Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/01BE40Sedlar

AUTOMATSKA VERIFIKACIJA REPLIKACIJE U OKVIRU SCADA SISTEMA

AUTOMATIC REPLICATION VERIFICATION WITHIN A DISTRIBUTED SCADA SYSTEM

Milica Sedlar, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu je opisan postupak automatizacije testova za verifikaciju konfiguracije u okviru DMS-a (Distributed Management System). Takođe je opisan i proces testiranja verifikacije konfiguracije, kao i alati i tehnologije koje su korišćene za izradu praktičnog dela zadatka. Automatizacija testova za verifikaciju konfiguracije se sastoji u manipulaciji lokalnim i daljinskim prekidačima u produkcionom sajtu i proveri replikacije u drugim sajtovima u sistemu pomoću aplikacije.*

Ključne reči: *DMS, replikacija, verifikacija*

Abstract – *This paper describes the procedure for automation of tests to verify the configuration on DMS (Distributed Management System). It also describes the process of testing to verify the configuration, as well as the tools and technologies used to create a practical part of the task. Automation of tests to verify the configuration consists in manipulating local and remote switches in the production site and checking replication in other sites in the system using the application.*

Keywords: *DMS, replication, verification*

1. UVOD

Testiranje softvera je od izuzetne važnosti u procesu životnog ciklusa razvoja softvera. Kako bi se u što većoj meri smanjile greške koje nastaju prilikom razvoja softverskog proizvoda, proces testiranja softvera u velikoj meri utiče na smanjenje nastalih grešaka kao i na kvalitet samog proizvoda. Kako bi se proces testiranja ubrzao, a samim tim i smanjila ljudska greška prilikom testiranja, vrši se automatizacija test slučajeva.
Verifikacija softvera predstavlja jednu fazu testiranja softvera. U ovoj fazi testiranja se utvrđuje da li softver odgovara zahtevima korisnika.
Konfiguracija predstavlja instalaciju i podešavanje softvera kako bi isti bio spreman za dalje testiranje ili isporuku krajnjem korisniku.
U ovom radu će biti opisana automatizacija testova za verifikaciju konfiguracije distribuiranog sistema. Kako je distribuirani sistem kompleksan po svojoj strukturi jer se sastoji od više samostalnih računara međusobno povezanih računarskom mrežom, samim tim je i verifikacija konfiguracije kompleksna. Upravo se zbog kompleksnosti verifikacije konfiguracije distribuiranog sistema javila ideja da se ova vrsta testova automatizuje, odakle je i nastalo celokupno istraživanje kao i tema rada. Testovi za verifikaciju konfiguracije čija automatizacija će biti opisana u ovom radu jesu testovi za replikaciju.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Darko Čapko, vanr. prof.

2. TESTIRANJE SOFTVERA

Testiranje softvera [1][2] je proces analize elemenata softvera kako bi se utvrdile razlike između postojećeg stanja i zahteva korisnika, ali i kako bi se ustanovile karakteristike softvera.
To je jedan od najviše korišćenih metoda za upravljanje rizikom, a sve u cilju verifikacije ispunjenja funkcionalnih zahteva.
Testiranje softvera predstavlja petu fazu u procesu razvoja softvera po metodologiji „Vodopad“ čija aktivnost je integrisanje svih komponenti, verifikacija, validacija, instalacija i obuke, a čiji je rezultat dobijanje funkcionalnog sistema.

3. KORIŠĆENE TEHNOLOGIJE I ALATI

U toku procesa istraživanja i implementiranja aplikacije za automatsko testiranje verifikacije konfiguracije sistema korišćeno je više različitih alata i tehnologija koje će biti opisane u nastavku.

