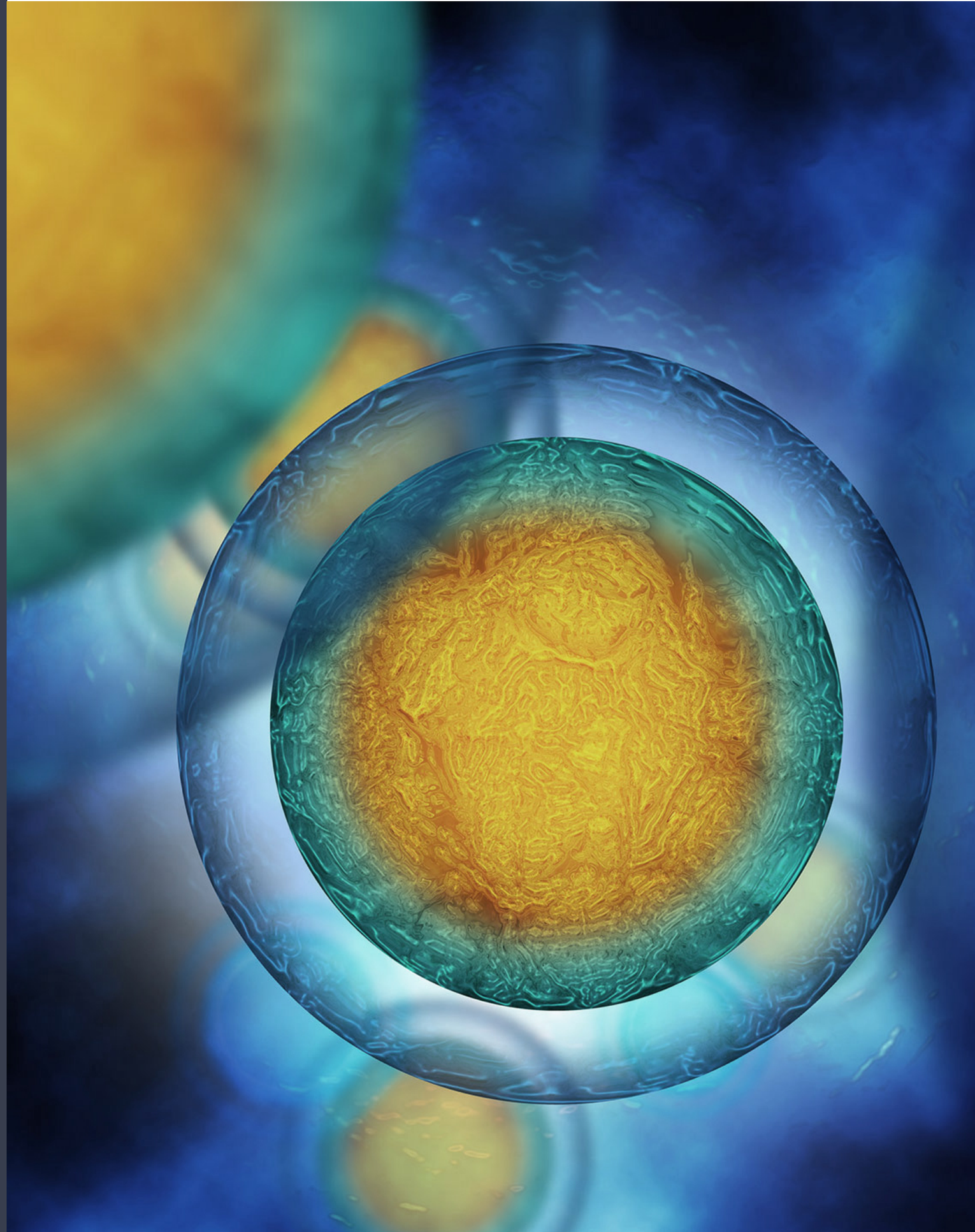
FE3CWS

A CEFP 2019 NYÁRI ISKOLA TANANYAGA

A 2017-1-SK01-KA203-035402 számú ERASMUS+ projekt O4 megjelölésű intelektuális kimenete



