Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42
DOI: https://doi.org/10.24867/09BE25Stankovic

GENERIČKI IEC 61968-100 ADAPTER

GENERIC IEC 61968-100 ADAPTER

Đina Stanković, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisan je dizajn i implementacija generičkog adaptera za razmenu poruka u skladu sa standardom IEC 61968. Objašnjen je postupak testiranja i prikazane su performanse razvijenog rešenja. Na samom kraju je dat pregled mogućnosti za dalji razvoj adaptera.*

Ključne reči: *Web servisi, SOAP (Simple Object Access Protocol), XSD (XML Schema), mapiranje, adapter, IEC 61968*

Abstract – *This paper describes the design and implementation of a generic adapter compliant with IEC 61968. The test procedure is explained and the performance of the solution is presented. At the end, an overview of the possibilities for further development of the adapter is given.*

Keywords: *Web services, SOAP (Simple Object Access Protocol), XSD (XML Schema), mapping, adapter, IEC 61968*

1. UVOD

Uočena je mogućnost razvoja generičkog adaptera za razmenu poruka u skladu sa *IEC 61968* standardom, pri čemu adapter može posredovati u razmeni poruka između sistema. Postojanje generičkog adaptera donosi olakšicu u vidu izbegavanja promene implementacije, iznova i iznova, za svaki tip poruke ili sisteme sa kojima se komunicira, jer se konfiguracija adaptera podešava preko *UI*-a, bez spuštanja servisa. Zadatak generičkog adaptera je da obradi primljeni *SOAP* zahtev, pri čemu vrši određene transformacije podataka, i potom, da kreiranu izlaznu poruku pošalje odredišnom sistemu, čiji će odgovor samo proslediti izvornom sistemu.

U nastavku je data teorijska osnova neophodna za razumevanje rada adaptera. Opisan je dizajn i implementacija projektnog rešenja, kao i performanse koje podrazumevaju vreme potrebno da se obradi određen broj poruka.

2. TEORIJSKE OSNOVE

2.1. Integracije

Integracije obezbeđuju razmenu podataka između različitih sistema. Obično jedan sistem predstavlja izvor podataka, dok drugi sistem koristi te podatke na sebi svojstven način.

Zasnivaju se na nekoliko standarda i tehnologija: *CIM, Web Services, XML, XSD,* itd.

CIM (Common Information Model) je standard koji je razvio EPRI (*Electric Power Research Institute*), a koji je službeno usvojio *IEC* (*International Electrotehnical Commission*). Koristi se za definisanje integracionih modela: *XML* i *RDF* (*XSD* i *RDFS*), gde se kasnije generišu poruke u skladu sa *XSD* šemama [1].

XML (Extensible Markup Langugage) je standardni skup pravila za definisanje formata podataka propisan od strane *W3C* (*World Wide Web Consortium*), razumljiv i čoveku i računaru, definiše opštu sintaksu za označavanje podataka pomoću odgovarajućih tagova [2].

XSD (XML Schema) je način za strukturiranje pravila kreiranja *XML* dokumenta. Korišćenjem *XML* šeme moguće je detaljno opisati strukturu XML elemenata: kardinalnost, tip podatka, format podatka [3].

2.2. IEC 61986-100 standard

IEC 61968 standard je namenjen za podršku integracija aplikacija u elektroenergetskim objektima sa naglaskom na interfejse sistema za upravljanje distribucijom.

Međunarodni standard *IEC 61968-100* pripremljen od strane *IEC* tehničkog komiteta 57: Upravljanje i komunikacija u elektroenergetskom sistemu. Definiše skup implementacionih profila za *IEC 61968*, koristeći raspoložive integracione tehnologije, uključujući *WS (Web Service*) ili *JMS (Java Messaging Service)* [4].

2.3.1. Glagoli

Deo standarda identifikuje skup glagola (verbs) koji se koriste u komunikaciji i detaljno opisuje korišćenje svakog od njih.Postojeći glagoli su prikazani u tabeli 1.

