Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/07BE38Marojevic

DSL I GENERATOR KODA ZA PODRŠKU DIGITALIZACIJE NAFTNIH POLJA

DSL AND CODE GENERATOR FOR DIGITALIZATION OF OIL FIELDS SUPPORT

Ana Marojević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – SOFTVERSKO INŽENJERSTVO I INFORMACIONE TEHNOLOGIJE

Kratak sadržaj – *U ovom radu predstavljeni su domen specifičan jezik za podršku digitalizacije naftnih polja koji obuhvata definisanje podataka, podršku za višejezički korisnički interfejs, sistem za konverziju veličina u različite jedinice i generator osnove projekta u skladu sa takvom definicijom. Jezik je implementiran uz oslonac na Microsoft DSL Tools alate i T4 template za generisanje izlaza.*

Ključne reči: *Generator koda, domen specifičan jezik, Microsoft DSL Tools*

Abstract – *This paper presents a domain-specific language for digitalization of oil fields support that includes defining data, multilingual support, unit conversion system and corresonding code generator. The language is implemented using Microsoft DSL Tools and generator uses T4 template for generating output.*

Keywords: *Code Generator, Domain Specific Language, Microsoft DSL Tools*

1. UVOD

Danas softver kontroliše mnogo aspekata našeg života. Rad programera sastoji se u mapiranju konkretnog problema na rešenje koje razrešava dati problem. Da bi zabeležili takvo rešenje, programeri koriste jezike za programiranje. Programski jezik pruža sloj između programera i postojeće hardverske i/ili softverske platforme na kojoj se program izvodi [1]. Kako bi se olakšao rad, taj međusloj može se premestiti bliže programeru, tj. može se smanjiti jaz između domena problema i domena rešenja. Programski jezici koji imaju za cilj poboljšanje mapiranja problema na rešenje poznati su kao domen specifični jezici (*DSL – Domain Specific Languages*), za razliku od jezika opšte namene (*GPL – General Purpose Languages*).

Modelovanje i definisanje jezika specifičnih za domen u poslednjoj deceniji su znatno napredovali, što je omogućilo da programeri i domenski eksperti mogu da se fokusiraju na konkretne domenske probleme koje rešavaju. Izrada poslovnog sistema podrazumeva implementaciju više stotina međusobno povezanih domenskih entiteta i bar toliko ekranskih formi.

Implementacija na klasičan način bi podrazumevala ručno kodiranje što dovodi do velikih problema ne samo u izgradnji takvog sistema, već i u održavanju i migraciji sistema na nove platforme [2].

Motivacija za ovaj master rad javila se kao odgovor na realan problem jednog softverskog proizvoda u oblasti digitalizacije naftnih polja sa posebnim akcentom na proračun preostalih naftnih rezervi. Rukovodioci projekta su naftni inženjeri koji, iako informatički obrazovani, nemaju mnogo dodira sa programiranjem. S obzirom da su podaci usko vezani za domen rudarstva i naftne industrije, a programeri nemaju dovoljno znanja iz ove oblasti, bilo je neophodno osmisliti rešenje da naftni inženjeri budu ti koji će kreirati opis podataka na dovoljno jednostavan način, a programeri dobiti te podatke u obliku prilagođenom njihovim potrebama. Ideja je da se izbegne definisanje podataka na papiru ili elektronskom obliku od strane inženjera a zatim da programeri isto to pretvaraju u programski kod. Pored toga, već je postojala konvencija za definisanje podataka iz prethodnog softverskog rešenja u vidu tri XML (*eXtensible Markup Language*) dokumenta koju su inženjeri hteli da zadrže, te je bilo potrebno prilagoditi se takvoj strukturi. Aplikacija je tehničke prirode i koristi ogroman broj podataka koji se u naftnoj industriji izražavaju u nekoliko različitih sistema mernih jedinica. Te podatke je potrebno definisati prilikom kreiranja modela, što je bio dodatan motiv za nastanak generatora kako bi se izbegle greške prilikom ručnog pisanja koda. Zamisao je da inženjeri formulišu jedinstveni opis podataka koji se skladišti na jednom mestu, a da se implementira generator koji će kreirati i dobavljati podatke iz baze u obliku prilagođenom aplikaciji koja koristi tu bazu podataka.

