Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/33BE33Stojkic

ЈЕЗИК СПЕЦИФИЧАН ЗА ДОМЕН ВИЗУАЛИЗАЦИЈЕ ЈЕЗИКА

A DOMAIN-SPECIFIC LANGUAGE FOR THE VISUALIZATION OF LANGUAGES

Радиша Стојкић, *Факултет техничких наука, Нови Сад*

Област –ПРИМЕЊЕНЕ РАЧУНАРСКЕ НАУКЕ И ИНФОРМАТИКА

Кратак садржај – *У овом раду описан је текстуални доменски језик намењен дефинисању визуализације језика креираних помоћу библиотеке textX. Представљене су употребљене технологије и окружења, као и основни појмови везани за језике специфичне за домен и графове. Дат је преглед постојећих решења у овој области, као и кључни делови имплементације самог viewX језика и екстензије за Visual Studio Code. Рад садржи и практичан пример употребе и визуалног приказа, а завршава се освртом на постигнуте циљеве и могућности даљег развоја решења.*

Кључне речи: *Језици специфични за домен, екстензије за VS Code, граф, визуелизација*

Abstract – *This paper describes a textual domain-specific language designed for defining the visualization of languages created using the textX library. It presents the technologies and environments used, as well as the basic concepts related to domain-specific languages and graphs. An overview of existing solutions in this field is provided, along with key parts of the implementation of the viewX language and the Visual Studio Code extension. The paper also includes a practical usage example and a visual representation, concluding with a review of the achieved goals and possibilities for further development of the solution.*

Keywords: *Domain-Specific Languages, VS code extensions, graph, visualization*

1. УВОД

Програми и модели писани са текстуалном синтаксом су често непрегледни и тешко разумљиви. Додатни проблем се јавља ако људи нису упознати са самим појмовима и доменом проблема и језиком на којем је писан програм. Тада је тешко утврдити све односе и кључне детаље у програму и моделу. Један од начина да се превазиђу потешкоће у представи модела и програма јесте визуализација, помоћу које можемо представити све кључне елементе и њихове међусобне релације, што доприноси потпунијем и бољем разумевању модела.

У зависности од мотива и потреба, разликују се различите визуелне репрезентације. Неке од њих су графикони, дијаграми, мреже, графови и многе друге. Главни изазов код визуализације модела је како јасно приказати структуре података у облику графа. Једно од основних питања јесте генеричност решења, како би се омогућило приказивање различитих модела писаних на истом језику уз могућност да корисник манипулише самим приказом датог модела.

Основни мотив овог рада јесте надоградња језика за опис визуализације модела писаних на језицима који су развијени помоћу *textX* библиотеке, као и поједностављење перспективе самог модела. Текстуални језик *ViewX* развијен је као одговор на потребу за једноставнијом и јаснијом визуализацијом доменских модела. Језик је реализован кроз мастер рад на Факултету техничких наука у Новом Саду. *ViewX* служи за директно дефинисање правила визуализације елемената, а развијен је помоћу *textX* библиотеке. Помоћу правила се могу описати особине, структура и елементи графа, као и начин на који се они приказују. Комплетно решење је остварено као екстензија за *Visual Studio Code* едитор, потпомогнуто *third-party* библиотекама, *Python* окружењем и употребом *viewX* језика 0.

Једна од најважнијих предности решења описаног у овом раду јесте концепт одвајања садржаја модела од начина његове визуализације. Садржај чини саму структуру и податке модела, док начин приказа дефинише *viewX* модел, омогућавајући флексибилност приказа. То значи да се један модел може приказати на више различитих начина у зависности од потребе, док се истовремено више различитих модела може представити на исти начин. Овакав приступ обезбеђује прилагодљивост и универзалност у визуализацији, што је кључ за широку примену и лакшу интерпретацију сложених система.

2. ТЕОРИЈСКЕ ОСНОВЕ

Језици специфични за домен (*DSL*) су програмски језици дизајнирани за решавање проблема у јасно дефинисаној области. Они нуде једноставнију синтаксу и већу ефикасност унутар свог домена, али су мање флексибилни од језика опште намене (*GPL*), који су универзални и примењиви у различитим контекстима.

