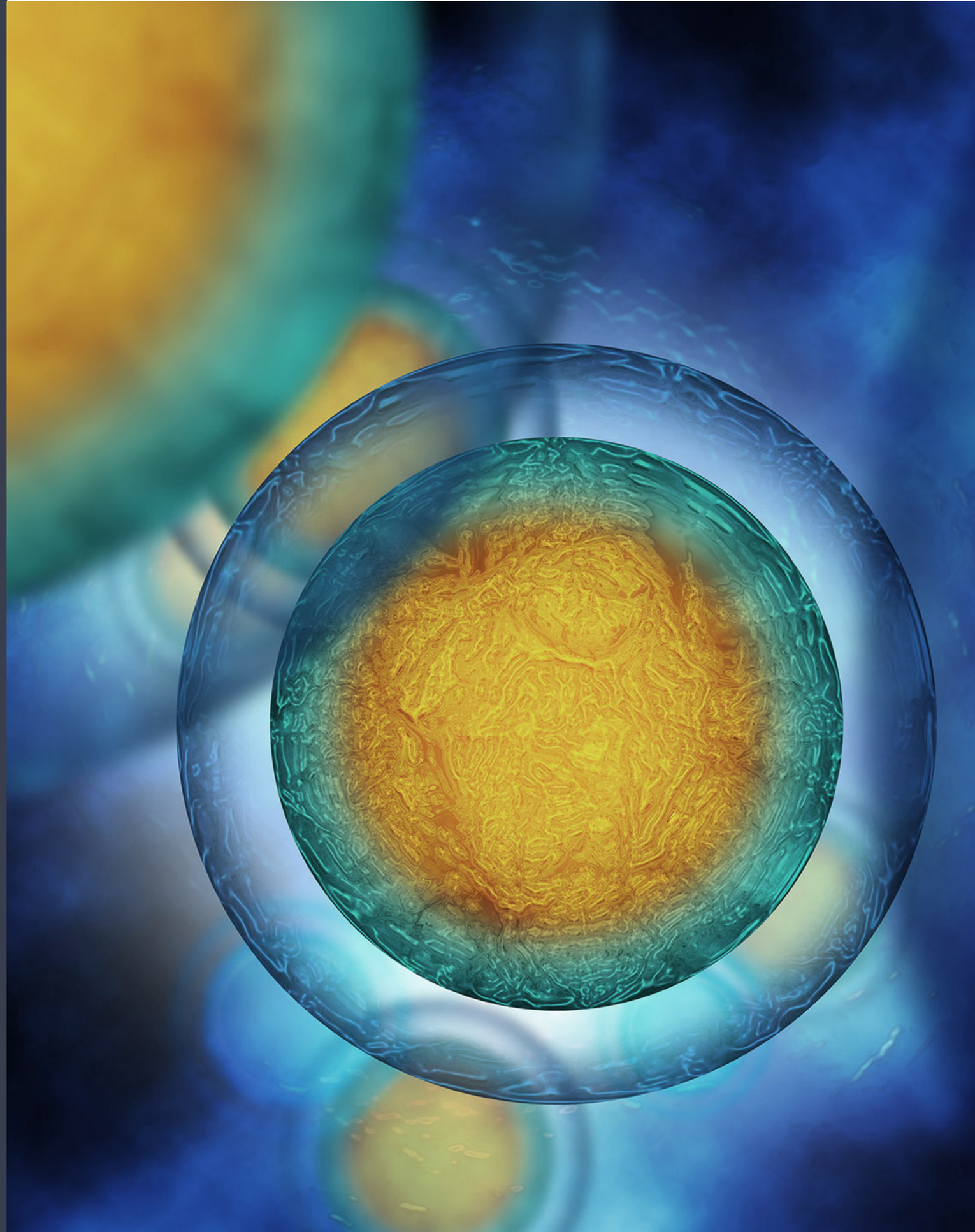
FE3CWS

MATERIAL DIDACTIC PENTRU ȘCOALA DE ȚEFP 2019

Cod:
Output intelectual nr. 4 al proiectului Erasmus+ 2017-1-SK01-KA203-035402



CUPRINS

abreviat:

