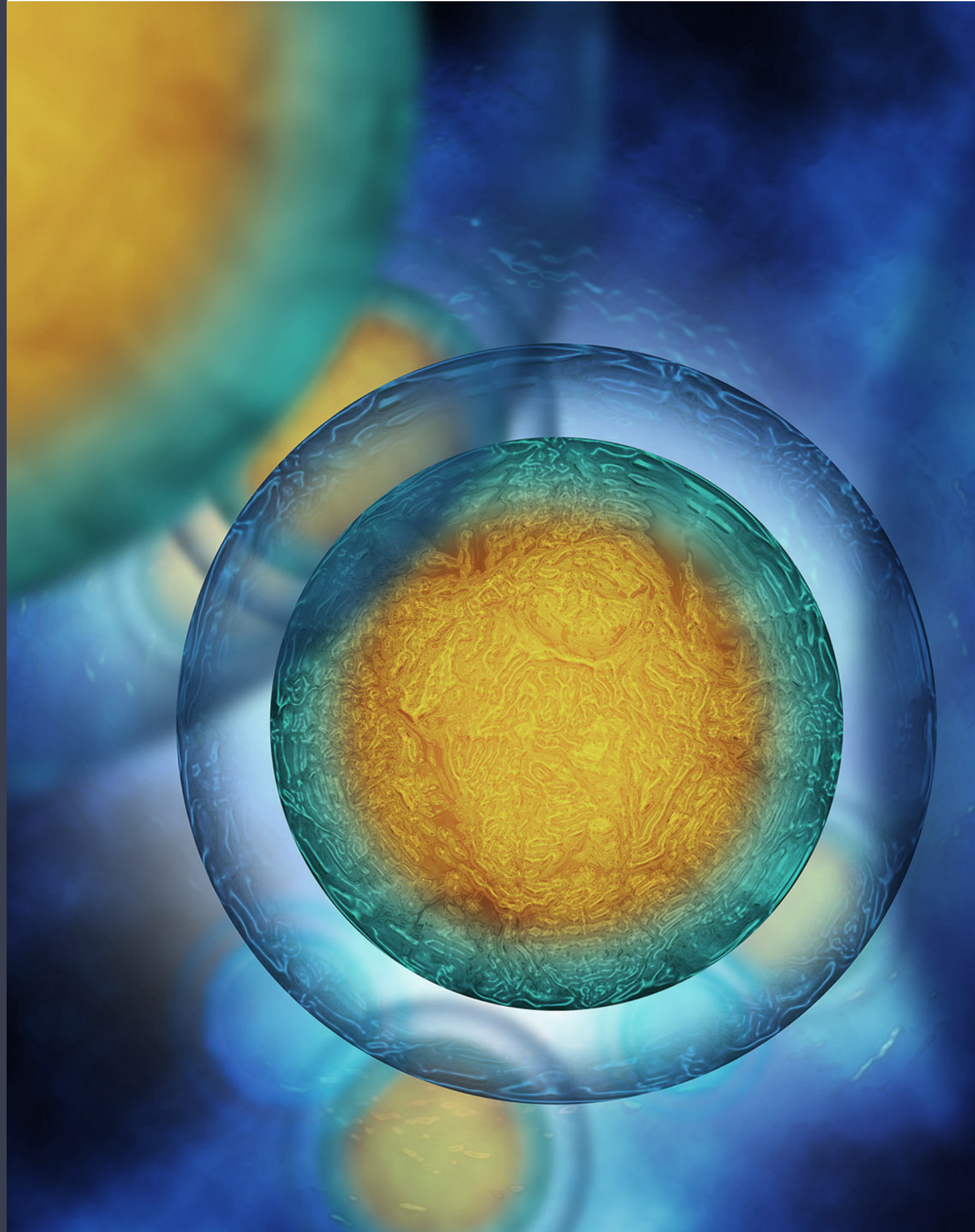
FE3CWS

MATERIÁL K LETNEJ ŠKOLE CEFP 2019

Intelektuálny výstup č.4
ERASMUS+ projektu číslo 2017-1-SK01-KA203-035402



Niekoľko slov

O OBSAHU

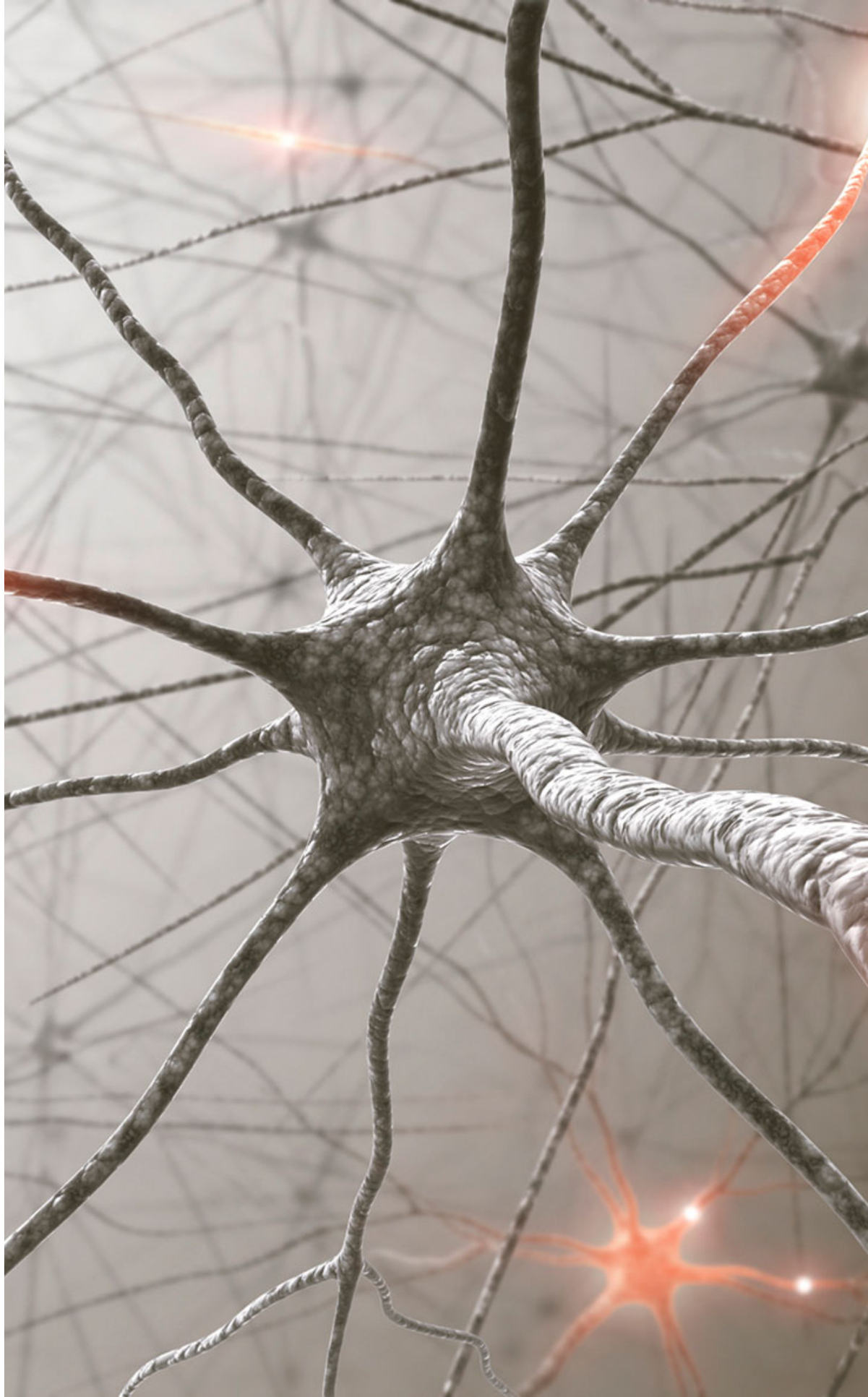
- 10 tém o kompozícii, zrozumitelnosti a korektnosti softvéru
- 20 autorov zo 7 európskych univerzít z Holandska, Chorvátska, Maďarska, Portugalska a Slovenska
- Dostupný v 7 jazykoch: anglický, maďarský, slovenský, chorvátsky, rumunský, bulharský a portugalský

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

Letná škola CEFP 2019 (Central European Functional Programming) je druhým intenzívnym školením vysokoškolských študentov a pedagógov, ktorá rozširuje komunitu Central European Functional Programming (CEFP). Uskutočnila sa v rámci implementácie ERASMUS+ projektu číslo 2017-1-SK01-KA203-035402 s názvom "Focusing Education on Composability, Comprehensibility and Correctness of Working Software", a to v období medzi 17. a 21. júnom 2019.

Materiály obsiahnuté v tejto publikácii boli vytvorené a prezentované v rámci vyššie uvedeného projektu. Táto brožúra je tlačenou obdobou intelektuálneho výstupu č. 4 tohoto projektu.

OBSAH



