Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.9
DOI: https://doi.org/10.24867/17BE04Travica

ВИЗУЕЛИЗАЦИЈА И НАВИГАЦИЈА НАД ИСТОРИЈОМ ПРОМЕНА СОФТВЕРСКОГ ПРОЈЕКТА

SOFTWARE PROJECT CHANGES VISUALIZATION AND NAVIGATION

Милица Травица, *Факултет техничких наука, Нови Сад*

Област – СОФТВЕРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *Истражени и анализирани су Atom текстуални едитор, CodeRibbon пакет за навигацију (настао на основу Patchworks едитора) и визуелизација историје промена над пројектом. Имплементирана је визуелизација историје промена у оквиру пакета CodeRibbon.*

Кључне речи: *управљање конфигурацијом софтвера, Atom, CodeRibbon, визуелизација, навигација, историја промена*

Abstract – *The Atom text editor, CodeRibbon navigation package (based on the Patchworks editor) and visualization of project changes history were researched and analyzed. Visualization of project changes history has been implemented within the CodeRibbon package.*

Keywords: *Software Configuration Management, Atom, CodeRibbon, visualization, navigation, history of change*

1. УВОД

Atom [1] је текстуални едитор. Он је *GitHub* пројекат отвореног кода и бесплатан је за коришћење. *Atom* омогућава додавање нових функционалности инсталирањем пакета, односно тема. Одређени пакети и теме се инсталирају приликом инсталирања самог едитора. Пакет који ће бити анализиран у раду је *CodeRibbon* [2].

CodeRibbon је имплементација *Patchworks* [3] пројекта за *Atom*. *Patchworks* је пројекат у оквиру ког је рађено на бржој и једноставнијој навигацији и организацији фајлова у којима се налази програмски код у интегрисаном развојном окружењу. Пројекат је реализован имплементацијом навигационе траке која садржи *Patch*-еве, у којима се налази програмски код.

Приликом развоја софтверског пројекта уз коришћење текстуалних едитора користе се и системи за контролу верзија (енгл. *Version Control System - VCS*). Они се могу користити независно од едитора али могу бити и подржани у њима. Програмери током развоја софтверског пројекта користе *VCS* јер им помаже у управљању изворним кодом, као и чувању сваке верзије пројекта. Такође, омогућава и сарадњу између програмера.

Циљ рада јесте да се истражи и анализира текстуални едитор *Atom* са акцентом на пакету *CodeRibbon*. Овај пакет биће надограђен са додатном функционалношћу визуелизације и навигације над историјом промена. Конкретније, корисник ће имати могућност увида у личне промене над *Git* пројектима у оквиру *patch*-а, биће му приказан граф промена, као и детаљи о конкретној промени.

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

2. АНАЛИЗА И СТАЊЕ У ОБЛАСТИ

2.1 *Atom*

Atom текстуални едитор отвореног кода је бесплатан за коришћење. Он је направљен коришћењем *HTML*, *CSS*, *JavaScript* и *Node.js*, а заснован је на *Electron* радном оквиру. Могуће је користити га на *Windows*, *Linux* и *OS X* оперативном систему [1].

Atom нуди проширивост и приступачност [4]. Он је хакабилан и прилагодљив јер даје сваком кориснику могућност да мења сам едитор. Корисник може директно мењати код едитора или неког од његових пакета, како би побољшао своје корисничко искуство. Он даје могућност инсталирања нових пакета и деинсталирања постојећих. Такође, корисник може да креира нове пакете, који могу бити сачувани локално или подељени са осталим корисницима [5].

2.2 *Patchworks* и *CodeRibbon*

Patchworks је текстуални едитор који се разликује од других едитора по начину навигације фајлова. Навигација није традиционална, односно базирана на фајловима, а ни *canvas* базирана. *Patchworks* навигација је базирана на *patch grid*-у и траци. Овакав тип навигације помаже програмерима да брже пролазе кроз фајлове, троше мање времена и праве мање грешака у навигацији [6].

Patch је едитор у ком се налази фрагмент кода или фајл над којим могу да се врше измене. *Patch grid* представља главни видљиви део едитора, где могу да се врше измене кода. Разлика у односу на остале едиторе је што корисник истовремено има у видном опсегу 6 фајлова које може да мења, односно приказани су му као *grid* 3x2. Тренутно видљиви *grid* не приказује све отворене *patch*-еве (под условом да их има више од 6), али их трака садржи, односно *grid* приказује само део траке. Корисник има могућност да се креће кроз траку лево и десно и тако мења *patch grid* приказ. Поред тога има могућност да из птичије перспективе види целу траку или одређени део ње, што зависи од укупног броја *patch*-ева [3, 7].

