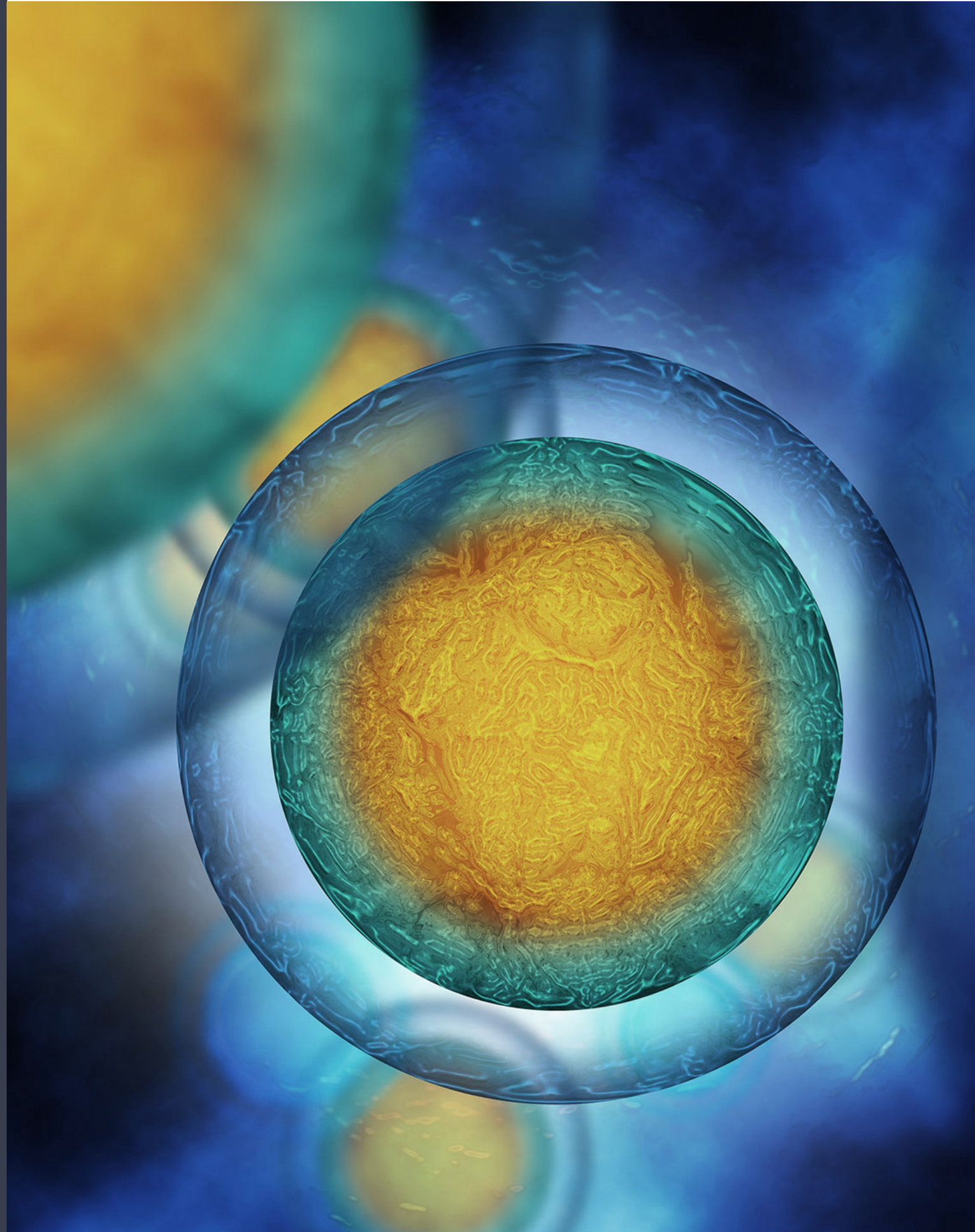
FE3CWS

УЧЕБНИ МАТЕРИАЛИ ЗА ЛЯТНО УЧИЛИЩЕ CEFP 2019

Интелектуален резултат 4 на проекта ERASMUS + 2017-1-SK01- KA203-035402



Някои думи по

съдържанието

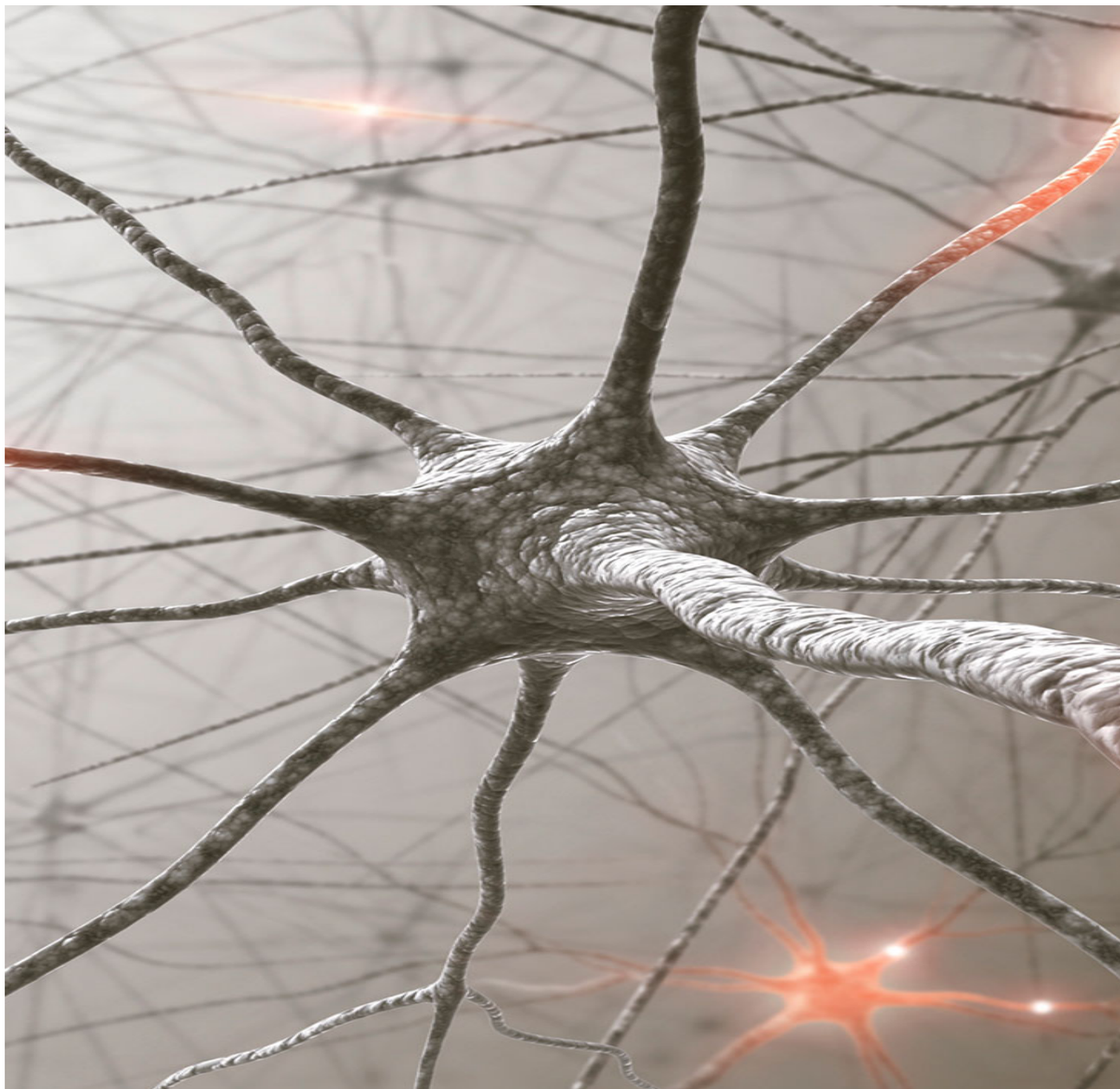
- **10 теми, свързани със състава на софтуера, разбирането и коректността**
- **20 автори от 7 европейски университета от Хърватия, Унгария, Холандия, Португалия и Словакия**
- **Предлага се на 7 езика: английски, унгарски, словашки, хърватски, румънски, български и португалски**

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

Лятното училище за централноевропейско функционално програмиране (CEFP) е втората интензивна програма за студенти и преподаватели, разширяваща общността на лятното училище за функционално програмиране (CEFP) в рамките на проект ERASMUS + № 2017-1-SK01 -KA203-035402 „Фокусиране на образованието върху комбинируемостта, разбираемостта и коректността на работния софтуер“, което се проведе между 17 и 21 юни 2019 г.

Включеният материал е създаден и представен в рамките на гореспоменатия проект. Тази публикация е печатната версия на интелектуалния продукт O4 на проекта.

Съдържание



Визуално въвеждане на прототипи чрез използване на програмно ориентирано програмиране

В този курс създаваме приложения с помощта на визуален асистент за програмиране ориентирано към задачи (Task Oriented Programming (TOP)). TOP е иновативна система за програмиране, която софтуерните инженери могат да използват за бързо изграждане на прототипни уеб приложения предназначено за множество потребители. Основният начин за моделиране на приложения в TOP е чрез създаване на задачи. Те представляват части от реални работни задачи, които могат да бъдат изпълнявани от хора или системи. С помощта на само няколко операции, те могат да бъдат събрани в по-големи и по-мощни задачи.

Ще разгледаме основните концепции на TOP, изучавайки някои примерни приложения, като едновременно с това показваме как можем да ги моделираме с помощта на задачи в среда за визуално развитие. Визуализираната среда насочва софтуерните инженери по време на моделирането. Инструментът представя разумен начини за създаване и разширяване на задачи като дава насоки как да се решат грешки свързани с типа и обхвата. Резултатът е правилен и лесно построим програмен код.