VIZUÁLNE PROGRAMOVANIE S VYUŽITÍM ÚLOHOVO ORIENTOVANÉHO PROGRAMOVANIA

V tomto kurze budeme vytvárať aplikácie pomocou vizuálneho asistenta pre úlohovo orientované programovanie (Task Oriented Programming, TOP). TOP je nový programovací model, ktorý vývojári môžu použiť na rýchle prototypovanie webových aplikácií pre viacerých používateľov. Hlavným spôsobom modelovania aplikácií v TOP je vytváranie úloh (tasks). Úlohy predstavujú kúsky skutočnej práce, ktorú môžu vykonávať ľudia alebo systémy. Pomocou niekoľkých operácií sa dajú spojiť do väčších a výkonnejších úloh.

Základné koncepty TOP preskúmame pomocou niekoľkých príkladov aplikácií a zároveň ukážeme, ako ich modelovať pomocou úloh vo vizuálnom vývojovom prostredí. Toto vizuálne prostredie vedie vývojárov počas procesu modelovania. Prostredie umožňuje iba rozumné spôsoby vytvárania a rozširovania úloh a poskytuje rady, ako vyriešiť typové a rozsahové chyby. Výsledkom je správny a kompilovateľný programový kód.

Študenti sú povzbudzovaní, aby počas kurzu rozšírili aplikácie, ktoré sú im poskytnuté ako príklady. Náš vizuálny prístup vyžaduje iba základné znalosti o programovaní a údajových typoch. Tento kurz je prerekvizitou kurzu o mTasks.