Tabela 1 - *Glagoli i njihova upotreba*

Request Verb	***Reply Verb***	***Event Verb***	***Usage***
get	reply	(none)	query
create	reply	created	transaction
change	reply	changed	transaction
cancel	reply	canceled	transaction
close	reply	closed	transaction
delete	reply	deleted	transaction
execute	reply	executed	transaction

2.3.2. Imenice

Imenice (*nouns*) se koriste za identifikaciju tipa informacija koje se razmenjuju. Često se nazivaju profilima. Svaka imenica ima odgovarajuću *XML* šemu, koja je definisana u okviru jedinstvenog *namespace*-a, samo za tu imenicu.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio doc. dr Milan Gavrić.

2.3. Web Servisi

Web servisi [5] se zasnivaju se na klijent-server arhitekturi koja se oslanja na komunikaciju preko internet protokola. Ovim je omogućena razmena podataka između softverskih aplikacija koje su napisane različitim programskim jezicima i/ili pokrenute na različitim sistemskim platformama.

Web servisi koriste *XML* za kodiranje i dekodiranje podataka, a *SOAP* za razmenu podataka. Za transport se obično koristi *HTTP* protokol.

WSDL (Web Services Description Language) je jezik za opisivanje web servisa.

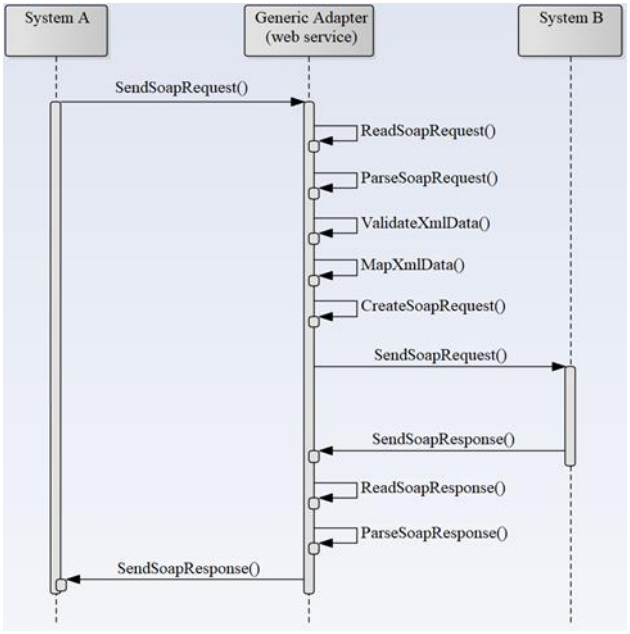
UDDI (Universal Description, Descovery and Integration) predstavlja centralizovanu lokaciju koja obezbeđuje mehanizam za registrovanje i pronalaženje web servisa.

