Performance Prediction of Cyber-Physical Systems Product Lines in Dynamic Environments

Marco Wijaya
SnT - University of Luxembourg
Luxembourg, Luxembourg
marco.wijaya@uni.lu

Sami Lazreg
SnT - University of Luxembourg
Luxembourg, Luxembourg
sami.lazreg@uni.lu

Tagir Fabarisov
SnT - University of Luxembourg
Luxembourg, Luxembourg
tagir.fabarisov@uni.lu

Andreas Hein
SnT - University of Luxembourg
Luxembourg, Luxembourg
andreas.hein@uni.lu

Maxime Cordy
SnT - University of Luxembourg
Luxembourg, Luxembourg
maxime.cordy@uni.lu

Abstract

Evaluating performance properties across highly configurable Cyber-Physical Systems (CPS) remains a significant challenge due to the size of configuration spaces and the cost of domain-specific simulations. Computing the fitness of a configuration is itself complex, as CPS behaviors depend on non-linear, system-level interactions that are difficult to model and predict. Although sampling and prediction techniques have shown strong potential in software domains, their extension to CPS contexts involving physical simulations remains largely unexplored. In this work, we highlight this challenge through a CubeSat product line case study and provide preliminary results combining simulation and machine learning to predict system-level metrics. Our findings, based on baseline prediction accuracy, illustrate the difficulty of the problem and motivate further research on performance prediction methods for CPS product lines.

ACM Reference Format:
Marco Wijaya, Sami Lazreg, Tagir Fabarisov, Andreas Hein, and Maxime Cordy. 2025. Performance Prediction of Cyber-Physical Systems Product Lines in Dynamic Environments. In *29th ACM International Systems and Software Product Line Conference - Volume A (SPLC-A '25), September 01–05, 2025, A Coruña, Spain.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3744915.3748476

1 Context, Motivation, and Background

The automated configuration and performance evaluation of CPS product lines (CPS-PL) remains a persistent challenge due to the size and complexity of their configuration spaces. Highly configurable systems, ranging from various components such as software, electronics, and mechanical components, require efficient strategies to explore variability while keeping computational costs tractable. Exhaustive evaluation across the entire configuration space is impractical in real-world scenarios due to the combinatorial explosion of valid configurations. To address this, previous research has mainly explored sampling techniques, performance prediction methods, and configuration-space learning, often leveraging Machine Learning (ML), constraint solving, and coverage-driven heuristics to approximate system behavior from limited observations [5].

Most of these works have achieved strong results in the context of software product lines, focusing on attributes such as execution time, memory consumption, or throughput [20, 21]. Typically, software systems offer relatively lightweight and static evaluation environments, where performance data can be gathered rapidly, configuration variables are discrete or binary, and the relationships between configuration options and performance outcomes are often less affected by external factors [26]. Several background studies illustrate the evolution of performance prediction for configurable software systems. In [27], the authors investigated regression models such as CART, Bagging, Random Forest, and Support Vector Machines across six real-world systems. Although their findings showed promising predictive performance, the studies remained focused on binary configuration spaces and lightweight software execution metrics. Similarly, [10] proposed DECART, a refinement of regression tree learning that incrementally selects configurations to optimize learning efficiency. While DECART improved predictive accuracy in software settings, it focused on lightweight performance properties and did not address the additional complexity introduced by physical simulations. The study in [1] analyzed different sampling strategies for performance prediction in the x264 video encoder. Although the analysis remained focused on software performance metrics, without tackling CPS-specific dynamics. Finally, [16] compared ten sampling algorithms for fault detection efficiency in configurable software systems. The presented findings focused on software system properties.

While sampling and learning techniques have shown strong results in configurable software systems [7, 21], their direct application to CPS-PL remains limited. CPS performance properties depend on continuous, time-dependent physical phenomena and dynamic environmental interactions [9, 15], which are difficult to capture through configuration parameters alone. For example, evaluating a satellite configuration involves simulating thermal dissipation, battery dynamics, and orbital mechanics—non-linear behaviors requiring costly, domain-specific simulations. These characteristics introduce scaling and generalization issues that software-oriented methods are not designed to handle [30]. In this work, we highlight these challenges through a CubeSat case study, using limited simulation data to predict system-level metrics such as mass and thermal

This work is licensed under a Creative Commons Attribution 4.0 International License.
SPLC-A '25, A Coruña, Spain
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2024-6/25/09
https://doi.org/10.1145/3744915.3748476

load. Our findings aim to illustrate the difficulty of performance prediction in CPS-PL and motivate the development of scalable, CPS-specific prediction methods. This short paper presents baseline results toward that goal, emphasizing the need for methods that can effectively support CPS-PL evaluation under realistic physical and environmental constraints.

