Model-based Stochastic Error Propagation Analysis for Cyber-Physical Systems

Tagir Fabarisov[1], Nafisa Yusupova[2], Kai Ding[3], Andrey Morozov[4], Klaus Janschek[1]

[1]Institute of Automation, Technische Universität Dresden, Georg-Schumann-Str. 11, 01187 Dresden, Germany, {tagir.fabarisov, klaus.janschek}@tu-dresden.de

[2]Faculty of Computer Science and Robotics, Ufa State Aviation Technical University, ul. K. Marx 12, 450000 Ufa, Russia, yussupova@ugatu.ac.ru

[3]Bosch (China) Investment Ltd., Corporate Research, Shanghai, China, kai.ding@cn.bosch.com

[4]Institute of Industrial Automation and Software Engineering, University of Stuttgart, Germany, andrey.morozov@ias.uni-stuttgart.de

Abstract: Industry 4.0 is the current trend of automation and data exchange in manufacturing technologies that is focusing on the creation of smart factories with the modular structured Cyber-Physical Systems (CPS), in tight cooperation with humans. This trend also implies that the systems become more complex, heterogeneous, and distributed especially their network and software parts. This makes the CPS highly critical subject to failures at different levels, including software, hardware, and human operators. Consequently, ensuring reliable and safe operation under the presence of non-avoidable threats also becomes a more complicated task. The proper analysis of the CPS requires thorough comprehension of both the dependability properties of system components and their interactions as well as structural and behavioral aspects of the complete system. Such an analysis of complex and mutually interlinked system properties puts considerable challenges on appropriate methods for modeling and analysis, as well as, on the related applied software tools. The Dual-graph Error Propagation Model (DEPM), developed in our lab, is a mathematical abstraction of the main future system's properties, which are vital for the determination of the error propagation processes. It is a useful analytical instrument for the evaluation of the influence of particular faults and errors to the overall system behavior. OpenErrorPro is our analytical software tool for stochastic error propagation analysis that supports the DEPM framework. Using OpenErrorPro, a DTMC model could be automatically generated from a DEPM, and the reliability metrics, in addition to, error propagation path, can be computed. This could be implemented for the analysis of the heterogeneous CPS components. The necessary steps for the DEPM framework extension, required for such an implementation, are discussed in this paper.

Keywords: Cyber-Physical System; Industry 4.0; Markov chain model; Probabilistic Model Checking; Error propagation model; reliability; safety; dependability; model-based system; model-based analysis; control flow; data flow; optimization

1 Introduction

Industry 4.0 is the current trend of automation and data exchange in manufacturing technologies that is focused on the creation of smart factories with the modular structured Cyber-Physical Systems (CPS) in tight cooperation with humans. This trend also implies that the systems become more and more complex, heterogeneous, and distributed, especially their network and software parts. This makes the CPS highly critical subject to failures at different levels including software, hardware, and human operators. Consequently, ensuring reliable and safe operation under the presence of non-avoidable threats also becomes more and more complicated task. The proper analysis of the CPS requires thorough comprehension of both dependability properties of system components and their interactions as well as structural and behavioral aspects of the complete system. Such an analysis of complex and mutually interlinked system properties puts considerable challenges on appropriate methods for modeling and analysis and on applied software tools.

Model-based System Engineering (MBSE) [1] approaches help both to simplify and speed up system development and provide semi-formal information for earlier system analysis. However, reliability and safety evaluation methods like Fault Tree Analysis and Failure Mode and Effect Analysis that are recommended in nowadays industrial standards fail to describe system behavioral aspects in a sufficiently deep manner. Additional sophisticated and highly specialized methods for the analysis of the effects of faults are required, such as the method for error propagation analysis which is discussed in this paper. Model-based analysis is MBSE-oriented analysis for the earlier design phases. One of such is the error propagation analysis (EPA). Employing EPA allows a design engineer to find and fix, or at least mitigate, the errors and flaws of the system design on its earlier design phase.

