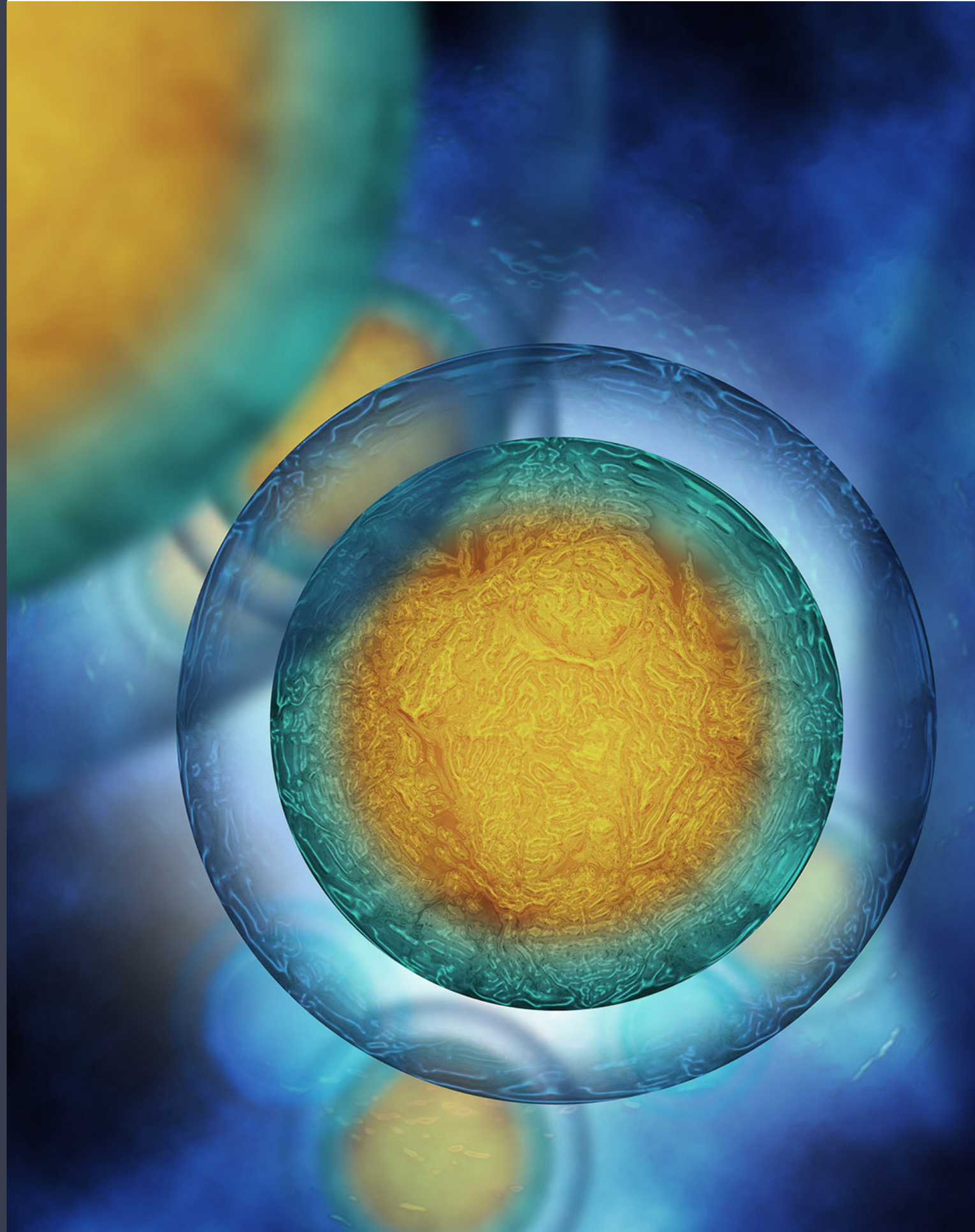
FE3CWS

MATERIAL DE ENSINO DA ESCOLA DE VERÃO CEFP 2019

Produção intelectual 4 do projeto ERASMUS + 2017-1-SK01-KA203-035402



Algumas palavras sobre o

CONTEÚDO

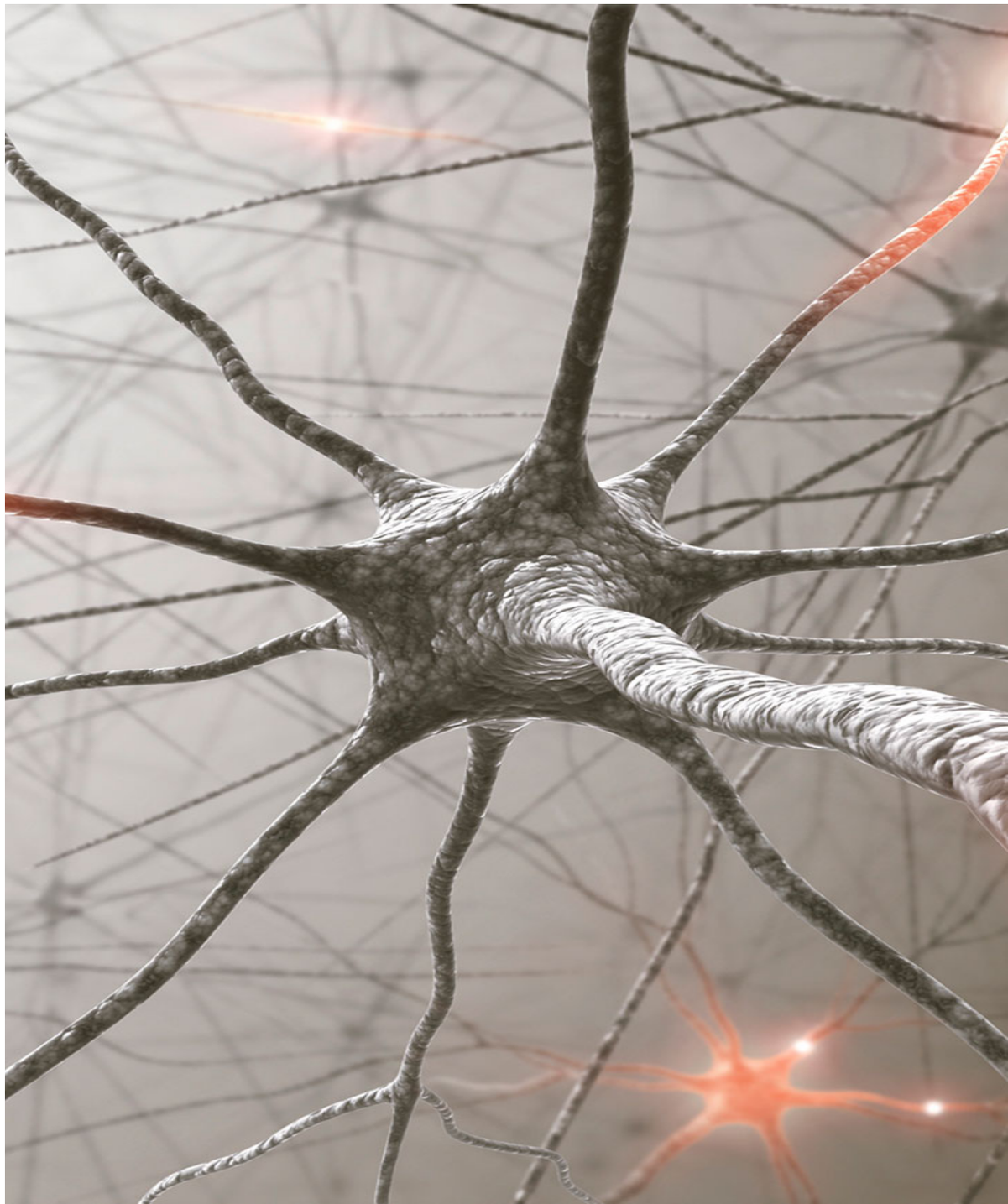
- 10 tópicos relacionados à composição, compreensão e correção de software
- 20 autores de 7 universidades europeias da Croácia, Hungria, Holanda, Portugal e Eslováquia
- Disponível em 7 idiomas: inglês, húngaro, eslovaco, croata, romeno, búlgaro e português

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

A Escola de Verão da Programação Funcional da Europa Central (CEFP) é o segundo programa intensivo para alunos do ensino superior e professores que estendem a comunidade da escola de verão da Programação Funcional da Europa Central (CEFP) no âmbito do projeto ERASMUS + 2017-1-SK01 -KA203-035402 "Focusing Education on Composability, Comprehensibility and Correctness of Working Software", realizada entre 17 e 21 de junho de 2019.

O material incluído foi criado e apresentado no quadro do projeto mencionado acima. Esta publicação é a versão impressa da saída intelectual O4 do projeto.

ÍNDICE



PROTOTIPAGEM VISUAL USANDO PROGRAMAÇÃO ORIENTADA A TAREFAS

Neste curso, criaremos aplicações usando um assistente visual para Programação Orientada a Tarefas (TOP). O TOP é um novo paradigma de programação que os desenvolvedores podem usar para prototipar rapidamente aplicativos da web para múltiplos utilizadores. A maneira central de modelar aplicações no TOP é criando tarefas. As tarefas representam partes do trabalho do mundo real que podem ser executadas por pessoas ou por sistemas. Usando várias operações, elas podem ser combinadas em tarefas maiores e mais poderosas.

Vamos explorar os conceitos básicos do TOP estudando alguns aplicações de exemplo, mostrando como modelá-los usando as Tarefas em um ambiente de desenvolvimento visual. O ambiente visual guia os desenvolvedores durante o processo de modelagem. A ferramenta apenas apresenta maneiras sãs de criar e expandir tarefas e fornece dicas sobre como resolver erros de tipo e escopo. Isso resulta num código de programa correto e compilável.

Os alunos são incentivados a estender os exemplos de aplicações de exemplo numa sessão prática. Nossa abordagem visual requer apenas conhecimentos básicos sobre programação e tipos de dados. A introdução no TOP e seus princípios de modelagem são um pré-requisito no curso sobre mTasks.

