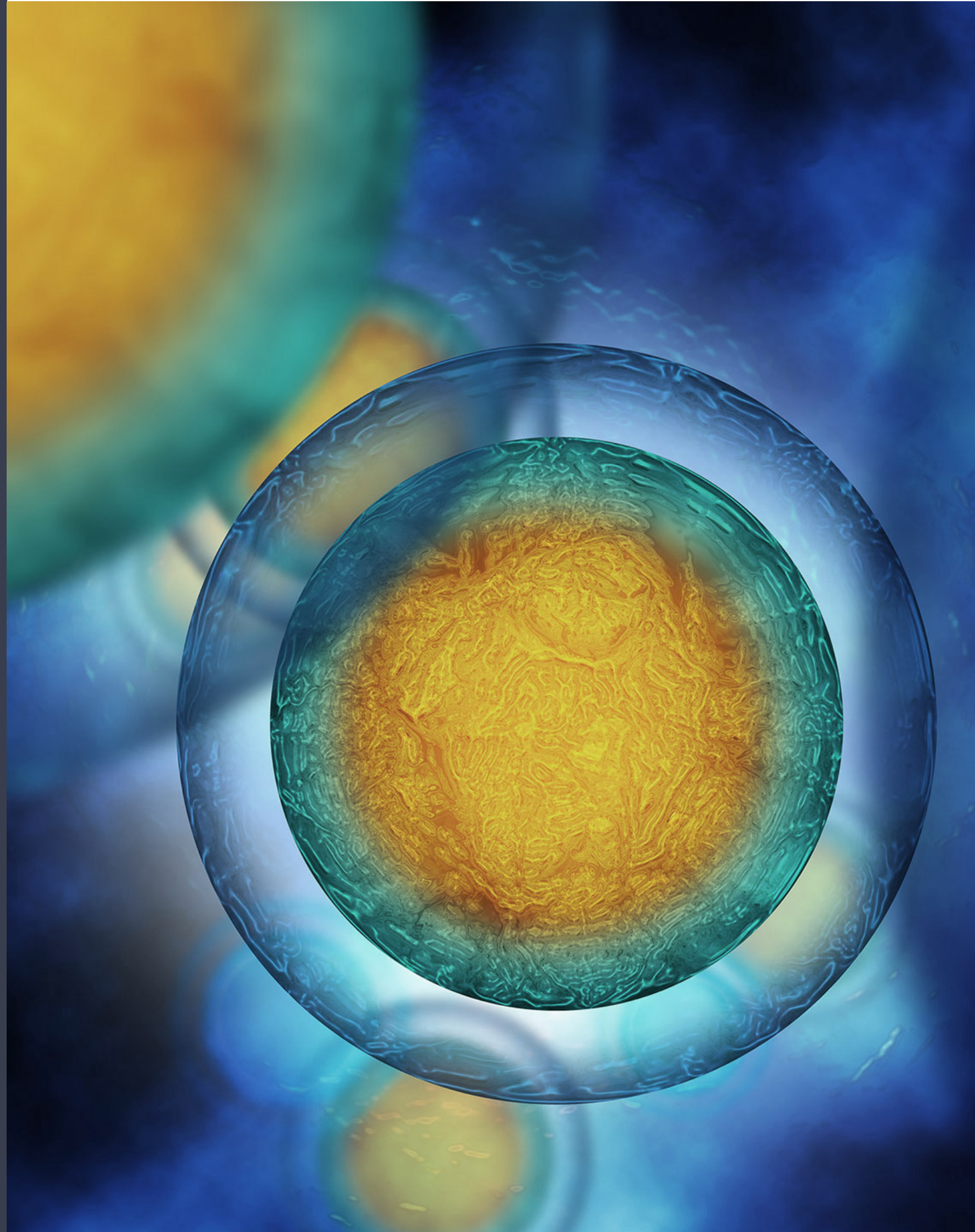
FE3CWS

MATERIJAL ZA LJETNU ŠKOLU CEFP 2019

Intelektualni doprinos O4 iz ERASMUS+ projekta 2017-1-SK01-KA203-035402



Nekoliko riječi o

SADRŽAJU

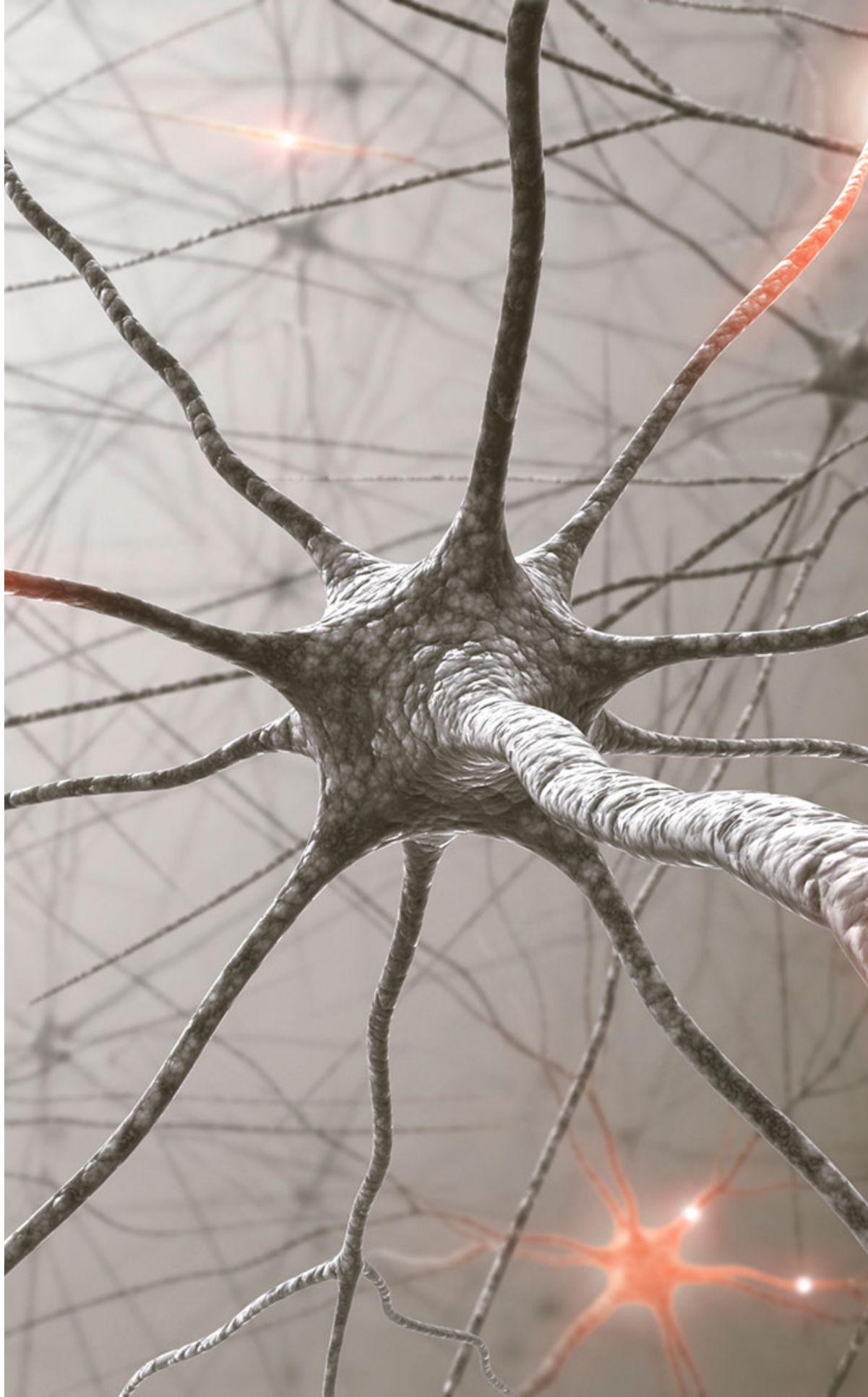
- **10 tema o kompoziciji, razumljivosti i ispravnosti softvera**
- **20 autora iz 7 europskih sveučilišta iz Hrvatske, Mađarske, Nizozemske, Portugala i Slovačke**
- **Dostupan na 7 jezika: engleski, mađarski, slovački, hrvatski, rumunjski, bugarski i portugalski**

Co-funded by the Erasmus+ Programme of the European Union

Ljetna škola Centralno-europskog funkcionalnog programiranja (CEFP) je drugi intenzivan program za učenike i nastavno osoblje visokog obrazovanja koji šire zajednicu ljetne škole Centralno-europskog funkcionalnog programiranja (eng. Central European Functional Programming, CEFP) u okviru ERASMUS + projekta Br. 2017-1-SK01-KA203-035402 pod naslovom "Fokusiranje obrazovanja na kompatibilnost, razumljivost i ispravnost radnog softvera" koji se održava od 17. do 21. lipnja 2019. godine.

