A Survey on the Use of Large Language Models in Model-Based Design of Cyber-Physical Systems

Minmin Feng
Virginia Tech
minminf@vt.edu

***Abstract*—Model-Based Design (MBD) is pivotal for developing complex Cyber-Physical Systems (CPS), yet faces challenges in managing intricacy and labor-intensive processes. Large Language Models (LLMs) have emerged with significant potential to revolutionize engineering practices. This survey provides a comprehensive overview of LLM applications in MBD for CPS, covering a broad scope including Kahn Process Networks, discrete-event models, UML/SysML, and specific domains like automotive. We explore LLM use-cases across the MBD lifecycle, from requirements engineering and model synthesis to verification and validation. Key challenges, such as ensuring the correctness of LLM-generated artifacts and integrating domain-specific knowledge, are discussed alongside open research problems. This work aims to map the current landscape and highlight future directions for leveraging LLMs to enhance MBD methodologies for CPS development, ultimately aiming for more efficient, reliable, and innovative system design.**

***Index Terms*—Large Language Models, Model-Based Design, Cyber-Physical Systems, Survey, Kahn Process Network, Discrete-Event Models, UML, SysML, Automotive.**

I. Introduction

Cyber-Physical Systems (CPS) integrate computation with physical processes, and Model-Based Design (MBD) is a crucial methodology for their development, offering a structured approach to manage complexity and ensure correctness [1], [2]. However, MBD for complex CPS still faces challenges, including significant manual effort, ensuring consistency across diverse models, and adapting to evolving requirements [3]. Recently, Large Language Models (LLMs) have demonstrated remarkable capabilities in natural language understanding, generation, and even code synthesis, presenting new opportunities to augment engineering workflows [4], [5].

This survey investigates the emerging applications of LLMs in the MBD of CPS. Our scope is broad, encompassing various modeling paradigms taught in relevant curricula and used in practice, such as synchronous data-flow models (e.g., Simulink), finite-state reactive models (e.g., Stateflow), discrete-event models, Kahn Process Networks, continuous-time physical domain models (e.g., Modelica), and system architecture modeling languages like UML/SysML. We also consider applications within specific domains, notably automotive systems. The primary contribution of this work is a structured overview of how LLMs can be utilized across the MBD lifecycle, from requirements engineering and model synthesis to verification, validation, and maintenance. We identify key research trends, highlight representative studies, and discuss the significant challenges and open research problems that need to be addressed for the effective and reliable adoption of LLMs in this critical field. Our literature search followed established systematic review principles to gather relevant publications from major academic databases.

This paper is organized as follows: Section II briefly introduces core concepts of MBD, CPS, and LLMs. Section III presents a taxonomy of MBD artifacts and modeling approaches relevant to LLM integration. Section IV details LLM applications across the MBD lifecycle. Section V discusses representative use cases, with a focus on domain-specific considerations. Section VI outlines the challenges and open research problems. Finally, Section VII concludes the paper and suggests future research directions. This survey draws inspiration from the layered perspective of Burns & Davis on mixed criticality systems in its chapter organization [6].

II. Preliminaries

This section provides a concise overview of Model-Based Design (MBD), Cyber-Physical Systems (CPS), and Large Language Models (LLMs) to establish the foundational context for this survey.

A. Model-Based Design (MBD)

MBD is a design methodology that emphasizes the use of formal models throughout the system development lifecycle, from requirements specification, design, and implementation to verification and validation [7], [8]. Key MBD phases typically include system requirements analysis, architectural design, detailed component modeling, simulation, automatic code generation, and continuous testing. MBD aims to improve design quality, manage complexity, facilitate early detection of errors, and enhance communication among development teams.

B. Cyber-Physical Systems (CPS)

CPS are engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components [9], [10]. They represent a confluence of cyber (computation, communication, control) and physical (mechanical, electrical, chemical) elements. CPS are prevalent in diverse sectors such as automotive, aerospace, industrial automation, healthcare, and smart grids. Designing CPS involves addressing challenges related to safety, reliability, real-time performance, security, and the interaction between discrete software logic and continuous physical dynamics.