3.1. Microsoft Visual Studio

Microsoft Visual Studio [3] je integrisano razvojno okruženje koje se koristi za razvoj računarskih programa, kao i web stranica, web aplikacija, web servisa i mobilnih aplikacija.
Visual Studio koristi razvojne platforme kao što su Windows API, Windows Forms, Windows Presentation Fundation kao i mnoge druge platforma. Integrisani debager funkcioniše jednako dobro kako na nivou debagovanja izvornog koda tako i na mašinskom nivou debagovanja.
Microsoft Visual Studio podržava trideset šest različitih programskih jezika. Ugrađeni programski jezici su: C, C++, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML i CSS. Podrška za ostale programske jezike, kao što su Python, Ruby, Node.js kao i mnogi drugi, je potrebno instalirati kao posebne dodatke.

3.2. Extensible Markup Lenguage (XML)

XML [4] predstavlja markup jezik koji definiše skup pravila za enkodiranje dokumenata u format koji je čitljiv i za čoveka i za mašinu. Definisan je od strane World Wide Web Consortium-a (W3C) kroz XML 1.0 specifikaciju i predstavlja besplatan i otvoren standard. Predstavlja tekstualni format podataka sa podrškom Unicode karaktera za veliki broj govornih jezika.
Kao dodatak tome što je XML dobro formatiran, može se proveriti validnost XML dokumenta. To znači da je moguće izvršiti proveru da li su svi elementi u dokumentu opisani u skladu sa sintaksnim pravilima određenim XML šemom. XML procesori koji služe za obradu podataka mogu voditi računa o validnosti samog dokumenta, ali ukoliko vode, moraju imati mehanizam za prijavu grešaka ukoliko je dokument nevalidan.

3.3. Windows Power Shell

Windows PowerShell je [5] skript jezik za pisanje željenih komandi koje je potrebno izvršiti. Dizajniran je posebno za potrebe sistemske administracije radi što brže automatske administracije operativnih sistema kao što su Linux, macOS, Unix i Windows.
Windows PowerShell podržava command-line i skript okruženja eliminišući kopleksne probleme i dodajući nove karakteristike. Baziran je na objektima, a ne na tekstu, tako da se na izlazu i dobijaju objekti, a ne tekst. Izlazni objekat jedne komande je moguće poslati kao ulazni objekat drugoj komandi.
Windows PowerShell olakšava proces tranzicije od pisanja komandi interaktivno do kreiranja i pokretanja skripti.

3.4. C# Programski jezik

Programski jezik C# je dizajniran da bude upotpunosti kompatibilan sa .NET kontrolisanim programskim okruženjem. Dizajniran je da bude platformski nezavisan. Kako je to objektno-orijentisani programski jezik, spada u programske jezike višeg reda i predstavlja deo .NET kontrolisanog programskog okruženja.
C# programski jezik ne podržava višestruko nasleđivanje, pa se kao rešenje ovog problema koriste interfejsi. Takođe, ovaj programski jezik ima još jednu korisnu karakteristiku, a to je „garbage collector“. To znači da nije potrebno praviti destruktor za svaku klasu. Moguće je i direktno pristupiti memoriji upotrebom pokazivača, ali pokazivače neće „počistiti“ garbage collector sve dok se to posebno ne navede.

3.5. Active Directory Services Interface Edit (Adsi Edit)

Active Directory Services [6] je repozitorijum mreže i aplikacija koje koriste više korisnika ili više nekih drugih aplikacija. Alat za pristupanje servisu aktivnog direktorijuma je Adsi Edit. Pomoću ADSI Edit alata je moguće na jednostavan način manipulisati objektima koji se nalaze na servisu direktorijuma. Kao što je moguća jednostavna manipulacija objektima, na isti način je moguća i jednostavna manipulacija atributima tih objekata.
ADSI Editi dolazi u sklopu instalacije Windows Servera.
Za pristupanje objektima servis direktorijuma se koristi LDAP (Lightweight Directory Access Protocol) standard koji definiše kako je taj servis direktorijum zaista implementiran i dostupan.