3. SPECIFIKACIJA ZAHTEVA

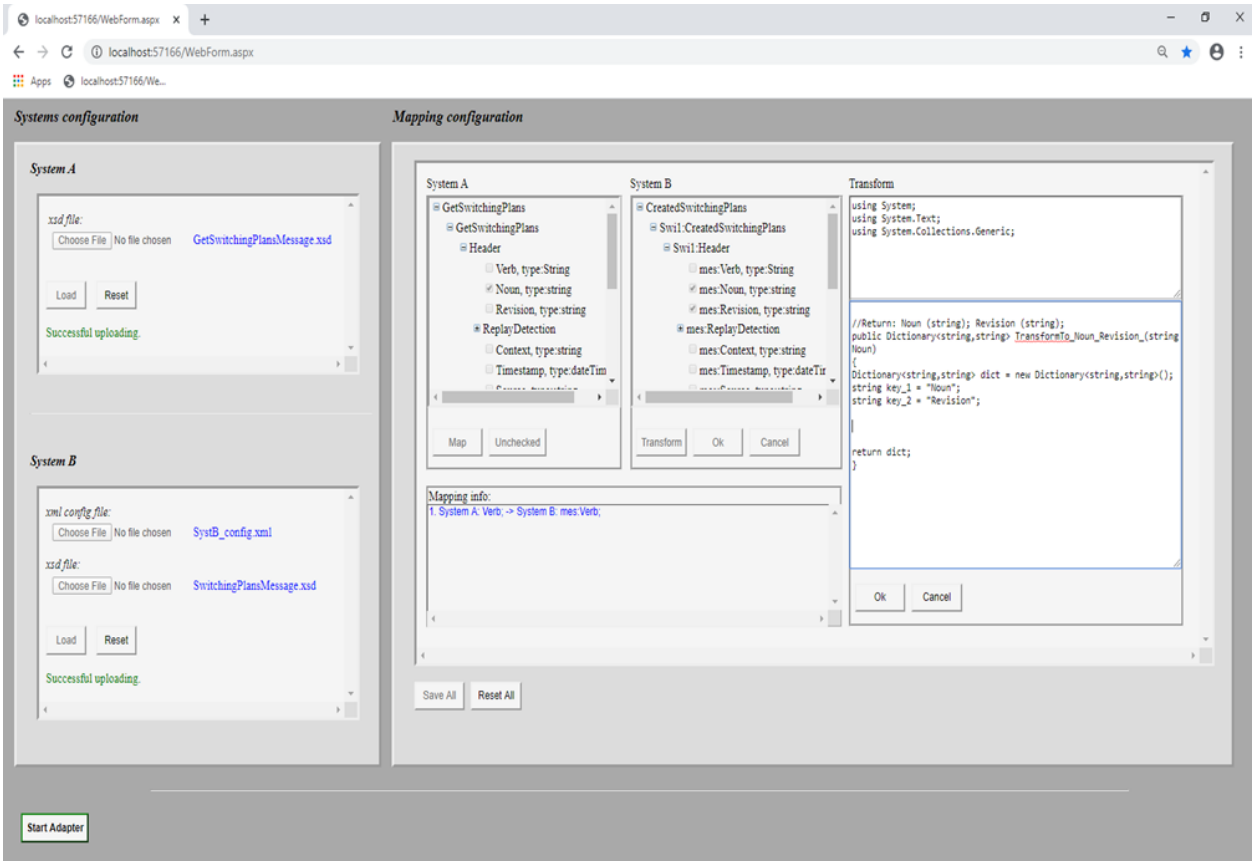
Podešavanje generičkog adaptera je dizajnirano da zadovolji sledeće zahteve:

- Učitavanje konfiguracija sistema između kojih posreduje
- Resetovanje učitanih sistemskih konfiguracija
- Prikaz XSD šema sistema u vidu TreeView-a
- Mogućnost mapiranja između sistema: jedan-na-jedan, jedan-na-više, više-na-jedan, više-na-više
- Mogućnost kreiranja transformacije prilikom mapiranja (obavezno kod mapiranja više-na-jedan i više-na-više)
- Odbacivanje podešenog mapiranja
- Čuvanje mapiranih podataka u SQLite bazi
- Prikaz mapiranih podataka
- Resetovanje celokupnog mapiranja
- Startovanje i zaustavljanje rada adaptera

Na slici 1 prikazan je dijagram sekvenci, koji predstavlja uspešan scenario obrade *SOAP* zahteva i komunikaciju sa sistemima. U slučaju da se desi neka greška prilikom obrade *SOAP* zahteva, adapter šalje tu informaciju sistemu A. Slika 2 prikazuje korisnički interfejs generičkog adaptera, u trenutku pisanja transformacije.