Uključeni materijal izrađen je i predstavljen u okviru navedenog projekta. Ova publikacija je tiskana verzija intelektualnog doprinosa O4 iz ovoga projekta.

SADRŽAJ



VIZUALNA IZRADA PROTOTIPA UPOTREBOM PROGRAMIRANJA ORIJENTIRANOG NA ZADATKE

U ovom ćemo tečaju izraditi aplikacije pomoću vizualnog pomoćnika za programiranje orijentirano na zadatke (eng. Task Oriented Programming, TOP). TOP je nova paradigma programiranja koju programeri mogu koristiti za brzu izradu prototipa web aplikacija za višestruke korisnike. Središnji dio modeliranja aplikacija u TOP-u je kreiranje zadataka. Zadaci predstavljaju dijelove posla iz stvarnog svijeta koja mogu izvoditi ljudi ili sustavi. Pomoću nekoliko operacija mogu se kombinirati u veće i snažnije zadatke.

Predstaviti ćemo osnovne koncepte TOP-a proučavajući primjere aplikacija, prikazujući kako ih se može modelirati koristeći Zadatke u okruženju za vizualni razvoj. Vizualno okruženje vodi programere tijekom procesa modeliranja. Alat prikazuje samo razumne načine stvaranja i proširivanja zadataka te daje savjete kako riješiti pogreške tipa i opsega. To rezultira ispravnim programskim kodom koji prolazi kompilaciju.

Studenti će biti potaknuti proširiti predstavljene primjere aplikacija u praktičnom dijelu. Naš vizualni pristup zahtijeva samo osnovna znanja programiranja i tipova podataka. Uvod u TOP i principe modeliranja preduvjet su za ovu lekciju posvećenu mTask-u.

PROGRAMIRANJE ORIJENTIRANO NA ZADATKE ZA INTERNET STVARI

Internet Stvari (eng. Internet of Things, IoT) sastoji se od uređaja koji osjećaju, djeluju i komuniciraju s drugim sustavima na Internetu. Tipični zahtjevi za IoT uređaje su niska cijena i potrošnja energije. To se postiže pokretanjem IoT uređaja malim mikroprocesorima s malom količinom memorije i procesorske snage. Većina ovih sustava nema odgovarajući operativni sustav i samo pokreću određeni program da bi izvršili predviđeni zadatak.

Zbog toga programiranje IoT-a vrlo je izazovno. Pojedinačni program koji se izvodi na takvom uređaju mora kombinirati sve podvrste zadataka, poput nadziranja ulaza, kontrole perifernih uređaja i komunikacije. Razni uređaji koji surađuju moraju se složiti s korištenim protokolom i moraju riješiti zloglasne istodobne programske probleme.

U ovom predavanju predstaviti ćemo se uvod u Programiranje orijentirano na zadatke (eng. Task Oriented Programming, TOP) za IoT. U našem TOP pristupu, mTask sustav upravlja transparentnom komunikacijom između uređaja i njihovih poslužitelja. Cijeli je sustav programiran u jedinstvenom funkcionalnom programu visoke razine. Za svaku podvrstu zadatka sustava definiramo odgovarajući mTask. Ove podvrste zadataka mogu biti sastavljene od strane sustava za kombiniranje jednostavnih u snažnije zadatke. Takvi zadaci mogu pregledavati posredne vrijednosti drugih podvrsta zadataka te komunicirati s bilo kojim drugim zadatkom u sustavu putem zajedničkih izvora podataka. Podskupovi zadataka za IoT uređaj dinamički se šalju na uređaj i tamo interpretiraju. Sustav jakog tipa sprječava probleme s vremenom izvođenja. Ovaj TOP pristup uvelike pojednostavljuje razvoj softvera za IoT.

Literatura

Peter Achten. Clean for Haskell98 Programmers. 13th July 2007.

Peter Achten, Pieter Koopman and Rinus Plasmeijer. 'An Introduction to Task Oriented Programming'. In: Central European Functional Programming School. Springer, 2015, pp. 187– 245.

Douglas Adams. The Hitchhiker's Guide to the Galaxy Omnibus: A Trilogy in Four Parts. Vol. 6. Pan Macmillan, 2017.

Matheus Amazonas Cabral De Andrade. 'Developing Real Life, Task Oriented Applications for the Internet of Things'. Master's Thesis. Nijmegen: Radboud University, 2018. 60 pp.

Tom Brus et al. 'Clean – a language for functional graph rewriting'. In: Conference on Functional Programming Languages and Computer Architecture. Springer, 1987, pp. 364– 384.

Jacques Carette, Oleg Kiselyov and Chung-Chieh Shan. 'Finally tagless, partially evaluated: Tagless staged interpreters for simpler typed languages'. In: Journal of Functional Programming 19.5 (Sept. 2009), p. 509. issn: 0956-7968, 1469-7653. doi: 10.1017/S0956796809007205.

James Cheney and Ralf Hinze. First-class phantom types. Cornell University, 2003.

Li Da Xu, Wu He and Shancang Li. 'Internet of things in industries: a survey'. In: Industrial Informatics, IEEE Transactions on 10.4 (2014), pp. 2233–2243.

L. M. G. Feijs. 'Multi-tasking and Arduino : why and how?' In: Design and semantics of form and movement. 8th International Conference on Design and Semantics of Form and Movement (DeSForM 2013). Ed. by L. L. Chen et al. Wuxi, China, 2013, pp. 119–127. isbn: 978-90-386-3462-3.

Patrick C. Hickey et al. 'Building embedded systems with embedded DSLs'. In: ACM Press, 2014, pp. 3–9. isbn: 978-1-4503-2873-9. doi: 10.1145/2628136.2628146.

Pieter Koopman, Mart Lubbers and Rinus Plasmeijer. 'A Task-Based DSL for Microcomputers'. In: Proceedings of the Real World Domain Specific Languages Workshop 2018 on - RWDSL2018. Vienna, Austria: ACM Press, 2018, pp. 1–11. isbn: 978-1-4503-6355-6. doi: 10.1145/3183895.3183902.

Mart Lubbers. 'Task Oriented Programming and the Internet of Things'. Master's Thesis. Nijmegen: Radboud University, 2017. 69 pp.

Mart Lubbers, Pieter Koopman and Rinus Plasmeijer. 'Multitasking on Microcontrollers using Task Oriented Programming'. In: 2019 42st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). COnference on COmposability, COmprehensibility and COrrectness of Working Software. Opatija, Croatia: IEEE, 2019, pp. 1842–1846.

Mart Lubbers, Pieter Koopman and Rinus Plasmeijer. 'Task Oriented Programming and the Internet of Things'. In: Proceedings of the 30th Symposium on the Implementation and Application of Functional Programming Languages. International Symposium on Implementation and Application of Functional Languages. Lowell, MA: ACM, 2018, p. 12. isbn: 978-1-4503-7143-8. doi: 10.1145/3310232.3310239.

Rinus Plasmeijer, Peter Achten and Pieter Koopman. 'iTasks: executable specifications of interactive work flow systems for the web'. In: ACM SIGPLAN Notices 42.9 (2007), pp. 141–152.

Rinus Plasmeijer and Pieter Koopman. 'A Shallow Embedded Type Safe Extendable DSL for the Arduino'. In: Trends in Functional Programming. Vol. 9547. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016. isbn: 978-3-319-39109-0 978-3-319-39110-6. doi: 10.1007/978-3-319-39110-6.