У раду [6] описано је истраживање у којем се пореди *Patchworks* са едитором базираним на фајловима (*Eclipse*) и *canvas* базираним едитором (*Code Bubbles*).

Петнаест испитаника, који су имали претходног искуства са *Eclipse* радним окружењем, је приступило овом истраживању. Нико није претходно користио *Code Bubbles* и *Patchworks*. Резултати овог истраживања су показали да: учесник за навигацију утрошио је приближно дупло мање времена коришћењем *Code Bubbles* и *Patchworks* едитора, најмање грешака је прављено коришћењем *Patchworks* едитора и испитаници су утрошили у просеку осам минута више у односу на *Patchworks* за отварање и измену кода ако су користили *Code Bubbles*.

Закључак који је изведен из рада [3] је да ни једна од четири утврђене стратегије не утиче значајно на брзину навигације програмера. Такође, примећено је и да су све стратегије које су примењене приликом навигације над *Patchworks* едитором брже од навигације *Eclipse* радног окружења. За истраживање је коришћен симулатор који је базиран на стварним подацима о навигацији програмера.

Рад [7] говори о истраживању које је извршено поређењем начина навигације 32 *LabVIEW* програмера коришћењем два различита едитора, од којих је први *Patchworks*, а други едитор са картицама. У раду су дошли до закључка да су коришћењем *Patchworks* едитора учесници направили мање кликова по навигацији него коришћењем едитора заснованог на картицама. Такође установили су пет принципа на које је потребно обратити пажњу приликом дизајнирања навигације за едитор кода.

У раду [8] је представљено решење које ће примењивати *Patchworks* начин навигације у другим едиторима. То се постиже креирањем пакета, односно *plugin*-а за већ постојеће, популарне и коришћене едиторе. Пакет допуњава функционалности едитора тако што омогућава да радни простор корисника буде попут радног простора у *Patchworks* едитору. Разлика у односу на *Patchworks* је што је *grid* подесиве величине. Овај пакет тј. *plugin* се зове *CodeRibbon*. Аутори ове идеје су креирали *CodeRibbon* за едиторе *Atom* и *VS Code*.

2.3 Управљање конфигурацијом софтвера

Управљање конфигурацијом (енгл. *Configuration management - CM*) је јединствена идентификација, контролисано складиштење, контролисана измена и извештавање о статусу производа, његових компоненти и радних верзија током животног циклуса система [9]. Традиционална дефиниција управљања конфигурацијом (софтвера) подразумева идентификовање конфигурације, управљање променама, праћење статуса и ревизију и верификацију [10]. Управљање конфигурацијом, на начин који се и дан данас користи, је настало крајем шездесетих година прошлог века. Посматра се као процес, јер је развој софтвера који се посматра као колекција међусобно зависних процеса имао утицаја на *CM*.

2.4 Системи за контролу верзија

Системи за контролу верзија (енгл. *Version control systems - VCS*) су део система за управљање конфигурацијом (софтвера). У чланку [11] се истиче да је коришћење система за контролу верзије приликом развоја пројекта једнако важно колико и коришћење доброг едитора, интерпретера или компајлера. Говори се и о погодностима које доноси коришћење система за контролу верзије, попут измене датотеке без писања кода преко туђег, није потребно уступање датотеке и сл.

Типови приступа системима за контролу верзија су: централизовани систем за контролу верзија (енгл. *Centralized Version Control System - CVCS*) и дистрибуирани систем за контролу верзија (енгл. *Distributed Version Control System – DVCS*). Основна разлика између дистрибуираних и централизованих система је што дистрибуирани системи не захтевају централни репозиторијум. Сваки корисник има копију репозиторијума локално, на свом рачунару [12].

2.5 *Git*

Git је дистрибуирани систем за контролу верзија. Настао је 2005. године, а његов творац је *Linus Torvalds*. Настаје као ревизија система који је коришћен за развој *Linux* језгра [13].

У књизи [14] је објашњено да је *Git* имплементиран тако да складишти објекте, у којима се налазе оригиналне датотеке, подаци о аутору, датуми и друго. Постоје четири врсте објекта које могу бити складиштене, а то су: *Blob, Tree, Commit* и *Tag*.