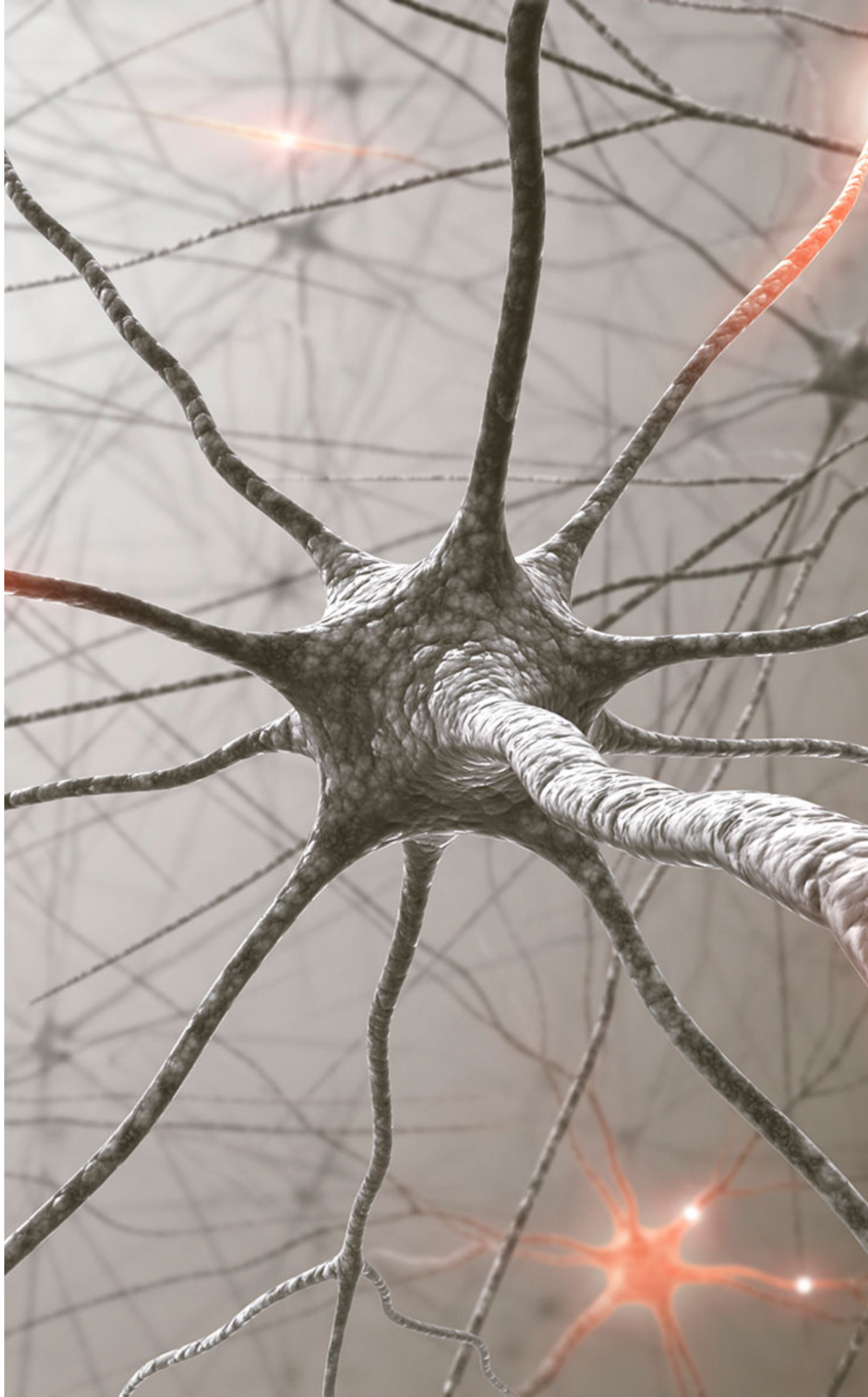
- **10 teme, legate de dezvoltarea programelor complexe: compoziția sistemelor, inteligibilitatea codului, corectitudinea codurilor**
- **20 autori din 7 universități europene din Croația, Olanda, Portugalia, Slovacia și Ungaria**
- **Materialul didactic este disponibil în 7 limbi: bulgară, croată, engleză, maghiară, portugheză, română, slovacă**

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

Școala de vară de programare funcțională centrală europeană (CEFP) este al doilea program intensiv pentru studenții de învățământ superior și personalul didactic care extinde comunitatea școlii de vară de programare funcțională central-europeană (CEFP) în cadrul proiectului ERASMUS + nr. 2017-1-SK01 -KA203-035402 „Educația focalizată pe compozibilitate, înțelegere și corectitudinea software-ului de lucru", care a avut loc între 17 și 21 iunie 2019.

Materialul prezentat este produs în cadrul proiectului menționat mai sus. Publicația prezentă este formatul gata de tipar al output-ului intelectual "O4".

CUPRINS



PROTOTIPIZARE VIZUALĂ FOLOSIND PROGRAMARE ORIENTATĂ SPRE TASK-URI (TASK-ORIENTED PROGRAMMING)

În acest curs creăm aplicații folosind un asistent vizual pentru programarea orientată spre sarcini (TOP). TOP este o nouă paradigmă de programare pe care dezvoltatorii o pot utiliza pentru prototipul rapid al aplicațiilor web cu mai mulți utilizatori. Modul central de modelare a aplicațiilor în TOP este prin crearea de Sarcini – task-uri. Sarcinile reprezintă aspecte a lumii reale instrumentate de oameni sau de sisteme. Folosind câteva operații, acestea pot fi combinate în sisteme mari. Vom explora conceptele de bază ale paradigmei TOP prin studiul unor exemple, arătând modul de modelare folosind task-uri și un mediu de dezvoltare vizuală. Mediul vizual ghidează dezvoltatorii în modelare. Instrumentul acceptă doar operații legitime de a crea și extinde task-uri și oferă indicii spre rezolvarea erorilor de tip și de extindere. Sistemul rezultat este un program corect.

Studenții sunt încurajați să extindă exemplele de aplicații într-o sesiune practică. Abordarea noastră vizuală necesită doar cunoștințe de bază privind programarea și tipurile de date. Introducerea pe TOP și principiile sale de modelare sunt o condiție prealabilă a cursului pe mTasks.

