Feature Model Slicing for Real-time Selection of Mission-critical Edge Application

Uwe Gropengießer*
Technical University of Darmstadt
Darmstadt, Germany

Julian Liphardt*
Technical University of Darmstadt
Darmstadt, Germany

Michael Matthé†
University of Stuttgart
Stuttgart, Germany

Max Mühlhäuser*
Technical University of Darmstadt
Darmstadt, Germany

ABSTRACT

At first glance, running mission-critical applications at the edge appears to be an opportunity to benefit from scalability and reusability. The low latency to the edge makes it particularly interesting for mission-critical applications. The hardware heterogeneity of the edge, coupled with the strict requirement for the execution time of a mission-critical application, creates the need for flexible application control and, at the same time, increases the complexity of modeling such systems. With its Feature Models (FMs), software product line engineering offers a modeling option for various alternative compositions of an application. However, the calculation of valid configurations takes too long for the dynamic adaptation of an application flow of a mission-critical application. This paper presents an approach for slicing FMs to support mission-critical applications. Our approach supports the strict requirements on the execution time of mission-critical applications.

CCS CONCEPTS

• **Software and its engineering → Software product lines**; **Distributed systems organizing principles**.

KEYWORDS

Software Product Lines, Feature Model, Approximate Computing, Edge Computing

ACM Reference Format:
Uwe Gropengießer, Julian Liphardt, Michael Matthé, and Max Mühlhäuser. 2024. Feature Model Slicing for Real-time Selection of Mission-critical Edge Application. In *39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24), October 27-November 1, 2024, Sacramento, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3691620.3695321

1 INTRODUCTION

In today's era of technological advancement, edge computing solutions play a critical role in optimizing response times and resource efficiency, especially in mission-critical applications. These applications, which often have strict execution time requirements, require innovative approaches to overcome the inherent challenges of distributed systems. Edge computing offers an attractive solution by decentralizing processing directly at the first entry point of the network (e.g., a 5G tower), close to the data source, minimizing latency, and optimizing bandwidth utilization [3].

However, a key challenge when implementing such systems is the complexity of configuring and adapting these systems to dynamic environmental conditions and different requirements. Feature Models (FMs) [1] offer a powerful method for describing the diverse configuration options of software products, which can be particularly useful in a hardware heterogeneous environment such as edge computing. By modeling different functionalities and properties, they make it possible to record and manage the numerous variants of applications systematically.

Moreover, the constrained resources at the edge frequently necessitate a trade-off between computation time and resource usage. Approximate computing becomes relevant in precisely these situations. Approximate calculation methods tolerate estimable inaccuracies in the calculations to improve energy efficiency or shorten computation times [5]. For mission-critical applications that need to meet strict deadlines, approximate computing offers a way to perform computations under strict constraints by balancing adequate accuracy with a considerable reduction in resource consumption.

Existing work that uses FMs to deploy applications at the edge focuses on optimizing energy [2]. However, mission-critical applications must fulfill a wide range of requirements. One essential requirement is adherence to a maximum execution time. It is, therefore, essential that the possible execution types can be found quickly at an edge. To this end, we address how we can split FMs so they can meet the strict deadline of a mission-critical application.

2 FEATURE MODEL SLICING

The complexity of FMs in mission-critical applications at the edge presents significant challenges, particularly regarding execution time and storage needs. These challenges escalate as the software complexity increases, leading to an exponential rise in the number of valid configurations. Fig. 1 illustrates the execution times using the SAT4J solver [7] for FMs with varying numbers of valid configurations. The complexity of the FMs is influenced by the number of tasks and the available approximation options per task. In detail, the overall complexity of the model, which consists of several tasks, is calculated as the product of the number of configurations of all tasks. If t is the number of tasks and each task has a_i alternatives, then the overall complexity is $\prod_{i=1}^{t} a_i$. As shown in Fig. 1, the execution time without slicing the FM can extend to several hours.

*{gropengiesser@tk, julian.liphardt@stud, max@informatik}.tu-darmstadt.de
†michael.matthe@ipvs.uni-stuttgart.de

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ASE '24, October 27-November 1, 2024, Sacramento, CA, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1248-7/24/10
https://doi.org/10.1145/3691620.3695321

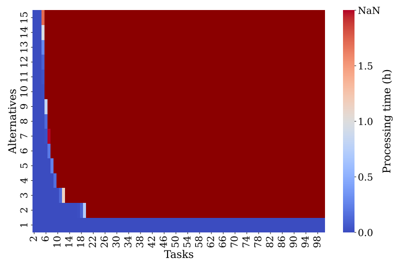

Figure 1: Heatmap of Average Execution Times without Slicing, Varying between 2 to 100 Tasks and 2 to 15 Alternatives per Task.

If we take the combination of $t = 7$ and $a_i = 6$ as an example, this already results in a complexity of 279,936 for the FM. The execution time here is already around half an hour.

To mitigate these issues, we introduce a novel approach that decomposes existing FMs into Partial Feature Models (PFMs) during the offline phase, which is performed prior to any service requests and involves "single-time tasks". This decomposition significantly reduces the complexity of the FMs and accelerates the process of identifying valid configurations, thereby enhancing system responsiveness during the online phase.