2. DEFINICIJA POJMOVA

2.1. Model

Model predstavlja opis, ili specifikaciju sistema i njegovog okruženja kreiranu za određenu namenu. Najčešće je model predstavljen kao kombinacija crteža i teksta. Tekst može biti zadat jezikom za modelovanje ili prirodnim jezikom [3].

Model je apstrakcija nečega što u stvarnosti postoji. Nikada nije preslikana slika realnog sveta, uvek se razlikuje od onoga što modeluje. Često su detalji izostavljeni u modelu. Model nikada neće dati potpuno iste odgovore kao modelovani sistem ali se može očekivati da te razlike budu u projektovanim granicama.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Milosavljević, red. prof.

2.2. Meta-model

Meta-model je model koji definiše apstraktnu sintaksu jezika koji se koristi da opiše model. Apstraktna sintaksa određuje pravila validnosti jezičkih iskaza sa stanovišta njegove strukture bez razmatranja konkretne reprezentacije iskaza (konkretne sintakse) [4]. Svaki meta-model je takođe model i sadrži koncepte domena, njihove međusobne veze i ograničenja.

2.3. Transformacija

Transformacija modela je proces pretvaranja jednog modela u drugi model istog sistema.

Kada je reč o arhitekturi vođenoj modelima, model nezavisan od platforme i ostale informacije se kombinuju transformacijom kako bi se dobio model specifičan za platformu. Zatim se drugom transformacijom model specifičan za platformu prevodi u kod. Ove transformacije su ključne za razvojni proces arhitekture vođene modelima.

Izlaz transformacije se definiše definicijom transformacije. Ona se sastoji od kolekcije pravila transformacije, koje nedvosmisleno specificiraju način na koji se jedan deo modela koristi za kreiranje dela drugog modela.

2.4. Generator koda

Generator koda je alat ili resurs koji generiše određeni kod. Postoji nekoliko pristupa generisanju koda. Prvi je naivan pristup koji se izvodi kombinovanjem fragmenata koda upotrebom komandi print oblika. Kada je većina generisanog koda fiksna i samo određeni delovi koda zavise od modela, tada je preporučljivo koristiti obrađivače šablona (*template engines*) jer oni omogućavaju umetanje varijabilnih delova u ostatak koda. Fiksni delovi generisanog koda su definisani bez izmena, a varijabilni delovi su definisani upotrebom posebnih iskaza šablona. Obrađivač šablona kasnije te iskaze interpretira.

Osnovni razlozi za generisanje koda su [5]:

- Produktivnost – jednom napisan generator koda može se iskoristiti koliko god puta je potrebno. Davanje specifičnih ulaza i pozivanje generatora znatno je brže od pisanja koda ručno, stoga omogućava uštedu vremena
- Jednostavnost – generisanje koda se vrši iz apstraktnog opisa što znači da je izvor poverenje taj opis, a ne kod. Takav opis je obično lakše analizirati i proveriti u poređenju sa celim generisanim kodom
- Prenosivost – isti apstraktni opis se može koristiti za stvaranje različitih vrsta artefakata
- Konzistentnost – generisanjem koda se uvek dobija kod koji se očekuje, naravno osim u slučaju grešaka u generatoru, pa je samim tim i kvalitet koda konzistentan

2.5. Jezici specifični za domen

Jezici opšte namene su dovoljno dobri alati za sve vrste programa, ali nisu specifični za određenu oblast. Čak i u polju opšteg programiranja postoje različiti jezici, od kojih svaki pruža različite prednosti za rešavanje određenih zadataka. Što su zadaci specifičniji, više je razloga za korišćenje jezika specifičnih za domen.

Jezik specifičan za domen je jezik koji je optimizovan za određenu vrstu problema, koji se naziva domen. Zasniva se na apstrakcijama koje su usko usklađene sa domenom za koji je jezik osmišljen.