PROGRAMARE ORIENTATĂ SPRE SARCINI – TOP – PENTRU IOT

Internet of Things (IoT) sunt colecția de dispozitive care măsoară, observă, acționează și comunică cu alte sisteme de pe internet. Caracteristicile tipice dispozitivelor IoT sunt prețul mic și consumul redus. Acest lucru este posibil de microprocesoare mici, cu memorie minimă și putere redusă de procesare. Majoritatea acestor sisteme nu au un sistem de operare ci rulează doar un program specific pentru a executa sarcinilor dorite.

Constrângerile de sus fac ca programarea IoT să fie provocatoare: programul de pe un astfel de dispozitiv trebuie să interacționeze toate subtask-urile, cum ar fi monitorizarea intrărilor, controlul perifericelor și comunicarea. Diferitele dispozitive care cooperează trebuie să aibă un protocolul comun de comunicare și trebuie să rezolve problemele de concurență ivite.

În cadrul prezentării vom oferi o aplicație a programării orientate spre sarcini (TOP) pentru IoT. În abordarea noastră, comunicarea dintre dispozitive și servere este gestionată transparent de sistemul mTask. Întregul sistem este programat într-un singur program funcțional de nivel înalt. Pentru fiecare subtask al sistemului definim un proces mTask corespunzător. Aceste subtask-uri pot fi compuse cu combinatori pentru a ajunge la task-uri mai complexe. Aceste task-uri pot folosi valorile intermediare ale altor subtask-uri, precum și pot comunica cu orice alt task din sistem prin intermediul surselor de date partajate (SDS). Subtask-urile pentru un dispozitiv IoT sunt livrate dinamic pe dispozitiv și interpretate acolo. Această abordare TOP simplifică foarte mult dezvoltarea de software pentru IoT.

Literatură

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.

VOPSIȚI PROGRAMELE ÎN VERDE - DESPRE EFICIENȚA ENERGETICĂ A IMPLEMENTĂRILOR STRUCTURII DE DATE

Eficiența energetică a fost o preocupare atât pentru inginerii hardware și software de nivel scăzut de ani de zile [1], [2], [3]. Cu toate acestea, mișcarea tot mai mare la nivel mondial către durabilitate, inclusiv durabilitatea în software [4], combinată cu natura sistemică a eficienței energetice ca atribut de calitate au motivat studiul impactului energetic al aplicațiilor software în execuție. Această tendință i-a determinat pe cercetători să evalueze tehnicile, instrumentele și limbajele existente pentru dezvoltarea aplicațiilor dintr-o perspectivă centrată în energie. Lucrări recente au studiat efectul pe care factori precum obuscarea codului [5], apeluri API Android [6], refactorieri de coduri orientate pe obiect [7], construcții pentru execuție concomitentă [8] și tipuri de date [9] asupra eficienței energetice . Analiza impactului diverșilor factori asupra energiei este importantă pentru dezvoltatorii de programe și software.

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

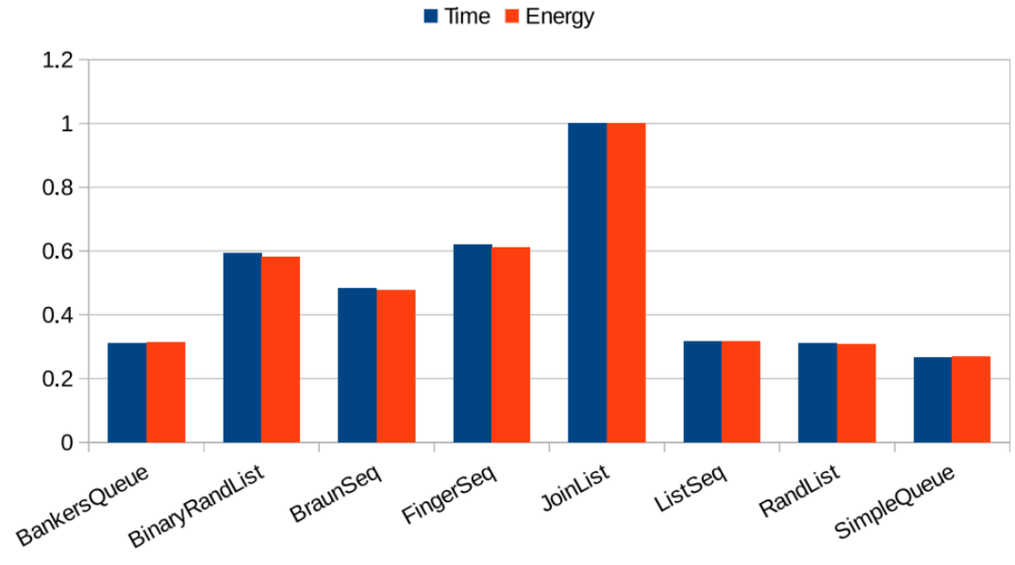
iters	operation	base	aux
1	add	100000	100000
1000	addAll	100000	1000
1	clear	100000	n.a.
1000	contains	100000	1
5000	containsAll	100000	1000
1	iterator	100000	n.a.
10000	remove	100000	1
10	removeAll	100000	1000
5000	toArray	100000	n.a.
10	retainAll	100000	1000

În această prezentare analizăm și comparăm eficiența energetică a diferitelor implementări pentru abstractizări de date, precum secvențe, seturi sau colecții asociative. Pentru fiecare implementare, inspectăm modul în care operațiunile cum ar fi adăugarea, ștergerea sau căutarea de elemente gestionează sarcini diferite. Subiectele studiului nostru sunt un limbaj de programare funcțional[10,11] și unul orientat pe obiecte[13,14].

