UNIVERSITÄT
LEIPZIG

SOFTWARE
SYSTEME

Seminar Automated Software Engineering

SoSe 2022

Norbert Siegmund

Themen

Arten:

- Replikation
- Anwendung
- Literaturstudie (Survey)

Replication

- Nachvollziehen eines Ansatzes
- Anwendung bzw. Nach-Implementierung
- Ausführen von Experimenten

(J) Deep Learning Methods for Configurable Software Systems

Auch im Software Engineering hat Deep Learning Einzug gefunden. In der Literatur wurden Deep Neural Networks (DNNs) und Generative Adversarial Networks eingesetzt um die Performance von Software unter einer bestimmten Konfiguration vorauszusagen. Beim Umgang mit Konfigurationen, also strukturierten Daten, sind jedoch auch andere Ansätze des Maschinellen Lernens geeignet und der Vorteil von DNN fraglich und oft nicht replizierbar.

Vorwissen: Machine Learning, Artificial Neural Networks / Deep Learning

Inhalt:

- Implementieren von DNNs als Replikation von bestehenden Arbeiten
- Nutzung von NVIDIA-A100-Server-Hardware

Ergebnis: Vergleich der Machine-Learning-Ansätze

Literatur:

- Ha, Huong, and Hongyu Zhang. 2019. “DeepPerf: Performance Prediction for Configurable Software with Deep Sparse Neural Network.” *Proceedings - International Conference on Software Engineering*, May, 1095–1106. https://doi.org/10.1109/ICSE.2019.00113.
- Shu, Yangyang, Yulei Sui, Hongyu Zhang, and Guandong Xu. 2020. “Perf-AL: Performance Prediction for Configurable Software through Adversarial Learning.” In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–11. Bari Italy: ACM. https://doi.org/10.1145/3382494.3410677.

(J) Causal Reasoning for Configurable Software Systems

Den Energieverbrauch von konfigurierbaren Softwaresystemen für bestimmte Konfigurationen vorherzusagen, ist gerade ein Problem, wenn man mit verschiedener Hardware und Workloads arbeitet. Dann werden bestehende Performance-Modelle ungenau. Der Ansatz UNICORN behauptet, auch Hardware- und Workload-übergreifend zu funktionieren, belegt das aber nur mit wenigen Fallstudien. Gelten die Behauptungen auch mit anderer Hardware?

Vorwissen: (Machine Learning, Docker)

Inhalt:

- Daten-Transformation
- Replikation mithilfe von Docker-Containern
- Erstellung von Causal Graphs und Machine-Learning-Modellen

Ergebnis: Evaluation und Vergleich mit Original-Paper

Literatur:

- Iqbal, Md Shahriar, Rahul Krishna, Mohammad Ali Javidian, Baishakhi Ray, and Pooyan Jamshidi. 2022. “Unicorn: Reasoning about Configurable System Performance through the Lens of Causality.” In *Proceedings of the Seventeenth European Conference on Computer Systems*, 199–217. Rennes France: ACM. https://doi.org/10.1145/3492321.3519575.
- https://github.com/softsys4ai/unicorn/blob/master/artifact/OTHERS.md

(Sim) Configuration File Specification with Association Rule Learning

Santolucito et al. propose ConfigV, a verification framework that aims at ensuring the correctness of configuration files. They derive configuration specifications by using a machine learning approach. Their tools checks whether configuration files meet the specification. This way, they are able to detect different kinds of configuration errors, such as ordering, type, and missing entry errors.

Vorwissen: Machine Learning, Programming Skills

Inhalt/Evaluierung

- Erhebung neuer Daten
- Replikation des Ansatzes
- Evaluierung und Vergleich mit Original-Paper

Literatur/Quellen:

- Santolucito et al. 2017. Synthesizing Configuration File Specifications with Association Rule Learning.
- ConfigV (https://github.com/ConfigV/ConfigV)

(Sim) Configuration Error Injection Testing

In recent years, several tools have been developed that inject misconfigurations into software systems in order to systematically analyze the reliability of software systems in the presence of misconfigurations.

