|  |  |
| --- | --- |
|  | Zbornik radova Fakulteta tehničkih nauka, Novi Sad |

**UDK: 004.42:004.738.12**

**DOI:** [**https://doi.org/10.24867/28BE40Panic**](https://doi.org/10.24867/28BE40Panic)

**DIZAJN I IMPLEMENTACIJA JEZIKA SPECIFIČNOG ZA DOMEN LOGIČKIH ENIGMATSKIH IGARA**

**THE DESIGN AND IMPLEMENTATION OF DOMAIN-SPECIFIC LANGUAGE FOR LOGIC PUZZLE GAMES**

Sara Panić, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

**Kratak sadržaj –**  *Cilj rada jeste istraživanje, projekto­vanje i razvijanje jezika koji će omogućiti lakše i brže predstavljanje i analizu različitih vrsta logičkih enigmat­skih igara. Nakon toga implementiran je prototip sistema za implementaciju i izvršavanje logičkih enigmatskih iga­ra koristeći razvijeni jezik što uključuje razvoj osnovnih modula za rad sa zagonetkama i njihovu specifikaciju i interpretaciuju.*

**Ključne reči:** *JSD, textX, gramatika, enigmatika, logičke igre*

**Abstract** – *The aim of the work is to research, design, and develop a language that will enable easier and faster representation and analysis of various types of logical puzzle games. After that, a prototype system for imple­menting and executing logical puzzle games was imple­mented using the developed language, which includes the development of basic modules for working with puzzles and their specification and interpretation.*

**Keywords:** *DSL, textX, grammar, enigma, logic games*

**1. UVOD**

Enigmatika obuhvata širok spektar logičkih igara koje za­htevaju rešavanje zagonetki ili mozgalica. Ove igre pred­stavljaju zanimljiv domen istraživanja u računarstvu i veš­tačkoj inteligenciji. Ono što ih čini izazovom u ovoj sferi jeste kombinatorika i logička analiza koje su neophodne da bi se igra definisala i rešila, često uz kombinaciju spe­cifičnih strategija i tehnika. Računarske verzije ovih igara omogućavaju korisnicima da ih rešavaju na svojim uređa­jima, često uz podršku naprednih algoritama za generisa­nje, interpretaciju i rešavanje istih.

S obzirom na to, ovaj rad se fokusira na razvoj jezika za opisivanje pravila i karakteristika logičkih igara enigmatike. Korišćen je alat textX za definisanje gramatike jezika, što je omogućilo preciznu specifikaciju igara kroz formalne strukture jezi­ka. Praktični deo za cilj ima primenu teorijskog znanja i rezultata istraživanja i predstavlja projekat koji se oslanja na implementaciju veb aplikacije koja koristi definisani jezik za generisanje i reprezentaciju logičkih igara. Apli­kacija se sastoji iz korisničkog interfejsa koji omogućava unos novih primera igara, ali i rešavanje već unetih.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NAPOMENA:**

**Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Igor Dejanović.**

Serverski deo koristi parser zasnovan na textX-u za interpretaciju i generisanje igara na osnovu unetih specifikacija.

**2. KLASIFIKACIJA LOGIČKIH IGARA**

Enigmatiku definišemo kao veštinu sastavljanja i rešava­nja zagonetki. Pored toga što se tretira kao razbibriga i zabava, ima veliku ulogu u testiranju inteligencije, proveri i sticanju opšteg znanja. Često se naziva i "gimnastikom uma i igrom duha" [1]. Razlikujemo više grupa enigma­tike: mrežaste zagonetke, zagonetke sa premetanjem slova, logičke igre, rebusi, kriptogrami, matematičke, likovne zagonetke i još mnogo drugih.

Sudoku (eng. *sudoku*) - Sudoku predstavlja popularnu zagonetku koja zahteva veštinu, strategiju i zaključivanje iz pretpostavki sa složenim uslovima [2]. Igra se na tablici od najčešće 9x9 polja podeljenih na 3x3 regiona, gde igrač mora da popuni sva polja ciframa od 1 do 9 tako da svaki red, kolona i svaki 3x3 region sadrže sve cifre bez ponavljanja. Postoje različite varijacije sudoku igre koje uključuju različite rasporede startnih cifara ili dodatna pravila koja usložnjavaju igru (na primer: dijagonalni, geometrijski, parni-neparni, samuraji itd.). Takođe, možemo ih razlikovati i po kompleksnosti, koju najčešće definišemo brojem cifara koje su inicijalno prikazane na poljima. Pravilo koje mora biti zadovoljeno da bi sudoku bio rešiv, bez obzira na kompleksnost, jeste da je inicijalno dato minimum 17 brojeva [3].

