On Learning Parametric Dependencies from Monitoring Data

Johannes Grohmann, Simon Eismann, Samuel Kounev
{first_name}.{last_name}@uni-wuerzburg.de
University of Würzburg, Germany

Abstract

A common approach to predict system performance are so-called architectural performance models. In these models, parametric dependencies describe the relation between the input parameters of a component and its performance properties and therefore significantly increase the model expressiveness. However, manually modeling parametric dependencies is often infeasible in practice. Existing automated extraction approaches require either application source code or dedicated performance tests, which are not always available. We therefore introduced one approach for identification and one for characterization of parametric dependencies, solely based on run-time monitoring data.

In this paper, we propose our idea on combining both techniques in order to create a holistic approach for the identification and characterization of parametric dependencies. Furthermore, we discuss challenges we are currently facing and potential ideas on how to overcome them.

1 Introduction

The parameterization of a software performance model, i.e., the values for model parameters such as loop frequencies, branching probabilities or resource demands, is a significant factor influencing its prediction accuracy. However, these parameters often depend on the input given to the respective component (e.g., the time required to sort a list depends on its size). Therefore, many architectural performance models include so-called *parametric dependencies* [11], allowing to explicitly model input parameters and their influence on model parameters. These dependencies describe, e.g., the resource demand of a function call in dependence of its input parameters.

However, modeling parametric dependencies is cumbersome in practice, as it requires expert knowledge, is quite error-prone and requires a significant amount of human work. For example, in a case study by Krogmann et al. [4] more than 24 hours were required to manually model the parametric dependencies in a small system. Therefore, manually modeling parametric dependencies for large systems is infeasible. Manual effort generally hinders the adoption of performance modeling techniques in practice [14].

Krogmann et al. [4] use genetic search to find dependencies between a component's input parameters and the number of executed bytecode instructions. Other approaches like Mazkatli et al. [13] incrementally enrich performance models based on individual code changes. However, these approaches require either application source code or dedicated performance tests to extract the parametric dependencies.

We previously proposed to identify parametric dependencies solely from run-time monitoring data [15]. The used monitoring data contains call-chain traces as well as logs of individual parameter values together with the resource demand or the response time of each function execution, as observable e.g., by Kieker [7]. Similarly, it is possible to characterize a set of given parametric dependencies using monitoring data and standard machine learning regression techniques [10]. Therefore, we combine both approaches in order to create one holistic and autonomic approach able to learn parametric dependencies from black-box monitoring data. In this paper, we present our approach on creating such a combined approach and outline some of the challenges we are expecting on our way.

2 Learning Parametric Dependencies from Monitoring Data

We structure the task of extracting performance models containing characterized parametric dependencies and place it in the context of existing related work in Figure 1. Starting with a set of monitoring traces from a software system, one can utilize existing approaches to extract a performance model without parametric dependencies [1, 3, 5, 6, 9, 16]. At the same time, the monitoring data is fed into our dependency learning pipeline. This task can be split into two sub-tasks: (1) detecting the dependencies, that is, identifying which parameters influence a model variable, and (2) characterizing the dependencies, that is, describing how the value of a parameter can be derived from the influencing parameters. In the following, we will outline our techniques for the two sub-tasks in more detail.

2.1 Identifying Parametric Dependencies

The first step is to identify which model variables are dependent on which input parameters. Our approach achieves this by leveraging feature selection techniques from machine learning. The identification of any parametric dependencies between different variables of the

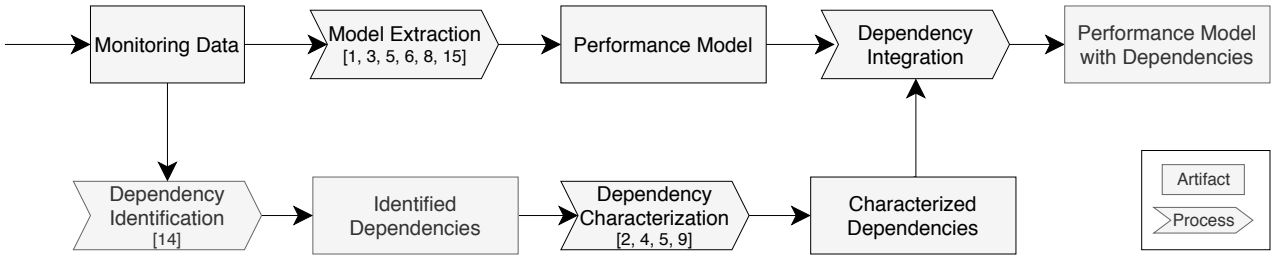

Figure 1: Model extraction workflow.

