Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/33BE30Vulin

JEZIK ZA OPIS PRAVILA ZA IGRE SA KARTAMA

A LANGUAGE FOR DESCRIBING CARD GAME RULES

Ana Vulin, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu je predstavljen dizajn i implementacija jezika specifičnog za domen (DSL) namenjenog opisivanju pravila za igre sa kartama. Sistem obuhvata textX parser, interpreter u Python-u, kao i veb klijent sa podrškom za interaktivnu vizuelizaciju. Cilj je omogućiti lako kreiranje igri sa kartama bez potrebe za programerskim znanjem.*

Ključne reči: *DSL, textX, Python, WebSocket, vizualizacija, igre sa kartama*

Abstract – *The paper presents the design and implementation of a domain-specific language (DSL) intended for describing the rules of card games. The system includes a textX parser, a Python-based interpreter, and a web client with support for interactive visualization. The goal is to enable easy creation of card games without the need for programming knowledge.*

Keywords: *DSL, textX, Python, WebSocket, visualization card games*

1. UVOD

Kartaške igre predstavljaju jedan od najrasprostranjenijih oblika društvene zabave, ali i pogodan model za proučavanje i implementaciju različitih pravila, strategija i interakcija. Njihova struktura – sa jasno definisanim fazama, pravilima i uslovima pobede – čini ih idealnim kandidatom za formalizaciju kroz *DSL*. Cilj ovog rada je razvoj i implementacija *DSL* rešenja koje omogućava jednostavno, fleksibilno i proširivo definisanje i pokretanje različitih kartaških igara.

Predloženo rešenje – *CardGame DSL* – omogućava korisnicima da tekstualno opišu pravila igre koristeći specijalizovanu DSL gramatiku, a sistem automatski generiše i izvršava logiku igre, čime se ubrzava razvoj i olakšava učešće osobama bez programerskog znanja.

Sistem koristi *textX* za definisanje i parsiranje *DSL* sintakse, dok je izvršna logika realizovana u *Python-u* kroz specijalizovan interpreter. Klijentska aplikacija je razvijena u *JavaScript-u* sa *HTML* i *CSS* podrškom, a komunikacija klijent–server odvija se u realnom vremenu putem *WebSocket-a*, čime je omogućena modularnost i ponovna upotreba sistema.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Igor Dejanović, red. prof.

2. TEORIJSKE OSNOVE

2.1. Jezici specifični za domen (*DSL*)

Jezik specifičan za domen [1] je jezik napravljen za rešavanje problema u određenom domenu. Na primer, *HTML* je *DSL* za opisivanje veb stranica, a *SQL DSL* za rad sa bazama podataka. Nasuprot DSL-u se nalazi jezik opšte namene (*GPL – General Purpose Language*) koji ima široku primenu u različitim oblastima razvoja softvera (*Python*, *Java*,..)

2.1.1. Apstraktna i konkretna sintaksa

Apstraktna sintaksa [2] – logička struktura koda bez detalja koji nisu bitni za razumevanje. Najčešće se koristi u vidu stable apstraktne sintakse (AST – Abstract Syntax Tree).

Konkretna sintaksa – način na koji korisnik unosi programski kod ili model. Može biti tekstutualna, grafička, tabelarna.

U okviru tekstualne konkretne sintakse definiše se gramatika. Predstavlja formalizovani skup pravila koji definiše dopustive nizove karaktera i njihove međusobne odnose.

2.1.2. Parsiranje

Proces parsiranja uzima tekst u konkretnoj sintaksi i pretvara ga u strukturu u apstraktnoj sintaksi (AST).

2.1.3. Interpretacija i kompajliranje

Postoje dva glavna načina izvršavanja koda ili naredbi – interpretacija i kompajliranje/generisanje koda. Interpretacija – izvorni kod ili neka njegova reprezentacija (npr. *AST*) se direktno čita i izvršava od strane interpretera. Kompajliranje ili generisanje koda – proces stvaranja koda napisanog u nekom jeziku u niz instrukcija ili mašinski kod koji procesor može direktno izvršiti.

2.2. Igre sa kartama

Igre sa kartama [3] predstavljaju raznovrstan oblik društvenih igara i sve više se koriste u obrazovanju. Mogu se klasifikovati prema svrsi i pravilima, npr. igre na sreću, strateške, pamćenja i opažanja, socijalne i edukativne igre.