Camil Staps and Mart Lubbers. The Clean language search engine. 2017. url: https: //cloogle.org.

Jurrien Stutterheim, Peter Achten and Rinus Plasmeijer. 'Maintaining Separation of Concerns Through Task Oriented Software Development'. In: Trends in Functional Programming. Ed. by Meng Wang and Scott Owens. Vol. 10788. Cham: Springer International Publishing, 2018, pp. 19–38. isbn: 978-3-319-89718-9 978-3-319-89719-6. doi: 10.1007/978- 3-319-89719-6_2.

OBOJIMO PROGRAME U ZELENO – ENERGETSKA UČINKOVITOST IMPLEMENTACIJA PODATKOVNIH STRUKTURA

Energetska učinkovitost već godinama brine i inženjere hardvera i softwera niske razine [1], [2], [3]. Međutim, rastući svjetski trend je težnja ka održivosti, uključujući održivost softvera [4], u kombinaciji sa sustavnom prirodom energetske učinkovitosti kao atributa kvalitete, motivirali su proučavanje energetskog utjecaja aplikacijskog softvera u radu. Taj trend motivirao je istraživače da evaluiraju postojeće tehnike, alate i jezike za razvoj aplikacija iz energetski usmjerene perspektive. Nedavna istraživanja proučavala su utjecaj čimbenika, kao što su prikrivanje koda [5], Android API pozivi [6], objektno orijentirana obnova kodova [7], konstrukcije za istodobnu izvedbu [8] i vrste podataka [9] na energetsku učinkovitost. Analiza utjecaja različitih čimbenika na energiju važna je za inženjere u razvoju, ali i održavanju softvera.

Collections	Associative Collections	Sequences
EnumSet StandardSet UnbalancedSet LazyPairingHeap LeftistHeap MinHeap SkewHeap SplayHeap	AssocList PatriciaLoMap StandardMap TernaryTrie	BankersQueue SimpleQueue BinaryRandList JoinList RandList BraunSeq FingerSeq ListSeq RevSeq SizedSeq MyersStack

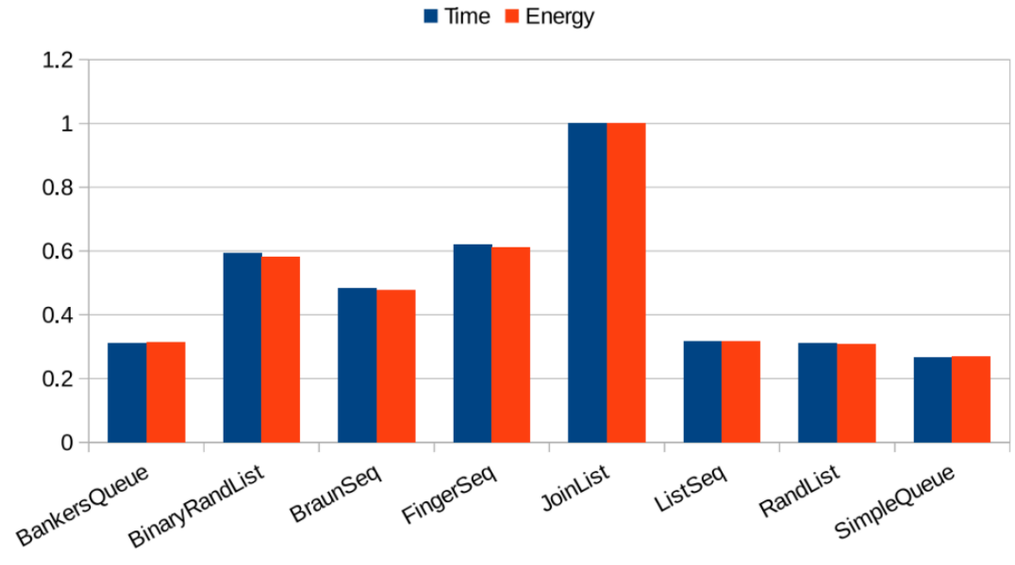
iters	operation	base	aux
1	add	100000	100000
1000	addAll	100000	1000
1	clear	100000	n.a.
1000	contains	100000	1
5000	containsAll	100000	1000
1	iterator	100000	n.a.
10000	remove	100000	1
10	removeAll	100000	1000
5000	toArray	100000	n.a.
10	retainAll	100000	1000

U ovoj pokaznoj vježbi analiziramo i uspoređujemo energetsku učinkovitost različitih implementacija za konkretne apstrakcije podataka, poput sekvenci, skupova ili asocijativnih zbirki. Za svaku implementaciju provjeravamo kako operacije poput dodavanja, brisanja ili pretraživanja elemenata podnose različita radna opterećenja. Predmeti našeg istraživanja su funkcionalni programski jezik [10,11] i objektno orijentirana paradigma [13,14].

U okruženju funkcionalnih jezika, uspoređivali smo implementacije prikazane na Slici 1. koristeći operacije prikazane na Slici 2. U domeni objektno-orijentiranih jezika, analizirali smo implementacije prikazane na Slici 3., pomoću operacija prikazanih na Slici 4.

Naš cilj je pružiti razvojnim inženjerima korisne informacije koje su već integrirane u prateće alate i koji mogu usmjeriti zelenu konstrukciju softvera. Uspjeli smo pokazati da se iste operacije, koje su dostupne u različitim izvedbama, mogu značajno razlikovati u pogledu vremena izvođenja i potrošnje energije. Kao primjer, na Slici 5 prikazani su rezultati operacije uklanjanja za apstrakciju implementacije Sekvenci dostupne u Haskell-ovoj biblioteci Edison.

Sets	Lists	Maps
ConcurrentSkipListSet	*ArrayList*	*ConcurrentHashMap*
CopyOnWriteArraySet	*AttributeList*	*ConcurrentSkipListMap*
HashSet	*CopyOnWriteArrayList*	*HashMap*
LinkedHashSet	*LinkedList*	*Hashtable*
TreeSet	*RoleList*	*IdentityHashMap*
	RoleUnresolvedList	*LinkedHashMap*
	Stack	*Properties*
	Vector	*SimpleBindings*
		TreeMap
		UIDefaults
		WeakHashMap



Sets	Lists	Maps
add	*add*	*clear*
addAll	*addAll*	*containsKey*
clear	*add(index)*	*containsValue*
contains	*addAll(index)*	*entrySet*
containsAll	*clear*	*get*
iterateAll	*contains*	*iterateAll*
iterator	*containsAll*	*keySet*
remove	*get*	*put*
removeAll	*indexOf*	*putAll*
retainAll	*iterator*	*remove*
toArray	*lastIndexOf*	*values*
	listIterator	
	listIterator(index)	
	remove	
	removeAll	
	Continues...	

Literatura

[1] A. Chandrakasan, S. Sheng, and R. Brodersen, “Low-power cmos digital design,” Solid-State Circuits, IEEE Journal of, vol. 27, no. 4, pp. 473– 484, Apr 1992.

[2] V. Tiwari, S. Malik, and A. Wolfe, “Power analysis of embedded software: a first step towards software power minimization,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 2, no. 4, pp. 437–445, Dec 1994.

[3] W. Yuan and K. Nahrstedt, “Energy-efficient soft real-time cpu scheduling for mobile multimedia systems,” in Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. ACM, 2003, pp. 149–163.

[4] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, M. Mahaux, B. Penzenstadler, G. Rodríguez-Navas, C. Salinesi, N. Seyff, C. C. Venters, C. Calero, S. A. Koçak, and S. Betz, “The karlskrona manifesto for sustainability design,” CoRR, vol. abs/1410.6968, 2014.

[5] C. Sahin, P. Tornquist, R. Mckenna, Z. Pearson, and J. Clause, “How does code obfuscation impact energy usage?” in 30th IEEE International Conference on Software Maintenance and Evolution, Victoria, BC, Canada, September 29 - October 3, 2014, 2014, pp. 131–140.