BEVEZETÉS

gyanánt

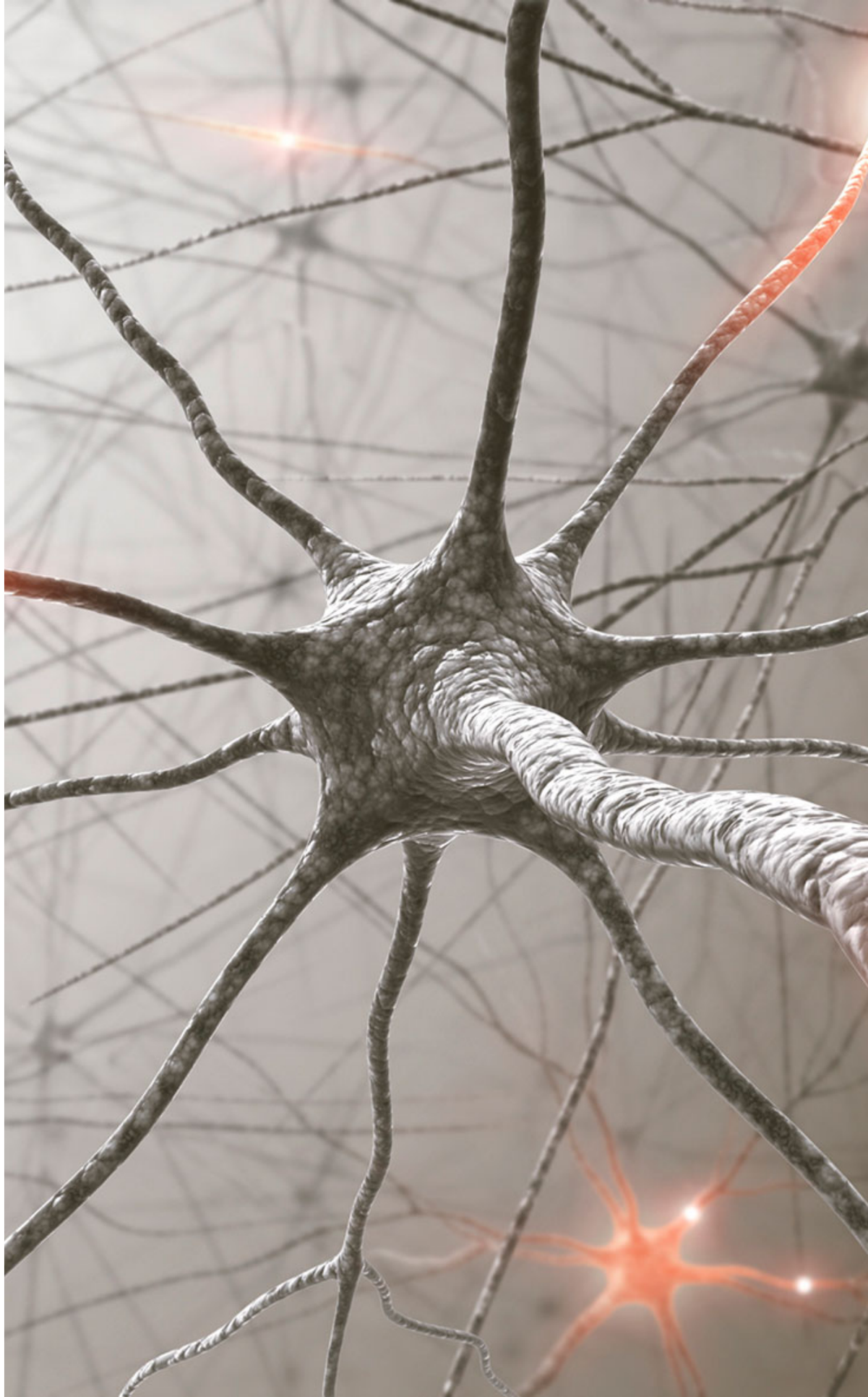
- **10 téma szoftverfejlesztésről, -megértésről és helyességről**
- **20 szerző Európa 7 egyeteméről, Horvátországból, Magyarországról, Hollandiából, Portugáliából és Szlovákiából**
- **Elérhető 7 nyelven: angolul, magyarul, szlovákul, horvátul, románul, bolgárul és portugálul**

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

A közép-európai funkcionális programozás (CEFP) nyári iskolája a második intenzív program felsőoktatási tanulók és tanárok számára a közép-európai funkcionális programozás (CEFP) nyári iskolája közösségének kiterjesztését célzó 2017-1-SK01-KA203-035402 számú “Focusing Education on Composability, Comprehensibility and Correctness of Working Software” ERASMUS+ projekt keretében, amely 2019. június 17. és 21. között került megrendezésre.

A mellékelt anyagot a fenti projekt keretében hozták létre és mutatták be. Ez a kiadvány a projekt szellemi kimenetének nyomtatott formátumú verziója.

TARTALOMJEGYZÉK



VIZUÁLIS PROTOTÍPUS KÉSZÍTÉSE TASZK-ORIENTÁLT PROGRAMOZÁSSAL

E kurzus során vizuális segédprogram segítségével készítünk Taszk-orientált programozási (TOP) alkalmazásokat. A TOP egy olyan új programozási paradigma, mely segítségével a fejlesztők gyorsan készíthetnek többfelhasználós web alapú prototípusokat. A TOP alkalmazásokat elsősorban „Task”-ok létrehozásával modellezzük. A Taszkok a valós világ olyan szeleteit reprezentálják, melyeket a felhasználó vagy egy rendszer hajt végre. Pár hasznos művelet segítségével ezeket nagyobb, erősebb Taszkokká tudjuk összeépíteni.

Pár alkalmazás vizsgálatával bemutatjuk a TOP alapvető koncepcióit, és megmutatjuk, hogyan modellezzük őket Taszkok segítségével egy grafikus fejlesztői környezetben. A grafikus környezet segíti a fejlesztőt a modellezés során. Az eszköz csak a Taszkok létrehozásának és kiterjesztésének helyes módjait jeleníti meg, és tanácsokat ad a típus és hatóköri hibák megoldására. Mindez korrekt és kompatibilis programkódot eredményez.

A hallgatókat a gyakorlati rész során arra biztatjuk, hogy fejlesszék tovább a példaalkalmazásokat. Grafikus megközelítésünk csak a legalapvetőbb programozási és adatszerkezeti ismereteket követeli meg. A TOP és modellezési elveibe való bevezetés az mTask kurzus előfeltétele.