2 Preliminary Study

This section introduces the challenges of performance evaluation in CPS-PL, illustrates them through a CubeSat case study, and motivates the preliminary methodology and research questions.

2.1 Challenges in CPS-PL Evaluation

CPS-PL manages variability across families of embedded systems interacting with physical environments. Domains such as automotive, robotics, and space systems involve selecting and tuning subsystems to meet mission-specific requirements. Performance evaluation in CPS-PL is challenging due to large, combinatorial configuration spaces [8] and the influence of dynamic environmental conditions. Metrics such as mass, thermal stability, and energy margins stem from complex, interdependent physical phenomena, often requiring costly simulations [25]. Unlike software product lines, where automated testing is affordable, simulating large CPS configuration sets is time- and resource-intensive. This often limits exploration, slows feasibility studies, and reinforces reliance on known architectures [2].

This study explores predictive modeling to reduce simulation effort. We focus on two questions:

RQ1: Can short-duration simulations support accurate prediction of long-mission performance?

RQ2: How does training sample size affect prediction accuracy and ranking reliability[1]?

2.2 Subject System: CubeSat Product Line

We consider a CubeSat-based communication platform as a representative CPS-PL, characterized by tight integration of embedded computation, energy-constrained operation, environmental coupling, and physical variability [4, 13, 22]. The product line encompasses diverse configurations tailored to different mission requirements, with variability across critical subsystems such as:

- **Communication:** different frequency bands (e.g., VHF, X-band), antenna types, and data rates;
- **On-Board Computing (OBC):** processors with different power/performance profiles and buffer capacities;
- **Energy Subsystems:** combinations of solar array technologies, battery chemistries, and radiator materials;
- **Payload:** data acquisition capabilities depending on Point-of-Interest (POI) detection performance;
- **Thermal Control:** passive regulation via radiators with distinct surface properties.

The configuration space (Fig. 1) includes both discrete architectural options and continuous parameter ranges [14, 29].

[1]Degree to which the predicted ranking of CubeSat configurations matches the ranking of the 15-orbit ground truth.

Simulation Environment. To support system-level evaluation, we developed a domain-specific simulator inspired by NASA's CADRE architecture [12]. Given a CubeSat configuration, the simulator performs time-step simulations of a full orbital mission, modeling:

- **Payload acquisition:** data is generated when the satellite is over a POI;
- **OBC processing:** raw data is buffered, processed and staged for downlink;
- **Communication:** data is downloaded during ground station (GS) passes;
- **Power system dynamics:** solar power generation, battery charging/discharging, power draw breakdown;
- **Thermal behavior:** heat generation, radiation, and environmental exchange;

The dashboards in Fig. 2 provide full visibility into these telemetry variables during the simulation. Although structurally similar to a digital twin, this simulator is not intended as a full digital-twin representation, but rather as a domain-specific abstraction for performance evaluation within a constrained modeling scope. The simulation represents some CubeSat's metrics evolution during operation (from upper left to right), such as state-of-charge of battery, temperature profile, accumulative data throughput, communication data rates, power generated & consumed, payload point of view from ground station (POI-GS), heat generated by components, and sunlight/eclipse timeline.

Score Computation. Each simulation outputs a normalized mission score, reflecting the CubeSat's ability to carry out a data-driven mission under realistic constraints. The final score aggregates five subsystem-specific scores. Each domain score encodes multi-criteria performance (e.g., thermal stability, power availability, buffer efficiency) and is derived from telemetry traces. For instance, battery state-of-charge (SoC) in Fig. 2 determines the score of power domain by assessing whether the minimum SoC is lower than threshold or not. The radar chart in Fig. 4 illustrates a sample configuration's subscore profile.

