse välja lahenduse kaugemaid perspektiive: kuidas saab loodud skripti edasi arendada ja teistes projektides rakendada.

Tööl on kaheksa lisa: terminid, ühe mooduli uuendamisprotsessi mudel, mitme mooduli uuendamisprotsessi mudel, skripti projekti juhend, skripti avalik juhend, kolleegide tagasiside uue skripti kohta, Bash skripti lähtekood ja litsents.

Käesoleva bakalaureusetöö tulemuseks on töötav, avatud lähtekoodiga skript Bash keeles ja selle dokumentatsioon.

1 Probleemi püstitus

Versioonide uuendamine on tarkvara arendamisel sage ja vajalik protsess. Käesolevas töös kirjeldatud projektis etendab see protsess väga olulist rolli.

Projekti rakendus on kirjutatud Javas ja kasutusel on war-failid. Lisaks on projektis kasutatud modulaarset süsteemi, mis tähendab, et kogu rakenduse kood on jagatud mooduliteks ja iga mooduli kohta on olemas eraldi war-fail. Kokku on umbes 20 moodulit (täpne arv sõltub kliendist, sest eri klientidel on erinev moodulite komplekt).

Rakendust kasutavad 10 klienti, kellest kolmel on olemas kolm rakendust: toode, demorakendus (kus kliendid tutvuvad enne tooteuuendust uute funktsioonide ja parandustega) ja testrakendus (kus projekti meeskond testib toote funktsioone enne selle tarnimist kliendile). Seitsmel kliendil on olemas kaks rakendust: toode ja testrakendus. See tähendab, et kokku on $3 * 3 + 7 * 2 = 23$ kliendirakendust. Lisaks on projektis veel umbes seitse testrakendust, mida kasutatakse eri faasides eri funktsioonide testimiseks. Seega tuleb kokku $23 + 7 = 30$ rakendust (arvestamata kohalikke ja virtuaalmasinaid, mida iga meeskonnaliige võib vajadusel luua), mida on samuti vaja regulaarselt uuendada. See tähendab, et rakenduse uue versiooni paigaldamiseks igasse veebiserverisse on vaja teha $30 * 20 = 600$ uuendust.

Toodud arvud näitavad, et versiooni manuaalne uuendamine nõuab palju aega ja tähelepanu, mistõttu otsustati uuendamisprotsessi automatiseerida.

Tabelis 1 on esitatud ühe mooduli manuaalse uuendamisprotsessi sammud.

Tabel 1. Manuaalse uuendamisprotsessi sammud.

Samm	**Tegevus**
1	Uue versiooninumbriga war-faili allalaadimine serverisse.
2	Kui veebiserveriks on Oracle WebLogic, siis war-faili prekompileerimine (Apache Tomcat veebiserveril seda sammu ei ole).
3	Faili ümbernimetamine.
4	Faili paigaldus.
5	Vana faili eemaldamine.
6	Uue versiooni staatuse kontrollimine.
7	JIRA[3] töö uuendamine. JIRA saadab automaatselt kollegidele teate versiooniuuenduse kohta.
8	Allalaaditud faili kustutamine.

Sammus 7 mainitud JIRA põhifunktsiooniks on ülesannete haldamine. Projektis seda kasutatakse versiooniuuenduste jälgimiseks: iga mooduli ja rakenduse kohta on olemas eraldi JIRA töö (*issue*), kus uuendatakse selle pealkirja (*summary*)

[3] `https://www.atlassian.com/software/jira`

versioonivahetuse järel. Need kasutajad, kes jälgivad (*watching*) selliseid töid saavad teavitusi töö pealkirja muutmise kohta. See võimaldab mitte ainult teavitada inimesi toimunud uuenduste kohta, vaid ka tekitada töölaud (*dashboard*), kus on nähtavad kõik versioonid, mis on paigaldatud konkreetsetes rakendustes.

Mitme mooduli uuendamisel tuleb läbida kõik sammud Tabelist 1 iga mooduliga eraldi. Neile tegevustele kuluv aeg sõltub moodulist (kuna mõnes moodulis võib olla teisest kordades rohkem koodi, siis selle allalaadimine ja paigaldamine võtab rohkem aega), aga keskmiselt on see 1.5 minutit. Kõikide moodulite uuendamine võib võtta aega ühe tunni, sest lisandub veel aeg õige versiooninumbri leidmiseks, war-faili aadressi kopeerimiseks, JIRA õige töö leidmiseks ja avamiseks jne. See tähendab, et inimlike vigade tekkimise tõenäosus on suur (nt mooduli vale versiooni allalaadimine).

Tabelis 2 on toodud nõuded versiooniuuenduse lihtsustamise lahendusele. Nende nõuete kehtestamisel on peetud silmas, et tulemuseks oleks lahendus, millesse on mõtet aega investeerida, sest uue protsessi loomine on ajamahukas tegevus ja sellest saadav tulu peab olema suurem sellele kulutatud ajast.

Tabel 2. Nõuded versiooniuuenduse lihtsustamise lahendusele.

Nr	**Nõue**	**Põhjendus**
1	Kiirus.	Uuendamisprotsess peab võtma varasemast vähem aega.
2	Mugavus.	Uuendamisprotsess peab eeldama võimalikult vähe käsitsi tegevusi.
3	Ühe ja mitme mooduli uuendamine.	Lahendus peab võimaldama nii ühe kui ka mitme mooduli samaaegset uuendamist.
4	Töötamine eriveebiserveritel.	Projekt kasutab kahte erinevat veebiserverit: Apache Tomcat 8 ja Oracle WebLogic 12.
5	Teavitussüsteem.	Meeskonnaliikmed soovivad saada tehtud uuenduste kohta teavitusi, mis tähendab.
6	Logimissüsteem.	Peab olema võimalus vaadata, kes ja millal uuendas mingit moodulit mingis rakenduses.
7	Mittetehniline lahendus.	Lahenduse kasutamine ei tohi nõuda erilisi tehnilisiteadmisi.
8	Lukustamissüsteem.	Uuendustega võib tegeleda samaaegselt mitu inimest, mis tähendab, et lahendusel peab olema lukustamissüsteem, vältimaks paralleelseid uuendusi samas serveris.
9	Töötamine mitmete serveritega.	Toodangurakendused kasutavad kahte serverit, seega peab lahendus oskama uuendada versiooni automaatselt kahel serveril.
10	Hallatav lahendus.	Lahendus peab olema täielikult projekti kontrolli all, et seda oleks võimalik igal ajal hallata, seadistada ja parandada.
11	Kiire arendusprotsess.	Lahenduse loomine ei tohi võtta palju aega, sest projektil puuduvad selleks inimressursid.
12	Töötamine moodulite erisufiksitega.	Projektis on olemas Eesti ja Leedu rakendused, kuhu paigaldatakse samad moodulid, aga erinevate sufiksitega. Näiteks, kui Eesti rakenduses paigaldatakse moodulit `person`, siis Leedu rakenduses `person-lt`, kuna seal on olemas eriloogika Leedu isikute jaoks.
13	Odav.	Lahendus ei tohi võtta palju raha.

