TARTU ÜLIKOOL
Arvutiteaduse instituut
Infotehnoloogia eriala

Irina Ivanova

Versiooniuuenduse raamistik kasutades skriptimiskeelt Bash

Bakalaureusetöö (6 EAP)

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

Tartu 2016

Versiooniuuenduse raamistik kasutades skriptimiskeelt Bash

Lühikokkuvõte:
Projektis, kus on kasutusel modulaarne süsteem, toimub palju rakenduse uuendusi, eriti kui seda rakendust kasutavad mitu klienti. Manuaalne versioonivahetuse protsess võtab palju aega ja suurendab inimliku vea tekkimise tõenäosust. Bakalaureusetöö teoreetilise osa eesmärgiks on leida võimalikud lahendused uuenduse protsessi automatiseerimiseks ning valida neist parim. Praktilise osa eesmärgiks on valmistada valitud lahendust ja rakendada seda projektis.

Võtmesõnad:
Bash skript, versiooni uuendus, Apache Tomcat, Java, modulaarne süsteem

Version Update Framework Using Scripting Language Bash

Abstract:
There are a lot of application updates in the project, where modular system is in use, esecially if many clients are using this application. Manual version update process takes a lot of time and increases the possibility of human errors. The aim of the bachelor's thesis theoretical part is to find out the 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 to apply it in the project.

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

Sisukord

Sissejuhatus

Käesolevas lõputöös lahendatakse rakenduslikku probleemi toimivas tarkvaraprojektis. Probleem seisneb selles, et projekti olemuse tõttu rakenduse versiooni uuendamise protsess võtab kaua aega ja eeldab palju manuaalset sekkumist. See raiskab ressursse ja suurendab inimliku vea tekkimise tõenäosust. Bakalaureusetöö teoreetilise osa eesmärgiks on leida võimalikud lahendused uuenduse protsessi automatiseerimiseks ning valida neist parim. Praktilise osa eesmärgiks on valmistada valitud lahendust ja rakendada seda projektis.

Töö koosneb viiest osast. Esimeses osas püstitatakse probleem: kirjeldatakse projekti arhitektuuri, manuaalse uuendamise protsessi, probleeme, mis vana protsess toob ja nõudeid uuele lahendusele. Peamine ülesanne seisneb selles, et rakendus koosneb paarikümnest moodulitest (täpne arv sõltub kliendi nõuetest), mida on vaja paigaldada 31 keskkonda. Selleks, et paigaldada rakenduse uut versiooni kõikidesse veebiserveritesse, on vaja teha umbes 600 uuendust (mitte arvestades test rakenduste vaheuuendustega). Lisaks on projektis kasutusel kaks erinevat veebiserverit: Oracle WebLogic 12[1] ja Apache Tomcat 8[2]. Uuenduse protsessi automatiseerimine peab võtma arvesse kõik need nõuded.

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 maksmus, võimetus hallata protsesse ja vajadusel parandada vigu, vajadus kulutada palju aega toode tundmiseks ja rakendamiseks.

Kolmandas osas kirjeldatakse valitud lahendust: mis on skripti tööpõhimõte ja arhitektuur ja kuidas seda saab paigaldada ja kasutada. Skripti põhifunktsionaalsus on mooduli faili alla laadimine, vana versiooni maha võtmine, uue versiooni paigaldamine ning kolleegide teavitamine toimunud versiooniuuendusest. Lisaks on oluliseks funktsionaalsuseks erinevad kontrol-

[1] `https://www.oracle.com/middleware/weblogic`
[2] `http://tomcat.apache.org`

lid, mis vähendavad tõenäosust paigaldada vale versioon ja logimise funktsionaalsus, mis võimaldab kätte saada põhjalikku infot iga uuenduse kohta.

Neljandas osas uuritakse, kuidas antud lahendus mõjus projektile: võrreldakse versiooni uuendusele kulunud aega manuaalse ja automatiseeritud protsessi puhul ja lisatakse kolleegide tagasisidet uue raamistiku kasutamisele.

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

Töös on olemas kuus lisa: terminid, Bash skripti lähtekood, skripti avalik juhend, skripti projekti juhend, kolleegide tagasiside uue raamistiku kohta ja litsents.

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

1 Probleemi püstitus

Versioonide uuendamine on sageli esinev vajalik protsess tarkvara arenduses. Projektis, mida kirjeldatakse käesoleva töö raames, etendab see protsess väga olulist rolli.

Projekti rakendus on kirjutatud Javas[3] ja kasutusel on WAR-failid. Lisaks rakendab projekt modulaarsuse 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 erinevatel klientidel on erinev moodulite komplekt).

Rakendust kasutavad 10 klienti, neist kolmel on olemas kolm rakendust: toode, demo (kus kliendid tutvuvad uut funktsionaalsust ja parandusi enne toote uuendust) ja test (kus projekti meeskond testib funktsionaalsust enne tarnimist kliendile). Seitsmel kliendil on olemas kaks rakendust: toode ja test. See tähendab, et kokku on olemas $3 * 3 + 7 * 2 = 23$ kliendirakendust. Lisaks on veel umbes 7 projekti testrakendust, mida kasutatakse erinevatel faasidel erineva funktsionaalsuse testimise jaoks. Seega kokku tuleb $23 + 7 = 30$ rakendust, mida on vaja regulaarselt uuendada (mitte arvestades kohalik- ja virtuaalmasinatega, mida iga meeskonnaliige võib vajadusel luua). See tähendab, et rakenduse uue versiooni paigaldamiseks igale veebiserverile on vaja teha $30 * 20 = 600$ uuendust.