DOLGOK INTERNETÉNEK (IOT) TASZK-ORIENTÁLT PROGRAMOZÁSA

A Dolgok Internete (IoT) olyan eszközökből áll, melyek más Internet eszközöket érzékelnek, velük tevékenykednek és kommunikálnak. Alapvető követelmény velük szemben, hogy olcsók legyenek és kevés energiát fogyasszanak. Mindezt úgy szokás elérni, hogy vezérlésüket kicsiny mikroprocesszor végzi, minimális memóriával és végrehajtási teljesítménnyel. A legtöbb ilyen eszköz még rendes operációs rendszerrel sem rendelkezik, hanem a feladatnak megfelelő célszoftvert futtat.

Mindez komoly kihívást jelent az IoT eszközök programozása során. Az ilyen eszközön futó egyetlen program kénytelen átlapolni minden részfeladatot, mint a bemenetek felügyeletét, a perifériák vezérlését és a (külső) kommunikációt. Az egymással együttműködő eltérő eszközök meg kell állapodjanak a használt protokollban és nehéz konkurens problémákat kell megoldaniuk.

Ebben az előadásban interaktív bevezetést tartunk a Dolgok Internete Taszk-orientált programozásába (TOP). A mi TOP megközelítésünkben az eszközök és szervereik közötti kommunikációt elrejti az mTask rendszer. A teljes rendszer egyetlen magasszintű funkcionális programként kerül leprogramozásra. A rendszer részfeladataihoz egy-egy kapcsolódó mTask-ot hozunk létre. Ezeket a részfeladatokat taszk-kombinátorokkal építjük össze erősebb taszkokká. Ezek a taszkok hozzáférnek más altaszkok közbülső értékeihez, és bármely más taszkkal tudnak kommunikálni megosztott adatforrásokon (SDS) keresztül. Az altaszkokat dinamikusan töltjük fel az eszközökre és azok helyben értelmeződnek. A futási idejű problémákat az erős típusrendszer előzi meg. Mindez a megközelítés nagyban leegyszerűsíti a Dolgok Internetének szoftverfejlesztését.

Irodalom

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.

FESSED A PROGRAMOD ZÖLDRE! – AZ ADATSZERKEZETEK ENERGIAHATÉKONY IMPLEMENTÁLÁSA

Az energia-hatékonyság évek óta fontos szempont mind a hardver világában, mind az alacsonyszintű szoftverek fejlesztői között [1], [2], [3]. Ugyanakkor a fentartható fejlődés irányába mutató világméretű mozgalmak növekvő szerepe [4] – beleértve a fentartható szoftvermérnöki tevékenységet, kombinálva az energia-hatékonyság rendszerszintű figyelembevételével, mint minőségi mutatóval – további motivációt adott a felhasználói programok végrehajtásának energiafogyasztására tett hatása vizsgálata szempontjából. Ezek a folyamatok arra késztették a fejlesztőket, hogy energia-felhasználási szempontok szerint újraértékeljék az alkalmazások elkésztéséhez használt létező módszereket, eszközöket és programozási nyelveket. A jelen mű olyan szempontokat vesz figyelembe, mint a kód-összezavarás [5], az Android API hívások [6], az objektumelvű kód-refaktorálása [7], a konkurens végrehajtás eszköztára [8], és az adattípusok hatása [9] az energiafogyasztásra. Ezen összetevők elemzése mind a szoftverfejlesztők, mint a karbantartást végzők számára fontos.

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

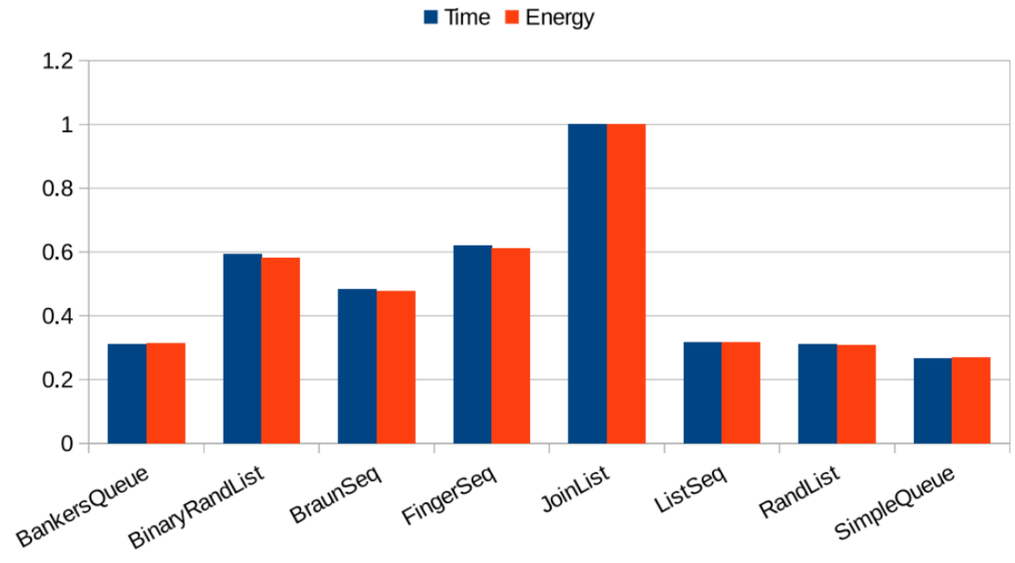
Ebben a tananyagban elemezzük és összehasonlítjuk az olyan konkrét adatabsztrakciók energiahatékonyságát, mint a Sorozatok, Halmazok, és Asszociatív adatszerkezetek. Mindegyik implementáció esetén megvizsgáljuk, hogy az olyan műveletek, mint a hozzáadás, törlés vagy keresés hogyan viselkednek különböző terhelések esetén. Vizsgálatunk tárgya egy-egy funkcionális [13,14] és objektumelvű programozási nyelv [13,14].

