UDC 004.582 DOI https://doi.org/10.35546/kntu2078-4481.2024.1.33

YU. S. MANZHOS
Candidate of Technical Sciences, Associate Professor,
Associate Professor at the Department of Software Engineering
National Aerospace University "Kharkiv Aviation Institute"
ORCID: 0000-0002-4910-7285

YE. V. SOKOLOVA
Candidate of Technical Sciences, Associate Professor,
Associate Professor at the Department of Software Engineering
National Aerospace University "Kharkiv Aviation Institute"
ORCID: 0000-0002-1497-4987

THE SOFTWARE DEVELOPMENT LIFECYCLE OF CYBER-PHYSICAL SYSTEMS

This work provides a comprehensive overview of the Software Development Lifecycle (SDLC) of Cyber-Physical Systems (CPS). The successful development of CPS heavily relies on implementing a comprehensive SDLC model that integrates various established methodologies and techniques. This model includes Model in the Loop (MIL), Software in the Loop (SIL), Processor in the Loop (PIL), and Hardware in the Loop (HIL), complemented by N-version Programming and formal verification and validation, including compile-time verification. By incorporating these elements, developers gain a structured framework to optimize workflows, ensure process consistency, and manage risks associated with system complexity. Including compile-time verification enables the early detection and resolution of potential issues, further enhancing CPS solution reliability and robustness. Additionally, N-version programming allows developers to improve software quality and reliability while efficiently managing resources. Moreover, integrating Field-Programmable Gate Arrays (FPGAs) into CPS architectures presents a scalable and adaptable solution to address performance challenges encountered by embedded processors. FPGAs offer parallel processing capabilities and hardware acceleration features, enabling CPS developers to enhance system performance, responsiveness, and reliability. This capability proves invaluable in meeting the stringent requirements of critical applications across diverse domains, including aerospace, defense, healthcare, and industrial automation. Adopting a comprehensive SDLC model facilitates the delivery of CPS solutions that meet stringent quality standards, tackle evolving technological challenges, and fulfill diverse stakeholder requirements. By catalyzing innovation and progress, this approach empowers CPS developers to confidently navigate technological complexities and deliver solutions that have a meaningful impact on society and the world. Through continuous refinement and advancement, CPS technology continues to push the boundaries of what is possible, driving progress and shaping the future of interconnected systems and environments.

Key words: *CPS, FPGA, HIL, MIL, PIL, SDLC, SIL.*

Ю. С. МАНЖОС
кандидат технічних наук, доцент,
доцент кафедри інженерії програмного забезпечення
Національний аерокосмічний університет
«Харківський авіаційний інститут»
ORCID: 0000-0002-4910-7285

Є. В. СОКОЛОВА
кандидат технічних наук, доцент,
доцент кафедри інженерії програмного забезпечення
Національний аерокосмічний університет
«Харківський авіаційний інститут»
ORCID: 0000-0002-1497-4987

ЖИТТЄВИЙ ЦИКЛ РОЗРОБЛЕННЯ КІБЕРФІЗИЧНИХ СИСТЕМ

У даній роботі надається огляд комплексної моделі розроблення програмного забезпечення (ПЗ) для кіберфізичних систем (КФС) життєвого циклу (ЖЦ), що включає в себе різноманітні методології та методи: Model in the Loop (MIL), Software in the Loop (SIL), Processor in the Loop (PIL), та Hardware in the Loop (HIL), доповнені N-версійним програмуванням, формальною верифікацією та валідацією, включаючи верифікацію на етапі компіляції. Інтеграція цих елементів у ЖЦ надає розробникам можливість оптимізувати робочі процеси, забезпечити сталість процесів та управляти ризиками, пов'язаними зі складністю системи. Включення