3. IMPLEMENTACIJA REŠENJA

Sistem se sastoji iz tri samostalne komponente:

- DSL-a za formulisanje opisa podataka i kreiranje XML zapisa iz formirane definicije
- Generatora klasa entiteta iz XML definicije, klasa koje odgovaraju povratnim vrednostima procedura, sloja za komunikaciju sa bazom podatka i podrškom za dependency injection mehanizam
- Konzolne aplikacije za generisanje JSON (*JavaScript Object Notation*) fajlova za podršku za rad sa multi-jezičkim korisničkim interfejsom i sistemom za konverziju veličina u različite sisteme jedinica

3.1. DSL za opis definicije podataka

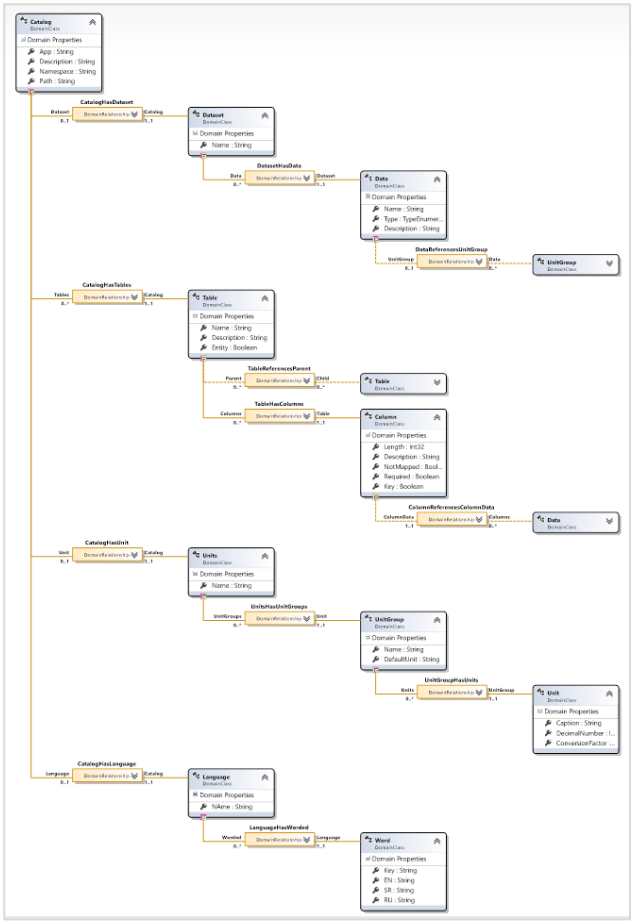
Prva komponenta sistema jeste DSL za opis podataka sistema i generisanje XML datoteka iz opisane definicije. Oslanjajući se na Microsoft DSL Tools, kreiran je meta-model i editor modela.

Definicija DSL-a obuhvata dva aspekta, izgled elemenata modela i informacije koje model nosi. Osnovni pojam u jeziku je katalog (*Catalog*). Katalog predstavlja korensku klasu i objedinjava sve ostale elemente modela. Katalog u sebi sadrži podatke vezane za projekat kao što su naziv aplikacije, njen opis, namespace koji će se koristiti za generisanje klasa i putanju do projekta. Katalog ima četiri direktna podelementa i to su definicija svih podataka (*Dataset*), tabele (*Table*), merne jedinice (*Units*) i jezici (*Language*). Meta-model prikazan je na Slici 1.

Dataset element označava skup svih podataka neophodnih za rad jednog sistema. Svaki podatak koji će se koristiti u aplikaciji mora biti definisan unutar ovog elementa, bilo da je to podatak koji se skladišti u bazu podataka ili virtuelni podatak neophodan za implementaciju nekog dela aplikacije.