C. Large Language Models (LLMs)

LLMs are advanced artificial intelligence models, typically based on the Transformer architecture [11], trained on vast amounts of text and code data. They exhibit remarkable capabilities in understanding and generating natural language, translating between languages, writing different kinds of creative content, and answering questions in an informative way [4], [5]. Key LLM abilities relevant to MBD include natural language understanding (NLU) for interpreting requirements, natural language generation (NLG) for creating documentation or model descriptions, and code generation from high-level specifications [12], [13]. Popular examples include the GPT series, LLaMA, and BERT [14]–[16].

D. Course-Specific Models and Corresponding Sections

This survey covers several modeling formalisms emphasized in the course curriculum. Below is a guide to where these specific models are primarily discussed in the subsequent sections:

- **Simulink:** Discussed in Section IV (Model Synthesis and Translation, p. 2) and other examples throughout the survey.
- **Stateflow:** Discussed in Section IV (Model Synthesis and Translation, p. 2) and relevant examples.
- **Hybrid Automata:** While not a dedicated subsection, concepts related to hybrid automata are implicitly covered in discussions of modeling systems with both discrete and continuous dynamics, for instance, within the scope of discrete-event models and continuous-time physical domain models (see Section I, p. 1 and Section IV, p. 2).
- **Kahn Process Networks (KPN):** Discussed in Section IV (Dataflow and Discrete-Event Model Generation, p. 2).
- **Discrete-Event Models:** Discussed in Section IV (Dataflow and Discrete-Event Model Generation, p. 2) and related examples.
- **UML/SysML:** Discussed in Section IV (Model Synthesis and Translation, p. 2) and throughout the survey as a general-purpose modeling language.

Note: Page numbers are placeholders and will be updated upon final compilation.

III. Taxonomy of MBD Artefacts and Modelling Approaches

MBD for CPS employs a variety of modeling artifacts and languages. Understanding these is key to contextualizing LLM applications. This section briefly categorizes major approaches relevant to the scope of this survey.

- **Synchronous Data-flow Models:** Used for signal processing and control systems with deterministic timing (e.g., Simulink by MathWorks [17], SCADE by ANSYS [18]). LLMs can assist in model generation from specifications or translating between these and other model types.
- **Finite-state Reactive Models:** Describe systems reacting to discrete events via state transitions (e.g., Stateflow by MathWorks [19], UML State Machines [20], Lustre). LLMs can help generate or verify state logic from textual descriptions.
- **Discrete-Event Models & Process Networks:** Simulate systems with state changes at discrete time points, often asynchronous (e.g., SimEvents by MathWorks, Ptolemy II [21]). Kahn Process Networks (KPNs) [22] model concurrent processes communicating via channels, relevant for streaming applications. LLMs could aid in scenario generation or performance analysis.
- **Continuous-Time Physical-Domain Models:** Describe physical system behavior using differential-algebraic equations (e.g., Modelica [23], Simscape by MathWorks [24]). LLMs might assist in component model generation or parameter estimation from experimental data descriptions.
- **System Architecture & Requirements Models:** Capture high-level structure, behavior, and requirements (e.g., SysML [25], AADL [26], UML for system-level views). LLMs show promise in translating natural language requirements into these architectural models [27], [28] or checking consistency.

This taxonomy highlights the diversity of models where LLMs could be applied, ranging from detailed component design to system-level architecture and requirements management.

IV. LLM Applications in the MBD Lifecycle

LLMs offer diverse capabilities to augment various stages of the MBD lifecycle for CPS. This section outlines key application areas, focusing on conciseness due to page constraints.

A. Model Synthesis and Translation

LLMs can assist in generating models from high-level descriptions and translating between different formalisms.

- **NL Requirements to Models:** LLMs can parse natural language (NL) requirements to propose initial structures in SysML, UML, or domain-specific languages. For instance, Timperley et al. [28] demonstrated LLM-driven generation of Capella system architecture models for spacecraft from requirements. This includes generating Stateflow charts or basic Simulink diagrams from precise NL descriptions.
- **Cross-Language/Paradigm Translation:** LLMs can facilitate interoperability by translating models, e.g., between Simulink and Modelica, or from SysML to Simulink/AADL, though ensuring semantic equivalence is a challenge.
- **Dataflow and Discrete-Event Model Generation:** Recent research explores LLMs for automatically generating dataflow graphs, such as for Kahn Process Networks (KPNs), and assisting in the analysis of aspects like channel buffering. Furthermore, LLMs are being investigated for generating scripts for discrete-event simulation environments like Ptolemy II and aiding in scenario-based simulations [29].