Slika 1- *Dijagram sekvenci*



Slika 1. *Dizajn korisničkog interfejsa generičkog adaptera*

Funkcionalnosti generičkog adaptera:

- Obrada primljenog *SOAP* zahteva (čitanje i parsiranje poruke radi dobijanja *XML* podataka)
- Validiranje *XML* podataka
- Mapiranje *XML* podataka, u skladu sa podešenom konfiguracijom
- Kreiranje i slanje *SOAP* zahteva odredišnom sistemu (sistemu B)
- Parsiranje *SOAP* odgovora (od sistema B)
- Slanje odgovora izvornom sistemu (sistemu A)

4. IMPLEMENTACIJA GENERIČKOG ADAPTERA

4.1. Podešavanje konfiguracije adaptera

Učitavanje sistemskih konfiguracija

Potrebno je učitati:

- za sistem A:
  - *XSD* fajl – za validaciju primljene poruke
- za sistem B:
  - *XML* fajl – sadrzi *URI* i *Method*
  - *XSD* fajl – za kreiranje izlazne poruke

Od učitanih *XSD* fajlova kreira se *XmlSchemaSet*, koja se konvertuje u *XML* elemente. Po završetku konvertovanja, *XML* fajlovi se čuvaju na određenoj lokaciji. *XML* fajlovi dobijeni konvertovanjem *XSD* fajlova odredišnog sistema u nastavku se koriste za kreiranje izlazne poruke. Na osnovu dobijenih *XML* fajlova, kreira se *TreeView* prikaz šema, koji omogućava podešavanje mapiranja.

Podešavanje mapiranja

Postoji 4 tipa mapiranja:

- jedan-na-jedan - sa/bez transformacije
- jedan-na-više - sa/bez transformacije
- više-na-jedan - obavezna transformacija
- više-na-više - obavezna transformacija

Prilikom kreiranja transformacije podataka, odabrani elementi šeme sistema A predstavljaju parametre metode, dok odabrani elementi šeme sistema B predstavljaju povratnu vrednost.

- Ukoliko je u pitanju mapiranje više-na-jedan ili jedan-na-jedan, povratna vrednost je tip tog jednog elementa.
- Ukoliko je u pitanju mapiranje više-na-više ili jedan-na-više, povratna vrednost je *Dictionary<string,string>*, gde je *Key* element na koji se mapira *Value*, bez obzira kog tipa su odabrani elementi. Ova povratna vrednost je odabrana jer se svakako u *XML* elementu upisuju vrednosti tipa *string*.

Nastali kostur metode transformacije može da se dopuni željenim kodom. Po završetku, čuvanjem kreirane metode, ona se čuva u *transform.cs* koja se kompajlira radi provere ispravnosti napisanog koda.

Nakon podešenog mapiranja, prilikom čuvanja istog, sve informacije se čuvaju u *SQLite* bazi. Ukoliko postoji više odabranih elemenata, oni se čuvaju u istom objektu. U bazi se čuvaju putanje do odabranih elemenata, njihovi tipovi i, ukoliko postoji, naziv metode transformacije, za svako podešeno mapiranje.

Prilikom čuvanja definisanog mapiranja podataka, datoteka *transform.cs* se kompajlira i dobija se *dll* fajl, koji se koristi prilikom transformacija podataka.

4.2. Funkcionalnost adaptera

Obrada zahteva

Po prijemu zahteva, adapter pristupa obradi poruke. Najpre pročita SOAP zahtev, potom preuzme XML podatke, nakon čega ih validira pomoću *XSD* fajlova učitanih prilikom konfigurisanja izvornog sistema (sistem A). Ukoliko podaci nisu validni, adapter vraća odgovor izvornom sistemu sa informacijom da su podaci nevalidni. Ukoliko validacija uspešno prođe, sledi mapiranje podataka.

Mapiranje podataka

Prilikom mapiranja, podaci se čuvaju u ranije kreiranom izlaznom *XML* fajlu, nastalom konvertovanjem učitanih *XSD* fajlova odredišnog sistema (sistema B).