3. SPECIFIKACIJA SISTEMA

3.1. Jezici specifični za domen (*DSL*)

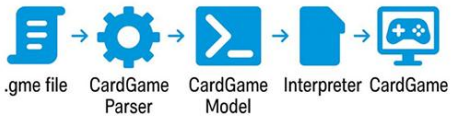
Glavni cilj sistema jeste da omogući lako i brzo kreiranje kartaške igre koristeći unapred definisan skup pravila i to na način koji je intuitivan i pristupačan i osobama bez programerskog znanja.

3.2. Postojeća rešenja

Postojeća rešenja kao što su biblioteka Cardamom.js [4] i Construct 3 [5] podržavaju razvoj kartaških igara. Cardamom.js omogućava rad sa kartama. Špilovima i rukama kroz funkcije za kreiranje, mešanje i upravljanje. Construct 3 pruža vizuelni, objektno orijentisan sistem za razvoj 2D igara bez pisanja koda.

3.3. Arhitektura obrade i interpretacije *DSL-a*

Jezik *CardGame* je eksterni *DSL*. Osnovna komponenta sistema je *Game Parser*, koji na osnovu definisane gramatike parsira specifikaciju koja opisuje igru. *Parser* vrši osnovnu validaciju specifikacije (*.gme* fajla) i kao rezultat vraća model koji se nakon toga prevodi u interni *CardGame Model* čija je uloga semantička validacija. *Interpreter* predstavlja glavni deo sistema i vrši tumačenje pravila igre na osnovu modela, omogućavajući njihovu primenu i izvršavanje. Na slici 1 je dat šematski prikazan put od *.gme* fajla do igre.



Slika 1: *Arhitektura obrade i interpretacije* DSL-*a*

3.3.1. *Meta-model*

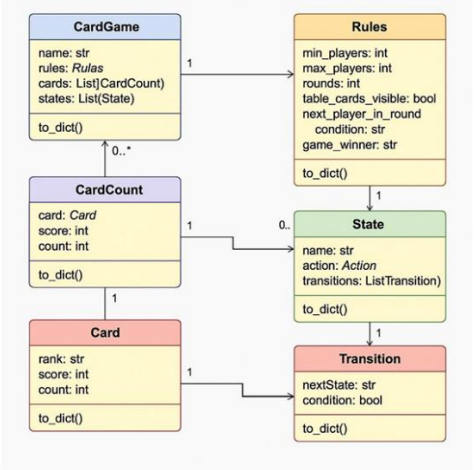
Meta-model se sastoji od elemenata: *Game*, *Rules, State, Transition, Action, CardCount, Card, EnumValue, Rank, Suit* i drugih pomoćnih klasa kao što su *ParamList*, *Param*. Svaka klasa ima svoje atribute i međusobne relacije.

3.3.2. *Parser*

Za obradu specifikacija napisanih u *CardGame DSL-u* koristi se biblioteka *textX*, koja omogućava definisanje gramatike i automatsko generisanje parsera.

3.3.3. *Model*

Radi postizanja veće stabilnosti, podaci dobijeni iz parsera se ne koriste direktno nego se transformišu u interni model. Interni model (slika 2) predstavlja skup specijalizovanih klasa dizajniranih da reprezentuju elemente igre sa svim potrebnim svojstvima i pravilno implementiranom logikom.



Slika 2: *Dijagram klasa internog modela*

3.3.4. *Interpreter*

Interpreter je centralni deo sistema razvijen u *Python-u* koji, koristeći biblioteku *textX*, tumači specifikacije igara definisanih u *CardGame DSL-u* i upravlja logikom i tokom igre. Integrisan je u *backend Flask* aplikaciju organizovanu po *kontroler-servis-repozitorijum* arhitekturi, uz korišćenje *WebSocket* protokola za realnovremensku komunikaciju sa frontend-om. Stanja igre, potezi i rezultati se čuvaju u relacionoj bazi podataka, što omogućava kontinuitet partije. Sistem je dizajniran tako da omogući proširivost i ponovnu upotrebu, bez potrebe za dodatnim kodiranjem pri definisanju novih igara

4. IMPLEMENTACIJA SISTEMA

4.1. Tehnologije

Ovaj sistem kombinuje *textX*, *Python, Flask, PostgreSQL*, *WebSocket, HTTP, JavaScript, HTML i CSS* za izradu interaktivnog *DSL-a* i veb aplikacije. *Backend* upravlja igrom i komunikacijom u realnom vremenu dok *frontend* dinamički generiše interfejs.