Într-un cadru funcțional, am comparat implementările prezentate în figura 1 și anume folosirea operațiilor prezentate în figura 2. În tărâmul orientat pe obiect, am analizat implementările prezentate în figura 3, și anume folosind operațiile prezentate în figura 4.

Scopul nostru este de a oferi dezvoltatorilor informații care să poată fi acționate deja integrate în instrumentele de sprijin și care pot orienta construcția de software verde. Am putut să arătăm că aceeași operație pusă la dispoziție în implementări diferite poate diferi semnificativ atât în ceea ce privește timpul de rulare, cât și consumul de energie. Ca exemplu, în figura 5 prezentăm rezultatele operațiunii de eliminare a implementărilor de abstractizare a secvențelor disponibile în Biblioteca Edison a lui 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...	

Literatură

[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

SOFTWARE VERDE ÎNTR-UN CURS DE INGINERIE

Dezvoltarea durabilă a devenit o temă din ce în ce mai importantă nu numai în politica mondială, ci și o temă din ce în ce mai importantă pentru ingineri. Inginerii software nu fac excepție, așa cum se arată în diverse studii de cercetare recente. În ciuda cercetării intense asupra software-ului ecologic, învățământul informatic de astăzi nu reușește să abordeze responsabilitatea noastră pentru mediu. Prezentăm un modul pe software verde pe care l-am realizat ca parte a unui curs avansat de inginerie software. Vă prezentăm un catalog de arhetipuri de energie – energy smells – și metode de refactorare spre verde. Rezultatele preliminare arată impactul pozitiv asupra studenților care sunt capabili să optimizeze consumul de energie al sistemelor software.

Literatură

[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

PROFILARE DE ENERGIE PENTRU APLICAȚII SOFTWARE PENTRU PROIECTE JAVA

Acest tutorial abordează eficiența energetică a aplicațiilor software implementate în limbajul de programare Java. Prima parte descrie stadiul actual al tehnologiei în profilarea de energie, precum și opțiunile pentru afișarea consumului de energie a segmentelor codului sursă. În continuare, este prezentată o metodă de analiză a codului personalizat pentru afișarea informațiilor care se adresează procesorului, memoriei de operare și hard disk-ului. După această analiză, urmează o scurtă introducere în aplicația Java implementată. Pentru a demonstra utilizarea practică a aplicației, sunt create mai multe soluții de testare, unde am măsurat consumul de energie. Cu fiecare exemplu, punem accent pe rezolvarea unei probleme cu cel puțin două soluții pentru a determina care implementare are o intensitate energetică mai mică. Rezultatele exemplelor fac parte și din acest tutorial.