B. Model Understanding and Refactoring

LLMs can aid in comprehending and improving existing complex models.

- **Documentation and Explanation:** Generating NL summaries or detailed explanations of model components and their functionalities.
- **Requirements-Driven Analysis:** Fischer et al. [30] explored LLM-based slicing of Simulink models based on requirements, aiding debugging and impact analysis.
- **Quality Improvement:** LLMs can potentially detect model clones, inconsistencies, or suggest refactoring for style harmonization and quality assessment.

C. Code Generation and Deployment Support

While MBD tools have code generators, LLMs can offer enhancements.

- **Optimizing Auto-Generated Code:** Refining code (e.g., from Simulink Coder) based on non-functional requirements like performance or adherence to standards (e.g., MISRA C).
- **Deployment Artifacts:** Generating auxiliary scripts, configuration files, or even safety case templates and certification document drafts [31].

D. Verification, Validation, and Testing (VV&T)

LLMs can contribute to generating test artifacts and analyzing results.

- **Formal Specification Generation:** Assisting in drafting formal properties for model checking from NL requirements.
- **Test Case Generation:** Creating test inputs, scripts, or oracles for MIL/SIL/HIL testing based on models or coverage criteria.
- **Simulation Support:** Aiding in simulation scenario generation from NL and analyzing large simulation datasets to identify patterns or anomalies.

E. Model Maintenance and Evolution

LLMs can support the ongoing evolution of CPS models.

- **Legacy Model Comprehension:** Assisting engineers in understanding older models with scarce documentation.
- **Change Impact Analysis:** Helping identify parts of models or downstream artifacts affected by changes in requirements or model components.

These applications highlight LLMs as potential AI assistants, though reliability and correctness of their outputs remain critical research areas.

V. Representative Use Cases and Domain-Specific Considerations

The application of LLMs in MBD for CPS is gaining traction in various domains, each with unique demands. Due to stringent page limits, this section provides a highly condensed overview.

Automotive Domain: The automotive industry, with standards like AUTOSAR and safety requirements (ISO 26262), explores LLMs for ADAS/AV model development, validation, and managing functional safety documentation. For example, LLMs could assist in generating simulation scenarios for ADAS from textual descriptions or help trace requirements to AUTOSAR model elements.

Aerospace Domain: Characterized by extreme safety and reliability (e.g., DO-178C/DO-331), aerospace utilizes MBD for flight control and avionics. Timperley et al. [28] demonstrated LLM-assisted generative design of spacecraft system architectures using Capella, showing potential in early-stage design. LLMs might also aid in generating documentation for certification.

Other Domains: Industrial automation sees LLMs for robot task planning from NL commands or PLC program generation. Smart grids and healthcare are also emerging areas where LLMs could assist in modeling complex systems or translating clinical needs into model specifications.

In all domains, the challenge lies in adapting LLMs to specific engineering knowledge, safety standards, and verification needs. The focus is often on LLMs as assistants to domain experts rather than autonomous designers.

VI. Challenges and Open Research Problems

Despite promising applications, significant challenges must be addressed for the reliable use of LLMs in MBD for CPS [32].

- **Correctness, Reliability, and Verifiability:** Ensuring LLM-generated artifacts (models, code) are semantically correct and type-safe is paramount, especially for safety-critical systems. Mitigating LLM "hallucinations" (generating plausible but false information [33]) and formally verifying LLM outputs are major hurdles. Explainability and traceability of LLM-generated designs are also crucial for trust and debugging.
- **Integration of Domain-Specific Knowledge:** Generic LLMs often lack deep, specialized engineering knowledge. Effectively incorporating domain-specific constraints, physical laws, and safety standards (e.g., ISO 26262, DO-178C) into LLM reasoning, possibly via fine-tuning or retrieval-augmented generation (RAG), is essential [28].
- **Scalability, Usability, and Toolchain Integration:** LLM solutions must scale to industrial-size models and integrate seamlessly with existing MBD tools (e.g., Simulink, Modelica tools, SysML modelers). Designing effective human-LLM interaction paradigms for collaborative design is key, as is managing computational costs.
- **Security and Privacy:** Protecting intellectual property (IP) in MBD models when using LLM services is a concern. Vulnerabilities like prompt injection and ensuring data privacy during fine-tuning also need attention [34].
- **Data Availability and Quality:** High-quality, domain-specific MBD datasets for training or fine-tuning LLMs are scarce. Effectively representing complex graphical or equation-based models for LLM processing is also an ongoing research area.
- **Ethical and Societal Implications:** Issues such as potential deskilling of engineers, biases in LLM-generated designs, and accountability for LLM-induced errors need careful consideration.

