How Configurable is the Linux Kernel? Analyzing Two Decades of Feature-Model History – RCR Report

ELIAS KUITER, University of Magdeburg, Germany
CHICO SUNDERMANN, TU Braunschweig, Germany
THOMAS THÜM, TU Braunschweig, Germany
TOBIAS HESS, University of Ulm, Germany
SEBASTIAN KRIETER, TU Braunschweig, Germany
GUNTER SAAKE, University of Magdeburg, Germany

This is the RCR report accompanying our TOSEM'25 paper *How Configurable is the Linux Kernel? Analyzing Two Decades of Feature-Model History*. In this report, we bundle all data relevant to our paper for the purpose of reproducibility and long-term archival. This includes the feature-model extraction tool TORTE, as well as a comprehensive feature-model dataset and experimental results.

CCS Concepts: • **Software and its engineering** → **Software product lines**; *Software evolution*; • **Theory of computation** → *Automated reasoning*.

Additional Key Words and Phrases: Linux kernel, feature modeling, software product lines, software variability

ACM Reference Format:
Elias Kuiter, Chico Sundermann, Thomas Thüm, Tobias Heß, Sebastian Krieter, and Gunter Saake. 2025. How Configurable is the Linux Kernel? Analyzing Two Decades of Feature-Model History – RCR Report. 1, 1 (August 2025), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Overview

Today, the operating system Linux is widely used in diverse environments, as its kernel can be configured flexibly. In many configurable systems, managing such variability can be facilitated in all development phases with product-line analyses. These analyses often require knowledge about the system's features and their dependencies, which are documented in a feature model. Despite their potential, product-line analyses are rarely applied to the Linux kernel in practice, as the Linux kernel's feature model continues to pose challenges for scalable and accurate analysis. Unfortunately, these challenges also severely limit our knowledge about two fundamental metrics of the kernel's configurability, namely its number of features and configurations. We identify four key limitations in the literature related to the scalability, accuracy, and influence factors of these metrics, and, by extension, other product-line analyses: (1) Analysis results for the Linux kernel are not comparable, because relevant information is not reported; (2) there is no consensus on how to define features in Linux, which leads to flawed analysis results; (3) only few versions of the Linux kernel have ever been analyzed, which are all outdated; and (4) the kernel is perceived as complex, although we lack empirical evidence that supports this claim.

Authors' Contact Information: Elias Kuiter, kuiter@ovgu.de, University of Magdeburg, Germany; Chico Sundermann, chico.sundermann@tu-braunschweig.de, TU Braunschweig, Germany; Thomas Thüm, thomas.thuem@tu-braunschweig.de, TU Braunschweig, Germany; Tobias Heß, tobias.hess@uni-ulm.de, University of Ulm, Germany; Sebastian Krieter, sebastian.krieter@tu-braunschweig.de, TU Braunschweig, Germany; Gunter Saake, saake@ovgu.de, University of Magdeburg, Germany.

2025. Manuscript submitted to ACM

Table 1. Artifacts contributed in our TOSEM paper [2].

#	Artifact	Location	License
1.	Feature-Model Extraction Tool TORTE	https://zenodo.org/doi/10.5281/zenodo.8190055 (*latest version:* https://github.com/ekuiter/torte)	LGPL v3
2.	Feature-Model Dataset + Experimental Results	https://zenodo.org/doi/10.5281/zenodo.8190055	LGPL v3

In the TOSEM paper associated with this report [2], we address these limitations with a comprehensive, empirical study of the Linux kernel's configurability, which spans its feature model's entire history from 2002 to 2024. We address the above limitations as follows: (1) We characterize parameters that are relevant when reporting analysis results; (2) we propose and evaluate a novel definition of features in Linux as a standardization effort; (3) we contribute **TORTE, a tool that analyzes arbitrary versions of the Linux kernel's feature model**; and (4) we investigate the current and possible future configurability of the kernel **on more than 3,000 feature-model versions**. Based on our results, we highlight eleven major insights into the Linux kernel's configurability and make seven actionable recommendations for researchers and practitioners.

2 Artifacts

2.1 Feature-Model Extraction Tool TORTE

To make the feature model of Linux more amenable to analysis, we contribute TORTE, a Docker-based tool that reproducibly automates extraction, transformation, and analysis of feature models. Our tool allows us to publish the first consistent, two-decade-long history of the feature model of Linux, including recent releases (see Artifact 2). To the best of our knowledge, this is the first feature-model history of this large timescale, which enables evolutionary analyses on the kernel for the first time. In addition, our tool supports arbitrary source trees, revisions, and architectures of Linux, which allows for more tailored analysis needs.

Our tool TORTE is publicly available on our GitHub page and archived on Zenodo (cf. Table 1). It comes with extensive documentation in a `README` file. Here, we quickly recount its basic usage.