Kao što je ranije spomenuto, mapiranje može biti sa ili bez transformacije. Interfejs *IXmlDataMapper* prikazuje koje vrste mapiranja postoje (Listing 1).

```
internal interface IXmlDataMapper
{
    void MappingOne
        (MapperInfo systA_info, List<string> systB_paths);
    void MappingToOne_WithTransform
        (List<MapperInfo> systA_infos, string systB_path, string transformFunct);
    void MappingToMulti_WithTransform
        (List<MapperInfo> systA_infos, List<string> systB_paths, string transformFunct);
}
```

Listing 1 - IXmlDataMapper interfejs

Mapiranje *XML* podataka vrši se na osnovu podataka sačuvanih prilikom konfigurisanja mapiranja. Po startovanju rada adaptera, svi podaci iz baze se preuzmu, pripreme za mapiranje i sačuvaju u listu kroz koju se kasnije, prilikom mapiranja, iterira i za svaki objekat se vrši mapiranje. U zavisnosti od vrste mapiranja, što je poznato na osnovu propertija pročitanog objekta, poziva se odgovorajuća metoda iz gore prikazanog interfejsa.

Ukoliko je potrebno izvršiti transformaciju pre mapiranja, poziva se metoda iz *ITransformer* interfejsa (Listing 2). Transformacija se vrši pomoću ranije kreiranog i spomenutog *dll* fajla. Refleksijom se preuzima odgovarajuća metoda, čije ime se dobija kroz parametar funkcije *GetTransformResult*. Po izvršenju metode, dobijeni rezultat je povratna vrednost koja se mapira na izlaz.

```
internal interface ITransformer
{
    object GetTransformResult
        (List<MapperInfo> systA_values, string transformFunct);
}
```

Listing 2 - ITransformer interfejs

Komunikacija sa odredišnim sistemom (sistemom B)

Po završetku mapiranja, ukoliko je sve uspešno prošlo, potrebno je poslati novonastale podatke odredišnom sistemu (sistem B). Pristigli odgovor sistema B se zatim parsira i prosleđuje izvornom sistemu (stsremu A).

5. TESTIRANJE

Kako bi se testirao generički adapter, kreirana su dva sistema, kao konzolne aplikacije.

Izvorni sistem (sistem A), učitava *XML* fajl – podatke koje treba poslati. Kreira *SOAP* zahtev, koji potom šalje adapteru. Po završetku procesiranja, adapter novonastale

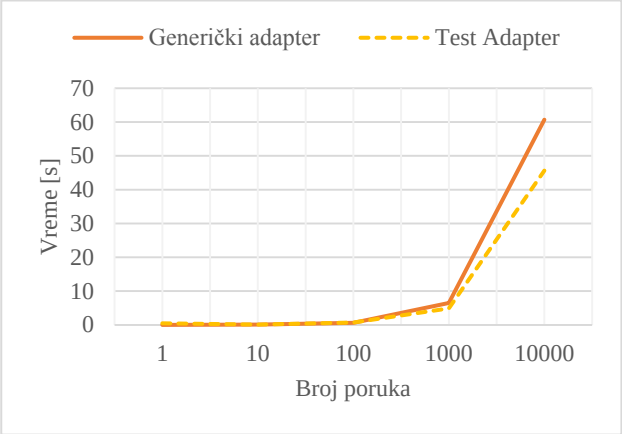
podatke šalje odredišnom sistemu (sistemu B) takođe putem *SOAP* zahteva. Odredišni sistem je konzolna aplikacija koja startuje svoj servis. Na taj način je omogućeno da primi poruku od adaptera. Po prijemu i obradi dobijene poruke, odredišni sistem odgovara generičkom adapteru, koji dobijeni odgovor parsira i šalje izvornom sistemu.

Ukoliko se u bilo kom trenutku obrade zahteva desi izuzetak, izvorni sistem dobija informaciju o tome.