PROGRAMAÇÃO ORIENTADA A TAREFAS PARA A INTERNET DAS COISAS

A Internet das Coisas (IoT) consiste em dispositivos que detectam, agem e se comunicam com outros sistemas na Internet. Os requisitos típicos dos dispositivos de IoT são que eles devem ser baratos e consumir pouca energia. Isso é conseguido dirigindo os dispositivos IoT por pequenos microprocessadores com pequenas quantidades de memória e poder de processamento. A maioria desses sistemas não possui um sistema operacional adequado e apenas executa um programa específico para executar a tarefa pretendida.

Isso torna a programação da IoT muito desafiadora. O programa único em execução nesse dispositivo deve intercalar todas as subtarefas, como entradas de monitoramento, controle de periféricos e comunicação. Vários dispositivos que cooperam precisam concordar com o protocolo usado e resolver os notórios problemas de programação simultânea.

Nesta palestra, apresentaremos uma introdução prática à Programação Orientada a Tarefas (TOP) para a IoT. Em nossa abordagem TOP, a comunicação entre os dispositivos e seus servidores é tratada de forma transparente pelo sistema mTask. Todo o sistema é programado em um único programa funcional de alto nível. Para cada subtarefa do sistema, definimos um mTask correspondente. Essas subtarefas podem ser compostas por combinadores de tarefas para tarefas mais poderosas.

Essas tarefas podem inspecionar valores intermediários de outras subtarefas, bem como se comunicar com qualquer outra tarefa no sistema via SDS (Shared Data Sources).

Subtarefas para um dispositivo IoT são enviadas dinamicamente para o dispositivo e interpretadas lá. O sistema de tipos fortes evita problemas do tipo de tempo de execução. Essa abordagem TOP simplifica bastante o desenvolvimento de software para a IoT.

References

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.

PINTANDO PROGRAMAS DE VERDE - A EFICIÊNCIA ENERGÉTICA DAS ESTRUTURA DE DADOS

A eficiência energética tem sido uma preocupação para engenheiros de hardware e software de baixo nível há anos [1], [2], [3]. No entanto, o crescente movimento mundial em direção à sustentabilidade, incluindo a sustentabilidade em software [4], combinado com a natureza sistêmica da eficiência energética como um atributo de qualidade, motivou o estudo do impacto energético do software aplicativo em execução. Essa tendência levou os pesquisadores a avaliar técnicas, ferramentas e linguagens existentes para o desenvolvimento de aplicativos a partir de uma perspectiva centrada em energia. Trabalhos recentes estudaram o efeito que fatores como ofuscação de código [5], chamadas da API do Android [6], refatorações de código orientadas a objetos [7], construções para execução simultânea [8] e tipos de dados [9] têm sobre a eficiência energética. Analisar o impacto de diferentes fatores na energia é importante para desenvolvedores e mantenedores de software.

Neste tutorial, analisamos e comparamos a eficiência energética de diferentes implementações para abstrações de dados concretas, como Sequências, Conjuntos ou Coleções Associativas.

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

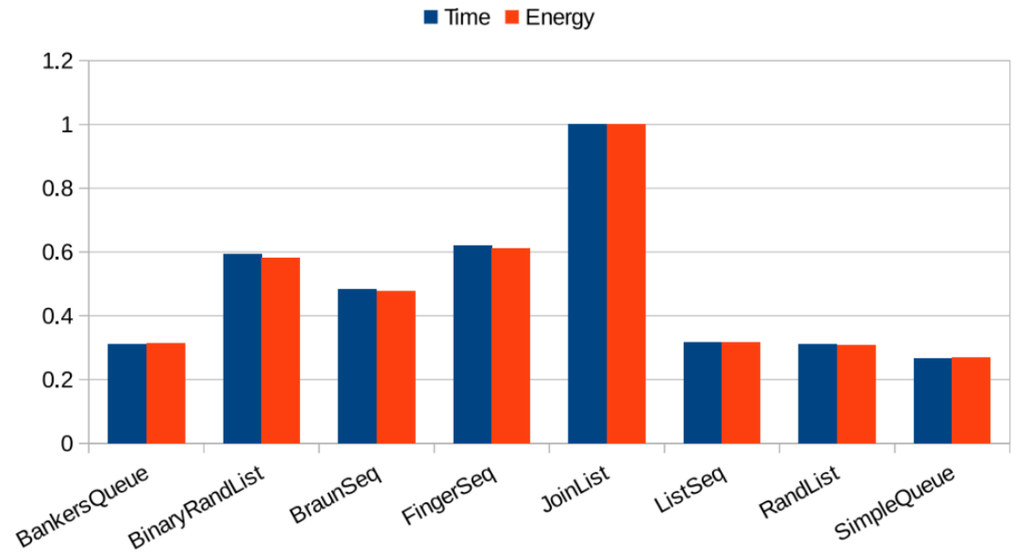
Para cada implementação, inspecionamos como operações como adicionar, excluir ou procurar elementos lidam com diferentes cargas de trabalho. Os assuntos de nosso estudo são uma linguagem de programação funcional [10,11] e uma orientada a objetos [13,14].