верифікації на етапі компіляції дозволяє на ранніх етапах виявляти та вирішувати потенційні проблеми, що покращує коректність ПЗ. N-версійне програмування дозволяє розробникам не тільки покращувати якість та надійність ПЗ КФС, але й ефективно використовувати ресурси. Інтеграція елементів Field-Programmable Gate Arrays (FPGAs) у архітектуру КФС робить можливим створення масштабованих та адаптивних рішень для подолання проблем продуктивності, з якими стикаються вбудовані процесори. Використання паралельного оброблення даних та апаратного прискорення дозволяє розробникам КФС покращити продуктивність та надійність системи, що є надзвичайно важливим у задоволенні вимог до критичних застосувань у різних галузях, таких як авіація, оборона, охорона здоров'я та промислова автоматизація. Таким чином, впровадження комплексної моделі ЖЦ стає ключовим чинником у покращенні програмних рішень для КФС. Запропоновані рішення відповідають високим стандартам якості, ефективно вирішують різноманітні технологічні завдання та забезпечують задоволення різноманітних потреб користувачів. Зазначений підхід виступає каталізатором для інновацій та прогресу, надаючи розробникам КФС впевненість у здатності подолати технологічні виклики та розробити програмні рішення, які мають значний вплив на суспільство та світ у цілому. Ця стратегія дозволяє розробникам відмінно орієнтуватися в сучасних технологічних складнощах та створювати продукти, що відповідають сучасним вимогам ринку та мають значний соціальний вплив.

Ключові слова: *КФС, ЖЦ, FPGA, HIL, MIL, PIL, SDLC, SIL.*

Formulation of the problem

Developing software for cyber-physical systems (CPS) [1] involves utilizing various concepts and models covering various technologies and devices. The CPS has brought a new era of connected devices and systems with healthcare and transportation applications. However, the reliability and security of these systems are critical concerns that must be addressed to ensure their safe and effective operation [2].

To ensure the standardization of development processes and enhance the reliability of CPS, we introduce a comprehensive lifecycle model. This model integrates established methodologies like Model in the Loop (MIL) [3], Software in the Loop (SIL) [3], Processor in the Loop (PIL) [4], and Hardware in the Loop (HIL) [5]. Additionally, it incorporates formal verification and validation (FV&V) techniques [6]. This holistic approach aims to streamline development while ensuring rigorous testing and validation at every stage of the CPS lifecycle.

This will contribute to achieving high quality and reliability metrics, critical components of Industry 5.0 [7]. However, effectively implementing this model requires addressing issues related to integrating diverse technologies, optimal resource utilization, and ensuring interaction between different stages of the lifecycle.

Analysis of recent research and publications

MIL is a development concept based on utilizing mathematical and software models of CPS at the customer requirements stage [3]. This approach integrates the model into a closed loop with real hardware and software. The MIL concept is widely used in developing control systems in aviation, the automotive industry, space technology, and industrial automation. Its use allows for effective error detection and correction and increases the efficiency and reliability of embedded systems. The result of work using the MIL methodology is a set of models describing various elements of CPS: sensors, actuators, and the external environment (such as aircraft, automobiles, machines, etc.).

SIL is a development concept for embedded system software based on customer requirements [3; 5]. SIL is one of the stages in the software development lifecycle and is based on using models. This concept allows for software testing without the physical presence of CPS hardware, which helps reduce resource costs for development.

PIL is a development concept for future CPS software on an embedded processor based on customer requirements [4]. This approach allows for identifying and correcting errors related to software interaction with a specific processor.

HIL is a development concept for future CPS software on embedded hardware based on customer requirements [5]. This methodology allows for identifying and correcting errors related to the interaction of CPS hardware with its processor.

The interrelation of these concepts, resulting in an adequate and correct CPS, is shown in Fig. 1.

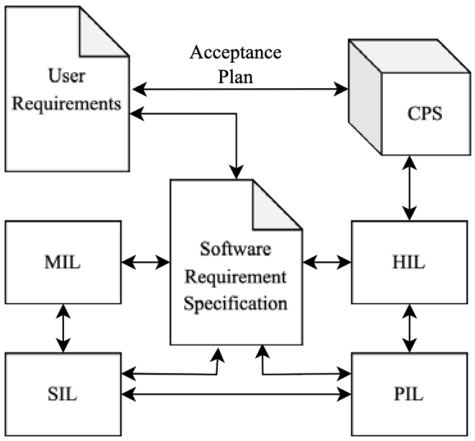

Fig. 1. CPS software development lifecycle

The development of CPS commences with defining requirements, drafting acceptance testing plans, and formulating detailed requirement specifications, accompanied by models depicting CPS components such as sensors, actuators, and the control entity. When constructing these models poses challenges, reverting to the customer's requirements is crucial. Precise mathematical and software models are employed to develop CPS algorithmic frameworks according to the SIL concept. Addressing any issues involves revisiting and refining the customer's requirements, creating new models, and resuming SIL processes.