Toodud numbrid näitavad, et versiooni manuaalne uuendus võtab palju aega ja tähelepanu, seetõttu otsustati lihtsustada ja automatiseerida uuenduse protsessi.

[3] `https://www.oracle.com/java`

Manuaalne uuenduse protsess

Ühe mooduli manuaaluuenduse protsessi sammud on esitatud Tabelis 1.

Tabel 1. Manuaalse uuenduse protsessi sammud.

Samm	**Tegevus**
1	WAR-faili uue koodiga alla laadimine serveri peale.
2	Kui veebiserveriks on Oracle WebLogic, siis WAR-faili prekompileerimine (Apache Tomcat veebiserveril seda sammu ei ole).
3	Faili ümber nimetamine.
4	Faili paigaldus.
5	Vana faili eemaldamine.
6	Uue versiooni staatuse kontrollimine.
7	JIRA[4] töö uuendamine, mis automaatselt saadab emaile kolleegidele, et versioon on uuendatud (iga mooduli ja keskkonna kohta on olemas eraldi JIRA töö).
8	Alla laaditud faili kustutamine.

Mitmete moodulite uuendamisel tuleb teha kõik need sammud iga mooduli jaoks. Aeg, mida võtavad need sammud, sõltub moodulist (kuna mõnes moodulis võib olla kaks korda rohkem koodi, kui teises, mis tähendab, et selle allalaadimine ja paigaldamine võtab kaks korda rohkem aega), aga keskmiselt see on 1.5 minutit. Kõikide moodulite uuendamine võib võtta tund aega, kuna lisandub veel aeg, mida inimene raisab õige versiooni numbri leidmiseks, WAR-faili aadressi kopeerimiseks, õige JIRA töö leidmiseks ja avamiseks jne. See tähendab, et inimlike vigade tekkimise tõenäosus on suur (nt vale versiooni valimine).

Tabelis 2 on toodud nõuded versiooni uuenduse lihtsustamise lahendusele, kus kirjeldatakse lahendust, millele on mõtet investeerida aega protsessi muutmiseks – muul juhul uue protsessi loomine võtab rohkem aega, kui toob kasu.

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

Tabel 2. Nõuded versiooni uuenduse lihtsustamise lahendusele.

Number	**Nõue**
1	Uuenduse protsess peab võtma vähem aega.
2	Uuenduse protsess peab nõudma nii vähe käsitsi tegevusi, kui on võimalik.
3	Lahendus peab võimaldama uuendada nii üht moodulit, kuid ka mitu moodulit korraga.
4	Projekt kasutab kaks erinevat veebiserverit: Apache Tomcat 8 ja Oracle WebLogic 12. Lahendus peab töötama sama moodi kõikidel serveritel, sõltumata sellest, mis tarkvara seal on paigaldatud.
5	Mõned meeskonnaliikmed soovivad saada teavitusi toimunud uuenduste kohta, mis tähendab, et lahendusel peab olema teavituste süsteem.
6	Peab olema võimalus vaadata kes ja millal uuendas mingit moodulit mingis keskkonnas, mis tähendab, et peab olema logimise süsteem.
7	Uuendustega tegelevad testijad, seega protsess ei tohi olla liiga tehniline.
8	Uuendustega võib tegeleda mitu inimest samal ajal, mis tähendab, et lahendusel peab olema lukustamise süsteem, et vältida paralleelselt uuendust samal serveril.
9	Toodang keskkonnad kasutavad 2 serverit, seega lahendus peab oskama uuendada versiooni automaatselt kahel serveril.
10	Lahendus peab olema täielikult projekti kontrolli all, selleks, et selle haldamist, seadistamist ja parandamist oleks võimalik teostada igal ajal.
11	Lahenduse loomine ei tohi võtta palju aega, sest projektil puuduvad selleks inimresurssid.
12	Lahendus ei tohi võtta palju raha.

2 Võimalikud lahendused

Uuenduse protsesside automatiseerimiseks ja haldamiseks on olemas palju valmistooteid. Kõige populaarsemad nendest on Atlassian Bamboo [1], Chef [2], Jenkins [3] ja Ansible Tower [4].

Kõikide valmistoodete rakendamisel alati on olemas oma eelised ja puudused.

Eelised

- Ei ole vaja kulutada aega süsteemi kirjutamisele, mis tegelikult on juba leiutatud.
- Kuna rakendus on juba turul ja on populaarne, siis on suur tõenäosus, et see on kvaliteetne ja vastab kasutajate ootustele.
- On olemas kogukond tarkvara kasutajatest, kes vajadusel saavad aidata ja jagada enda kogemust.

Puudused

- Suurem osa valmistoodetest on tasulised (Bamboo ja Jenkins on erandiks) ja hind alati sõltub serverite, rakenduste ja moodulite arvust, mis vaadatud projekti korral on suur.
- Valmistoode õppimine, paigaldamine ja seadistamine võtab palju aega, eriti kui see pakub laia valiku funktsionaalsusi.
- Projekt sõltub tootjast – vigade olemasolul ei saa olla kindel, et neid parandatakse lähimal ajal või parandatakse üldse; probleemide tekkimisel ei saa olla kindel, et tootja suudab pakkuda kasutajatuge.
- Projekt peab usaldama tootjat – usaldama, et valmistoode on turvaline ja stabiilne.