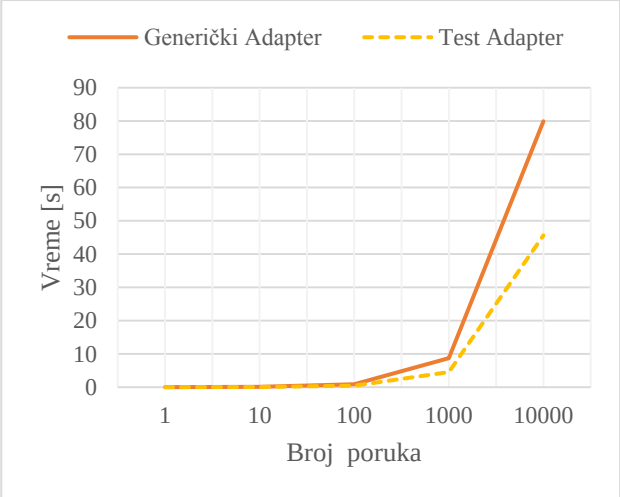
6. PERFORMANSE PROJEKTNOG REŠENJA

Radi testiranja performansi Generičkog adaptera, kreiran je Test adapter u vidu konzolne aplikacije koja startuje *web* servis. Kod Test adaptera ulazna poruka se deserijalizuje u objekat, dok su metode transformacije napisane u kodu i direktno im se pristupa.

Prilikom testiranja oba adaptera, testirana je brzina obrade određenog broja poruka (1,10,100,1000,10000) sa 5 ili 20 transformacija. Na grafiku 1 prikazano je potrebno vreme da oba adaptera obrade određen broj poruka sa 5 transformacija, dok grafik 2 prikazuje potrebno vreme obrade poruka sa 20 transformacija.



Grafik 1 - Potrebno vreme oba adaptera da obrade poruke sa 5 transformacija



Grafik 2 - Potrebno vreme oba adaptera da obrade poruke sa 20 transformacija

Na osnovu izvršenih merenja, izveden je zaključak da je za obradu do 1.000 poruka, nezavisno od broja transformacija, kod oba adaptera, potrebno manje od 10 sekundi, dok je za obradu 10.000 poruka test adapteru potrebno 45 sekundi (bez obzira na broj transformacija), a generičkom adapteru oko 60 sekundi za 5 transformacija, a 80 sekundi za 20 transformacija.

Mereno vreme zavisi od brzine internet konekcije, opterećenja sistema, kao i od same konfiguracije računara. Konfiguracija računara na kojem je vršeno testiranje:

- Processor: Interl(R) Core(TM) i3-3120 CPU @ 3.30GHz
- RAM: 24GB
- System type: Windows 64-bit Operating System, x64 based processor

7. ZAKLJUČAK

Ovim radom opisan je dizajn i implementacija generičkog adaptera za razmenu poruka, u skladu sa *IEC 61968* standardom. Adapter može da posreduje između proizvoljnih sistema, što se podešava kroz konfiguraciju preko *UI*-a. Imajući u vidu da kreirano rešenje podržava komunikaciju koja uključuje samo dva sistema, rešenje bi se moglo proširiti podrškom za komunikaciju koja uključuje više sistema.

8. LITERATURA

[1] EPRI (Electric Power Research Institute), "*Common Information Model Primer*" – Fifth Edition, Palo Alto, SAD, 2019.

[2] Bipin Joshi, "*Beginning XML with C# 7"*, Edition 2, Apress, November 2017.

[3] Priscilla Walmsley, "*Definitive XML Schema*", Prentice Hall, SAD, 2012.

[4] IEC 61968-100 Edition 1.0: "*Application integration at electric utilities – System interfaces for distribution management – Part 100: Implementation Profiles"*, 2013.

[5] G. Woolf, B. Hohpe, "*Enterprise Integration Patterns*", Boston, Addison-Wesley Professional, SAD, 2003.

Kratka biografija:

Đina Stanković rođena je 16.05.1995. godine u Novom Sadu, gde je 2010. godine završila osnovnu školu "Dušan Radović". Gimnaziju

"Svetozar Marković" u Novom Sadu, opšti smer, završila je 2014. godine. Iste godine upisala je osnovne akademske studije na Fakultetu tehničkih nauka u Novom Sadu, smer Elektroenergetski softverski inženjering. Osnovne studije je završila u roku, 2018. godine, nakon čega je, u oktobru iste godine, upisala master akademske studije na Fakultetu tehničkih nauka, smer Primenjeno softversko inženjerstvo.