EPA can be based either on formal methods, e.g. probabilistic model checking techniques [2], or fault injection approaches. The formal analysis can be performed in early system design phases and strongly depends on the system model quality. Fault injections require either an already implemented system or its executable model. Analytical methods suffer from the state space explosion. The time required for the analysis can grow exponentially with the system model complexity. Usually, there is a kind of system model complexity threshold. After this threshold is exceeded, the required time grows so fast that the analytical approach becomes practically inapplicable. Conducting simulation-based analysis

with fault injections allows extending the number of experiments for reaching a wanted confidence level. However, fault injection methods have problems with the simulation of rare events, especially when we want to model faults that have extremely low activation probabilities.

2 State of the Art

2.1 Reliability and Safety Analysis

Classical methods, such as Fault Tree Analysis (FTA, IEC 61025), Markov Analysis (IEC 61165) [3], Failure Mode and Effect Analysis (FMEA, IEC 60812), and Hazard and Operability Analysis (HAZOP, IEC 61882) [4] are recommended in current safety standards. However, due to the increasing system complexity, these methods fail to describe aspects of modern mechatronic systems and CPS in sufficiently deep manner. FMEA, FMECA, and HAZOP are top-level qualitative methods that require numerical input from quantitative methods like FTA or Markov Analysis. For instance, the methods like FTA are not suitable for the reliability analysis of the systems with complex interactions between components and enhanced software parts. Markov methods can cover these aspects, but in most cases, the described solutions are either simple and high-level or prone to the state space explosion.

Model checking proposes advanced methods that allow a system designer, given a formal model of a system, exhaustively and automatically, check whether this model meets a given specification. Available model checking tools are based on state-based models such as state machines, discrete and continuous Markov chains, and Markov Decision Process models (e.g. PRISM, MRMC, STORM, SHAREP, SPIN, NuSMV) [5], Petri Net models (e.g. TAPAAL, ROMEO, ORIS, TimedNet) [6], or more high-level formal descriptions like AADL-based SLIM language (COMPASS) [7] or AltaRica language (OpenAltarica, ARC Studio) [8]. These powerful methods require specific and rather deep knowledge of the model checking such as discussed formal models and the formalization of the required system properties using temporal logic. Most of these methods are oriented to the manual and top-level analysis.

2.2 System Modeling

Several examples of widely used system modeling paradigms can be employed for the Industry 4.0 CPS systems. Structured Analysis [9] is rather old, relatively simple and straightforward design method that fits well for high-level functional design, small projects, and fast prototypes. In comparison with the modern

modeling methods, such as UML/SysML, the structured analysis doesn't provide enough design capabilities for complex system aspects. UML [10] with its extension SysML (Unified and Systems Modeling Languages) [11] is a popular and universal design approach. UML models can cover all phases of the system development life cycle. A great many UML/SysML design tools are available and integrated into industrial processes, including, MagicDraw, IBM Rational family, and UML Enterprise Architect. Another tools support automated UML-based MBSE including the "executable UML" and auto code.

Simulink and Stateflow [12] is the dominating modeling paradigm for dynamic systems, embedded control systems, and digital signal processing. Simulink provides a "native" interface for control engineers in the form of combined block diagrams (Simulink) and state charts (Stateflow) [13]. This Mathworks toolset includes built-in code generation and deployment mechanisms. This is also a perfect method for fast prototyping and software and hardware in the loop. The Mathworks tools do not support well early phases of the development such as functional design, or top-level composition. Therefore, Simulink/Stateflow is often used in together with other composition models like general UML or AADL.

AADL (Architecture Analysis & Design Language) [14] is a united framework for the model-based engineering of embedded real-time systems. AADL is strongly oriented to software and hardware co-design of real-time systems. One of the key features of this method is the inherited analytical capabilities. AADL has interfaces with analytical tools including COMPASS [7] for reach dependability and performance analysis, PRISM [5] for stochastic model checking, and OpenFTA [15] for fault tree analysis. AADL, like the Simulink/Stateflow, is also very specific and has relatively poor tool support (OSATE [16]) in comparison with Simulink/Stateflow or UML.