2 Võimalikud lahendused

Uuendamisprotsesside automatiseerimiseks ja haldamiseks on olemas palju valmistooteid. Töös vaadeldatakse neli: Atlassian Bamboo [1], Chef [2], Jenkins [3] ja Ansible Tower [4].

Kõigi valmistoodete rakendamisel on alati nii eeliseid kui ka puudusi. Valmistoodete eelisteks on:

- Ei ole vaja kulutada aega juba olemasoleva süsteemi kirjutamisele.
- Turul olevate populaarsete rakenduste puhul on suur tõenäosus, et need on kvaliteetsed ja vastavad kasutajate ootustele.
- Vajadusel saab rakendada konkreetse tarkvara kasutajate kogukonna oskusteavet, abi ja kogemusi.

Olemasolevate süsteemide rakendamisel on ka terve rida puudusi:

- Suurem osa valmistoodetest on tasulised (Jenkins on erandiks) ning nende hind tavaliselt sõltub serverite, rakenduste ja moodulite arvust, mis vaadeldavas projektis on suur.
- Valmistoote tundmaõppimine, selle paigaldamine ja seadistamine võtab palju aega, eriti kui see pakub laia funktsioonide valikut.
- Projekti töö sõltub tootjast – vigade ilmnemisel ei saa olla kindel kas ja millal need parandatakse; probleemide tekkimisel ei saa olla kindel, et tootja suudab pakkuda kasutajatuge.
- Projekt peab usaldama tootjat – kui toode kood ei ole avalik, siis ei saa kontrollida selle turvalisust ja jõudlust.

Alternatiiv valmistoote kasutamisele on kirjutada oma lahendus. Kuna selline lahendus peab suhtlema teiste veebiteenustega ja töötama kõikides operatsioonisüsteemides sama moodi, siis valiti antud juhul lahenduse aluseks skriptimiskeel Bash [5, 6], mis on kõige mugavam veebiteenustega töötamiseks.

Tabelis 3 on toodud valmistoodete vastavus Tabelis 1 esitatud nõuetele. Kõige olulisem on 11. nõue, mille kohaselt "lahenduse loomine ei tohi võtta palju aega, sest projektil puuduvad selleks inimressursid".

Tabel 3. Võimalike lahenduste vastavus nõuetele.

Lahendus / Nõude Nr	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Atlassian Bamboo	x	x	x	x	x	x	x	x	x			x	x
Chef	x	x	x	x	x	x	x	x	x			x	
Jenkins	x	x	x	x	x	x	x	x	x	x		x	x
Ansible Tower	x	x	x	x	x	x	x	x	x			x	
Bash skript	x	x	x	x	x	x	x	x	x	x	x	x	x

Atlassian Bamboo nõuab erilist tähelepanu, kuna projektis kasutatakse teisi Atlassiani tooteid, nagu JIRA, Confluence[4] ja Fisheye[5]. See tähendab esiteks, et Bamboo on antud projekti jaoks tasuta (tavaliselt on see tasuline), teiseks, Bamboo integreerub väga hästi projektiga, võimaldades luua seoseid rakenduse koodi, muudatuste kirjelduse ja uuenduste vahel. Lisaks, kasutavad antud toodet ka ettevõtte teised projektid, mis annab võimaluse kaasata inimesi, kellel on olemas teadmised ja kogemused Bamboo seadistamisest.

Teadmised aitavad seda rakendada võimalikult kiiresti, kuid kogemus näitas, et Bamboo lõplik rakendamine võttis teistes projektides siiski palju aega: näiteks, kulus ühes projektis ühe mooduli seadistamisele 1,5 kuud, mis 20 mooduli korral tähendaks $\sim 20 * 1.5 \approx 30$ kuud. Lisaks võib võtta üks uuendus palju aega, sest Bamboo tööpõhimõte eeldab kolmanda serveri kasutamist uuenduse tööplaani teostamiseks ja selliste serverite arv on piiratud. See tähendab, et kui korraga soovitakse uuendada mitut rakendust, siis võib tekkida ootejärjekord.

Chef ja Ansible Tower ei täida nõuded 10, 11 ja 13 – projekt ei saa neid kontrolleerida, nad nõuavad palju aega teadaõppimiseks ja seadistamiseks (ettevõtes puudub vastav kogemus) ja nad nõuavad palju raha. Näiteks, Ansible Tower Premium [6] pakub ööpäevaringset klienditugi, 100 sõlme (*node*) ja maksab €12,286 ($14,000) [7] aastas. Chef Premium [8] pakub ööpäevaringset klienditugi ja €111.45 ($127) aastas ühe sõlmi eest, mis 100 sõlme korral on €11,145.24 ($12,700).

Jenkins ei vasta kõige olulisemale nõudele 11 – võtab aega toode uurimisele ja rakendamisele. Kuna see ei tegele ainult war-faili paigaldamisega, vaid ka selle ehitamisega, see tähendab, et projektis on vaja lisaks muuta versioonide ehitamise protsessi. See kaasneb rohkem inimressurssi, mis tähendab rohkem kulusid.

[4] `https://www.atlassian.com/software/confluence`

[5] `https://www.atlassian.com/software/fisheye`

[6] `https://www.ansible.com/pricing`

[7] Mõlemad Ansible Tower ja Chef hinnad on arvutatud kasutades Eesti panga eurokurssi kalkulaatorit veebilehel `http://www.eestipank.ee/valuutakursid` (10.05.2016).

[8] `https://www.chef.io/pricing`

Kõige sobilikum oleks uue Bash skripti kirjutamine, kuna see on ainus lahendus, mis vastab kõigile nõuetele ja, mis kõige olulisem, sealhulgas ka 11. nõudele. Võrreldes Bamboo lahendusega on sellel valikul ka üks puudus: seda ei saa integreerida teiste Atlassiani toodetega. Samas on sellel lahendusel ka oma lisaeelised:

- Skripti on võimalik arendada osade kaupa: alguses luua lihtsamad funktsioonid, mida hiljem saab järk-järgult täiendada uute ja keerulisemate funktsioonidega. See võimaldab kasutada ressursse väikeste osadena ja samal ajal töötada parema süsteemiga.
- Lahendus on väga paindlik: saab arendada konkreetsele projektile vajalikke funktsioone ning lisada seadistamise võimaluse, et sama lahendust saaks kasutada erinevates serverites ja rakendustes.
- Lahendus ei nõua projekti arhitektuuri ja protsesside muutmist: arendajate jaoks uue versiooni loomise protsess jääb samaks.