Subsequently, in alignment with the PIL paradigm, rigorous validation of software algorithm models occurs on the embedded processor designated for the future CPS. Successful verification yields accurate software implementations meeting customer requirements and executable on the embedded processor. If algorithmic functions falter, revisiting customer requirements, refining or amending them, and resuming SIL processes are paramount.

Following the HIL principle, software functionality undergoes scrutiny within the CPS's hardware environment. If verifying within this context proves challenging, refining requirements becomes necessary.

Enhancing the existing software development life cycle entails leveraging formal verification techniques rooted in the physical dimension and orientation transformation checks. These methods ensure rigorous validation against physical parameters, bolstering CPS functionalities' reliability and performance.

Formulation of the research purpose

This research aims to devise a cohesive methodology tailored to the software development lifecycle of CPS, a cornerstone of the Industry 5.0 paradigm. Our principal aim is to augment the efficacy of established methodologies such as MIL, SIL, PIL, and HIL, integrating them with advanced FV&V techniques based on a specialized C++ type library [8], which implements the concept of the Hoare verifying compiler [9] and N-version programming to improve software reliability, as well as FPGA implementation to enhance performance. The ultimate objective is to ensure that CPS software achieves unparalleled quality and reliability, seamlessly meeting the demands of the contemporary environment.

Presentation of the main research material

As illustrated in Fig. 1, developing a comprehensive set of mathematical and software models becomes imperative after formulating requirement specifications outlined in Fig. 2.

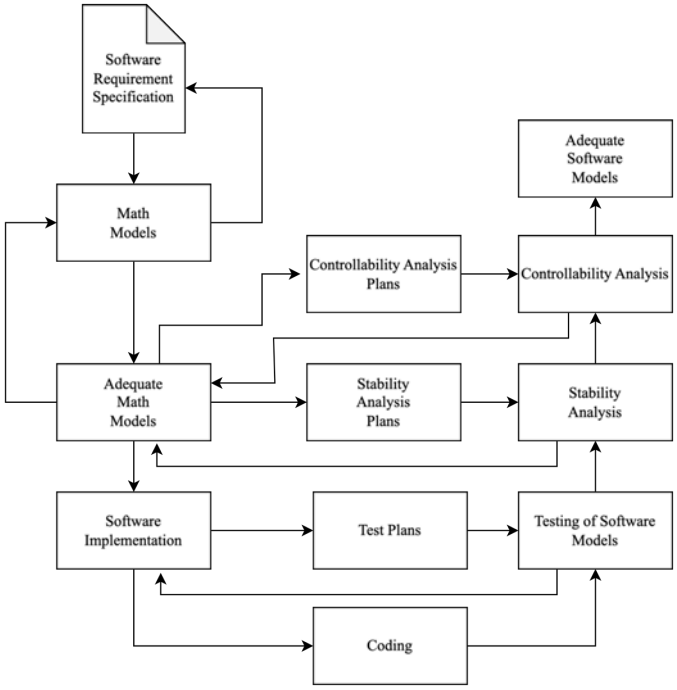

Fig. 2. Lifecycle of Cyber-Physical System Models Development

Initially, mathematical models of CPS are iteratively developed by software requirements specification and utilizing specialized tools such as Matlab /Simulink [10]. Based on these adequate mathematical models, their software implementations and stability and controllability verification plans are constructed. A specialized type library is employed to enhance the quality of software models, supporting physical dimensions and spatial orientation of software variables while providing additional checks during compilation and execution [11]. The software implementation of models forms the basis for constructing autonomous software model test plans. Subsequent stages involve coding and autonomous testing of software models. Following this, stability testing of models is conducted, and if necessary, we return to constructing adequate mathematical models with possible refinement of requirements. Then, stable software models are tested for controllability. If models fail to exhibit controllability, we must revisit requirement refinement. CPS's constructed adequate software models are further utilized (Fig. 3).