ÚLOHOVO ORIENTOVANÉ PROGRAMOVANIE PRE INTERNET VECÍ

Internet vecí (Internet of Things, IoT) pozostáva so zariadení, ktoré vnímajú, konajú a komunikujú s inými systémami cez Internet. Typické požiadavky na zariadenia internetu vecí sú, že musia byť lacné a musia spotrebovať málo energie. Dosahuje sa to riadením IoT zariadení malými mikroprocesormi s maličkým množstvom pamäte a minimálnym výpočtovým výkonom. Väčšina z týchto systémov nemá plnohodnotný operačný systém a iba spustia konkrétny program na vykonanie danej úlohy.

Preto je programovanie IoT zariadení veľmi náročné. Jeden program bežiaci na takomto zariadení musí „prepletať" všetky podúlohy, ako je monitorovanie vstupov, riadenie periférnych zariadení a komunikácia. Rôzne zariadenia, ktoré spolupracujú, sa musia dohodnúť na použitom protokole a musia vyriešiť notoricky známe problémy súbežného programovania.

Obsahom prednášky je praktický úvod do úlohy orientovaného programovania (TOP) pre internet vecí. V našom TOP prístupe je komunikácia medzi zariadeniami a ich servermi realizovaná transparentne systémom mTask. Celý systém je naprogramovaný ako jeden vysokoúrovňový funkcionálny program. Pre každú čiastkovú úlohu systému definujeme zodpovedajúci mTask. Tieto podúlohy môžu byť skladané do zložitejších a výkonnejších úloh pomocou. Úlohy navyše môžu kontrolovať sprostredkované hodnoty iných podúloh a komunikovať s akoukoľvek inou úlohou v systéme prostredníctvom zdieľaných zdrojov údajov (Shared Data Sources, SDS). Podúlohy pre zariadenie IoT sa do zariadenia dynamicky dodávajú a interpretujú v ňom. Silný typovací systém zabraňuje výskytu problémov za behu. Tento prístup (TOP) výrazne zjednodušuje vývoj softvéru pre internet vecí.

Literatúra

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.

NAMAĽUJTE VAŠE PROGRAMY NA ZELENO – O ENERGETICKEJ EFEKTÍVNOSTI IMPLEMENTÁCIÍ ÚDAJOVÝCH ŠTRUKTÚR

Energetická efektívnosť sa už roky týka hardvérových aj softvérových inžinierov [1], [2], [3]. Rastúci celosvetový posun smerom k udržateľnosti vrátane udržateľnosti softvéru [4] v kombinácii so systémovým charakterom energetickej účinnosti ako atribútu kvality motivoval štúdium energetického vplyvu aplikačného softvéru pri jeho vykonávaní. Táto tendencia viedla vedcov k zhodnoteniu existujúcich techník, nástrojov a jazykov pre vývoj aplikácií z energetického hľadiska. Nedávna práca skúmala vplyv, ktorý majú na energetickú účinnosť faktory, ako je nejasnosť kódu [5], volania rozhrania Android API [6], objektové orientácie kódovania [7], konštrukcie na paralelné vykonávanie [8] a typy údajových štruktúr [9]. , Analýza vplyvu rôznych faktorov na energiu je dôležitá pre vývojárov a správcov softvéru.

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

V tomto tutoriáli analyzujeme a porovnávame energetickú účinnosť rôznych implementácií pre konkrétne dátové abstrakcie, ako sú sekvencie, množiny alebo asociatívne polia. Pri každej implementácii kontrolujeme, ako operácie, ako je pridávanie, mazanie alebo vyhľadávanie prvkov, spracovávajú rôzne pracovné zaťaženia. Predmetom našej štúdie je funkcionálny programovací jazyk [10,11] a objektovo orientovaný jazyk [13,14].

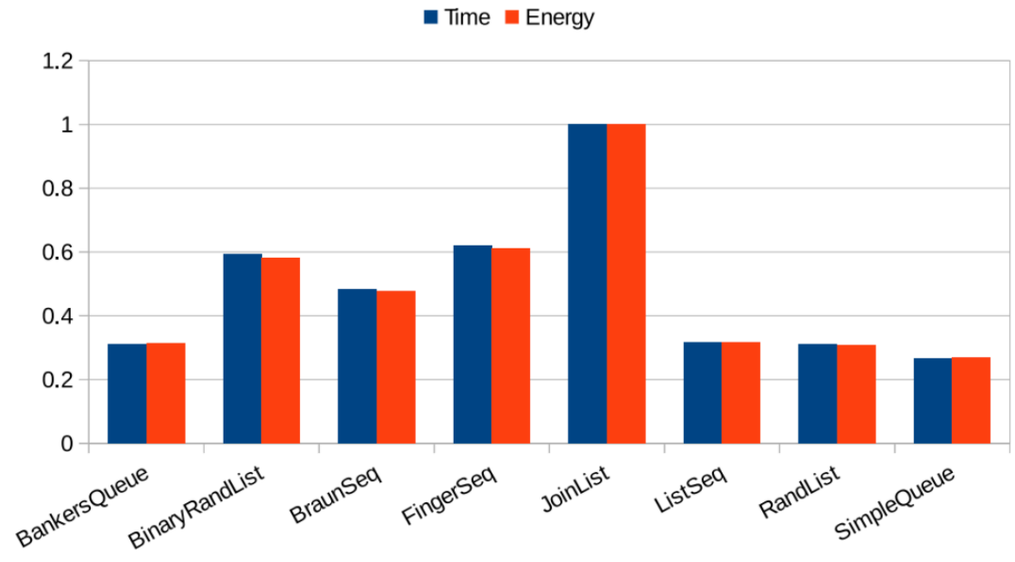
Vo funkcionálnom prostredí sme porovnali implementácie uvedené na obrázku 1, konkrétne pomocou operácií uvedených na obrázku 2.