Примери *DSL* језика су *HTML* (структура веб страница) и *SQL* (рад са базама података), док су типични *GPL* језици *Python* и *Java*. Предност *DSL*-а је у прилагођености терминологији и потребама

НАПОМЕНА
Овај рад проистекао је из мастер рада чији ментор је био др Игор Дејановић, ред. проф.

специфичне области, што убрзава развој и смањује потребу за техничким знањем. Мана је мања могућност поновне употребе и теже одржавање у односу на *GPL* језике 0.

Граф је математичка структура *G = (V, E)* где је *V* скуп чворова, а *E* скуп ивица. Може бити усмерен, неусмерен или пондерисан. У рачунарству се користи за моделовање односа, попут мрежних веза, друштвених мрежа или структуре података. Визуализација графова олакшава анализу, али изазови укључују претрпаност приказа и распоред чворова, што се решава филтрирањем, хијерархијским приказима и алгоритмима за аутоматски распоред. Репрезентација графова обично се остварује помоћу матрице суседства (брза провера везе, већа потрошња меморије), листе суседства (ефикасна за ретке графове) или листе грана. Графови налазе примену у рачунарству кроз оптимизацију мрежа, анализу друштвених мрежа, базе података и алгоритме претраге 0.

3. КОРИШЋЕНЕ СОФТВЕРСКЕ ТЕХНОЛОГИЈЕ

Visual Studio Code (*VSC*) је едитор кода који подржава велики број програмских језика и екстензија. Уз интегрисани терминал, *Git* подршку и аутоматско довршавање кода, омогућава продуктиван рад на различитим платформама (*Windows, MacOS, Linux*). *Marketplace* нуди хиљаде екстензија, као што су *Prettier, ESLint* и *Python*, које проширују функционалност едитора 0.

Python је флексибилан и лак за учење програмски језик, погодан за објектно-оријентисано, функционално и процедурално програмирање. Динамичка типизација и богата стандардна библиотека, уз подршку за пакете попут *NumPy, SciPy* и *Pandas*, чине *Python* погодним за научне апликације, машинско учење и анализу података. Једноставност синтаксе омогућава брзо развијање и тестирање апликација.

textX је алат за креирање језика специфичних за домен (*DSL*) у *Python*-у. Користи *PEG* граматику и *Arpeggio* парсер за аутоматско генерисање парсера и мета-модела. Омогућава модуларизацију граматика, конфигурацију парсера, постпроцесирање модела и визуализацију *GraphViz*-ом. Захваљујући једноставности, *textX* је погодан за развој сложених апликација и симулација са прилагођеним језицима.

Cytoscape.js је библиотека за визуализацију и манипулацију графовима у веб окружењу. Подржава усмерене, неусмерене и тежинске графове, интерактивност, динамичке измене и *WebGL/Canvas* рендеровање. Омогућава прилагођавање стила чворова и ивица, као и проширење функционалности додатцима за анализу мрежа и алгоритме.

Jinja2 је *Python* шаблонски механизам за генерисање динамичког *HTML*-а и других текстуалних формата. Подржава *placeholder*-е, петље, услове, филтере и макрое, што олакшава одвојен приказ података од логике апликације. Брзо рендерује сложене шаблоне и интегрише се са веб оквирима као што су *Flask* и *Django*.

4. ПРЕГЛЕД СТАЊА У ОБЛАСТИ

Савремени софтверски системи постају све сложенији, што захтева напредне алате за визуализацију и управљање њиховим компонентама. Алати за визуализацију омогућавају боље разумевање података и ефикасније управљање сложеним структурама. Од графичких едитора до система за визуализацију великих графова, они пружају креирање прилагођених интерфејса, визуелизацију података у реалном времену и интеракцију са елементима. У овом поглављу представљени су *Graphiti*, *Sirius*, *Gephi* и *Neo4j Bloom*.

4.1. *Graphiti*

Graphiti је оквир за визуализацију графичких едитора у *Eclipse* окружењу, омогућавајући лако креирање графичких објеката као што су чворови и ивице. Интеграција са *EMF*-ом омогућава трансформацију текстуалних или табеларних података у интерактивне графичке приказе. Примена је у *UML* дијаграмима, *BPMN* процесима и визуализацији архитектуре софтвера 0.