[6] M. L. Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. D. Penta, and D. Poshyvanyk, “Mining energy-greedy API usage patterns in android apps: an empirical study,” in 11th Working Conference on Mining Software Repositories, MSR 2014, Proceedings, May 31 - June 1, 2014, Hyderabad, India, 2014, pp. 2–11.

[7] C. Sahin, L. Pollock, and J. Clause, “How do code refactorings affect energy usage?” in Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2014, pp. 36:1–36:10.

[8] G. Pinto, F. Castor, and Y. D. Liu, “Understanding energy behaviors of thread management constructs,” in Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, ser. OOPSLA ’14. New York, NY, USA: ACM, 2014, pp. 345–360. [Online]. Available: http://doi.acm.org/10.1145/2660193.2660235

[9] K. Liu, G. Pinto, and Y. Liu, “Data-oriented characterization of application-level energy optimization,” in Proceedings of the 18th International Conference on Fundamental Approaches to Software Engineering, ser. Lecture Notes in Computer Science, vol. 9033, 2015, pp. 316–331.

[10] Luis Gabriel Lima, Francisco Soares-Neto, Paulo Lieuthier, Fernando Castor, Gilberto Melfe, João Paulo Fernandes: Haskell in Green Land: Analyzing the Energy Behavior of a Purely Functional Language. SANER 2016: 517-528

[11] Luis Gabriel Lima, Francisco Soares-Neto, Paulo Lieuthier, Fernando Castor, Gilberto Melfe, João Paulo Fernandes, On Haskell and energy efficiency. Journal of Systems and Software 149: 554-580 (2019)

[12] Rui Pereira, Marco Couto, João Saraiva, Jácome Cunha, João Paulo Fernandes: The influence of the Java collection framework on overall energy consumption. GREENS@ICSE 2016: 15-21

[13] Rui Pereira, Pedro Simão, Jácome Cunha, João Saraiva: jStanley: placing a green thumb on Java collections. ASE 2018: 856-859

ZELENO RAČUNARSTVO U KOLEGIJU INŽENJERSTVA

Održivi razvoj postaje sve važnija tema ne samo u svjetskoj politici, nego i centralna tema za inženjersku profesiju diljem svijeta. U brojnim istraživačkim studijama je pokazano kako programski inženjeri nisu nimalo iznimka. Unatoč intenzivnim istraživačkim aktivnostima usmjerenim na zelene programske proizvode, današnje preddiplomsko obrazovanje u računarstvu često zanemaruje naše dužnosti prema okolišu. Stoga, predstavljamo modul posvećen zelenom softwareu koji smo uveli u napredni kolegij programskog inženjerstva. Predstaviti ćemo katalog indikatora loše energetske učinkovitosti i zelenih adaptacija, za koje znamo da pomažu studentima u razumijevanju i optimizaciji potrošnje energije programskih sustava.

Literatura

[1] A. Chandrakasan, S. Sheng, and R. Brodersen, “Low-power cmos digital design,” Solid-State Circuits, IEEE Journal of, vol. 27, no. 4, pp. 473– 484, Apr 1992.

[2] V. Tiwari, S. Malik, and A. Wolfe, “Power analysis of embedded software: a first step towards software power minimization,” Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 2, no. 4, pp. 437–445, Dec 1994.

[3] M. L. Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. D. Penta, and D. Poshyvanyk, “Mining energy-greedy API usage patterns in android apps: an empirical study,” in 11th Working Conference on Mining Software Repositories, MSR 2014, Proceedings, May 31 - June 1, 2014, Hyderabad, India, 2014, pp. 2–11.

[4] C. Sahin, L. Pollock, and J. Clause, “How do code refactorings affect energy usage?” in Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2014, pp. 36:1–36:10.

[5] G. Pinto, F. Castor, and Y. D. Liu, “Understanding energy behaviors of thread management constructs,” in Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications, ser. OOPSLA ’14. New York, NY, USA: ACM, 2014, pp. 345–360.

[6] K. Liu, G. Pinto, and Y. Liu, “Data-oriented characterization of application-level energy optimization,” in Proceedings of the 18th International Conference on Fundamental Approaches to Software Engineering, ser. Lecture Notes in Computer Science, vol. 9033, 2015, pp. 316–331.

[7] Luis Gabriel Lima, Francisco Soares-Neto, Paulo Lieuthier, Fernando Castor, Gilberto Melfe, João Paulo Fernandes: Haskell in Green Land: Analyzing the Energy Behavior of a Purely Functional Language. SANER 2016: 517-528

[8] Rui Pereira, Marco Couto, João Saraiva, Jácome Cunha, João Paulo Fernandes: The influence of the Java collection framework on overall energy consumption. GREENS@ICSE 2016: 15-21

[9] Rui Pereira, Pedro Simão, Jácome Cunha, João Saraiva: jStanley: placing a green thumb on Java collections. ASE 2018: 856-859

ENERGETSKO PROFILIRANJE PROGRAMSKIH APLIKACIJA ZA PROJEKTE PISANE U JAVI

Ova pokazna lekcija posvećena je energetskoj učinkovitosti programskih aplikacija implementiranih u programskom jeziku Java. Prvi dio opisuje trenutno stanje u energetskom profiliranju te mogućnosti za prikaz potrošnje energije dijelova programskog koda. Potom ćemo prikazati vlastitu metodu analize koda za prikaz informacija o procesoru, radnoj memoriji i tvrdog diska. Nakon analize slijedi kratak uvod u naše aplikacije implementirane u Javi. Kako bi se demonstrirala praktičnost aplikacije, napravljeno je nekoliko testnih slučajeva u kojima se mjeri potrošnja energije. Svakim primjerom staviti naglasiti ćemo rješavanje drugačijeg problema koristeći barem dva moguća rješenja kako bi ustanovili koja solucija ima niže energetske zahtjeve. Rezultati primjera također sačinjavaju ovu pokaznu lekciju.

Boolean vs. boolean (billion times)

Type	AVG exec (s)	AVG CPU NRG (W)	AVG RAM NRG (W)	AVG HDD NRG (W)	Test count
boolean	0,49900	6,31915	0,00087	n/a	10000
Boolean	0,49879	6,26819	0,00071	n/a	10000

```
Data types boolean vs Boolean
boolean g = false;
 for (long i = 0 ; i<1000000000;i++){
    g = true;
}

Boolean h = false;
for (long i = 0 ; i<1000000000;i++){
    h = true;
}
```

Double vs. double (billion times)

Type	AVG exec (s)	AVG CPU NRG (W)	AVG RAM NRG (W)	AVG HDD NRG (W)	Test count
double	1,01489	6,18481	0,00096	n/a	9643
Double	5,66532	7,41853	0,01001	n/a	9294

```java
// STRING CREATOR  - StringBuilder vs. StringBuffer
StringBuilder test = new StringBuilder();
for(long i=0;i<=20000000;i++) { //100M Java heap space
    test.append(i);
}

StringBuffer stringBuffer = new StringBuffer();
for(int i=0;i<=20000000;i++) {
    stringBuffer.append(i);
}

// STRING CREATOR  -- += vs. concat
String testString = "";
for(long i=0;i<=50000;i++){
    testString = testString.concat(String.valueOf(i));
    testString += String.valueOf(i);
}
```

```java
sorting.bubbleSort();
sorting.selectionSort();
sorting.insertionSort();
sorting.quickSort();
sorting.mergeSort();
sorting.heapSort();
```

```java
matrixMultiplication.strassenAlgMultiplication();
matrixMultiplication.classicalMultiplication();
```

```java
hashGenerator.md5();
hashGenerator.sha1();
hashGenerator.sha256();
hashGenerator.sha384();
hashGenerator.sha512();
```

```java
// TreeMap vs HashMap vs LinkedHashMap 10;
TreeMap<Integer, Integer> treeMap = new TreeMap<>();
HashMap<Integer,Integer> hashMap = new HashMap<>();
LinkedHashMap<Integer, Integer> linkedHashMap = new LinkedHashMap<>();

for (int i = 0; i < 5000000; i++) {
    treeMap.put(i, v: i + 1);
    linkedHashMap.put(i, v: i + 1);
    hashMap.put(i, v: i + 1);
}
```

Literatura

[1] D. Li, W. G. J. Halfond, An investigation into energy-saving programming practices for android smartphone app development, in Proc. of the 3rd International Workshop on Green and Sustainable Software, GREENS 2014, ACM, 2014, pp. 46–53.