3.6. Domain Name System (DNS)

DNS [6] predstavlja najveću digitalnu bazu podataka na svetu. Web pretraživači komuniciraju putem internet protokol (IP) adresa. DNS prevodi imena domena, imena mašina i imena servisa u IP adrese, takođe IP adrese prevodi u imena mašina, ali i imena mašina prevodi u alternativna imena (aliase). Svaki uređaj koji je povezan na internet ima jedinstvenu IP adresu koju druge mašine koriste kako bi pronašle uređaj. Za mapiranje podataka koji govore DNS serveru sa kojom IP adresom se povezuje svaki domen, kao i kako se rukovodi sa zahtevima koji se šalju svakom domenu, koriste se DNS rekordi.

4. AUTOMATIZACIJA TESTOVA ZA VERIFIKACIJU KONFIGURACIJU REPLIKACIJE U DISTRIBUIRANOM SISTEMU

Distribuirani sistem je zbog njegove složenosti veoma teško konfigurisati i taj proces se najčešće izvodi sporo. Kako je konfiguracija ovakvog sistema ispraćena velikim brojem konfiguracionih koraka, te konfiguracione korake je potrebno i verifikovati. Verifikacija konfiguracije [7] se svodi na test slučajeve koje je potrebno izvršiti kako bi se utvrdilo da li je sistem dobro konfigurisan i da li je spreman za dalju upotrebu. Verifikacija konfiguracije distribuiranog sistema je od velikog značaja iz razloga što loše konfigurisan sistem može dovesti do otkaza sistema. S obzirom na to da je konfiguracija kompleksna, i verifikacija konfiguracije je takođe kompleksna i spora. Kako bi se ubrzao proces i otklonila mogućnost greške prilikom verifikacije konfiguracije ovakvih sistema, automatizacija testova ove vrste se našla kao moguće rešenje.
Osim povećanja tačnosti, automatizacija ove vrste testova doprinosi i smanjenju vremena koje je potrebno utrošiti da bi se konfiguracija verifikovala.
Verifikacija konfiguracije replikacije među sajtovima je od izuzetne važnosti za upravljanje distribuiranim sistemima kao što je DMS jer i najmanja konfiguraciona greška može da ima katastrofalne posledice. Zbog posledica koje konfiguracione greške mogu da uzrokuju, javila se potreba da se istraži mogućnost detektovanja grešaka na vreme, sa velikom pouzdanošću i u što kraćem vremenskom periodu, još u fazi verifikacije. Upravo je ta potreba stvorila ideju za razvoj aplikacije koja ubrzava i povećava pouzdanost verifikacije.

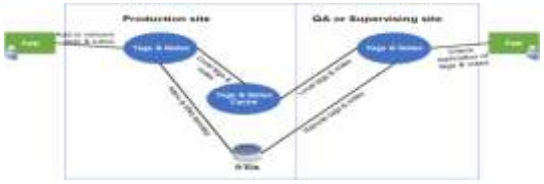
4.1. Opis automatizacije testova za verifikaciju konfiguraciju replikacije

Kako verifikacija konfiguracije replikacije u DMS sistemu traje dugo, a pritom je mogućnost ljudske greške neizbežna, razvijena je aplikacija kako bi se ovaj proces ubrzao i kako bi se greške svele na minimum.
Automatizovani su verifikacioni testovi za proveru konfiguracije replikacije u distribuiranom sistemu koji

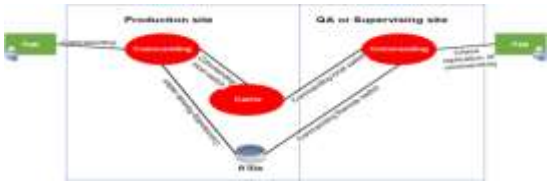
ima tri sajta. Test slučajevi verifikuju replikaciju u DMS sistemu tako što se vrši manipulacija prekidačima (komandovanje (otvaranje i zatvaranje) lokalnog i daljinskog prekidača kao i dodavanje i brisanje oznaka i beleški na prekidačima). Komandovanje i dodavanje oznaka i beleški se vrši u produkcionom sajtu, dok se provera replikacije vrši u sajtovima za kontrolu kvaliteta i nadgledanje.