4.2. *Sirius*

Sirius омогућава креирање прилагођених графичких модела специфичних за домен (*DSL*) у *Eclipse* окружењу. Корисници могу дефинисати статичке и динамичке моделе, као и интерактивне приказе у реалном времену. Флексибилан је за рад у индустрији, инжењерингу, финансијама и научним истраживањима, уз интеграцију са *EMF* и *GMF* 0.

4.3. *Gephi*

Gephi је *open - source* алат за визуализацију и анализу графова, посебно погодан за социјалне мреже, биоинформатику и велике податке. Омогућава интерактивну манипулацију графовима, прилагођавање боја, величине и облика чворова и анализа метрика као што су централност и кластеризација. Ограничење је обрада веома великих графова 0.

4.4. *Neo4j Bloom*

Neo4j Bloom визуелно приказује графове у *Neo4j* бази података. Пружа динамичан и интерактиван приказ, са могућношћу прилагођавања изгледа графова и истраживања релација између чворова. Најпогоднији је за податке већ у *Neo4j*, а за напредну аналитику потребан је *Cypher* 0.

5. ИМПЛЕМЕНТАЦИЈА

Ово поглавље описује кључне делове имплементације екстензије за *Visual Studio Code*, која омогућава интерактивну визуализацију графова и управљање моделима. Решење је реализовано у *TypeScript*-у, интегрисано са *Python* модулима преко *textX* и *Python-shell*, што омогућава креирање граматике *viewX* језика за опис *layout*-а и елемената графа.

Архитектура решења комбинује више компоненти:

- ***BrowserSync*** – синхронизује измене у реалном времену између више клијената, обезбеђујући једноличан приказ.

- ***Socket.io*** – омогућава двосмерну комуникацију, при чему се клијенти региструју у собе и добијају све поруке о стању графа, чиме се постиже слабо спрегнута и флексибилна клијент-сервер архитектура.
- ***Cytoscape.js*** – за приказ и интеракцију са графовима, укључујући креирање подграфова сложених чворова и прилагођене облике чворова.

Python скрипте користе *Jinja2* шаблоне за генерисање *HTML*-а и *layout*-а графова, а интеграција са екстензијом омогућава њихово динамичко извршавање. Решење је *cross-platform* (*Windows* и *Linux*) са динамичким препознавањем ОС-а ради активирања одговарајућег виртуелног окружења.

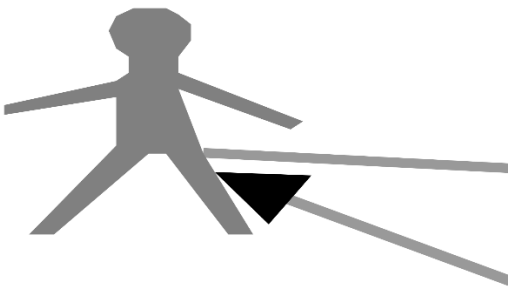
Кроз проширења *viewX* језика и интеракцију са *Cytoscape.js*, екстензија омогућава оптимизован приказ графова са скривањем и приказом потомака сложених чворова, дефинисање облика чворова полигонима и флексибилну визуелизацију која реагује на измене модела у реалном времену. Ово омогућава дугорочну функционалност и прилагодљивост решења различитим сценаријима рада.

6. ДЕМОНСТРАЦИЈА

Проширења реализована у *viewX* екстензији представљена кроз пример примене *mini BPMN* језик за моделовање пословних процеса. *BPMN (Business Process Model and Notation)* је стандардизован графички нотацијски систем за опис пословних токова, омогућавајући комуникацију између техничких и пословних корисника. Циљ примера је демонстрирати могућности екстензије за визуализацију модела креираних помоћу библиотеке *textX*.