2.3 Experimental Setup

The evaluation dataset includes 3,500 valid CubeSat configurations, systematically sampled[17] from a full configuration space of about 350,000 combinations. Each configuration was simulated under three mission durations:

- 4 orbital revolutions (short mission, approximately 6 hours),
- 8 orbital revolutions (medium mission, approximately 12 hours),
- 15 orbital revolutions (full mission day, approximately 24 hours).

The objective is to evaluate how effectively models trained on configurations simulated over short or medium mission durations can estimate the behavior of those same configurations over longer missions. This reflects the practical constraints of product line engineering, where exhaustive simulation of all variants under full mission conditions is infeasible.

Experimental campaigns were structured along two independent dimensions:

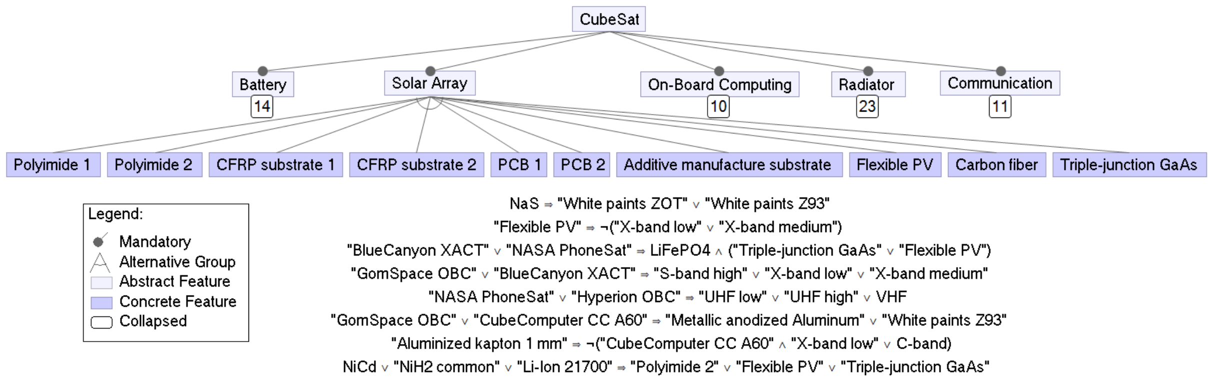
CubeSat
Battery
14
Solar Array
On-Board Computing
10
Radiator
23
Communication
11
Polyimide 1
Polyimide 2
CFRP substrate 1
CFRP substrate 2
PCB 1
PCB 2
Additive manufacture substrate
Flexible PV
Carbon fiber
Triple-junction GaAs
Legend:
Mandatory
Alternative Group
Abstract Feature
Concrete Feature
Collapsed
NaS ⇒ "White paints ZOT" ∨ "White paints Z93"
"Flexible PV" ⇒ ¬("X-band low" ∨ "X-band medium")
"BlueCanyon XACT" ∨ "NASA PhoneSat" ⇒ LiFePO4 ∧ ("Triple-junction GaAs" ∨ "Flexible PV")
"GomSpace OBC" ∨ "BlueCanyon XACT" ⇒ "S-band high" ∨ "X-band low" ∨ "X-band medium"
"NASA PhoneSat" ∨ "Hyperion OBC" ⇒ "UHF low" ∨ "UHF high" ∨ VHF
"GomSpace OBC" ∨ "CubeComputer CC A60" ⇒ "Metallic anodized Aluminum" ∨ "White paints Z93"
"Aluminized kapton 1 mm" ⇒ ¬("CubeComputer CC A60" ∧ "X-band low" ∨ C-band)
NiCd ∨ "NiH2 common" ∨ "Li-Ion 21700" ⇒ "Polyimide 2" ∨ "Flexible PV" ∨ "Triple-junction GaAs"

Figure 1: Feature Model of the CubeSat product line.