Vorwissen: Programming Skills (Python, Java)

Inhalt/Ergebnis:

- Implementierung (Replikation) bestehender Configuration Error Injection Testing Strategien

Literatur/Quellen:

- Wang et al. 2021. Challenges and Opportunities: An In-Depth Empirical Study on Configuration Error Injection Testing
- Keller et al. 2008. ConfErr: A tool for assessing resilience to human configuration errors
- Li et al. 2017. Conftest: Generating comprehensive misconfiguration for system reaction ability evaluation
- Li et al. 2018. ConfVD: System Reactions Analysis and Evaluation Through Misconfiguration Injection
- CeitInspector (https://github.com/ConfEIT-code/CeitInspector)
- ConfErr (https://github.com/lokeller/conferr)

(Mue) Uniform Random Sampling for Configuration Spaces

Drawing sample configurations from the configuration space is key to obtaining robust performance prediction models; different approaches are used to achieve uniform coverage by using heuristics (distance-based sampling, feature balance) or analytic approaches (partitioning the configuration space with counting BDDs); approaches can be re-implemented and compared, potentially literature summary on how approaches tackle binary (categorical) as well as numeric (interval) data.

Vorwissen/Requirements: SAT/SMT Solver, Variability Modeling

Inhalt/Ergebnis:

- Implementierung (Replikation) bestehender Sampling-Strategien
- Vergleich dieser im Bezug auf Abdeckung des Konfigurations-Raums, Probleme von Samplen mit SMT-Solvern

Literatur:

- Munoz et al.: Uniform Random Sampling Product Configurations of Feature Models That Have Numerical Features. SPLC 2019
- Kaltenecker et al.: Distance-Based Sampling of Software Configuration Spaces. ICSE 2019
- Henard et al.: Combining multi-objective search and constraint solving for configuring large software product lines. ICSE 2015
- Oh et al.: Finding near-optimal configurations in product lines by random sampling. FSE 2017
- …

(M) Multi-Objectivizing Energy-Tuning with Method-Level Energy-Influence Models

Multi-Objectivizing wird verwendet, wenn man eine komplizierte und holprige Zielfunktion (Objective) optimieren möchte. Hierbei nimmt man zur eigentlichen Zielfunktion (bspw. Energieverbrauch) noch eine weitere (bspw. Performance) hinzu, um die Optimierung robuster zu machen. Neue Zielfunktionen sollten neue Informationen besitzen, sich jedoch von der originalen Zielfunktion unterscheiden. Methoden-Energie-Modelle bieten sich an, zusätzlichen Kontext aus dem Source Code in die Optimierung einzubringen.

Vorwissen: Search-Based Software Engineering

Inhalt:

- Implementierung des ‘Multi-Objectivizing Software Configuration Tuning’ - Ansatzes
  - mögliche neue Zielfunktionen: Methoden-Modelle, Ausführzeit (black-box)
- Anwendung auf Daten von realen Softwaresystemen

Ergebnis:

- Evaluation und Vergleich mit Single-Objective-Optimierung

Literature:

- Chen, Tao, and Miqing Li. 2021. “Multi-Objectivizing Software Configuration Tuning (for a Single Performance Concern).” *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, August, 453–65. https://doi.org/10.1145/3468264.3468555.

Anwendung

- Anwendung von Konzepten aus Vorlesung & Literatur
  - mit eigenen Ideen
- ohne einen konkreten Ansatz aus der Literatur

(J) Optimizing Energy Consumption and Execution Time

Oftmals ist die default-Konfiguration von Softwaresystemen sub-optimal. Um die Response Time oder den Energieverbrauch zu optimieren, bieten sich kombinatorische Multi-Objective Meta-Heuristiken aus der Vorlesungen an.

Vorwissen: Search-Based Software Engineering

Inhalt:

- Sichtung realer Mess-Daten
- Implementieren von verschiedenen kombinatorischen Multi-Objective Meta-Heuristiken
- Anwendung auf Daten von realen Softwaresystemen

Ergebnis:

- Evaluation und Vergleich mit Default-Konfiguration

Literatur:

- Vorlesung

(Sie) Search strategies for source code generation using Transformers

Transformers have evolved to become one of the most used architectures for generating source code. Several models have become available in the last years: CodeGPT, GPT-Neo, CodeParrot, PolyCoder. They are now a key part of the code completion systems and are currently used for predicting a single token or a complete statement. When generating complete sequences the model has to sample from the prediction space in order to generate the next token. The most popular approaches are Top-K sampling and beam search decoding, however there are few studies comparing the quality of samples generated by both search methods.