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

V objektovo orientovanej oblasti sme analyzovali implementácie uvedené na obrázku 3, konkrétne pomocou operácií uvedených na obrázku 4.

Naším cieľom je poskytnúť vývojárom akcie, ktoré sú už integrované v podporných nástrojoch a ktoré môžu viesť k zelenej konštrukcii softvéru. Dokázali sme, že tá istá operácia, ktorá je k dispozícii v rôznych implementáciách, sa môže výrazne líšiť z hľadiska výkonnosti aj spotreby energie. Ako príklad na obrázku 5 zobrazujeme výsledky operácie odstraňovania implementácií abstrakcie sekvencií, ktoré sú k dispozícii v Edisonovej knižnici jazyka Haskell.

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...	

Referencie

[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

ZELENÝ SOFTVÉR V INŽINIERSKOM KURZE

Trvalo udržateľný rozvoj sa stáva čoraz dôležitejšou témou nielen vo svetovej politike, ale aj v inžinierstve. Ako ukazujú viaceré nedávne výskumné štúdie, softvéroví inžinieri nie sú výnimkou. Napriek intenzívnemu výskumu v oblasti ekologického softvéru sa dnešné vysokoškolské vzdelávanie v oblasti výpočtovej techniky často zlyháva v upriamení pozornosti na našu environmentálnou zodpovednosť. V tejto výučbovej aktivite predstavíme modul zeleného softvéru, ktorý je súčasťou nášho pokročilého kurzu softvérového inžinierstva. Predstavujeme katalóg energetických „vôní“ a zelených refaktorov, ktoré podľa našich predbežných výsledkov pomáhajú študentom v uvažovaní nad a optimalizácii energetickej spotreby softvérových systémov.

Referencie

[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

ENERGETICKÁ PROFILÁCIA SOFTVÉROVÝCH APLIKÁCIÍ PRE JAVA PROJEKTY

Tento tutoriál sa zameriava na energetickú efektívnosť softvérových aplikácií napísaných v programovacom jazyku Java. Prvá časť popisuje súčasný stav v oblasti energetického profilovania a možnosti zobrazovania spotreby energie segmentov zdrojového kódu. Ďalej je uvedená špecifická metóda analýzy kódu na zobrazovanie informácií, ktorá sa týka procesora, operačnej pamäte a pevného disku. Po tejto analýze nasleduje krátky úvod do implementovanej Java aplikácie. Aby sa demonštrovalo praktické použitie aplikácie, vytvára sa niekoľko testovacích riešení, kde sa meria spotreba energie. V každom príklade kladieme dôraz na vyriešenie jedného problému pomocou najmenej dvoch riešení, aby sme určili, ktorá implementácia má nižšiu energetickú náročnosť. Výsledky príkladov sú tiež súčasťou tohto tutoriálu.