Ukrštene reči (eng. *crosswords*) - Ukrštene reči pred­stavljaju zabavnu i izazovnu mrežastu zagonetku u kojoj igrač treba da pronađe i ukrsti skrivene reči u mreži slova. Ova igra nije samo popularna za rekreaciju, već i za razvoj vokabulara i veštine logičkog razmišljanja [4]. Zahteva od igrača poznavanje opšte kulture i informisa­nosti i široko znanje iz različitih oblasti. Reči se unose horizontalno ili vertikalno i ukrštaju se najmanje na jednom mestu, deljenjem jednog slova. Po sadržaju ih možemo podeliti na klasične, tematske, simetrične, kriptične i tako dalje, a po strukturi na pravilne (pravougaone) i one koje su nepravilnog oblika.

Osmosmerka (eng. *words search*) – Jednostavna ali efikasna igra koja pruža zadovoljstvo svim uzrastima. Koristi se u edukativne svrhe kako bi pomogla razvijanju jezičkih veština, proširivanju rečnika i unapređenju sposobnosti koncentracije. Za razliku od prethodne dve igre, gde se u prazna polja unose cifre i slova, postavka osmostazne predstavlja već popunjenu mrežu slova. Cilj igre je da igrači pažljivo pregledaju slova i pronađu sve zadate reči koje su unutar nje skrivene. Reči mogu biti, kako i sam naziv igre govori, napisane u svih osam smerova.

**3. JEZICI SPECIFIČNI ZA DOMEN**

**3.1. Definicija**

Jezici specifični za domen su namenjeni ograničenoj oblasti primene. Sastoje se od pojmova koji su relevantni za određeni domen, omogućavajući konciznije i razumljivije opise rešenja za te probleme. Ključna prednost JSD je što omogućava stručnjacima iz određene oblasti da lakše razumeju i učestvuju u razvoju rešenja, čak i ako nisu programeri. Sintaksa ovih jezika je često prilagođena konkretnim konceptima iz domena, što olakšava komunikaciju između inženjera i stručnjaka iz tog područja. JSD su poznati kao "mali jezici" (eng. little languages), a uspešnim se smatra onaj koji je fokusiran na uzak, dobro definisan domen i efikasno ga pokriva.

**3.2. Prednosti i mane**

Ukoliko je JSD dobro definisan, prednosti koje pruža su značajne [5]: uticaj na produktivnost programera, uticaj na kvalitet softvera, duži životni vek aplikacije i ovi jezici su samodokumentujući.

Nasuprot svim prednostima, postoje i izazovi na koje se nailazi pri korišćenju JSD-a, kao što su [6]: cena razvoja, dizajn jezika, održavanje i evolucija, zavisnost od alata i takozvani JSD pakao (nagomilavanje novih umesto korišćenja ili proširivanja već postojećih jezika).

**3.3. Faze razvoja i elementi**

Prilikom izgradnje svaki proizvod prolazi kroz određene faze razvoja. Proces razvoja jezika specifičnog za domen, prema knjizi "Domain-Specific Languages" autora Martina Faulera, može se opisati sledećim fazama [5]: istraživanje domena, identifikacija ključnih apstrakcija, dizajn jezičkih apstrakcija, implementacija jezika, testiranje i iterativni razvoj, održavanje i podrška.

Svaki softverski jezik, bez obzira da li je opšte namene ili specifičan za domen sastoji se iz apstraktne sintakse, jedne ili više konkretne sintakse i semantike [7].

Apstraktna sintaksa predstavlja strukturu podataka tipa stablo koje sadrži koncepte domena, njihove osobine i međusobne relacije bez detalja notacije. Definiše struktu­ru i određuje pravila validnosti jezičkih iskaza. Apstrak­tno sintaksno stablo je usmereno, labelirano stablo gde čvorovi predstavljaju instance koncepta apstraktne sintak­se. Jezici za definisanje apstraktnih sintaksi jezika se u domenu modelovanja nazivaju meta-metamodeli, odno­sno meta-metamodel je apstraktna sintaksa takvog jezika.

Konkretna sintaksa predstavlja vizuelnu reprezentaciju jezika putem koje korisnik obavlja interakciju sa računarom, odnosno može se reći da je interfejs između jezika i korisnika. Definiše izgled iskaza na nekom jeziku i može biti tekstualna, grafička, tabelarna, tipa stabla, hibridna itd. Za jedan jezik dovoljna je jedna konkretna sintaksa ali može postojati i više njih.