2.6 Визуелизација *git* историје промена

Бољи преглед и лакше разумевање промена које су се десиле над пројектом омогућава добра визуелизација историје промена. *GitVS* једно је од решења за представљање визуализације историја промена у *git* пројектима, који је описан у раду [15]. Из рада се може закључити да граф промена заједно са детаљима о *commit*-у представља једно од решења за овај проблем.

Програмерима приликом рада на пројекту може да буде од значаја да виде разлику између тренутне и неке од претходних верзија изабраног фајла. Омогућавање ове функционалности сматра се важним, о чему и говори [16]. Ту је описан *AZURITE plugin*, отвореног кода за *Eclipse* који пружа корисницима увид у историју промена над фајловима.

3. КОРИШЋЕНЕ ТЕХНОЛОГИЈЕ

Atom текстуални едитор је имплементиран коришћењем веб технологија. У почетку код је писан у *CoffeeScript*-у, али је касније већина кода пребачена у *JavaScript* [4]. С обзиром да су творци пакета *CodeRibbon* приликом његове израде користили *JavaScript*, он је коришћен и приликом додавања функционалности визуелизације и навигације над историјом промена. За имплементацију коришћене су и библиотеке *Vis.js* и *Dugite*.

3.1 *Vis.js*

Vis.js је библиотека за визуелизацију, која је оптимизована да ради са великом количином података. Компоненте које садржи ова библиотека су: *Network, Timeline, Graph3d, Graph2d* и *DataSet* [17].

Подаци се чувају у *DataSet*-у, у облику кључ-вредност. *DataSet* омогућава додавање, брисање и измену података, као и ослушкивање догађаја. Такође, пружа и могућност филтрирања, сортирања и конвертовања података [17].

Network се користи за визуелни приказ мрежа, које могу да садрже чворове и гране. Могуће је подешавати боју, величину, облике, стилове и друго. Може да ради са великим бројем чворова и грана. *Network* је подељен на неколико модула, који су задужени за управљање неким његовим делом. У питању су модули: *configure, edges, groups, interaction, layout, manipulation, nodes* и *physics* [17].

3.2 *Dugite*

Dugite повезује *JavaScript*, тј. *Node* пројекте са *git*-ом. Могуће је користити све команде које су доступне у *git*-у, што олакшава рад са *git* репозиторијумом из *JavaScript* пројекта. Ову библиотеку користи и *GitHub Desktop* апликација. Битни делови, које садржи *Dugite*, су: *GitProcess, IGitResult, IGitExecutionOptions, GitError, GitNotFoundErrorCode* и *RepositoryDoesNotExistErrorCode* [18].

4. СПЕЦИФИКАЦИЈА И ИМПЛЕМЕНТАЦИЈА

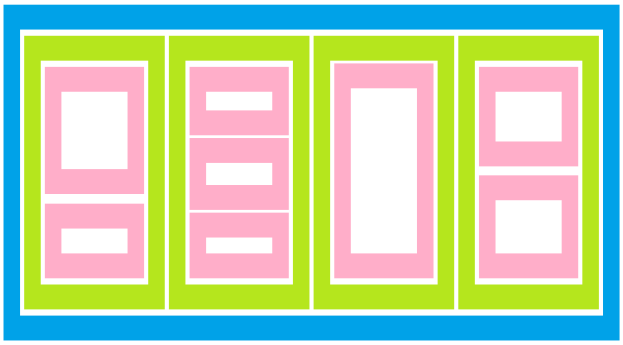
4.1 Спецификација

Начин навигације који је имплементиран у пакету *CodeRibbon* према истраживањима доприноси смањењу времена потрошеног за навигацију и мањем броју навигационих грешака. С друге стране системи за контролу верзија се користе на већини пројеката, *git* је један од њих. Јавља се потреба за бележењем стања пројекта у одређеном временском тренутку, као и увид у та стања. С обзиром на претходно, дошло се на идеју да би било пожељно у таквој навигацији имати и увид у историју над *git* репозиторијумом. Узимајући у обзир закључке који су изведени у поглављу Анализа и стање у области, створена је спецификација за функционалност визуелизације и навигације над историјом промена софтверског пројекта у оквиру *CodeRibbon* пакета. Она обухвата:

- Граф приказ историје промена на коме су приказани *commit*-и, везе између њих и референце. Иницијално учитани граф обухвата 50 последњих *commit*-а али постоји могућност учитавања и осталих *commit*-а. Корисник може да види који све *commit*-и спадају под одређену референцу. Такође, доступан је увид у детаље *commit*-а, као и у којим *commit*-ма је одређени фајл измењен.
- Приказ разлике између тренутне верзије изабраног фајла и неке од његових претходних верзија.

