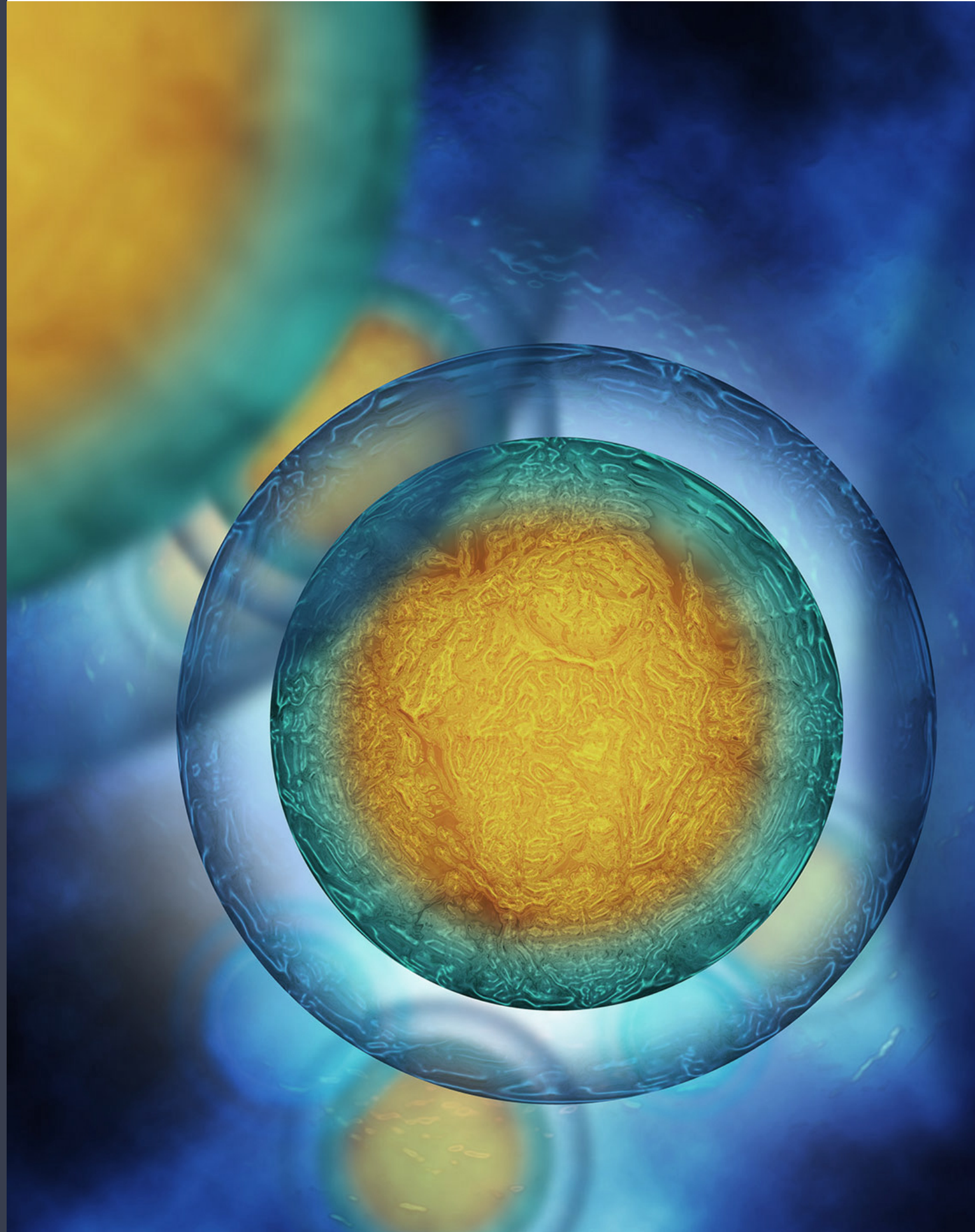
FE3CWS

MATERIAL DE TREINAMENTO DE PROFESSORES DE AMSTERDAM

Produção intelectual 2 do projeto ERASMUS + 2017-1-SK01-KA203-035402



Algumas palavras sobre o

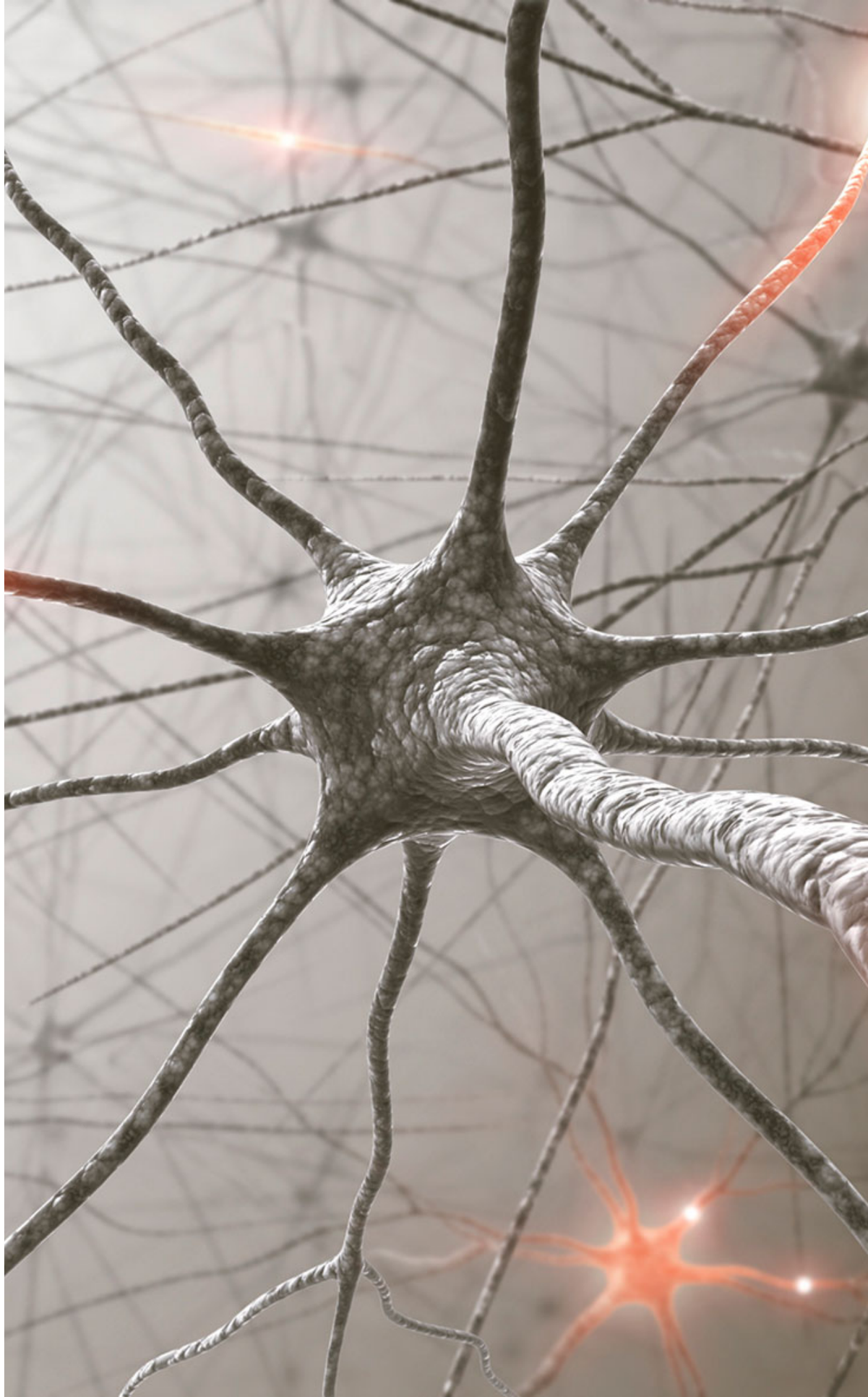
CONTEÚDO

- 6 tópicos relacionados à composição, compreensão e correção de software
- Disponível em 7 idiomas: inglês, húngaro, eslovaco, croata, romeno, búlgaro e português

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

ÍNDICE

1. Computação em nuvem centrada no utilizador na educação
2. Focando a educação em medições de eficiência energética durante o teste de software
3. Uma Disciplina em Engenharia de Software Verde
4. Ensino de Programação Oreintada a Tarefas
5. Uma Abordagem Interactiva no Ensino de Redes de Petri
6. CodeCompass: Uma Framework Extensível para Compreenção de Código



Computação em nuvem centrada no utilizador na educação

Ana Oprescu

Universidade de Amsterdão,
Amsterdão, Holanda
a.m.oprescu@uva.nl

Resumo A computação em nuvem tornou-se uma tecnologia essencial e, portanto, parte de muitos currículos de ciência da computação. A pesquisa centrada no utilizador concentra-se em estratégias para estimar tempos de execução e custos para disponibilizar aplicativos do mundo real na nuvem.

1 Computação em nuvem centrada no utilizador

Dentro da área de computação em nuvem, um aspecto importante é ajudar os usuários em suas decisões. Tais decisões estão relacionadas às seguintes perguntas:

- Como é que o aplicativo se comporta nos recursos virtualizados?
- Quantos recursos virtuais de que tipo de qual provedor de nuvem deve ser adquirido para implantação do aplicativo?
- Por quanto tempo? Quanto vai custar?

Essas perguntas são tipicamente modeladas como um problema de planejamento, trabalhando sob a suposição de que não há conhecimento a priori sobre o aplicativo. Um conjunto simples e simples de requisitos é que o aplicativo é implantado com sucesso e os custos são minimizados.

1.1 A arquitetura de um escalonador para uma aplicação em nuvem

O escalonador do BaTS [4] foi desenvolvido para ajudar os usuários a implantar seus aplicativos na nuvem. É necessária uma abordagem de agendamento automático para conseguir isso e verifica regularmente o progresso da implantação

A figura 1 descreve a arquitetura do BaTS. Durante a *sampling phase*, o BaTS coleta estatísticas sobre os tempos de execução de algumas das tarefas do aplicativo, usando amostragem com substituição. Aqui, apenas uma pequena amostra é necessária (30 a 50 tarefas) para calcular a média e o desvio padrão do tempo de execução das tarefas em várias ofertas de nuvem. A regressão linear é usada para otimizar o cálculo das estimativas de orçamento e produção.

Durante a *fase de execução*, em intervalos regulares, a configuração atual é reavaliada, para verificar se o agendamento selecionado ainda é viável. Se uma

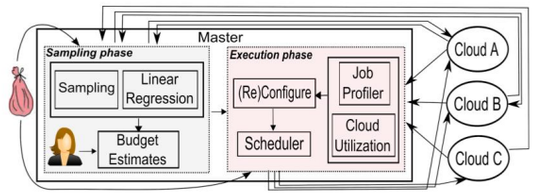


Figura 1. Arquitectrura do BaTS.

violação do orçamento for esperada, serão adquiridas máquinas mais lucrativas (melhor relação preço/desempenho). Se uma violação de fabricação for esperada, máquinas mais rápidas serão adquiridas.

2 Usando a metodologia BaTS nos recursos da AWS

Apresentamos o problema de ajudar os proprietários de aplicativos que procuram selecionar as melhores opções em termos de recursos virtualizados ao implantar seu aplicativo nos recursos do Amazon EC2 (AWS) [7].

2.1 A fase de amostragem otimizada

A ideia principal é usar o tempo médio de execução para cada tipo de recurso virtualizado para calcular orçamento e makespan estimativas. Contudo, a obtenção dessas estatísticas pode resultar em custos, considerando os vários tipos de ofertas do AWS EC2 (atualmente 123 [8]). A figura 2 é exibida aleatoriamente tarefas selecionadas de um aplicativo em que os tempos de execução das tarefas seguem alguns distribuição. Os tempos de execução dessas tarefas são usados para calcular as estatísticas de um oferta na nuvem. Após coletar as estatísticas de todas as ofertas de nuvem disponíveis, podemos calcular o orçamento e estimativas de produção. Supondo que gostaríamos de avaliar todas as ofertas atuais do AWS EC2, isso significaria a execução de 30 tarefas em cada um dos 123 tipos de máquinas. Se simplesmente executássemos conjuntos diferentes de tarefas selecionadas aleatoriamente (no total 3690), isso levaria para uma longa (e cara) fase de amostragem, possivelmente tornando irrelevante qualquer decisão do usuário por dois razões: a) restam poucas tarefas a serem executadas eb) o orçamento do usuário já pode ter sido excedido. Se executássemos o mesmo conjunto selecionado aleatoriamente em cada tipo de máquina, isso ainda levaria a uma fase de amostragem longa (e cara).