[2] D. Li, Y. Jin, C. Sahin, J. Clause, W. G. J. Halfond: Integrated energy-directed test suite optimization, in Proc. of the 2014 International Symposium on Software Testing and Analysis, ISSTA 2014, ACM, 2014, pp. 339–350.

[3] M. Santos, J. Saraiva, Z. Porkolab, D. Krupp: Energy Consumption Measurement of C/C++ Programs Using Clang Tooling, in Proceedings of the SQAMIA 2017: 6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Z. Budimac, ed., Belgrade, Serbia, 11-13.9.2017, Paper No. 15, 8 pages, also published online by CEUR Workshop Proceedings No. 1938 ISSN 1613-0073.

[4] J.Saraiva, M.Couto, Cs.Szabo, D.Novak: Towards Energy-Aware Coding Practices for Android, Acta Electrotechnica et Informatica, Vol. 18, No. 1, 2018, pp. 19–25. https://doi.org/10.15546/aeei-2018-0003

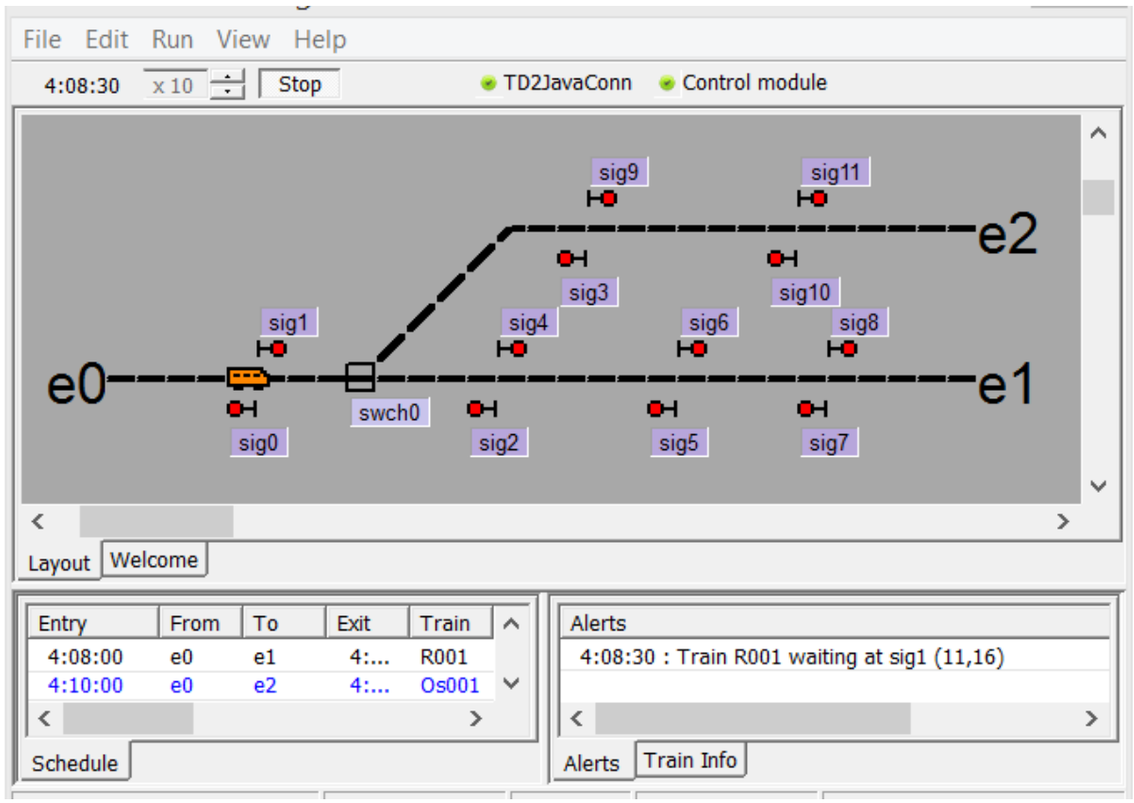
[5] Cs. Szabo, E.M.M. Alzeyani: Measuring Energy Efficiency of Selected Working Software, Studia Universitatis Babe¸s-Bolyai Informatica, Vol. 63, No. 1, 2018, pp. 5–16. https://doi.org/10.24193/subbi.2018.1.01

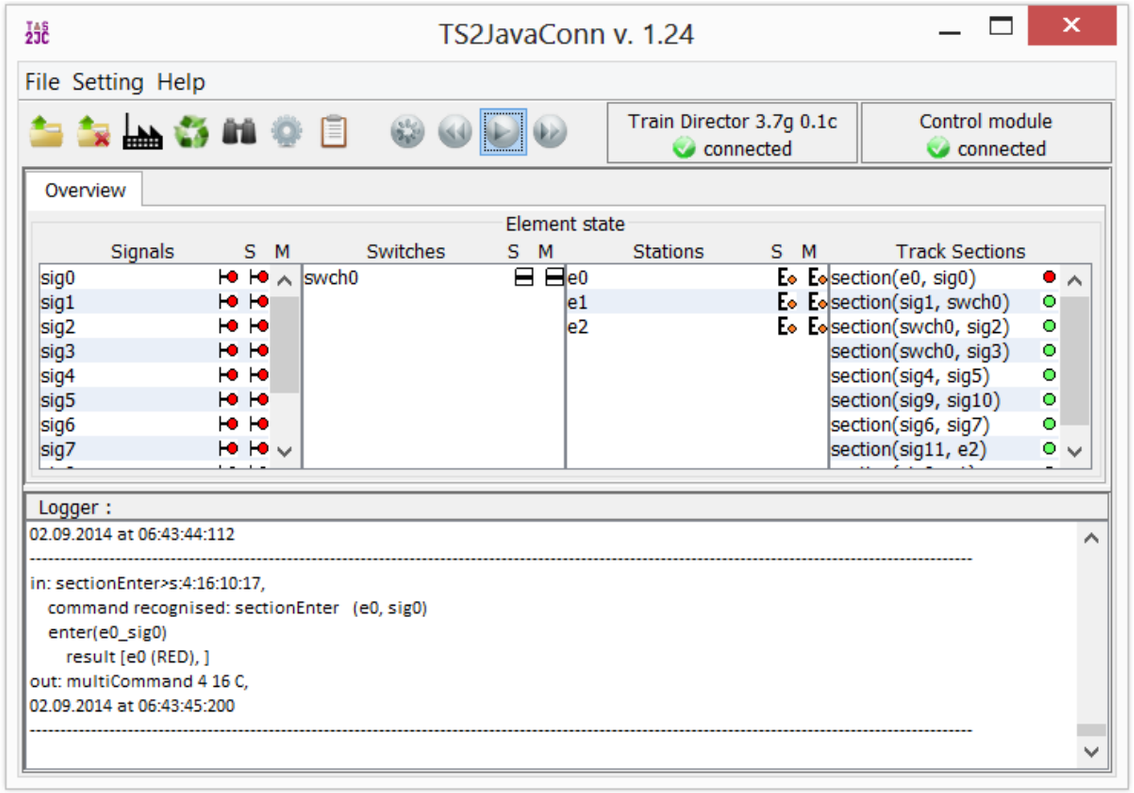
[6] Cs. Szabo, J. Saraiva: Focusing software engineering education on green application development, in Conference of Information Technology and Development of Education – ITRO 2017, Novi Sad, Serbia, pp. 165–169, ISBN 978-86-7672-302-7.