4.2 Имплементација

Полазни фајл *CodeRibbon* пакета је *code-ribbon.js* и налази се у *lib* фолдеру. Овај пакет је имплементиран тако да *CodeRibbonManager* представља полазни објекат за приказ навигације. Она садржи *grid* са произвољним бројем *patch*-ева и траком. У њему се налазе сви *patch* објекти који постоје у отвореној апликацији. Односно један од његових атрибута је *CodeRibbonRibbonContainer* објекат који заправо представља траку. Он садржи више *CodeRibbonSingleStrip* објеката који репрезентују део траке, у коме су *patch*-еви приказани један испод другог. Објекат који означава *patch* је *CodeRibbonPatch*. Он може да садржи више ставки али и ни једну. Слика 1. графички приказује начин имплементације *CodeRibbon* пакета. Плавом бојом је представљен *CodeRibbonRibbonContainer*, зеленом *CodeRibbonSingleStrip*, а са розе *CodeRibbonPatch* објекат.

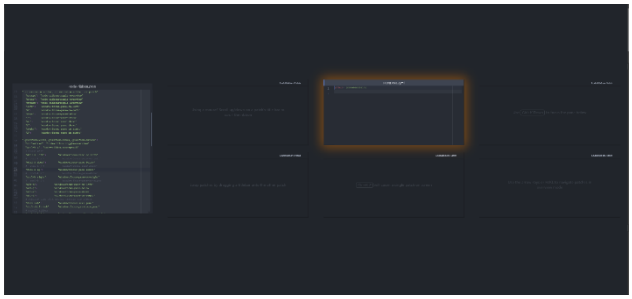


Слика 1. *Графички приказ начина имплементације CodeRibbon пакета*

За развој функционалности визуелизације и навигације над историјом промена креирани су фајлови који се налазе у *visualization* фолдеру. *JavaScript* фајлови који се у њему налазе су: *commit.js, file-diff.js, git-utils.js, graph-utils.js* и *network-graph.js*. Поред тога додати су одређени делови кода у већ постојећим фајловима. У оквиру *lib* фолдера то су фајлови: *code-ribbon-manager.js, code-ribbon-patch.js* и *code-ribbon.js*. У датотеци *git-utils.js* налазе се функције и константе за рад са *Dugite* библиотеком. У *commit.js* фајлу смештена је класа *Commit*. Класа *NetworkGraph* (*network-graph.js*) и класа *FileDiff* (*file-diff.js*) представљају нове ставке које могу бити приказане у *patch*-у.

5. ИЗГЛЕД АПЛИКАЦИЈЕ

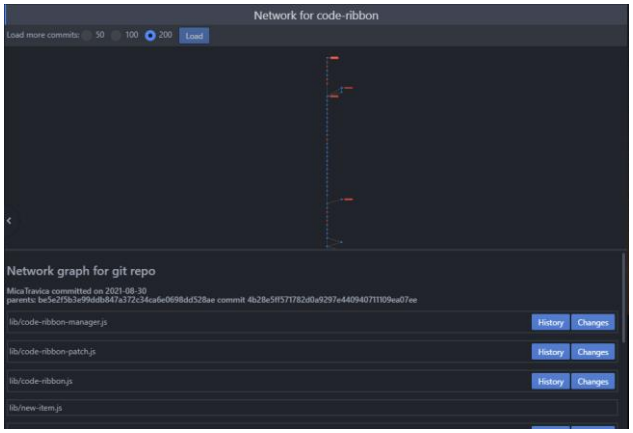
Слика 2. приказује удаљени приказ *CodeRibbon* навигације, односно целу навигациону траку. *Patch* који је тренутно у фокусу издваја се по бојама ивица, уоквирен је наранџастом. Могуће је променити *patch* на којем је фокус, изаћи из приказа целе траке, додавати нове колоне са лева или десна, као и обрисати колону. У *patch*-у који нема ставки могуће је користити брзу претрагу фајлова у оквиру пројекта.



Слика 2. *CodeRibbon трака*

Нова функционалност омогућава увид историје промена над пројектом, учитавање старијих измена, детаље о одређеном *commit*-у, приказ *commit*-а који спадају под одређену референцу, приказ *commit*-а у ком је мењан изабрани фајл и разлике између тренутне и неке од претходних верзија фајла. Слика 3. приказује историју промена за фајл *lib/code-ribbon-manager.js*. Са ње се могу видети детаљи о изабраном

commit-у, као и функционалности које постоје за мењане фајлове.