Otimizamos essa fase usando regressão linear para reduzir o número total de tarefas que precisam ser executadas antes da preparação das estatísticas. Executamos o mesmo conjunto de 7 tarefas selecionadas aleatoriamente em cada tipo de máquina e coletamos tempos de execução [9]. Em seguida, executamos 23

tarefas selecionadas aleatoriamente nas máquinas que primeiro se tornam disponíveis. A figura 3 ilustra nossa abordagem. Usando os tempos de execução das 7 tarefas replicadas, estabelecemos um relacionamento linear entre os tempos de execução das tarefas do aplicativo em todos os tipos de máquinas. Usamos essas relações lineares para mapear os 23 tempos de execução para todos os outros tipos de máquinas. Depois que o mapeamento é concluído, temos um conjunto de 30 tempos de execução para cada tipo de máquina, enquanto executamos apenas 884 tarefas em vez de 3690.

2.2 Diferentes ordens de magnitude nos preços

Depois que as estimativas de tempo de execução são obtidas, as estimativas de orçamento e de produção são calculadas usando um algoritmo modificado de Bounded Knapsack [11]. No entanto, ao considerar os recursos do AWS EC2 com um modelo de preço diferente, como instâncias spot, essa abordagem não aumenta devido às diferentes ordens de magnitude.

Para resolver esse problema, trocamos o determinismo da abordagem Bounded Knapsack pela escalabilidade de uma abordagem de algoritmo genético [12]. Usando um algoritmo genético, podemos aproximar a frente de Pareto (conjunto ótimo) de agendas viáveis para uma determinada aplicação e um determinado conjunto de tipos de máquinas. A Figura 4 mostra a frente real de Pareto e duas estimativas para um aplicativo com uma distribuição multimodal de tempos de execução da tarefa. Nossa solução gera frentes precisas de Pareto em 1 segundo.

A principal conclusão aqui foi que, para garantir uma boa cobertura da frente real de Pareto, a função de condicionamento físico também deve recompensar o makepan mais rápido/mais barato.

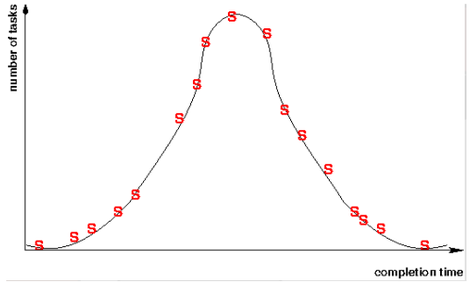


Figura 2. Fase de amostragem padrão.

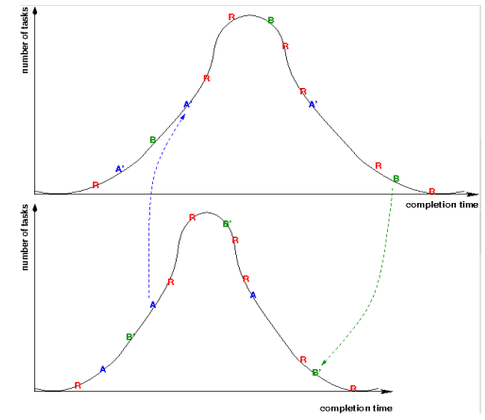


Figura 3. Fase de amostragem otimizada.

2.3 A última fase da computação

Na fase final da computação, a suposição subjacente de que o tempo de computação é "fluido"precisa ser abordada [10]. Neste ponto, por definição, o aplicativo contém poucas tarefas para a suposição. Analisamos várias abordagens para resolver esse problema, concentrando-se em estimar melhor as necessidades finais de computação da aplicação, de modo que a alocação das tarefas finais para as máquinas seja ideal. Nossas abordagens variaram do conhecimento perfeito dos tempos de execução restantes ao conhecimento zero (tempos de execução aleatórios). O impacto foi insignificante. Portanto, o problema precisava ser tratado em um estágio diferente, a saber, na fase de estimativa do orçamento e da produção.

Para esse fim, o BaTS mantém o controle das frações estimadas de unidades de tempo finais não utilizadas estimadas. O BaTS fornece uma almofada para lidar com discrepâncias que estão fora da unidade de tempo final responsável e adiciona recursos virtualizados e/ou tempo ao cronograma. No entanto, os valores discrepantes ainda podem levar a violações.

3 Usando a metodologia BaTS na educação

No currículo de Mestrado em Ciência da Computação da Universidade de Amsterdã, o curso "Serviços da Web e sistemas baseados em nuvem"é ministrado há

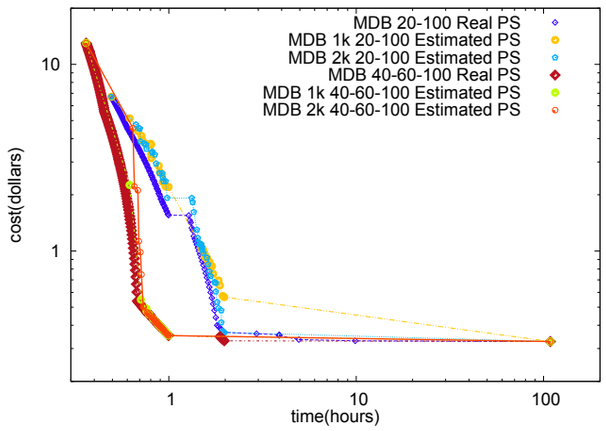


Figura 4. Pareto fronts de um aplicativo com uma distribuição multimodal de tempos de execução de tarefas e dois conjuntos de recursos diferentes

vários anos. Um objetivo importante deste curso é familiarizar os alunos com os desafios dos sistemas baseados em nuvem. O trabalho prático deste curso envolve o desenvolvimento de um planejador rudimentar para recursos virtualizados e a análise de seu comportamento em um ambiente interno versus um comercial na nuvem.

A configuração interna consiste em uma implantação do OpenNebula [5] no DAS [6], o cluster de computação nacional holandês. O OpenNebula é uma solução de código aberto para implantações de infraestrutura como serviço: recursos físicos são gerenciados e oferecidos como recursos virtuais. Aqui, os alunos têm um limite superior no número de máquinas virtuais que podem ser acionadas simultaneamente.

A configuração da nuvem comercial consiste nas ofertas de nuvem Amazon EC2 [7]. Aqui, os alunos têm um orçamento que podem ser usados para qualquer aquisição de recursos relacionados ao laboratório. A avaliação de seu trabalho de laboratório leva em consideração qualquer violação do orçamento.

A carga de trabalho que o planejador precisa gerenciar é um aplicativo paralelamente embaraçoso: uma coleção de tarefas de processamento de cadeias relacionadas à análise de sentimentos das mensagens do Twitter.

Em geral, o resultado do trabalho de laboratório mostrou que os alunos foram capazes de entender a diferença entre a aquisição de recursos virtuais por esforços e comercial. Eles geralmente desenvolviam agendadores orientados pela lucratividade, enquanto os alunos com melhor desempenho desenvolviam agendadores mais sofisticados, onde os usuários podiam escolher entre várias políticas: mais rápido, mais barato e mais rentável.

4 Conclusão & Trabalho Futuro

Abordagens estocásticas para agendamento de nuvem centrado no usuário são promissoras. A incorporação da pesquisa na educação assim que ela atinge algum estado estável é muito importante.

Como trabalho futuro, gostaríamos de oferecer suporte às funções do Haskell AWS Lambda implantadas por meio da implementação da API do Haskell AWS.

Agradecimentos

Este artigo é parte da produção intelectual O2 do projeto Erasmus+ Key Action 2 project No. 2017-1-SK01-KA203-035402, the Strategic Partnership for Higher Education *"Focusing Education on Composability, Comprehensibility and Correctness of Working Software" (3COWS).*

Aviso Legal

As informações e visões definidas neste artigo são de responsabilidade do(s) autor(es) e não reflectem necessariamente a opinião oficial da União Europeia. Nem instituições e órgãos da União Europeia, nem qualquer pessoa que atue nome pode ser responsabilizado pelo uso que possa ser feito do informações nele contidas.

Referências

1. Newman, Sam. Building Microservices. O'Reilly Media, Inc., 2015.
2. Fowler, M., Lewis, J.: Microservices. `http://martinfowler.com/articles/microservices.html` (March 2014), Last accessed: 15-08-2018
3. Balalaie, Armin, Abbas Heydarnoori, and Pooyan Jamshidi. "Migrating to cloud-native architectures using microservices: An experience report."arXiv preprint arXiv:1507.08217 (2015).
4. AM Oprescu. Stochastic Approaches to Self-Adaptive Application Execution on Clouds. PhD Thesis, Amsterdam, Vrije Universiteit, 2013.
5. `https://opennebula.org/`, Last accessed: 15-11-2018.
6. `https://www.cs.vu.nl/das5/`, Last accessed: 15-11-2018.
7. `https://console.aws.amazon.com/ec2/v2/home`, Last accessed: 15-11-2018.
8. `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html`, Last accessed: 15-11-2018.
9. A.-M. Oprescu; T. Kielmann; H. Leahu, *Budget estimation and control for bag-of-tasks scheduling in clouds*, 2011, Parallel Processing Letters, vol. 21.

10. A.-M. Oprescu; T. Kielmann; H. Leahu, *Stochastic tail-phase optimization for bag-of-tasks execution in clouds*, 2012, Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing.
11. A.-M. Oprescu; T. Kielmann; *Bag-of-tasks scheduling under budget constraints*,2010, IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom).
12. A. Vintila; A.-M. Oprescu; T. Kielmann; *Fast (re-) configuration of mixed on-demand and spot instance pools for high-throughput computing*, 2013, Proceedings of the first ACM workshop on Optimization techniques for resources management in clouds.

Focando a educação em medições de eficiência energética durante o teste de software

Csaba Szabó[0000−0001−5147−2452]

Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics, Technical University of Košice,
Letná 9, 04200 Košice, Slovakia
csaba.szabo@tuke.sk

Resumo A missão dos professores de engenharia de software é preparar futuros engenheiros de software que possam dominar todos os problemas durante todo o ciclo de vida do software. Além das habilidades relacionadas à compreensão das necessidades das partes interessadas e à composição do software de trabalho correspondente, a capacidade de verificar a exatidão dos resultados também desempenha um papel importante. Neste artigo, focamos no último conjunto de habilidades. Nós nos concentramos em testes, onde o nível de automação é menos importante. Enfatizamos a natureza da medição dos testes, mais precisamente, nos concentramos na medição do consumo de energia de software que pode ser fornecida durante o teste em diferentes níveis. Todos esses aspectos são apresentados do ponto de vista de um educador de engenharia de software, com o objetivo de apresentar como configurar uma sessão de laboratório de engenharia de software que se concentre na medição de eficiência energética durante testes de software e, com os comentários dos autores sobre esta proposta.

Keywords: Consumo de Energia · Medição · Engenharia de Software na Educação · Teste de Software.

1 Definição do Problema

Um dos desafios para os fabricantes de baterias é quanto tempo a bateria pode operar sem ser continuamente carregada e, é claro, existem muitos outros desafios, como o tamanho que afeta muito a forma do dispositivo e o outro fator que é uma característica importante da bateria. a bateria é o peso do identificador. A bateria é considerada um pouco mais leve em comparação com o dispositivo que precisa de uma bateria para funcionar. O desafio aqui é como diminuir e diminuir o tamanho e, certamente, uma alta eficiência em termos de tempo de operação do dispositivo móvel sem ser carregado.

No topo desse desafio de hardware, existe o seu desafio de software irmão, a saber, que o próprio software deve suportar economia de energia. Fazer isso sem limitar a experiência do usuário é considerado hoje em dia um objetivo silencioso, mas importante, de cada desenvolvimento de software direcionado a qualquer tipo de dispositivo portátil.

Lembrar que o consumo de energia de qualquer dispositivo móvel é influenciado a partir dos aplicativos em execução, desde o nível de acesso dos serviços básicos até o humor real do usuário, desenvolver software para esses dispositivos já é um desafio [1]. Pode-se dizer que o desafio do desenvolvimento de software é sempre o mesmo, mas temos que apontar a "mobilidade" como propriedade chave do sistema aqui. O status da energia da bateria também determina o desempenho do sistema devido à configuração no nível do sistema operacional - conhecida como "preferências de economia de energia".