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);
}
```

Referencie

[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

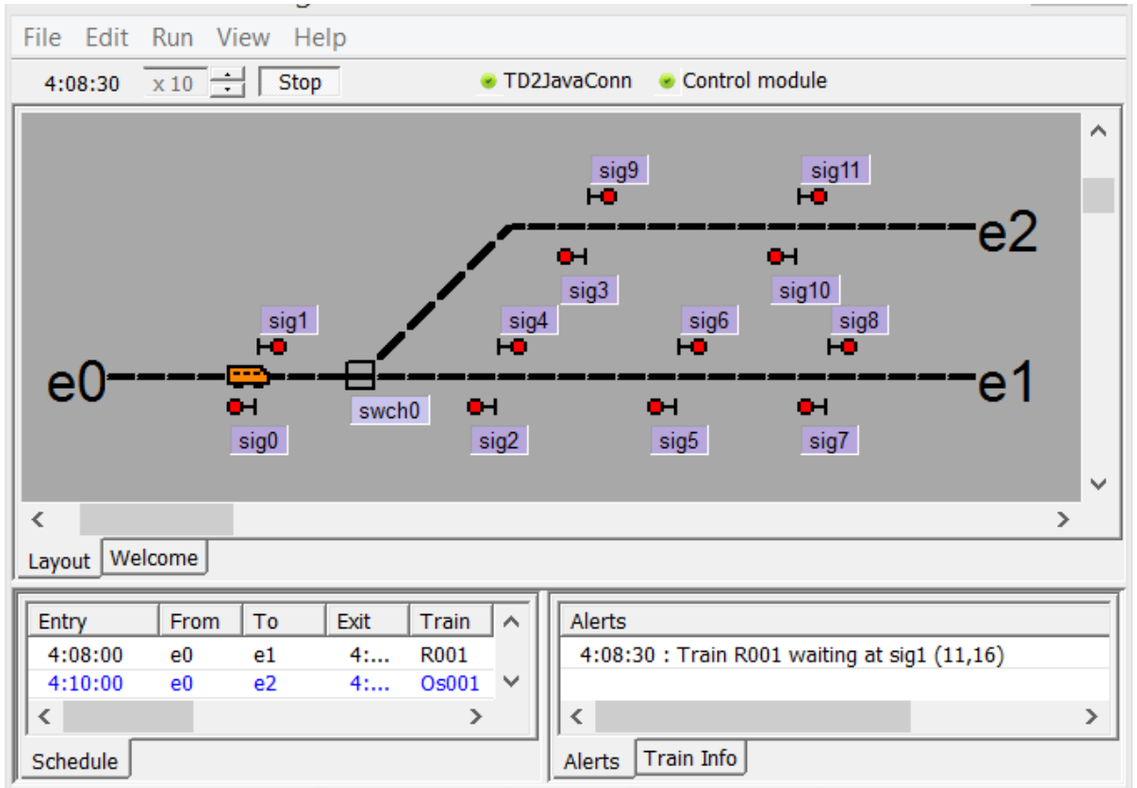
[5] Cs. Szabo, E.M.M. Alzeyani: Measuring Energy Efficiency of Selected Working Software, Studia Universitatis Babe¸s-Bolyai Informatica, Vol. 63, No. 1, 2018, pp. 5–16. https://doi.org/10.24193/subbi.2018.1.01

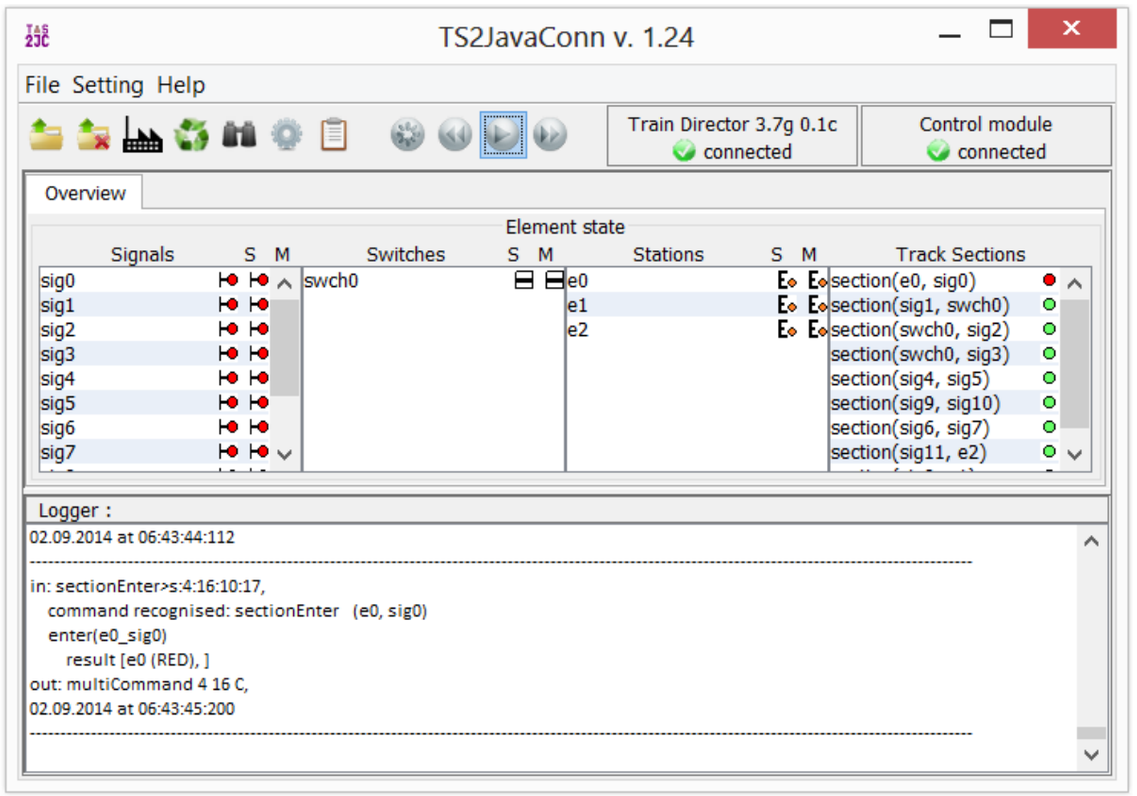
[6] Cs. Szabo, J. Saraiva: Focusing software engineering education on green application development, in Conference of Information Technology and Development of Education – ITRO 2017, Novi Sad, Serbia, pp. 165–169, ISBN 978-86-7672-302-7.

VÝVOJ KOREKTNÉHO SOFTVÉRU POMOCOU B-METÓDY

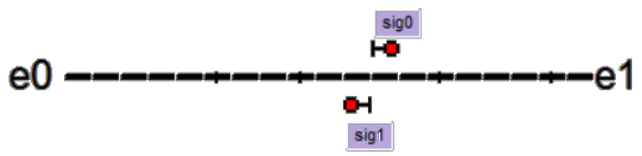
Jedným z uznávaných prístupov k vývoju korektných softvérových systémov je použitie formálnych metód na ich špecifikáciu a verifikáciu. Formálne metódy (FM), sú rigorózne techniky, postavené na matematickom základe, ktoré slúžia na špecifikáciu, analýzu, vývoj a verifikáciu hardvéru a softvéru. Slovo „rigorózne“ tu znamená, že každá FM poskytuje formálny jazyk s jednoznačne definovanou syntaxou a sémantikou. Matematickým základom je matematický aparát, ako formálna logika či teória množín, použitý na definovanie tohto formálneho jazyka.

Jednou z FM používaných v priemyselnej praxi je B-metóda (B-Method) [1,2,3,4]. B-metóda je stavovo založená a modelovo orientovaná FM, ktorá sa primárne používa v železničnom sektore. Tu bola použitá na vývoj z hľadiska bezpečnosti kritického softvéru pre automatizované linky metra (vrátane linky 4 v Budapešti). Sila B-metódy spočíva v jej dobre