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);
}
```

Literatură

[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

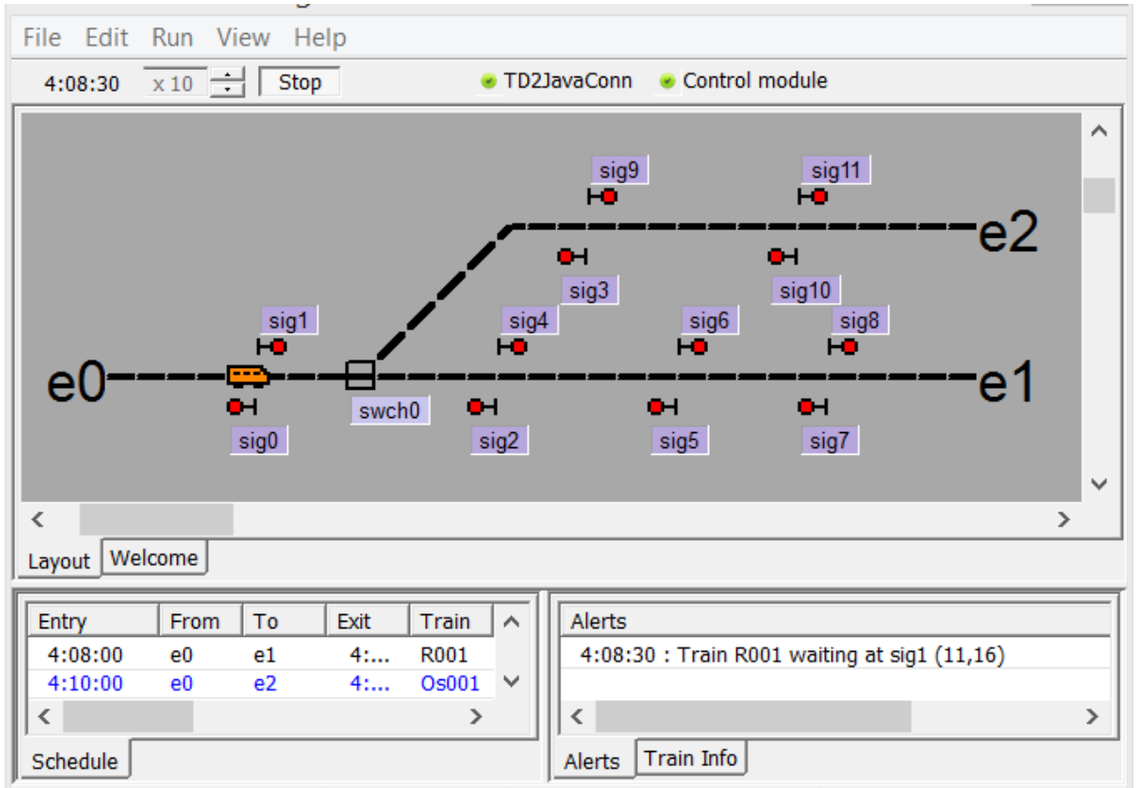
[6] Cs. Szabo, J. Saraiva: Focusing software engineering education on green application development, in Conference of Information Technology and Development of Education – ITRO 2017, Novi Sad, Serbia, pp. 165–169, ISBN 978-86-7672-302-7.

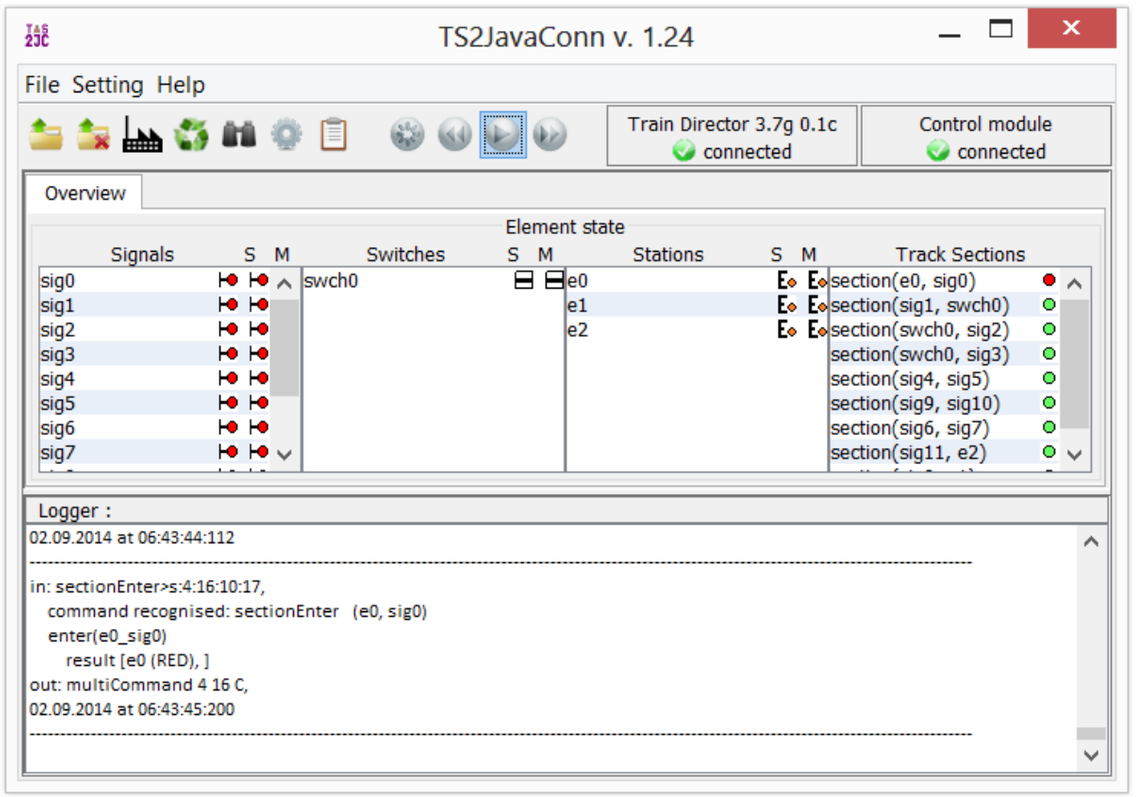
DEZVOLTAREA SOFTWARE-ULUI CORECT CU METODA B

Una dintre abordările bine recunoscute ale dezvoltării sistemelor software corecte este utilizarea metodelor formale – formal methods (FM) – pentru specificarea și verificarea lor. FM sunt tehnici riguroase pentru specificarea, analiza, dezvoltarea și verificarea software-ului și hardware-ului. Riguros înseamnă că o metodă formală furnizează un limbaj formal cu sintaxă și semantică delimitată folosit ca un aparat matematic (logică formală, teoria seturilor) pentru a defini limbajul.

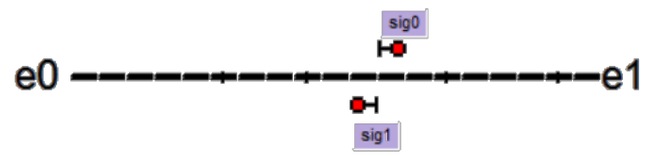
Unul dintre FM-urile utilizat în practica industrială este B-Method, o metodă formală bazată pe stări, destinată dezvoltării de software. B-Metod este folosită în principal în sectorul feroviar pentru sisteme software critice pt siguranță – safety-critical systems – în sisteme de metrou urbane automatizat (inclusiv cel din Budapesta). Puterea metodei B constă într-un proces de dezvoltare bine definit, care permite specificarea unui sistem software ca o colecție de componente numite ”mașini B” și să rafineze o astfel de specificație abstractă la una concretă. Specificația concretă poate fi tradusă automat în ADA, C sau alt limbaj de programare. Consistența internă a specificației abstracte și corectitudinea fiecărei etape sunt verificate prin dovedirea unui set de predicate numite obligații de probă (PObs). Întregul proces de dezvoltare, inclusiv dovedirea, este susținut de un software, numit Atelier B.