A funkcionális esetben az 1. ábrán látható adatszerkezeteken hajtottunk végre műveleteket a 2. ábrán látható műveletekkel paraméterekkel.

Az objektumelvű esetben a 3. ábrán látható adatszerkezeteken hajtottunk végre műveleteket a 4. ábrán látható műveletekkel és paraméterekkel.

A célunk az volt, hogy a fejlesztőken olyan közvetlenül felhasználható információkkal lássuk el, melyek közvetlenül alkalmazhatóak a zöld szoftver konstrukciókhoz és már elérhetőek a fejlesztő-támogató eszközökbe integrálva. Meg tudtuk mutatni, hogy ugyanazon művelete eltérő implementációi lényegileg különböznek mind a futási idő, mind az energiafelhasználás szempontjaiból. Például, az 5. ábra a Sorozat absztrakció törlés műveletét mutatja a Haskell Edison könyvtárából.

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...	

Irodalom

[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

ENERGIABARÁT SZOFTVER MÉRNÖKI KURZUSON

A fentartható fejlődés nem csak a politika világában, de a mérnökök között is egyre inkább központi fontosságú témává vált. Számos kutatás mutatja, hogy ez alól a szoftvermérnökök sem kivételek. A „zöld”, környezetbarát szoftverek intenzív kutatása ellenére a jelenlegi számítástechnikai felsőoktatás még nem célozza meg a környezettudatosságot. Szeretnénk bemutatni azt a környezetbarát oktatási modulunkat, melyet a haladó szoftverfejlesztési kurzusunk részeként vezettünk be. Bemutatjuk az energiahatékonyság szempontjából kérdéses minták és a „zöld” kódátalakítások gyűjteményét, melyek előzetes eredményeink szerint segítik a hallgatókat, hogy elemezzék és optimalizálják a szoftverrendszerek energiahatékonyságát.

Irodalom

[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

JAVA PROJEKTEK ENERGIAHASZNÁLATÁNAK ELEMZÉSE

Ez a szeminárium Java programozási nyelven megvalósított szoftveralkalmazások energiahatékonyságra fókuszál. Az első részben ismertetjük a tudomány jelenlegi állását az energiahasználat elemzése terén és hogy mi módon lehet a kód egyes részeinek energiafelhasználását kijelezni. Utána egy saját forráskód elemző módszert mutatunk be, mely a processzor, a memória és a tároló merevlemezzel kapcsolatos információkat jelzi ki. Ezek után röviden bemutatjuk az elkészült Java alkalmazást. Az alkalmazás gyakorlati hasznosságát demonstrálandó több tesztesetet hoztunk létre, melyekkel mérjük az energiafelhasználást. Minden példa esetében a feladatot legalább kétféleképen oldottuk meg, hogy meghatározhassuk az alacsonyabb energiaigényűt. A példák eredményei a szeminárium részét képezik.