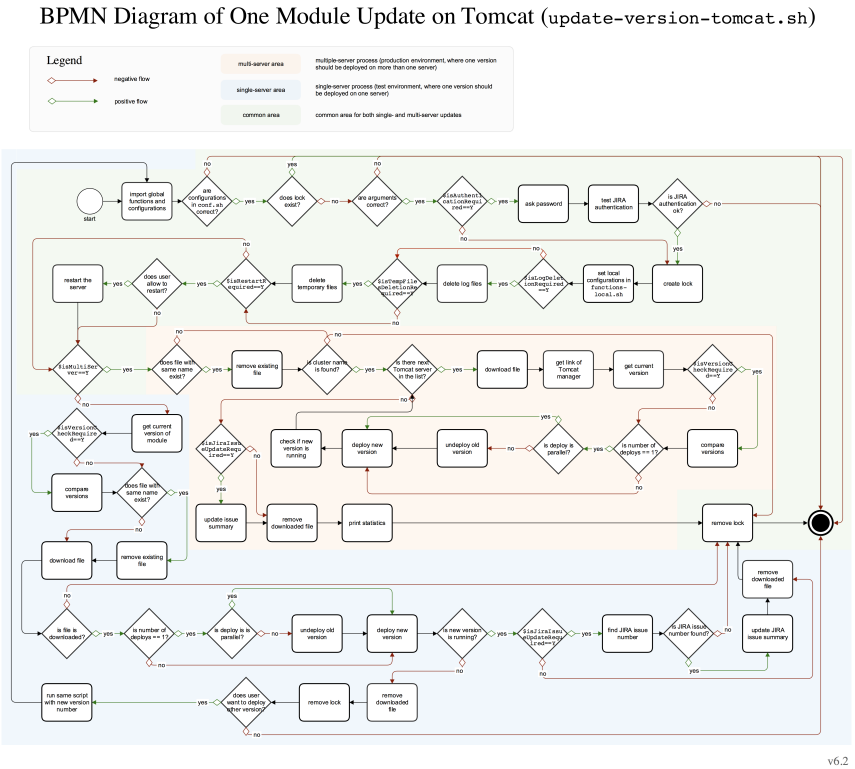
Arvestades kõikide lahenduste eeliseid ja puudusi, otsustati Bash skripti kirjutamise kasuks.

3 Versiooniuuenduse automatiseerimine kasutades skriptimiskeelt Bash

3.1 Lahenduse arhitektuur ja tööpõhimõte

Skript automatiseerib kõik tegevused Tabelis 1. See käivitatakse serveris, kus on vaja uuendada kas ühte või mitut moodulit ning parameetritena antakse ette mooduli nimetus, versioon ja JIRA kasutajatunnus (mitme mooduli uuendamisel mooduli nimetus ja versioon puuduvad).

Joonisel 1 on näidatud protsessid, mis toimuvad ühe mooduli uuendamisel Tomcat 8 veebiserveris (faili `update-version-tomcat.sh` käivitamisel). Joonis on originaalkujul Lisas II.



Joonis 1. Skripti protsessimudel ühe mooduli uuendamisel Tomcat 8 veebiserveril.

Joonisel 1 on toodud äriprotsesside mudel (*Business Process Modelling Notation*[9]), mis kirjeldab skripti teostatavaid samme ja kontrolle. Positiivsel töövool on rohelised nooled ja negatiivsel punased. Ühe mooduli uuendamisprotsess sõltub sellest, mitmes serveris uuendust tehakse: testrakendustel on üks server ja toodangurakendustel kaks serverit (skript töötab ka suurema serverite arvuga). Selle põhjal on töövoog jagatud kolmeks osaks: rohelise taustaga alal asuvad protsessid, mis on samad nii ühe kui ka mitme serveri puhul; kollasel taustal on mitme serveri protsessid ja sinisel ühe serveri protsessid. Voogude põhierinevus on kohtades, kus lõpetatakse skripti töö: nt kui mitme serveri korral ei õnnestunud mooduli paigaldamine ühes serveris, siis tuleb jätkata teise serveriga. Lisaks näidatakse mitme serveri korral töö lõpus statistikat (Joonis 2), millest selgub uuendamise seis igas serveris.

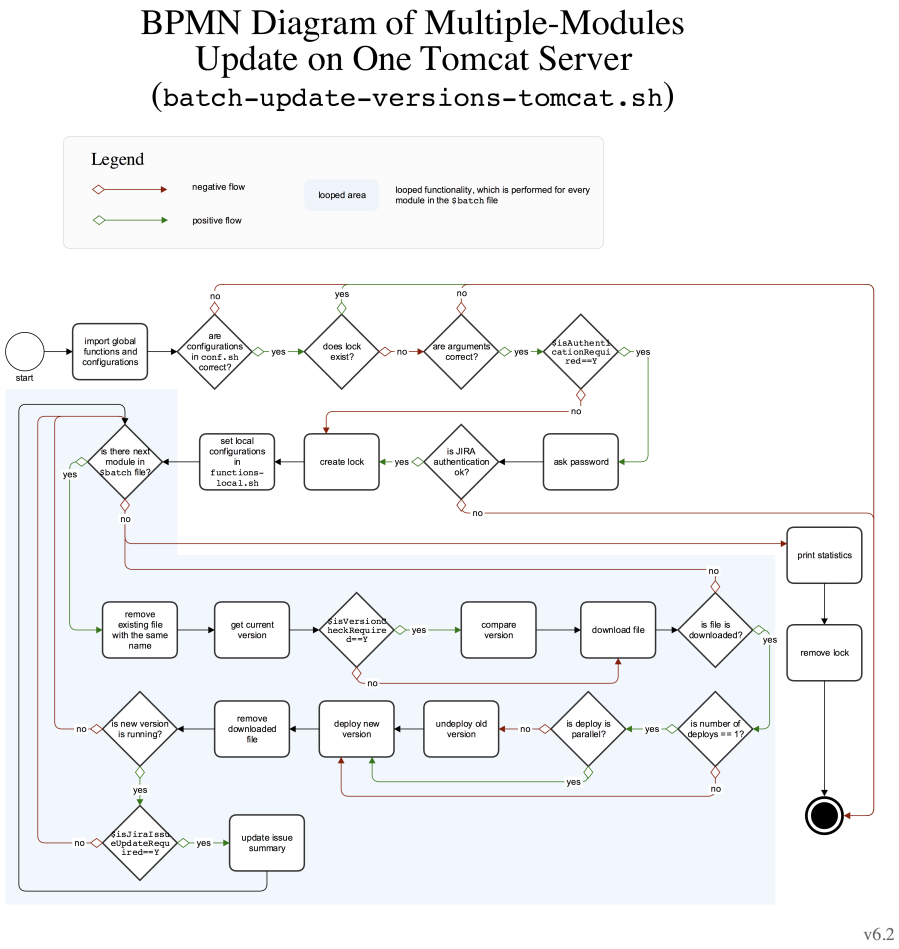
```
**************************************************
********************STATISTICS********************
                DOWNLOAD ERRORS: 1
                        system-0.0.0.0
                PRECOMPILE ERRORS: 0
                UNDEPLOY WARNINGS: 0
                DEPLOY ERRORS: 0
                RUN ERRORS: 0
                JIRA ERRORS: 0
                VERSION WARNINGS: 1
                        system: old cycle 3.13.1.7 is newer than new cycle 0.0.0.0


                DEPLOYED MODULES: 2
                        admin-3.13.1.12
                        authorization-2.12.1.41
**************************************************



        Removing lock file...
        OK: lock file UPDATING_BATCH_MODULES_irina.loc is removed
```

Joonis 2. Kolme mooduli uuendamise statistika.