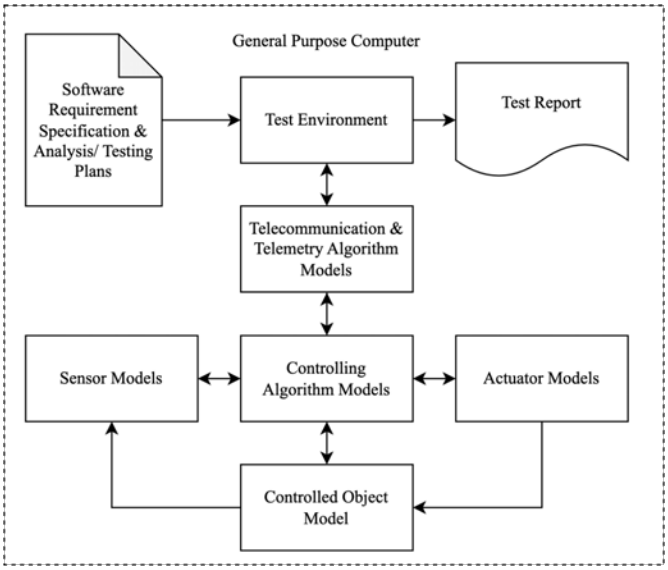

Fig. 3. Scheme of CPS Model Interaction Verification

Test verification plans are formulated based on the requirement specifications. These plans are then loaded into a testing environment, which interacts with models of telecommunication and telemetry algorithms. These algorithms further interact with models of control algorithms, sensors, actuators, and the controlled object. Actuator models control the object, while sensor models transmit information about the object's state to the control algorithms. Telecommunication and telemetry algorithms generate telemetry data for testing purposes and real-time maintenance. Telecommunication algorithms, based on Fourier & Chebyshev transform, are employed to reduce data volume [12; 13].

Control algorithms are developed based on customer requirements, and software models are created using a type library [11], which enables formal verification of the software code during compilation and execution. Software models of algorithms are developed for general-purpose computers. Autonomous testing is performed using tools such as CPPUnit [14]. The next stage, corresponding to the SIL concept, involves verifying software models of control algorithms (see Fig. 4).

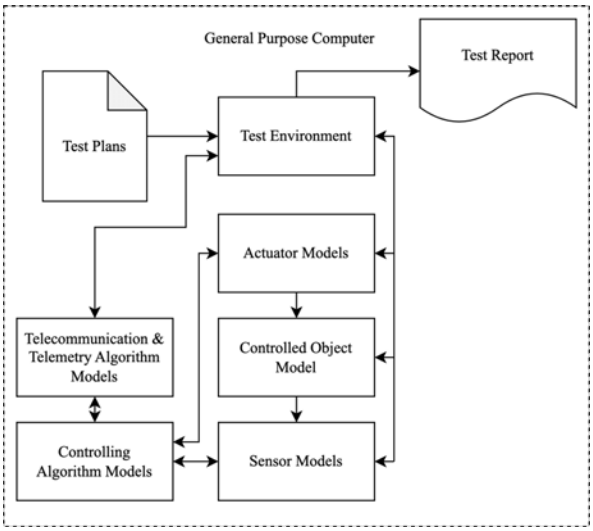

Fig. 4. Scheme for verifying software models of CPS algorithms

Dynamic testing is conducted based on test plans and allows for verifying the interaction of algorithms both among themselves and with sensors and actuators.

CPS software can be used sequentially in multiple configurations in the corresponding configuration file to perform various tasks in different hardware environments. According to the author's experiences, the flight software can include such configurations as orbit insertion, separation, Earth, Sun, and Moon orientation, targeting a specific star, forming a corrective impulse, stabilization, etc. Of course, over time, hardware may fail, but thanks to redundancy, the system can reconfigure itself. Therefore, sensors and actuators should be duplicated, and their operational status constantly monitored. A control object model allows for additional evaluation of the operation status of sensors and actuators by predicting the system's state and comparing it with the actual state obtained from sensors. To verify the sequential operation of the software in different configurations, the scheme depicted in Fig. 5 is used. The test plans define the set of configurations.