ViewX омогућава дефинисање облика чворова кроз координате полигона, боју и величину. На пример, ентитет *Human* може се описати као полигон са дефинисаним тачкама, бојом и стрелицама ка другим ентитетима (Слика 1):

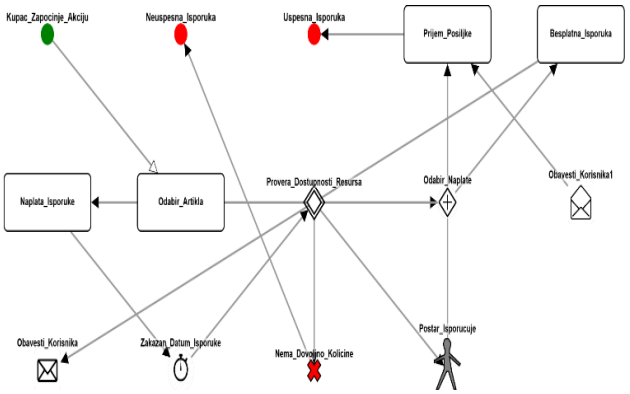
```
view Human as polygon {
    shape-polygon-points: 0.04 -0.88, 0.09 -0.84, ..., -0.09 -0.88
    background: gray
    width: 400
    height: 400
    link {to: Human.action {arrow: black 2 triangle}}
}
```



Слика 1. *Визуализација чвора Human као полигон*

Екстензија омогућава дефинисање распореда елемената графа. На пример, grid layout организује елементе у мрежу (Слика 2):

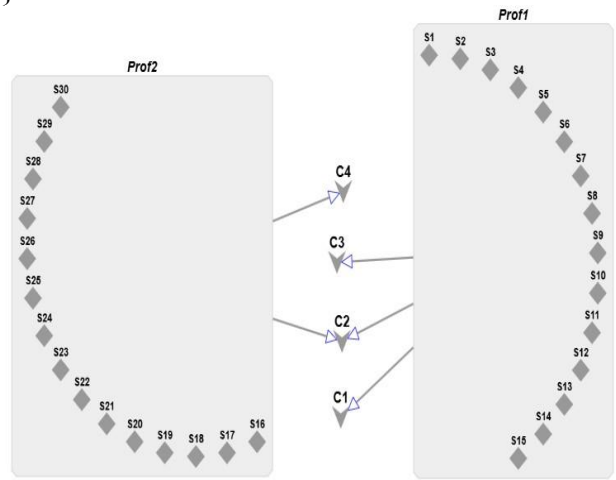
```
layout {
    name: grid
    rows: 3
    cols: 5
    animate: true
    animationDuration: 300
    fit: true
    avoidOverlap: true
}
```



Слика 2. *Grid layout графа*

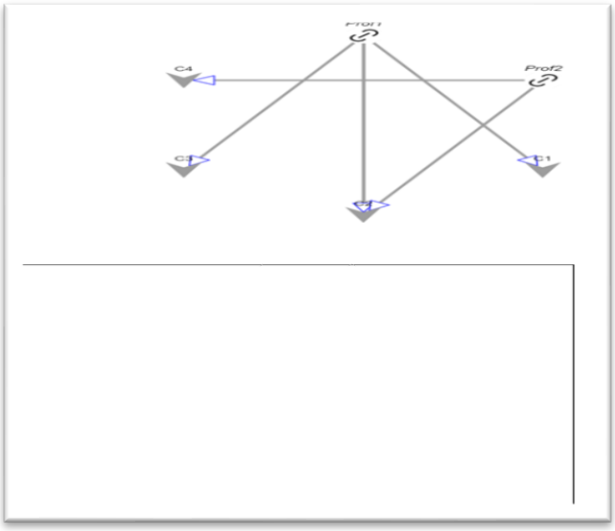
ViewX подржава приказ сложених чворова и њихових потомака. Кључна реч *show* приказује све потомке сложеног чвора, као што је приказано на слици 3.

```
view Student as diamond child of Professor show {
    label: Student.name {font: 15}
}
```