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);
}
```

Irodalom

[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

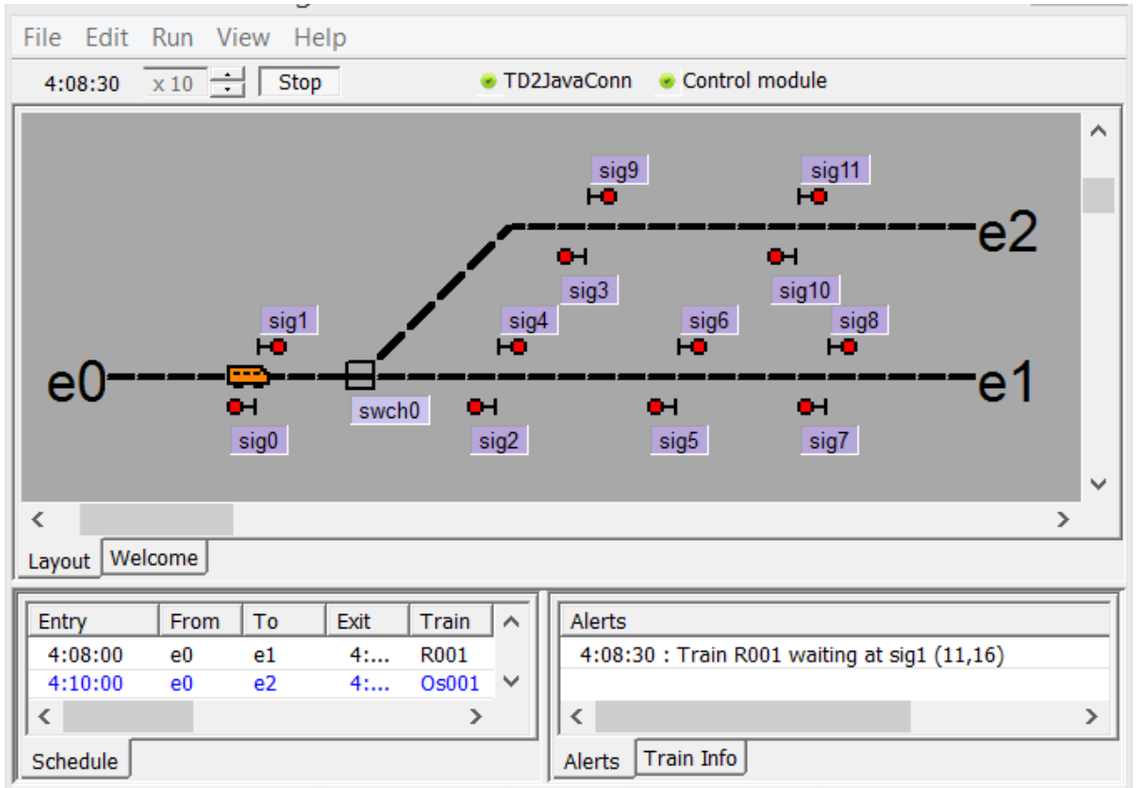
[5] Cs. Szabo, E.M.M. Alzeyani: Measuring Energy Efficiency of Selected Working Software, Studia Universitatis Babe¸s-Bolyai Informatica, Vol. 63, No. 1, 2018, pp. 5–16. https://doi.org/10.24193/subbi.2018.1.01

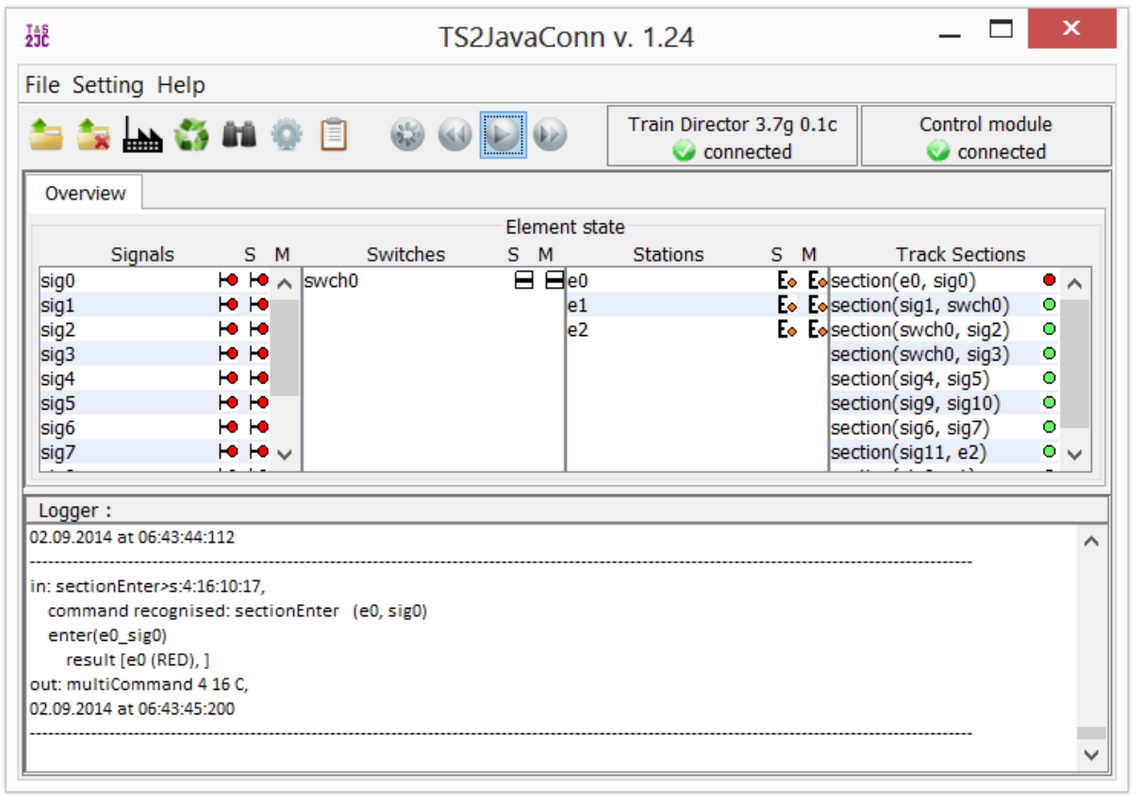
[6] Cs. Szabo, J. Saraiva: Focusing software engineering education on green application development, in Conference of Information Technology and Development of Education – ITRO 2017, Novi Sad, Serbia, pp. 165–169, ISBN 978-86-7672-302-7.

HELYES PROGRAMOK FEJLESZTÉSE A B-MÓDSZERREL

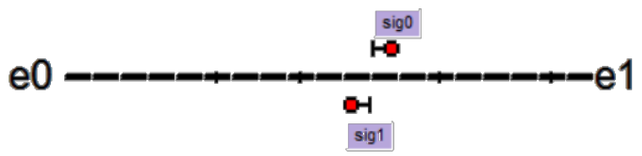
Bizonyítottan helyes programok fejlesztésének egyik elismert módja formális módszerek (FM) alkalmazása a specifikáció és verifikáció során. A formális módszerek szigorú matematikai alapokon nyugvó technikák szoftver- vagy hardver rendszerek specifikációja, elemzése, fejlesztése és ellenőrzése céljából. A szigorúság itt azt jelenti, hogy a formális módszerek egyértelmű szintaxissal és szemantikával rendelkező formális nyelvet definiálnak, míg a matematikai alapok azt, hogy ez a nyelv bizonyos matematikai eszközök (formális logika, halmazelmélet) segítségével megadott.

Az ipar által használt egyik formális módszer a B-Módszer [1,2,3,4], A B-Módszer állapot alapú, modell elvű formális módszer szoftverfejlesztés számára. A B-Módszert elsősorban a vasutaknál használják, automatizált városi földalatti vasúti rendszerek biztonságkritikus szoftvere számára (többek közt Budapesten is). A B-Módszertan ereje a jól definiált fejlesztési folyamatban rejlik, amelyik