Em ambiente funcional, comparámos as implementações apresentadas na Figura 1, ou seja, usando as operações apresentadas na Figura 2.

No domínio orientado a objetos, analisámos as implementações apresentadas na Figura 3, nomeadamente usando as operações apresentadas na Figura 4.

O nosso objetivo é o de fornecer aos desenvolvedores informações acionáveis que já tenham sido integradas em ferramentas de suporte e que podem orientar a construção de software ecológico. Pudemos mostrar que a mesma operação disponibilizada em diferentes implementações pode diferir significativamente em termos de tempo de execução e consumo de energia. Como exemplo, na Figura 5, descrevemos os resultados da operação de remoção para as implementações de abstração de Sequências disponíveis na biblioteca Edison de 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...	

References

[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 NUM CURSO DE ENGENHARIA

O desenvolvimento sustentável tornou-se um tema cada vez mais importante, não apenas na política mundial, mas também um tema cada vez mais central para as profissões de engenharia em todo o mundo. Os engenheiros de software não são uma exceção, como mostrado em várias pesquisas recentes. Apesar da intensa pesquisa sobre software ecológico, o ensino de computação de hoje em dia geralmente falha em abordar nossa responsabilidade ambiental. Apresentamos um módulo sobre software ecológico que introduzimos como parte de um curso avançado em engenharia de software. Apresentamos um catálogo de cheiros de energia e refatorações verdes, que nossos resultados preliminares mostram que ajudam os alunos a raciocinar e otimizar o consumo de energia dos sistemas de software.