Dok je konkretna sintaksa interfejs jezika ka korisniku, apstraktna se ponaša kao API za pristup programima od strane alata za obradu - validatori, transformacije i generatori koda.

Semantika je poslednji element nekog jezika i ona određuje smisao napisanih jezičkih iskaza. Definisanje semantike može se vršiti pomoću više tehnika, ali je u praksi najzastupljenije prevođenje na jezik koji već ima definisanu semantiku putem prevodioca ili interpretira. Prevodjenje se obavlja kompajliranjem, odnosno generisanjem koda najčešće na jezike opšte namene i može da ide sve do nivoa mašinskog jezika definisanog u hardveru računara.

**3.4. Model i meta-model**

Model i meta-model su ključni koncepti u razvoju jezika specifičnih za domen (JSD) i njihovoj implementaciji. Model se često definiše kao opis ili reprezentacija sistema, dok semantički model pruža specifikaciju pojmova unutar jezika, nezavisno od same sintakse jezika.

Meta-model, s druge strane, predstavlja formalni opis elemenata modela i načina njihove povezanosti. U kontekstu jezika specifičnih za domen, meta-model opisuje pojmove iz određenog domena, njihova ograničenja i relacije među njima.

Često se meta-model koristi za definisanje strukture samog jezika, a termin "apstraktna sintaksa" se koristi kao sinonim za meta-model, suprotstavljajući se konkrentoj sintaksi koja opisuje specifičnu reprezentaciju jezika (tekstualnu, grafičku, tabelarnu, itd.).

Formalno definisan meta-model, zajedno sa definicijom konkretne sintakse, omogućava automatsko generisanje parsera ili editora jezika specifičnih za domen.

**3.5. Parser**

Parseri predstavljaju suštinske komponente u razvoju programskih jezika i interpretatora. Oni igraju ključnu ulogu u obradi ulaznih nizova znakova i simbola u strukture koje su pogodne za dalju analizu i izvršavanje. Parseri se koriste za konverziju ulaznog teksta u apstraktne strukture podataka, kao što su Sintaksno stablo (*CST*) ili Stablo apstraktne sintakse (*AST*), na osnovu formalno definisanih gramatika programskog jezika.

Proces parsiranja obično se sastoji iz tri osnovna koraka: leksička analiza (ulazni tekst se razlaže na tokene ili lekseme), sintaksna analiza (proverava i validira redosled tokena u skladu sa definisanom gramatikom i kreira stablo parsiranja) i semantička analiza (dodeljuje značenje parsiranom ulazu, prevodeći ga u oblik razumljiv računaru i izvršavajući semantičke akcije nad čvorovima stabla parsiranja) [8].

**4. TEXTX**

**4.1. Definicija**

TextX je alat koji služi za definisanje JSD-a u Python okruženju. Omogućava korisnicima da definišu svoje tekstualne jezike na jednostavan način, čime mogu da kreiraju potpuno nove ili podrže već postojeće tekstualne jezike ili formate datoteka.

Ključne funkcionalnosti textX-a uključuju: automatsko generisanje meta-modela i parsera na osnovu jednog opisa gramatike, automatsko povezivanje referenci između entiteta jezika (engl. *linking*), održavanje odnosa roditelj-dete u modelu, kontrolu parsera u pogledu obrade razma­ka i ključnih reči, mogućnost postprocesiranja modela, podršku za modularizaciju gramatike, kombinovanje više meta-modela (engl. *multi-meta-model*), vizualizaciju kako meta-modela tako i parsiranih, modela radi lakšeg razumevanja i analize strukture jezika [9].

**4.2. Gramatika**

Sintaksa jezika i meta-model definisani su textX gramatikom. Gramatika predstavlja formalan sistem za opisivanje sintakse ili jezičke strukture, odnosno opisuje pravila koja određuju kako se ispravne rečenice ili izrazi mogu formirati iz komponenti jezika.

Osnovni elementi gramatike uključuju terminalne simbole (osnovne jedinice jezika, koje ne mogu biti dalje razložene) i neterminalne simbole (razvijaju se u skladu sa pravilima gramatike).

U kontekstu računarskih nauka, gramatika opisuje kako se tokeni (simboli programskog jezika) mogu kombinovati da bi se dobili validni programski izrazi ili naredbe. Često se predstavlja u Bakus-Naurovoj formi (*BNF*) ili sličnim notacijama koje opisuju pravila za razvijanje neterminala u termine.