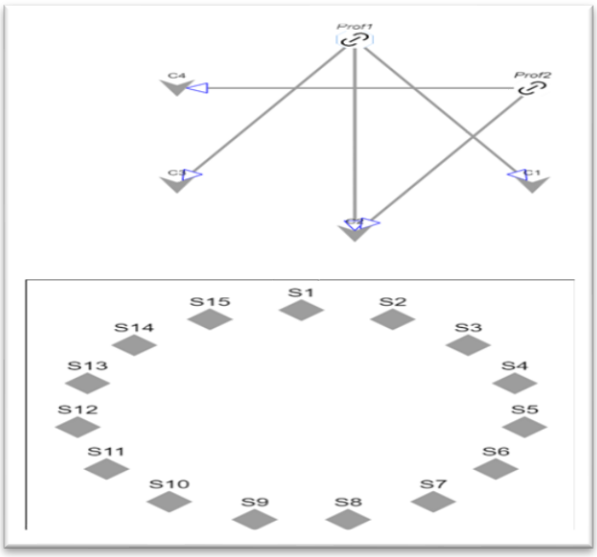


Слика 3. *Сложени чворови са потомцима*

Ради оптимизације, сложени чворови могу се приказати без потомака (Слика 4), а подграф потомака се генерише у посебном прозору испод главног графа (Слика 5).



Слика 4. *Визуализација сложеног чвора без потомака*



Слика 5. *Визуализација подграфа сложеног чвора*

7. ЗАКЉУЧАК

Да би се ово решење издвојило као универзално и свеобухватно, неопходна је његова додатна надоградња. Тренутно је визуализација реализована као *2D* просторна репрезентација. Као једно од могућих унапређења издваја се увођење *3D* просторне репрезентације графа, која би омогућила интуитивније разумевање и већи ниво детаља. На тај начин би се превазишла ограничења која произилазе из дводимензионалног приказа.

У оквиру овог рада реализовано је решење које се издваја од постојећих тиме што нуди различите нивое визуализације и степене детаљности графа. Омогућено је оптимизовање приказа дефинисањем сложених чворова и креирањем подграфова, што знатно смањује количину информација у једном приказу и тиме повећава прегледност. Посебан значај има увођење механизама који кориснику омогућавају да самостално управља приказом, било кроз приказ само сложених чворова, било кроз њихово експлицитно развијање у посебним подграфовима.

Креирање прилагођених облика чворова, спецификација изгледа и анимација представљају додатне функционалности које обогаћују решење. Ове особине, у комбинацији са пажљивим избором новијих и стабилнијих верзија компоненти, доприносе дуговечности и поузданости решења. Оптимизација приказа графа постигнута је не само кроз управљање нивоом детаља, већ и кроз примену алгоритама распоређивања елемената који омогућавају већу читљивост и структурисаност визуализације.

Решење пружа висок степен прилагодљивости и слободе у креирању и управљању визуализацијом. У будућности, проширење функционалности кроз графичку синтаксу и интеграцију више различитих библиотека за приказ графова представљаће значајан корак ка унапређењу.

Иако решење не подржава *3D* форму, оно се истиче флексибилношћу, стабилношћу и могућношћу оптимизације, пружајући корисницима алат за дубље разумевање сложених структура и њихових релација.

8. ЛИТЕРАТУРА

[1] Мастер рад – "Подршка визуализацији језика креираних употребом *textX* библиотеке у оквиру *Visual Studio Code* едитора", Даниел Купчо, Факултет техничких наука у Новом Саду, 2018

[2] *Martin Fowler:* "*Domain-Specific Languages*", *Addison-Wesley Signature Series*, 2010

[3] "*Graph theory*", *Wikipedia*, слободна енциклопедија, https://en.wikipedia.org/wiki/Graph_theory, (приступљено у децембру 2024.)

[4] https://en.wikipedia.org/wiki/Visual_Studio_Code, (приступљено у децембру 2024.)

[5] https://eclipse.dev/graphiti/, (приступљено у децембру 2024.)

[6] Sirius - https://www.eclipse.org/sirius/ (приступљено у децембру 2024.)

[7] https://gephi.org/users/supported-graph-formats/, (приступљено у децембру 2024.)

[8] https://neo4j.com/product/bloom/, (приступљено у децембру 2024.)

Кратка биографија:

Радиша Стојкић је рођен 24.01.2000. године у Зворнику, Босна и Херцеговина. Основну школу "Десанка Максимовић" у Челопеку завршио је 2015. године. Исте године уписује гимназију, општи смер, у ЈУ Средњошколски центар "Петар Кочић" Зворник. Године 2019. гимназију завршава као носилац Вукове дипломе. Факултет техничких наука, смер Рачунарство и аутоматика, уписује 2019. године у Новом Саду, где 2023. године завршава основне академске студије.