RAZVOJ ISPRAVNOG PROGRAMSKOG PROIZVODA UPOTREBOM METODE B

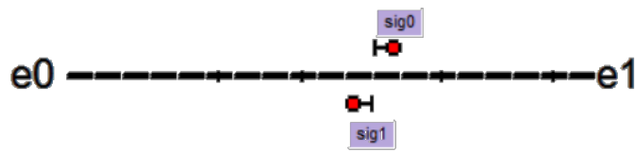
Jedan od dobro prepoznatih pristupa razvoju ispravnih programskih sustava je upotreba formalnih metoda (FM) za njihovo specificiranje i provjeru. FM su rigorozne matematički utemeljene tehnike za specifikaciju, analizu, razvoj i verifikaciju softvera i hardvera. Biti rigorozan znači da formalna metoda pruža formalni jezik s nedvosmisleno definiranom sintaksom i semantikom, a biti matematički utemeljen znači da se neki matematički aparat (formalna logika, teorija skupova itd.) Koristi za definiranje jezika.

Jedna od FM koja se koristi u industrijskoj praksi je Metoda-B [1,2,3,4], formalna metoda orijentirana modelu čija namjena je razvoj softvera. Metoda-B primarno se koristi u željezničkom sektoru, za softver kritičan na sigurnost koji djeluje u pozadini automatiziranih sustava gradske podzemne željeznice (uključujući i onaj u Budimpešti). Snaga metode-B leži u dobro definiranom razvojnom procesu koji omogućava specificiranje programskog sustava kao zbirke komponenti koje se naziva strojevima-B i preciziranje takve apstraktne specifikacije u konkretnu. Konkretna specifikacija može se automatski prevesti na ADA, C ili neki drugi programski jezik. Unutarnja konzistentnost apstraktne specifikacije i ispravnost svakog koraka preciziranja provjerava se dokazivanjem skupa predikata koji se zovu dokazne obveze (eng. proof obligation, PObs). Čitav razvojni proces, uključujući dokazivanje, podržan je programskim alatom industrijske kvalitete imena Atelier B [5].

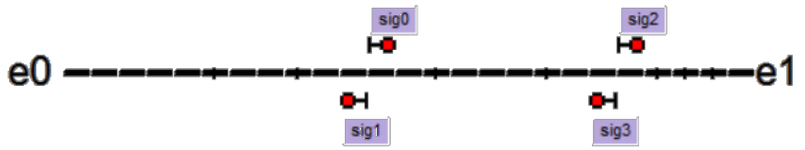




Ova pokazna vježba služi kao postepen i praktičan uvod u Metodu-B. Tijekom vježbe, polaznici će razviti jednostavan softverski kontroler za željeznički scenarij. Moći će pokrenuti scenarij s kontrolerom u skupu alata Train Director / TS2JavaConn (TD / TS2JC) [6,7]. Skup alata sastoji se od modificirane verzije simulacijske igre Train Director [8] (Slika 1) i aplikacije zvane TS2JavaConn (Slika 2), koji omogućuje upotrebu zasebno razvijenih programskih kontrolera sa simulacijskom igrom. S namjerom korištenja skupa alata za izradu prototipa kontrolera [9], kasnije se proširio prilagođenom verzijom 3D simulatora vlaka Open Rails [10].



Nakon upoznavanja s Metodom-B, polaznici tečaja dobivaju kontroler za scenarij željezničke pruge s jednim kolosijekom s dva dijela (Slika 3). Kontroler (Programski odsječak 1) napisan je jezikom Metode-B. Tijekom tečaja razvijaju se i provjeravaju kontroler za jedno-kolosiječni željeznički scenarij s tri dijela (Slika 4).



Programski odsječak 1. Kontroler za željeznički scenarij s dva odjeljka, napisana u jeziku Metode-B

```
MACHINE  route2sec

SETS

   PROP_SIGNAL={green, red};

   PROP_SECTION={free,occup}

CONCRETE_VARIABLES

   e0, e1, sig0, sig1, e0_sig1, sig0_e1

INVARIANT

   e0:PROP_SIGNAL & e1:PROP_SIGNAL &  sig0:PROP_SIGNAL & sig1:PROP_SIGNAL
   &

   e0_sig1:PROP_SECTION & sig0_e1:PROP_SECTION &

   (e0=green => sig1=red) & (sig1=green => e0=red)  &

   (e1=green => sig0=red) & (sig0=green => e1=red)  &

   (e0=green   => e0_sig1=free) & (sig1=green => e0_sig1=free) &

   (e1=green   => sig0_e1=free) & (sig0=green => sig0_e1=free)



INITIALISATION

   e0:=red || e1:=red ||    sig0:=red || sig1:=red ||    e0_sig1:= free || sig0_e1:= free

OPERATIONS

   ss <-- getSig_sig0 = BEGIN ss:=sig0 END;

   ss <-- getSig_sig1 = BEGIN ss:=sig1 END;

   ss <-- getEntry_e0 = BEGIN ss:=e0 END;

   ss <-- getEntry_e1 = BEGIN ss:=e1 END;

   reqGreen_e0 = IF sig1=red & e0_sig1=free THEN e0:=green END;

   reqGreen_e1 = IF sig0 = red & sig0_e1 = free THEN e1:=green END;

   reqGreen_sig0 = IF e1=red & sig0_e1= free THEN  sig0:=green END;

   reqGreen_sig1 = IF  e0 = red & e0_sig1 = free THEN  sig1:=green END;

   enterNI_e0_sig1 = BEGIN e0_sig1:=occup || e0:=red || sig1:=red END;

   enterIN_sig0_e1 = BEGIN sig0_e1:=occup || sig0:=red || e1:=red END;

   enterNI_e1_sig0 = BEGIN sig0_e1:=occup || sig0:=red || e1:=red END;

   enterIN_sig1_e0 = BEGIN e0_sig1:=occup || e0:=red || sig1:=red END;

   leaveNI_e0_sig1 = BEGIN e0_sig1:=free  END;

   leaveIN_sig0_e1 = BEGIN sig0_e1:=free  END;

   leaveNI_e1_sig0 = BEGIN sig0_e1:=free  END;

   leaveIN_sig1_e0 = BEGIN e0_sig1:=free  END

END
```

Literatura

1. Abrial, J. R.: The B-Book: Assigning Programs to Meanings, Cambridge University Press, 1996.

2. Abrial, J. R. : Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010.

3. Lano, K.: The B Language and Method: A Guide to Practical Formal Development, Springer-Verlag, FACIT series, 1996.

4. Schneider, S.: The B-Method: An Introduction, Palgrave Macmillan, Cornerstones of Computing series, 2001.

5. https://www.atelierb.eu/

6. Korečko, Š., Sorád, J.: Using simulation games in teaching formal methods for software development, In: Innovative Teaching Strategies and New Learning Paradigms in Computer Programming, R. Queirós, Ed., pp. 106–130, IGI Global, 2015.

7. Korečko, Š., Sorád, J., Dudláková, Z., Sobota, B.: A Toolset for Support of Teaching Formal Software Development In: Lecture Notes in Computer Science : Software Engineering and Formal Methods : 12th Internatinal Conference, SEFM 2014, pp. 278-283, Springer, 2014.

8. https://www.backerstreet.com/traindir/en/trdireng.php

9. Korečko, Š., Sobota, B.: Computer Games as Virtual Environments for Safety-Critical Software Validation, in Journal of Information and Organizational Sciences, vol. 41, no. 2, 2017.

10. http://openrails.org

PROGRAMIRANJE NAPREDNOG UPRAVLJANJA I ORKESTRACIJE VIRTUALNIH MREŽNIH RESURSA – ODABRANE STUDIJE SLUČAJA

Nove funkcije za upravljanje i orkestraciju su normirani za uporabu u raspodijeljenim i virtualiziranim mrežnim okruženjima. Njihova osnovna namjena je pružiti siguran i pouzdan rad aplikacija koje koriste mrežne funkcije. Stoga, kao nastavak naših prethodnih lekcija u kojima su predstavljeni osnovni koncepti, u ovoj lekciji predstaviti ćemo odabrane studije slučaja u kojima su navedene funkcije implementirane te ćemo objasniti napredne mehanizme u pozadini i pojednostaviti njihovu primjenu.