Previous Knowledge: SBSE, Deep learning, Natural language processing

Contents:

- Adaptation of current open source implementations
  - CodeParrot: https://huggingface.co/lvwerra/codeparrot
  - GPT-Neo: https://huggingface.co/EleutherAI/gpt-neo-125M
- Implementation of sampling strategies

Result:

- Evaluation and comparison of quality of generated samples
- Analysis of the impact on the performance of each technique.

Literature:

- Welleck et al, 2019. Neural text degeneration with unlikelihood training
- Lu et al, 2021. CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation (**CodeGPT**)
- Feng et al, 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages

Literaturstudie (Survey)

- Literaturrecherche zum Thema
- Zusammenfassen der wissenschaftlichen Arbeiten
- Identifizierung von Trends und Probleme

(M) Method-Level Energy Profiling

Conventional power meters for computers are too slow to sample methods , but how can fast energy measurements be traced back to methods? What approaches exist and how do they compare?

Context:

- Energie Debugging Automatisieren

Inhalt:

- In Literatur genutzte Leistungsmesssysteme
- Entwickelte Methoden, um Quelltext-Regionen den Energieverbrauch zuzuordnen

Literatur:

- Bedard, Daniel, Robert Fowler, Min Yeol Lim, and Allan Porterfield. 2009. “PowerMon 2: Fine-Grained, Integrated Power Measurement RENCI Technical Report TR-09-04.”
- Hackenberg, Daniel, Thomas Ilsche, Robert Schöne, Daniel Molka, Maik Schmidt, and Wolfgang E. Nagel. 2013. “Power Measurement Techniques on Standard Compute Nodes: A Quantitative Comparison.” In *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 194–204. https://doi.org/10.1109/ISPASS.2013.6557170.
- Flinn, J., and M. Satyanarayanan. 1999. “PowerScope: A Tool for Profiling the Energy Usage of Mobile Applications.” In *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, 2–10. New Orleans, LA, USA: IEEE. https://doi.org/10.1109/MCSA.1999.749272.
- Di Nucci, Dario, Fabio Palomba, Antonio Prota, Annibale Panichella, Andy Zaidman, and Andrea De Lucia. 2017. “PETrA: A Software-Based Tool for Estimating the Energy Profile of Android Applications.” In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, 3–6. Buenos Aires: IEEE. https://doi.org/10.1109/ICSE-C.2017.18.
- Mukhanov, L., Pavlos Petoumenos, Zheng Wang, K. Parasyris, Dimitrios S. Nikolopoulos, B. D. Supinski, and H. Leather. 2017. “ALEA: A Fine-Grained Energy Profiling Tool.” *ACM Trans. Archit. Code Optim.* https://doi.org/10.1145/3050436.
- Wei, Guang, Depei Qian, Hailong Yang, Zhongzhi Luan, and Lin Wang. 2019. “FPowerTool: A Function-Level Power Profiling Tool.” *IEEE Access* 7: 185710–19. https://doi.org/10.1109/ACCESS.2019.2961507.
- Babakol, Timur, Anthony Canino, Khaled Mahmoud, Rachit Saxena, and Yu David Liu. 2020. “Calm Energy Accounting for Multithreaded Java Applications.” In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 976–88. Virtual Event USA: ACM. https://doi.org/10.1145/3368089.3409703.

(M) Beyond Software Configurations

How are hardware and workload varied in literature? Only whole hardware or workload change? What parametric hardware and workload are used?

Kontext:

- Ist derzeit schon automatischer Transfer von Modellen möglich?

Inhalt:

- In welchen Arbeiten wird der Einfluss von verschiedener Hardware bzw. verschiedenem Workload auf NFP untersucht?
- Wird die gesamte Hardware / der gesamte Workload geändert oder nur einzelne Parameter?

Literatur:

- Iqbal, Krishna, Javidian, Ray, and Jamshidi. 2022. Unicorn: Reasoning about Configurable System Performance through the lens of Causality. arXiv preprint arXiv:2201.08413
- Valov, Pavel, Jean-Christophe Petkovich, Jianmei Guo, Sebastian Fischmeister, and Krzysztof Czarnecki. 2017. “Transferring Performance Prediction Models Across Different Hardware Platforms.” In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering - ICPE ’17*, 39–50. ACM Press. https://doi.org/10.1145/3030207.3030216.
- Zamora, Yuliana, Bethany Lusch, Venkat Vishwanath, Ian Foster, Murali Emani, and Henry Hoffmann. 2021. “Cross Architecture Performance Prediction,” 20.

(Sim) Inferring Configuration Dependencies/Constraints

Misconfigurations have become one of the dominant causes for configuration errors.