С помощта на практическа работна сесия, студентите се насърчават да разширят обхвата на познанията си относно примерните приложения. Нашият визуален подход изисква само основни познания за програмиране и типове данни. Въвеждането на TOP и съответните му принципи за моделиране са задължително изискване за курса по mTasks.

Програмно ориентирано програмиране за Интернет на нещата

Интернет на нещата (Internet of Things (IoT)) е изграден от устройства, които могат да усещат, действат и комуникират с други системи в интернет пространството. Типичните изисквания за IoT, е че устройсвата трябва да са евтини и да консумират малко енергия. Това се постига чрез малки микропроцесори, с ограничени количества памет и процесорна мощ. Те са тези, които управляват IoT. Повечето такива системи нямат подходяща операционна система и единствено стартират конкретна програма при изпълнение на планираната задача.

Това прави програмирането на IoT много по-голямо предизвикателство. Програмата, която се изпълнява на такова устройство трябва да остави на заден план всички подзадачи, като например, мониторинг входове, контролиране на периферни устройства и комуникация. Различните устройства, които работят заедно трябва да се синхронизрат относно използвания протокол, за да се решат най-наложащите съществуващи програмни проблеми.

В тази лекция ще изнесем практическо обучение по програмиране ориентирано към задачи (Task Oriented Programming (TOP)) за Интернет на нещата. При този подход комуникацията между устройствата и техните сървъри се обработва прозрачно от системата mTask. Цялата система е програмирана на високо ниво в една функционална програма. За всяка подзадача на системата дефинираме съответстваща mTask. Тези подзадачи могат да бъдат съставени от комбинатори на по-мощни задачи. Тези задачи могат да изследват стойности, свързани с посредник на други подзадачи, както и да комуникират с всяка друга задача в системата чрез споделени източници на данни (Shared Data Sources (SDS)).

Подзадачите за Инренет на нещата се изпращат директно на устройството и се изпълняват там. Мощната система, отговаряща за типа устройство, предотвратява проблеми с времето за изпълнение. TOP подходът значително опростява развитието на софтуерни продукти за Интернет на нещата.

Източници

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.

Оцветете програмите си в зелено – енергийната ефективност при приложението на структури от данни

Енергийната ефективност е грижа както за хардуерните, така и за софтуерните инженери на ниско ниво [1], [2], [3]. Нарастващото движение в световен мащаб към устойчивост, включително устойчивост на софтуера [4], съчетано със системния характер на енергийната ефективност като качествен атрибут, мотивира изследването на енергийното въздействие на приложен софтуер в изпълнение. Тази тенденция е накарала изследователите да оценят съществуващите техники, инструменти и езици за разработване на приложения от енергоцентрична гледна точка. Неотдавнашна работа е проучила ефекта, който фактори като обфуксация на кода [5], Android API обаждания [6], обектно-ориентирани кодови рефакторинга [7], конструкции за едновременно изпълнение [8] и типове данни [9] имат върху енергийната ефективност , Анализът на въздействието на различни фактори върху енергията е важен за разработчиците и поддържащите софтуер.

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

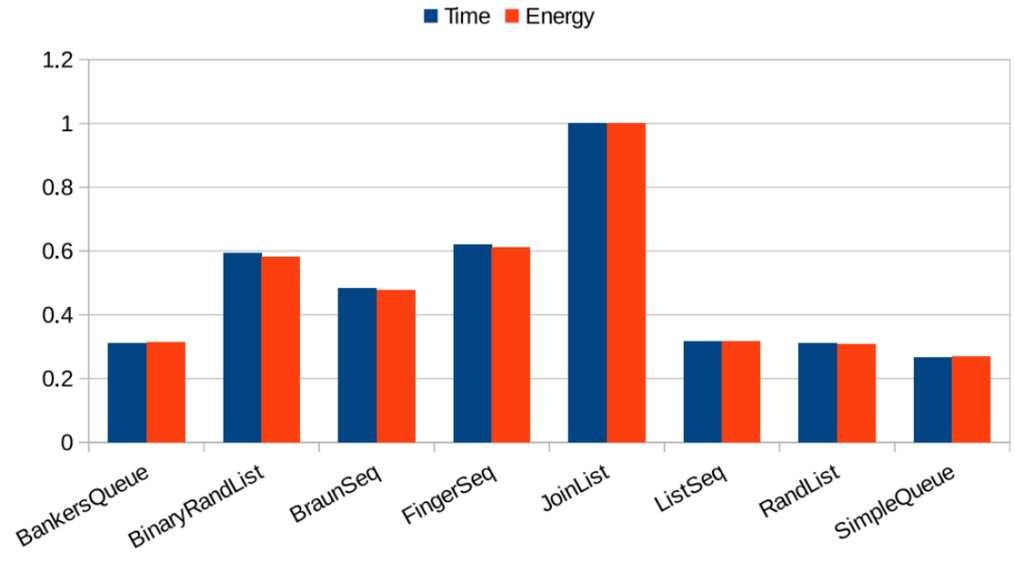
iters	operation	base	aux
1	add	100000	100000
1000	addAll	100000	1000
1	clear	100000	n.a.
1000	contains	100000	1
5000	containsAll	100000	1000
1	iterator	100000	n.a.
10000	remove	100000	1
10	removeAll	100000	1000
5000	toArray	100000	n.a.
10	retainAll	100000	1000