Human-in-the-Loop, Human-in-the-Loop Cyber-Physical Systems, Cyber-Physical-Human Systems, and "Internet of All" concepts share the common idea to consider the human as a part of a larger Cyber-Physical System (CPS). This idea has drawn considerable interest in recent years, as it is comprehensively surveyed in [17].

2.3 Dual-Graph Error Propagation Model

The analytical methods for model-based system reliability and safety analysis is required in order to ensure the reliable and safe operation under the presence of non-avoidable in the numerous applications in Industry 4.0. The proper analysis of the CPS requires thorough comprehension of both dependability properties of system components and their interactions as well as structural and behavioral aspects of the complete system. Such a method will find its usage in safety critical industrial domains including aerospace, automotive, transportation, medical and robotics applications, where a failure or malfunction may result in environmental

harm, severe equipment damage or even serious injuries of the personal. Industry 4.0, which is the current trend of automation and data exchange in manufacturing technologies, creates what is called a Smart Factory that consists of the modular structured Cyber-Physical Systems in tight cooperation with humans. With a robust increasing of these trends, aforementioned will find its implementation in the context of smart factory's reliability and safety and, therefore, will benefit to the environmentally friendly manufacturing technologies.

Therefore, the analysis of fault activation, error propagation, and timing properties of a given CPS is viable in order to ensure the safe operation. The error propagation analysis can be based either on formal methods, e.g., probabilistic model checking techniques, or fault injection approaches. The formal analysis can be performed in early system design phases and strongly depends on the system model quality, and subsequently be used when using the fault injection approaches.

The Dual-graph Error Propagation Model (DEPM) [18], developed in our lab, is a mathematical abstraction of the main future system's properties, which are vital for the determination of the error propagation processes. It is a useful analytical instrument for the evaluation of the influence of particular faults and errors to the overall system behavior.

DEPM framework describes the process-oriented model that allows the reliability modeling of heterogeneous CPS components, their interaction, nontrivial failure scenarios, multiple failure modes, hierarchical compositions, data errors propagation, timing aspects, and sophisticated control and data flow structures with branching, loops, and guarded stochastic transitions. The DEPM allows the computation of the reliability metrics using underlying Discrete-time Markov chain (DTMC) models. DEPMs can be automatically generated from common CPS models including the Simulink/Stateflow models [19], UML models [20], SysML models [21], AADL [22] models, as well as software source code using LLVM [23]. This allows not only the automated application of our method, but also the analysis of systems developed with a combination of modeling paradigms. For example, Simulink/Stateflow is often used together with other composition models developed using UML or AADL. The DEPM is a mathematical model that captures system control and data flow structures and reliability properties of system components.

An example of a Dual-graph Error Propagation Model, that describe a reference Cyber-Physical System is presented in Figure 1. This system consists of two autonomous and connected cars, and a navigation system connected via a network. Two cars are equipped with controllers and a set of sensors. Controllers collect the data from sensors regarding the obstacles on the course, as well as the latitude and longitude coordinates of a car. These data are being transmitted between two cars in order to ensure the safety. The critical failure of the system is defined as *"sensor does not provide necessary data."* Another failure of the system could be

described as *"network is down."* Assume that the sensor system is prone both to transient, bounded in time, and permanent, continuous in time, faults. Our system is tolerant to transient faults since the coordinates from the navigation system allows the system to survive during short sensor downtime. The permanent sensor fault leads to the system failure in case that the network is down unless an operator detects the failure and fixes it.

The DEPM combines two directed graph models: A Control Flow Graph (CFG) and a Data Flow Graph (DFG). The nodes of the graphs represent executable system elements and (generic) data storages. The rounded rectangles with black borders represent the elements *aController*, *aCar*, *aSensors*, *Network*, *NavSystem*, *bCar*, *bSensors*, *bController*, and one special element *RepairMode* that models the corrective actions. The CFG arcs model the control flow transitions (black lines) between the elements. The DFG arcs model the data flow transitions (blue lines) between the elements and data storages (rectangles with blue borders). The data storages are specified with finite sets of string or integer values. For example, data *Ping* takes values in a range *[0, 300]* that represent the network latency and *aNetState* takes values between *{OK, FAIL}*, representing the operational and fail states of the network. The DEPM also supports hierarchical systems. Elements *aSensors* and *bSensors* of a top-level DEPM contain an internal DEPMs with a set of *Sensor* elements. These elements are sub-systems, that contains a DEPM models itself. While the reliability metrics are being compute, these models should be calculated first in order to compute the high-level model.