Joonisel 2 on esitatud statistika, mida kuvatakse mitme mooduli uuendamisel või ühe mooduli uuendamisel mitmes serveris. See annab võimaluse aru saada uuenduse staatusest lugemata tervet teate. Antud juhul on toodud kolme mooduli – `system`, `admin` ja `authorization` – uuendamise statistika, mille kohaselt on kaks moodulit edukalt uuendatud ja `system` mooduliga tekkis kaks probleemi: uue versiooni number `0.0.0.0` on väiksem vana versiooni numbrist `3.13.1.7` ja skriptil ei õnnestunud alla laadida uue versiooni faili nimega `system-0.0.0.0`.

[9] `http://www.bpmn.org`

Joonisel 3 on näidatud protsesse, mis toimuvad mitme mooduli uuendamisel Tomcat 8 veebiserveris (faili `batch-update-versions-tomcat.sh` käivitamisel). Sinise taustaga alal on protsessid, mis korduvad iga mooduli juures. Joonis on originaalsuuruses Lisas III.



Joonis 3. Skripti protsessimudel mitme mooduli uuendamisel Tomcat 8 veebiserveris.

Olulised funktsioonid, mis on olemas nii mitme kuid ka ühe mooduli uuendamisel, on lukustamine, logimine, teate andmine, kasutaja teavitamine ja ühiste failide sünkroniseerimine.

Lukustamine vastab 8. nõudele Tabelis 2. Töö alguses kontrollib skript, kas serveris eksisteerib lukufail; kui mitte, siis skript tekitab selle ja jätkab tööd. Lukustamise loogika võimaldab vältida paralleelseid uuendusi samas veebserveris (lukufaili nimetus defineeritakse globaalse muutuja `$lock` abil).

Logimine vastab 6. nõudele – skript salvestab logid enda tegevustest (globaalne muutuja – `$log`) ning nendest logidest saab teada, kes ja millal mingit moodulit uuendas.

Skript väljastab iga tehtud sammu kohta värvilise teate (Joonis 6). See on seotud nõuetega 1 ja 7: värviline teate aitab kiiremini aru saada uuenduse staatusest (kas see on õnnestunud; kui mitte, siis milline samm ebaõnnestus). See säästab uuendamisprotsessis aega, sest kasutajatel ei ole vaja uurida serveri logifaile, et aru saada, mis sammul protsess katkes. Lisaks kuulub esimese nõude alla ka terminali teavitus (mis teostatakse `$bell` vahendiga), millega antakse teada, kui skript ootab kasutajalt mingit sisendit (nt JIRA parooli) või kui töö on lõppenud. See tähendab, et kasutaja ei pea uuenduse staatusest aru saamiseks jälgima skripti tööd.

Failide sünkroniseerimine on seotud 10 nõudega. Kuna skriptifailid peavad asuma igas serveris, siis on neid vaja parandada ja täiendada mitu korda. Selleks, et vältida töö dubleerimist, asuvad ühised failid (mis vastutavad loogika eest ja ei ole serveri seadistusfailid) repositooriumis. Igas serveris on seadistatud Crontab [7] tööd, mis kontrollivad regulaarselt, kas repositooriumis on olemas faili uus versioon. Kui see on olemas, siis see laaditakse alla. See tähendab, et skriptiparandusi ja täiendusi on vaja teha ainult üks kord repositooriumis. Joonisel 4 on toodud näide Crontab tööde kirjeldamisest.

```
### version-updater
*/1 * * * * cd ~/version-updater/; wget -N --no-cache http://path_to_repo/readme.txt
*/1 * * * * cd ~/version-updater/; wget -N --no-cache http://path_to_repo/extended-modules.txt
*/1 * * * * cd ~/version-updater/; wget -N --no-cache http://path_to_repo/functions.sh
*/1 * * * * cd ~/version-updater/; wget -N --no-cache http://path_to_repo/functions-tomcat.sh
*/1 * * * * cd ~; wget -N --no-cache http://path_to_repo/update-version-tomcat.sh
*/1 * * * * cd ~; wget -N --no-cache http://path_to_repo/batch-update-versions-tomcat.sh
```

Joonis 4. Näide Crontab tööde kirjeldusest ühisfailide automaatseks allalaadimiseks repositooriumist.

Tabelis 4 on esitatud Bash skripti vastavus kõigile nõudele Tabelist 2.

Tabel 4. Bash uuendamisskripti vastavus nõutele Tabelist 2.

Nõue	**Vastavus**
Kiirus.	Uuendamisprotsess võtab varasemast vähem aega – selle kohta on toodud statistika peatükis 4 "Uue lahenduse mõju projektile".
Mugavus.	Uuendamisprotsess nõuab ühe skripti käivitamist. Lisaks on realiseeritud värviline teate ja statistika.
Ühe ja mitme mooduli uuendamine.	On olemas eraldi skriptid ühe ja mitme mooduli uuendamiseks.
Töötamine eriveebiserveritega.	Lahendus töötab veebiserveritega WebLogic 12 ja Tomcat 8 (vajadusel seda võib kohandada Tomcat 6 ja 7 jaoks).
Teavitussüsteem.	Skriptil on kasutusel JIRA tööde uuendamine, mis saadab teavitusi huvitatud inimestele.
Logimissüsteem.	On olemas logifail.
Mittetehniline lahendus.	Kasutaja ei pea aru saama tehnilisi detaile, et käivitada skripti sisendparameetritega.
Lukustamissüsteem.	On olemas lukustamissüsteem.
Töötamine mitme serveriga.	On olemas funktsioon mitme serveri uuendamiseks.
Hallatav lahendus.	Skripti lähtekood on kättesaadav kõigile meeskonnaliikmele ja võib olla parandatud ja täiendatud igal hetkel.
Kiire arendusprotsess.	Skripti arendus, parandus ja paigaldus kõigile keskkonnale võttis umbes 150 tundi aega.
Töötamine moodulite erisufiksitega.	On olemas funktsioon erisufiksitega töötamiseks.
Odav.	Ühe inimtöö tunnihind projekti jaoks on umbes €45, mis tähendab, et skripti arendus maksis projektile $150 * 45 =$ €6750. Lisaks skripti kirjutati osade kaupa: alguses tehti põhisammu 1-5 ja 8 Tabelist 1, hiljem sammu 6-7 ja kõik nõuded Tabelist 2, lõpus täiendavaid funktsioone ja kontrolle, mis lihtsustavad skripti kasutamist ja vähendavad vea tekkimise tõenäosust.