Using predefined rules, our method utilizes specialized slicing techniques to streamline the structure of FMs. These rules are designed to identify crucial junctures for decomposition, specifically: 1) initiating and terminating parallel executions and 2) capping the complexity of sequences based on set thresholds. By breaking down the FM into manageable PFMs, each representing a simplified fragment of the original model, our approach not only speeds up the identification of valid configurations but also cuts down on storage needs by diminishing redundancy.

The reassembly of these PFMs into a valid configuration of the original FM occurs dynamically during the online phase, allowing the system to adapt to real-time changes. This is done by focusing only on parts of the FM relevant to the available edge hardware, reducing processing. Finally, we transform the FMs into a graph structure, enabling existing work [6] to meet the strict performance criteria of mission-critical applications and react to changing conditions.

3 EVALUATION

To demonstrate the efficacy of our approach, we evaluated FMs with varying complexities: a *Small* FM with 4,096 valid configurations and a *Medium* FM with 139,968 valid configurations. These models were assessed against distinct hardware configurations of edge devices as detailed in Table 1.

Table 1: Edge Configurations

Edge	*GPU*	*CPU Architecture*	*Sensor*	*others*
Country Side	false	x86	A	...
Highway	true	x86, x64	A, C	...
Small City	true	x86, x64	A, B	...
Medium City	true	x86, x64, ARM	A, B, C	...
Full	true	x86, x64, ARM	A, B, C, D	...

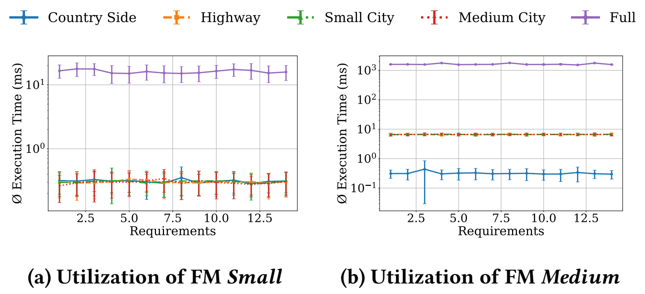

Figure 2: Average Execution Time for FMs, Combined with Varying Available Edge from Table 1 and Requirements.

To ensure accurate timing measurements, we utilized the Java Microbenchmark Harness (JMH) [4], conducting 30 iterations with three warm-up runs in *SingleShot* mode on an edge node (ThinkStation P3 equipped with an Intel Core i9-13900K processor, 13th generation, 3 GHz, and 32 GiB of RAM).

The results shown in Fig. 2 illustrate the execution times needed to compute all valid configurations per edge based on the hardware requirements each FM must meet. The execution times are plotted as lines with logarithmic scaling to highlight variations due to different requirement levels. As expected, execution times increase with the complexity of the requirements, with the longest times seen in the *full-edge* scenario, which serves as a worst-case benchmark without configuration reduction. In contrast, when specific requirements are unavailable, the number of valid configurations decreases, significantly reducing execution times—by at least 100× for the *Medium* FM scenario compared to the *full-edge* scenario, as shown in Fig. 2b. Notably, without slicing, processing took around half an hour at double the complexity, but with slicing, it was reduced to milliseconds to one second at half the complexity.

ACKNOWLEDGMENTS

This work has been funded by the German Research Foundation (DFG) within the Collaborative Research Center (CRC) 1053 MAKI and by the German Federal Ministry of Education and Research (BMBF) – 01IS17050.

REFERENCES

[1] Don Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Software Product Lines*, Henk Obbink and Klaus Pohl (Eds.). Springer, Berlin, Heidelberg, 7–20. https://doi.org/10.1007/11554844_3
[2] Angel Cañete, Mercedes Amor, and Lidia Fuentes. 2021. Energy-Efficient Deployment of IoT Applications in Edge-Based Infrastructures: A Software Product Line Approach. *IEEE Internet of Things Journal* 8, 22 (Nov. 2021), 16427–16439. https://doi.org/10.1109/JIOT.2020.3030197
[3] Batyr Charyyev, Engin Arslan, and Mehmet Hadi Gunes. 2020. Latency Comparison of Cloud Datacenters and Edge Servers. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. IEEE, Taipei, Taiwan, 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322406
[4] OpenJDK Jakob Jenkov. 2024. *Java Microbenchmark Harness (JMH)*. Retrieved September 13, 2024 from https://github.com/openjdk/jmh
[5] Sparsh Mittal. 2016. A Survey of Techniques for Approximate Computing. *Comput. Surveys* 48, 4 (May 2016), 1–33. https://doi.org/10.1145/2893356
[6] Parul Pandey and Dario Pompili. 2016. MobiDiC: Exploiting the Untapped Potential of Mobile Distributed Computing via Approximation. *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)* 38 (2016), 1–9. https://doi.org/10.1109/PERCOM.2016.7456515 Publisher: IEEE.
[7] Artois Universit and CNRS. 2024. *Sat4j*. Retrieved September 13, 2024 from http://www.sat4j.org/