The cyclic process starts with the execution of the initial element *Network* and in a nominal case continues with the execution of either *aCar*, *bCar* or *NavSystem*. The control flow transition from *Network* to *RepairMode* represents the network failure detection. This is a guarded stochastic transition specified by control flow commands of the element *Network* shown in the gray rectangle expressed in PRISM input language.

The command *"0.01:(cf'=RepairMode)&(NetState'=FAIL)"* specifies that the process jumps from *Network* to *RepairMode* with the probability 0.01 only if *Ping* less than 200. Similar to the control flow commands, the elements can have data flow commands that specify the fault activations during the element execution and the propagation of the errors from data inputs to outputs. For example, the data flow commands of the element *aController* manage the data *aObstacles*. The commands of the element *aController* model transient and permanent generator faults. The transient fault *"(aNetState'=FAIL)&(aSenState'=OK)"* occurs with the probability 0.04 and the permanent fault *"(aNetState'=FAIL)& (aSenState'=FAIL)"* with the probability 0.02. As we can see, in case of the permanent fault, the value of *aSenState* is changed to *FAIL* and stays *FAIL* unless we jump to the element *RepairMode*. The red arrow-shaped nodes such as *SensorFailure* and *PackageDrop* specify system failures using expressions that contain the values of data storages and the special variable *cf* that models the current control flow state. A fixed execution time is defined for each element in

order to compute time-related reliability metrics such as MTTF or mean downtime. In this example, the execution of the element *RepairMode* takes much longer (300s) than the execution of the *Network* elements (1s).

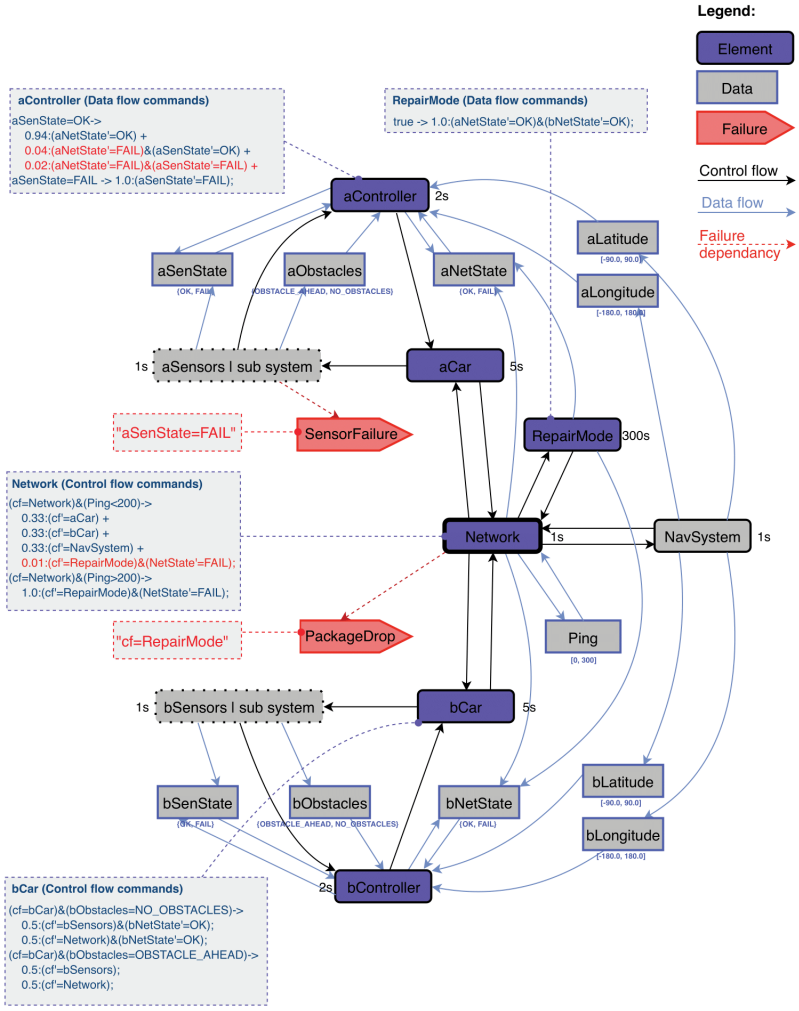

Figure 1
An example of a Dual-graph Error Propagation Model for Cyber-Physical System