Need eelised ja puudused kehtivad kõikide valmistoodete kohta.

Alternatiiviks valmistoote kasutamisele on olemas võimalus kirjutada enda lahendus. Kuna see peab suhtlema teiste veebiteenustega ja töötama kõikidel

operatsioonsüsteemidel, siis valiti skriptimiskeel Bash [5, 6].

Tabelis 3 on toodud valmistoodete vastavus nõuetele, mis on esitatud Tabelis 1. Kõige kriitilisem on nõue 11 "lahenduse loomine ei tohi võtta palju aega, sest projektil puuduvad selleks inimresurssid".

Tabel 3. Võimalikke lahenduste vastavus nõuetele.

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

Atlassian Bamboo vajab erilist tähelepanu, kuna projektis on kasutusel teised Atlassian tooded, nagu JIRA, Confluence[5] ja Fisheye[6]. Esiteks, see tähendab, et Bamboo on tasuta antud projekti jaoks (tavaliselt see on tasuline), teiseks, – Bamboo integreerub väga hästi projektiga, mis võimaldab tekitada seoseid rakenduse koodi, muudatuste kirjelduse ja uuenduse vahel. Lisaks, antud toodet kasutavad firmas teised projektid ka, mis annab võimaluse kaasata inimesi, kellel on olemas teadmised ja kogemus Bamboo seadistamise kohta.

Teadmised aitavad rakendada seda võimalikult kiiresti, kuid kogemus näitas, et Bamboo lõplik rakendamine võttis teistes projektides palju aega: näiteks, ühes projektis ühe mooduli seadistamine võttis 1.5 kuud, mis 20 moodulite korral tähendaks $\sim 20 * 1.5 \approx 30$ kuud. Lisaks üks uuendus võib võtta 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 mitu rakendust, siis võib tekkida ootejärjekord.

Tabelist 3 on näha, et kõige sobilikum lahendus oleks enda Bash skripti kirjutamine, kuna see on ainuke, mis vastab kõikidele nõuetele ja, mis on

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

olulisem, vastab nõuele 11. Võrreldes Bamboo lahendusega sellel valikul on olemas üks puudus: seda ei saa integreerida teiste Atlassian toodetega. Kuid on olemas ka lisaeelised.

- Skripti on võimalik arendada osade kaupa: alguses teha lihtsam funktsionaalsus, mida hiljem järk-järgult täiendada uue ja keerulisema funktsionaalsusega. See võimaldab kasutada ressurse väikeste osadena ja samal ajal töötada parema süsteemiga.
- Lahendus on väga paindlik: on võimalik arendada just seda funktsionaalsust, mis on vajalik projekti jaoks ja lisada seadistamise võimalust, et sama lahendust saaks kasutada erinevate serverite ja rakenduste korral.
- Lahendus ei nõua projekti arhitektuuri ja protsesside muutmist: arendajate jaoks uue versiooni tekitamise protsess jääb samaks.

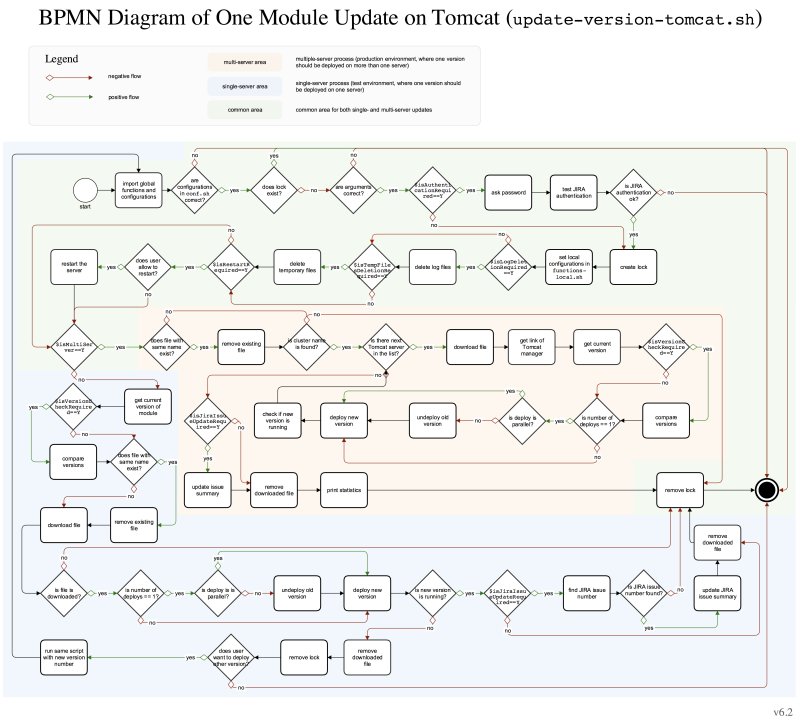
Arvestades kõikide lahenduste eeliste ja puudustega valiti Bash skripti kirjutamine.

3 Lahendus

3.1 Skripti arhitektuur ja tööpõhimõte

Skript automatiseerib kõik tegevused Tabelist 1. Seda käivitatakse serveril, kus on vaja uuendada kas üht või mitu moodulit ja parameetritena antakse ette mooduli nimetust, versiooni ja JIRA kasutajatunnust (mitmete moodulite uuendamisel mooduli nimetus ja versioon puuduvad).