В този урок анализираме и сравняваме енергийната ефективност на различни реализации за конкретни абстракции на данни като последователности, набори или асоциативни колекции. За всяка реализация проверяваме как операции като добавяне, изтриване или търсене на елементи се справят с различни натоварвания. Обектите на нашето изследване са функционален език за програмиране [10,11] и обектно-ориентиран език [13,14].

Във функционална настройка сравнихме реализациите, представени на Фигура 1, а именно използвайки операциите, представени на Фигура 2. В обектно-ориентираната сфера анализирахме реализациите, представени на фигура 3, а именно използвайки операциите, представени на фигура 4.

Нашата цел е да предоставим на разработчиците полезна информация, която вече е интегрирана в поддържащи инструменти и която може да управлява изграждането на зелен софтуер. Успяхме да покажем, че една и съща операция, предоставена в различни реализации, може да се различава значително както по време на изпълнение, така и по отношение на консумацията на енергия. Като пример, на фигура 5 изобразяваме резултатите от операцията за премахване за реализациите на абстракцията на последователности, налични в библиотеката на Edison на 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...	

Източници

[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

Зелен софтуер в курс по инженерство

Темата за устойчиво развитие се превръща във все по-важна тема не само в Световната политика, но и заема все по-централна позиция в инженерните професии по целия свят. Както е посочено в множество последни научни изследвания, в това число, софтуерните инженери не правят изключение. Въпреки интензивното изследване на зеления софтуер, днешното университетско образование често не успява да се справи с екологичната ни отговорност. Представяме модул на зеления софтуер, който сме въвели като част от напреднал курс по софтуерно инженерство. Представяме каталог с енергийни миризми и зелени преустройства, които според наши предварителни резултати показват, че помагат на студентите да оптимизират консумацията на енергия на софтуерните системи.

Източници

[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

Профилиране на енергийни приложения за софтуер за JAVA проекти

Този урок засяга енергийната ефективност на софтуерни приложения, реализирани на езика за програмиране - Java. Първата част описва сегашното състояние в областта на енергийното профилиране, както и опциите за показване на консумацията на енергия на сегменти от програмния код. Представя се метод за анализ на персонализиран код за показване на информация, адресиращ процесора, оперативната памет и твърдия диск. След този анализ следва кратко въведение към реализираното приложение на Java. Създават се няколко тестови решения, при които се измерва консумацията на енергия, за да се демонстрира практическото приложение. С всеки пример поставяме акцент върху решаването на един проблем с поне две решения, за да определим коя реализация има по-ниска енергоемкост. Резултатите от примерите са също включени в този урок.