References

[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

PERFIL DE ENERGIA DE APLICATIVOS DE SOFTWARE PARA PROJETOS JAVA

Este tutorial aborda a eficiência energética de aplicativos de software implementados na linguagem de programação Java. A primeira parte descreve o estado da arte atual em criação de perfil de energia, bem como opções para exibir o consumo de energia de segmentos do código fonte. A seguir, é apresentado um método de análise de código personalizado para exibição de informações, abordando o processador, a memória operacional e o disco rígido. Após essa análise, é apresentada uma breve introdução ao aplicativo Java implementado. Para demonstrar o uso prático da aplicação, várias soluções de teste são criadas, onde medimos o consumo de energia. Em cada exemplo, enfatizamos a solução de um problema com pelo menos duas soluções para determinar qual implementação possui menor intensidade de energia. Os resultados dos exemplos também fazem parte deste 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);
}
```

References

[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

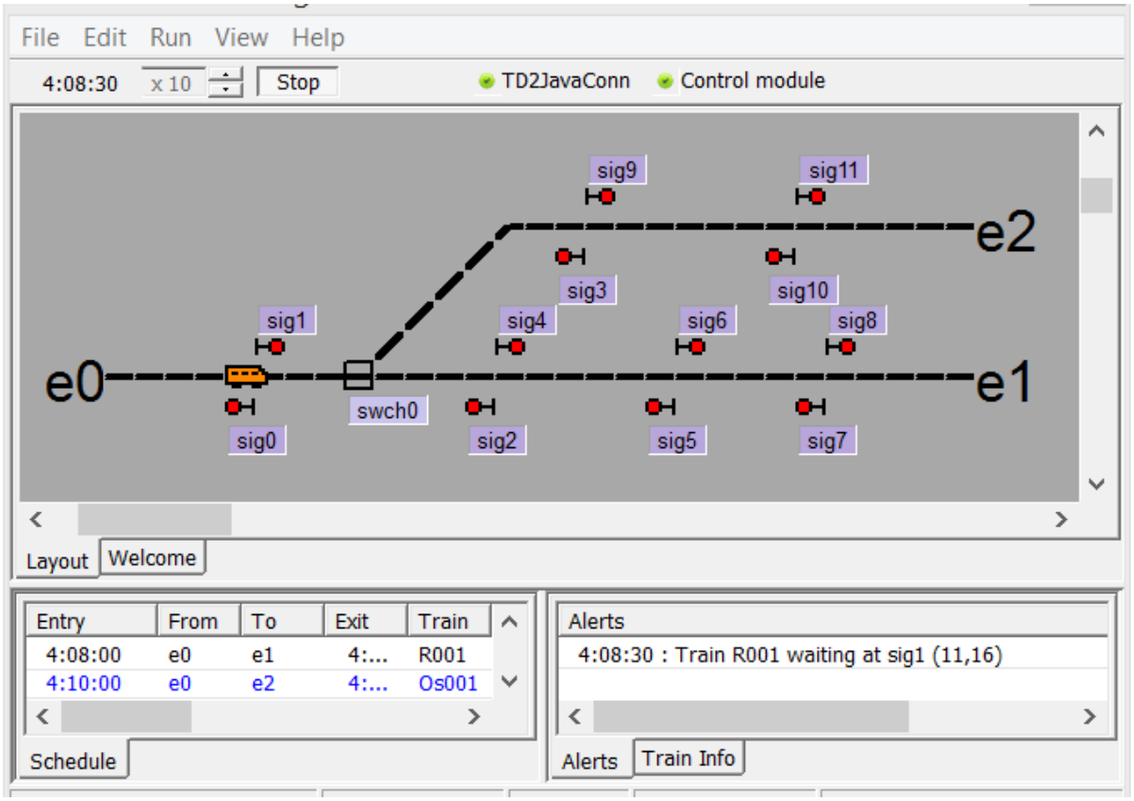
[5] Cs. Szabo, E.M.M. Alzeyani: Measuring Energy Efficiency of Selected Working Software, Studia Universitatis Babe¸s-Bolyai Informatica, Vol. 63, No. 1, 2018, pp. 5–16. https://doi.org/10.24193/subbi.2018.1.01

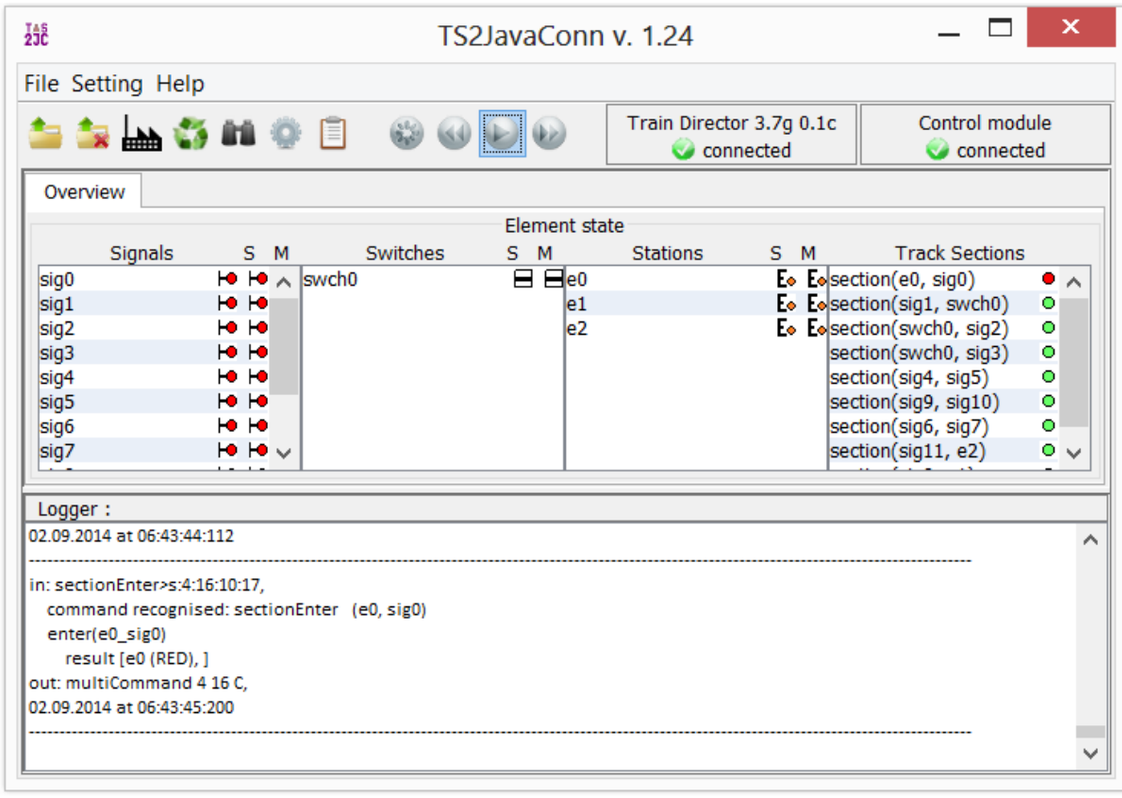
[6] Cs. Szabo, J. Saraiva: Focusing software engineering education on green application development, in Conference of Information Technology and Development of Education – ITRO 2017, Novi Sad, Serbia, pp. 165–169, ISBN 978-86-7672-302-7.

DESENVOLVIMENTO DE SOFTWARE CORRETO COM MÉTODO B

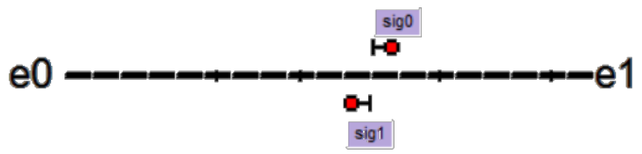
Uma das abordagens bem reconhecidas para o desenvolvimento de sistemas de software corretos é a utilização de métodos formais (FM) para sua especificação e verificação. FM são rigorosas técnicas baseadas matematicamente na especificação, análise, desenvolvimento e verificação de software e hardware. Rigoroso significa que um método formal fornece uma linguagem formal com sintaxe e semântica definidas inequivocamente e com base matematicamente significa que algum aparato matemático (lógica formal, teoria dos conjuntos etc.) é usado para definir a linguagem.