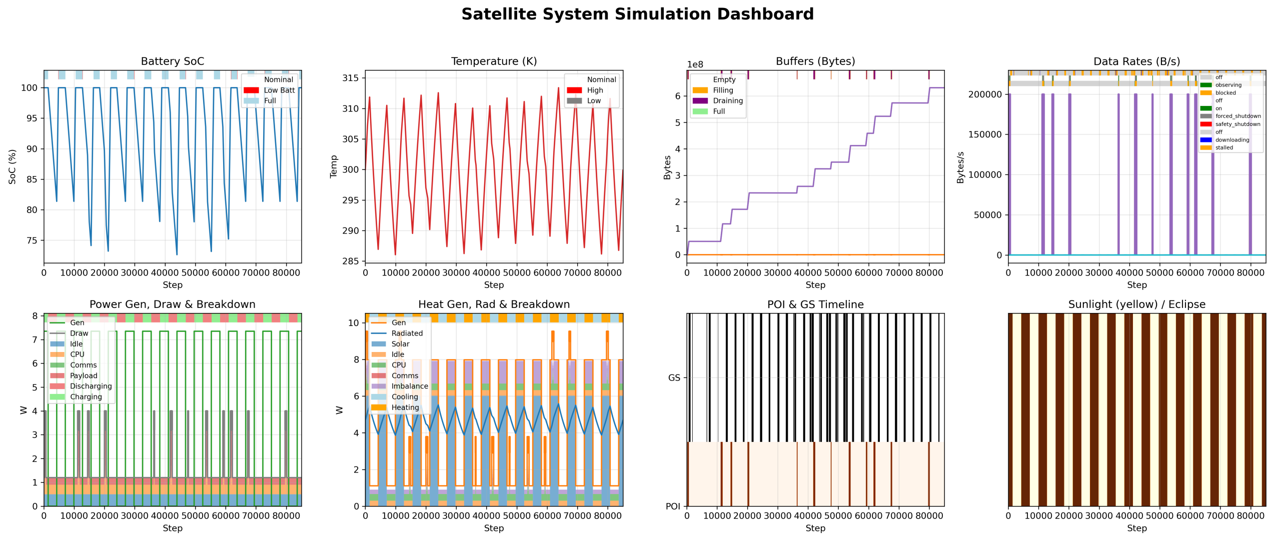

Figure 2: Telemetry dashboards from simulation of a CubeSat's configuration sample.

- **Sample Size:** Three different training set sizes were considered—small (35 configurations), medium (350 configurations), and large (3,500 configurations).
- **Simulation Duration:** Simulations for 4, 8, and 15 orbital revolutions provided multiple evaluation horizons for analyzing prediction generalization.

For each combination of sample size and mission duration, subsets were randomly drawn and and split into 80% for training and 20% for testing. These subsets were used to train and evaluate machine learning models, including regression trees, random forests, and linear regressors. Predictive performance was assessed using classical regression metrics (MSE, RMSE, R^2) and rank-based metrics (Spearman's ρ), with a specific focus on how well models preserved the relative ordering of configurations compared to the 15-orbit simulation results. The predicted performances are configuration scores that are derived from subsystem dynamics (to be detailed later in Section 3).

The goal of prediction is not to estimate the exact performance value, but to identify promising product configurations efficiently. In the context of CPS-PL engineering, relative ranking is sufficient to support early-stage selection of candidate variants for further analysis, significantly reducing the simulation effort required across large configuration spaces. *Availability.* All datasets, simulation outputs, and analysis scripts used in this study will be made publicly available in a companion repository to facilitate reproducibility and foster further research.[2]

3 Results and discussion

Table 1 reports the predictive performance of three regression models—regression trees, random forests, and linear regression—evaluated across varying sample sizes and simulation durations. Each cell presents the mean and standard deviation across ten randomized trials, measuring model performance using four key

[2]https://github.com/marcowijaya001/CubeSatDynamicEnv

Table 1: Performance of Different Models Across Sample Sizes and Simulation Durations (mean ± std)