É ainda mais difícil se um (no nosso caso, o professor) tiver que preparar os alunos para esses desafios. Todas as "melhores práticas" e "dicas de economia de energia" já conhecidas devem ser apresentadas em um contexto que seja facilmente compreensível para os alunos. Isso pode ser feito posicionando os conceitos em um ambiente conhecido, como teste de software e automação de teste [2], no nosso caso. É isso que pretendemos com este artigo, este é o conteúdo das próximas seções, começando com a proposta, seguida de uma avaliação e finalizando com mais dicas de melhoria.

2 Solução Proposta

Como foi afirmado acima, temos que encontrar o melhor ambiente adequado para introduzir medidas de consumo de energia [3] e práticas de avaliação de eficiência energética. Poderia ser tanto o desenvolvimento inicial do software quanto a evolução do software, pois as duas fases do desenvolvimento do ciclo de vida genérico do software oferecem oportunidades para medir o produto que está sendo desenvolvido/desenvolvido [4].

Com a escolha do desenvolvimento inicial do software, o benefício é que todas as atividades podem se concentrar em questões relacionadas à economia de energia, enquanto a escolha da alternativa de evolução do software oferece a possibilidade de avaliar a melhoria na implementação do produto. Por outro lado, a evolução do software requer a existência de um software em funcionamento no início, enquanto o desenvolvimento inicial de software é o processo que cria o produto começando com o primeiro requisito do software.

Colocando as duas abordagens possíveis no ambiente de ensino, a melhor opção seria usar as duas. Um semestre para o desenvolvimento inicial de software e outro para a evolução do mesmo software.

Normalmente, o professor não tem dois semestres consecutivos para apresentar o conteúdo do curso da maneira apresentada no parágrafo anterior. Essa é a razão pela qual temos que decidir que tipo de desenvolvimento usar para introduzir as práticas selecionadas. Do ponto de vista da arquitetura do processo de desenvolvimento, e pelo fato de o desenvolvimento inicial também poder ser evolutivo, decidimos pela evolução do software. Inclui muitas atividades de desenvolvimento inicial (exceto coleta e análise antecipada de requisitos) e enfatiza a importância dos testes e avaliações.

Essa opção permite ao professor:

– Deixar os alunos olharem para trás em seu histórico de desenvolvimento (projetos anteriores) para serem críticos para si mesmos.
– Deixar os avaliar seus resultados usando métricas de código e medição (ou estimativa) de consumo de energia.
– Deixar os integrar as atividades acima nos processos padrão de verificação & e validação da evolução do software.

A linguagem de programação do desenvolvimento não é importante; portanto, o aluno pode selecionar qualquer um dos seus projetos anteriores para a evolução - ou todos eles, se estiverem competindo no número de projetos ou linguagens de programação utilizados. Porém, a linguagem de programação geralmente determina ou limita o ambiente de desenvolvimento e as ferramentas usadas. A seleção dessas ferramentas e seus plug-ins também oferece um bom suporte para avaliação de métricas de código.

A medição do consumo de energia e a avaliação da eficiência energética geralmente requerem uma ferramenta diferente, pois existem poucos ambientes de desenvolvimento que integram a medição ou estimativa do consumo de energia até o momento.

Em relação aos testes como base de medição, devemos observar que a análise estática de código é usada para fazer parte dos testes de software. Além disso, partes selecionadas da base de código do aplicativo estão sendo executadas durante o teste de caixa branca (principalmente teste de unidade), que por uma pequena extensão da medição do consumo de energia pode apresentar o consumo de energia do caso de teste - uma visão indireta do consumo de energia de o código testado. Durante o teste da caixa preta, o aplicativo completo está sendo testado usando cenários de teste. Esses cenários de teste são principalmente comparáveis aos casos de uso do software para uso diário, outros representam os cenários limítrofes - incluindo aqueles em que o usuário está de mau humor. Medir o consumo de energia da execução desses testes de caixa preta dá uma olhada aproximada (mas direta) no consumo de energia do produto.

Aqui está o principal benefício da evolução (quando comparado ao desenvolvimento inicial de software). O professor pode preparar a versão inicial para a evolução, incluindo código que pode ser aprimorado, lista de bugs conhecidos e a base de teste! A existência de teste para reteste e teste de regressão é muito importante aqui, pois a produtividade da evolução pode ser aumentada por esse recurso.

Assumindo que todas as etapas acima foram feitas, a integração das medidas de eficiência energética no processo de teste é vista da perspectiva das atividades do aluno da seguinte forma:

1. Selecione o produto que poderia ser seu projeto anterior ou em um repositório.
2. Avalie usando análise de código estático, teste, medição de energia, pesquisa de usabilidade etc.
3. Melhorar (tipos diferentes de evolução, como adicionar / alterar funcionalidade, reparar ou adaptar)
4. Teste novamente para ter certeza de que você eliminou um defeito ou falha
5. Teste de regressão (incluindo reavaliação)
6. Concluir resultados (faça um veredicto final em todos os dados disponíveis, incluindo eficiência energética)

3 Discussão

Como a medição do consumo de energia é relativamente nova em comparação com outras técnicas, como análise de código estático, teste de caixa preta / branca e depuração, pode ser o ponto de falha. Mas se for combinado com esses princípios anteriores, criando uma pontuação composta para cada aluno, a propriedade crítica será muito menor.

Observando as possibilidades de classificação, podemos encontrar vários "níveis de liberdade", que se oferecem para serem usados separadamente ou como parte de uma composição de nota:

1. número do item de diferentes projetos,
2. número do item das linguagens de programação aplicadas / usadas,
3. qualidade do código do item do produto final,
4. eficiência energética do produto final,
5. melhoria da qualidade do código do item durante a evolução do software,
6. melhoria da eficiência energética durante a evolução do software.

Considerando todos os "níveis de liberdade" de conclusão de tarefas, muitas competições podem ser definidas para os alunos competitivos, enquanto os realizadores coletam "pequenas vitórias" em número de projetos, o objetivo dos perfeccionistas é otimizar todo o código métricas e minimizar o consumo de energia. Para o aluno médio, a melhoria da eficiência energética e a compreensibilidade do código podem ser a meta alcançável.

A competição dos estudantes pode ser ainda mais apoiada, não permitindo que eles retornem aos seus projetos anteriores, mas permitindo que eles escolham a partir de um repositório selecionado. Como nos jogos de computador, todos os níveis estão igualmente disponíveis para todos. Também para apoiar a eqüidade, também poderia ser criado um fórum público comum de alunos e professores.

Nosso trabalho futuro nessa área se concentrará na configuração de um ambiente portátil integrado de desenvolvimento, testes e estimativa de consumo de energia. Esse ambiente será usado no quadro da evolução do software ou dos assuntos de desenvolvimento inicial para apoiar a educação sobre a medição da eficiência energética durante os testes de software. Isso pode limitar a criatividade dos alunos ao oferecer uma caixa de areia semi-fechada, pois o hardware desempenha um papel muito importante na arquitetura atual de estimativa de consumo de energia. Algumas pesquisas visam romper essa limitação - estamos ansiosos por esses resultados para integrá-los.

Agradecimentos

Este artigo é parte da produção intelectual O2 do projeto Erasmus+ Key Action 2 project No. 2017-1-SK01-KA203-035402, the Strategic Partnership for Higher Education *"Focusing Education on Composability, Comprehensibility and Correctness of Working Software" (3COWS).*

Aviso Legal

As informações e visões definidas neste artigo são de responsabilidade do(s) autor(es) e não reflectem necessariamente a opinião oficial da União Europeia. Nem instituições e órgãos da União Europeia, nem qualquer pessoa que atue nome pode ser responsabilizado pelo uso que possa ser feito do informações nele contidas.

Referências

1. J. Saraiva, M. Couto, Cs. Szabó, D. Novák: 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
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. Porkoláb, 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. Cs. Szabó, 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.

Uma Disciplina em Engenharia de Software Verde *

João Paulo Fernandes[1] and João Saraiva[2]

[1] CISUC & Universidade de Coimbra, Portugal
[2] HASLab/INESC TEC & Depart. de Informática, Univ. do Minho, Portugal
jpf@dei.uc.pt , saraiva@di.uminho.pt

Resumo Este relatório técnico descreve a pesquisa desenvolvida no Green Laboratório de Software das Universidades de Coimbra e Minho, que foi apresentado na primeira reunião de formação de professores do Erasmus + projeto "Focusing Education on Composability, Comprehensibility and Correctness of Working Software". Apresenta um *ranking* verde para linguagens de programação e estruturas de dados e técnicas para localizar uso anormal de energia em sistemas de software.

Keywords: Computação Verde, Energy-aware Software, Análise de Código Fonte

1 Motivation

O uso generalizado atual de computação sem fio, mas poderosa dispositivos, como smartphones, laptops etc., está mudando a maneira como fabricantes de computadores e engenheiros de software desenvolvem seus produtos. De fato, o tempo de execução do computador/software, que foi o objetivo primário no século passado, não é mais o único preocupação. O consumo de energia está se tornando um gargalo crescente para sistemas de hardware e software. Como conseqüência, a pesquisa sobre O software verde é uma área de pesquisa relevante e ativa.

Este relatório descreve brevemente a pesquisa que está sendo desenvolvida em software verde no Green Software Laboratory (GSL). GSL consiste em vários grupos de pesquisa portugueses, incluindo dois locais do projeto "'Focusing Education on Composability, Comprehensibility and Correctness of Working Software". GSL é uma iniciativa para desenvolver técnicas e ferramentas destinadas a reduzir o consumo de energia vários sistemas de computação (móveis, programas, bancos de dados etc.). GSL foco especificamente no lado do software, onde ele se aplica (fonte técnicas de análise e transformação para detectar anomalias em consumo de energia e definir otimizações para reduzir tais consumo.

* Este trabalho for parcialmente financiado por ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, e pela FCT - Fundacão para a Ciência e a Tecnologia através do projeto POCI-01-0145-FEDER-016718 and UID/EEA/50014/2013.

No século passado, a eficiência de um sistema de software foi focada principalmente no tempo de execução e na eficiência do consumo de memória. Hoje em dia, desenvolvedores de software costumam fazer a pergunta "um programa mais rápido é também um programa mais ecológico?". Existem muitos aspectos de um sistema de software que influencia seu desempenho energético: a linguagem de programação e seu modelo de execução (compilado em código binário ou em uma máquina virtual, código interpretado, avaliação lenta versus rigorosa, uso de tempo de execução avaliação parcial, etc). A eficiência do modelo de memória e bibliotecas de idiomas também influenciam o desempenho. A complexidade do algoritmo usado para implementar o problema de computador desejado, também influencia o desempenho: se o algoritmo implementado precisar fazer mais mais do que o estritamente necessário, mais CPU e energia serão usava.

Neste documento, relatamos brevemente os resultados da pesquisa alcançados em GSL, nomeadamente na análise da eficiência energética da programação idiomas (Secção 2), bibliotecas de estrutura de dados (Secção 3) e do código-fonte do software (Secção 4).

2 Linguagens de Programação Verdes

Uma questão interessante que surge quando se discute energia nas linguagens de programação é se uma linguagem mais rápida também é uma linguagem com eficiência energética, ou não. Comparando linguagens de software, entretanto, é uma tarefa extremamente complexa, pois a execução de um linguagem é influenciada pela qualidade de seu compilador, virtual máquina, coletor de lixo, etc.

No Laboratório de Software Verde, estudamos, avaliamos e comparamos os desempenho de (um total de) 27 dos softwares mais usados línguas. Usamos dois repositórios de problemas de computadores diferentes: textit Jogo de benchmark de linguagem de computador (CLBG)[3] e os repositórios de *Código Rosetta*[4] [1,2,3]. Ambos os repositórios definir um conjunto de tarefas do computador e fornecer implementações em uma grande grupo de linguagens de programação. Enquanto o CLBG foi adaptado para analisar desempenho do tempo de execução dos idiomas, *Código Rosetta* foi definido com mais propósitos de compreensão do programa.