Um dos FM usados na prática industrial é o Método B, um método formal baseado em estado e orientado a modelo, destinado ao desenvolvimento de software. O método B é usado principalmente no setor ferroviário, para o software de segurança crítica por trás dos sistemas automatizados de metrô (incluindo o de Budapeste). A força do Método B reside em um processo de desenvolvimento bem definido, que permite especificar um sistema de software como uma coleção de componentes chamados máquinas B e refinar uma especificação abstrata desse tipo para uma concreta. A especificação concreta pode ser traduzida automaticamente para ADA, C ou outra linguagem de programação. Uma consistência interna do a especificação abstrata e a correção de cada etapa de refinamento são verificadas através da comprovação de um conjunto de predicados chamados obrigações de prova (PObs). Todo o processo de desenvolvimento, incluindo a prova, é suportado por uma ferramenta de software de força industrial chamada Atelier B.

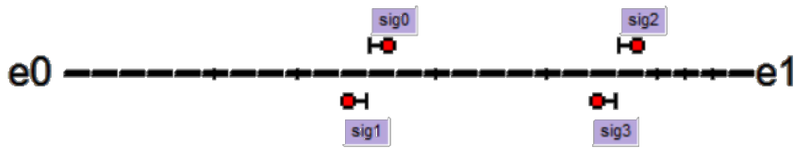




Este tutorial serve como uma introdução prática e gentil ao Método B. Durante o tutorial, os participantes desenvolverão um controlador de software simples para um cenário ferroviário Train Director/TS2JavaConn (TD/TS2JC) toolset [6,7]. O conjunto de ferramentas consiste em uma versão modificada do jogo de simulação Train Director [8] (Fig. 1) e um aplicativo chamado TS2JavaConn (Fig. 2), que permite o uso de controladores de software desenvolvidos separadamente com o jogo de simulação. Com a ambição de usar o conjunto de ferramentas para prototipagem de controladores, mais tarde [9] foi estendido por uma versão personalizada do simulador de trem 3D Open Rails [10].



Após uma introdução ao Método B, os participantes do curso recebem um controlador para um cenário ferroviário de via única com duas seções (Fig.3). O controlador está escrito no idioma do método B. Durante o curso, eles desenvolvem e verificam um controlador para um cenário ferroviário de via única com três seções (Fig. 4).



Listagem 1. O controlador para o cenário ferroviário com duas seções, escritas no idioma do Método 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
```