4.1.1. *textX*

TextX [6] je *meta-jezik* za definisanje gramatika *DSL-a* u *Python-u*. U ovom rešenju koristi se za parsiranje CardGame skripti i generisanje *AST*.

4.1.2. *Python*

Python [7] je dinamički jezik pogodan za obradu teksta, integraciju i *backend* aplikacije. Korišćen je za *parser*, *backend* server i upravljanje bazom podataka.

4.1.3. *Flask*

Flask [8] je *backend framework* za *Python*, pogodan za veb aplikacije. U sistemu *CardGame* omogućava *HTTP* zahteve, upravljanje stanjem igre i *WebSocket* komunikaciju.

4.1.4. *WebSocket*

WebSocket [9] je protokol koji omogućava dvosmernu komunikaciju u realnom vremenu. Ovde se koristi za razmenu poruka između igrača, bez potrebe za učestalim *HTTP* pozivima.

4.1.5. *HTTP*

HTTP [10] je osnovni protokol za komunikaciju između klijenta i servera. U sistemu se koristi za učitavanje stranica, prijavu i registraciju korisnika.

4.1.6. *PostgreSQL*

PostgreSQL [14] je objektno-relaciona baza podataka poznata po stabilnosti i skalabilnosti. U *CardGame* rešenju služi za skladištenje korisničkih podataka, rezultata i stanja partija.

4.1.7. *HTML, CSS, JavaScript*

HTML [11] definiše strukturu veb interfejsa i sadržaja aplikacije. *CSS [12]* određuje izgled veb stranice, uključujući boje, fontove i razmeštaj elemenata. *JavaScript [13]* omogućava interaktivnost i dinamičko ažuriranje veb interfejsa. U ovom rešenju upravlja *WebSocket* komunikacijom, generisanjem *HTML* sadržaja i reagovanjem na korisničke akcije.

4.2. *CardGame DSL* i gramatika

CardGame DSL je realizovan kao eksterni tekstualni *DSL*. *CardGame*, osnovni element gramatike, prikazan je na slici 3. Sadrži ime igre kao i blokove *rules, states i cards* koji opisuju pravila, stanja i karte

```
CardGame:
    'game' name=ID
     rules=Rules
     states=States
     cards=Cards
;
```

Slika 3: *Element gramatike* CardGame

Element *Rules* prikazan je na slici 4 i predstavlja pravila koja uključuju broj igrača, broj rundi, način izbora sledećeg igrača i kriterijum za pobedu.

```
Rules:
    'rules'
        'min_players' min_players=INT
        'max_players' max_players=INT
        'rounds' rounds=INT
        ('table_cards_visible' table_cards_visible=BOOLEAN)?
        'next_player_in_round_condition' next_player_in_round_condition=NextPlayerCondition
        'game_winner' game_winner=GameWinner
;
```

Slika 4: *Element gramatike* Rules

Element *States* prikazan na slici 5 predstavlja stanja odnosno logičke faze igre koje sadrže akcije i tranzicije ka drugim stanjima.

```
States:
    'states'
        states+=State
;
State:
    'state' name=ID
    'do' action=Action
     transitions+=Transition*
;
```

Slika 5: *Element gramatike* States

Element *Action* prikazan na *slici 6* predstavlja radnju koja se izvršava u okviru određenog stanja. Svaka akcija ima svoj jedinstveni naziv *(name=ID)* kao i opcionalni *deo ('(' params=ParamList ')')?* koji predstavlja parametre akcije koji mogu, a ne moraju biti prisutni.

```
Action:
    name=ID ('(' params=ParamList ')')?
;
```

Slika 6: *Element gramatike* Action

Element *Transition* prikazan na slici 7 predstavlja tranziciju koja pokazuje u koje sledeće stanje igra prelazi. Sastoji se od ključne reči *then* nakon čega sledi naziv sledećeg stanja, a zatim sledi opcioni deo *('if' condition=BOOLEAN)?*. Ovaj deo je opcion jer se mogu javiti različiti scenariji:

1. Ako nema definisanih tranzicija to je kraj igre
2. Ako postoji samo jedna tranzicija bez uslova, prelaz u sledeće stanje se odvija automatski
3. Ako postoji više tranzicija sa uslovima, prelaz se odvija u prvo stanje čiji je uslov ispunjen
4. Ako postoji više tranzicija bez uslova, sledeće stanje zavisi od igrača ili nekog spoljašnjeg faktora

```
Transition:
    'then' nextState=[State] ('if' condition=BOOLEAN)?
;
```