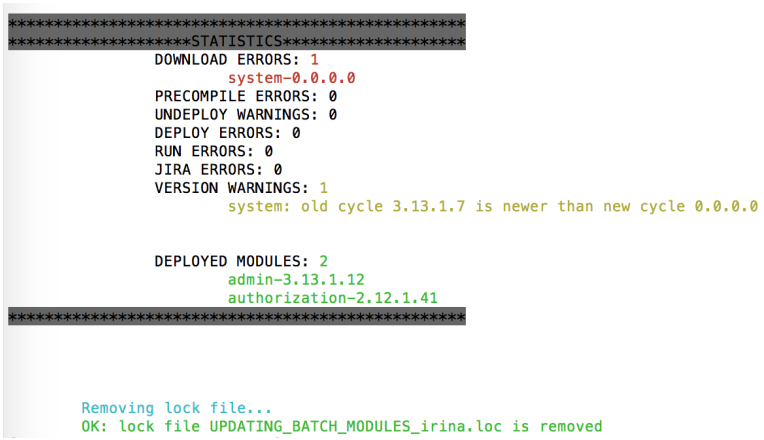
Joonisel 1 on näidatud protsessid, mis toimuvad ühe mooduli uuendamisel Tomcat 8 veebiserveril (faili `update-version-tomcat.sh` käivitamisel).



Joonis 1. Skripti protsesside mudel ühe mooduli uuendamisel Tomcat 8 veebiserveril [7].

[7] `https://raw.githubusercontent.com/iriiiina/bachelors-thesis/master/thesis/diagrams/BPMN-diagram-one-module-tomcat.png`

Joonisel 1 on toodud äriprotsesside mudel (*Business Process Modelling Notation*[8]), mis kirjeldab samme ja kontrolle, mida teostab skript. Positiivsel töövoogul on rohelised nooled ja negatiivsel – punased. Ühe mooduli uuenduse protsess sõltub sellest, mitmel serveril toimub uuendus: test rakendustel on üks server ja toodang rakendustel on kaks (skript töötab ka suurema serverite arvuga). Selle põhjal töövoog on jagatud kolmeks osaks: alal rohelise taustaga asuvad ühised protsessid mitme ja ühe serveri jaoks; kollasel taustal on mitmete serverite protsessid ja sinisel – ühe serveri protsessid. Voogude põhierinevus on kohtades, kus lõpetatakse skripti tööd: nt kui mitmete serverite korral ei õnnestunud paigaldada moodulit ühel serveril, siis tuleb jätkata teise serveriga. Lisaks, mitmete serverite korral töö lõpus näidatakse statistikat (vt Joonis 2), mille järgi on näha uuenduse seisu igal serveril.



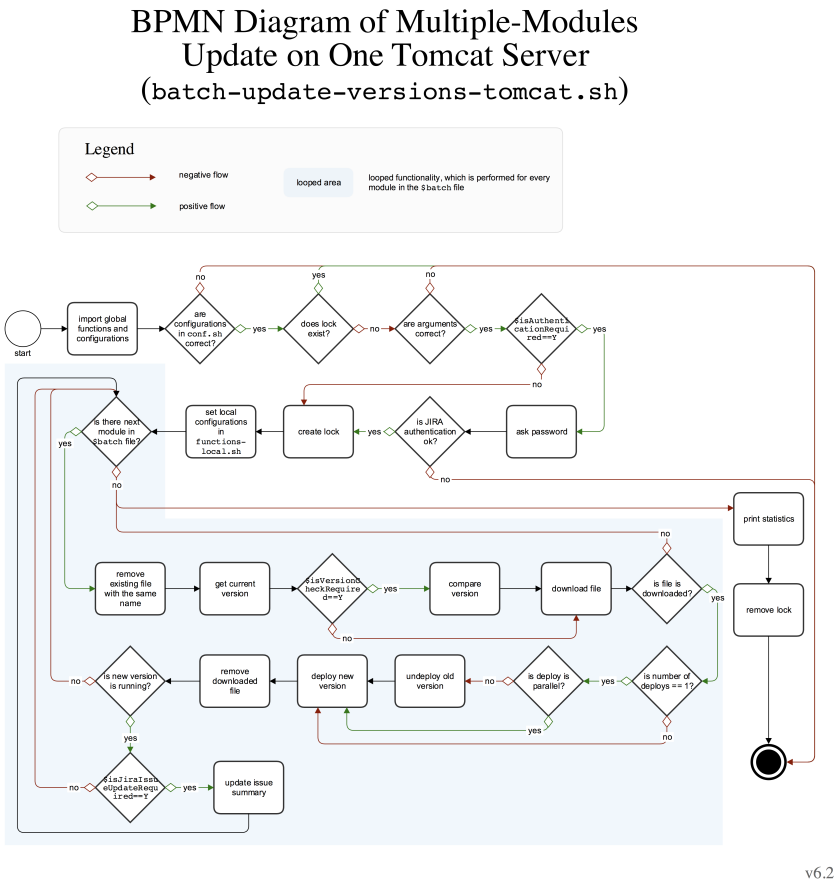
Joonis 2. Kolme mooduli uuenduse statistika.