References

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

PROGRAMAÇÃO DE GESTÃO AVANÇADO E ORQUESTRAÇÃO DE RECURSOS DE REDE VIRTUALIZADOS - SELEÇÃO DE ESTUDOS DE CASO

Novas funções de gestão e orquestração (MANO) são padronizadas para uso em redes distribuídas e virtualizadas ambientes. Sua principal função é fornecer operação segura e confiável de aplicações usando funções de rede. Portanto, como continuação de nossa palestra anterior, onde fornecemos conceitos básicos, aqui nesta palestra forneceremos uma seleção de casos estudos onde essas funções são implementadas e explicam os mecanismos avançados por trás e a simplicidade de inscrição.

References

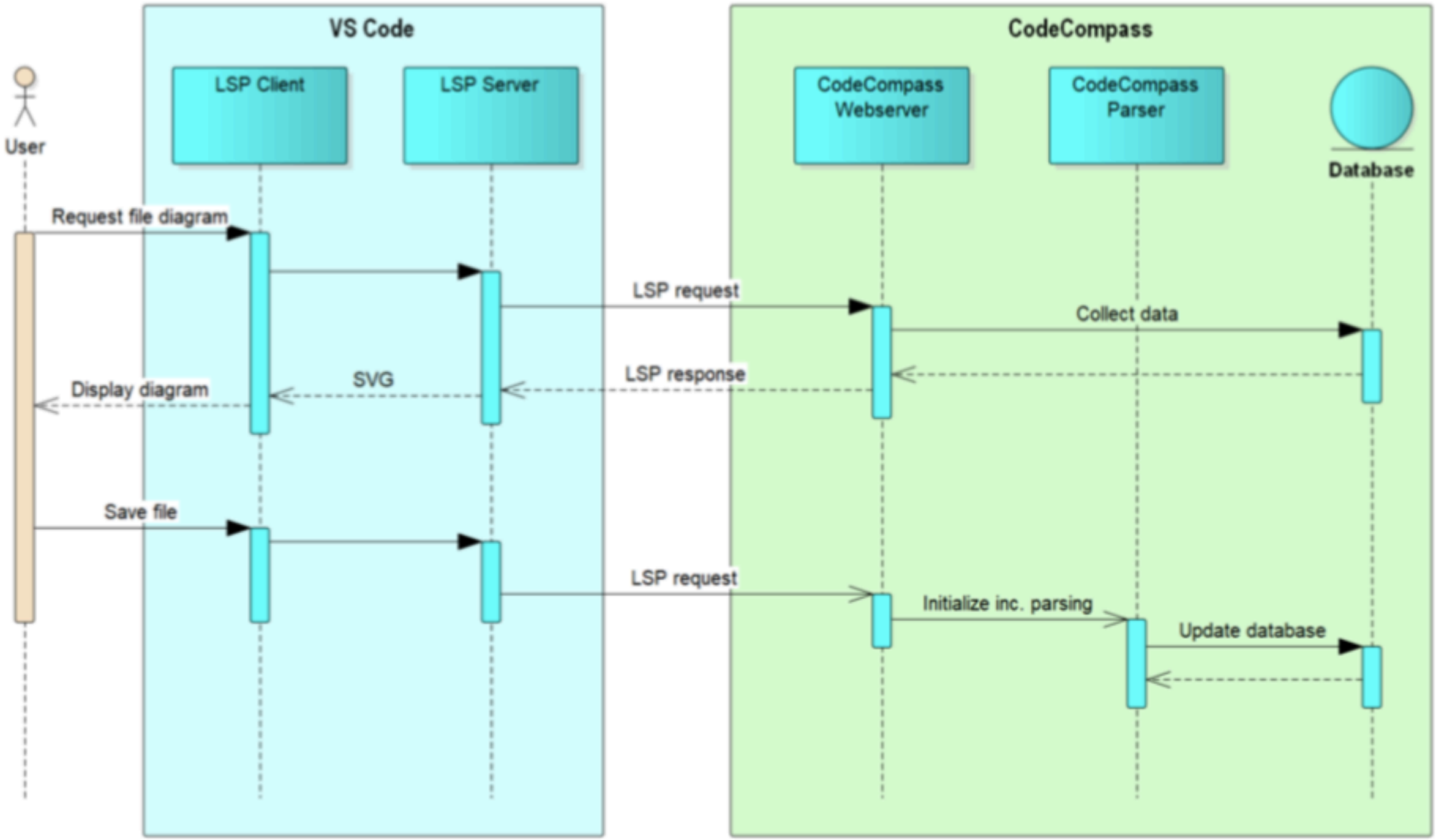
[1] ETSI Industry Specification Group (ISG) NFV: ETSI GS NFV- MAN 001 v1.1.1: Network Functions Virtualisation (NFV); Management and Orchestration European Telecommunications Standards Institute (ETSI), 2014, https://www.etsi.org/deliver/etsi_gs/NFV- MAN/001_099/001/01.01.01 60/gs_NFV-MAN001v010101p.pdf,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

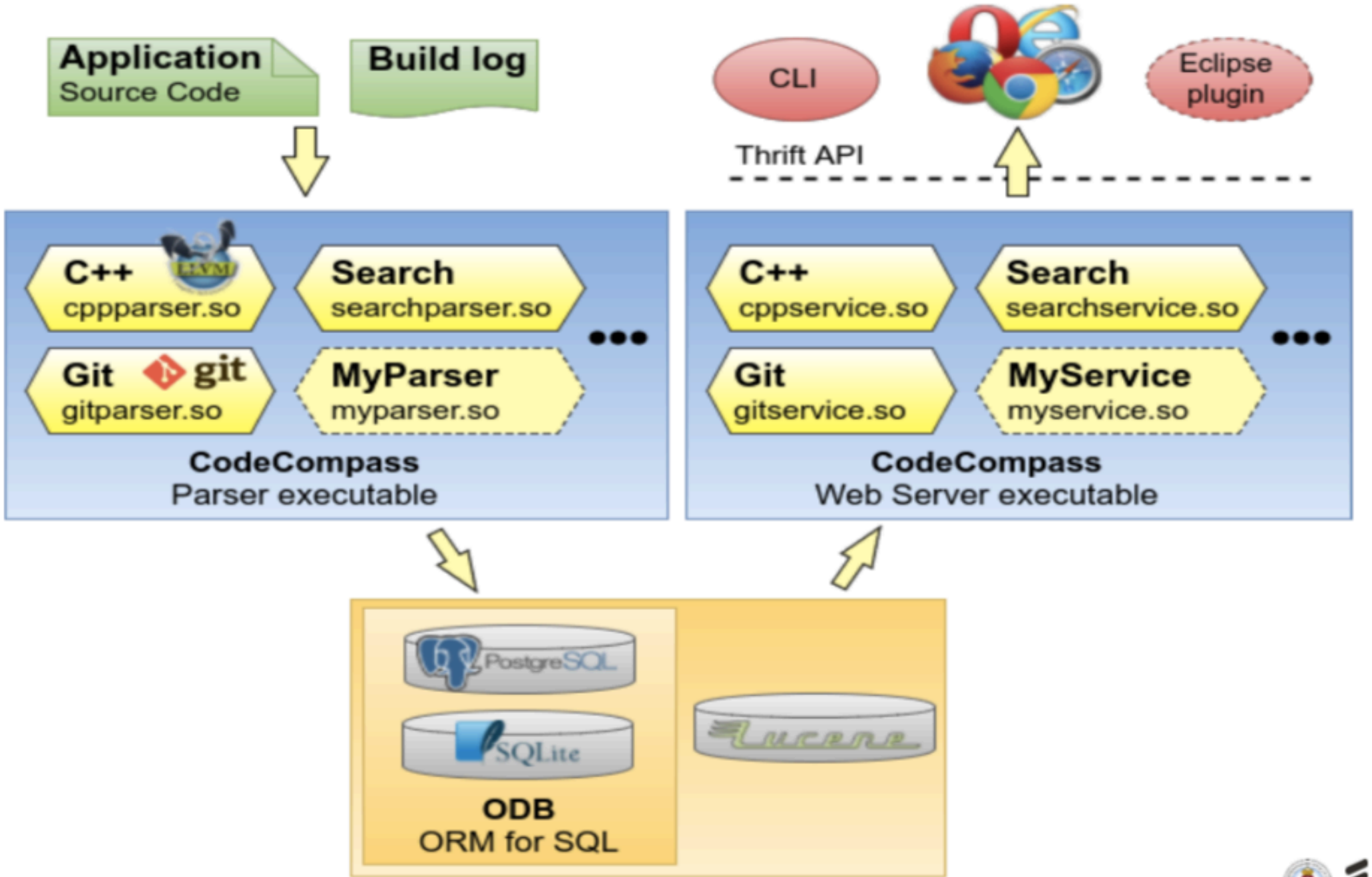
[2] Han, B., Gopalakrishnan, V., Ji, L., Lee, S.: Network function virtualization: Challenges and opportunities for innovations. IEEE Communications Magazine 53(2), 90–97 (2015)