Skript koosneb 16 failist, mida võib jagada kolme gruppi. Mõned failid on spetsiifilised kas Tomcat või WebLogic veebiserveri jaoks. Mõnede failide nimetusi võib muuta globaalsete muutujate abil. Igas failis on olemas kommentaarid, mis selgitavad faili ja muutujate eesmärke.

Ühised failid – failid, mis on ühised kõigile serveritele (mõned neist sõltuvad veebiserverist) ja mida saab sünkroniseerida.

- `version-updater/functions.sh` – globaalsed üldised funktsioonid, mis ei sõltu veebiserverist;
- `version-updater/functions-tomcat.sh` – globaalsed Tomcat 8 funktsioonid, mida kasutatakse ainult Tomcat veebiserveris;
- `version-updater/functions-wl.sh` – globaalsed WebLogic 12 funktsioonid, mida kasutatakse ainult WebLogic veebiserveris;
- `deploy-undeploy-script.py` – Python[10] skriptid WebLogic veebiserveris moodulite uuendamiseks;
- `update-version-tomcat.sh` – ühe mooduli uuenduse skript, mis sisaldab funktsioonide kasutamise õiget järjekorda Tomcat veebiserveris;
- `batch-update-versions-tomcat.sh` – mitme mooduli uuendamise skript Tomcat veebiserveri jaoks;
- `update-version-wl.sh` – ühe mooduli uuendamise skript WebLogic veebiserveri jaoks;
- `batch-update-versions-wl.sh` – mitme mooduli uuendamise skript WebLogic veebiserveri jaoks;
- `version-update/extended-modules.txt` – nimekiri moodulitest, millel on olemas erinevad sufiksid (nõue 12 Tabelist 2);
- `version-updater/readme.txt` – üldine informatsioon skripti autori ja juhendi kohta.

Kohalikud konfiguratsioonifailid – failid, mis sõltuvad serverist, kuhu skript on paigaldatud.

- `version-updater/conf.sh` – konfiguratsioonifail, kus asuvad globaalsed muutujad, mille väärtused sõltuvad serverist, kuhu skript on paigaldatud;
- `version-updater/functions-local.sh` – kohalikud funktsioonid, mis sõltu--vad serverist;

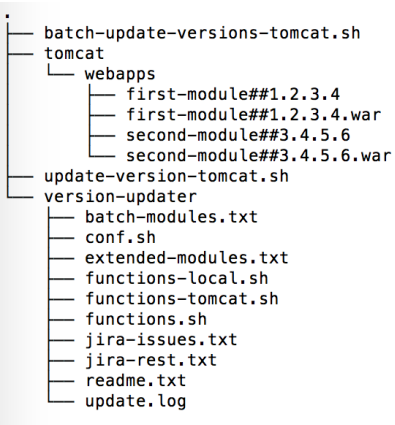
[10] `https://www.python.org`

- `version-updater/batch-modules.txt` – fail, kus kasutaja defineerib moodulite nimetusi ja versioone mitme mooduli uuendamiseks (faili nimetus defineeritakse globaalse muutujaga `$batch`);
- `version-updater/jira-issues.txt` – fail, kus on kaardistatud moodulite nimetused ja JIRA tööde koodid (on vajalik ainult JIRA kasutamisel ja on defineeritav globaalse muutujaga `$issues`).

Kohalikud väljundfailid – failid, mida skript täidab töö ajal ja mida kasutaja ei tohi muuta.

- `version-updater/update.log` – fail, kus logitakse skripti tegevusi (faili nimetus defineeritakse globaalse muutujaga `$log`);
- `version-updater/jira-rest.txt` – tühifail, mida skript kasutab REST [8] päringute genereerimiseks JIRA töö uuendamisel [9] (on vajalik ainult JIRA kasutamisel ja on defineeritav globaalse muutujaga `$rest`).

Joonisel 5 on toodud näide skriptifailide struktuurist serveris, kus on kasutusel Tomcat 8. Serveri juurkaustas asub alamkaust `/tomcat` veebiserveriga, kuhu on paigaldatud moodulid `first-module` versiooniga `1.2.3.4` ja `second-module` versiooniga `3.4.5.6`. Skripti uuendamisfailid asuvad rakenduse serveri juurkaustas: `batch-update-versions-tomcat.sh` ja `update-version-tomcat.sh`. Skripti abifailid asuvad kaustas `/version-updater`.



Joonis 5. Skripti failide struktuur veebiserveris Tomcat 8.

Failid lõppkasutajale on `update-version-*.sh` ja `batch-update-versions-*.sh` (lisaks ka logifail `update.log` toimunud uuendustest info saamiseks). Kõik ülejäänud failid on mõeldud skripti töö toetamiseks ja seadistamiseks.

3.2 Skripti paigaldamine

Koodirepositooriumis on olemas kõik skriptifailid, mille saab vajadusel alla laadida serverisse, kohalikku või virtuaalmasinasse. Samuti on olemas zip-fail, kus on arhiveeritud kõik failid vastavalt skripti versioonile ja veebiserverile (Tomcat ja WebLogic jaoks on eraldi arhiivid). Skripti paigaldamise protsessi on kirjeldatud juhendis ja see koosneb neljast sammust:

- Vastava zip-faili allalaadimine serverisse, kohalikku või virutaalmasinasse.
- Allalaaditud faili lahtipakkimine.
- Järgmiste failide täitmine serveri-spetsiifiliste andmetega:
  - `version-updater/conf.sh`
  - `version-updater/functions-local.sh`
  - `version-updater/jira-issues.txt`
- Crontab tööde seadistamine ühisfailide sünkroniseerimiseks.

Lisaks, Bash skripti kasutamiseks kasutajal peavad olema paigaldatud järgmised vahendid:

- Bash käsureaprotsessor;
- veebiserver WebLogic 12 või Tomcat 8 (vajadusel saab skripti kohandada Tomcat 6 ja 7 versioonidele);
- standardsed UNIX vahendid: `find` [11], `grep` [12], `wget` [13] (kui mooduli failid asuvad veebis), `curl` [14];
- standardne käsurea `bell` funktsioon.

[11] `http://www.gnu.org/software/findutils/manual/html_mono/find.html`
[12] `https://www.gnu.org/software/grep`
[13] `https://www.gnu.org/software/wget`
[14] `https://curl.haxx.se`

3.3 Skripti kasutamine

Skripti käivitamiseks kasutatakse kahte faili: üht ühe mooduli uuendamiseks `update-version-*.sh` ja teist mitme mooduli uuendamiseks `batch-update-versions-*.sh` (`*` asemel on `tomcat` või `wl`). Kõiki ülejäänud faile kasutab skript enda töös ja kasutaja ei pea neid käivitama.