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);
}
```

Източници

[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

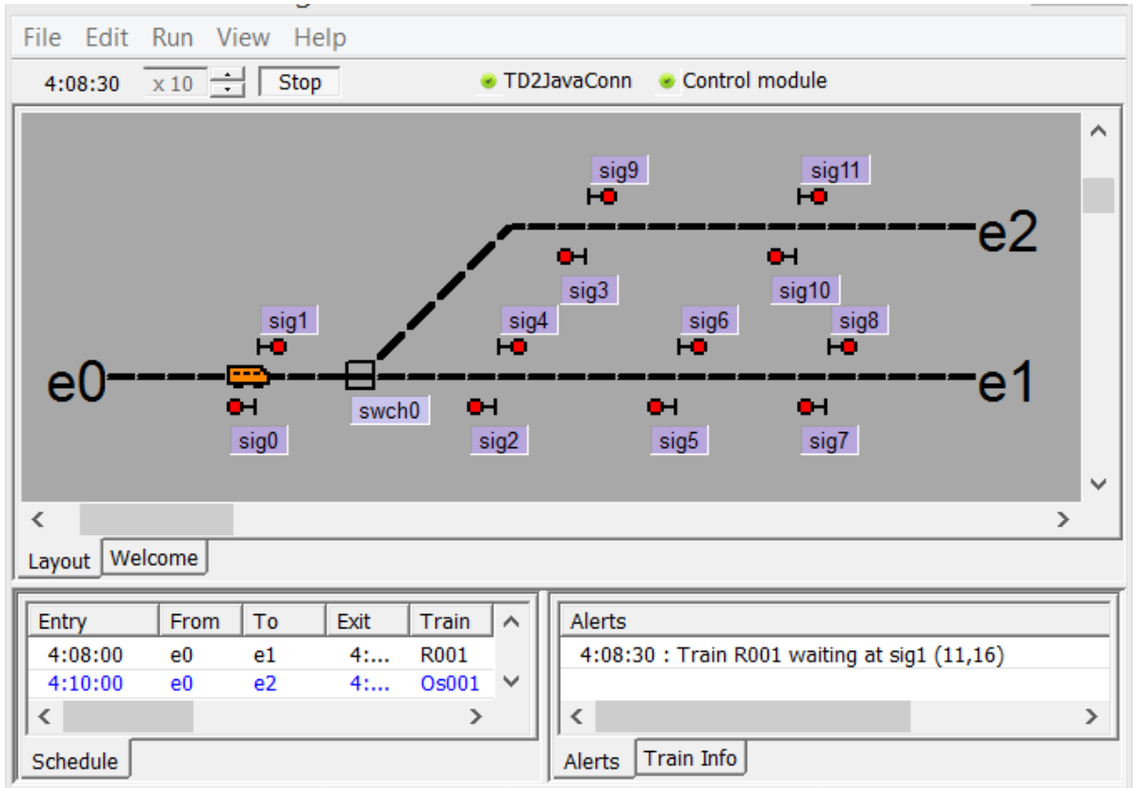
[6] Cs. Szabo, J. Saraiva: Focusing software engineering education on green application development, in Conference of Information Technology and Development of Education – ITRO 2017, Novi Sad, Serbia, pp. 165–169, ISBN 978-86-7672-302-7.

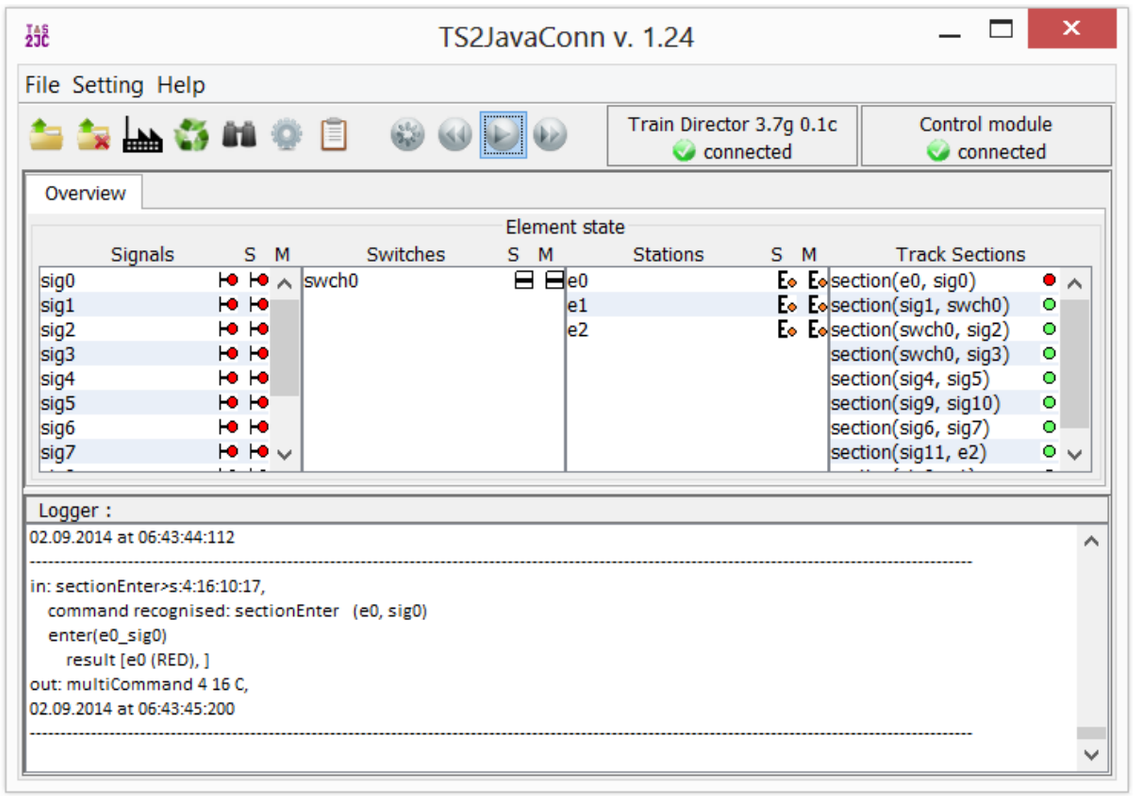
Разработване на правилен софтуер с B-метод

Един от добре познатите подходи към развитието на правилни софтуерни системи с цел конкретизиране и проверка на софтуер е използването на формални методи (ФМ). ФМ са строги математически базирани техники за спецификация, анализ, разработка и проверка на софтуер и хардуер. Строги означава, че формалният метод осигурява формален език с недвусмислено дефиниран синтаксис и семантика, а математически базирани означава, че за определяне на езика се използва математически апарат (формална логика, теория на множествата и т.н.).

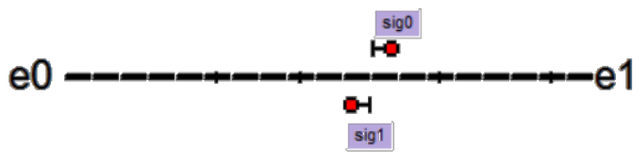
Един от ФМ, използван в индустриалната практика, се нарича B- Method [1,2,3,4]. Той е държавен, моделно ориентиран формален метод, който е предназначен за разработване на софтуер. B- Method се използва предимно в железопътния сектор, за критичния за безопасността софтуер зад автоматизираните системи на градското метро (включително това в Будапеща). Силата на B-Method се състои в конкретно дефинирания процес на разработка, който позволява да се определи софтуерна система като съвкупност от компоненти, наречени B-машини и да се сведе такава абстрактна спецификация