3 Analytical Toolset for Error Propagation Analysis

3.1 Analytical Toolset for Model-based Stochastic EPA

Using the DEPM framework described in the previous chapter allow a designer to calculate the dependability metrics, e.g. a mean number of errors, mean time to failure etc., using discrete-time Markov chain (DTMC) models. A DTMC model describes system dynamics as a stochastic process in terms of errors occurrence and propagation. OpenErrorPro [24] is our analytical software tool for stochastic error propagation analysis, that support DEPM framework. Using OpenErrorPro a DTMC model could be automatically generated from a DEPM and the reliability matrices cold be calculated, for instance, the MTTF metric for *"SensorFailure"* failure (Figure 1).

OpenErrorPro is implemented in Python and consists of three main parts: error propagation library, graphic user interface (GUI), and model transformation algorithms. For efficient computation of the generated DTMC models, that can be formally analyzed, the OpenErrorPro uses a built-in interface with PRISM software [5]. The tool also integrates optimization methods against state space explosion of DTMC models such as the automatic nesting algorithm and data flow slicing [25]. The model transformation algorithms allow automatic generation of DEPMs from the baseline system models UML/SysML [21], Simulink/Stateflow [19], AADL [22]. After that OpenErrorPro automatically creates discrete time Markov chain models using the DEPM representation and computes required numerical reliability properties using PRISM software. The most relevant and close methods and tools are high-level model checkers COMPASS, OpenAltaRica, and Figaro, which are discussed and compared with OpenErrorPro in [24]. OpenErrorPro enables the reliability-related features analysis of CPS that cannot be modelled by other methods for quantitative system-level reliability analysis. In Figure 2 a GUI of the toolset with an open DEPM model of the CPS presented in previous chapter is shown.

3.2 Model-based Stochastic EPA for Cyber-Physical Systems

The DEPM framework as well as OpenErrorPro tool have been applied for several case studies during the last years [26]. Different types of mechatronic systems have been analyzed, including the moving and flying robots, and specific model-based software systems from the space and automotive industrial domains. The DEPM is a powerful analytical instrument for the modeling of a broad set of dependability-related features that influence error propagation processes. The feasibility of the DEPM approach has been witnessed by a number of case studies including navigation system of a mobile robot [27], automated medical patient

table motion control [22], robotic software for space-to ground haptic feedback control [18], and embedded flight control software of a UAV [21]. DEPM framework could be applied for CPS in the analysis of the propagation of typical CPS errors from potentially faulty components to technically accessible system variables, e.g. network, sensor outputs, the software of the controllers, embedded boards, and other computing units. As it was presented in [24], the DEPM framework could be applied for the complex model design. OpenErrorPro helped to perform the sensitivity analysis of system reliability to identify the places where it makes sense to apply system redundancy. In the work, authors have encountered state space explosion issues of the underlying Markov chain models. Nevertheless, they resolved them with the optimization trick using the fact, that on the DEPM-level OpenErrorPro provides a fully automated access to the control and data flow structures.