Acest tutorial servește ca o introducere practică a metodei B. În timpul tutorialului, participanții vor dezvolta un controler software simplu pentru un scenariu feroviar. Aceștia vor putea rula scenariul cu controlorul într-un set de instrumente care conține joc de simulare corespunzător.

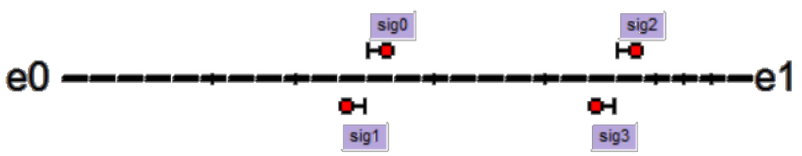




Acest tutorial servește ca o introducere blândă, practică, a metodei B. În timpul tutorialului, participanții vor dezvolta un controler software simplu pentru un scenariu feroviar. Aceștia vor putea rula scenariul cu controlerul din setul de instrumente Train Director / TS2JavaConn (TD / TS2JC) [6,7]. Setul de instrumente constă dintr-o versiune modificată a jocului de simulare Train Director [8] (Fig. 1) și o aplicație numită TS2JavaConn (Fig. 2), care permite utilizarea controlerelor software dezvoltate separat cu jocul de simulare. Cu ambiția de a utiliza setul de instrumente pentru prototiparea controlerului, a fost extins ulterior [9] printr-o versiune personalizată a simulatorului de trenuri 3D Open Rails [10].



După o introducere a metodei B, participanților la curs li se oferă un controlor pentru un scenariu feroviar cu o singură cale cu două secțiuni (Fig.3). Controlerul este scris în limba B-Metodei. În timpul cursului, ei dezvoltă și verifică un controler pentru un scenariu feroviar cu o singură cale cu trei secțiuni (Fig. 4).



Listare 1. Controlerul pentru scenariul feroviar cu două secțiuni, scrise în limba Metodei 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
```