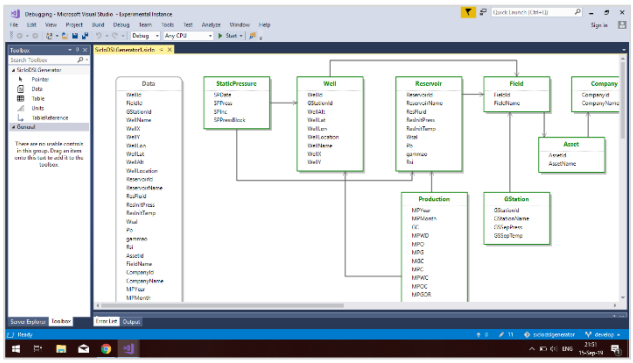
Domain klasa *Table* reprezentuje fizičku tabelu u bazi podataka. U okviru tabele moguće je definisati naziv tabele, opis i označiti da li je potrebno generisati dodatne anotacije za Entity Framework ORM (*Object Relational Mapper*) i sloj aplikacije koji komunicira sa bazom podataka. Svaka tabela se sastoji iz kolona, na nivou kojih se ograničava maksimalna dužina polja, obaveznost polja, da li se radi o virtuelnom podatku i da li je kolona deo ključa tabele. Kolona mora biti vezana sa instancom *Data* klase, koju je potrebno prethodno definisati.

Sistem za konverziju podataka u različite merne jedinice oslanja se na *Units Domain* klasu. Ona reprezentuje skup svih grupa mernih jedinica (npr. temperatura, pritisak, dužina, itd.). Svakom podatku koji predstavlja veličinu se dodeljuje grupa kojoj pripada, koja ima svoj naziv i osnovnu jedinicu u kojoj je vrednost tog podatka zapisana u bazi i na osnovu nje se izračunavaju izvedene jedinice. Svaka od grupa mernih jedinica sadrži listu jedinica (*Unit*) sa svojim nazivom, brojem decimala sa kojim se prikazuje i konverzionim faktorom koji služi za pretvaranje vrednosti podatka iz osnovne jedinice u izvedenu.



Slika 1. *Meta-model jezika za opis definicije podataka*

Language element označava rečnik aplikacije za podršku višejezičkog korisničkog interfejsa. Postoji skup predefinisanih jezika koji će se koristiti u sistemu i za svaku reč ili skup reči koji će se prikazivati na klijentskoj aplikaciji unosi se adekvatan prevod za dati jezik.



Slika 2 – *Primer kreiranja modela*

Unutar domen specifičnog jezika moguće je definisati ograničenja radi provere ispravnosti modela. Microsoft DSL Tools ima ugrađenu validaciju ispravnosti kreiranja elemenata i veza između njih u skladu sa definisanom semantikom jezika. Parcijalne klase u C#-u omogućuju definisanje iste klase na više mesta. Ova tehnika pruža mogućnost da se generisani kod odvoji od ručno pisanog koda. Dodatna ograničenja se definišu dodavanjem metoda validacije u klase domena ili klase relacija. Kada se pokrene provera, bilo od strane korisnika ili pod kontrolom programa, određene ili sve validacione metode se izvršavaju. Svaka metoda se primenjuje na svaku instancu klase, a u svakoj klasi može biti nekoliko metoda provere. Metoda validacije izveštava o svim greškama i nepravilnostima modela.

3.2. Generator koda

Generator osnove projekta iz XML definicije je Windows Forms aplikacija koja generiše sve entitete iz određene baze podataka i pruža temelj za izradu projekta. Generator koristi tri XML dokumenta na osnovu kojih generiše željene izlaze. *Catalog*, *Units* i *Dictionary* dokumenti nalaze se u bazi sačuvani kao binarni fajlovi.

Aplikacija za koju se generiše osnova, za sada i jedina, je ASP.NET veb aplikacija za digitalizaciju naftnih polja. ASP.NET Web API je radni okvir koji olakšava implementaciju HTTP servisa. Predstavlja idealnu platformu za izgradnju RESTful aplikacija u .NET Framework-u.