Joonisel 2 on toodud statistika, mida kuvatakse mitmete moodulite uuendamisel või ühe mooduli uuendamisel mitmetel serveritel. See annab võimalust aru saada uuenduse staatust ilma terve väljundi lugemata. Antud juhul on toodud kolme mooduli `system`, `admin` ja `authorization` uuenduse statistika, mis näitab, et kaks moodulit on uuendatud edukalt ja `system` mooduliga oli kaks probleemi: uue versiooni number `0.0.0.0` on väiksem, kus vana versiooni number `3.13.1.7` ja skriptil ei õnnestunud alla laadida uue versiooni

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

faili `system-0.0.0.0`.

Joonisel 3 on toodud protsessid, mis toimuvad mitmete moodulite uuendamisel Tomcat 8 veebiserveril (faili `batch-update-versions-tomcat.sh` käivitamisel). Sinise taustaga alal on protsessid, mis korduvad iga mooduli jaoks.



Joonis 3. Skripti protsesside mudel mitmete moodulite uuendamisel Tomcat 8 veebiserveril [9].

[9] `https://iriiiina.gitbooks.io/version-updater-manual/content/BPMN-diagram-multiple-module-update-tomcat.png`

Oluline funktsionaalsus, mis on olemas nii mitmete kuid ka ühe mooduli uuendamisel on lukustamine, logimine, väljundi andmine, kasutaja teavitamine ja ühiste failide sünkroniseerimine.

Lukustamine katab nõuet 8 Tabelist 2. Töö alguses skript kontrollib, kas serveril eksisteerib lukufail, kui mitte, siis tekitab seda ja jätkab tööd. Lukustamise loogika võimaldab vältida paralleelset uuendust sama veebserveri peal (lukufaili nimetust defineeritakse globaalse muutuja `$lock` abil).

Logimine katab nõuet 6 – skript salvestab logid enda tegevuste kohta (globaalne muutuja – `$log`), mille järgi saab teada kes ja millal uuendas mingit moodulit.

Skript annab värvilist väljundit iga tehtud sammu kohta (vt Joonis 6). See on seotud nõuetega 1 ja 7: värviline väljund aitab kiiremini aru saada uuenduse staatust (kas see on õnnestunud, kui mitte, siis samm ebaõnnestus), mis salvestab uuenduse protsessi aega; testijatel ei ole vajadust uurida serveri logifailid, et aru saada mis sammul protsess oli katkestatud. Lisaks, nõuele 1 kulub ka terminali teavitus (mis on realiseeritud `$bell` vahendiga), annab teada, kui skript ootab mingit sisendit kasutaja poolt (nt JIRA parooli) või kui töö on lõppenud. See tähendab, et kasutaja ei pea jälgima skripti tööd, et aru saada uuenduse staatust.

Failide sünkroniseerimine on seotud nõuega 10. Kuna skripti failid peavad asuma igal serveril, see tähendab, et nende parandust ja täiendust on vaja teha mitu korda. Selleks, et vältida töö dubleerimist, ühised failid (mis vastavad loogika eest ja ei ole serveri seadistusfailid) asuvad repositooriumis ja igal serveril on seadistatud Crontab [9] tööd, mis regulaarsed kontrollivad, kas repositooriumis on olemas faili uus versioon. Kui on olemas, siis seda laaditakse alla. See tähendab, et skripti parandusi ja täiendusi on vaja teha ainult üks kord repositooriumis. Joonisel 5 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 5. Crontab tööde kirjelduse näide ühisfailide automaatse alla laadimiseks repositooriumist.

Skript koosneb 16 failidest. Neid võib jagada kolmeks grupiks. 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 eesmaärke.

Ühised failid – failid, mis on ühised kõikide serverite jaoks (mõned sõltuvad veebiserverist) ja mida võib 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 veebiserveril;
- `version-updater/functions-wl.sh` – globaalsed WebLogic 12 funktsioonid, mida kasutatakse ainult WebLogic veebiserveril;
- `deploy-undeploy-script.py` – Python[10] skriptid WebLogic veebiserveril moodulite uuenduse jaoks;
- `update-version-tomcat.sh` – ühe mooduli uuenduse skript, kus on kirjutatud funktsioonide kasutamine õiges järjekorras Tomcat veebiserveri jaoks;
- `batch-update-versions-tomcat.sh` – mitmete moodulite uuenduse skript Tomcat veebiserveri jaoks;
- `update-version-wl.sh` – ühe mooduli uuenduse skript WebLogic veebiserveri jaoks;

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

- `batch-update-versions-wl.sh` – mitmete moodulite uuenduse skript WebLogic veebiserveri jaoks;
- `version-update/extended-modules.txt` – nimekiri moodulitest, milledel on olemas erinevad prefiksid (näiteks, Eesti keskkondades paigaldatakse moodulit `person`, kuid Leedu keskkondades `person-lt`, kuna seal on olemas eriloogika Leedu isikute jaoks);
- `version-updater/readme.txt` – üldine informatsioon skripti autori ja juhendi kohta.