monitoring stream can be framed as a classic application of feature selection: We define one model parameter as a target parameter and consider all other model parameters as potential features. The challenges when applying feature selection to this domain are to obtain suitable measurement streams, to filter and select the most promising dependencies, and to discard a detected dependency if there is no modeling gain. We propose a generic algorithm for the automated identification of parametric dependencies on monitoring streams and apply three different heuristics to filter the identified dependencies [15]. These heuristics utilize domain knowledge to drastically decrease the number of identified dependencies reducing them to only performance relevant ones. After evaluating a filter-based, a wrapper-based and an embedded feature selection technique, we show that the filter-based approach outperforms the competing techniques by identifying 11 performance-relevant dependencies in our TeaStore [12] case study.

2.2 Characterizing Dependencies

After a set of possible parametric dependencies is identified, these dependencies can then be characterized as also done by [2, 4, 5]. Our approach is to focus on standard regression techniques. Since the characterization of any labeled dependency is a relatively straightforward regression task, our study creates a representative data set containing parametric dependencies and evaluates how well a range of machine learning approaches can characterize the contained parametric dependencies [10]. The study finds that no machine learning approach performs well for all parametric dependencies, i.e., that there is no single algorithm best suited for the task. Therefore, we propose a meta-selector selecting the most appropriate regression technique for every dependency, based on the characteristics of the available data. This meta-selector reduces the prediction error compared to the best individual approach by 30%.

3 Remaining Challenges

Our next step is now to conduct a comprehensive case study combining both approaches and the model extraction pipeline as depicted in Figure 1 in order to evaluate the impact of extracting parametric dependencies. However, while doing so, we still face many open challenges, some of which we outline below.

3.1 Feature Generation

One of the most important aspects when applying machine learning techniques are the available features. When utilizing our approach, it is necessary to generate machine learning features from the monitoring data. This implies that all call parameters are logged. However, while some primitive types are easy to convert to machine learning features (e.g., integers, floats, or enumerations), other more complex data types (e.g., lists, JSON data blocks, Strings, or complex queries) pose some significant challenges. Some of these can be solved by domain knowledge (e.g., quantifying a list by its length or classifying different SQL queries into a set of categories), but this limits the autonomy and the generalizability of the approach and is therefore undesirable. One idea could be to utilize clustering in order to group different requests into different categories based on the response of the software system.

3.2 Estimating Resource Demands

Next to accurately monitoring the call parameters, monitoring the model parameters for each corresponding call is equally important. Some model parameters (e.g., branching probabilities or external call counts) are easy to measure. However, some model parameters like resource demands are very challenging to obtain. As measuring resource demands usually introduces a lot of overhead, resource demands are often estimated [8]. However, most estimators can not estimate demands for individual calls. Unfortunately, the resource demands of every single call together with the corresponding parameter values are required in order to draw conclusions about their correlation. One solution is to approximate the resource demand with the response time of a function call and to keep the load on the system sufficiently low in order to reduce queueing delay. However, this solution has disadvantages so other solutions to this problem might be beneficial.

3.3 Baseline Selection

As the manual modelling of dependencies in performance models is actually infeasible, it is hard to compare the automatically extracted models to manual

models on a large scale. One possibility is to compare the detected dependencies with the ones found by other approaches (that use source code or dedicated measurements). However, this puts the alternative approaches in the role of the gold-standard which can be problematic as it assumes that the alternative approaches work perfectly. Additionally, as they are based on a different paradigm, they need to have access to different and/or more information. Therefore, the only meaningful comparison can be done by evaluating the accuracy gain of the model enriched with parametric dependencies compared to a basic model, not containing any (or few) parametric dependencies. However, there is no way of determining the "upper limit" or the gold standard when using this approach.