Sa Entity Framework-om pristup podacima se vrši pomoću modela. Model se sastoji od klasa entiteta i kontekstnog objekta koji predstavlja sesiju sa bazom podataka i omogućava kreiranje upita i čuvanje podataka. Pored svojstava klase i anotacija, neophodnih za rad Entity Framework-a, generiše se atribut „*Property*“ iznad svojstava koja se mogu izraziti u različitim mernim jedinicama.

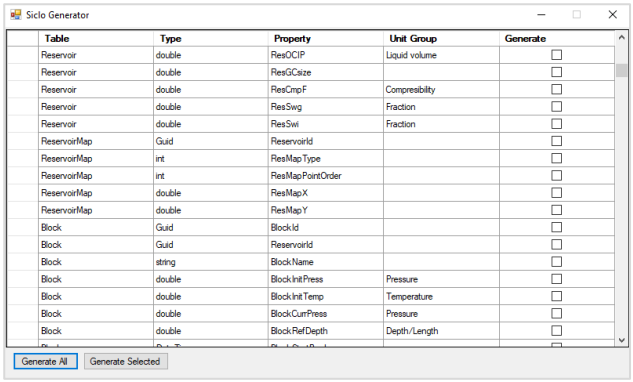
DbContext klasa je sastavni deo Entity Framework okvira. Instanca *DbContext* klase predstavlja most između entiteta i baze podataka i omogućava upravljanje konekcijom, konfigurisanje modela i relacija, olakšava upite, skladištenje, praćenje promena i mnoge druge stvari vezane za rad sa bazom podataka. Generiše se klasa koja nasleđuje *DbContext* i definiše kolekciju *DbSet*-ova koji reprezentuju kolekcije entiteta u kontekstu. *OnModelCreating()* metoda ima instancu *ModelBuilder* klase kao parametar i poziva se od strane framework-a kada se kontekst prvi put kreira kako bi se izgradio model i njegova mapiranja u memoriji.

Implementacija *Repository pattern*-a pomaže da izolujemo pristup podacima od ostatka aplikacije. Repozitorijum klasa koristi Entity Framework-ovu kontekstnu klasu podataka za obavljanje CRUD (*Create, read , update and delete*) operacija. Generator podržava generisanje ovih tipičnih metoda upravljanja podacima, a to su metode *FindAll(), Create(), Update() i Delete().*

Dependency injection (DI) je tehnika za razvijanje aplikacije sa što većim stepenom nezavisnosti. Nezavisnosti u smislu da svaki modul aplikacije treba da bude jedinstven i da ne zavisi od ostalih modula. Generatorom je omogućena osnova i olakšana upotreba *dependency injection* mehanizma, koristeći Simple Injector *open source* biblioteku za .NET.

Koristeći fiksne blokove teksta i umetanjem varijabilnih delova iz modela kreiraju se *.cs* fajlovi na izabranoj putanji. S obzirom da se u aplikaciji za koju je generator prvobitno pravljen većina podataka dobija pozivima uskladištenih SQL (*Structured Query Language*) procedura, koje ne vraćaju entitetske klase, generator je proširen delom za generisanje klasa sa proizvoljnim svojstvima, koja u ovom slučaju odgovaraju povratnim vrednostima procedura

Generatorom je omogućeno brže i efikasnije kreiranje temelja projekta i izbegavanje grešaka prilikom ručno pisanog koda, što i jeste osnovna prednost svakog generatora. Na slici 3 prikazan je izgled aplikacije za generisanje podataka.



Slika 3. *Izgled aplikacije za generisanje podataka na osnovu XML zapisa*

3.3. Aplikacija za generisanje JSON dokumenata za merne jedinice i višejezičku podršku

Generator JSON fajlova za implementaciju konverzije veličina u merne jedinice i podršku višejezičkog korisničkog interfejsa je konzolna aplikacija koja ima za cilj da izgeneriše JSON fajlove sa neophodnim podacima za prethodno navedene funkcionalnosti.