Ühe mooduli uuendamise skriptile on võimalik anda neli sisendparameetrit:

```
./update-version-*.sh MODULE_NAME MODULE_VERSION JIRA_USERNAME [p]
```

Näide:

```
./update-version-tomcat.sh admin 1.1.1.1 irina
```

- `MODULE_NAME` – kohustuslik parameeter, mis määrab uendatava mooduli nimetuse (näiteks Tomcat-i puhul on see sama nimetus, mis tee (*path*) Tomcat Manager-is [15]).
- `MODULE_VERSION` – kohustuslik parameeter, mis määrab paigaldatava versiooni numbri.
- `JIRA_USERNAME` – parameetrit tuleb kasutada siis, kui JIRA autentimine on kohustuslik (globaalne muutuja `$isAuthenticationRequired="Y"`).
- `p` – mittekohustuslik parameeter, mille sisestamisel teostatakse paralleelne uuendus [16] (ei toimu vana versiooni eemaldamist).

Mitme mooduli uuendamise skriptile on võimalik anda kaks sisendparameetrit:

```
./batch-update-versions-*.sh JIRA_USERNAME [p]
```

Näide:

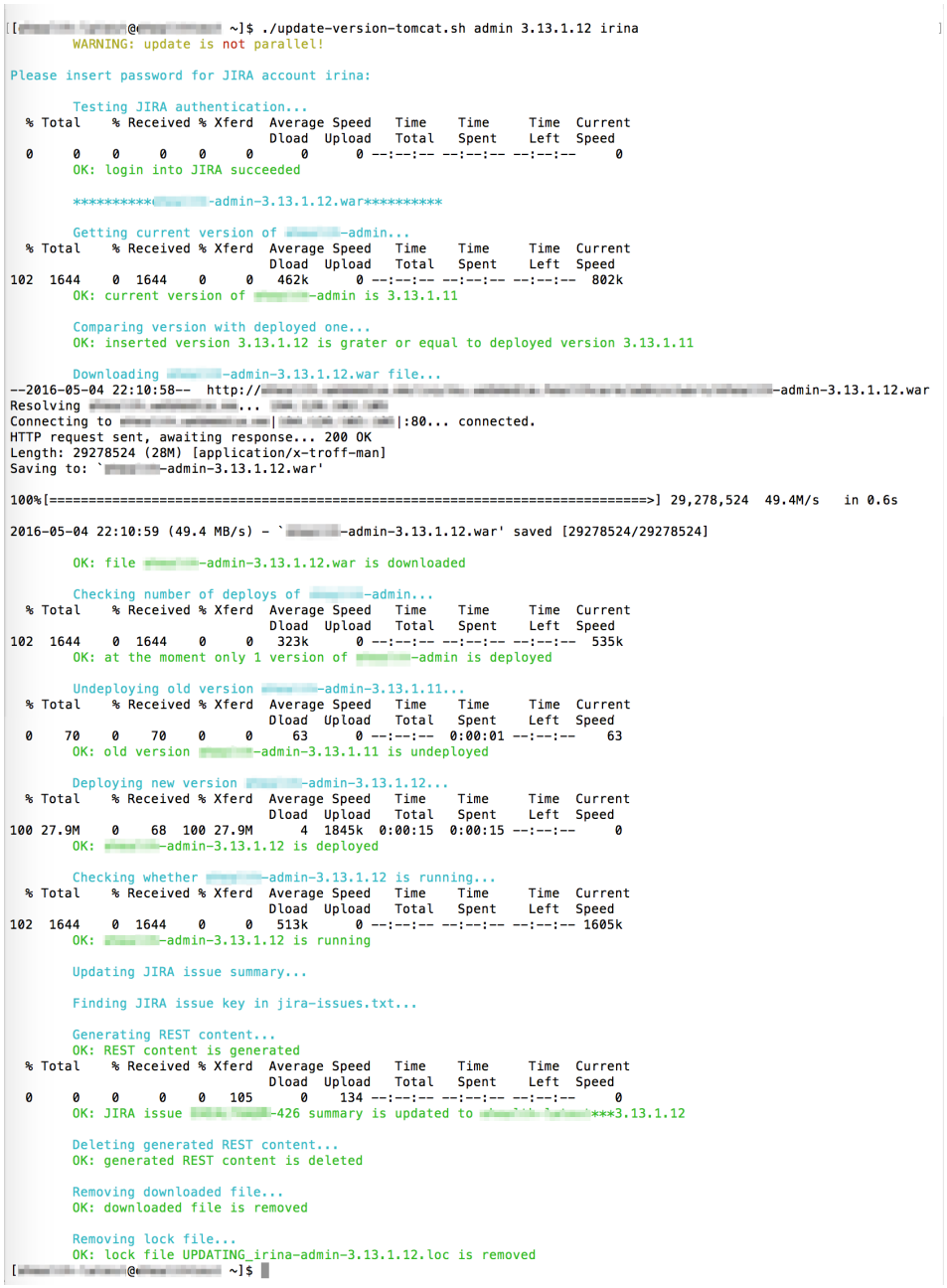
```
./batch-update-versions-tomcat.sh irina p
```

Sisendparameetrid `JIRA_USERNAME` ja `p` on samad, mis ühe mooduli uuendamise korral.

Joonisel 6 on näidatud ühe mooduli edukas uuendus. Skript annab teate kõigist sooritatud sammudest.

[15] `https://tomcat.apache.org/tomcat-7.0-doc/manager-howto.html`

[16] `https://tomcat.apache.org/tomcat-8.0-doc/config/context.html#Parallel_deployment`



Joonis 6. Ühe mooduli edukas uuendamine skripti abil.

3.4 Dokumentatsioon

Skripti kohta on olemas dokumentatsioon, mis on uuendamise oluline osa. Skripti juhend kirjeldab, kuidas skripti on võimalik kasutada, paigaldada ja seadistada. Juhend on kättesaadav kõigile meeskonnaliikmetele, mis tähendab, et igaühel on võimalus seda kasutada. Nii skripti projekti juhend kui ka avalik juhend on ära toodud töö Lisades IV ja V.

4 Uue lahenduse mõju projektile

Automaatne versiooniuuendus asendas täielikult vana käsitsiprotsessi. Sellel on olemas nii oma eelised kui ka puudused.

Kõige olulisem eelis on see, et skript täidab kõik Tabelis 2 esitatud nõuded, neist enim väärivad märkimist nõuded 1 ja 2 – protsessi kiirus ja mugavus.

Töö käigus mõõdeti vana ja uue uuendamisprotsessi kiirust. Automaatse protsessi alguseks loeti skripti käivitamise käsku ja lõpuks skripti teavitust töö lõpetami--sest. Skripti tööaja mõõtmiseks kasutati UNIX käsurea funktsiooni `time` [17]. Käsitsi uuendamisprotsess koosneb sammudest, mida kirjeldati Tabelis 1 ja sellele kulunud aega mõõdeti stopperiga.