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

COMPREENSÃO DE CÓDIGO COM SUPORTE AVANÇADO A FERRAMENTAS

Neste tutorial, apresentaremos as ferramentas de compreensão de código de ponta para os alunos. Vamos dar uma teoria base para a compreensão do código, métodos de navegação e visualização de código e abordagens para aplicá-los na prática desenvolvimento de software. Na sessão prática, demonstraremos como configurar um conjunto de ferramentas específico: CodeCompass com análise incremental, Visual Studio Code como ferramenta front-end e o uso do Language Server Protocol. Analisaremos uma biblioteca de código aberto e encontraremos e corrigiremos um bug específico usando o conjunto de ferramentas.


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"
            }
        }
    }
}
```

References

[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.

PROGRAMAÇÃO DE MATRIZ FUNCIONAL COM ATRIBUIÇÃO ÚNICA C: OPORTUNIDADES E DESAFIOS

O SAC (Single Assignment C) é, em vários aspectos, uma linguagem de programação funcional fora do comum. Como o nome sugere, o SAC combina uma sintaxe do tipo C com uma semântica puramente funcional e sem estado. Originalmente motivada para facilitar a adoção por programadores com um histórico imperativo, a escolha oferece visões surpreendentes sobre o que constitui uma construção de linguagem de programação imperativa "típica" ou funcional "típica". Novamente do lado exótico para uma linguagem funcional, o SAC enfatiza matrizes multidimensionais, em vez de listas e árvores. A programação de matrizes trata matrizes multidimensionais de uma maneira holística: as funções mapeiam matrizes de argumentos potencialmente enormes para resultar em matrizes com uma semântica de chamada por valor, e as novas operações de matrizes são definidas pela composição das existentes. O SAC é uma linguagem de alta produtividade para domínios de aplicações que lidam com grandes coleções de dados de maneira computacionalmente intensiva.

Ao mesmo tempo, o SAC também é uma linguagem de alto desempenho que compete com linguagens imperativas de baixo nível através de tecnologia de compilação. A visão abstrata em matrizes combinada com a semântica funcional suporta programas de longo alcance transformações. Um sistema de tempo de execução altamente otimizado cuida do gerenciamento automático de memória, com foco em recursos imediatos reutilização de memória. Por último, o compilador SAC explora a semântica sem estado do SAC e a natureza paralela aos dados do SAC programas para aceleração totalmente direcionada ao compilador em uma grande variedade de arquiteturas de máquinas contemporâneas, desde servidores para aceleradores de GPGPU e clusters de estações de trabalho.

As palestras motivam o design da linguagem do SAC e fornecem uma introdução prática à programação de array como um paradigma. Examinamos todos os aspectos, desde o cálculo do array subjacente até o design da linguagem concreta, com procurando código funcional, discuta uma infinidade de exemplos, explore desafios de compilação e, eventualmente, veja alguns resultados de desempenho em várias arquiteturas de computação paralela.

PADRÕES DE COMPUTAÇÃO DISTRIBUÍDA EQUILIBRADA

O desenvolvimento de software simultâneo de última geração fez uso extensivo de várias metodologias e abordagens para obter alta velocidade. No entanto, o paralelismo continua sendo um dos domínios mais difíceis, especialmente no caso de padrões baseados abordagens de programação. O principal objetivo é explorar esquemas de computação paralela em um novo ambiente, para ilustrar a adequação e aplicabilidade em novas configurações de computação distribuída. A quantidade de paralelismo é explorada com base em muitos fatores, como: padrão de computação aplicado, granularidade refinada, semântica de nós distribuídos, fluxo de dados, e especialmente balanceamento de carga.

References

[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.