до конкретна. Конкретната спецификация може да бъде автоматично преведена на ADA, C или друг език за програмиране. Вътрешната съгласуваност на абстрактната спецификация и коректността на всеки етап на уточняване се проверяват чрез доказване на набор от твърдения, наречени доказателствени задължения (Proof Obligations (PObs)). Целият процес на разработка, включително доказателствената част, се поддържа от софтуерно устройство за индустриална сила, наречено Atelier B[5].

Този урок служи като леко, практично, въведение в B-метод. По време на урока участниците ще разработят прост софтуерен контролер за железопътен сценарий.

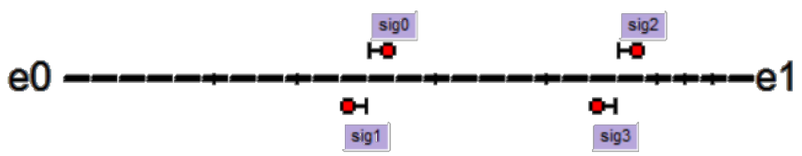




Те ще могат да изпълняват сценария с контролера в инструментариума Train Director / TS2JavaConn (TD / TS2JC) [6,7]. Наборът от инструменти се състои от модифицирана версия на симулационната игра Train Director [8] (фиг. 1) и приложение, наречено TS2JavaConn (фиг. 2), което позволява използването на отделно разработени софтуерни контролери с симулационната игра. С амбицията да се използва наборът от инструменти за прототипиране на контролера, по-късно [9] беше разширен чрез персонализирана версия на Open Rails [10] 3D симулатор на влак.



След запознаване с B-метода, на участниците в курса се дава контролер за сценарий за еднопътна железопътна линия с два участъка (фиг.3). Контролерът (Lising 1) е написан на езика на B-Method. По време на курса те разработват и проверяват контролер за едноколесен железопътен сценарий с три участъка (фиг. 4).



Листинг 1. Контролерът за железопътния сценарий с две секции, написани на езика на B-метод.