lehetővé teszi, hogy a szoftver rendszert B-gépeknek nevezett komponensek halmazaként adjuk meg, és az így megadott absztrakt specifikációt finomítsuk konkrét specifikációvá. A konkrét specifikáció már automatikusan fordítható ADA, C vagy valamely más programozási nyelvre. Az absztrakt specifikáció belső konzisztenciáját és a finomítás egyes lépéseit predikátumok halmazának bizonyításával igazolhatjuk, ezeket bizonyítási követelménynek (proof obligation, PObs) nevezzük. A teljes fejlesztési folyamat, beleértve a bizonyítást is ipari méretekben is használható szoftvereszközzel, az Atelier B-vel támogatott [5].

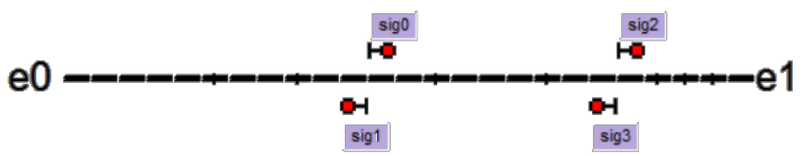




Ez a lecke a B-Módszer egy nem túl nehéz, gyakorlati jellegű bevezetőjeként szolgál. A lecke során a résztvevők egy egyszerű vasúti vezérlő szoftvert fognak kifejleszteni. A hallgatók elsajátítják az esettanulmány futtatását a TrainDirector/ TS2JavaConn (TD/TS2JC) eszközzel [6,7]. Az eszközrendszer a TrainDirector [8] szimulációs játék (1.ábra) módosított változata és a TS2JavaConn alkalmazás (2. ábra) kombinációja, mely lehetővé teszi a különállóan fejlesztett vezérlő használatát a szimulációs játékban. Abból a célból, hogy egy prototípus eszközt hozzunk létre vezérlők számára, az eszközöket kiterjesztjük az Open Rails [10] 3D szimulátorra is.



A B-módszer bevezető után a résztvevők egy egyvágányú, két szekcióból álló vasúti rendszer vezérlő rendszerét fogják kifejleszteni (3. ábra). A vezérlő (1. kódlista) B-módszerben van megírva. A kurzus során a hallgatók kifejlesztenek egy 3 szekcióból álló vezérlőt és igazolják a helyességét (4. ábra).



1. kódlista A vasúti esettanulmány vezérlőjének kódja két szekció esetén B-módszerben