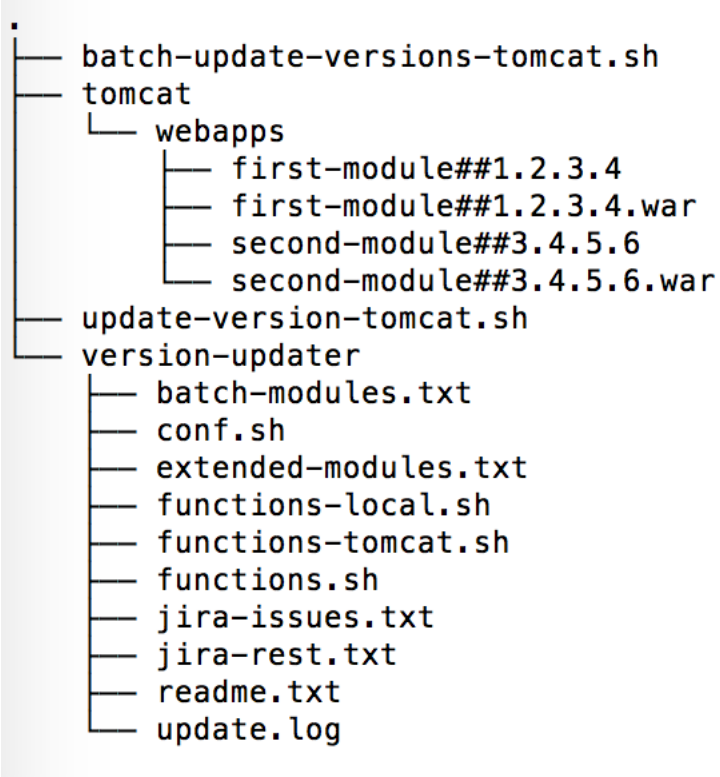
Kohalikud konfiguratsiooni failid – failid, mis sõltuvad serverist, kus skript on paigaldatud.

- `version-updater/conf.sh` – konfiguratsiooni fail, kus asuvad globaalsed muutujad, mille väärtused sõltuvad serverist, kus skript on paigaldatud;
- `version-updater/functions-local.sh` – kohalikud funktsioonid, mis sõltuvad serverist;
- `version-updater/batch-modules.txt` – fail, kus kasutaja defineerib moodulite nimetusi ja versioone mitmete moodulite uuendamiseks (faili nimetust 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 mis ei tohi olla muudetud kasutaja poolt.

- `version-updater/update.log` – fail, kus logitakse skripti tegevusi (faili nimetust defineeritakse globaalse muutujaga `$log`);
- `version-updater/jira-rest.txt` – tühifail, mida skript kasutab REST [7] teenuste genereerimiseks JIRA töö uuendamise [8] jaoks (on vajalik ainult JIRA kasutamisel ja on defineeritav globaalse muutujaga `$rest`).

Joonisel 4 on toodud näide skripti failide struktuurist serveril, kus on kasutusel Tomcat 8. Serveri juurkaustas asub alamkaust `/tomcat` veebiserveriga, kus on paigaldatud moodulid `first-module` versiooniga `1.2.3.4` ja `second-module` versiooniga `3.4.5.6`. Uuenduse failid asuvad serveri juurkaustas: `batch-update-versions-tomcat.sh` ja `update-version-tomcat.sh`. Skripti abifailid asuvad kaustas `/version-updater`.



Joonis 4. Skripti failide struktuur veebiserveriga Tomcat 8.

Failid, mida peab kasutama lõppkasutaja on `update-version-*.sh` ja `batch-update-versions-*.sh` (lisaks, logifail`update.log` toimunud uuenduste info saamiseks). Kõik ülejäänud failid on skripti töö toetamiseks ja seadistamiseks.

3.2 Skripti paigaldus

Projekti repositooriumis on olemas kõik skripti failid, mida vajadusel saab alla laadida serveri, kohaliku või virtuaalmasina peale. Samuti on olemas zip-fail, kus on arhiveeritud kõik failid vastavalt skripti versioonile ja veebiserverile (on olemas eraldi arhiivid Tomcat ja WebLogic jaoks). Skripti paigaldamise protsess on kirjeldatud juhendis ja koosneb neljast sammust.

- Vastavata zip-faili alla laadimine serveri, kohaliku või virutaalmasina peale.
- Alla laaditud faili lahti pakkimine.
- 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.

3.3 Skripti kasutus

Skripti käivitamiseks kasutatakse kaks faili: ühe mooduli uuendamiseks `update-version-*.sh` ja mitmete moodulite uuendamiseks `batch-update-versions-*.sh` (`*` asemel on `tomcat` või `wl`). Kõik ülejäänud failid skript kasutab enda töös ja kasutaja ei pea neid käivitama.

Ühe mooduli uuendamise skriptile on võimalik anda 4 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 mooduli nimetust, mida soovitakse uuendada (näiteks, Tomcat-i puhul see on sama nimetus, mis tee (*path*) Tomcat Manager-is [11]).
- `MODULE_VERSION` – kohustuslik parameeter, mis määrab versiooni, mida soovitakse paigaldada.
- `JIRA_USERNAME` – parameeter on kohustuslik juhul, kui JIRA autentimine on kohustuslik (globaalne muutuja `$isAuthenticationRequired="Y"`).
- `p` – mitte kohustuslik parameeter, mille sisestamisel teostatakse paralleelset uuendust [12] (ei toimu vana versiooni eemaldamist).