```
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
```

Източници

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

Програмиране на усъвършенствано управление и организиране на виртуализирани мрежови ресурси - избор на казуси

Новите функции за управление и оркестрация (MANO) са стандартизирани за използване в разпределени и във виртуализирани мрежови среди. Основната им роля е да осигурят безопасна и надеждна работа на приложения, използващи мрежови функции. Като продължение на предишната лекция, където предоставихме основните понятия, в тази лекция представяме селекция от казуси, при които тези функции се прилагат като се обяснява семплостта и усъвършенстваните механизми зад тяхното приложение.

Източници

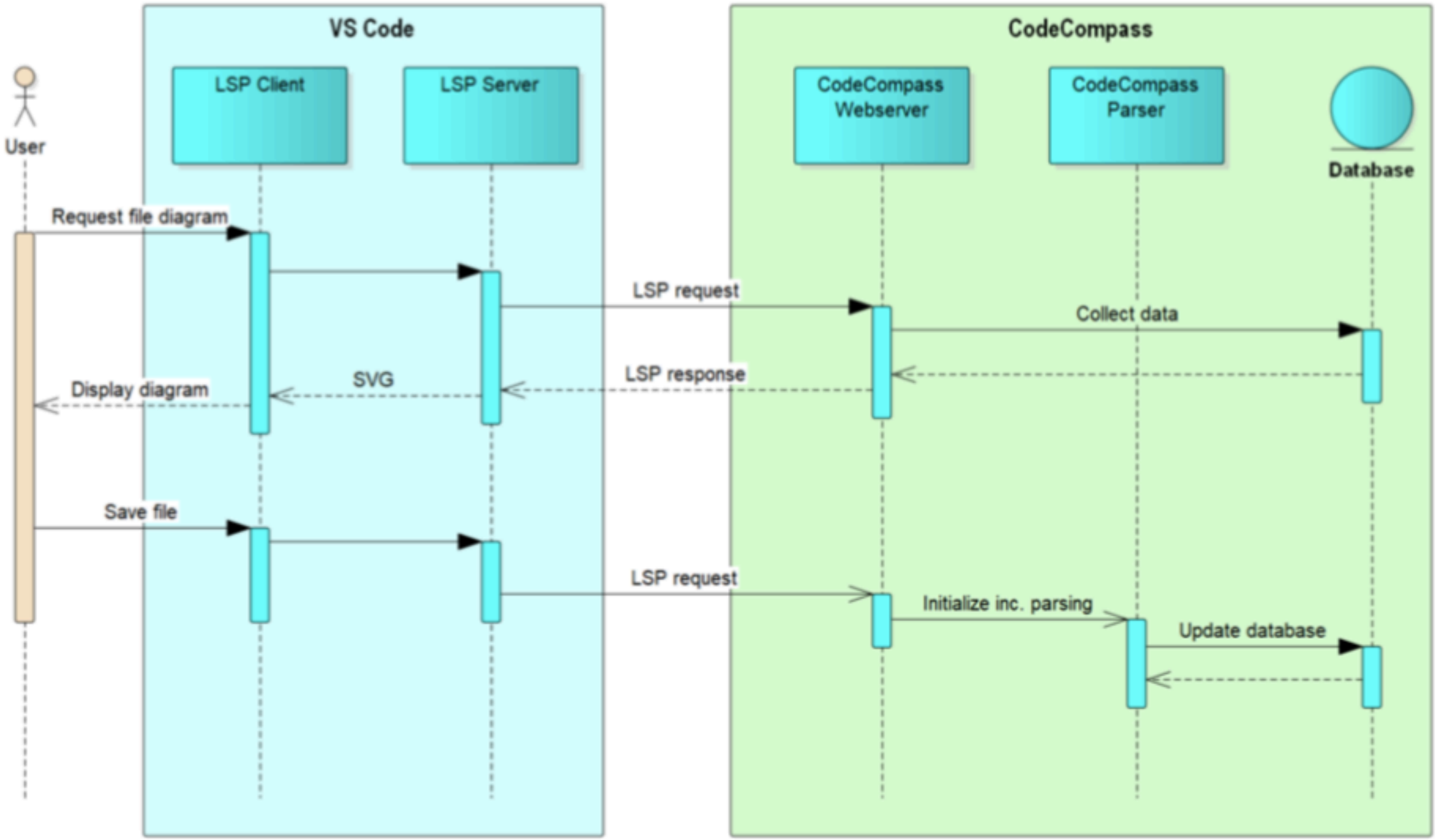
[1] ETSI Industry Specification Group (ISG) NFV: ETSI GS NFV- MAN 001 v1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration European Telecommunications Standards Institute (ETSI), 2014, https://www.etsi.org/deliver/etsi_gs/NFV- MAN/001_099/001/01.01.01 60/gs_NFV-MAN001v010101p.pdf,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

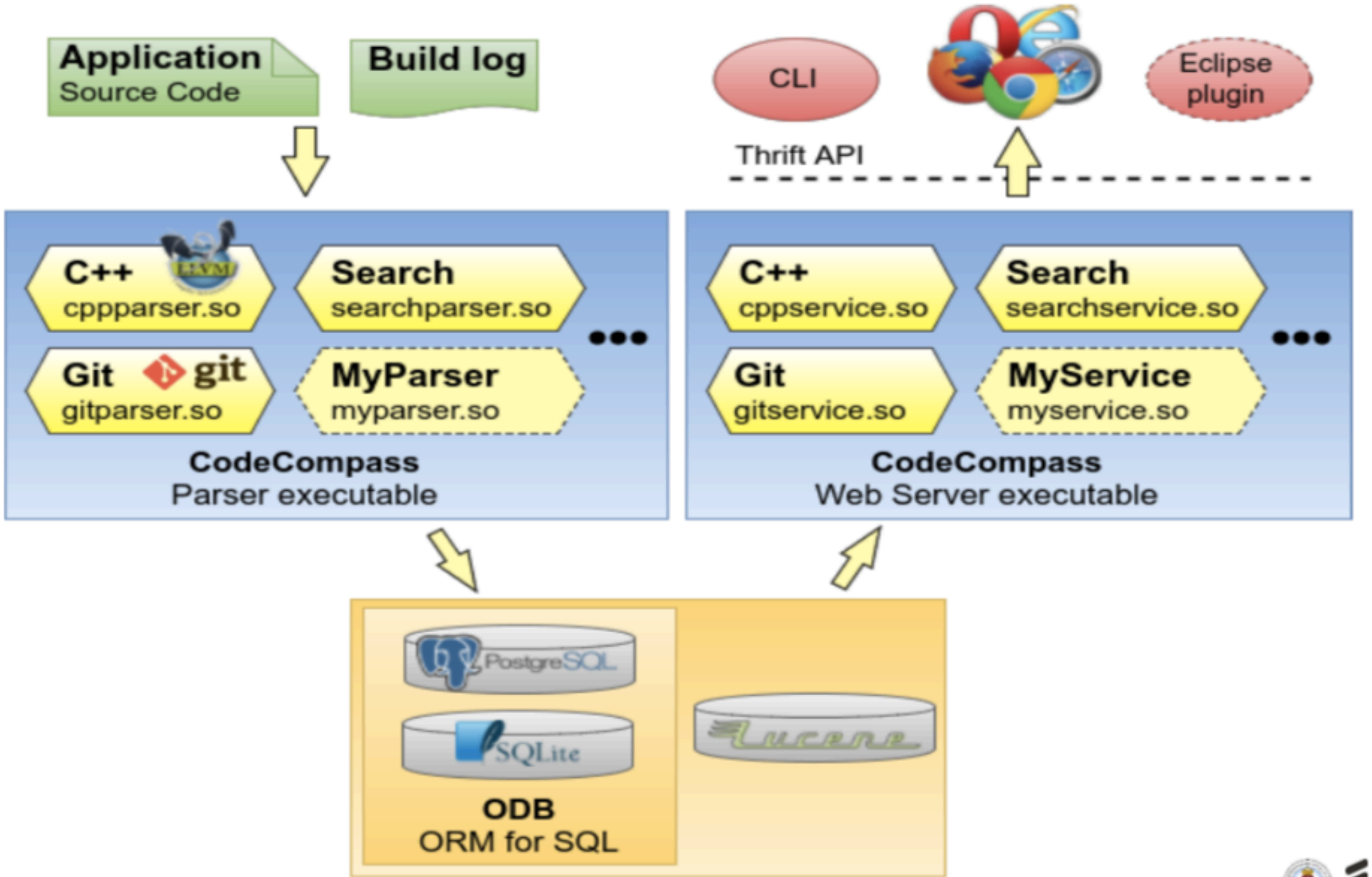
[2] Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine 53(2), 90–97 (2015)

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

Разбиране на код с разширена поддръжка на инструменти

В този урок ще запознаем студентите със състоянието на инструментите за разбиране на свръхмодерен код. Ще предоставим теоретична основа за разбиране на кода, навигация и методи за визуализация на кодове, както и подходи за тяхното прилагане в практическата разработка на софтуер. В практическата сесия демонстрираме как се настройва конкретен набор от инструменти: CodeCompass с инкрементален анализ, Visual Studio Code като фронт-енд инструмент и използване на протокола за езиков сървър. След това анализираме библиотека с отворен код и намериме и поправяме конкретна грешка в нея с помощта на инструментариума.


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"
            }
        }
    }
}
```