Nós compilamos/executamos esses programas usando o estado da arte compiladores, máquinas virtuais, intérpretes e bibliotecas para cada língua. Em seguida, monitoramos o tempo de execução, o pico e o desempenho geral. consumo de memória e consumo de energia da CPU/DRAM/GPU. Nós produzimos um ranking energético das 27 línguas e também analisamos resultados de acordo com o tipo de execução dos idiomas (compilado, virtual máquina e interpretada) e paradigma de programação (imperativo, funcional, orientado a objeto, script) usado. Para cada um dos tipos de execução e paradigmas de programação, compilamos um software classificação da língua de acordo com cada objetivo con-

[3] http://benchmarksgame.alioth.debian.org/
[4] http://www.rosetta.org

siderado individualmente (por exemplo, tempo ou consumo de energia). Nossos primeiros experimentos mostram resultados esperados, como a linguagem C, sendo mais rápida e idioma mais verde, no entanto, também mostra idiomas mais lentos mais eficiente em termos energéticos do que outros [2,3].

3 Estruturas de dados Verdes

A Linguagem/paradigma de programação e seu poderosas otimizações, não é o único aspecto que influencia a energia consumo de um sistema de software. De fato, um programa também pode se tornar mais eficiente "justamente " otimizando as suas bibliotecas [4,5]. A maioria linguagens oferecem bibliotecas poderosas para manipular estruturas de dados. No GSL estudamos o desempenho energético de duas estruturas de dados avançadas amplamente utilizado nas linguagens de programação Java e Haskell.

Em Java, realizamos um estudo detalhado em termos de consumo de energia da biblioteca *Java Collections Framework* (JCF) [5]. Nós consideramos os três grupos diferentes de estruturas de dados usuais, Sets, Listse Mapse, para cada um desses grupos, estudamos o consumo de energia de cada uma de suas diferentes implementações e métodos [4]. Este JCF a conscientização energética não pode ser usada apenas para orientar os desenvolvedores de software escrevendo software Java mais ecológico, mas também na otimização de Java legado código. Desenvolvemos uma ferramenta de refatoração de estrutura de dados Java, denominada *jStanley*, que refatora o código-fonte Java quando é mais ecológico a coleção está disponível [6]. Também executamos uma inicial avaliação com 7 projetos Java publicamente disponíveis, onde pudemos melhorar o consumo de energia entre 2% e 17%.

Em Haskell, estudamos o consumo de energia de *Edison*[6], uma biblioteca totalmente madura e bem documentada de dados puramente funcionais estruturas [7]. Edison fornece diferentes estruturas de dados funcionais para implementar três tipos de abstrações: *Sequences* (listas, filas e estacas), *Coleções* (conjuntos e pilhas) e *Coleções Associativas* (mapas e relações finitas). Analisamos 16 implementações dessas estruturas de dados enquanto mede métricas de energia e tempo [5]. Nós investigou ainda mais o impacto no consumo de energia do uso de diferentes otimizações de compilação. Concluímos que o consumo de energia é diretamente proporcional ao tempo de execução e que a energia consumo de DRAM representando entre 15 e 31% do total consumo de energia. Por fim, também concluímos que otimizações podem têm um impacto positivo ou negativo no consumo de energia.

4 Código Fonte Verde

Não apenas as linguagens e as bibliotecas de estrutura de dados influenciam a energia consumo, algoritmos e práticas de programação também desempenham um papel fundamental na eficiência dos programas. Na GSL, adaptamos falhas conhecidas técnicas de localização para localizar estaticamente "vazamentos de energia " (vistos como ineficiência energética, portanto, falhas de energia) no código fonte das aplicações [8,9,10,11].

Nós definimos SPELL - *SPectrum-based Energy Leak Localization* para determinar áreas vermelhas (energia ineficiente) no software. Um primeiro estudo experimental mostra que programadores especializados, com acesso ao vazamentos de energia detectados pelo SPELL, foram capazes de otimizar melhor a energia consumo dos programas (entre 15% e 74%), do que especialistas com nenhuma informação ou a informação fornecida por programas padrão criador de perfil (tempo de execução). Também estudamos o comportamento energético de C/C++ programas [12].

O uso generalizado de dispositivos não conectados e o advento do Internet das Coisas, está mudando a maneira como os engenheiros de software desenvolvem seu software. O software precisa ser executado em uma variedade de dispositivos móveis e consumo de energia é uma das principais preocupações ao desenvolver Programas. Linhas de Produtos de Software (SPL) surgiram como um importante disciplina de engenharia de software que permite o desenvolvimento de software que compartilha um conjunto comum de textitfeatures. Na GSL, definimos técnicas de análise estática para raciocinar sobre o consumo de energia em SPLs com base na compilação condicional. Tais técnicas permitem que o software desenvolvedores para identificar (não) verde *produtos* e/ou *features* em um SPL [13].

O Android é um ecossistema amplamente utilizado para dispositivos e análise e otimização de energia de software é um ativo área de pesquisa. A equipe GSL desenvolveu vários técnicas [14,15] e ferramentas para analisar e otimizar o consumo de energia no código fonte do Android aplicações [16,17].

Atualmente, a maioria dos dados armazenados em nossos dispositivos móveis (arquivos, fotos, vídeos) também é armazenado na nuvem fornecida pelo ecossistema do sistema operacional do dispositivo. Esses sistemas em nuvem são dados centros que executam diariamente uma grande quantidade de processos de consulta de dados, monitorado e controlado por gerenciamento de banco de dados altamente sofisticado sistemas, responsáveis por estabelecer um processamento eficiente de consultas planeja apoiá-los. Os sistemas de banco de dados geralmente dependem de planos que otimizar o tempo de resposta. Nós projetamos e desenvolvemos uma alternativa método para definir planos de consumo de energia para banco de dados consultas [18,19]. Nossos primeiros resultados experimentais mostram que o uso de heurísticas de otimização permite ganhos significativos, tanto em termos de consumo de energia quanto do tempo gasto com a execução de consultas.

5 Conclusões

Este relatório técnico descreveu a pesquisa desenvolvida no Green Laboratório de Software, ou seja, um ranking verde das linguagens de programação e estruturas

[5] `docs.oracle.com/javase/7/docs/technotes/guides/collections/index.html`
[6] `hackage.haskell.org/package/EdisonAPI-1.3/docs/Data-Edison.html`

de dados, técnicas para detectar ineficiência energética em o código fonte de um sistema de software e uma execução de consulta com consciência de energia planejar sistemas de banco de dados.

Agradecimentos

Este artigo é parte da produção intelectual O2 do projeto Erasmus+ Key Action 2 project No. 2017-1-SK01-KA203-035402, the Strategic Partnership for Higher Education *"Focusing Education on Composability, Comprehensibility and Correctness of Working Software" (3COWS)*.

Aviso Legal

Referências

1. Couto, M., Pereira, R., Ribeiro, F., Rua, R., Saraiva, J.: Towards a green ranking for programming languages. In: Proceedings of the 21st Brazilian Symposium on Programming Languages. SBLP (2017) 7:1–7:8 (best paper award).
2. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J.: Energy efficiency across programming languages: How do energy, time, and memory relate? In: Proc. of the 10th ACM SIGPLAN Int. Conference on Software Language Engineering. SLE 2017, New York, NY, USA, ACM (2017) 256–267
3. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J.: Ranking programming languages by energy efficiency. Science of Computer Programming (2018) Submitted.
4. Pereira, R., Couto, M., Saraiva, J., Cunha, J., Fernandes, J.P.: The Influence of the Java Collection Framework on Overall Energy Consumption. In: 5th Int. Workshop on Green and Sustainable Software. GREENS '16, ACM (2016) 15–21
5. Melfe, G., Fonseca, A., Fernandes, J.P.: Helping developers write energy efficient haskell through a data-structure evaluation. In: Proceedings of the 6th International Workshop on Green and Sustainable Software. GREENS '18, New York, NY, USA, ACM (2018) 9–15
6. Pereira, R., Simão, P., Cunha, J., Saraiva, J.: jStanley: Placing a Green Thumb on Java Collections. In: 33rd ACM/IEEE International Conference on Automated Software Engineering. ASE 2018, New York, NY, USA, ACM (2018) 856–859
7. Lima, L.G., Melfe, G., Soares-Neto, F., Lieuthier, P., Fernandes, J.P., Castor, F.: Haskell in Green Land: Analyzing the Energy Behavior of a Purely Functional Language. In: Proc. of the 23rd IEEE Int. Conf. on Software Analysis, Evolution, and Reengineering (SANER'2016), IEEE (2016) 517–528
8. Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J.P., Saraiva, J.: Helping programmers improve the energy efficiency of source code. In: Proc. of the 39th Int. Conf. on Soft. Eng. Companion, ACM (2017)
9. Pereira, R.: Locating energy hotspots in source code. In: Proceedings of the 39th International Conference on Software Engineering Companion. ICSE-C '17, Piscataway, NJ, USA, IEEE Press (2017) 88–90 (ACM SRC silver award).
10. Pereira, R.: Energyware Engineering: Techniques and Tools for Green Software Development. PhD thesis, Depart. de Informática, Universidade do Minho (2018)
11. Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J.P., Saraiva, J.: Spelling out energy leaks: Aiding developers locate energy inefficient code. (2018) (submitted).
12. Santos, M., Saraiva, J., Porkoláb, Z., Krupp, D.: Energy consumption measurement of c/c++ programs using clang tooling. SQAMIA'17 - CEUR Workshop Proceedings **1938** (2017)
13. Couto, M., Borba, P., Cunha, J., Fernandes, J.P., Pereira, R., Saraiva, J.: Products go green: Worst-case energy consumption in software product lines. In: Proceedings of the 21st International Systems and Software Product Line Conference - Volume A. SPLC '17, ACM (2017) 84–93
14. Couto, M., Carção, T., Cunha, J., Fernandes, J.P., Saraiva, J.: Detecting anomalous energy consumption in android applications. In Quintão Pereira, F.M., ed.: Programming Languages: 18th Brazilian Symposium, SBLP 2014, Maceio, Brazil, October 2-3, 2014. Proceedings. (2014) 77–91
15. Cruz, L., Abreu, R.: Performance-based guidelines for energy efficient mobile applications. In: 4th International Conference on Mobile Software Engineering and Systems. MOBILESoft '17, Piscataway, NJ, USA, IEEE Press (2017) 46–57
16. Couto, M., Cunha, J., Fernandes, J.P., Pereira, R., Saraiva, J.: Greendroid: A tool for analysing power consumption in the android ecosystem. In: 2015 IEEE 13th International Scientific Conference on Informatics. (Nov 2015) 73–78
17. Cruz, L., Abreu, R., Rouvignac, J.N.: Leafactor: Improving energy efficiency of android apps via automatic refactoring. In: IEEE/ACM International Conference on Mobile Software Engineering and Systems, MobileSoft 2017. (2017)
18. Gonçalves, R., Saraiva, J., Belo, O.: Defining energy consumption plans for data querying processes. In: 2014 IEEE International Conference on Big Data and Cloud Computing (BdCloud)(BDCLOUD). Volume 00. (Dec. 2015) 641–647
19. Belo, O., Gonçalves, R., Saraiva, J.: Establishing energy consumption plans for green star-queries in data warehousing systems. In: 2015 IEEE International Conference on Data Science and Data Intensive Systems. (Dec 2015) 226–231

Ensino de Programação Oreintada a Tarefas

Pieter Koopman and Rinus Plasmeijer

Institute for Computing and Information Sciences
Radboud University Nijmegen, The Netherlands
{pieter,rinus}@cs.ru.nl

Resumo Na escola de inverno Composibilidade, Compreensibilidade, Correção, haverá dois tutoriais sobre Programação Orientada a Tarefas e sistemas concretos com base nesse paradigma. O sistema iTask oferece uma interface baseada na Web para que os humanos vejam suas tarefas e compartilhem seu progresso com essas tarefas. O sistema mTask aplica os mesmos conceitos para especificar as tarefas executadas pelos microprocessadores. Nesta contribuição, justificamos as decisões tomadas na formação de professores em Amsterdã sobre o que e como esses tópicos serão apresentados na escola de inverno. Devido ao tempo limitado e fundo diversificado da audiência, focaremos no uso prático do paradigma. Mal há tempo para enfrentar os desafios e a beleza da construção desses sistemas.

Keywords: Programação Orientada a Tarefas · Languagens de Domínio Específico · Microprocessador · Generação de Código · Simulação · Programação Genérica · Monadas

1 Introduction

A Programação Orientada a Tarefas, TOP, é um estilo de programação centrado no conceito de tarefas executadas por seres humanos e máquinas. Essas tarefas são especificadas por funções comuns em uma linguagem de programação funcional. Em todos os nossos exemplos, usaremos Clean [6]. A semântica das tarefas é bem diferente da avaliação simples de funções. Uma tarefa é avaliada repetidamente até produzir um valor estável ou seu resultado não é mais necessário. Os resultados intermediários da tarefa podem ser observados por outras tarefas. As tarefas podem ser compostas por combinadores de tarefas.