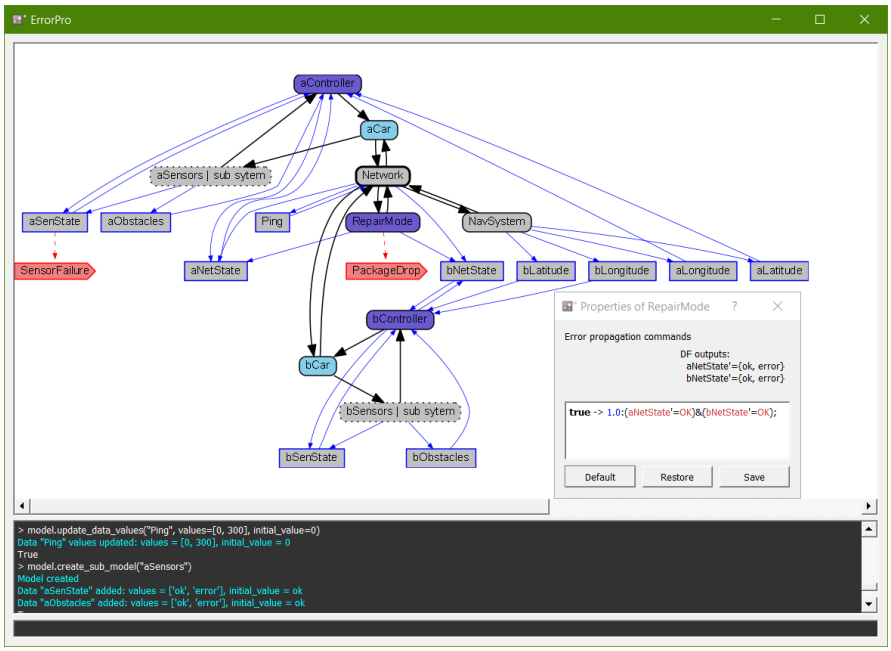

Figure 2

A GUI of the OpenErrorPro toolset with DEPM of a Cyber-Physical System. Note how in *"Properties of RepairMode window"* the data flow commands represents a repairing after the failure.

For the implementation of DEPM in CPS reliability analysis, there is required a generic approach for the automated creation of a formal hierarchical DEPM from semi-formal heterogeneous CPS models. It can be divided in the following steps. First, the given CPS is being decomposed into individual components and mapped into the DEPM elements. While performing this step, there must be a persistent conceptual separation between hardware and software parts, as it's done

previously in [28] for hardware and in [19] for software. The hardware-level DEPMs will model sensors, actuators, network, and computing units. The software of the computing units will be modelled as sub-DEPMs of the corresponding hardware components. For the model-based software, e.g. implemented in Simulink, the block functions will play the role of the DEPM elements [18]. For the manually implemented software LLVM-based method [29] will be used that decomposes the code into single instructions. Second step is to map the CPS data flow structure into DEPM data storages and data flows. This information is commonly available in some specific format, e.g. SysML Internal Block Diagrams, UML Component Diagrams, AADL models for hardware or Simulink block diagrams for model-based software. The key challenge is the integration of the DEPMs generated from several types of baseline models in such a way that the final hierarchical DEPM will capture comprehensively the propagation of the common CPS errors and their combinations. For that the design of a generic approach to extend the DEPM with the CPS error types and define corresponding error propagation rules is needed. After that, the next step is to mark the DEPM data storages that model technically accessible variables and critical variables, e.g. actuator commands.

The propagation of an error to a critical DEPM part will cause a system failure. The DEPM, generated following the discussed steps, will contain no probabilistic parameters. The further extension of the DEPM with the probabilistic parameters will enable the quantitative estimation of the probability that errors will be detected before they propagate to the critical system parts. First, the information about the system operational profile will be added, such as sensor rates, network transmission times, sample times and the control flow structure of the software components. This information will be mapped into the DEPM control flows either from available behavioral models, e.g. UML/SysML Activity Diagrams, State Machines, or AADL models with timing properties, or via the extension of the system with logging, execution, and evaluation of the gathered statistical information about the execution patterns of the CPS components. One of the challenges is to distinguish between stochastic and deterministic control flow events taking into account the imperfect synchronization of the CPS components. Finally, the DEPM must be extended with the probabilities of the error propagation through the components and fault activation. The error propagation from inputs to outputs of hardware components can be described either deterministically from the known component specification or using local fault injections. The error propagation through the software parts can be computed with the corresponding sub-DEPMs, as it shown in [24]. Fault activation probabilities can be evaluated using the available sources of the failure data such as FIDES [30], the Electronic Parts Reliability Data (ERPD) [31], or the Non-Electronic Parts Reliability Data (NRPD) [32]. However, these sources address the “complete” component failures. The probabilities for different error types are extremely hard to find. Thus, all the available probabilistic information will be added into the DEPM and cover unknown probabilities with local