Sample	Simulation	Model	MSE	RMSE	R^2	ρ
Small (35)	Short (4 orbits)	Regression Tree	1.7008 ± 0.6962	1.2735 ± 0.2813	-0.9622 ± 1.4542	0.2680 ± 0.3880
		Random Forest	0.8173 ± 0.3331	0.8812 ± 0.2019	0.1319 ± 0.5821	0.5857 ± 0.3625
		Linear Regression	940.4120 ± 1641.8401	21.6410 ± 21.7274	-1380.6243 ± 2956.9776	-0.0357 ± 0.4866
	Mid (8 orbits)	Regression Tree	1.2291 ± 0.6535	1.0597 ± 0.3258	-1.0853 ± 2.1418	0.3858 ± 0.3239
		Random Forest	0.7733 ± 0.2695	0.8601 ± 0.1830	-0.2721 ± 1.2258	0.6357 ± 0.2870
		Linear Regression	1040.0509 ± 1924.3486	21.9577 ± 23.6202	-3286.1770 ± 8075.4151	-0.0179 ± 0.4616
	Full (15 orbits)	Regression Tree	1.2825 ± 0.7463	1.0544 ± 0.4131	-2.8635 ± 5.2311	0.4852 ± 0.2689
		Random Forest	0.9009 ± 0.4418	0.9067 ± 0.2806	-1.7688 ± 4.3767	0.5821 ± 0.1779
		Linear Regression	939.4106 ± 1758.8759	20.6649 ± 22.6357	-6393.6271 ± 17278.2610	-0.0179 ± 0.3903
Medium (350)	Short (4 orbits)	Regression Tree	0.8311 ± 0.2660	0.9000 ± 0.1450	0.4655 ± 0.1909	0.7364 ± 0.0647
		Random Forest	0.3874 ± 0.0948	0.6178 ± 0.0754	0.7541 ± 0.0552	0.8274 ± 0.0269
		Linear Regression	0.9345 ± 0.1755	0.9622 ± 0.0928	0.4113 ± 0.0707	0.6373 ± 0.0553
	Mid (8 orbits)	Regression Tree	0.5831 ± 0.1599	0.7554 ± 0.1115	0.5539 ± 0.1525	0.7685 ± 0.0701
		Random Forest	0.3297 ± 0.0971	0.5682 ± 0.0829	0.7515 ± 0.0660	0.8331 ± 0.0293
		Linear Regression	0.8612 ± 0.1661	0.9237 ± 0.0897	0.3562 ± 0.0668	0.6322 ± 0.0687
	Full (15 orbits)	Regression Tree	0.5709 ± 0.2338	0.7388 ± 0.1585	0.5031 ± 0.2652	0.7790 ± 0.0842
		Random Forest	0.3247 ± 0.1006	0.5629 ± 0.0883	0.7283 ± 0.0853	0.8288 ± 0.0334
		Linear Regression	0.8596 ± 0.1738	0.9223 ± 0.0943	0.2911 ± 0.0918	0.6117 ± 0.0847
Large (3500)	Short (4 orbits)	Regression Tree	0.1408 ± 0.0147	0.3747 ± 0.0195	0.9152 ± 0.0090	0.9004 ± 0.0072
		Random Forest	0.0601 ± 0.0043	0.2451 ± 0.0089	0.9637 ± 0.0030	0.9224 ± 0.0055
		Linear Regression	0.8375 ± 0.0409	0.9149 ± 0.0225	0.4956 ± 0.0223	0.6644 ± 0.0119
	Mid (8 orbits)	Regression Tree	0.1392 ± 0.0266	0.3715 ± 0.0339	0.9027 ± 0.0189	0.9202 ± 0.0108
		Random Forest	0.0569 ± 0.0061	0.2382 ± 0.0127	0.9603 ± 0.0043	0.9445 ± 0.0065
		Linear Regression	0.7848 ± 0.0388	0.8856 ± 0.0219	0.4525 ± 0.0206	0.6660 ± 0.0148
	Full (15 orbits)	Regression Tree	0.1116 ± 0.0278	0.3314 ± 0.0422	0.9170 ± 0.0179	0.9358 ± 0.0206
		Random Forest	0.0570 ± 0.0094	0.2380 ± 0.0201	0.9574 ± 0.0062	0.9680 ± 0.0065
		Linear Regression	0.8083 ± 0.0445	0.8987 ± 0.0249	0.3946 ± 0.0210	0.6500 ± 0.0186

metrics: mean squared error (MSE) [28], root mean squared error (RMSE) [11], coefficient of determination (R^2) [18], and Spearman's rank correlation (ρ) [23]. Since the simulation environment was developed by the authors, the predictive models are evaluated on a system that shares assumptions with the simulator, which may limit generalizability. This constitutes a potential threat to validity and will be addressed in future evaluations with independent high-fidelity simulators.