Na escola de inverno Composability, Comprehensibility, Correctness, em Kosice, haverá duas sessões no TOP. Elas têm o título *Porque é que a programação orientada a tarefas é importante* e *Programação Funcional de Dispositivos*. Ambas as sessões consistirão em uma palestra e trabalho prático dos participantes. Neste artigo, motivamos as decisões sobre o conteúdo e a organização dessas sessões após as discussões na formação dos professores.

2 Audiência

A escola de inverno de composição, compreensibilidade e correção é para alunos de graduação, mestrado e doutorado, bem como para professores. Discussões na formação de professores revelaram que a experiência em programação funcional é diversa, tanto na quantidade de experiência quanto na linguagem. Os idiomas usados variam de idiomas puros e preguiçosos como Haskell e Clean a Erlang, Scheme e Scala.

Isso implica que não podemos assumir uma sólida experiência de programação funcional comum. Apenas uma parte do público estará familiarizada com conceitos como digitação forte, funções de ordem superior, classes de construtor de tipo, monadas e genéricos. Embora esses tópicos sejam os elementos básicos do TOP, não podemos assumir que eles sejam conhecidos por todos os participantes.

3 Programação Orientada a Tarefas

A Programação Orientada a Tarefas é baseada em um pequeno número de domínios tarefas primitivas específicas. Essas tarefas primitivas geralmente interagem com o ambiente, por exemplo, seres humanos executando parte das tarefas ou hardware interagindo com o mundo físico. Os combinadores de tarefas são usado para compor tarefas a partir de tarefas menores.

As tarefas podem se comunicar por meio de seus resultados, bem como por meio de dados compartilhados Fontes, SDSs. Esse SDS contém dados digitados que podem ser acessados via primitivas como `get` e `set`. Essas primitivas agem no estado da tarefa para garantir transparência referencial.

Para reutilizar tipos de dados e cálculos de uma linguagem existente, um sistema TOP geralmente é construído como uma linguagem específica de domínio, DSL, incorporada a uma linguagem de programação (funcional) existente.

4 O Sistema iTask

O sistema iTask foi a primeira implementação do TOP [5]. É um DSL incorporado na linguagem de programação funcional Clean. O sistema iTask facilita a interação com o trabalhador humano, gerando tipos de páginas da web. Essas páginas são exibidas em um dos navegadores existentes. A página fornece informações das tarefas atuais para um usuário. O usuário pode interagir com o sistema iTask preenchendo formulários e pressionando botões.

4.1 Used Techniques

O sistema iTask transmite um estado muito semelhante a uma mônada de estado. Os operadores para `return`, bind (`>>=`) e sequência (`>>|`) são muito semelhantes às versões monádicas conhecidas [4,7]. Isso requer funções de ordem superior e operadores de infixo definidos pelo usuário. Para reutilizar o símbolo do operador para diferentes Mônadas, elas são definidas como classes de construtor de tipo.

Os combinadores de tarefas são todas funções de ordem superior, geralmente operadores de infixos definidos pelo usuário, manipulando os resultados da tarefa

e o estado global da tarefa. Específico para TOP é que as tarefas produzem resultados intermediários que podem ser observados enquanto as tarefas são repetidas repetidamente até produzir um resultado estável ou que o resultado não seja mais usado. Isso requer funções de ordem superior, preguiça e coleta automática de lixo.

O sistema iTask gera um servidor da web usado pelos trabalhadores humanos para encontrar sua tarefa. Como todos os servidores da web, isso requer serialização e desserialização do estado para armazenar e recuperar o estado atual. Ao preencher formulários da Web para tipos de dados algébricos arbitrários, os usuários indicam seu progresso nas tarefas. Todos esses recursos são implementados usando programação genérica.

Para implementar as tarefas que monitoram o valor de um SDS de maneira eficiente, existe um sistema oculto de publicação/assinatura para cada SDS que ativa tarefas usando esse SDS quando seu valor é atualizado.

Além dessas propriedades, o sistema iTask usa muitas técnicas adicionais. Por exemplo, a execução de partes de tarefas em um intérprete em execução no navegador para garantir uma resposta rápida do sistema para tarefas altamente interativas, como um desenho.

4.2 Ensino na Escola da Inverno

Qualquer ensino de programação requer experiência prática em programação, usando as técnicas educadas para dominá-las. Isso vale também para o TOP. Como consequência, dividimos as quatro horas disponíveis para *Por que a programação orientada a tarefas é importante?* em duas partes de tamanho quase idêntico.

Na primeira parte, descreveremos o conceito de TOP usando o sistema iTask. Dado o histórico da maioria dos estudantes, temos que pular quase todos os detalhes sobre a implementação do sistema e precisamos nos concentrar no uso da biblioteca. Esta biblioteca é na verdade uma DSL incorporada superficial para o TOP. Na palestra, usamos um conjunto de primitivas sem gastar muito tempo para explicar sua arquitetura e implementação.

Para o trabalho prático, dividiremos o projeto de exemplo básico existente em um conjunto de pequenos projetos TOP independentes. As atribuições consistirão em pequenas variações desses projetos para experimentar o sabor da programação orientada a tarefas.

Os programadores funcionais mais experientes podem pular a maioria dos exercícios básicos e pular diretamente para tarefas mais avançadas. Dessa forma, poderemos nos adaptar às habilidades individuais de cada participante.

5 O Sistema mTask

Microprocessadores são sistemas de computador com recursos de computação muito limitados. Eles geralmente têm uma taxa de clock bastante baixa e restrições severas de memória, como alguns KB de memória para armazenar os dados de um programa em execução. Esses processadores baratos são a força motriz de muitos elementos na Internet das Coisas, IoT. Em tais sistemas de microprocessador, normalmente é necessário monitorar várias portas de entrada, bem como controlar algumas saídas com base nessas observações. Devido às restrições de hardware, normalmente não há sistema operacional oferecendo suporte.

O paradigma TOP fornece threads leves que são muito adequados para monitorar e coordenar o progresso dessas tarefas simples bem definidas. Essas tarefas podem ser executadas em sua própria velocidade, enquanto combinadores e fontes de dados compartilhadas são usadas para coordená-las. A execução do sistema iTask nos dispositivos IoT nos permitiria construir programas que são parcialmente executados em um servidor Web e nos dispositivos IoT. As limitações dos microprocessadores tornam impossível executar um programa iTask completo em dispositivos de IoT.

Para aproximar a solução ideal, desenvolvemos o sistema mTask [3,2]. Trata-se de uma DSL rasa e integrada com várias visualizações que pode ser usada como parte do sistema iTask. Ele suporta o paradigma TOP, incluindo os mesmos resultados de tarefas que o sistema iTask, combinadores de tarefas e fontes de dados compartilhadas. Por construção, esse DSL não possui funções de ordem superior nem tipos de dados recursivos. Devido às restrições impostas neste DSL, os programas mTask podem ser compilados no código que é executado nos microprocessadores. Apesar dessas restrições, o DSL é muito adequado para especificar as tarefas a serem executadas nos dispositivos IoT de maneira fácil e concisa.

5.1 Tecnicas Utilizadas

O sistema mTask é uma DSL superficialmente integrada, com múltiplas visualizações, com base nas classes de construtor de tipo [1]. Cada instância dessas classes define uma interpretação, chamada *view*, de um programa construído a partir dessas primitivas. As visualizações típicas implementam impressão bonita, geração de código para microprocessadores e simulação dos programas mTask como um programa iTask.

O DSL é extensível por construção para reutilizar as bibliotecas existentes para periféricos como sensores de temperatura, monitores e servomotores. É simples adicionar uma biblioteca como uma linguagem primitiva ao sistema mTask, introduzindo uma nova classe de construtor de tipo e as instâncias necessárias.

Para tornar a implementação mTask portátil para muitos microprocessadores diferentes e facilitar a reutilização de bibliotecas C++ existentes, a exibição de geração de código produz código C++ para a plataforma Arduino, em vez de código de máquina nativo para algum processador específico. O compilador avr-gcc dentro da plataforma Arduino pode traduzir o código C++ gerado e as bibliotecas usadas no código nativo para muitos microprocessadores diferentes.

5.2 Ensino na Escola da Inverno

Ser capaz de executar programas TOP de alto nível em um minúsculo microprocessador interagindo com periféricos é atraente para um tutorial na escola de inverno. No entanto, a execução de um programa mTask em um microprocessador real requer grande parte dos alunos; eles devem compor um programa mTask dentro do sistema iTask, executar o programa iTask para obter o código C++, alimentar esse código C++ no Arduino IDE, conectar o Arduino IDE ao microprocessador e selecionar as opções corretas, carregar o programa compilado para o microprocessador e, finalmente, execute-o. Todas essas etapas são fáceis, mas todo o processo produz apenas um resultado quando todas as etapas são executadas corretamente. Como o programa gerado será executado em um microprocessador sem sistema operacional e a depuração de periféricos de entrada/saída muito limitada, esse programa é um desafio. Após ampla discussão, essa sequência de etapas foi considerada ambiciosa demais para o tempo e a audiência determinados.

Felizmente, a visualização do simulador do sistema mTask oferece uma alternativa muito mais fácil de usar. Essa visualização transforma o programa mTask em um programa iTask comum. O simulador oferece uma execução passo a passo do programa mTask. Ele exibe um rastreio da última etapa da última tarefa executada e o estado de todos os periféricos e fontes de dados compartilhadas. O relógio, o valor dos SDSs e o estado dos periféricos podem ser alterados interativamente para controlar a execução e investigar vários cenários. Isso torna o trabalho prático deste tutorial um sucessor direto do trabalho prático do tutorial anterior do iTask.

Foi decidido agendar esses tutoriais em um dia com a palestra iTask e o trabalho prático associado pela manhã e o tutorial mTask à tarde. Dessa forma, a sessão mTask pode desenvolver diretamente o conhecimento e as habilidades adquiridas na sessão iTask. O entendimento do TOP que foi declarado pela manhã será aprofundado à tarde.

6 Conclusão

Para os dois tutoriais no TOP, há muitos tópicos mais interessantes que podem ser abordados no tempo determinado para o público da escola de inverno. Nas sessões, focaremos no entendimento e uso do paradigma TOP por exemplos relativamente simples. Exemplos levemente avançados serão usados para ilustrar os recursos dessa abordagem. No trabalho prático, focaremos nos exercícios ilustrativos que são principalmente variantes de exemplos usados no tutorial. Para os participantes avançados, haverá algumas tarefas desafiadoras, bem como a oportunidade de discutir os aspectos dos sistemas em profundidade.

Agradecimentos

Este artigo é parte da produção intelectual O2 do projeto Erasmus+ Key Action 2 project No. 2017-1-SK01-KA203-035402, the Strategic Partnership for Higher Education *"Focusing Education on Composability, Comprehensibility and Correctness of Working Software" (3COWS).*

Aviso Legal

Referências

1. Carette, J., Kiselyov, O., Shan, C.c.: Finally tagless, partially evaluated: Tagless staged interpreters for simpler typed languages. J. Funct. Program. **19**(5), 509–543 (Sep 2009). https://doi.org/10.1017/S0956796809007205, http://dx.doi.org/10.1017/S0956796809007205
2. Koopman, P., Lubbers, M., Plasmeijer, R.: A task-based dsl for microcomputers. In: Proceedings of the Real World Domain Specific Languages Workshop 2018. pp. 4:1–4:11. RWDSL2018, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3183895.3183902, http://doi.acm.org/10.1145/3183895.3183902
3. Koopman, P., Plasmeijer, R.: A shallow embedded type safe extendable DSL for the Arduino. In: Revised Selected Papers of the 16th International Symposium on Trends in Functional Programming - Volume 9547. pp. 104–123. TFP 2015, Springer-Verlag New York, Inc., New York, NY, USA (2016). https://doi.org/10.1007/978-3-319-39110-6_6, http://dx.doi.org/10.1007/978-3-319-39110-6_6
4. Peyton Jones, S.L., Wadler, P.: Imperative functional programming. In: Proceedings of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 71–84. POPL '93, ACM, New York, NY, USA (1993). https://doi.org/10.1145/158511.158524, http://doi.acm.org/10.1145/158511.158524
5. Plasmeijer, R., Achten, P., Koopman, P.: iTasks: executable specifications of interactive work flow systems for the web. In: Hinze, R., Ramsey, N. (eds.) Proceedings of the ICFP'07. pp. 141–152. ACM, Freiburg, Germany (2007)
6. Plasmeijer, R., van Eekelen, M., van Groningen, J.: Clean language report (version 2.2) (2011), http://clean.cs.ru.nl/Documentation
7. Wadler, P.: Comprehending monads. In: Proceedings of the 1990 ACM Conference on LISP and Functional Programming. pp. 61–78. LFP '90, ACM, New York, NY, USA (1990). https://doi.org/10.1145/91556.91592, http://doi.acm.org/10.1145/91556.91592

Uma Abordagem Interactiva no Ensino de Redes de Petri

Štefan Korečko

Department of Computers and Informatics,
Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 041 20 Košice, Slovakia
stefan.korecko@tuke.sk

Resumo Os métodos formais pertencem às técnicas que, quando usadas adequadamente, podem contribuir significativamente para a correção de um sistema de software ou hardware em desenvolvimento. Um dos métodos adequados para sistemas com comportamento simultâneo ou não determinístico é a linguagem de modelagem de redes de Petri coloridas. Neste artigo, é descrita uma atividade de ensino que visa explicar os princípios básicos da linguagem e alguns de seus recursos funcionais relacionados à programação. A duração da atividade é de duas horas e meia e envolveu uma construção de modelo interativo com participação ativa do público.