nondeterminism. OpenErrorPro, discussed in previous chapter, supports automatic generation of DTMC (Discrete-time Markov Chain) models from a DEPM and analyzing them using the interface with modern probabilistic model checkers such as PRISM [5] and Storm [33]. These model checkers support local nondeterminism, however, in the future we will need to switch from DTMCs to (Markov Decision Process) models. The more information we feed to DEPM the more precise the estimation will be. The nondeterminism will result in several equally suitable sets, that have to be considered, in the subsequent processing steps of DEPM approach.

Conclusion

The Dual-graph Error Propagation Model has been discussed in this paper. DEPM is a process-oriented model that allows the reliability modeling of heterogeneous components, their interaction, nontrivial failure scenarios, multiple failure modes, hierarchical compositions, data errors propagation, timing aspects, and sophisticated control and data flow structures with branching, loops and guarded stochastic transitions. The DEPM allows the computation of the reliability metrics using underlying Discrete-time Markov chain models. DEPMs can be automatically generated from common CPS models, including the Simulink/Stateflow, UML/SysML, AADL models, as well as, software source code using LLVM software for stochastic error propagation analysis. OpenErrorPro error propagation tool, based on the described DEPM, has also been discussed in this article. This tool plays the role of an intermediate analytical model between the baseline system model and the formal mathematical model that can be formally analyzed. It can automatically create discrete-time Markov chain models using the DEPM representation and calculate the required numerical reliability properties. In addition to traditional reliability metrics, OpenErrorPro enables the evaluation of customizable reliability metrics and the application of effective optimizations, against the state space explosion of underlying Markov chain models already on the DEPM level. This toolset has been proven to have functionality and capability, for implementation in the analysis of the heterogeneous components of a Cyber-Physical System. As it was discussed herein, OpenErrorPro could be applied for the analysis of CPS. In order to be able to generate a formal DEPM from the CPS representation, future efforts will expand the DEPM approach, including such capabilities as discussed in this paper.

Acknowledgment

This work has been supported by DAAD “Michail Lomonosov” program jointly with Russian Ministry of Science and Higher Education funds.

References

[1] Gianni, Daniele; D'Ambrogio, Andrea; Tolk, Andreas, eds. Modeling and Simulation-Based Systems Engineering Handbook (1 ed.) USA: CRC Press, 2014, ISBN 9781466571457

[2] C. Baier and J.-P. Katoen. Principles of model checking. MIT press, 2008

[3] N. B. Fuqua, "The applicability of markov analysis methods to reliability, maintainability, and safety," Selected Topic in Assurance Related Technologies (START), Vol. 2, No. 10, pp. 1-8, 2003

[4] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the stateof-the-art in modeling, analysis and tools," Computer science review, Vol. 15, pp. 29-62, 2015

[5] Marta Kwiatkowska, Gethin Norman and David Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In Proc. 23rd International Conference on Computer Aided Verification (CAV'11), volume 6806 of LNCS, pp. 585-591, Springer, 2011

[6] Katoen J P, Zapreev I S, Hahn E M, et al. The ins and outs of the probabilistic model checker MRMC [J]. Performance evaluation, 2011, 68(2): 90-104

[7] Bozzano M, Cimatti A, Katoen J P, et al. The COMPASS Approach: Correctness, Modeling and Performability of Aerospace Systems[C]//SAFECOMP. 2009, 5775: 173-186

[8] Arnold A, Point G, Griffault A, et al. The AltaRica formalism for describing concurrent systems [J]. Fundamenta Informaticae, 1999, 40(2, 3): 109-124

[9] E. Yourdon, Modern Structured Analysis. Yourdon Press, 1989

[10] OMG. OMG Unified Modeling Language (OMG UML), 2019

[11] Delligatti, Lenny. SysML Distilled: A Brief Guide to the Systems Modeling Language, Addison-Wesley Professional, 2013, ISBN 978-0-321-92786-6