Valiti neli protsessi, mis teostati samas serveris ja samade war-failidega: kolme kõige populaarsema mooduli eraldi uuendamine ja kõigi moodulite uuendamine (valitud rakenduses oli neid 18). Iga protsess sooritati käsitsi ja automaatselt kolmel korral, võrdlemiseks kasutati katsete keskmisi väärtusi. Tulemused on toodud Tabelis 5.

Tabel 5. Vana ja uue meetodi tööaeg.

Moodul	**Automatiseeritud uuendus**	**Käsitsi uuendus**	**Uuendusi 2015-s**	**Säästetud aeg 2015-s**
admin	27s	80s	67 tk	1t
treatment	61s	112s	240 tk	3t 44m
reception	38s	100s	197 tk	3t 39m
kõik (18 tk)	552s	1048s	–	–

Kaks viimast veergu näitavad säästetud aega 2015. aastal (terve aasta kasutati uuendamiseks automaatset skripti). Veerus "Uuendusi 2015-s" on toodud vastava mooduli uuenduste arv ühes testrakenduses aasta jooksul. Veerus "Säästetud aeg 2015-s" on ära näidatud 2015. aastal kokkuhoitud aeg, mis on saadud käsitsi ja automaatse uuendamise aja erinevuse korrutamisel uuenduste arvuga.

Tegelikkuses on kokkuhoitud aeg veelgi suurem, sest vana protsess nõudis tähelepanu kogu protsessi vältel. Uut skripti kasutades saavad kasutajad tegeleda paralleelselt teiste ülesannetega ega pea jälgima skripti tööd (skript annab ise teada kui uuendamine on lõpule viidud).

Peamiseks puuduseks on, et kasutajad ei tunne enam versiooniuuenduse protsessi tausta ja vigade tekkides (kas tingituna serverist, veebiserverist või skriptist) ei oska neid parandada. See on üldine automatiseerimise probleem, millega tuleb mugavuse nimel leppida.

[17] `http://pubs.opengroup.org/onlinepubs/9699919799/utilities/time.html`

Käesoleva töö raames küsiti tagasisidet automatiseeritud uuendamisprotsessi kohta seda kasutanud meeskonnaliikmetelt. Allpool on toodud mõned väljavõtted, terviktekstid on kättesaadavad Lisas VI.

> "I also like that scalability has been considered when writing the script + many optional features have been implemented which can be used if one desires to do so." (Martin Rakver, Hosting Services and Application Manager)

> "I personally think that version-update script has significantly improved the speed of version update." (Kalle Jagula, QA Specialist)

> "Version updater script has standardized the way our environments are updated, reducing learning curve for new people performing the task. Downside is that people are unaware, what happens in the background and can't handle simple errors in the process anymore." (Tanel Käär, System Architect)

> "The scripts allowed me to conveniently perform the updates while letting me concentrate on solving the primary problems and not leaving the environment containing all necessary information." (Klaus-Eduard Runnel, Senior Programmer)

> "Samuti tooks välja, et uuendamise skriptid on väga hästi dokumenteeritud ja põhjalikult kirja pandud kuidas toimivad." (Helina Ziugand, QA Specialist)

5 Lahenduse edasiarendamine

Lähitulevikus planeeritakse lahendada skripti põhiprobleemi, mis seisneb selles, et kasutajad ei tunne versiooniuuendamise protsessi ja ei oska parandada uuendamise ajal tekitatavaid vigu. Selleks skriptile lisatakse funktsioon, mis loeb serveri ja veebiserveri logid, leiab sealt tekitatud vea põhjust, seletab seda inimkeeles ja pakub kasutajale vastavat lahendust.

Samuti planeeritakse realiseerida kolleegi soovitust samaaegsest mitme rakenduse uuendamisest, selleks et ühe korraga uuendada samad moodulid ja samad versioonid mitmel rakendusel.

Kaugtulevikus planeeritakse tsentraliseerida skripti, et iga rakenduse uuendamine toimuks ühest kohast ja uuendamiseks ei oleks vaja siseneda igasse serverisse eraldi.

Peale projekti plaane oodatakse skripti parandust ja täiendust GitHub kogukonnast, kuna Bash skripti kood on avalik (ainult Apache Tomcat veebiserverile) ja asub GitHub repositooriumis (Lisa VII). See tähendab, et igaüks, kellel on sarnane tarkvaraarenduse süsteem, saab seda kasutada ja täiendada vastavalt oma nõuetele. Igas failis on kommentaarid, mis selgitavad faili eesmärke ja kasutamisviisi. Samuti on GitHub platvormil olemas võimalus vigadest teatamiseks ja uute funktsioonide tellimiseks.

Skripti rakendamise lihtsustamiseks on olemas avalik dokumentatsioon, mis asub GitBook repositooriumis. See sisaldab kasutusjuhendit, paigaldamise ja seadistamise juhendit ning kirjeldab, kuidas on võimalik skripti tööd UNIX standardsete vahenditega laiendada.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli automatiseerida versiooniuuenduse protsessi konkreetses projektis, et säästa aega ja vähendada inimlike vigade arvu.

Bakalaureusetöö teoreetilise osa käigus sõnastati probleemi lahendusele 13 nõuet ja uuriti nelja erinevat tööriista versiooniuuenduste haldamiseks. Mitte ükski neist ei vastanud kõigile nõuetele. Lahenduseks valiti Bash skripti kirjutamine.

Bakalaureusetöö praktilise osa käigus kirjutati versioonide uuendamiseks Bash skript, mis vastab kõigile nõuetele. Samuti koostati juhend selle skripti paigaldamiseks, seadistamiseks ja kasutamiseks.

Kokkuvõtteks võib öelda, et bakalaureusetöös püstitatud eesmärgid on täidetud. Kirjutatud Bash skript aitab kiiremini ja mugavamalt projektis versioone uuendada. Seda kinnitavad nii meeskonnaliikmete tagasiside kui ka käsitsi ja automatiseeritud protsessi kiiruse võrdlemine.

Loodud lahendust saavad kasutada ja täiendada ka teised inimesed, sest selle kood ja dokumentatsioon on avalikult kättesaadavad.

Kasutatud materjalid

[1] Atlassian Bamboo koduleht. [Veebis, 5.05.2016].
`https://www.atlassian.com/software/bamboo`

[2] Chef Delivery koduleht. [Veebis, 5.05.2016].
`https://www.chef.io`

[3] Jenkins koduleht. [Veebis, 5.05.2016].
`https://jenkins.io`

[4] Ansible Tower koduleht. [Veebis, 5.05.2016].
`https://www.ansible.com`

[5] Bash-keele kasutusjuhend. [Veebis, 5.05.2016].
`https://www.gnu.org/software/bash/manual/bashref.html`

[6] Stephen G. Kochan ja Patrick Wood, "Unix Shell Programming," *Sams Publishing*, 2003

[7] Crontab-vahendi kasutusjuhend. [Veebis, 5.05.2016].
`https://help.ubuntu.com/community/CronHowto`

[8] Todd Fredrich, "RESTful Service Best Practices," *Pearson eCollege*, mai 2012.
`http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf`

[9] JIRA REST API Reference. [Veebis, 5.05.2016].
`https://docs.atlassian.com/jira/REST/latest`

Lisad

I Terminid

Demorakendus – rakendus, mida kliendid kasutavad funktsioonide testimiseks.