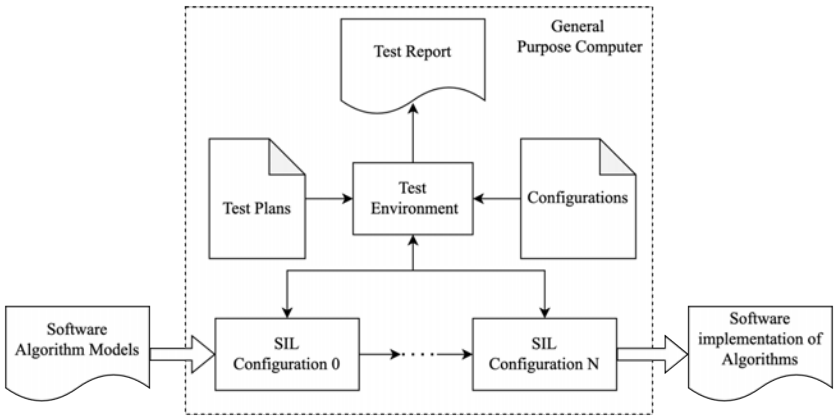

Fig. 5. Scheme for verifying the consistency of CPS software configurations

Each configuration defines the operating mode of the CPS software, corresponding to the current task of the CPS and the state of the hardware. In case of successful verification, we obtain a set of verified control algorithms; otherwise, modifying the corresponding algorithms with inter-mode interfaces is necessary. Next, according to the PIL concept, verifying the operation of the control program on the embedded processor is required (Fig. 6).

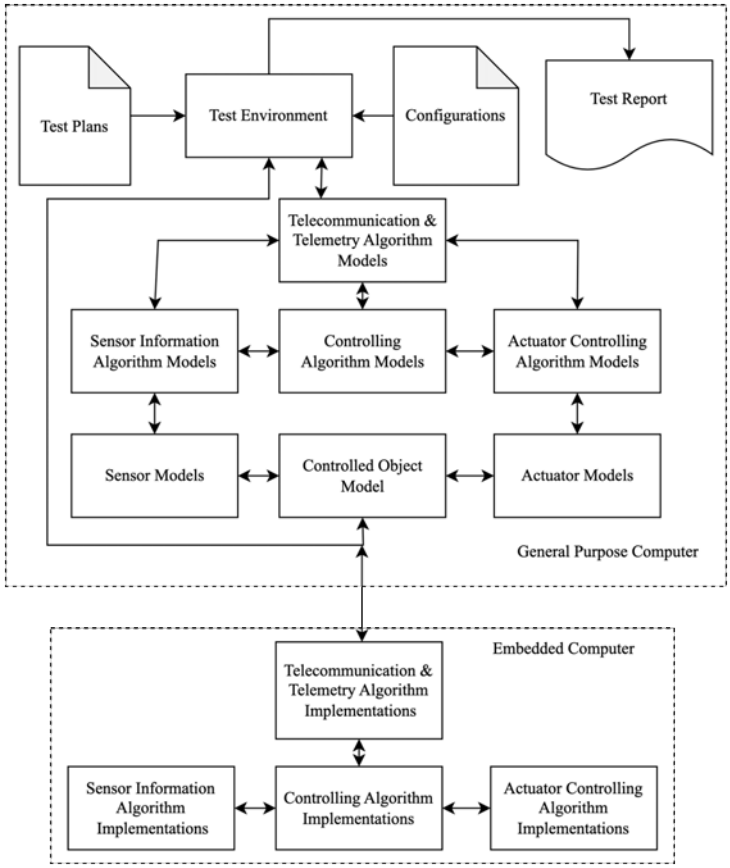

Fig. 6. Scheme for verifying the control program of the CPS on the target processor

To verify the operation of the control program on the embedded processor, two computers are used: a general-purpose computer running the test environment, algorithm models, devices, and the control object, and the embedded computer of the CPS running the program implementation of algorithms. According to the configuration and the test plan, the test environment configures the models on the general-purpose computer and the CPS program on the embedded computer. Additionally, data from sensor models are fed into the input of the embedded computer. The results of the control algorithms operating the actuators from the embedded computer are sent to the test environment of the general-purpose computer. This allows for comparing the results of the general-purpose and embedded computers' operation and detecting errors in the algorithm implementations. The verification results generate a test report.