definovanom vývojovom procese, ktorý umožňuje špecifikovať softvérový systém ako kolekciu komponentov nazývaných B-stroje a následne zjemniť (tzn. konkretizovať) túto abstraktnú špecifikáciu do implementovateľnej podoby. Implementovateľnú podobu je potom možné automaticky preložiť do programovacích jazykov ako ADA, C alebo Java. Vnútornú konzistentnosť abstraktnej špecifikácie a korektnosť každého kroku zjemnenia je možné overiť matematickým dôkazom skupiny formúl, nazývaných povinné dôkazy. Celý vývojový proces, vrátane dôkazov, je podporovaný vývojovým prostredím pre B-metódu, nazývaným Atelier B [5].

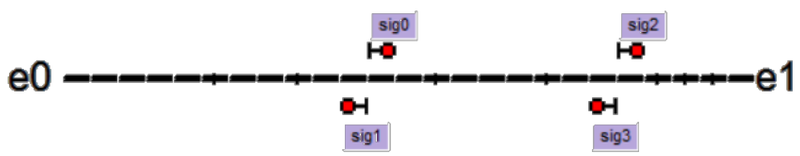




Tento krátky kurz slúži ako jemný a praktický úvod do B-metódy. Počas neho účastníci vyvinú jednoduchý riadiaci program pre železničnú trať. Vytvorený program budú môcť aj spustiť a to pomocou špeciálneho softvéru, nazvaného Train Director/ TS2JavaConn (TD/TS2JC) [6,7]. Softvér pozostáva z upravenej simulačnej hry Train Director [8] (Obr. 1) a aplikácie TS2JavaConn (Obr. 2), ktorá umožňuje použitie osobitne vyvinutého riadiaceho softvéru so simulačnou hrou. S cieľom použitia tohto softvéru pre prototypovanie riadiacich modulov, bol TD/TS2JC rozšírený [9] o upravenú verziu 3D vlakového simulátora Open Rails [10].



Po úvode do B-metódy účastníci kurzu dostanú riadiaci program pre jednokoľajnú trať s dvoma úsekmi (Obr. 3). Tento riadiaci program (Výpis 1) je napísaný v jazyku B-metódy. Počas kurzu účastníci vyvinú a overia riadiaci program pre jednokoľajnú trať s troma úsekmi (Obr. 4).



Výpis 1. Riadiaci program pre jednokoľajnú trať s dvoma úsekmi, napísaný v jazyku B-metódy.