Addressing these challenges requires interdisciplinary research efforts.

VII. Conclusion

This survey has explored the burgeoning role of Large Language Models in the Model-Based Design of Cyber-Physical Systems. We have outlined the broad scope of MBD artifacts and modeling languages, including Kahn Process Networks, discrete-event models, and UML/SysML, where LLMs can be applied. Key LLM use-cases across the MBD lifecycle—from model synthesis and requirements translation (e.g., NL to SysML/Capella [28]) to model understanding (e.g., Simulink model slicing [30]), code generation enhancements, and VV&T support—demonstrate significant potential to augment engineering workflows. Applications in demanding domains like automotive and aerospace further highlight this promise, albeit with domain-specific challenges.

Despite the rapid advancements, the reliable and safe integration of LLMs into MBD for safety-critical CPS faces substantial hurdles. Ensuring the correctness and verifiability of LLM-generated artifacts, effectively incorporating deep domain knowledge, addressing scalability, managing security and IP concerns [34], and mitigating ethical implications are paramount. Current LLMs are best viewed as powerful co-pilots for engineers, requiring human oversight and validation.

Future research should prioritize robust verification techniques for LLM outputs, domain-specific LLM adaptation, explainable AI for MBD, hybrid AI approaches combining LLMs with symbolic reasoning, and the development of standardized benchmarks. Addressing these areas will be crucial for unlocking the transformative potential of LLMs, paving the way for more efficient, innovative, and reliable design of next-generation CPS.

Furthermore, a promising avenue involves developing agent-based cloud verification frameworks. Such frameworks could leverage intelligent agents to manage and execute model verification tasks in scalable cloud environments, potentially utilizing virtual machines. Integrating LLMs within these frameworks could enhance the verification process by translating natural language requirements into formal specifications, providing human-understandable feedback from complex verification results, and facilitating transformations between diverse modeling formalisms [35]. This approach offers advantages like computational resource scalability, parallel processing capabilities for extensive verification campaigns, and dynamic access to updated tool libraries [36]. However, practical implementation requires addressing communication protocols, security mechanisms, and latency management. Case studies exploring the verification of LLM-generated models within such frameworks would be particularly insightful [37], [38]. Continued innovation in these areas will be vital for realizing the full potential of LLMs in engineering the complex cyber-physical systems of tomorrow.

References

[1] A. Sangiovanni-Vincentelli, P. S. Duggirala, and C. Li, "Model-based design for cyber-physical systems," *Foundations and Trends® in Electronic Design Automation*, vol. 6, no. 2, pp. 83–200, 2012.

[2] A. M. Madni and M. Sievers, "Model-based systems engineering: motivation, current status, and research opportunities," *Systems Engineering*, vol. 21, no. 3, pp. 172–190, 2017.

[3] M. Shafique, T. N. M. Zaini, R. Passarella, J. KLUČAROV 'A, S. Rehman, and A. Yasin, "Model based design approach in development of cyber physical systems: A survey," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 5, p. 4271, 2019.

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[5] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[6] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 82:1–82:37, 2017.

[7] M. Broy, "What is model-based software engineering? - a personal view," *Software & Systems Modeling*, vol. 8, no. 4, pp. 449–450, 2009.

[8] S. G. Kelkar and J. A. Farrell, "Model-based design of embedded control systems," *IEEE Control Systems Magazine*, vol. 33, no. 2, pp. 22–23, 2013.

[9] E. A. Lee, "Cyber physical systems: Design challenges," *University of California at Berkeley, Technical Report No. UCB/EECS-2008-8*, 2008.