Източници

[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.

Програмиране на функционални масиви с еднозначно значение C: възможности и предизвикателства

SAC (Single Assignment C) е функционален език за програмиране, който в няколко аспекта се отличава от останалите. Както подсказва името, SAC комбинира C-подобен синтаксис (с много къдрави скоби) и свободна, чисто функционална семантика. Езикът е първоначално предназначен да улесни разбирането от програмисти с внушителен опит, като широкият избор предлага изненадваща перспектива за това какво представлява "типична" функционална или "типична" императивна конструкция на езика за програмиране. Друга необичайна страна на функционалния език, SAC, е че набляга на многомерни масиви, вместо списъци и дървета. Програмирането на масив третира многоизмерните масиви по холистичен начин: функциите картографират потенциално огромни масиви от аргументи, за да се получат масиви със семантика на извикване по стойност, а новите операции с масив се определят от състава на съществуващите. SAC е език с висока производителност за приложенията, които интензивно обработват и изчисляват големи количеството данни.

В същото време SAC е и високоефективен език, конкуриращ се с императивни езици на ниско ниво чрез компилационна технология. Абстрактният поглед върху масивите, комбиниран с функционалната семантика, поддържат широкообхватните програмни трансформации.

Високо оптимизираната система за изпълнение се грижи за автоматичното управление на паметта с акцент върху незабавното повторно използване на паметта. Не на последно място, компилаторът на SAC използва безсемантичната семантика на SAC и паралелния на данните характер на SAC програмите, с цел напълно насочено от компилатора ускорение на голямо разнообразие от съвременни машинни архитектури, от многоядрени сървъри до GPGPU ускорители и съвкупности (clusters) от работни станции.

Лекциите мотивират езиковия дизайн на SAC и предоставят практическо въведение в парадигмата за програмирането на масиви. Разглеждаме всички аспекти от изчисляването на основния масив до конкретния дизайн на езика с внушително изглеждащ функционален код, обсъждаме множество примери, проучваме предизвикателствата при компилацията и евентуално виждаме някои резултати от производителността на различни паралелни изчислителни архитектури.

Балансирани модели на разпределени изчисления

Най-съвременната разработка на софтуер използва в дълбочина подходи и методи за постигане на висока скорост. Въпреки това, паралелизмът остава една от най-трудните области, особено в случай на моделирани подходи за програмиране. Основната цел е да се проучат паралелните изчислителни схеми в нова среда и да се илюстрира целесъобразността и приложимостта в новите разпределени настройки за изчисления. Размерът на паралелизма се изследва въз основа на много фактори като: приложен изчислителен модел с особена прецизност на дизайна, семантика на разпределени пресечни точки, поточно предаване на данни и по-специално балансиране на натоварването.

Източници

[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.