Effect of Sample Size. Across all simulation durations, increasing the training sample size consistently improves predictive performance (Fig. 5). With only 35 configurations, models—especially linear regression—show erratic and unreliable behavior, with $R^2 < 0$ in many cases. Random forests, however, already provide modest ranking reliability in this regime (e.g., $\rho = 0.59$ for short missions), even though generalization remains weak. As sample size increases to 350, prediction becomes significantly more stable: random forests reach $\rho = 0.83$ and $R^2 = 0.75$ on short missions, and perform similarly well on full (15-orbit) simulations. At the largest sample size (3,500), both random forests and regression trees yield strong generalization, with $R^2 > 0.92$ and $\rho > 0.94$ across all durations. These results show a clear transition between poor ($R^2 < 0$) and usable ($R^2 > 0.5$) prediction around 350 samples.

Effect of Simulation Duration. Interestingly, the prediction accuracy does not deteriorate significantly when training is based on short or mid-duration simulations (4 or 8 orbits) and evaluated on full-duration outcomes. For instance, at the medium sample size, random forests trained on 8-orbit data achieve $\rho = 0.83$ and $R^2 = 0.75$ on 15-orbit ground truth, closely matching full-duration training. This confirms that early-stage, lower-cost simulations can support sufficient ranking reliability of configurations over longer missions.

Model Comparison. Random forests consistently outperform other models across all settings. Their ability to capture non-linear interactions and hierarchical variability across subsystems makes them particularly well suited for CPS-PL configuration spaces. Regression trees provide a lightweight alternative with reasonably good ranking performance, though their absolute accuracy is lower. Linear regression fails to provide reliable estimates, especially in the small and medium sample regimes, where feature interactions dominate. *Ranking vs. Prediction Accuracy.* The primary objective of this study is not to precisely estimate performance scores, but rather to **rank configurations effectively** in order to guide selection and refinement within a large CPS-PL. From this perspective, Spearman's ρ provides the most relevant signal. Our results demonstrate that rank-preserving predictions are achievable with limited simulation effort, especially when using non-linear models like random forests. This enables early identification of promising configurations without exhaustive evaluation of the entire variability space. Moreover, feature importance (Fig. 3) shows how much attributes of components contribute to the configuration ranking. For example, a CubeSat with thoroughly selected radiator's emissivity would significantly impacts system performance compared to other attributes.

4 Conclusion and Future Plans

This study addressed the problem of performance prediction in CPS-PL. In such systems, exhaustive simulation of the configuration space is infeasible due to prohibitive computational costs. A case study was conducted on a CubeSat-based communication platform. A simulation pipeline was used to compute mass and thermal performance metrics across a large number of configurations. The results show that regression-based models (specifically, random forest) can provide interesting performance predictions and reliable rankings when trained on sufficiently large and uniformly sampled datasets. Even with a streamlined simulator, evaluating 3,500 configurations requires several hours of computation on a standard workstation. In industrial settings, this limitation becomes more severe. High-fidelity simulation tools (e.g., AGI STK, ESATAN-TMS, and Dymola) may take several days or even weeks to simulate a few hundred configurations [3]. These constraints highlight the need for more efficient evaluation methods. The goal is to reduce simulation effort while maintaining or improving prediction quality. The preliminary results indicate that exact performance prediction may not be required in early-stage configuration selection. This is often the case during feasibility studies or when preparing responses to Requests for Quotation. In such scenarios, the ability to rank candidate configurations with high reliability is often sufficient to support early decision-making and prototyping. Future work will

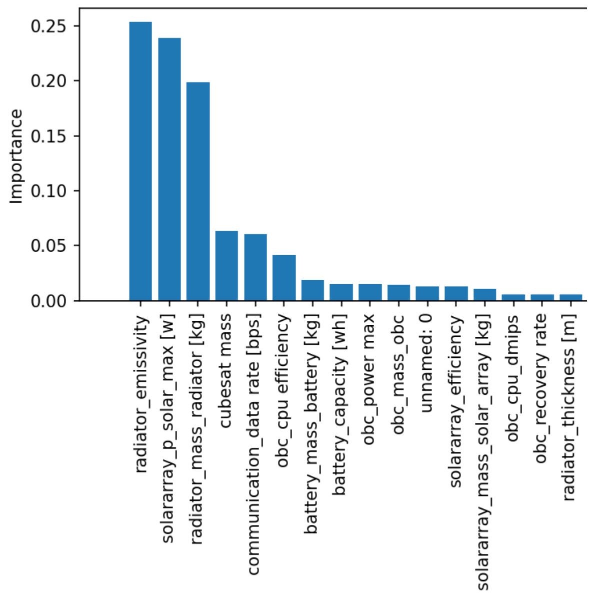

Figure 3: Feature importance (regression tree) for full-mission performance prediction.

