Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 52-12 (2019) 394–399

Model-based Dependability Analysis of Fault-tolerant Inertial Navigation System: A Practical Experience Report

Mikael Steurer [*] **Andrey Morozov** [*] **Klaus Janschek** [*] **Klaus-Peter Neitzke** [**]

[*] *Institute of Automation, Technische Universität Dresden, Dresden, Germany, (e-mail: [mikael.steurer, andrey.morozov, klaus.janschek]@tu-dresden.de).*

[**] *Institute for Informatics, Automation and Electronics, Hochschule Nordhausen, Nordhausen, Germany, (e-mail: klaus-peter.neitzke@hs-nordhausen.de*

Abstract: Model-based systems engineering approaches are commonly used to develop safety-critical mechatronic systems. Recently, a new SysML-based method for the dependability analysis of Unmanned Aerial Vehicles (UAVs) has been introduced. The method consists of three main steps: (i) creation of a structural SysML model using building blocks from the underlying UAV dependability profile that extends the model with block-level reliability and time properties, (ii) transformation of the semi-formal SysML model into a formal Dual-Graph Error Propagation Model (DEPM) that captures relevant structural and behavioral properties of the system, (iii) DEPM-based evaluation of system dependability metrics using Markov chain models and state-of-the-art probabilistic model checking techniques. This paper describes the practitioner experiences and lessons learned after the application of the aforementioned method to a sophisticated real-world embedded fault-tolerant inertial navigation system. The case study revealed two particular limitations that have been overcome by the optimization of the method against the state-space explosion of underlying Markov chain models and the introduction of a new computation algorithm for DEPMs with realistic extremely low fault activation probabilities.

Keywords: system analysis and design, systems architecture, reliability engineering, dependability, SysML, Markov chains, microsensors, space technology

1. INTRODUCTION

Safety is defined as the absence of catastrophic consequences on the user(s) and the environment Avižienis et al. (2004). The measurement of angular velocity and acceleration is an important task of every Unmanned Aerial Vehicle (UAV). Usually, an Inertial Measurement Unit (IMU) fulfills this task. A failure in the measurement results in a high risk to human and environment. Also, the integrated IMU data, merged with the data from other sensors, e.g. magnetoresistive sensors, help to navigate the UAV in case of degraded Global Navigation Satellite System (GNSS) or during indoor missions. The IMUFUSION case study system, presented in this paper, was developed in order to satisfy the following objectives: (1) record inertial measurement data in a near space vehicle with a specially designed, robust, and miniaturized system, (2) estimate the flight trajectory including orientation by the recorded inertial measurement data, and (3) integrate a redundancy concept for higher reliability, diagnosis capability, and accuracy. A prototypical IMU with redundant sensors, microcontrollers, and memory modules has been developed and tested in the gondola of the stratospheric research balloon of the Balloon EXperiments for University Students (BEXUS) program BEXUS (2018) and REXUS/BEXUS (2018). The IMUFUSION project was one of seven BEXUS experiments carried in two BEXUS balloons which are lifted to a payload depending altitude of 30 km with a flight duration of about five hours. The main hardware components and the wiring concept are shown in Fig. 1. Two Micro Controller Units (MCUs) with redundant memory modules, 3-axis gyroscopes, 3-axis acceleration sensors, and 3-axis magnetic field sensors are the central parts of the IMUFUSION system. These components are distributed on three Printed Circuit Boards (PCBs): *Master PCBA-1*, *Slave PCBA-2*, and *PCBA-4*. *PCBA-3* is responsible for the power management. During the experiment, the system recorded inertial data in a near space vehicle with a specially designed robust and miniaturized system and computed the time-dependent attitude and position of the gondola.

2. STATE OF THE ART

The classical Fault Tree Analysis (FTA) Ruijters and Stoelinga (2015) and the Reliability Block Diagram (RBD) Kim (2011) are the most common quantitative system-level reliability evaluation techniques. The FTA is a top-down approach in which an undesired state of a system