Literatură

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

MANAGEMENTUL AVANSAT ȘI A ORCHESTRAREA RESURSELOR DE REȚEA VIRTUALIZATE - STUDII DE CAZ

Noile funcții de gestionare și orchestrare (MANO) sunt standardizate pentru utilizare în medii de rețea distribuite și virtualizate. Rolul lor principal este de a asigura funcționarea sigură și fiabilă a aplicațiilor folosind funcții de rețea. Prin urmare, ca continuare a prelegerii noastre anterioare unde oferim concepte de bază, aici în această prelegere vom oferi o selecție de studii de caz în care aceste funcții sunt puse în aplicare și vom explica mecanismele avansate din spatele și simplitatea aplicării lor.

Literatură

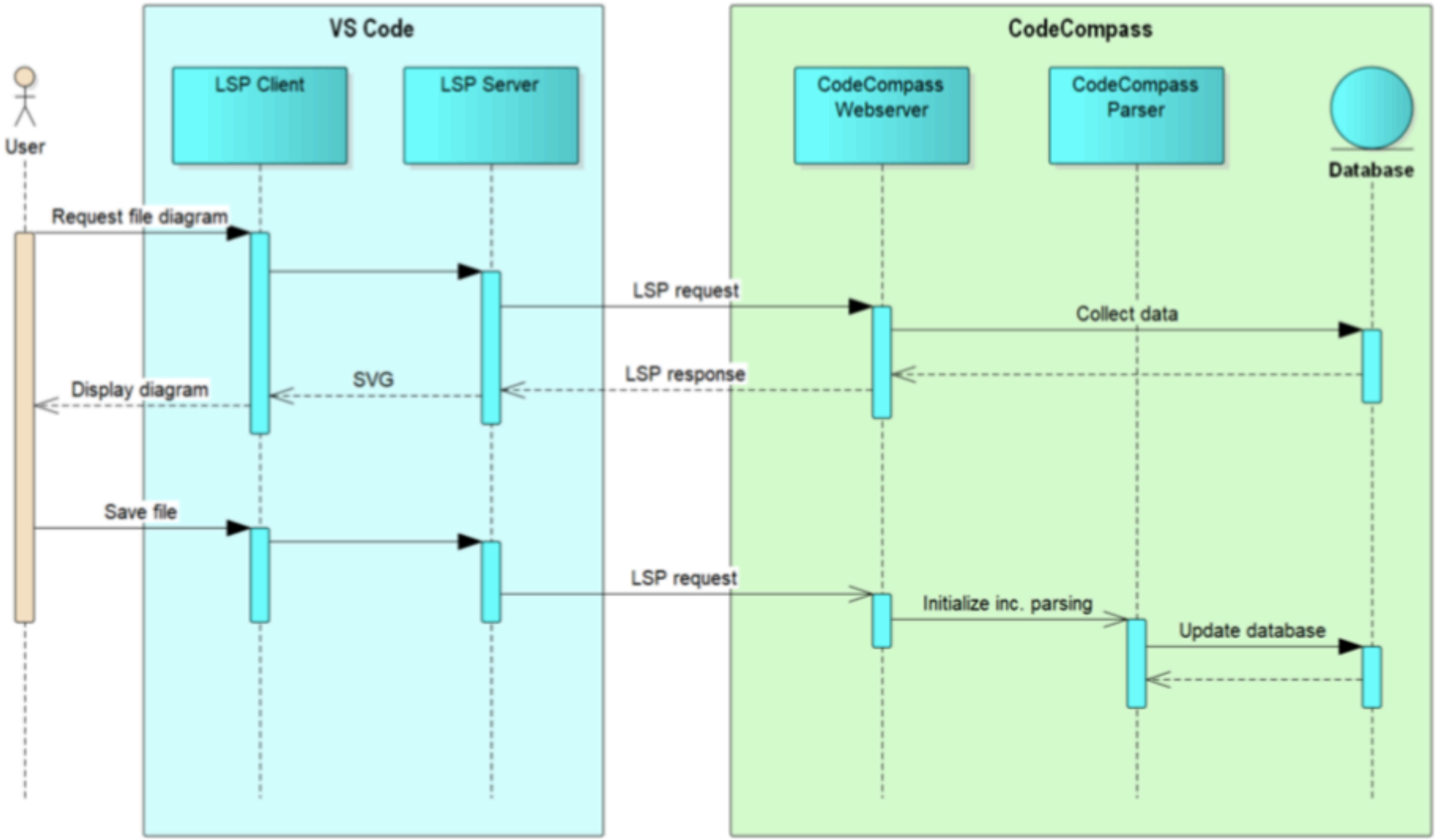
[1] ETSI Industry Specification Group (ISG) NFV: ETSI GS NFV- MAN 001 v1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration European Telecommunications Standards Institute (ETSI), 2014, https://www.etsi.org/deliver/etsi_gs/NFV- MAN/001_099/001/01.01.01 60/gs_NFV-MAN001v010101p.pdf,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

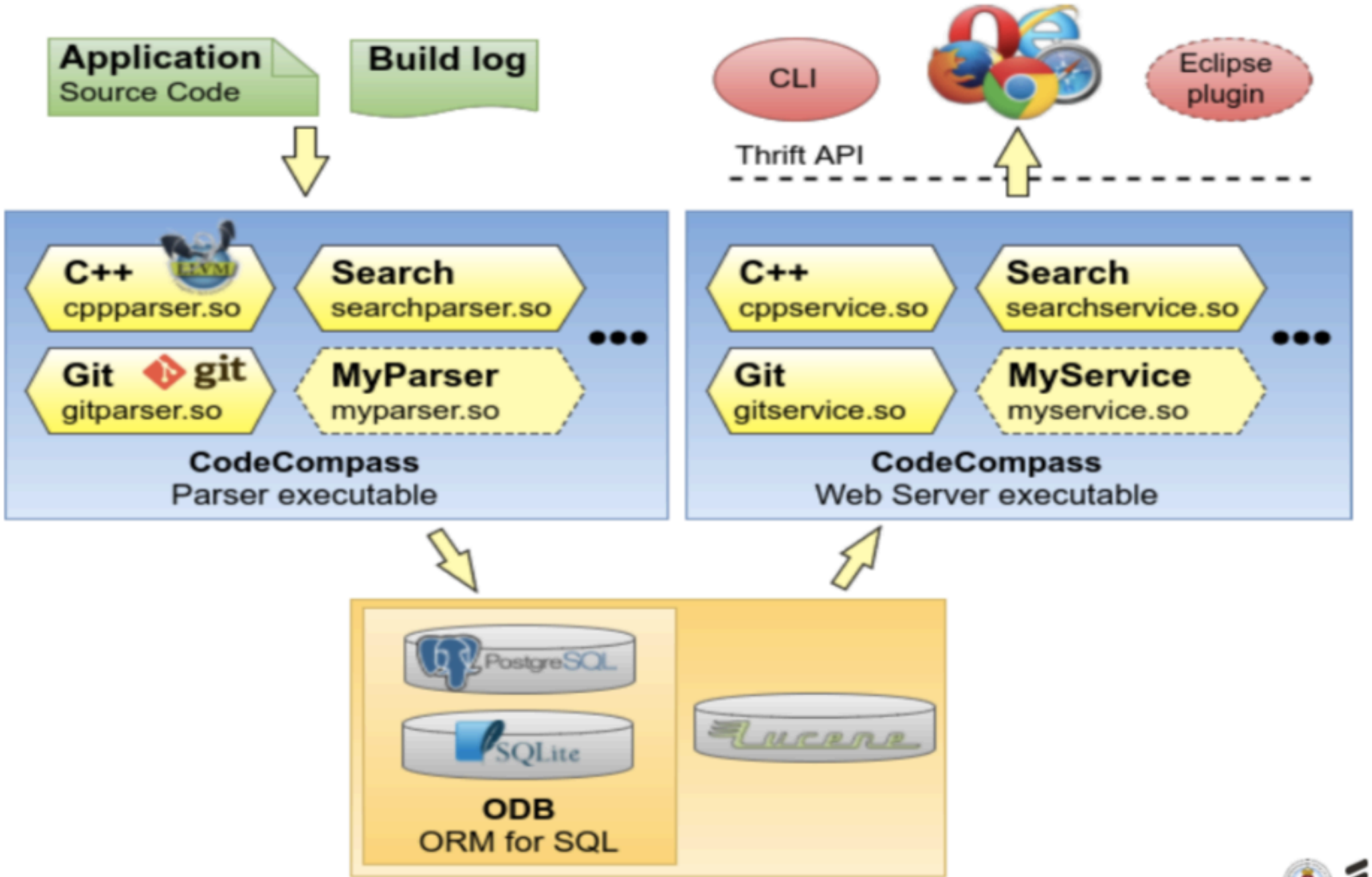
[2] Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine 53(2), 90–97 (2015)