1 Introduction

Considerando a crescente dependência da sociedade humana contemporânea em sistemas de computadores, sua correção deve ser de extrema importância. E uma das abordagens que podem contribuir significativamente para a correção é a utilização de métodos formais durante o desenvolvimento de software e hardware. Um método formal é uma técnica matematicamente baseada, que fornece uma linguagem formal com sintaxe e semântica definidas sem ambiguidade e um aparelho que permite executar tarefas de verificação, desenvolvimento e simulação com especificações do sistema, escritas na linguagem. Um dos membros significativos da família de métodos formais é a linguagem de modelagem Redes de Petri Coloridas (CPN). O CPN [3,2] combina o formalismo das redes de Petri cite petri98 com uma linguagem funcional para lidar com procedimentos de manipulação e decisão de dados. A linguagem funcional é chamada CPN ML e é uma versão ligeiramente modificada do Standard ML [4]. A linguagem CPN e as tarefas correspondentes de especificação, verificação e simulação são suportadas pelas ferramentas de software CPN[1].

Por mais de uma década, os CPN fazem parte de cursos de graduação relacionados a métodos formais, modelagem e simulação na instituição de origem do autor. Um dos métodos, aplicado pelo autor ao explicar os conceitos de CPN, é uma abordagem interativa com participação ativa do público. Aqui, o público

[1] http://cpntools.org/

escolhe o domínio e o processo para o qual um modelo de CPN será projetado e ajuda a criar as partes selecionadas. A experiência de uma implementação específica dessa abordagem em uma atividade de treinamento para professores universitários é descrita no restante deste artigo.

2 Actividade de Treino na Criação Interctiva de Modelos CPN

A atividade de treinamento foi organizada para cerca de 10 participantes, que eram professores universitários com certas linguagens funcionais fundo. Os participantes tinham limitado a nenhum conhecimento prévio de CPN. A duração total da atividade foi de cerca de 2,5 horas, excluindo pausas, e foi dividido em três fases.

A primeira fase levou cerca de 30 minutos e explicou os princípios básicos da CPN. Nomeadamente, esse CPN tem uma forma gráfica, um gráfico bipartido com dois tipos de vértices: lugares, desenhados como elipses e transições, desenhados como retângulos. Os locais contêm tokens, que representam um estado da rede e as transições podem ser entendidas como eventos, que alteram o estado consumindo os tokens existentes e criando novos.

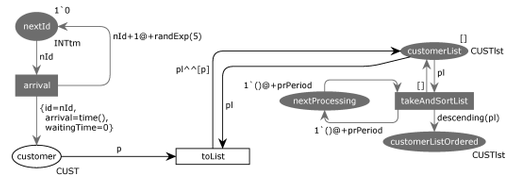


Figura 1. O modelo CPN inicial dado aos participantes na actividade

A segunda e terceira fase foram dedicadas à criação de um modelo de CPN. Como um dos objetivos da atividade era mostrar como alguns conceitos mais avançados do Standard ML, como estruturas e functores, podem ser usados nos modelos de CPN, os participantes receberam um modelo inicial de CPN, que já usava os conceitos antes do segundo fase iniciada. O modelo inicial é mostrado na Fig. 1. A parte que consiste nos nós `nextId`, `arrival` e `cliente` representa uma chegada de clientes, que chegam um a um para serem atendidos. A veiculação em si não é apresentada no modelo inicial. Em vez disso, existe a transição `toList`, que recebe um token de `customer` e adiciona seu valor a uma lista, mantida no local `customerList`. A transição `takeAndSortList` é acionado em intervalos regulares, definidos pelo valor `prPeriod`. Cada disparo de `takeAndSortList` es-

vazia a lista em `customerList`, classifica seu conteúdo e armazena a versão ordenada em `customerListOrdered`. O local `nextProcessing` é auxiliar e garante que `takeAndSortList` é acionado apenas em intervalos regulares. A classificação é fornecida por uma função chamada verb—descending—, que implementa o algoritmo Quicksort. A função utiliza estruturas e functores ML padrão.

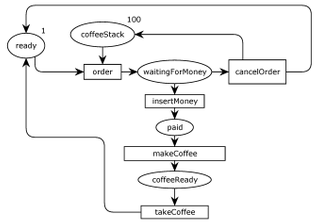


Figura 2. Modelo CPN da parte de serviço, creado interactovamente na segunda fase

Para a parte do serviço, os participantes da atividade decidiram modelar uma máquina de venda automática de café. Durante a segunda fase, eles participaram da criação de um modelo de CPN que captura a operação básica da máquina. O modelo é mostrado na Fig. 2. Em seu estado inicial, a máquina está pronta para atender um cliente (um token no local `ready`) e é preenchida com 100 doses de café (100 tokens em `coffeeStack`). A veiculação começa com um cliente solicitando um café acionando a transição `order`. Em seguida, a máquina aguarda a próxima etapa do cliente (um token em `waitingForMoney`). O cliente pode inserir dinheiro (disparando `insertMoney`) ou cancelar o pedido (disparando `cancelOrder`). O cancelamento retorna a máquina ao estado “ pronto ”. Se o dinheiro for inserido, a máquina prepara o café (disparando `makeCoffee`). Por fim, disparando `takeCoffee`, o cliente pega o café preparado e a máquina retorna ao estado “ pronto ”.

Após a segunda fase, houve um intervalo de cerca de 70 minutos. Durante o intervalo, o palestrante conectou o modelo da fase 2 às partes do modelo inicial e adicionou vértices e arcos descrevendo o comportamento do cliente. Ele também corrigiu algumas inconsistências no modelo, apontadas por um dos participantes. O modelo final de CPN resultante pode ser visto na Fig. 3. Os vértices retirados do modelo inicial (Fig. 1) sem nenhuma alteração são renderizados em cinza. O local `client` é substituído por `customerQueue`, que contém um token com uma lista de valores, representando uma fila de clientes aguardando a máquina. Em vez da transição `toList` existe uma peça de serviço, criada a partir do resultado da segunda fase (Fig. 2). A parte que serve do modelo final difere da Fig. 2 em três aspectos principais:

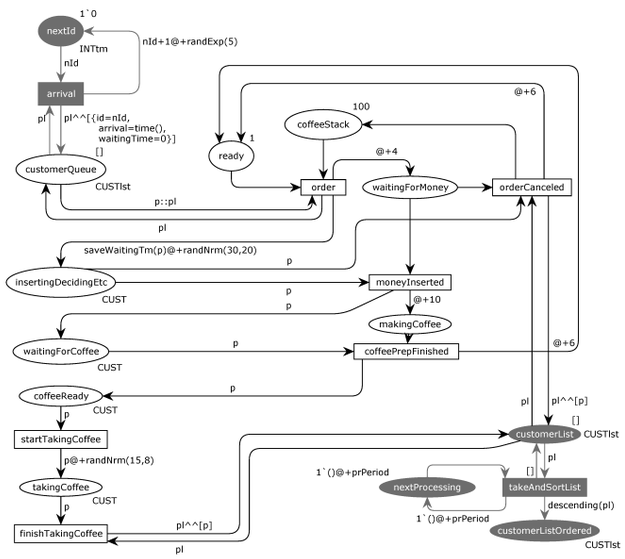


Figura 3. A forma final do modelo CPN da máquina de café

– Os tokens carregam informações sobre o cliente que está sendo atendido e as expressões de arco definem as durações das ações correspondentes.
– As inconsistências em relação ao papel dos lugares e transições são corrigidas. Agora, todas as transições representam eventos instantâneos. Por exemplo, a transição `makeCoffee` da Fig. 2 é substituído pelo lugar `makingCoffee` e a transição `takeCoffee` é substituído pelos vértices `startTakingCoffee`, `takingCoffee` e `finishTakingCoffee`.
– Ações e estados dos clientes e da máquina são modelados separadamente. Os vértices `customerQueue`, `insertingDecidingEtc`, `waitingForCoffee`, `coffeeReady`, `startTakingCoffee`, `takingCoffee` e `finishTakingCoffee` pertencem aos clientes, enquanto o restante da peça de serviço representa a máquina ou ambas as partes.

A terceira fase da atividade de treinamento foi dedicada à explicação do modelo final e a uma discussão sobre o lugar desses modelos no desenvolvimento de sistemas de computador corretos. Demorou cerca de 30 minutos.

3 Conclusion

A atividade de treinamento interativo, apresentada aqui, é adequada para cursos intensivos e curtos, que geralmente ocorrem durante as escolas de verão ou outros eventos de ensino semelhantes. A execução de teste descrita da atividade revelou que a doação do tempo original, que era de 2 horas, não era suficiente. Portanto, a terceira fase foi necessária, onde o palestrante apresentou o modelo final. Considerando o tempo necessário para a construção do modelo final pelo professor, serão necessárias outras pelo menos duas horas para executar todo o processo de criação do modelo de maneira interativa com o auditivo. Todos os modelos de CPN apresentados ou mencionados aqui podem ser obtidos mediante solicitação do autor.

Agradecimentos

Este artigo é parte da produção intelectual O2 do projeto Erasmus+ Key Action 2 project No. 2017-1-SK01-KA203-035402, the Strategic Partnership for Higher Education *"Focusing Education on Composability, Comprehensibility and Correctness of Working Software" (3COWS).*

Aviso Legal

As informações e visões definidas neste artigo são de responsabilidade do(s) autor(es) e não reflectem necessariamente a opinião oficial da União Europeia. Nem instituições e órgãos da União Europeia, nem qualquer pessoa que atue nome pode ser responsabilizado pelo uso que possa ser feito do informações nele contidas.

Referências

1. Desel, J., Reisig, W.: Place/transition petri nets. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, Lecture Notes in Computer Science, vol. 1491, pp. 122–173. Springer Berlin Heidelberg. DOI: 10.1007/3-540-65306-6 (1998)
2. Jensen, K.: An introduction to the theoretical aspects of coloured petri nets. In: A Decade of Concurrency, Reflections and Perspectives, REX School/Symposium. pp. 230–272. Springer-Verlag, London, UK. DOI: 10.1007/3-540-58043-3_21 (1994)
3. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer. DOI: 10.1007/b95112 (2009)
4. Milner, R., Tofte, M., Macqueen, D.: The Definition of Standard ML. MIT Press, Cambridge, MA, USA, DOI: 10.7551/mitpress/2319.001.0001 (1997), `http://sml-family.org/sml97-defn.pdf`

CodeCompass: Uma Framework Extensível para Compreenção de Código

Tibor Brunner

Faculty of informatics, Eötvös Loránd University,
Budapest, Hungary
bruntib@caesar.elte.hu

Resumo CodeCompass é uma ferramenta de código aberto para ajudar a entender grandes legados sistemas de software. Com base na infraestrutura do compilador LLVM/Clang, O CodeCompass fornece informações exatas sobre elementos complexos da linguagem C/C++. A ampla variedade de visualizações interativas inclui classe e diagramas de chamada de função; diagramas de arquitetura, componentes e interfaces e "aponta par " diagramas e muitos outros. O CodeCompass também utiliza build informações para explorar a arquitetura do sistema, bem como o controle de versão informações quando disponíveis. Os resultados da análise estática baseada em clang também são integrado. Embora a ferramenta se concentre principalmente em C e C++, ela também suporta Linguagens Java e Python. Tendo uma arquitetura extensível baseada na Web, plugável, o CodeCompass A estrutura pode ser uma plataforma aberta para maior compreensão do código, estática esforços de análise e métricas de software.