[10] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design Automation Conference*, 2010, pp. 731–736.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] M. Chen, J. Tworek, H. W. Jun, Q. Yuan, H. P. d. C. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tiefenbach, J. Tezak, M. Sidor, F. P. Such, R. Cummings, M. Plappert, F. Chantzis, E. Barnes, I. Biberman, J. Dunning, W. Saunders, L. Weng, J. Weng, P. Molino, J. Leike, B. Zoph, J. Wu, D. Amodei, and I. Sutskever, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[13] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *Transactions on Machine Learning Research*, 2022.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[15] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[16] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. K. Papi, M. Kloumann, A. Korenev, P. S. Kunc, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Ly, I. Molybog, O. Monville, T. Queiroz, R. Rameesh, S. Rungta, K. Saladi, T. Schelten, R. Silva, P. E. M. Sinnott, R. S. Tan, L. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[17] The MathWorks, Inc., "Simulink - simulation and model-based design," https://www.mathworks.com/products/simulink.html, 2024.

[18] ANSYS, Inc., "Ansys scade suite - embedded software development environment," https://www.ansys.com/products/embedded-software/ansys-scade-suite, 2024.

[19] The MathWorks, Inc., "Stateflow - design and simulate decision logic," https://www.mathworks.com/products/stateflow.html, 2024.

[20] Object Management Group, "Unified modeling language (uml)," https://www.omg.org/spec/UML/2.5.1/, 2017.

[21] Ptolemy Project, UC Berkeley, "Ptolemy ii," https://ptolemy.berkeley.edu/ptolemyII/, 2024.

[22] G. Kahn, "The semantics of a simple language for parallel programming," pp. 471–475, 1974.

[23] Modelica Association, "Modelica language specification," https://specification.modelica.org/, 2023.

[24] The MathWorks, Inc., "Simscape - physical modeling and simulation," https://www.mathworks.com/products/simscape.html, 2024.

[25] Object Management Group, "Systems modeling language (sysml)," https://www.omg.org/spec/SysML/, 2022.

[26] SAE International, "As5506c: Architecture analysis and design language (aadl)," https://www.sae.org/standards/content/as5506c/, 2022.

[27] M. Chami, S. Zoghbi, and J.-M. Bruel, "Generating sysml models from textual requirements using named entity recognition," in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 2019, pp. 568–572.

[28] L. R. Timperley, L. Berthoud, C. Snider, and T. Tryfonas, "Assessment of large language models for use in generative design of model based spacecraft system architectures," *Journal of Engineering Design*, vol. 36, no. 4, pp. 550–570, 2025.

[29] A. Ferrari and P. Spoletini, "Formal requirements engineering and large language models: A two-way roadmap," *Information and Software Technology*, vol. 181, p. 107697, 2025.

[30] M. Fischer, S. Asayesh, S. Heshmat, and L. Briand, "Requirements-driven slicing of simulink models using llms," in *Proceedings of the 27th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS 24)*, 2024.

[31] M. Al-Shahrani, E. Denney, G. Pai, and I. Habli, "Gpt-4 and safety-case generation: An exploratory analysis," in *Computer Safety, Reliability, and Security. SAFECOMP 2023 Workshops*, 2023.

[32] Y. Zhao, C. Peng, H. Zeng, and Z. Gu, "Llm-enabled cyber-physical systems: A survey of research opportunities and challenges," *arXiv preprint arXiv:2402.16308*, 2024.

[33] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, Q. V. Do, Y. Xu, and P. Fung, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *arXiv preprint arXiv:2302.04023*, 2023.

[34] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A survey on large language model (llm) security and privacy: The good, the bad, and the ugly," *High-Confidence Computing*, vol. 4, no. 2, p. 100211, 2024.

[35] H. Fahmy, A. Al-Batainah, C. G. Lisciandra, M. M. Bersani, M. Salnitri, F. Sironi, M. T. T. El-Din, and F. Alkhatib, "Formal methods meet llms: A survey on testing and verification for cyber-physical systems," *arXiv preprint arXiv:2311.06959*, 2023.

[36] M. M. Bersani and M. Garc
'ia-Valls, "Online verification in cyber-physical systems: Practical bounds for meaningful temporal costs," *Journal of Software: Evolution and Process*, vol. 30, no. 3, p. e1880, 2018.

[37] P. G. Jensen, K. G. Larsen, and M. Mikucionis, "Model-based testing of a controller for an autonomous vehicle using uppaal tron: an industrial case study," *Formal Aspects of Computing*, vol. 23, no. 6, pp. 737–755, 2011.

[38] M. Broy, I. H. Kruger, A. Pretschner, and C. Salzmann, "Virtual design and verification of automotive cyber-physical systems," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.