Literatura

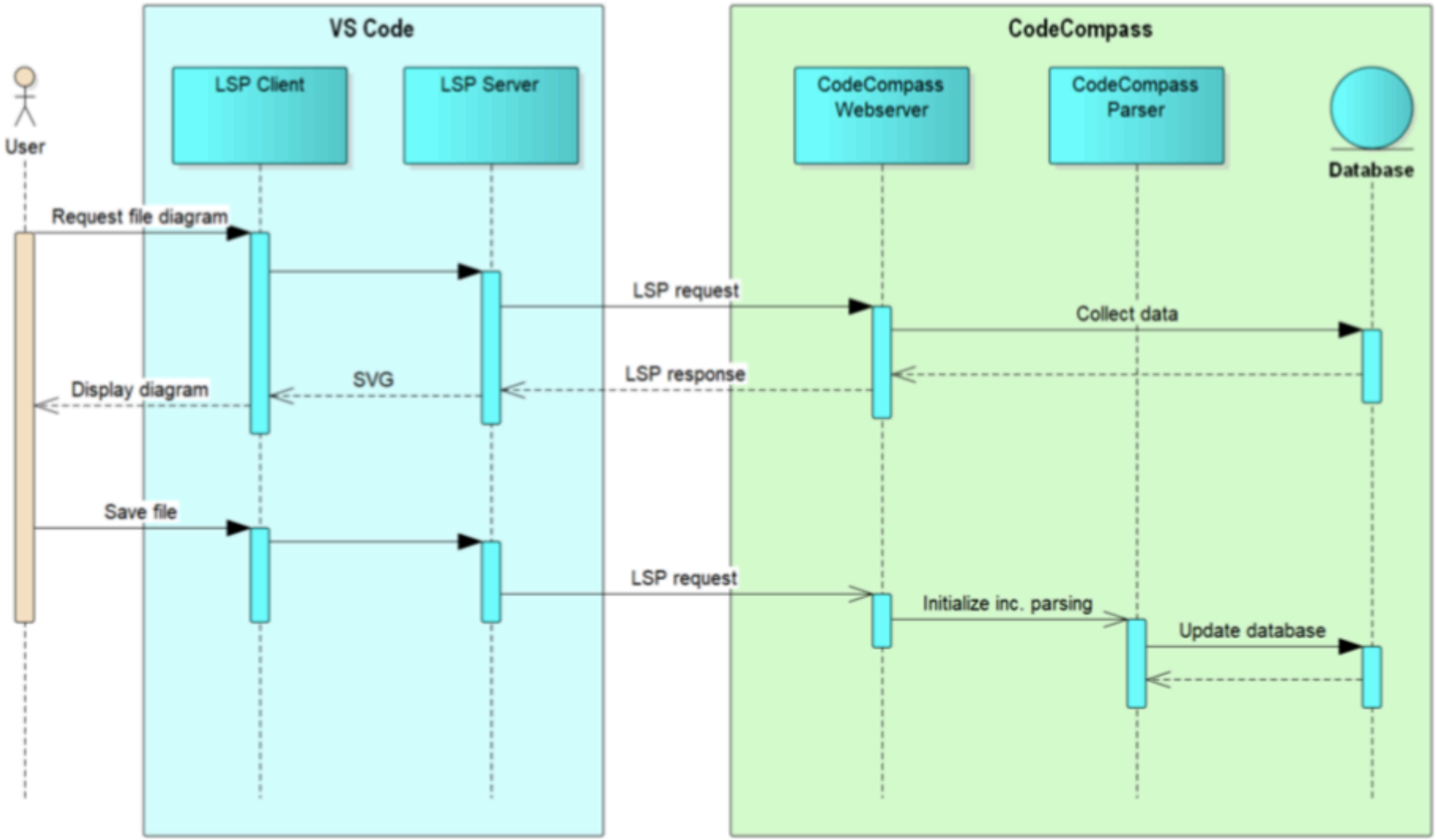
[1] ETSI Industry Specification Group (ISG) NFV: ETSI GS NFV- MAN 001 v1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration European Telecommunications Standards Institute (ETSI), 2014, https://www.etsi.org/deliver/etsi_gs/NFV- MAN/001_099/001/01.01.01 60/gs_NFV-MAN001v010101p.pdf, accessed July 1, 2018

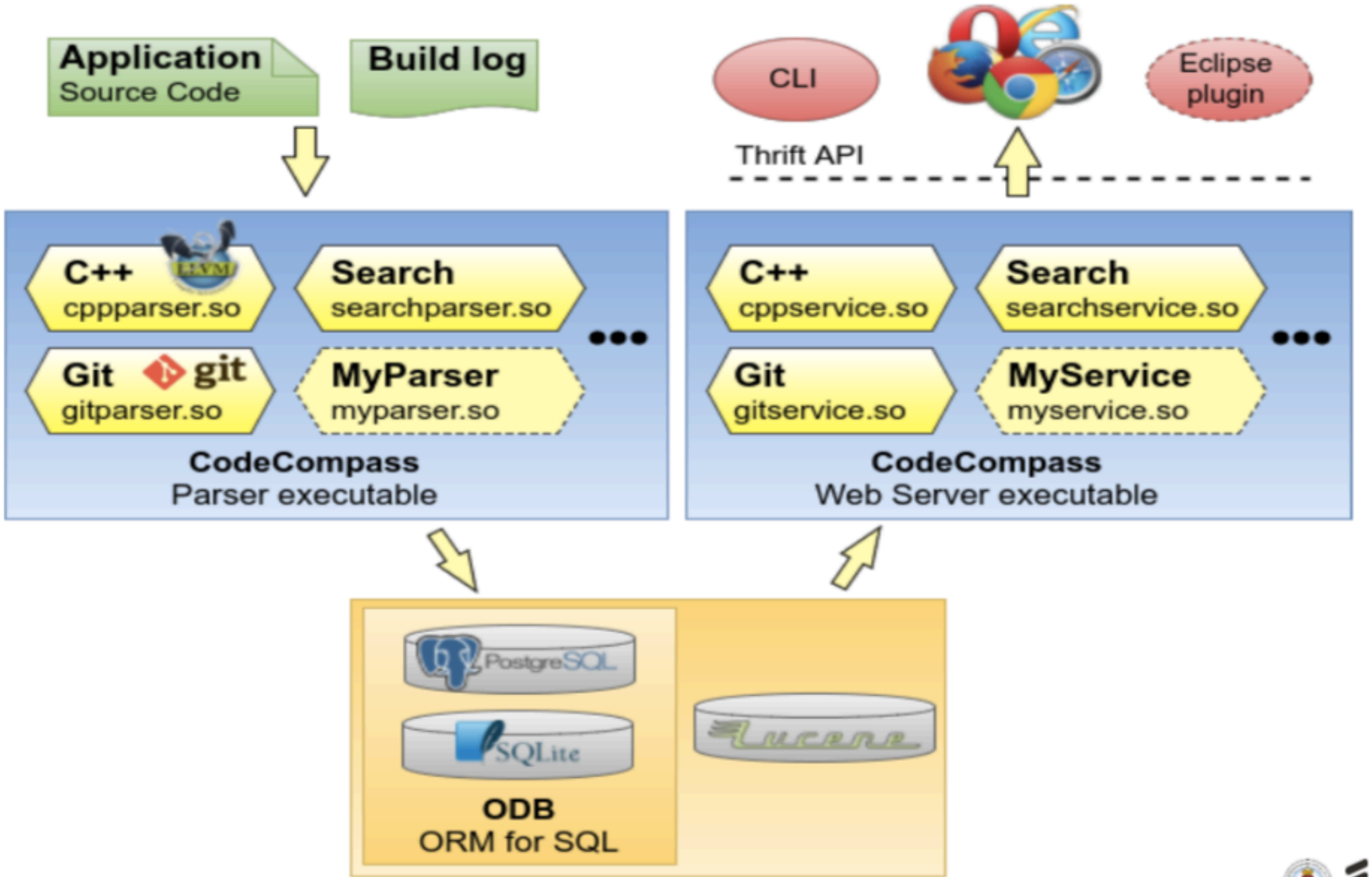
[2] Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine 53(2), 90–97 (2015)

[3] OpenStack Cloud Software. OpenStack Foundation (2018), www.openstack.org, accessed July 1, 2018

RAZUMIJEVANJE KODA POTPOMOGNUTO NAPREDNIM ALATIMA

U ovoj pokaznoj lekciji studentima ćemo predstaviti ponajbolje alate za razumijevanje programskog koda. Pružiti ćemo teoretsku podlogu za razumijevanje koda, metode za navigaciju u vizualizaciju koda te pristupe kojima ih primjenjujemo u razvoju programskog proizvoda. U praktičnom dijelu lekcije, pokazati ćemo kako se postavlja konkretni set alata: CodeCompass s inkrementalnim parsiranjem, Visual Studio Code kao ala za front-end i upotreba Language Server Protocol-a. Pri tome, parsirati ćemo knjižnice otvorenog koda te pronaći i ispraviti neispravnosti koristeći navedene skup alata.