Mitmete moodulite uuendamise skriptile on võimalik anda 2 sisendparameetrit:

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

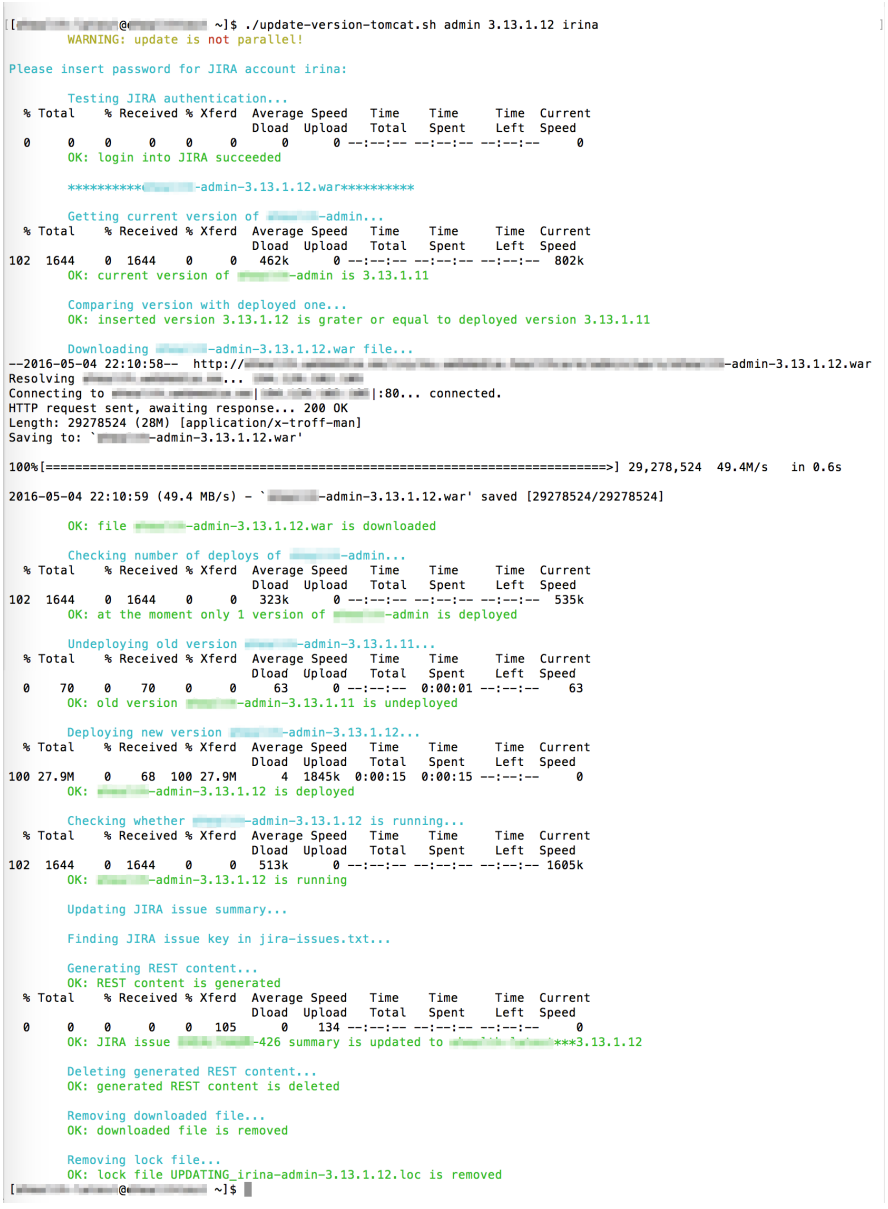
näide: `./batch-update-versions-tomcat.sh irina p`

- `JIRA_USERNAME` – parameeter on kohustuslik juhul, kui JIRA autentimine on kohustuslik (globaalne muutuja `$isAuthenticationRequired="Y"`).
- `p` – mitte kohustuslik parameeter, mille sisestamisel teostatakse paralleelset uuendust (ei toimu vana versiooni eemaldamist).

Joonisel 6 on näidatud ühe mooduli edukas uuendus. Skript annab väljundi kõikidest sammudest, mis oli tehtud.

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

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



Joonis 6. Ühe mooduli edukas uuendus skripti abil.

3.4 Dokumentatsioon

Skripti kohta on olemas dokumentatsioon, mis on uuenduse raamistiku oluline osa. Skripti juhend kirjeldab seda, kuidas saab kasutada, paigaldada ja seadistada skripti. Juhend on kätte saadav kõikidele meeskonna liikmetele, mis tähendab, et iga ühel on olemas võimalus seda kasutada. Mõlemad skripti projekti juhend ja avalik juhend on kättesaadavad töö lisas (Lisa III ja IV, vastavalt).

4 Mõju projektile

Automaatne versiooniuuenduse raamistik asendas täielikult vana käsitsi protsessi. Selles on olemas nii eelised, kuid ka oma puudused.

Kõige oluline eelis on see, et skript täidab kõik nõuded Tabelist 2 "Nõuded versiooni uuenduse lihtsustamise lahendusele". Kõige märgatavamad nendest on nõuded 1 ja 2 – protsessi kiirus ja mugavus.

On mõõdetud vana ja uue uuenduse protsesside kiirusi. Automaatse protsessi alguseks oli võetud skripti käivitamise käsu kirjutamine ja lõpuks skripti teavitamine, et töö on lõppenud. Skripti tööaja mõõtmiseks kasutati UNIX käsurea funktsiooni `time` [13]. Käsitsi uuenduse protsess koosneb sammudest, mis kirjeldati Tabelis 1 "Manuaalse uuenduse protsessi sammud" ja kulunud aeg mõõdeti stopperiga.

Valiti neli protsessi, mis teostati samal serveril ja samade WAR-failidega: kolm üksikute kõige populaarsemate moodulite uuendamine ja kõikide moodulite uuendamine (võetud keskkonnas oli neid 18 tk). Iga protsess oli tehtud vana käsitsi viisi ja uue automaatse skriptiga kolm korda ja võrdluseks on võetud katsetuste keskmised väärtused. Tulemused on toodud Tabelis 4.