U slučaju textX-a, koristi se proširena Bakus-Naurova forma (*Extended Backus–Naur Form - EBNF*) gde je moguće navesti pravilo u više redova.

Osim toga, EBNF uključuje mehanizme za unapređenje, definisanje broja ponavljanja, isključivanje alternativa, komentare itd.

Sledećih pet pravila čine osnovu textX tipova i automatski se konvertuju u Python tipove pri instanciranju objekata: ID, INT, FLOAT, BOOL, STRING.

Pored njih podržani su: pravilo prepoznavanja (string ili regex), sekvence (omogućavaju ulančane izraze kojima se precizira očekivani redosled), uređeni izbor, opcioni izraz, ponavaljanja (nula ili više, jedan ili više, grupe proizvoljnog redosleda), reference na druga pravila, dodele, sintatički predikati (pozitivni & i negativni !) i drugi.

**5. OPIS REŠENJA**

**5.1. Jezik i gramatika**

Nakon istraživanja oblasti jezika specifičnih za domen i textX-a kreiran je jezik za enigmatiku sa ekstenzijom *.lg* (*logic game*). Gramatika jezika započinje *Enigma* pravilom (slika 5.1.) koje služi da objedini sve tipove enigmi u jedan objekat.



Slika 5.1. Pravila *Enigma* i *EnigmaType*

Pravilo je jednostavno, sastoji se od polja *details* kojem se dodeljuje pravilo *EnigmaType* - uređeni izbor pravila i navodi da polje detalji mora biti definisano nekim od sledećih pravila: *Sudoku*, *Crosswords* ili *WordSearch*. Na slici 5.2. prikazan je primer gramatike za *WordSearch* entitet.



Slika 5.2. Primer gramatike za entitet *WordSearch*

**5.2. Parser**

Pomoću gramatike koja se prosleđuje funkciji *metamodel\_from\_file* iz Python paketa *textx.metamodel* kreira se meta-model ukoliko gramatika nema grešaka, u suprotnom se vraća izuzetak. Na osnovu meta-modela, textX može da izgradi parser koji čita i interpretira svaki deo ulaznog teksta u saglasnosti sa specifikacijama jezika. Takođe, na ovom nivou registruju se procesori koji će izvršiti dodatne validacije i modifikacije nad pročitanim objektima. U sklopu ovog rešenja korišćen je procesor objekata koji se poziva u paraleli sa parsiranjem nad svakim uspešno pročitanim objektom. Koristeći opcije procesora, izvršavaju se validacije nad sudokvom i osmosmerkom koje nije izvodljivo obaviti na nivou gramatike: provera da li sudoku ima 81 karakter, da li je inicijalno otkriveno minimum 17 cifara kako bi sudoku bio rešiv, ispitivanje da li postoje ponovljene cifre u istom redu, koloni ili kvadratu, provera da li je u sklopu osmosmerke izabrana opcija *auto\_generate* (generisanje mreže slova od strane sistema) i ukoliko jeste da li su unete dimenzije mreže.

**5.3. Server**

Server obezbeđuje infrastrukturu za dalju obradu i slanje podataka na korisnični interfejs. Razvijen je koristeći Python jezik i njemu kompatibilne Uvicorn veb server i FastAPI radno okruženje.

FastAPI pruža moderan okvir za razvoj API-ja, baziran na korišćenju i unapređenju mogućnosti tipiziranih nago­veštaja koje su uvedene u novijim verzijama Python-a, počevši od verzije 3.5 [10]. Ovim je omogućeno defini­sanje, validacija, automatska serializacija i deserija­liza­cija, kao i korišćenje tipova podataka u svojim opera­cija­ma API-ja. Odlikuje se brzinom, efikasnošću i jedno­stavnim definisanjem veb ruta i operacija korišćenjem Python dekoratora.

Uvicorn je moćan veb server posebno dizajniran za Python aplikacije koje koriste asinhronu obradu i bazirane su na ASGI (*async server gateway interface*) specifikaciji [11]. Izuzetno je brz i efikasan, što ga čini idealnim izborom za pokretanje modernih veb aplikacija. Podržava asinhrono izvršavanje koda i više radnih procesa. Funkcioniše na bazi osluškivanja soketa, zatim prima konekciju, procesuira je i prosleđuje.

**5.4. Klijentska aplikacija**

Klijentska aplikacija kreirana je u obliku SPA aplikacije pomoću React-a i predstavlja alat putem kog korisnici mogu da uvezu svoje primere enigmatskih igara (slika 5.3.), a zatim njih ili primere koje su drugi uvezli, rešavaju (slika 5.4.).