VS Code
CodeCompass
User
LSP Client
LSP Server
CodeCompass Webserver
CodeCompass Parser
Database
Request file diagram
LSP request
Collect data
Display diagram
SVG
LSP response
Save file
LSP request
Initialize inc. parsing
Update database


Application
Source Code
Build log
CLI
Eclipse plugin
Thrift API
C++
cppparser.so
Search
searchparser.so
Git
gitparser.so
MyParser
myparser.so
C++
cppservice.so
Search
searchservice.so
Git
gitservice.so
MyService
myservice.so
CodeCompass
Parser executable
CodeCompass
Web Server executable
PostgreSQL
SQLite
lucene
ODB
ORM for SQL

```
int main(int argc, char *argv[]) {
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;
    if(
    }

    if(
    } e
    }

    cou
    ret
}
```



```
[DEBUG] [LSP] Response content:
{
    "jsonrpc": "2.0",
    "id": "1",
    "result": {
        "uri": "\/home\/bonnie\/CodeCompass\/projects\/prime.cpp",
        "range": {
            "start": {
                "line": "16",
                "character": "9"
            },
            "end": {
                "line": "16",
                "character": "10"
            }
        }
    }
}
```

Literatura

[1] Jonathan Sillito, Gail C. Murphy, Kris De Volder. (2008). Asking and Answering Questions during a Programming Change Task. IEEE Transactions on Software Engineering, VOL. 34, NO. 4, July/August 2008.

[2] Porkolab, Zoltan & Brunner, Tibor & Krupp, Daniel & Csordas, Marton. (2018). Codecompass: an open software comprehension framework for industrial usage. 361- 369. 10.1145/3196321.3197546.

[3] Nathan Hawes, Stuart Marshall, Craig Anslow. (2015). CodeSurveyor: Mapping LargeScale Software to Aid in Code Comprehension. 2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT) , 27-28 Sept. 2015.

[4] Porkolab,Zoltan & Brunner,Tibor (2018). The codecompass comprehension framework. 393-396. 10.1145/3196321.3196352

[5] CodeCompass, https://github.com/Ericsson/CodeCompass. Last accessed 5 Nov 2018.

[6] Brunner, Tibor & Porkolab, Zoltan. (2017). Two Dimensional Visualization of Soft- ware Metrics. Proceedings of the Sixth Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications.

[7] B. De Alwis and G.C. Murphy. (1998). Using Visual Momentum to Explain Dis- orientation in the Eclipse IDE. Proc. IEEE Symp. Visual Languages and Human Centric Computing, pp. 51-54, 2006.

PROGRAMIRANJE FUNKCIJSKIH POLJA UPOTREBOM SINGLE ASSIGNMENT C-A: MOGUĆNOSTI I IZAZOVI

Single Assignment C (SAC) u nekoliko je aspekata neobičan oblik funkcijskog programiranja. Kao što mu ime sugerira, SAC povezuje sintaksu nalik C-u (s mnogim vitičastim zagradama) i semantiku funkcijskog programiranja oslobođenu stanja. Originalno je bio namijenjen za lakšu prilagodbu programerima naviklim na imperativnu pozadinu, te pruža iznenađujući uvid u ono što sačinjava „uobičajeni" funkcionalni ili „uobičajeni" imperativni kostur programskog jezika.

Kao egzotična narav ovog funkcionalnog jezika, SAC naglašava višedimenzionalne nizove, umjesto popisa i stabala. Programiranje matrice obrađuje višedimenzionalne nizove na holistički način: funkcije preslikavaju potencijalno ogromne nizove argumenata kako bi rezultirali nizovima sa semantikom pozivanja po vrijednosti, a nove operacije polja definirane su sastavom postojećih. SAC je jezik visoke produktivnosti za domene primjena kod kojih se velike količine podataka obrađuju na računalno zahtjevan način.

U isto vrijeme, SAC je i jezik visokih performansi koji se natječe s imperativnim jezicima niske razine kroz tehnologiju kompilacije. Apstraktni pogled na nizove u kombinaciji s funkcionalnom semantikom podržava dalekosežne programske transformacije. Visoko optimizirani sustav za vrijeme izvođenja brine se za automatsko upravljanje memorijom s naglaskom na instantnu ponovnu upotrebu memorije. Konačno, SAC prevodilac koristi semantiku SAC-a bez stanja i podatkovnu paralelnost SAC programa za potpuno ubrzanje usmjereno kompajlerom na raznim suvremenim arhitekturama strojeva, od višejezgrenih poslužitelja do GPGPU akceleratora i klastera radne stanice.

Predavanja pružaju motivaciju za jezični dizajn SAC-a te praktični uvod u programiranje uporabom nizova kao zasebne paradigme. Proučavamo sve aspekte, od osnovnog proračuna niza do konkretnog jezičnog dizajna s programskim kodom imperativnog izgleda, diskutiramo mnoštvo primjera, istražujemo izazove sastavljanja i na kraju vidimo neke rezultate performansi na različitim paralelnim računalnim arhitekturama.

OBRASCI BALANSIRANOG RASPODIJELJENOG RAČUNARSTVA

U razvoju modernih konkurentnih programskih proizvoda ekstenzivno se koristilo različite metodologije i pristupe kako bi se postiglo ubrzanje. Međutim, paralelizacija ostaje jedna od najtežih domena, poglavito u slučaju programskih pristupa temeljenih na obrascima. Osnovna svrha je osvrnuti se na sheme paralelnog računarstva u novim okruženjima, ilustrirati prikladnost i primjenjivost u modernim postavkama raspodijeljenog računarstva. Količina paralelizma koju prikazujemo temelji se na mnogim čimbenicima kao što su: granularnost primijenjenog računalnog obrasca, semantika raspodijeljenih čvorova, prijenos podataka i naročito balansiranje opterećenja.

Literatura

[1] Zsok V.: D-Clean Semantics for Generating Distributed Computation Nodes, Work- shop on Generative Technologies, WGT 2010, Satellite workshop at ETAPS 2010, Paphos, Cyprus, March 27, 2010, pp. 77–84.

[2] Zsok V., Hernyak Z., and Horvath, Z.: Designing Distributed Computational Skeletons in D-Clean and D-Box. Central European Functional Programming School CEFP 2005, First Summer School, Budapest, Hungary, July 4-15, 2005, Revised Selected Lectures, LNCS Vol. 4164, Springer-Verlag, 2006, pp. 223–256.

[3] Zsok V., Koopman, P., Plasmeijer, R.: Generic Executable Semantics for D-Clean, Proceedings of the Third Workshop on Generative Technologies, WGT 2011, ETAPS 2011, Saarbrucken, Germany, March 27, 2011, ENTCS Vol. 279, Issue 3, Elsevier, December 2011, pp. 85–95.

[4] Zsok V., Porkolab Z.: Rapid Prototyping for Distributed D-Clean using C++ Tem- plates, Annales Universitatis Scientiarum Budapestinensis de Rolando Eotvos Nominatae, Sectio Computatorica, Eotvos Lorand University, Budapest, Hungary, 2012, Vol. 37, pp. 19–46.

[5] Zsok V. et al.: Modeling CPS Systems using Functional Programming, Proc. of IFL17, Uni. of Bristol, pp. 168–174.