Tabel 4. Vana ja uue protsesside kiirused.

Moodul	**Aut. uuendus**	**Käsitsi uuendus**
admin	27s	80s
treatment	61s	112s
reception	38s	100s
kõik (18 tk)	9m 12s	34m 8s

Tegelikkuses kokkuhoidud aeg on veel suurem, sest vanas protsessis uuendaja tähelepanu oli hõivatud terve protsessi jooksul. Uue skripti kasutades testijatel on olemas võimalus tegeleda paralleelselt teiste ülesannetega ja mitte jälgida skripti tööd (see ise annab teada, kui uuendus on lõppenud).

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

Kõige olulisem puudus on see, et testijad enam ei tunne versiooni uuenduse protsessi tausta ja vigade tekkimise juhul (kas serveri, veebiserveri või skripti pärast) ei oska neid lahendada. See on üldine automatiseerimise probleem, millega tuleb leppida mugavuse vahetuseks.

Käesoleva töö raames küsiti tagasisidet uue raamistiku kohta meeskonnaliikmete käest, kes on seda kasutanud. Allpool on toodud mõned väljavõtted, terved tekstid on kättesaadavad Lisas V.

> "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 edasiarendus

Bash skripti kood Apache Tomcat veebiserveri jaoks on avalik ja asub GitHub repositooriumis (vt Lisa II). See tähendab, et igaüks, kellel on sarnane takrvaraarenduse süsteem, saab seda kasutada ja täiendada enda nõuete järgi. Iga faili sees on olemas põhjalikud kommentaarid, mis seletavad faili eesmärki ja selle kasutamisviisi, mis lihtsustab skripti arusaamist. Samuti GitHub platvormil on olemas võimalus raporteerida vigu ja tellida uut funktsionaalsust.

Nõuded Bash skripti avaliku versiooni kasutamiseks.

- Masin, kus on paigaldatud Bash vahend.
- Apache Tomcat 8 veebiserver (vajadusel skripti saab muuta Tomcat 7 ja 6 versioonide jaoks).
- Standardsed UNIX vahendid: `find` [14], `grep` [15], `wget` [16] (kui mooduli failid asuvad veebis), `curl` [17].
- Standardne käsurea `bell` funktsionaalsus, mis saadab teavitusi kasutajale terminali kaudu.

Lisaks on skriptil olemas avalik dokumentatsioon, mis asub GitBook repositooriumis (vt Lisa IV). See sisaldab kasutusjuhendit, paigaldamise ja seadistamise juhendit ja kirjeldab kuidas on võimalik laiendata skripti tööd standartsete UNIX vahenditega.

Koodi ja dokumentatsiooni avalikkus tähendavad mitte ainult seda, et igaüks saab rakendada seda omas projektis, aga ka olemasoleva lahenduse optimiseerimist ja uue funktsionaalsusega täiendamist.

[14] `http://www.gnu.org/software/findutils/manual/html_mono/find.html`
[15] `https://www.gnu.org/software/grep`
[16] `https://www.gnu.org/software/wget`
[17] `https://curl.haxx.se`

Kokkuvõte

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

Bakalaureusetöö teoreetilise osa käigus koostati 12 nõuet probleemi lahendusele ja uuriti 4 erinevat tööriista versiooni uuenduse haldamiseks. Mitte ükski neist ei vastanud kõikidele nõuetele. Lahenduseks valiti enda Bash skripti kirjutamist.

Bakalaureusetöö praktilise osa käigus oli kirjutatud Bash skript versioonide uuendamiseks, mis vastab kõikidele nõuetele. Oli tehtud ka juhend selle skripti paigaldamise, seadistamise ja kasutamise kohta.

Bakalaureusetöö püstitatud eesmärgid said täidetud ning kirjutatud Bash skript aitab kiiremini ja mugavam uuendada versioone projektis. Seda kinnitavad meeskonnaliikmete tagasiside ja vana ja uue protsesside aegade võrdlemine.

Loodud lahendust saavad kasutada ja täiendada ka teised projektid, kuna selle kood ja dokumentatsioon on avalikud.

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

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

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

Lisad

I Terminid

Demo rakendus – rakendus, mida kliendid kasutavad funktsionaalsuse testimiseks.

Moodul – tarkvara osa, kus on kogutud ja pakitud ühe funktsionaalsuse kood (üks WAR-fail on üks moodul)

Modulaarne süsteem – tarkvaraarenduse süsteem, mille korral kogu koodi jagatakse moodulite kaupa, selleks, et ühe funktsionaalsuse tarnimine ei nõuaks terve rakenduse uuendamist.

Raamistik (*framework*) – süsteem, mis kirjeldab protsessi teostamist. Töö raames raamistiku all mõeldatakse Bash skripti koodi ja selle dokumentatsiooni.

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

Server – riistvara- ja tarkvarasüsteem, kus on paigaldatud veebiserver.

Test rakendus – rakendus, mida projekti meeskonnaliikmed kasutavad tarkvara testimiseks.

Toodang rakendus – lõppkasutajate 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) (http://akit.cyber.ee/term/3152-veebiserver).

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

II Skripti lähtekood

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

III Skripti projekti juhend

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

IV Skripti avalik juhend

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

V 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 keskkonda 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.'

VI Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

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

1. annan Tartu Üikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Versiooniuuenduse raamistik 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