Slika 7: Element gramatike *Transition*

Element gramatike *Cards* prikazan na slici 8 predstavlja karte. Svaka karta (*Card*) je definisana preko ranga i boje, a navodi se i broj njenog pojavljivanja, kao i koliko poena vredi (*CardCount*).

```
Cards:
    'cards'
        cards+=CardCount
;

CardCount:
    'card' card=Card 'appears' count=INT 'times, worth' score=INT 'points'
;

Card:
    rank=Rank 'of' suit=Suit
;
```

Slika 8: *Element gramatike* Cards

4.3. Parsiranje i model

Parsiranje *.gme* datoteka realizovano je uz pomoć *textX* parsera. Parsirani podaci se transformišu u interni model koji vrši dodatnu semantičku validaciju. Interni model predstavlja skup *Python* klasa koje definišu strukturu igre. Ovaj model je osnova za interpretiranje logike igre.

4.4. *Interpreter* i povezivanje komponenti

Interpreter napisan u *Python-u* tumači pravila i upravlja tokom igre. On je integrisan u *Flask backend*, koji kroz *WebSocket* komunicira sa *frontend-om*. Ovo omogućava dinamički tok igre bez učitavanja stranice. Stanja igre i rezultati čuvaju se u *PostgreSQL* bazi podataka.

4.5. *Frontend* i korisnički interfejs

Frontend je razvijen u *HTML/CSS/JavaScript* tehnologijama. Koristi *WebSocket* za interaktivnu komunikaciju i prikazuje tok igre u realnom vremenu. Sistem omogućava registraciju, prijavu, pregled postojećih igara i igranje igre sa protivnikom.

4.6. Pomoćne funkcionalnosti

Sistem sadrži niz funkcionalnosti za podršku korisnicima kao što su *syntax highlighting*, podrška za *snippet-e* i automatska vizualizacija grafa prelaza stanja.

4.7. Konfiguracija i primer upotrebe sistema

Za pokretanje sistema instalirati *Python 3.6+, Flask, PostgreSQL, PyCharm ili VSC.* Klonirati repozitorijum *https://github.com/vulinana/card_game_dsl.git* U *PyCharm-u*, dodati interpreter iz foldera *card_game_dsl*. Dodati *.gme* fajl u *gme/games*. Nakon prijave, u glavnom interfejsu biće dostupna novododata igra.

5. PRIMER SKRIPTE U CARDGAME *DSL-u*

Igra memorijskih karata namenjena je za dva do četiri igrača, a cilj je pronaći što više parova istog ranga. Sastoji se od pet rundi u kojima se igrači smenjuju, dok se karte postavljaju licem nadole. Pobednik je onaj sa najviše poena. Osnovna pravila definisana *CardGame DSL-om* prikazana su na slici 9.

```
game memo_cards
rules

  min_players 2
  max_players 4
  rounds 5
  table_cards_visible false
  next_player_in_round_condition circle_order
  game_winner highest_score
```

Slika 9: *Osnovna pravila igre memorijskih karata*

Na početku igre, na sto se nasumično postavlja osam karata (*deal_table_cards*). Igrač koji je na potezu (*next_player*) bira dve karte (*action_phase*). Ova logika implementirana je kroz stanja prikazana na slici 10.

```
states

state deal_table_cards
  do deal_table_cards(8)
  then any_matching_table_cards

state next_player
  do next_player
  then action_phase

state action_phase
  do select_table_cards(2)
  then reveal
```

Slika 10: *deal_table_cards*, *next_player* i *action_phase*

Karte se okreću (*reveal*) i proverava da li su istog ranga (*matching*_cards). Ako nisu, vraćaju se licem nadole (*flip_back*), a ako jesu, igrač dobija poene (*calculate_points*) i karte se uklanjaju (remove_selected_cards). Prikaz stanja dat je na slici 11.

```
state reveal
  do reveal_selected_cards
  then matching_cards

state matching_cards
  do selected_cards_match(rank)
  then flip_back if false
  then mark_matching_cards_for_scoring if true

state flip_back
  do reset_table_cards_visibility
  then next_player

state mark_matching_cards_for_scoring
  do mark_matching_cards_for_scoring
  then calculate_points

state calculate_points
  do calculate_points
  then remove_selected_cards

state remove_selected_cards
  do remove_selected_cards
  then any_matching_table_cards
```