Слика 3. *Визуелни приказ историје промена, историја промене фајла*

6. ЗАКЉУЧАК

У раду је анализиран начин навигације у текстуалним едиторима и интегрисаним развојним окружењима. Текстуални едитор на који је стављен акценат је *Atom*, због своје проширивости. Истражен је начин навигације заснован на навигационој траци и *patch grid*-у, који је имплементиран у *Patchworks* едитору. Такође је истражен и *plugin CodeRibbon* који омогућава исту навигацију као и код *Patchworks* едитора. *Plugin* је реализован у едиторима *Atom* и *VSCode*. Изведени су закључци да граф промена заједно са детаљима о *commit*-у омогућава бољи преглед и лакше разумевање промена над пројектом и да увид у разлику између тренутне и неке од претходних верзија изабраног фајла корисници сматрају битним и корисним. На основу истраживања, анализа и донесених закључака утврђена је спецификација и имплементирана је додатна функционалност визуелизације историје промена *CodeRibbon* пакета.

Полазна тачка за даљи развој навигације и визуелизације над историјом промена софтверског пројекта може да буде једна од првобитних идеја. Она подразумева да се приказ разлика између две верзије истог фајла прикажу у једном *patch*-у, а у другом тренутна верзија тог фајла са могућношћу измене. Поред тога, приликом скроловања једног од *patch*-ева и други би био скролован. Како би се реализовала ова идеја потребно је изменити постојећи објекат у коме је имплементирана измена текста фајла или евентуално направити нови.

7. ЛИТЕРАТУРА

[1] „Atom“, [На мрежи]. Доступно на: https://atom.io/. [Последњи приступ Септембар 2021].

[2] „CodeRibbon”, [На мрежи]. Доступно на: https://utk-se.github.io/CodeRibbon/. [Последњи приступ септембар 2021].

[3] A. Z. Henley, A. Singh, S. D. Fleming и M. V. Luong, „Helping Programmers Navigate Code Faster with Patchworks: A Simulation Study”, у 2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2014.

[4] „Atom Flight Manual”, [На мрежи]. Доступно на: https://flight-manual.atom.io/. [Последњи приступ септембар 2021].

[5] L. B. a. A. S. M. K Sumangali, „A Comprehensive review on the open source hackable text”, у IOP Conf. Series: Materials Science and Engineering, 2017.

[6] A. Z. Henley и S. D. Fleming, „The Patchworks Code Editor: Toward Faster Navigation with Less Code Arranging and Fewer Navigation Mistakes”, у Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2014.

[7] A. Z. Henley, S. D. Fleming и M. V. Luong, „Toward Principles for the Design of Navigation Affordances in Code Editors: An Empirical Investigation”, у Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017.

[8] B. P. Klein и A. Z. Henley, „CodeRibbon: More Efficient Workspace Management and Navigation for Mainstream Development Environments”, 2021.

[9] A. M. J. Hass, Configuration Management Principles and Practice, Addison Wesley, 2002.

[10] B. Aiello и L. Sachs, Configuration Management Best Practices, Addison-Wesley, 2010.

[11] D. Spinellis, „Version Control Systems”, Tools of the trade, pp. 108-109, 2005.

[12] S. Otte, „Version Control Systems”, у Computer Systems and Telematics, 2009.

[13] D. Spinellis, „Git”, Tools of the trade, pp. 100-101, 2012.

[14] J. Loeliger, Version Control with Git, O’Reilly Media, Inc., 2009.

[15] K. J. North, A. Sarma и M. B. Cohen, „Understanding Git History: A Multi-Sense View”, у Proceedings of the 8th International Workshop on Social Software Engineering, 2016.

[16] Y. Yoon, B. A. Myers и S. Koo, „Visualization of Fine-Grained Code Change History”, у 2013 IEEE Symposium on Visual Languages and Human Centric Computing, 2013.

[17] „Vis.js”, [На мрежи]. Доступно на: https://visjs.org/. [Последњи приступ септембар 2021].

[18] „Dugite, документација”, [На мрежи]. Доступно на: https://github.com/desktop/dugite/tree/e57cb95c17c411192b0c66397e6e84e19595dce8/docs. [Последњи приступ септембар 2021].

Кратка биографија:

Милица Травица рођена је у Сомбору 1997. год. Мастер рад на Факултету техничких наука из области Електротехнике и рачунарства – Софтверско инжењерство, одбранила је 2021. год.
контакт: milica.travica@uns.ac.rs