```
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
```

Irodalom

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

ELOSZTOTT ÉS VIRTUALIZÁLT HÁLÓZATI ERŐFORRÁSOK FELÜGYELŐ ÉS SZERVEZŐ ELJÁRÁSAINAK HALADÓ PROGRAMOZÁSA – VÁLOGATOTT ESETTANULMÁNYOK

Szabványosított új felügyelő és szervező (management and orchestration, MANO) eljárások álnak rendelkezésünkre elosztott és virtualizált hálózati környezetekben. Szerepük elsősorban a hálózati funkcionalitások megbízható és biztonságos vezérlése. Korábbi előadásunk folytatásaként, melyben bemutatjuk az alapvető elveket, most pár válogatott esettanulmányt mutatunk be ahol ezeket a funkcionalitásokat implementálták és kifejtjük a mögöttes haladó módszereket, és rámutatunk a megoldás egyszerűségére.

Irodalom

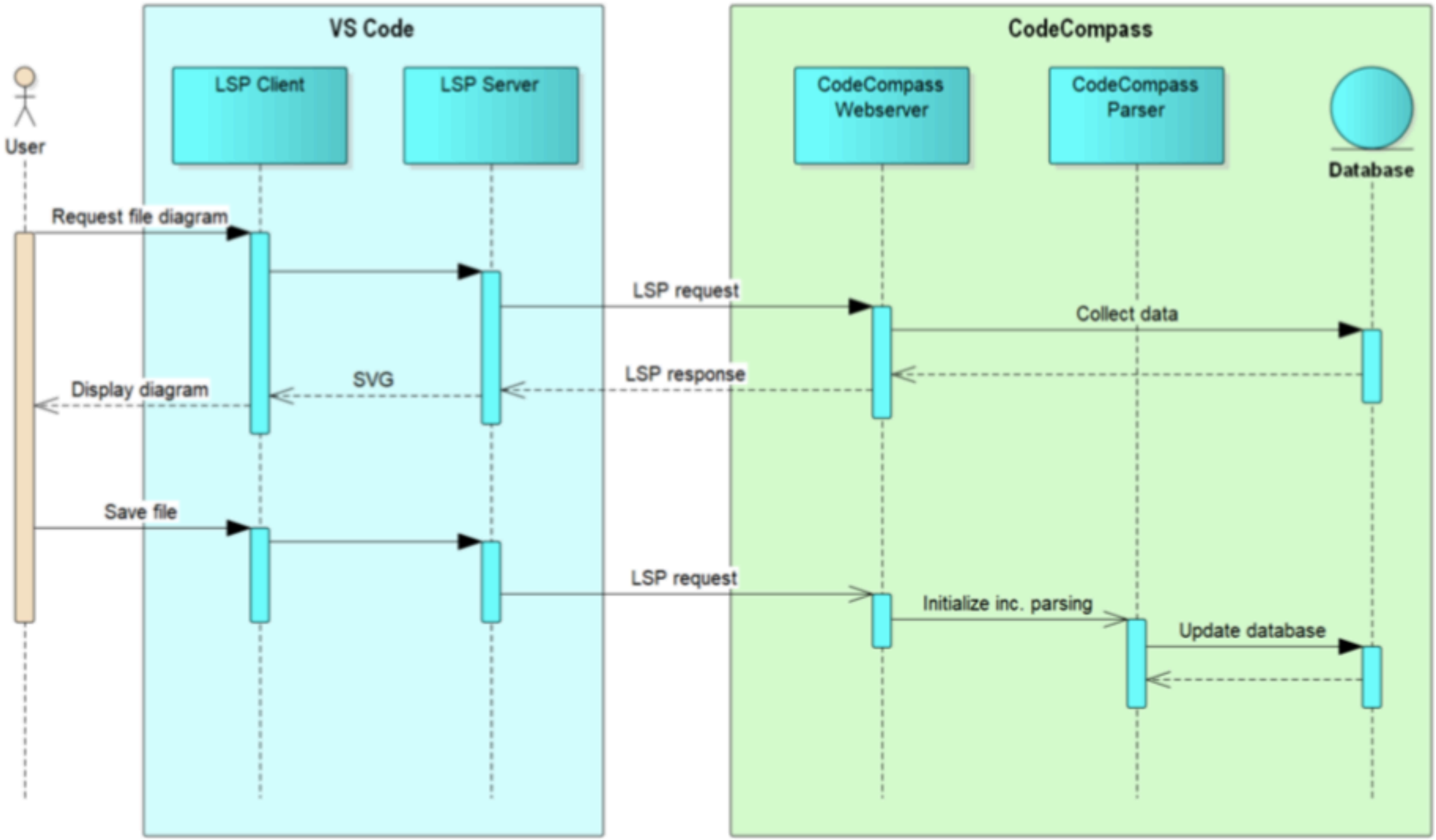
[1] ETSI Industry Specification Group (ISG) NFV: ETSI GS NFV- MAN 001 v1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration European Telecommunications Standards Institute (ETSI), 2014, https://www.etsi.org/deliver/etsi_gs/NFV- MAN/001_099/001/01.01.01 60/gs_NFV-MAN001v010101p.pdf,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

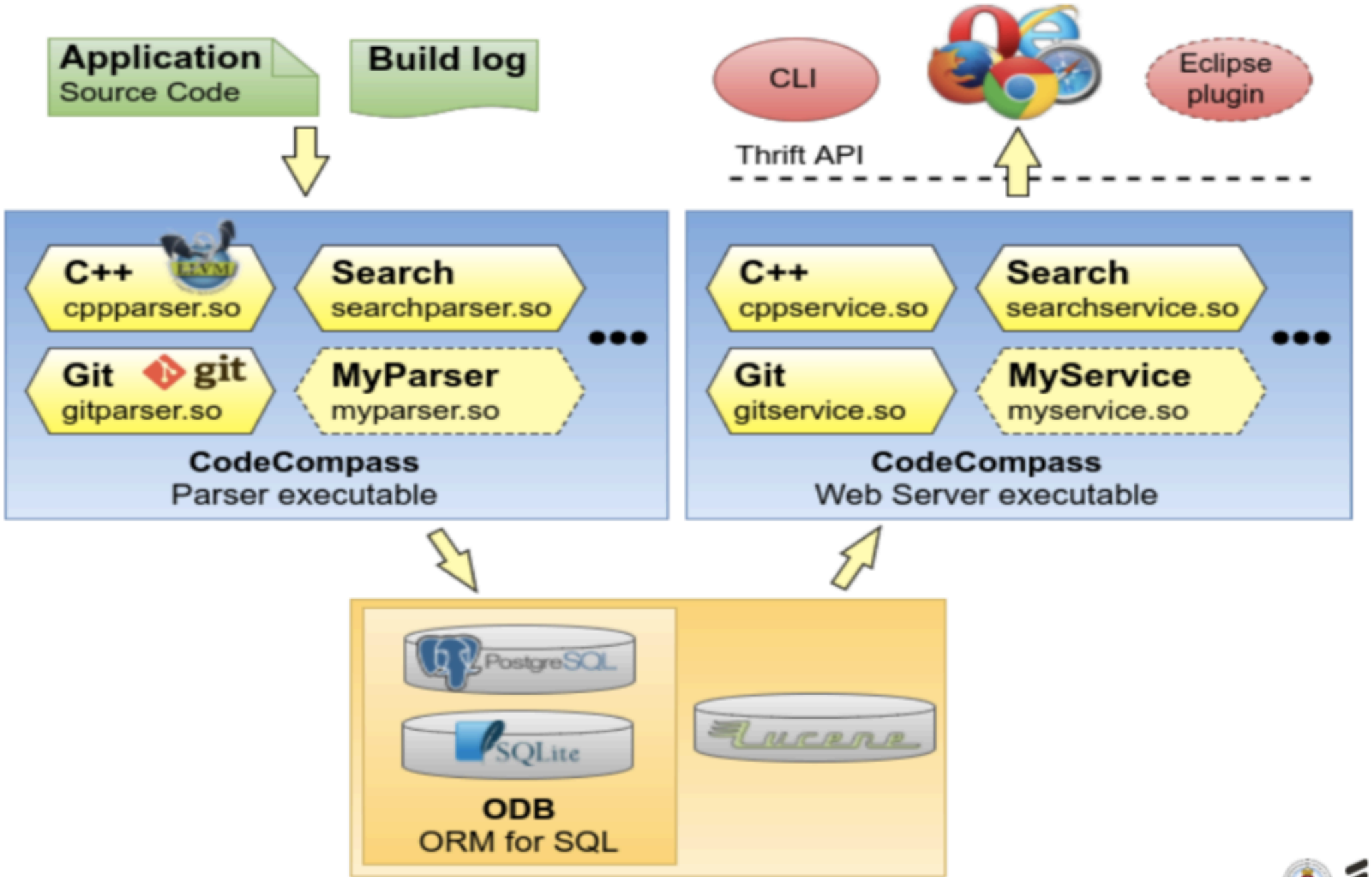
[2] Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine 53(2), 90–97 (2015)

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

KÓDMEGÉRTÉS TÁMOGATÁSA HALADÓ ESZKÖZÖKKEL

E lecke során bemutatjuk a hallgatók számára az élenjáró kódmegértést támogató eszközöket. Először elméleti áttekintést adunk a kódmegértésről, navigációról és kódvizualizációról, valamint alkalmazásukról a gyakorlati szoftverfejlesztésben. A gyakorlati részen bemutatjuk a CodeCompass munkafolyamatát, inkrementális elemzéssel és a Visual Studio Code editorral, mint front-end-el, és a Language Server Protocol-on keresztüli kommunikációt. Elemezni fogunk egy nyílt forráskódú szoftverrendszert és az eszközrendszer segítségével megkeresünk benne egy hibát és azt kijavítjuk.


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"
            }
        }
    }
}
```