Parallel development of multiple versions of software code is a beneficial practice that can significantly enhance program reliability. By concurrently working on different codebase versions, developers can cross-verify each version, thereby identifying and rectifying errors more effectively. This verification process leverages the assumption that errors in each version occur randomly, allowing for comprehensive coverage in error detection.

To illustrate, consider having one version of the program with a reliability factor of X and a second version with a reliability factor of Y. In this scenario, the overall reliability of the two programs, denoted as P, can be calculated using the formula $P = 1 – (1 – X) * (1 – Y)$. This formula accounts for the likelihood of errors in each version and computes the combined reliability of the programs. Such a substantial enhancement in reliability is only achievable when the control program is developed specifically for the target machine. In conclusion, parallel development fosters collaboration and innovation and significantly improves program reliability. By harnessing the collective efforts of developers and systematically verifying multiple versions of the code, organizations can ensure the delivery of robust and dependable software solutions [15].

In the subsequent phase (Fig. 7), the operational integrity of the CPS is assessed with actual devices, sensors, actuators, and the embedded computer governed by the developed program. Following the test plans and configurations, the

embedded computer is set up to execute designated tasks under predetermined conditions. In the final step, the developer employs the acceptance test plan to verify that the CPS software aligns with user requirements.

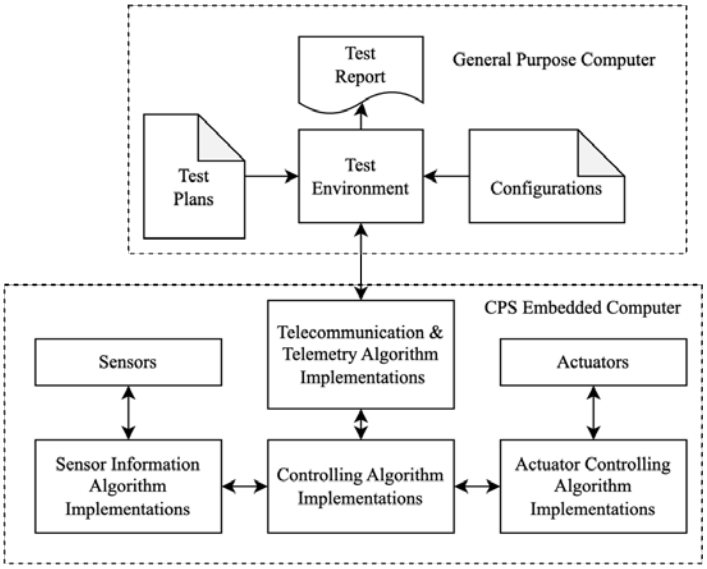

Fig. 7. Scheme of verification and validation of the CPS software

In critical CPS applications, the demands for processing power can often exceed the capabilities of traditional embedded processors. This shortfall in performance can arise due to the complexity of algorithms, the need for real-time data processing, or the sheer volume of computations required to maintain system responsiveness and reliability. In such scenarios, Field-Programmable Gate Arrays (FPGAs) [15; 16] emerge as a powerful solution to augment the capabilities of embedded processors. FPGAs offer the flexibility of hardware acceleration, enabling them to execute intensive tasks with remarkable efficiency. Their parallel processing capabilities allow for the simultaneous execution of multiple tasks, making them ideal for handling the high computational demands of critical CPS applications.

Moreover, FPGAs excel in real-time data processing thanks to their ability to process data at ultra-low latency. This ensures that critical data is processed and acted upon promptly, which is essential for maintaining CPS systems' responsiveness and effectiveness in dynamic environments.

Additionally, FPGAs can execute complex control algorithms with precision and reliability. Whether regulating autonomous systems, managing power distribution networks, or controlling robotic actuators, FPGAs provide the computational horsepower to ensure the smooth and efficient operation of CPS applications. FPGA-based solutions implementation involves several steps:

1. System Design: Define CPS architecture, identify critical functions, and partition between software and FPGA-based hardware.