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

ÎNȚELEGEREA CODULUI CU SPRIJIN AVANSAT PENTRU INSTRUMENTE

În acest tutorial vom prezenta studenților instrumente de înțelegere a codului de artă. Vom oferi o bază teoretică pentru înțelegerea codului, metode de navigare și vizualizare de coduri pentru a le aplica în dezvoltarea de software practic. În sesiunea de practică, vom demonstra cum să configurați un set de instrumente specifice: CodeCompass cu analiza incrementală, Visual Studio Code ca instrument front-end și utilizarea Protocolului Server Language. Vom analiza o bibliotecă open source și vom găsi și repara un bug specific folosind setul de instrumente.


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"
            }
        }
    }
}
```

Literatură

[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.

PROGRAMARE FUNCȚIONALĂ CU SISTEM DE ATRIBUIRE UNIC C: OPORTUNITĂȚI ȘI PROVOCĂRI

SAC (single assignment C) este – prin mai multe aspecte – un limbaj funcțional de programare ieșit din comun. După cum sugerează și numele, SAC combină sintaxa limbajului C cu o semantică pur funcțională fără stare. Motivată inițial pentru a ușura adopția de către programatori "imperativi", alegerea SAC oferă perspective surprinzătoare despre elementele „tipic" funcționale sau tipic „imperative". Contrar celor obișnuite unui limbaj funcțional, SAC pune accent pe matrici multidimensionale în loc de liste sau arbori. Programarea bazat pe matrici tratează matricile multidimensionale într-un mod holistic: definite ca funcții cu transmitere a parametrilor ca valori și noile operații pe matrici sunt definite prin compoziția celor existente. SAC este un limbaj de înaltă productivitate pentru domenii de aplicații care folosesc colecții mari de date.

În același timp, SAC este, de asemenea, un limbaj performant care concurează cu limbaje imperative de nivel scăzut prin tehnologia de compilare. Modul de definire abstractă a tabelelor, combinat cu semantica funcțională permite transformări optime ale programului. Un sistem de rulare extrem de optimizat are grijă de gestionarea automată a memoriei, cu accent pe reutilizarea imediată a memoriei. Nu în ultimul rând, compilatorul SAC exploatează semantica fără stări a SAC și natura paralelă a datelor programelor SAC pentru accelerații complet dirijate de compilator pe o mare varietate de arhitecturi de mașini contemporane, de la servere cu mai multe nuclee până la acceleratoare și clustere GPGPU și stații de lucru.

Motivăm conceperea limbajului SAC și oferim o introducere directă a programării cu matrici ca o paradigmă nouă. Analizăm toate aspectele, de la calculul matricii subiacente la proiectarea limbajului concret, cu cod funcțional cu aspect imperativ, discutăm o multitudine de exemple, explorăm provocări de compilare și, în cele din urmă, vedem rezultate de performanță pe diverse arhitecturi paralele de calcul.

TIPARE ECHILIBRATE DE CALCUL DISTRIBUIT

Dezvoltarea de software simultan de ultimă generație a făcut uz larg de diferite metodologii și abordări pentru a obține o viteză mare. Totuși, paralelismul rămâne unul dintre cele mai dificile domenii, în special în cazul abordărilor de programare bazate pe tipare. Scopul principal este explorarea schemelor de calcul paralele într-un mediu nou, pentru a ilustra adecvarea și aplicabilitatea în noi setări de calcul distribuite. Cantitatea de paralelism este explorată pe baza multor factori precum: granularitatea rafinată a modelului de calcul aplicat, semantica nodurilor distribuite, fluxul de date și mai ales echilibrarea sarcinii.

Literatură

[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.