1 Introduction

A correção de bugs ou o desenvolvimento de novos recursos requer uma compreensão confiante de todos os detalhes e consequências das mudanças planejadas. Ferramentas de compreensão de código pode ajudar a revelar as intenções originais e detalhes da implementação, construindo um modelo a partir do código fonte e outras informações disponíveis. Embora várias dessas ferramentas estejam disponíveis como propriedade proprietária ou software livre, seu conjunto de recursos é limitado.

O CodeCompass foi desenvolvido para eliminar essas restrições. O CodeCompass O projeto é um esforço conjunto de código aberto da Ericsson Ltd. e da Eötvös Loránd University, Budapeste, para ajudar a entender grandes sistemas de software. Para fornecer informações exatas sobre elementos complexos da linguagem C/C++, como sobrecarga, herança, uso de variáveis e tipos, possíveis usos de ponteiros de função e funções virtuais - recursos que vários ferramentas suportam apenas parcialmente - o CodeCompass é baseado em um compilador real, o Infraestrutura LLVM/Clang. Assim, elimina as fraquezas do habitual Ferramentas de compreensão "leves", como o OpenGrok.

CodeCompass, no entanto, não está restrito ao código fonte. Ele usa a compilação informações do sistema para revelar conexões arquiteturais. Também emprega as informações de controle de versão, se disponíveis, para que se possa identificar conexões entre diferentes arquivos de origem "acidentalmente" modificados no mesmo commit. Para ajudar na percepção rápida e precisa, o CodeCompass usa texto e gráficos. representação do sistema de software a ser compreendido. Vários (interativos) diagramas são acessíveis a partir dos gráficos habituais de chamadas de funções diagramas arquitetônicos. Para fornecer acesso fácil aos usuários, o CodeCompass possui um arquitetura. O cliente pode ser um navegador da Web padrão, um plug-in de editor ou qualquer aplicativo de terceiros. A comunicação é baseada em uma API REST e dimensiona bem para solicitações de clientes paralelas.

Neste artigo, compararemos o CodeCompass com as ferramentas de compreensão existentes e descreva seu conjunto de recursos. Na secção 2, apresentamos uma visão geral dos principais arquétipos de ferramentas existentes para compreensão de código. Nós introduzimos o arquitetura extensível do CodeCompass em 3. As principais características do a ferramenta é discutida na Secção 4. Resumimos o artigo na secção 5.

2 Trabalho Relacionado

No mercado de software, existem várias ferramentas que visam algum tipo de fonte compreensão de código. Alguns deles usam análise estática, outros examinam também a comportamento dinâmico do programa analisado. Essas ferramentas podem ser divididas em arquétipos diferentes baseados em suas arquiteturas e em seus princípios principais. Por um lado, as ferramentas estão tendo arquitetura servidor-cliente. Geralmente estes As ferramentas analisam o projeto e armazenam todas as informações necessárias em um banco de dados. o clientes (geralmente baseados na Web) são atendidos no banco de dados. Essas ferramentas podem ser integrado ao fluxo de trabalho à medida que o IC noturno é executado. Dessa forma, os desenvolvedores podem sempre navegue e analise toda a base de código grande e antiga. Também existem aplicativos pesados para o cliente em que parte menor da base de código é analisada. este é o caso de uso para editores IDE em que a modificação frequente da fonte requer atualização rápida do banco de dados sobre os resultados analisados. Nesta secção apresentamos algumas ferramentas utilizadas no ambiente industrial de cada categoria.

Woboq [3] é um navegador de código baseado na Web para C e C++. este A ferramenta possui diversos recursos que visam a navegação rápida de um projeto de software. O usuário pode encontrar rapidamente os arquivos e entidades nomeadas por um campo de pesquisa que fornece a conclusão do código para facilitar a usabilidade. A navegação na base de código é ativado por meio de uma página da Web que consiste em arquivos HTML estáticos. Esses arquivos são gerado durante um processo de análise. A vantagem dessa abordagem é que

o O cliente da Web será rápido, pois não é necessário computação on-the-fly no servidor lado enquanto navega.

Passar o mouse sobre uma função específica, classe, variável, macro etc. pode mostre as propriedades desse elemento. Por exemplo, no caso de funções, pode-se veja sua assinatura, local de definição e local de uso. Para as aulas um pode verificar o tamanho de seus objetos, o layout da classe e o deslocamento de seus membros e o diagrama de herança. Para variáveis, pode-se inspecionar seu tipo e locais onde são escritos ou lidos.

Em macros C e C++, formam uma sub-linguagem que é avaliada em uma pré-compilação degrau. Essa avaliação é uma substituição textual de tokens de macro, o que significa que a fase de compilação funciona com outro código que não o original. No Woboq, o valor final das expansões de macro também pode ser inspecionado.

Um recurso muito útil da ferramenta é o destaque semântico. Por esse recurso os diferentes elementos linguísticos podem ser facilmente distinguidos: a formatação de variáveis locais, globais ou membros, funções virtuais, tipos, typedefs, classes, macros etc. são todos diferentes.

O Woboq pode fornecer os recursos mencionados acima porque as informações necessárias são coletados em uma fase real de compilação. O projeto examinado primeiro deve ser compilado e analisado pelo Woboq. A análise é feita pela infraestrutura LLVM/Clang que disponibiliza toda a árvore de sintaxe abstrata. Desta forma, todas as peças de informação semântica pode ser extraída com a mesma semântica do programa final é ter. Isso também oferece uma desvantagem da ferramenta, ou seja, o Woboq pode apenas ser usado para navegar em projetos C e C ++.

OpenGrok [4] é uma pesquisa rápida de código-fonte e mecanismo de referência. Ao contrário do Woboq, essa ferramenta não possui linguagem profunda análise, portanto, não é capaz de fornecer informações semânticas sobre o entidades particulares. Em vez disso, ele usa *Ctags* [5] para analisar o código-fonte apenas textualmente e para determinar o tipo dos elementos específicos. A análise sintática simples permite distinguir funções, variáveis ou nomes de classe etc. A pesquisa entre eles é altamente otimizada e, portanto, muito rápido, mesmo em grandes bases de código. A pesquisa pode ser realizada via composto expressões (por exemplo, `defs: target`), contendo até caracteres curinga, além disso, os resultados podem ser restritos a subdiretórios. Além do texto procurar, existe a oportunidade de encontrar símbolos ou definições separadamente. A falta análise semântica permite que o Ctags suporte várias (41) programações línguas. Também uma vantagem dessa abordagem é que é possível atualizar de forma incremental o banco de dados de índice. O OpenGrok também oferece a oportunidade de coletar informações de sistemas de controle de versão como Mercurial, SVN, CSV etc.

Understand [6] não é apenas uma ferramenta de navegação de código, mas uma um IDE completo. Sua grande vantagem é que o código fonte pode ser editado e as alterações da análise podem ser vistas imediatamente.

Além das funções de navegação de código já mencionadas nas ferramentas anteriores, O Understand fornece muitas métricas e relatórios. Algumas dessas são as linhas de código (total/média/máxima globalmente ou por classe), número de classes acopladas/base/derivadas, falta de coesão [2], Complexidade de McCabe [1] e muitos outros. *Treemap* é um método de representação comum para todas as métricas. É uma vista retangular aninhada onde o aninhamento representa a hierarquia dos elementos e a cor e tamanho dimensões representam a métrica escolhida pelo usuário.

Para grandes bases de código, a inspeção da arquitetura é necessária. Entender pode mostrar diagramas de dependência com base em várias relações, como hierarquia de chamadas de função, herança de classe, dependência de arquivo, arquivo inclusão/importação. Os usuários também podem criar seu tipo de diagrama personalizado via API fornecida pela ferramenta.

Na programação, os conceitos principais são comuns entre as linguagens, mas existem alguns conceitos que são interpretados de maneira diferente em uma linguagem específica. Entenda pode lidar com idiomas $\sim$ 15 e pode fornecer idiomas específicos informações sobre o código, por exemplo análise de ponteiro de função em C/C++ ou pacote diagramas de hierarquia no Ada.

Entenda cria um banco de dados a partir da base de código. Toda a informação pode ser coletados por meio de uma API programável. Dessa forma, o usuário pode consultar todas as informações necessárias informações que não estão incluídas na interface do usuário.

CodeSurfer [7] é semelhante a Understand no sentido que também é um aplicativo de análise estática de cliente espesso. Seu alvo é entender projetos de código de máquina C/C++ ou x86. O CodeSurfer realiza uma profunda análise de linguagem que fornece informações detalhadas sobre o software comportamento. Por exemplo, ele implementa a análise de ponteiro para verificar quais ponteiros pode apontar para uma dada variável, lista as instruções que dependem de uma declaração por análise de impacto e usa a análise de fluxo de dados para identificar onde a variável recebeu seu valor etc.

3 A Arquitectura do CodeCompas

Na seção anterior, listamos alguns aspectos referentes às metas e arquiteturas de ferramentas de compreensão de código. Agora apresentamos onde o CodeCompass fica entre essas ferramentas.

O CodeCompass possui uma arquitetura cliente-servidor na qual apresenta o informações coletadas em uma fase de análise anterior. A razão pela qual isso a arquitetura escolhida vem do objetivo da ferramenta. Ao contrário do código editores, o CodeCompass foi planejado para ser uma ferramenta de compreensão de código. Lá Existem diferenças fundamentais entre esses dois casos de uso. Durante a escrita do código, programadores estão manipulando apenas alguns arquivos ao mesmo tempo. Em código No entanto, é necessário considerar as fontes de múltiplos módulos através da base de código. Nos editores, o preenchimento de código é um dos mais recursos úteis: o programador não quer se

lembrar de todos os métodos e campos de uma classe, mas requer que o editor os liste. Na compreensão do código, o É necessária uma ampla gama de visualizações para ter uma visão geral das relações de partes de código. Ao editar a fonte, o programador se concentra apenas em um fragmento relativamente pequeno do código, como uma função ou uma classe. Em código compreensão não é apenas o comportamento de baixo nível das funções, mas suas dependências e efeitos são considerados no contexto de alto nível sistema de módulos.

A principal interface do usuário do CodeCompass é baseada na Web. Todos os itens acima mencionados visualizações e funcionalidades podem ser consultadas através de uma API pública que é atribuído a um aplicativo de servidor. A interface da web lida com os casos de uso que vise tarefas de navegação, inspeção e compreensão rápidas e práticas. Contudo, O CodeCompass é mais do que apenas uma ferramenta de navegação de código. É também uma estrutura, isto é, um coletor extensível e apresentador de processos de análise estática. Naquela é por isso que a intenção não era criar um aplicativo pesado para o cliente que armazena os resultados da análise no lado do cliente, mas ser capaz de servir os vários necessidades dos usuários. Dessa forma, é possível implementar um script, por exemplo, que coleta o conjunto de funções que formam um fechamento por relação de chamada de função, especificando assim uma fatia coerente do software.

Outro requisito de design do CodeCompass era lidar com bases de código em larga escala e ainda atendendo às solicitações do usuário muito rapidamente, ou seja, em termos de segundos, no máximo. Isso é feito armazenando a menor quantidade de informações em um banco de dados suficiente para responder às solicitações. Como pretendemos dar Para resultados precisos para as consultas, é necessário um processo de análise anterior. No No primeiro, armazenamos toda a árvore de sintaxe abstrata da fonte, mas isso resultou em uma proporção de 1:1000 entre o código-fonte e o tamanho do banco de dados. Contudo, descobriu-se que na maioria dos casos os usuários estão interessados apenas em entidades nomeadas (função, variáveis, classes, macros etc.), por isso era desnecessário armazenar qualquer outra coisa, como estruturas de controle ou outras instruções. Não obstante, existem algumas tarefas que exigem mais do que as informações armazenadas, como um algoritmo de corte. Se o usuário quiser ver os efeitos da alteração do valor de uma variável, as declarações de modificação de estado devem ser levadas em consideração também. Isso requer a nova correção do código em tempo real.

4 Características do CodeCompass

Nesta seção, forneceremos uma visão geral sobre os caracteísticas disponíveis no a GUI padrão. Ao descrever recursos específicos do idioma, como listar chamadores de um método, sempre assumiremos que a linguagem do projeto é C ++ como com o suporte mais avançado do CodeCompass, mas recursos semelhantes são disponível para Java e Python.