2. FPGA Development: Develop and synthesize FPGA designs for hardware accelerators using HDLs like Verilog or VHDL.

3. Integration: Integrate FPGA-based accelerators into the CPS system, interfacing with software, sensors, actuators, and other devices.

4. Testing and Verification: Perform rigorous testing and verification of FPGA-based accelerators for correctness, reliability, and safety compliance.

5. Deployment and Maintenance: Deploy FPGA-based CPS software and provide ongoing maintenance and support for reliability and performance.

FPGA implementation enhances CPS performance, reliability, and responsiveness for critical aerospace, defense, healthcare, and industrial automation applications. Integrating FPGAs into CPS architectures offers a scalable and adaptable solution to address embedded processors' performance challenges. By leveraging FPGAs' parallel processing capabilities and hardware acceleration features, CPS developers can enhance system performance, responsiveness, and reliability, meeting critical applications' stringent requirements across various domains.

Conclusion

Implementing a comprehensive lifecycle model is vital for the success of the development of cyber-physical systems. By integrating established methodologies such as Model in the Loop, Software in the Loop, Processor in the Loop, and Hardware in the Loop, alongside N-version Programming and formal verification and validation techniques, along with compile-time verification, this model offers a structured and efficient framework for CPS software development.

This integrated approach empowers developers to optimize workflows, ensure consistency in development processes, and mitigate risks associated with system complexity. With compile-time verification, developers can proactively identify and address potential issues at an early stage of development, further enhancing the reliability and robustness of CPS solutions. Moreover, by incorporating N-version programming, developers can proactively improve software quality and reliability while minimizing resource usage for software development.

Incorporating FPGAs into CPS architectures provides a scalable and versatile solution to confront the performance hurdles embedded processors encounter. Harnessing the parallel processing capabilities and hardware acceleration features of FPGAs, CPS developers can bolster system performance, responsiveness, and reliability, meeting the rigorous demands of critical applications across diverse domains.

Ultimately, adopting a comprehensive lifecycle model facilitates the delivery of CPS solutions that meet stringent quality standards, address evolving technological challenges, and fulfill the diverse requirements of stakeholders. It serves as a cornerstone for fostering innovation, driving progress, and realizing the full potential of Cyber-Physical Systems across various domains. With this approach, CPS developers can confidently navigate the complexities of modern technology and deliver solutions that have a meaningful impact on the world.

References

1. Oks, S.J., Jalowski, M., Lechner, M. *et al.* (2022) Cyber-Physical Systems in the Context of Industry 4.0: A Review, Categorization and Outlook. *Inf Syst Front*. DOI: 10.1007/s10796-022-10252-x

2. Amit Kumar Tyagi, N. Sreenath. (2021) Cyber Physical Systems: Analyses, challenges and possible solutions. *Internet of Things and Cyber-Physical Systems*, 1, 22–33, DOI: 10.1016/j.iotcps.2021.12.002.

3. Dapynhunlang Shylla, Ayushi Jain, Pritesh Shah, Ravi Sekhar (2023). Model in Loop (MIL), Software in Loop (SIL) and Hardware in Loop (HIL) Testing in MBD. *4th IEEE Global Conference for Advancement in Technology (GCAT) At: Bangalore, India.* https: //www. researchgate.net /publication /376658429_Model_in_Loop_MIL_Software_in_Loop_SIL_and_Hardware_in_Loop_HIL_Testing_in_MBD. DOI: 10.1109/GCAT59970.2023.10353323

4. Amr Mohamed, A.N. Ouda, Jing Ren, Moustafa El-Gindy (2020). Processor-in-the-loop co-simulations and control system design for a scaled autonomous multi-wheeled combat vehicle. *International Journal of Automation and Control,* 14(2): 138. https://www.researchgate.net/publication/339709349_Processor-in-the-loop_co-simulations_and_control_system_design_for_a_scaled_autonomous_multi-wheeled_combat_vehicle. DOI: 10.1504/IJAAC.2020.105516