```
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
```

Referencie

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

PROGRAMOVANIE POKROČILÉHO RIADENIA A ORCHESTRÁCIE VIRTUALIZOVANÝCH SIEŤOVÝCH ZDROJOV – VÝBER PRÍPADOVÝCH ŠTÚDIÍ

Nové riadiace a orchestračné (MANO) funkcie sú štandardizované pre použitie v distribuovaných virtualizovaných sieťových prostrediach. Ich hlavnou úlohou je poskytovať bezpečnú a spoľahlivú prevádzku aplikácií pomocou sieťových funkcií. Preto, ako pokračovanie našej predchádzajúcej prednášky, kde sme sa zaoberali základnými pojmami, sa teraz budeme venovať prípadovým štúdiám, v ktorých sú tieto funkcie implementované, a vysvetlíme si za nimi skryté pokročilé mechanizmy a jednoduchosť ich aplikácie.

Referencie

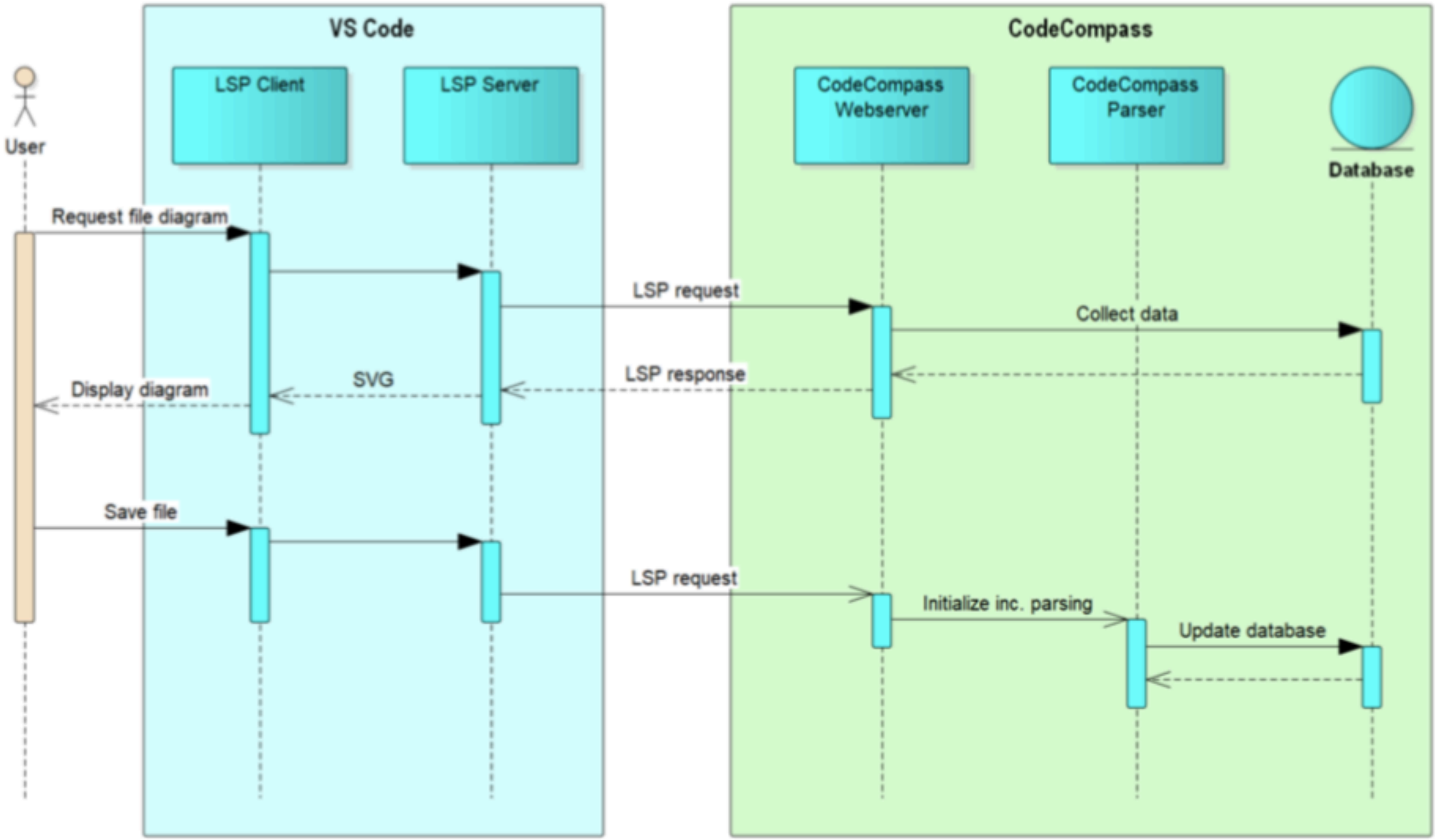
[1] ETSI Industry Specification Group (ISG) NFV: ETSI GS NFV- MAN 001 v1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration European Telecommunications Standards Institute (ETSI), 2014, https://www.etsi.org/deliver/etsi_gs/NFV- MAN/001_099/001/01.01.01 60/gs_NFV-MAN001v010101p.pdf,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

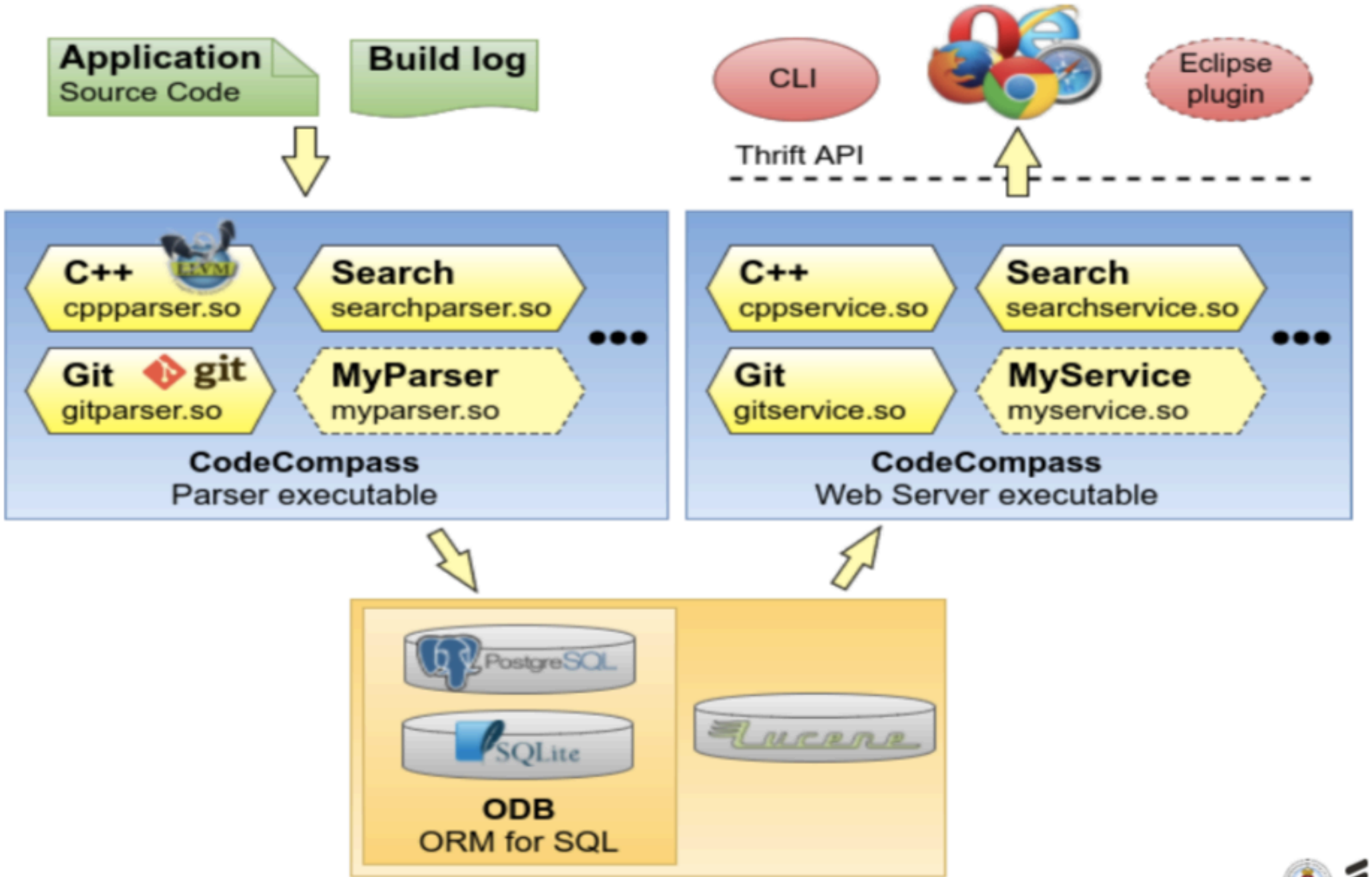
[2] Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine 53(2), 90–97 (2015)

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

POROZUMENIE KÓDU S POKROČILOU NÁSTROJOVOU PODPOROU

V tomto tutoriále študentom predstavíme najmodernejšie nástroje na porozumenie kódu. Poskytneme teoretické základy pre porozumenie kódu, metódy navigácie a vizualizácie kódov a prístupy na ich použitie v praktickom vývoji softvéru. V praktickej časti si ukážeme, ako nastaviť konkrétnu sadu nástrojov: CodeCompass s prírastkovou syntaktickou analýzou, Visual Studio Code ako front-end nástroj a použitie protokolu Language Server Protocol. Syntakticky budeme analyzovať vybranú knižnicu s otvoreným zdrojovým kódom a v nej nájdeme a opravíme konkrétnu chybu pomocou tejto sady nástrojov.


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"
            }
        }
    }
}
```