Replikacija između sajtova u distribuiranom sistemu kao što je DMS je od velike važnosti zbog praćenja eventualnih otkaza na distributivnoj mreži, kao i planiranja i testiranja preključivanja na istoj. Ukoliko nema replikacije između sajtova, može doći do velikih problema koje mogu da izazovu katastrofalne posledice čak i po život ljudi.

Svaki sajt ima računarski resurs na kome se nalazi servis koji je neophodno da bude startovan kako bi replikacija između sajtova bila moguća. Informacije o manipulaciji lokalnim prekidačima se čuvaju u keš memoriji koja je zajednička za sve sajtove u sistemu, dok se informacije o manipulaciji daljinskim prekidačima čuvaju u RealTime bazi podataka. Slika 4.1. prikazuje replikaciju oznaka i beleški, dok slika 4.2. prikazuje repikaciju komandovanja prekidačima.



Slika 4.1. *Prikaz replikacije oznaka i beleški*

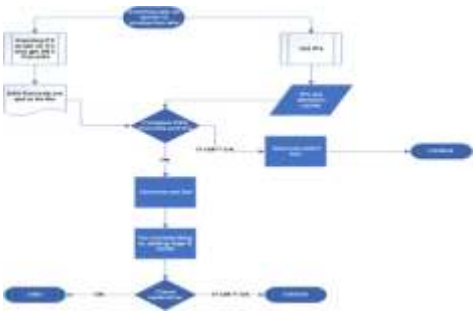


Slika 4.2. *Prikaz replikacije komandovanja prekidačima*

4.2. Opis rešenja automatizacije testova za verifikaciju konfiguracije replikacije

U toku procesa automatizacije testova za verifikaciju replikacije obuhvaćena su četiri testa, a to su komandovanje (otvaranje i zatvaranje) lokalnim prekidačima, komandovanje (otvaranje i zatvaranje) daljinskim prekidačima, dodavanje i brisanje oznaka na lokalnim i daljinskim prekidačima i dodavanje i brisanje beleški na lokalnim i daljinskim prekidačima. Ovim testovima se verifikuje replikacija između sajtova u DMS sistemu, a tok rada aplikacije za verifikaciju je prikazan na slici 4.3.

Aplikacija se pokreće u produkcionom sajtu na jednom računarskom resursu. Kada se aplikacija pokrene, vrši se provera da li je određen servis, koji je potreban za replikaciju između sajtova, startovan. Ta provera se vrši tako što se izvrši Power Shell skripta na DC-u sistema koja pročita DNS rekorde servisa i upiše u tekstualni fajl, nakon čega se ti DNS rekordi iz tekstualnog fajl uporede sa IP adresama na računarskom resursu.



Slika 4.3. *Prikaz toka rada aplikacije za verifikaciju replikacije*

Nakon što se proveri da li je servis startovan ili ne, vrši se komandovanje slučajno odabranim lokalnim i daljinskim prekidačima, kao i dodavanje oznaka i beleški na iste. Kada se završi manipulacija slučajno odabranim prekidačima, rezultat se upisuje u xml fajl.

Taj xml fajl se distribuira na računarske resurse na kojima se takođe nalazi određen servis koji je potreban za replikaciju, a koji se nalaze u drugim sajtovima. Računarski resursi na kojima se nalazi taj servis se čitaju iz ADSI Edita, a aplikacija uspe da pristupi ADSI Editu i pročita te informacije na osnovu LDAP putanje.

Replikacija se verifikuje u sajtu za nadgledanje i sajtu za kontrolu kvaliteta. Verifikacija konfiguracije replikacije se vrši tako što se čitaju podaci iz xml fajla, koji su dobijeni iz produkcionog sajta, i upoređuju sa podacima iz keš memorije, ako se manipuliše lokalnim prekidačima. Ukoliko se manipuliše daljinskim prekidačima, verifikacija konfiguracije se vrši poređenjem podataka iz xml fajla sa podacima iz RealTime baze.