4.1 Pesquisar

Provavelmente, o caso de uso mais fundamental de uma ferramenta de compreensão de código é procurando. Pode-se procurar um arquivo ou código-fonte. Para encontrar fonte elementos de código, a ferramenta fornece três possibilidades de pesquisa diferentes:

No modo *pesquisa de texto completo*, a frase de pesquisa é um grupo de palavras como "retorna um astnode*". Uma frase de consulta corresponde a um bloco de texto, se a pesquisa as palavras estão próximas umas das outras no código-fonte nessa ordem específica. Caracteres curinga, como * ou? pode ser usado, combinando qualquer múltiplo ou único personagem. Operadores lógicos como AND, OR, NOT podem ser usados para unir múltiplos consulta frases ao mesmo tempo.

Em um nível superior, é possível encontrar símbolos nos códigos-fonte *pesquisa de definição*. Aqui estamos usando CTags para indexar a base de código podendo assim encontrar variáveis, funções, classes, macros, etc. É importante saber que essa busca por entidade de idioma não tem nada a ver com profunda análise de idioma.

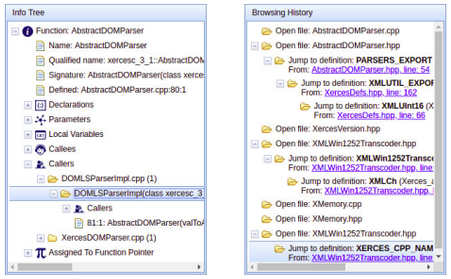
Durante a depuração de um programa, às vezes a única informação para começar é uma mensagem de saída no log do console emitido pelo nosso software. Este é o único traçar onde se pode começar, e.g. `"DEBUG INFO: TSTHan: sys_offset=-0.019821, drift_comp=-90.4996, sys_poll=5"`. Observe que esse pode conter timestamps ou outros fragmentos gerados dinamicamente, portanto é impossível encontrar esta mensagem como uma sequência direta. No entanto, no CodeCompass uma pesquisa difusa pode ser feita por emph log search.

4.2 Informações sobre símbolos de idioma

Quando o elemento for encontrado, a próxima etapa é coletar informações sobre isto. O usuário pode escolher "Árvore de informações" no menu pop-up após selecionar um entidade nomeada. Esta árvore contém todas as informações fornecidas por um idioma analisador. No caso de C/C++, estamos usando o compilador LLVM/Clang para buscar informações sobre os símbolos.

Para funções, podemos verificar seus parâmetros, variáveis locais, chamadores e callees. Uma característica interessante da árvore é que os chamadores são apresentados recursivamente, isto é, os filhos de um nó são os chamadores de uma função. Deles nós filhos são os chamadores dessas funções, e isso continua recursivamente, teoricamente, de volta à função principal. No entanto, chamadas de função nem sempre são diretos, mas podem acontecer através de ponteiros de função. Mesmo que isso é um comportamento dinâmico, o CodeCompass convoca todas as ocorrências em que uma função foi atribuído a um ponteiro de função e a chamada ocorre através deste ponteiro.

No caso de classes, as informações coletadas são os aliases (por `typedef` a classe pode ter um sinônimo), relações de herança (agrupadas visibilidade), amigos, métodos/campos (diretos ou herdados) e usos (como variável local/global, parâmetro de função/tipo de retorno ou campo de outro classe).



(a) Info tree of a function (b) Browsing history

Figura 1: Information collector panels

Para variáveis, é útil conhecer os lugares no código em que foi escrito e leia. Para tipos de enumeração, as constantes de enumeração são listadas com seus valores inteiros.

4.3 Diagramas

As visualizações são uma das representações mais úteis para os seres humanos visão geral de um sistema. O CodeCompass apresenta vários diagramas baseados em símbolos e arquivos. Esses diagramas são baseados em gráficos, ou seja, representam entidades e seus conexões. Estes também são diagramas interativos: passando o mouse sobre o nós a entidade representada é exibida na visualização de texto e clicando em a entidade selecionada se torna o nó central, mostrando suas relações de acordo com o tipo de diagrama.

O diagrama *Function call* mostra todos os chamadores e callees de uma função em um gráfico. O diagrama *herança da classe UML* mostra toda a cadeia de herança até a classe base raiz e recursivamente para todas as classes derivadas. Nós temos também implementou um diagrama *pointer analysis* que mostra a alocação objetos e os ponteiros que possivelmente apontam para eles. Claro que isso é um informações dinâmicas que podem ser coletadas apenas parcialmente em uma análise estática.

Um *diagrama de interface* chamado para um arquivo de origem C/C++ mostra quais cabeçalhos são "usados apenas" ou "implementados" pelo arquivo fornecido. Uso significa que um O arquivo de origem usa outro arquivo se houver um uso de símbolo nele declarado no outro arquivo. O relacionamento de implementação significa que um símbolo é declarado em um arquivo (formando assim uma interface) e definido em outro. Estes As relações também são aplicáveis aos diretórios, considerando os arquivos contidos. No caso de uma linguagem compilada, também existem os arquivos de saída, como objetos e executáveis. Com base nas informações de ligação, podemos apresentar quais fontes fazem uma arquivo binário acima.

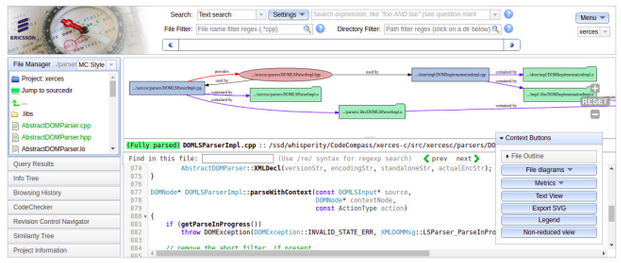


Figura 2: Diagrama do Interface

CodeBites fornece uma visualização diferente da fonte inspecionada código. Nesta visão, os nós do gráfico são as definições de determinados símbolos nomeados, como classes, funções, etc. A idéia é que um programador gostaria de descobrir essa entidade entendendo seu comportamento, mas sem perdendo o foco. Portanto, as partes do texto do código em um nó são clicáveis, aciona a adição da definição do elemento selecionado.

4.4 Visualizações do COntrlo de Versões

A visualização de informações de controle de versão é uma ajuda importante para entender evolução de software. Git *blame view* mostra linha por linha as alterações (confirma) para um determinado arquivo. As mudanças que ocorreram recentemente são mais claras verde, enquanto as alterações mais antigas são vermelhas mais escuras. Essa visão é excelente para revisar por que certas linhas foram adicionadas a um arquivo de origem. O CodeCompass também pode mostrar o Git confirma em uma lista filtrável ordenada no momento da confirmação. Esta pesquisa Esse recurso pode ser usado para listar as alterações feitas por uma pessoa ou para filtrar confirmações por palavras relevantes na mensagem de confirmação.

4.5 Metricas

O CodeCompass pode mostrar a complexidade ciclomática de McCabe [1], as linhas de código e o número de bugs encontrados pelas métricas do Clang Static Analyzer para arquivos individuais e resumidos nas hierarquias de diretórios. Essas métricas pode ser visualizado em um mapa em árvore, onde os diretórios

são indicados por caixas. o o tamanho da caixa e sua tonalidade de cor são proporcionais à métrica escolhida.

4.6 Histórico de Browsing

De Alwis e Murphy estudaram por que os programadores experimentam desorientação quando usando o IDE (Java Integrated Development Environment) do Eclipse [8]. Eles usam momento visual [9] técnica para identificar três fatores que podem levar à desorientação:

i a ausência de conexão do contexto de navegação durante a exploração do programa, ii debulhando entre os monitores para visualizar os trechos de código necessários e item a busca de subtarefas não relacionadas às vezes.

O primeiro fator significa que o programador, durante a investigação de um problema visita vários arquivos da seguinte maneira em uma cadeia de chamadas ou explora o uso de uma variável. No final de uma longa sessão de exploração, é difícil lembrar por que o a investigação terminou em um arquivo específico. A segunda razão para desorientação é a mudança frequente de diferentes visualizações no Eclipse. O terceiro O que contribui para o problema é que um desenvolvedor, ao resolver uma alteração de programa tarefa, avalia várias hipóteses, que são todas de compreensão individual subtarefas. Os programadores tendem a suspender uma subtarefa (antes de terminar) e mudar para outro. Por exemplo, o programador investiga como um valor de retorno de uma função é usada, mas depois muda para uma subtarefa que compreende o implementação da própria função. Observou-se que, por desenvolvedor, é difícil lembrar-se de uma subtarefa suspensa [10].

O CodeCompass implementa uma *histórico de navegação* que registros (em uma árvore formulário) o caminho da navegação no código-fonte. Uma nova subtarefa é representado por um novo ramo da árvore, enquanto os nós são saltos de navegação no código rotulado pelo contexto de conexão (como " pule para a definição de init "). Portanto, o problema i) e ii) é tratado pelos nós rotulados no histórico de navegação, enquanto o problema iii) é tratado pelos ramos atribuídos a subtarefas.

4.7 CodeChecker - C/C++ Reportar Errros

O Clang Static Analyzer implementa um mecanismo de execução simbólica avançado para relatar falhas de programação. O CodeCompass pode visualizar os bugs identificados pelo Clang Static Analyzer e Clang Tidy conectando-o a um servidor CodeChecker [11]. CodeCompass mostra a posição do erro e o caminho de execução simbólico que leva a uma falha.

4.8 Nomes e Tipos

O CodeCompass processa a documentação do Doxygen e os armazena para a função, tipo, definições de variáveis. Ele também fornece uma visão *type catalog* que lista os tipos declarados no espaço de trabalho organizado por uma exibição em árvore hierárquica de namespaces.

5 Sumário

Apresentamos o CodeCompass, uma ferramenta de análise estática para compreensão de software em larga escala. Foi projetado para evitar as várias faltas das ferramentas de compreensão existentes que são leves, fáceis usar, mas sem o profundo conhecimento de um compilador real; ou pesado, não escalável instalado na máquina cliente. Ter um plug-in baseado na Web, arquitetura extensível, a estrutura pode ser uma plataforma aberta para esforços de compreensão de código, análise estática e métricas de software. O feedback inicial do usuário e as estatísticas de uso sugerem que a ferramenta é útil para desenvolvedores em atividades de compreensão e é usado além dos tradicionais IDEs e outras ferramentas de referência cruzada.

Agradecimentos

Este artigo é parte da produção intelectual O2 do projeto Erasmus+ Key Action 2 project No. 2017-1-SK01-KA203-035402, the Strategic Partnership for Higher Education *"Focusing Education on Composability, Comprehensibility and Correctness of Working Software" (3COWS).*

Aviso Legal

Referências

1. Thomas J. McCabe, *A Complexity Measure*, IEEE Transactions on Software Engineering: 308–320, December 1976
2. Henderson-Sellers, *Object-Oriented Metrics: Measures of Complexity* Prentice-Hall, 1996, Upper Saddle River, NJ, ISBN-13: 978-0132398725
3. Woboq, https://woboq.com/codebrowser.html, 18. 03. 2018
4. OpenGrok, https://opengrok.github.io/OpenGrok, 18. 03. 2018
5. CTAGS, http://ctags.sourceforge.net, 18. 03. 2018
6. Understand, https://scitools.com, 18. 03. 2018
7. CodeSurfer, https://www.grammatech.com/products/codesurfer, 18. 03. 2018
8. B. De Alwis and G.C. Murphy, *Using Visual Momentum to Explain Disorientation in the Eclipse IDE*, Proc. IEEE Symp. Visual Languages and Human Centric Computing, pp. 51-54, 2006. [2] E. Baniassad and G. Murphy, "Conceptual Module Querying for Software Engineering," Proc. Int'l Conf. Software Eng., pp. 64-73, 1998.
9. D. D. Woods., *Visual momentum*, A concept to improve the cognitive coupling of person and computer. Int. J. Man-Mach. St., 21:229–244, 1984.

10. D. Herrmann, B. Brubaker, C. Yoder, V. Sheets, and A. Tio. *Devices that remind*, In F. T. Durso et al., editors, Handbook of Applied Cognition, pages 377–407. Wiley, 1999.
11. Daniel Krupp, Gyorgy Orban, Gabor Horvath and Bence Babati, *Industrial Experiences with the Clang Static Analysis Toolset*, EuroLLVM 2015 Confernece, April 2015