3.4 Performance Model Integration

The final task after obtaining and successfully characterizing a set of dependencies is to integrate them into a given performance model. Our approach does not focus on any specific modeling formalism in order to be independent of the underlying formalism. However, most of them expect a dependency formalization in the form of a mathematical expression, e.g., $f(x) = y$, as it is usually undesirable for architectural performance models to contain black-box functions as they are intended to be human interpretable. As our approach [10] applies different black-box machine learning algorithms, it is not trivial to convert them to a simple mathematical expression. We are therefore currently looking into solutions approximating the resulting black-box machine learning model with a mathematical expression.

4 Conclusion

In this paper, we proposed our approach on combining (i) feature selection techniques, and (ii) standard regression together with meta-selection to first identify and then characterize parametric dependencies for performance models solely based on monitoring data. We quickly summarized the results from our previous studies and outlined three remaining challenges when combining both approaches, namely (i) the generation of machine learning features from monitored parameters, (ii) the extraction of resource demand estimates for each of these calls, and (iii) the evaluation of our approach by selecting proper baselines.

References

[1] C. E. Hrischuk et al. "Trace-Based Load Characterization for Generating Performance Software Models". In: *IEEE Trans. Softw. Eng.* 25.1 (Jan. 1999), pp. 122–135.

[2] M. Courtois and M. Woodside. "Using Regression Splines for Software Performance Analysis". In: *Proceedings of the 2nd International Workshop on Software and Performance*. 2000, pp. 105–114.

[3] T. A. Israr et al. "Automatic Generation of Layered Queuing Software Performance Models from Commonly Available Traces". In: *Proceedings of the 5th International Workshop on Software and Performance*. WOSP '05. New York, USA: ACM, 2005, pp. 147–158.

[4] K. Krogmann, M. Kuperberg, and R. Reussner. "Using genetic search for reverse engineering of parametric behavior models for performance prediction". In: *IEEE Transactions on Software Engineering* 36.6 (2010), pp. 865–877.

[5] F. Brosig, N. Huber, and S. Kounev. "Automated Extraction of Architecture-Level Performance Models of Distributed Component-Based Systems". In: *26th IEEE/ACM International Conference On Automated Software Engineering (ASE 2011)*. Oread, Lawrence, Kansas, Nov. 2011.

[6] A. Mizan and G. Franks. "An automatic trace based performance evaluation model building for parallel distributed systems". In: *SIGSOFT Softw. Eng. Notes* 36.5 (Sept. 2011), pp. 61–72.

[7] A. van Hoorn, J. Waller, and W. Hasselbring. "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis". In: *Proceedings of the 3rd joint ACM/SPEC International Conference on Performance Engineering*. 2012, pp. 247–248.

[8] S. Spinner et al. "Evaluating Approaches to Resource Demand Estimation". In: *Perform. Evaluation* 92 (Oct. 2015), pp. 51–71.

[9] J. Walter et al. "An Expandable Extraction Framework for Architectural Performance Models". In: *Proceedings of the 3rd International Workshop on Quality-Aware DevOps (QUDOS'17)*. ACM, Apr. 2017.

[10] V. Ackermann et al. "Black-box Learning of Parametric Dependencies for Performance Models". In: *13th International Workshop on Models@run.time (MRT), co-located with ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems (MODELS 2018)*. (Oct. 14, 2018). CEUR Workshop Proceedings. Copenhagen, Denmark, Oct. 2018.

[11] S. Eismann et al. "Modeling of Parametric Dependencies for Performance Prediction of Component-based Software Systems at Run-time". In: *IEEE International Conference on Software Architecture*. 2018.

[12] J. von Kistowski et al. "TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research". In: *Proceedings of the 26th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*. MASCOTS '18. Sept. 2018.

[13] M. Mazkatli and A. Koziolek. "Continuous Integration of Performance Model". In: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*. ICPE '18. Berlin, Germany: ACM, 2018, pp. 153–158.

[14] C. Bezemer et al. "How is Performance Addressed in DevOps?" In: *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, ICPE 2019, Mumbai, India, April 7-11, 2019*. 2019, pp. 45–50.

[15] J. Grohmann et al. "Detecting Parametric Dependencies for Performance Models Using Feature Selection Techniques". In: *Proceedings of the 27th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*. MASCOTS '19. Rennes, France, Oct. 2019.

[16] S. Spinnner et al. "Online model learning for self-aware computing infrastructures". In: *Journal of Systems and Software* 147 (2019), pp. 1–16.