5. Jonis Kiesbye, David Messmann, Maximilian Preisinger, Martin Langer (2019). Hardware-In-The-Loop and Software-In-The-Loop Testing of the MOVE-II CubeSat. *Aerospace,* 6(12): 130, https: // www.researchgate.net /publication /337699363_Hardware-In-The-Loop_and_Software-In-The-Loop_Testing_of_the_MOVE-II_CubeSat, DOI: 10.3390/aerospace6120130

6. Krichen, Moez. (2023). A Survey on Formal Verification and Validation Techniques for Internet of Things. *Applied Sciences,* 13, no. 14: 8122. DOI: 10.3390/app13148122

7. Shanhe Lou, Zhongxu Hu, Yiran Zhang, Feng Yixiong et al. (2024). Human-Cyber-Physical System for Industry 5.0: A Review From a Human-Centric Perspective. *IEEE Transactions on Automation Science and Engineering,* PP(99): 1–18. https: //www.researchgate.net /publication /378053098_Human-Cyber-Physical_System_for_Industry_50_A_Review_From_a_Human-Centric_Perspective. DOI: 10.1109/TASE.2024.3360476.

8. Manzhos, Y.; Sokolova, Y. (2023) A Practical Type System for Formal Verification CPS &Amp; IoT C/C++ Programs. *Preprints*, 2023052228. DOI: 10.20944/preprints202305.2228.v1

9. Yu Tan, Dlanfu Ma, Lel Qlao (2021). A Formal Verification Method of Compilation Based on C Safety Subset. *Wireless Communications and Mobile Computing,* 2021:1–10. https: //www.researchgate.net /publication /353633617_A_Formal_Verification_Method_of_Compilation_Based_on_C_Safety_Subset. DOI: 10.1155/2021/8352267

10. Siqi Lin, Wei Yao, Yongxin Xiong, Yifan Zhao, et al. (2023). MatPSST: A Matlab/Simulink-based power system simulation toolbox for research and education. *IET Generation, Transmission & Distribution*, 17(10), https:// www.researchgate.net/ publication/ 368969051_MatPSST_A_MatlabSimulink-based_power_system_simulation_toolbox_for_research_and_education, DOI: 10.1049/gtd2.12805

11. Manzhos, Y., & Sokolova, Y. (2023) A Software Verification Method for the Internet of Things and Cyber-Physical Systems. *Computation,* no. 11 (7), article no. 135. DOI: 10.3390/computation11070135.

12. Manzos, Y., Sokolova, Y. (2020) The method of data compression in Internet of Things communication. *Radioelectronic and Computer Systems,* no. 4, pp. 57–67. DOI:10.32620/reks.2020.4.05 (In Ukrainian)

13. Manzhos, Y., & Sokolova, Y. (2022) A Method of IoT Information Compression. *International Journal of Computing,* vol. 21, Iss. 1, pp. 100–110. DOI: 10.47839/ijc.21.1.2523.

Gabor Marton, Zoltan Porkolab. Unit Testing in C++ with Compiler Instrumentation and Friends. *Acta Cybernetica*, 23(2): 659–686. https://www.researchgate.net/publication/320587819_Unit_Testing_in_C_with_Compiler_Instrumentation_and_Friends. DOI: 10.14232/actacyb.23.2.2017.14

14. Júlio Mendonça, Fumio Machida, Marcus Völp. (2023) Enhancing the Reliability of Perception Systems using N-version Programming and Rejuvenation. *DSN 2023 Workshop on Dependable and Secure Machine Learning (DSML 2023)* At: Porto, Portugal. https: //www.researchgate.net /publication /37211024_Enhancing_the_Reliability_of_Perception_Systems_using_N-version_Programming_and_Rejuvenation. DOI: 10.1109/DSN-W58399.2023.00044

15. Elias Vansteenkiste. (2016). *New FPGA Design Tools and Architectures.* [Thesis for: Doctor of Electrical Engineering]. https: //www.researchgate.net /publication /311845419_New_FPGA_Design_Tools_and_Architectures

16. Ang Li, David Wentzlaff. (2021). PRGA: An Open-Source FPGA Research and Prototyping Framework. *In The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '21)* (p. 127–137*)*. Association for Computing Machinery, New York, NY, USA, DOI: 10.1145/3431920.3439294