[12] Mathworks, Matlab & Simulink: Simulink User's Guide R2019b. Retrieved 2020

[13] Mathworks, Matlab & Simulink: Stateflow User's Guide R2019b, Retrieved 2020

[14] P. Feiler, D. Gluch, "Model-based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language," Addison-Wesley Professional, 2012

[15] Formal Software Construction Limited, OpenFTA, Version 1.0, User Manual

[16] Carnegie Mellon University, "Welcome to OSATE," Online: http://www.osate.org

[17] D. S. Nunes, P. Zhang, J. Sá Silva, "A survey on Human-in-the-Loop applications towards an Internet of All", IEEE Communications Surveys & Tutorials, Vol. 17, No. 2, 2015

[18] Morozov, Andrey & Janschek, Klaus & Krüger, Thomas & Schiele, André. (2016) Stochastic Error Propagation Analysis of Model-driven Space Robotic Software Implemented in Simulink. MORSE '16, Proceedings of the 3rd Workshop on Model-Driven Robot Software Engineering: 24-31

[19] K. Ding, A. Morozov, and K. Janschek. Reliability evaluation of functionally equivalent simulink implementations of a pid controller under silent data corruption. In 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE), pp. 47-57, IEEE, 2018

[20] K. Ding, T. Mutzke, A. Morozov, and K. Janschek. Automatic transformation of uml system models for model-based error propagation analysis of mechatronic systems. IFAC-PapersOnLine, 49(21):439-446, 2016, 7th IFAC Symposium on Mechatronic Systems MECHATRONICS 2016

[21] M. Steurer, A. Morozov, K. Janschek, and K.-P. Neitzke. Sysml-based profile for dependable uav design. In 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS), 2018

[22] A. Morozov, T. Mutzke, B. Ren, and K. Janschek. Aadl-based stochastic error propagation analysis for reliable system design of a medical patient table. In IEEE Annual Reliability & Maintainability Symposium (RAMS), 2018

[23] A. Morozov, Y. Zhou, and K. Janschek. Llvm-based stochastic error propagation analysis of manually developed software components. In Proceedings of the European Safety and Reliability Conference (ESREL18), 2018

[24] Morozov, A., Ding, K., Steurer, M., & Janschek, K. (2019, October) OpenErrorPro: A New Tool for Stochastic Model-Based Reliability and Resilience Analysis. In 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE) (pp. 303-312) IEEE

[25] Zhao F, Morozov A, Yusupova N I, et al. Nesting algorithm for dual-graph error propagation models[C]//CSIT'2016. 2016: 106-110

[26] Fabarisov, T., et al. "The efficiency comparison of the prism and storm probabilistic model checkers for error propagation analysis tasks." Industry 4.0 3.5 (2018): 229-231

[27] A. Morozov and K. Janschek. Probabilistic error propagation model for mechatronic systems. Mechatronics, 24(8):1189-1202, 2014

[28] A. Morozov, K. Janschek. Flight Control Software Failure Mitigation: Design Optimization for Software-implemented Fault Detectors. In Proceedings of 20th IFAC Symposium on Automatic Control in Aerospace ACA 2016, Sherbrooke, Quebec, Canada, 21-25 August 2016

[29] Vidineev, Viacheslav, et al. "LLVM-based C to DEPM transformation tool: New functionality and performance improvements." Информационные технологии интеллектуальной поддержки принятия решений. 2019

[30] F. Guide et al. Reliability methodology for electronic systems. FIDES group, 2009

[31] W. Denson, P. Jaworski, W. Crowell, and D. Mahar. Electronic parts reliability data 1997. 1996

[32] W. Denson, G. Chandler, W. Crowell, A. Clark, and P. Jaworski. Nonelectronic parts reliability data 1995, Technical report, RELIABILITY ANALYSIS CENTER GRIFFISS AFB NY, 1994

[33] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk. A storm is coming: A modern probabilistic model checker. In International Conference on Computer Aided Verification, pp. 592-600, Springer, 2017