Prerequisites. Our tool requires Docker to be set up, as well as some standard Unix tools (i.e., Git, curl, bash, coreutils, make, grep, and sed). If any tool is missing, TORTE will report this and not fail silently. Because TORTE contains precompiled solver binaries, the CPU architecture of the running machine matters. For most machines this will not be an issue, as their architecture is typically `amd64`. By means of Docker's platform feature, we also provide experimental support for `arm64`, which notably applies to Apple Silicon devices (e.g., MacBooks).[1]

Setup. On a typical Linux system with Docker, the following command suffices to execute the default experiment:

```
curl -s https://ekuiter.github.io/torte/ | sh
```

This will extract a single feature model (of the BusyBox system, not of Linux) and analyze it, storing all results in the `output` directory. For more comprehensive and system-specific installation instructions (including how to set up Docker on Windows and macOS), we refer to the `README` file. This file also contains a list of alternative experiments to execute, some of which concern the Linux kernel.

[1]This relies on QEMU-based emulation rather than recompilation, enabling `amd64` binaries to run on `arm64` hosts with some performance overhead and no need for recompilation. We opt not to recompile solvers from source code, which is not available for many historic solvers; and if it is, the provided toolchains are often brittle and make recompilation challenging [1]. More details on this are listed in the `README` file.

2.2 Feature-Model Dataset and Experimental Results

To improve our understanding of the kernel's configurability, we perform a large-scale evaluation of two configurability metrics (i.e., the number of features and configurations) on more than 3,000 feature models of Linux. We fully publish our evaluation in form of Artifact 2. In our evaluation, we pose and answer five research questions to investigate the current and possible future configurability of Linux.

Artifact Structure. The artifact contains numerous directories (referred to as "stages" in TORTE). Each directory contains a CSV file with measurements, several log files, and any files generated in this stage. The most important directories are:

```
kconfig/           raw feature models of the Linux kernel (i.e., non-CNF formulas)
dimacs/            DIMACS files (i.e., CNF formulas) of the Linux kernel's feature model
backbone-dimacs/   DIMACS files in which the backbone is explicitly encoded
model_to_uvl_featureide/   feature models of the Linux kernel in the standard UVL format
```

Steps to Reproduce. Our tool TORTE ships with the experiment[2] and data-analysis script[3] that we executed verbatim for obtaining the results and (interactive) diagrams in our paper. To fully recreate Artifact 2 from scratch, TORTE can be run with the following command:

```
curl -s https://ekuiter.github.io/torte/ | sh -s - linux-history-releases
```

However, we do not recommend this, as this experiment is time- and space-intensive.[4] To check the general functionality of TORTE, we instead recommend to modify the experiment to analyze fewer revisions architectures. For example, the following command will only extract, transform, and analyze a single recent revision (`v6.7`) of the Linux kernel:

```
curl -s https://ekuiter.github.io/torte/ | sh -s - linux-recent-release
```

Based on the final artifact generated by the `linux-history-releases` experiment, we also provide the Jupyter notebook we use for interactive data analysis and visualization. This script has been used to generate all the figures in the paper.

3 Badge Application

We apply for the following ACM artifact badges, with reasoning attached:

- **Artifacts Available v1.1** We fulfill the requirements for this badge by sharing all content publicly and openly (LGPL v3) on the long-term Zenodo archive with a stable DOI. Moreover, we maintain an up-to-date version of our tool on GitHub.
- **Artifacts Evaluated – Functional v1.1**
  - **Documented** Our tool repository has an extensive `README` file with all relevant instructions and information on the extraction, transformation, and analysis facilities.
  - **Consistent** All content shared in this report contributes to the paper in a major way, as the paper is inherently tied to the contributed tool and dataset.
  - **Complete** We publish the full dataset and software used in the paper.

[2] https://github.com/ekuiter/torte/blob/main/experiments/linux-history-releases.sh
[3] https://github.com/ekuiter/torte/blob/main/experiments/linux-history-releases.ipynb
[4] The extraction phase alone takes about one week and 170GiB free disk space to run. For an unattended run of the full evaluation, we recommend to use a machine with 1TiB of RAM and a 500GiB tmpfs ramdisk, as the experiment depends a lot on parallelization.

  - **Exercisable** The dataset shared in this report can be regenerated with the contributed tool on any machine fulfilling the specifications listed above.
- **Artifacts Evaluated – Reusable v1.1** Beyond the *Functional* badge, we believe that the shared artifacts can be reused in many contexts: Our tool torte is a general-purpose tool for analyzing feature models specified in the Kconfig language. It can be applied to a variety of product lines, and has already been reused in a number of different experiments. Moreover, our dataset includes thousands of feature models for the Linux kernel, which were not available before. This dataset can be used to analyze the evolution of the Linux kernel's feature model.

References

[1] Armin Biere, Mathias Fleury, Nils Froleyks, and Marijn J. H. Heule. 2023. The SAT Museum. In *Proc. Pragmatics of SAT Workshop (POS)*. 72–87.

[2] Elias Kuiter, Chico Sundermann, Thomas Thüm, Tobias Heß, Sebastian Krieter, and Gunter Saake. 2025. How Configurable is the Linux Kernel? Analyzing Two Decades of Feature-Model History. *Trans. on Software Engineering and Methodology (TOSEM)* (2025). doi:10.1145/3729423 To appear.