Sve veličine u aplikaciji mogu izraziti u *API* i *SI* sistemu jedinica. Pored toga, postoji mogućnost kreiranja sopstvenog sistema jedinica, odabirom željene jedinice za svaku grupu jedinica, ali takav scenario nije bilo moguće obuhvatiti generatorom.

Prvi korak pri generisanju jeste čitanje potrebnih XML dokumenata iz baze podataka. XML dokumenti se nalaze upisani kao binarni fajlovi u jednoj od tabela. Upravljanje XML dokumentom, njegovo učitavanje i pronalaženje elementa implementirana je uz oslonac na *XmlDocument* klasu .NET Framework-a. Nakon dobijanja neophodnih informacija kreiraju se JSON fajlovi koji se koriste za konverziju veličina u različite jedinice i podršku za multi-jezički korisnički interfejs.

4. ZAKLJUČAK

Zadatak ovog rada bio je da prikaže jezik za definisanje opisa podataka, sistem za konverziju veličina iz osnovnih u izvedene jedinice, podršku za višejezički korisnički interfejs i generator koji generiše osnovu projekta u skladu sa ovim definicijama. Iako je nastao kao rešenje za usko specificiran domen naftne industrije, dovoljno je generalizovan da bude upotrebljiv za bilo koju drugu aplikaciju slične arhitekture.

Omogućava definisanje opisa podataka, bez neophodnog programerskog znanja, obezbeđuje skladište tih podataka i pruža izgenerisanu osnovu projekta aplikacije koja koristi takvu definiciju podataka. Generator je zamišljen kao nezavisan međusloj između baze podataka i aplikacija koje je koriste. S obzirom da je projekat za koji je pre-vashodno implementiran velikog obima i da će se sastojati iz više samostalnih sistema, u budućnosti će biti proširen tako da obuhvata generisanje osnova aplikacija u drugim programskim jezicima i različitim platformama.

Prvobitna ideja i jeste da generator bude sprega između inženjera koji definišu model i programera koji treba na što brži i efikasniji način da dobiju model i osnovu projekta u obliku prilagođenom svojim potrebama. Budući da aplikacija sadrži pozamašan broj tabela, pored kojih se kontinuirano kreiraju nove, generator znatno ubrzava proces kreiranja odgovarajućih klasa entiteta i sloja za komunikaciju sa bazom podataka. Takođe otklanja mogućnost pravljenja ručno pisanih grešaka prilikom definisanja svojstava klasa i primenjivanje atributa koji određuju grupu jedinica kojima pripadaju svojstva, mapiranjem podataka iz XML dokumenta.

Dalja unapređivanja ovog alata mogla bi otpočeti proširivanjem generatora za podršku kreiranja osnove projekta na drugim programskim jezicima. Generator je trenutno samostalna aplikacija koja se mora pokrenuti kako bi se stekao željeni izlaz. U budućnosti bi se mogao napraviti okidač (*trigger*) kada dođe do izmene u bazi podataka koji će automatski pokrenuti generator i proizvesti novonastale ili modifikovane klase.

Što se tiče jezika specifičnog za domen koji trenutno rezultat opisa definicije podataka smešta u XML datoteku, trebalo bi ga proširiti opcijom za automatsko kreiranje i migriranje baze podataka.

5. LITERATURA

[1] U. Tikhonova, Engineering the Dynamic Semantics of Domain Specific Languages, 2017.

[2] I. Dejanović, Metamodel, editor modela i generator poslovnih aplikacija - Magistarska teza, Novi Sad, 2008.

[3] J. M. a. J. Mukerji, MDA Guide Version 1.0.1, 2003.

[4] I. Dejanović, Prezentacije sa predmeta Jezici specifični za domen.

[5] F. Tomassetti, A Guide to Code Generation, 2008.

Kratka biografija:

Ana Marojević rođena je 07.10.1994. u Novom Sadu. Fakultet tehničkih nauka upisuje 2013. godine, odsek Softversko inženjerstvo i informacione tehnologije. Diplomski rad odbranila je 2017. godine. Iste godine upisuje master akademske studije na Fakultetu tehničkih nauka.