React je biblioteka za razvoj korisničkog interfejsa (engl. *UI*) u JavaScript programskom jeziku. Dizajnirana je za izgradnju efikasnih, lakih za održavanje i visoko perfor­mantnih veb aplikacija. Koristi deklarativni pristup za opisivanje komponenti koje se mogu renderovati u odgovoru na promene stanja aplikacije.

Slika 5.3. Dijalog za uvoz novih primera



Slika 5.4. Primer igre – sudoku

**6. ZAKLJUČAK**

Primenom jezika specifičnog za domen na datom primeru dokazana je jedna od osnovnih prednosti ovih jezika, a to je prilagodljivost korisnicima koji ne poseduju prethodno tehničko znanje. Uočeni nedostaci postojećih rešenja bili su motivacija za razvoj sažete i jednostavne sintakse, pogodne za upotrebu osobama iz različitih sfera. Čak i kao takvo, rešenje ostavlja dosta prostora za unapređenje i dalji razvoj. Neke od ideja za dalji razvoj bi mogle bitit:

* Podrška za veći broj igara, odnosno proširivanje jezika da obezbedi generisanje većeg opsega enigmatskih igara.
* Omogućiti dodatne tipove već podržanih igara: napred­nije verzije sudokua, ukrštene reči nepravilnog oblika, ukrštene reči sa zagonetnom rečju ili tematske osmo­stazne.
* Unaprediti interakciju krajnjeg korisnika sa koris­nič­kim interfejsom, na primer korišćenje biblioteke
* OpenLayers koja bi vizuelnim i interaktivnim elementima omogućila bolje korisničko iskustvo.
* Implementacija baze podataka omogućila bi značajnu optimizaciju rešenja.
* Nadogradnja sistema tako da podrži registraciju korisnika, praćenje njihovih rezultata i uvođenje vremenskog ograničenja.
* Uvođenje multiplayer režima gde bi igrači mogli da rešavaju enigme zajedno ili se takmiče jedni protiv drugih u sklopu određenih turnira.
* Integracija sa određenim softverima koji bi omogućili obrazovni aspekt kao što su logičko razmišljanje, matematički koncepti ili drugi obrazovni ciljevi.

**7. LITERATURA**

[1] Wikipedia – enigmatika, https://sr.wikipedia.org/sr/Енигматика, na mreži, pristup: april 2024.

[2] Rationality - Constraints and Contexts, T.-W. Hung and T.J. Lane , Academic Press, 2016

[3] Sudoku Space, https://sudoku-space.com, na mreži, pristup: april 2024.

[4] Crossword Puzzles as a Learning Tool For Vocabulary Development, Wiwat Orawiwatnakul, Language Institute, Bangkok University, 2017

[5] M. Fowler, Domain-specific languages, Addison-Wesley, 2010

[6] M. Völter, „DSL Engineering: Designing, Implementing and Using Domain-Specific Languages”, 2013.

[7] I.Dejanović – Jezici specifični za domen, - „Uvod“ https://www.igordejanovic.net/courses/jsd/01-uvod/, na mreži, pristup: april 2024.

[8] I.Dejanović – Jezici specifični za domen – „Tekstualne sintakse“, https://www.igordejanovic.net/courses/jsd/03-tekstualne-sintakse/ ,na mreži, pristup: april 2024.

[9] I.Dejanović, R. Vaderna, G. Milosavljević, Ž. Vuković, „TextX: A Python tool for Domain-Specific Languages implementation”, Knowledge-Based Systems, 2017

[10] FastAPI, https://fastapi.tiangolo.com/,na mreži, pristup: april 2024

[11] Uvicorn, https://www.uvicorn.org/,na mreži, pristup: april 2024

**Kratka biografija:**

Sara Panić rođena je 21. marta 1997. godine u Štutgartu, u Nemačkoj. Osnovnu školu "Vuk Karadžić" završila je sa Vukovom diplomom, u Zrenjaninu 2012. godine. Iste godine upisuje, a 2016. završava opšti smer u "Zrenjaninskoj gimnaziji". Na jesen postaje student Fakulteta tehničkih nauka na Univerzitetu u Novom Sadu, smer Primenjeno softversko inženjerstvo. Diplomirala je u septembru 2020. godine nakon čega obrazovanje nastavlja na master akademskim studijama Primenjene računarske nauke i informatika, smer Softversko inženjerstvo. Paralelno sa pohađanjem master studija, od marta 2021. godine, godinu dana radila je kao Asistent - Saradnik u nastavi na osnovnim studijama Primenjeno softversko inženjerstvo.