Nakon što se replikacija verifikuje, potrebno je vratiti sistem u početno stanje. Vraćanje sistema u početno stanje se vrši u produkcionom sajtu tako što se predhodno dodate oznake i beleške obrišu, a lokalni i daljinski prekidači se otvore, odnosno zatvore u zavisnosti od predhodne akcije

5. REZULTATI TESTIRANJA

Testiranje aplikacije je vršeno u testnom okruženju koje se sastoji od tri sajta (produkcioni, sajt za nadgledanje i sajt za kontrolu kvaliteta) sa po jednim računarskim resursom sa određenim servisom u svakom sajtu.

U toku testiranja poređeno je vreme trajanja ručno rađene verifikacije konfiguracije replikacije sa vremenom koje je potrebno da se verifikacija uradi pokretanjem aplikacije.

Tabela 1. prikazuje vreme trajanja verifikacije konfiguracije replikacije rađene ručno, dok tabela 2. prikazuje vreme trajanja verifikacije konfiguracije replikacije rađen automatski, puštanjem aplikacije.

Prvi korak koji se vrši kada se verifikuje konfiguracija replikacije jest provera da li je startovan servis. Ručna provera ovog koraka zahteva da se određeni servis pronađe, detektuje i utvrdi njegovo stanje, dok se automatski (puštanjem aplikacije) stanje servisa utvrđuje iz programskog koda što skraćuje izvršavanje ovog koraka za 120s.

Tabela 1. *Vreme trajanja verifikacije konfiguracije replikacije rađene ručno*

Aktivnost za izvršavanje	Vreme trajanja izvršavanja aktivnosti [s]
Provera da li je servis startovan	300 s
Komandovanje lokalnim prekidačima	30 s
Komandovanje daljinskim prekidačima	30 s
Dodavanje oznaka	40 s
Dodavanje beleški	40 s
Provera replikacije u sajtu za nadzor	180 s
Provera replikacije u sajtu za kontrolu kvaliteta	180 s
Vraćanje lokalnog prekidača na početno stanje	30 s
Vraćanje daljinskog prekidača na početno stanje	30 s
Brisanje oznaka	30 s
Brisanje beleški	30 s
Ukupno vreme trajanja verifikacije	920 s

Tabela 2. *Vreme trajanja verifikacije konfiguracije replikacije rađene automatski, puštanjem aplikacije*

Aktivnost za izvršavanje	Vreme trajanja izvršavanja aktivnosti [s]
Provera da li je servis startovan	20 s
Komandovanje lokalnim prekidačima	15 s
Komandovanje daljinskim prekidačima	15 s
Dodavanje oznaka	15 s
Dodavanje beleški	15 s
Provera replikacije u sajtu za nadzor	15 s
Provera replikacije u sajtu za kontrolu kvaliteta	15 s
Vraćanje lokalnog prekidača na početno stanje	15 s
Vraćanje daljinskog prekidača na početno stanje	15 s
Brisanje oznaka	15 s
Brisanje beleški	15 s
Ukupno vreme trajanja verifikacije	170 s

Komandovanje lokalnim ili daljinskim prekidačima zahteva da se na šemi nađe željeni lokalni ili daljinski prekidač, a zatim je potrebno ručno promeniti vrednost prekidača. Automatsko komandovanje lokalnim ili daljinskim prekidačem se vrši tako što se iz keš memorije nasumično odabere lokalni ili daljinski prekidač i promeni se njegova vrednost, bez potrebe traženja prekidača na šemi. Ručno komandovanje lokalnim i daljinskim prekidačima ukupno traje 60s, dok ja ze ovaj korak automatski potrebno polovina vremena (30s).

Za dodavanje oznaka i beleški je automatski potrebno ukupno 30s. Ručno izvršavanje zahteva odabir željenog prekidača na šemi, zatim je potrebno zadati komandu da se doda oznaka ili beleška i na kraju je dodati. Dodavanje ručno samo oznake ili samo beleške traje 40s, što je za 10s više nego što je potrebno za izvršavanje oba ova koraka automatski. Ovde je vidno značajno smanjenje vremena trajanja verifikacije.