There are several approaches that tackle configuration errors by inferring configuration dependencies/constraints.

Inhalt/Ergebnis:

- Literaturrecherche
- Überblick/Zusammenfassung der Ansätze

Literatur:

- Chen et al. 2020. Understanding and discovering software configuration dependencies in cloud and datacenter systems.
- Zhou et al. 2021. ConfInLog: Leveraging Software Logs to Infer Configuration Constraints
- Xu et al. 2013. Do not blame users for misconfigurations.
- Nadi et al. 2015. Where do configuration constraints stem from? An extraction approach and an empirical study.
- Nadi et al. 2014. Mining configuration constraints: Static analyses and empirical results.

(Sie) Neural code summarization in Python

Code summarization attempts to generate clear and concise description of a program's functionality. Natural language summaries can help to reduce the effort of developers while helping them to comprehend a program. Motivated by one of the recent surveys on code summarization in Java (Shi et al, 2022), we propose a survey of neural-based code summarization techniques for Python.What neural based techniques are employed? What training pairs are more effective for a code summarization model? Comment-code pairs or "commit messages"-"code diff" pairs? What strategies are proposed by the researchers to evaluate the quality of the proposed summaries? What remain open challenges?

Previous Knowledge: Deep learning, natural language processing

Contents/Results:

- Literature research
- Overview/summary of approaches
- Identification of trends and open challenges

Literature:

- Shi et al 2022, On the Evaluation of Neural Code Summarization
- Ahmad et al 2020, A Transformer-based Approach for Source Code Summarization
- Iyer et al 2016, Summarizing source code using a neural attention model (First neural, but in Java)

(Sim) Learning Configuration Rules Using Machine Learning

Many approaches aim to detect configuration errors. In particular, a common approach relies on machine learning to automatically extract configuration rules that ensure the correctness of configuration files.

Inhalt/Ergebnis:

- Literaturvergleich
- Zusammenfassung und **Vergleich** der Ansätze

Literatur:

- Santolucito et al. 2016. Probabilistic Automated Language Learning for Configuration Files.
- Santolucito et al. 2017. Synthesizing Configuration File Specifications with Association Rule Learning.
- Zhang et al. 2014. EnCore: Exploiting System Environment and Correlation Information for Misconfiguration Detection

(Mue) Performance Test Case Prioritization (for config. software)

SBSE practices include reducing the number of tests as well as the overall test load, across module and system level. We can prioritise with regard to the number of configurations, the workloads, or the revisions selected for testing. What work exists on minimizing the test load at different granularities (module/system) and for different dimensions (configuration, workload, revisions), what SBSE techniques are used, which practices have already been introduced in industry, what remain open challenges?

Inhalt/Ergebnis:

- Literaturrecherche und Überblick/Zusammenfassung
- Identifizierung von Trends und offenen Challenges

Literatur:

- Daly, D..: Creating a Virtuous Cycle in Performance Testing at MongoDB. ICPE 2021
- Alcocer et al.. "Prioritizing versions for performance regression testing: the pharo case.Science of Computer Programming 191 (2020): 102415
- Laaber et. al: An Evaluation of Open-Source Software Microbenchmark Suites for Continuous Performance Assessment. MSR 2018
- Liao et al: Using blackbox performance models to detect performance regressions under varying workloads an empirical study. EMSE 2021

Typ	ID	Titel
Replikation	R1	(J) Deep Learning Methods for Configurable Software Systems
	R2	(J) Causal Reasoning for Configurable Software Systems
	R3	(Sim) Configuration File Specification with Association Rule Learning
	R4	(Sim) Configuration Error Injection Testing
	R5	(Mue) Uniform Random Sampling for Configuration Spaces
	R6	(M) Multi-Objectivizing Energy-Tuning with Method-Level Energy-Influence Models
Anwendung	A1	(J) Optimizing Energy Consumption and Execution Time
	A3	(Sie) Search strategies for source code generation using Transformers
Literaturstudie	L1	(M) Method-Level Energy Profiling
	L2	(M) Beyond Software Configurations
	L3	(Sim) Inferring Configuration Dependencies/Constraints
	L4	(Sie) Neural code summarization in Python
	L5	(Sim) Learning Configuration Rules using Machine Learning
	L6	(Mue) Performance Test Case Prioritization (for Configurable Software Systems)

UNIVERSITÄT
LEIPZIG