Moodul – tarkvara osa, kuhu on kogutud ja pakitud ühe funktsiooni kood (üks war-fail on üks moodul).

Modulaarne süsteem – tarkvaraarenduse süsteem, mille korral kogu kood jagatakse moodulitesse, et ühe funktsiooni tarnimine ei nõuaks terve rakenduse uuendamist.

Rakendus (*application*) – tarkvara, mis on paigaldatud veebiserverisse ja on kättesaadav kasutamiseks.

Server – riist- ja tarkvarasüsteem, kuhu on paigaldatud veebiserver.

Testrakendus – rakendus, mida projekti meeskonnaliikmed kasutavad tarkvara testimiseks.

Toodangrakendus – lõppkasutajatele mõeldud rakendus.

Veebiserver – klient-server-mudelil ja protokollil HTTP või HTTPS põhinev tarkvara, mis võtab vastu kasutajate brauseritelt tulevaid päringuid ja saadab vastuseks HTML-lehti ja faile (nt Apache Tomcat või Oracle WebLogic) [18].

Versioon – number, mis määrab war-faili sisu (war-fail sisaldab funktsionaalsust ja parandusi vastavalt versiooni numbrile).

Versiooniuuendamine – protsess, mille korral veebiserverisse paigaldatakse mooduli uuemat versiooninumbrit, selleks, et rakendada selle parandusi või täiendusi.

[18] `http://akit.cyber.ee/term/3152-veebiserver`

II Ühe mooduli uuendamise mudel

Skripti protsesside mudel ühe mooduli uuendamisel Tomcat 8 veebiserveril asub avalikus GitHub repositooriumis:
`https://raw.githubusercontent.com/iriiiina/bachelors-thesis/master/thesis/diagrams/BPMN-diagram-one-module-tomcat.png`

III Mitme mooduli uuendamise mudel

Skripti protsesside mudel mitme mooduli uuendamisel Tomcat 8 veebiserveril asub avalikus GitHub repositooriumis:
`https://iriiiina.gitbooks.io/version-updater-manual/content/BPMN-diagram-multiple-module-update-tomcat.png`

IV Skripti projekti juhend

Bash skripti projektisisene juhend:
`https://github.com/iriiiina/bachelors-thesis/blob/master/manual/Confluence%20Manual.pdf`

V Skripti avalik juhend

Bash skripti avalik juhend asub avalikus GitBook repositoorimus:
`https://iriiiina.gitbooks.io/version-updater-manual/content/`

VI Kolleegide tagasiside

Martin Rakver, Hosting Services and Application Manager

> I have not been using the script very much, because I rarely update modules nowadays. But what I can say is that it is definately in the right place – automating activities that testers would need do to on daily basis each time when deploying a new module. Come to think about it, there are actually quite many activites related to new module deployment – would be interesting to compare the time it takes to deploy a module manually vs using script + calculate about how much time we as a team are saving daily, montly, yearly (and get to do more important things with that time). I also like that scalability has been considered when writing the script + many optional features have been implemented which can be used if one desires to do so.

Kalle Jagula, QA Specialist

I personally think that version-update script has significantly improved the speed of version update. Especially with the settings for bulk-update, which improves the speed of modules' version update and manageability of test/demo environments.

Tanel Käär, System Architect

Version updater script has standardized the way our environments are updated, reducing learning curve for new people performing the task. Downside is that people are unaware, what happens in the background and can't handle simple errors in the process anymore.

Klaus-Eduard Runnel, Senior Programmer

I have mostly used the scripts to perform quick module and dependency updates in specific test environments when deploying quick fixes or debugging problems in the modules I'm responsible for. Deployment of artifacts is secondary task in such scenarios. The scripts allowed me to conveniently perform the updates while letting me concentrate on solving the primary problems and not leaving the environment containing all necessary information.

In some cases it was necessary to deploy custom-built experimental (unversioned) war-files. In these cases the scripts could not be used. They would have been useful though if such experiments would have been commited to feature branches and published as feature branch artifacts. (As of today, we are better prepared for such circumstances.)

The scripts were also useful to deploy predefined sets of modules to temporary test environments living on reusable virtual machine images. It is important to note that a module update triggers launching a set of associated sql scripts to bring a database schema up to date. In that way the version update scripts let us automate most of a process of setting up updated environment from virtual machine images.

It would be helpful if the scripts were adapted to perform changes on remote machines and to perform changes on multiple environments simultaneously.

Helina Ziugand, QA Specialist

Uuendamise skriptid on nii projektile kui ka minu igapäevasele tööle väga positiivselt mõjunud. Kui algselt oli uuendamise jaoks vaja mitu sammu käsitsi teha, siis nüüd käib see automaatselt ja kiiresti. Võiks öelda, et üsna tüütu oli alguses kogu see uuendamise protsess - pidevalt pidi jälgima, kas vaja käsureale kirjutada järgmist sammu, hiljem JIRAs vana versiooni numbrit muuta ja Weblogicust vana versioon kustutada. Praegu on väga mugav ühe käsklusega uuendamise skript tööle panna, mis korraga kõik vajalikud sammud ära teeb ja samal ajal kui skript jookseb ise teiste tegevustega jätkata. Väga selgelt erinevate värvidega on välja tootud error, warningu ja success teated, mis on lihtsasti märgatavad ja hästi loetavad.

Kui vaja on rohkem kui ühte moodulit uuendada, siis väga hea lahendus on selleks mitme moodulise uuendamise skript. Kui varem oli vaja rakenduse uue tsükli peale uuendada, siis võttis see ebamugavalt kaua aega ja vabatahtlikult kõige parema meelega seda ette ei tahtnud võtta. Nüüd on vaja ainult uuendatavad versioonid tekstifaili kirja panna ja ühe käsklusega skript käima tõmmata.

Samuti tooks välja, et uuendamise skriptid on väga hästi dokumenteeritud ja põhjalikult kirja pandud kuidas toimivad. Mul on väga hea meel, et versioonide uuendamine on skriptide abil nii lihtsaks, mugavaks ja meeldivaks tegevuseks saanud.'

VII Skripti lähtekood

Bash skripti kood asub avalikus GitHub repositooriumis:
`https://github.com/iriiiina/version-updater`

VIII Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Irina Ivanova**, (sünnikuupäev: 9.05.1988)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Versiooniuuenduse automatiseerimine kasutades skriptimiskeelt Bash**, mille juhendajad on Helle Hein ja Polina Morozova,
   (a) reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   (b) üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace-i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, 1.05.2016