Irodalom

[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.

FUNKCIONÁLIS TÖMB-PROGRAMOZÁS A SAC NYELVVEL: LEHETŐSÉGEK ÉS KIHÍVÁSOK

A SAC (Single Assignment C, egyértékadásos C) számos szempontból különleges funkcionális programozási nyelv. Ahogy a neve is sugallja, a SAC a C nyelvi szintaxist (sok kapcsos zárójellel) ötvözi az állapotmentes, tisztán funkcionális szemantikával. Az eredeti szándéka szerint az imperatív hátterű programozók számára könnyen elfogadható nyelv meglepő megoldásokat kínál a tipikus funkcionális- vagy imperatív nyelvi konstrukciók részére. A funkcionális oldalon nem megszokott módon listák vagy fák helyett a SAC a többdimenziós tömbökre helyezi a hangsúlyt. A tömbök programozása a többdimenziós tömböket holisztikus módon kezeli: a függvények a potenciálisan nagyméretű argumentum tömböket érték szerinti szemantikával képezi le az eredmény tömbökre és új tömbműveleteket a meglevők kompozíciójából hozhatunk létre. A SAC nyelv nagy fejlesztési hatékonyságú azokon az alkalmazási területeken, melyek nagyméretű adatgyűjteményekkel dolgoznak számításigényes módon.

Ugyanakkor a fordítási módszerek segítségével a SAC nagy futási hatékonyságú nyelv is egyben, versenyezve az alacsonyszintű imperatív nyelvekel. A tömbök absztrakt nézete a funkcionális szemantikával ötvözve nagyhatékonyságú programátalakításokat támogat. A magasszinten optimalizált futtató rendszer automatikus memóriakezelése a memória közvetlen újra-hasznosítására törekszik. Végül, de nem utolsósorban a SAC fordító, kihasználva a nyelv állapotmentes szemantikáját és adatpárhuzamos természetét, lehetővé teszi a teljesen fordító-vezérelt gyorsításokat a legtöbb mai hardver architektúrán a többmagos rendszerektől a GPGPU gyorsítókon át a munkaállomások klasztereiig.

Az előadás a SAC nyelv tervezési szándékaival indul és gyakorlati bevezetést ad a tömb programozásba, mint paradigmába. Megvizsgáljuk a háttérben húzódó tömb-számítások szempontjainak összes hatását a konkrét nyelv imperatívnak kinéző funkcionális kódjára, megbeszélünk számos példát, felderítjük a fordítás kihívásait és különböző párhuzamos architektúrán mérjük a hatékonyságot.

KIEGYENSÚLYOZOTT ELOSZTOTT SZÁMÍTÁSI MINTÁK

A konkurens szoftverek jelenlegi élenjáró fejlesztési módszerei változatos módszertanokat és megközelítéseket alkalmaznak a magas sebesség elérésére. Mindemellett a párhuzamosság az egyik legnehezebb terület maradt, különösen a minta alapú programozási megközelítés esetében. A fő cél a párhuzamos programozási sémák felfedezése egy új környezetben, hogy illusztráljuk alkalmasságukat és használhatóságukat új elosztott számítási felállások esetében. A párhuzamosság mértékét számos összetevő alapján vizsgáljuk, mint az alkalmazott számítási minta finomsága, az elosztott számítási csomópontok szemantikája, az adatáramlás és különösen a terhelés elosztás.

Irodalom

[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.