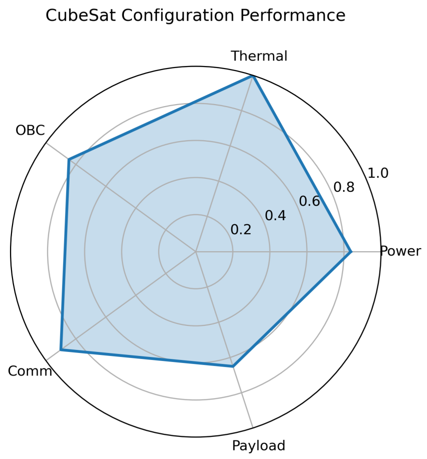

Figure 4: Subsystem performance radar chart used to compute the final mission score.

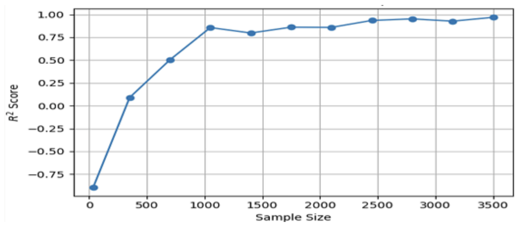

Figure 5: Prediction accuracy (R^2) vs sample size for full-duration simulations (regression tree).

therefore focus on ranking-oriented predictors. This includes the application of learning-to-rank algorithms [6], preference-based active sampling, and rank-aware surrogate modeling [19]. The feature importance analysis reveals that radiator-related variables have a disproportionate impact on prediction outcomes. This confirms the previously identified challenge in CPS performance influence modeling: external environmental factors can introduce abrupt behavioral boundaries, complicating both regression and ranking tasks.

Further work will extend the methodology toward bridging predictive modeling, fault detection, and behaviorally-aware learning. Planned developments include feature interaction analysis, improved sampling strategies, and the incorporation of behavioral modeling techniques [24], such as time-series architectures capable of capturing telemetry dynamics. The objective is to support configuration selection in CPS-PL by enabling predictive models that account for both performance and operational robustness. This would allow for more informed and efficient decisions in mission-driven systems subject to dynamic environment.

This work is supported by the Luxembourg National Research Funds (FNR) through the project grants INTER/FNRS/20/15077233/Scaling Up Variability/C23/IS/18177547/VARIANCE, C22/IS/17395798/SAMP.

References

[1] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, and Jean-Marc Jézéquel. 2020. Sampling effect on performance prediction of configurable systems: A case study. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 277–288.

[2] Liang Bao, Xin Liu, Ziheng Xu, and Baoyin Fang. 2018. Autoconfig: Automatic configuration tuning for distributed message systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 29–40.

[3] Salvatore Borgia, Francesco Topputo, et al. 2024. Multiphysics Acausal Modeling and Simulation of Satellites Using Modelica Library. In *LINKÖPING ELECTRONIC CONFERENCE PROCEEDINGS*, Vol. 207. Modelica Association and Linköping University Electronic Press, 157–169.

[4] Justin M Bradley and Ella M Atkins. 2015. Coupled cyber–physical system modeling and coregulation of a cubesat. *IEEE Transactions on Robotics* 31, 2 (2015), 443–456.

[5] Yuntianyi Chen, Yongfeng Gu, Lulu He, and Jifeng Xuan. 2019. Regression models for performance ranking of configurable systems: A comparative study. In *International Workshop on Structured Object-Oriented Formal Language and Method*. Springer, 243–258.

[6] Romain Deveaud, Josiane Mothe, and Jian-Yun Nie. 2016. Learning to rank system configurations. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. 2001–2004.

[7] Jingzhi Gong and Tao Chen. 2024. Deep configuration performance learning: A systematic survey and taxonomy. *ACM Transactions on Software Engineering and Methodology* 34, 1 (2024), 1–62.

[8] Alexander Grebhahn, Carmen Rodrigo, Norbert Siegmund, Francisco J Gaspar, and Sven Apel. 2017. Performance-influence models of multigrid methods: A case study on triangular grids. *Concurrency and Computation: Practice and Experience* 29, 17 (2017), e4057.