Referencie

[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.

FUNKCIONÁLNE VEKTOROVÉ PROGRAMOVANIE S SINGLE ASSIGNMENT C: PRÍLEŽITOSTI A VÝZVY

SAC (Single Assignment C) je vo viacerých ohľadoch neobvyklý funkcionálny programovací jazyk. Ako už názov napovedá, SAC kombinuje syntax podobnú jazyku C (s množstvom zložených zátvoriek) s bezstavovou, čisto funkcionálnou, sémantikou. Táto voľba, ktorá bola pôvodne motivovaná snahou uľahčiť prijatie jazyka programátormi, ktorí programovali v imperatívnych jazykoch, ponúka prekvapujúci vhľad do toho, čo predstavuje „typický" funkcionálny alebo „typický" imperatívny konštrukt programovacieho jazyka. Taktiež exoticky z pohľadu funkcionálnych jazykov, SAC preferuje namiesto zoznamov a stromov viacrozmerné polia. Vektorové programovanie (programovanie polí) zaobchádza s mnohorozmernými poliami holistickým spôsobom: funkcie mapujú potenciálne obrovské polia argumentov, aby vyústili do polí so sémantikou Call-by-Value, a nové operácie poľa sú definované skladaním existujúcich. SAC je vysoko produktívny jazyk pre aplikačné domény, kde prebiehajú intenzívne výpočty nad veľkými sadami údajov.

Vďaka svojej technológii kompilácie je SAC tiež vysoko výkonným jazykom, konkurujúcim nízkoúrovňovým imperatívnym jazykom. Abstraktný pohľad na polia kombinovaný s funkcionálnou sémantikou podporuje ďalekosiahle transformácie programov. Vysoko optimalizovaný runtime systém sa stará o automatickú správu pamäte so zameraním na okamžité opätovné použitie pamäte. V neposlednom rade kompilátor SAC využíva bezstavovú sémantiku SAC a údajovo paralelnú povahu SAC programov na plne kompilátorom riadené zrýchlenie pre veľké množstvo súčasných počítačových architektúr, od viacjadrových serverov po GPGPU akcelerátory a sieťové zoskupenia pracovných staníc.

Táto výučbová aktivita poskytuje praktický úvod do vektorového programovania so SAC ako paradigmy. Pozrieme sa na všetky aspekty od základného počtu polí po konkrétny jazykový dizajn s imperatívne vyzerajúcim hľadaním funkcionálnym kódom, budeme diskutovať o množstve príkladov, skúmať výzvy spojené s kompiláciou a nakoniec uvidíme niektoré výkonnostné výsledky na viacerých paralelných počítačových architektúrach.

VYVÁŽENÉ DISTRIBUOVANÉ VÝPOČTOVÉ VZORY

Najmodernejšie techniky vývoja súbežného softvéru v značnej miere využívajú rôzne metodiky a prístupy na dosiahnutie vysokého zrýchlenia. Paralelizmus však zostáva jednou z najťažších oblastí, najmä v prípade programovacích prístupov založených na vzoroch. Hlavným cieľom tejto výučbovej aktivity je preskúmať paralelné výpočtové schémy v novom prostredí, za účelom ilustrovania vhodnosti a použiteľnosti v nových distribuovaných výpočtových situáciách. Množstvo rovnobežnosti sa skúma na základe mnohých faktorov, ako napríklad: použitá výpočtová schéma zjemnenej granularity, sémantika distribuovaných uzlov, streamovanie údajov a najmä vyrovnávanie záťaže.

Referencie

[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.