Replikacija se provarava tako što se ode u sajt za nadzor ili u sajt za kontrolu kvaliteta i na šemi se pronađe lokalni ili daljinski prekidač kome je promenjena vrednost ili na koji je dodata oznaka ili beleška. Provera replikacije ručno po sajtu traje 180s. Automatski se replikacija proveri za 15s po sajtu jer se provera radi pristupanjem keš memorije, bez potrebe pokretanja šeme i traženja željenih prekidača. Provera konfiguracije replikacije automatski je znatno smanjena, čak za 165s po sajtu.
Automatizacijom vraćanja lokalnog i daljinskog prekidača na početno stanje, kao i brisanja oznaka i beleški je smanjeno vreme izvršavanja sa ukupno 120 s na ukupno 60 s, što je znatno uticalo na uštedu vremena koje je bilo potrebna za izvršavanje ovih koraka.

Ukupno vreme trajanja verifikacije konfiguracije replikacije je smanjeno sa 920s na 170s.

U tabelama se jasno vidi da je automatizacijom ova četiri testa za verifikaciju konfiguracije replikacije ušteda vremena velika.

6. ZAKLJUČAK

Cilj rada je bila automatizacija testova za verifikaciju konfiguracije replikacije u distribuiranom sistemu.

U prvom delu je bilo potrebno realizovati automatsko čitanje DNS rekorda sa DC-a, automatsko čitanje IP adresa sa računarskog resursa gde je potrebno izvršiti verifikaciju, kao i upoređivanje DNS rekorda i IP adresa radi utvrđivanja stanja servisa. Automatsko čitanje DNS rekorda je urađeno u skript jeziku Power Shell, dok je automatsko čitanje IP adresa i upoređivanje sa DNS rekordima urađeno u programskom jeziku C#.

U drugom delu je bilo potrebno automatizovati čitanje svih računarskih resursa sa određenim servisom sa svih sajtova iz Adsi Edita, dok je u trećem delu bilo potrebno realizovati manipulisanje lokalnim i daljinskim prekidačima i proveru replikacije. Drugi i treći deo su realizovani upotrebom programskog jezika C#.

Automatizacijom testova za verifikaciju konfiguracije je ubrzan proces verifikacije konfiguracije u distribuiranom sistemu, ali je povećana i pouzdanost verifikacije.

7. LITERATURA

[1] Elfriede Dustin, Jaff Rashka, Johan Pau, Automated Software Testing, Introduction, management and performance, New York, 2008.
[2] Myers:, Glendford J., The art of software testing., New Jersy, 2004.
[3] De, Alan, Visual SourceSafe: Microsoft's Source Destruction System. Highprogrammer.com, 2009.
[4] Pilgrim, Mark, The history of draconian error handling in XML, 2004.
[5] https://docs.microsoft.com/en-us/powershell/scripting/powershell-scripting?view=powershell-6 , datum pristupa 26.05.2018.
[6] Brian Desmond, Joe Richards, Robbie Allen, Alistair G. Lowe-Norris, Active Directory: Designing, Deploying, and Running Active Directory, 2010.
[7] David R. Bourgeois, James A. Ryan, Subhash C. Varshney, Data processing system with self testing and configuration mapping capability, USA, 1982.

Kratka biografija:

Milica Sedlar je rođena 1992. godine u Novom Sadu. Srednju školu je završila u Inđiji 2011. godine. Fakultet Tehničkih Nauka u Novom Sadu je upisala 2011. godine. Ispunila je sve obaveze i položila sve ispite predviđene studijskim programom na smeru Računarstvo i automatika, usmerenju Računarske nauke i informatika. Odmah nakon toga je upisala master akademske studije na smeru Primenjeno softversko inženjerstvo i ispunila sve svoje obaveze i položila sve ispite predviđene studijskim smerom.