Slika 11: *reveal, matching_cards, flip_back, mark_matching_cards_for_scoring, calculate points i remove_selected_cards*

Sistem proverava da li na stolu postoje parovi (*any_matching_table_cards*). Ako postoje, igra nastavlja sa sledećim igračem (*next_player*), a ako ne, počinje nova runda (*new_round*). Nakon provere broja rundi (*rounds_remaining*), igra se završava određivanjem pobednika (game_end) ili ponovnim deljenjem karata (deal_table_cards). Prikaz stanja dat je na slici 12

```
state any_matching_table_cards
  do any_matching_table_cards(rank)
  then next_player if true
  then new_round if false

state new_round
  do new_round
  then rounds_remaining

state rounds_remaining
  do check_if_rounds_remaining
  then deal_table_cards if true
  then game_end if false

state game_end
  do determine_game_winner
```

Slika 12: *any_matching_table_cards, new_round, rounds_remainning i game_end*

Na kraju je potrebno još definisati karte koje se mogu pojaviti u toku igre, kao i broj njihovog ponavljanja i broj poena koji nose (slika 13).

```
cards

  card 5 of hearts appears 4 times, worth 0 points
  card 14 of hearts appears 4 times, worth 10 points
  card 13 of hearts appears 4 times, worth 10 points
  card 5 of diamonds appears 4 times, worth 0 points
  card 13 of diamonds appears 4 times, worth 15 point
  card 10 of clubs appears 6 times, worth 10 points
  card 7 of clubs appears 4 times, worth 0 points
  card 6 of clubs appears 6 times, worth 0 points
  card 11 of clubs appears 4 times, worth 15 points
```

Slika 13: *Karte koje se mogu pojaviti u igri*

6. ZAKLJUČAK

Razvijeni sistem pruža efikasno rešenje za dizajn kartaških igara, zasnovano na dostupnosti i jednostavnosti. Deklarativni pristup omogućava intuitivno definisanje pravila, toka igre i ponašanja igrača, čime se uklanja potreba za poznavanjem programskih jezika i sistem postaje dostupan širokom spektru korisnika.

Dalji razvoj sistema omogućava dodavanje funkcionalnosti koje bi unapredile korisničko iskustvo, kao što su:

1. **Proširenje skupa interpretiranih akcija**
2. **Vizuelni editor pravila** – grafički interfejs za kreiranje i uređivanje pravila bez pisanja *DSL* koda.
3. **Automatska dijagnostika grešaka** – prepoznavanje nelogičnih mehanika uz predlog rešenja.
4. **Simulacija testnih partija** – automatsko izvođenje više partija radi analize funkcionalnosti igre.
5. **Interaktivni *debugger* pravila** – korak-po-korak interpretacija pravila i lakše otkrivanje grešaka.

LITERATURA

[1] https://en.wikipedia.org/wiki/Domain-specific_language (pristupljeno u maju 2025.)

[2] https://en.wikipedia.org/wiki/Abstract_syntax_tree (pristupljeno u maju 2025.)

[3] https://gambiter.com/cards/classified-index.html?utm_source=chatgpt.com (pristupljeno u junu 2025.)

[4] https://marianpekar.github.io/Cardamom.js/ (pristupljeno u maju 2025.)

[5] https://www.construct.net/en (pristupljeno u maju 2025.)

[6] https://textx.github.io/textX/ [pristupljeno u avgustu 2025.)

[7] https://www.python.org/doc/ (pristupljeno u maju 2025.)

[8] https://flask.palletsprojects.com/en/latest/ (pristupljeno u maju 2025.)

[9] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (pristupljeno u maju 2025.)

[10] https://developer.mozilla.org/en-US/docs/Web/HTTP (pristupljeno u maju 2025.)

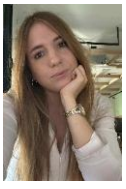
[11] https://developer.mozilla.org/en-US/docs/Web/HTML (pristupljeno u maju 2025.)

[12] https://developer.mozilla.org/en-US/docs/Web/CSS (pristupljeno u maju 2025.)

[13] https://developer.mozilla.org/en-US/docs/Web/JavaScript (pristupljeni u maju 2025.)

[14] https://www.postgresql.org/about/ (pristupljeno u junu 2025.)

Kratka biografija:



Ana Vulin rođena je u Sremskoj Mitrovici 2000. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo, odbranila je 2025.god.

kontakt: sm.vulinana@gmail.com