[9] Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, and Andrzej Wąsowski. 2013. Variability-aware performance prediction: A statistical learning approach. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 301–311.

[10] Jianmei Guo, Dingyu Yang, Norbert Siegmund, Sven Apel, Atrisha Sarkar, Pavel Valov, Krzysztof Czarnecki, Andrzej Wasowski, and Huiqun Yu. 2018. Data-efficient performance learning for configurable systems. *Empirical Software Engineering* 23 (2018), 1826–1867.

[11] Timothy O Hodson. 2022. Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not. *Geoscientific Model Development Discussions* 2022 (2022), 1–10.

[12] John T Hwang, Dae Young Lee, James W Cutler, and Joaquim RRA Martins. 2014. Large-scale multidisciplinary optimization of a small satellite's design and operation. *Journal of Spacecraft and Rockets* 51, 5 (2014), 1648–1663.

[13] Andrew T Klesh, James W Cutler, and Ella M Atkins. 2012. Cyber-physical challenges for space systems. In *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems*. IEEE, 45–52.

[14] Wiley J Larson and James Richard Wertz. 1999. Space mission analysis and design. (1999).

[15] Max Lillack, Johannes Müller, and Ulrich W Eisenecker. 2013. Improved prediction of non-functional properties in software product lines with domain context. In *Software Engineering 2013*. Gesellschaft für Informatik eV, 185–198.

[16] Flávio Medeiros, Christian Kästner, Márcio Ribeiro, Rohit Gheyi, and Sven Apel. 2016. A comparison of 10 sampling algorithms for configurable systems. In *Proceedings of the 38th International Conference on Software Engineering*. 643–654.

[17] Sayed A Mostafa and Ibrahim A Ahmad. 2018. Recent developments in systematic sampling: A review. *Journal of Statistical Theory and Practice* 12, 2 (2018), 290–310.

[18] Nico JD Nagelkerke et al. 1991. A note on a general definition of the coefficient of determination. *biometrika* 78, 3 (1991), 691–692.

[19] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. 2017. Using bad learners to find good configurations. In *Proceedings of the 2017 11th joint meeting on foundations of software engineering*. 257–267.

[20] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. 2018. Faster discovery of faster system configurations with spectral learning. *Automated Software Engineering* 25 (2018), 247–277.

[21] Juliana Alves Pereira, Mathieu Acher, Hugo Martin, Jean-Marc Jézéquel, Goetz Botterweck, and Anthony Ventresque. 2021. Learning software configuration spaces: A systematic literature review. *Journal of Systems and Software* 182 (2021), 111044.

[22] Hassan Reza, Christoffer Korvald, Jeremy Straub, Justin Hubber, Nicholas Alexander, and Abhinav Chawla. 2016. Toward requirements engineering of cyber-physical systems: Modeling CubeSat. In *2016 IEEE Aerospace Conference*. IEEE, 1–13.

[23] Philip Sedgwick. 2014. Spearman's rank correlation coefficient. *Bmj* 349 (2014).

[24] Shun-Yao Shih, Fan-Keng Sun, and Hung-yi Lee. 2019. Temporal pattern attention for multivariate time series forecasting. *Machine Learning* 108 (2019), 1421–1441.

[25] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. 2015. Performance-influence models for highly configurable systems. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. 284–294.

[26] Norbert Siegmund, Sergiy S Kolesnikov, Christian Kästner, Sven Apel, Don Batory, Marko Rosenmüller, and Gunter Saake. 2012. Predicting performance via automated feature-interaction detection. In *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 167–177.

[27] Pavel Valov, Jianmei Guo, and Krzysztof Czarnecki. 2015. Empirical comparison of regression methods for variability-aware performance prediction. In *Proceedings of the 19th international conference on software product line*. 186–190.

[28] Cort J Willmott and Kenji Matsuura. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate research* 30, 1 (2005), 79–82.

[29] Bruce Yost and Sasha Weston. 2024. *State-of-the-art small spacecraft technology*. Technical Report.

[30] Yi Zhang, Jianmei Guo, Eric Blais, and Krzysztof Czarnecki. 2015. Performance prediction of configurable software systems by fourier learning (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 365–373.