10.1016/j.ifacol.2019.11.275

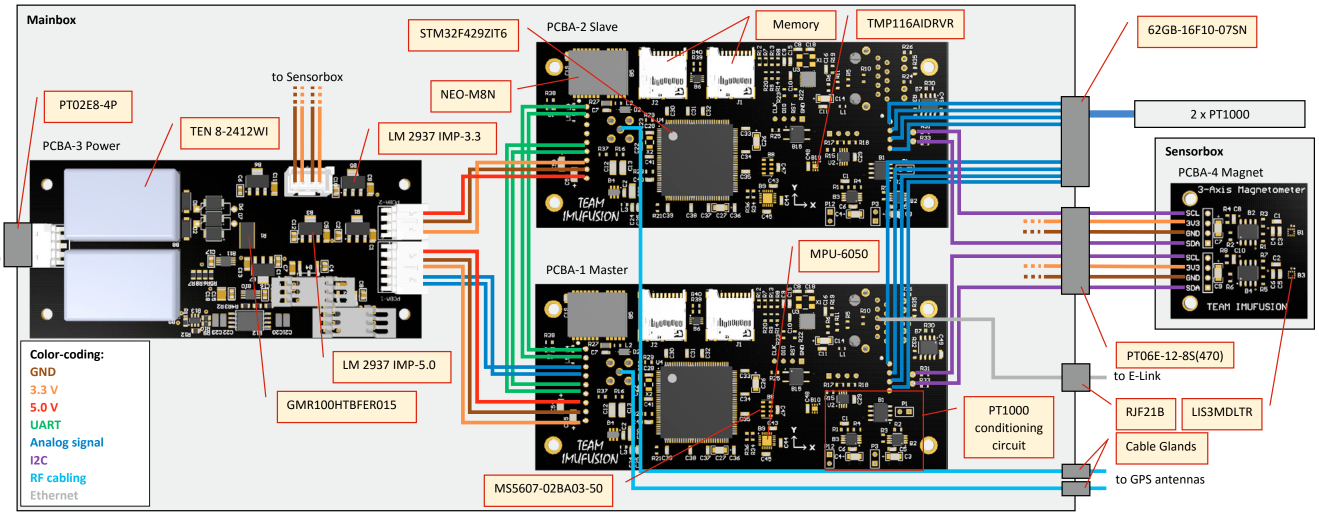


Fig. 1. The IMUFUSION system architecture: Electronic components and the wiring concept.

is analyzed using the Boolean logic that combines a series of lower-level events such as failures of individual components. RBDs have a similar underlying concept, but model the system success while fault trees model a system failure.

Model-based systems engineering approaches such as UML, SysML, AADL, and Simulink UML (2001); OMG (2008); Feiler et al. (2006); Ong (1998) are common to develop safety-critical systems. The system presented in this paper has been designed using the SysML. Reliability models can be generated automatically from various baseline system models, including SysML diagrams. Methods for the automatic generation of fault trees from SysML diagrams are proposed in Mhenni et al. (2014) and Machida et al. (2013) respectively. Other SysML-based methods for quantitative reliability analysis exploit various types of Markov chain models Debbabi et al. (2010); Ouchani et al. (2014); Jarraya et al. (2007); Ali et al. (2015); Baouya et al. (2015). Markov chain based methods provide access to powerful Probabilistic Model Checking (PMC) Baier and Katoen (2008) techniques that allow the evaluation of advanced, highly customizable, time-related reliability properties.

Besides the discussed general methods, there are more specific ones. A SysML-based method for the dependability analysis of UAVs is introduced in Steurer et al. (2018). The new domain-specific SysML UAV Dependability Profile (UDP), that captures reliability-related properties of UAV components and the transformation algorithm from profiled SysML models to the formal Dual-graph Error Propagation Models (DEPM) Morozov and Janschek (2014) for further extensive analysis are two key parts of this method. The UDP allows annotating SysML blocks and flows with update frequencies and reliability properties. The update frequencies are defined based on the technical description and the configuration of the deployed components. The reliability properties are defined using expert experience, common guidelines for part reliability evaluation, or the combination of both. The annotated SysML model is transformed into a DEPM using the algorithms presented in Steurer et al. (2018). The DEPM captures system and data flow structures, and reliability properties of system components and enables the computation of reliability metrics using underlying Discrete-Time Markov Chain (DTMC) models. Fig. 5 shows an example of a DEPM. The DEPM combines two directed graph models: a control flow graph and a data flow graph. The nodes of the graphs represent executable system elements (rounded rectangles) and data storages (rectangles with blue borders). Control flow arcs (black lines) model control flow transitions between the elements. Data flow arcs (blue lines) model data transfer between the elements and data storages. The reliability metrics are computed for the defined failures (highlighted in red) using automatically generated DTMC models whose states describe the current control flow state (which element will be executed next) and the states of all data storages. Technically a DEPM is transformed into one or several PRISM Kwiatkowska et al. (2002) models for numerical evaluation with stochastic model checkers like PRISM or STORM Dehnert et al. (2017).

The method presented in Steurer et al. (2018) has been demonstrated with a rather simple quadcopter case study. This paper describes the practitioner experiences and lessons learned after the full-scale method application to a real sophisticated inertial navigation system. The application has proven the feasibility of the method and revealed two particular limitations, which have been overcome by optimizing the method against the state-space explosion of underlying Markov chain models and introducing a new computation algorithm for DEPMs with extremely low fault activation probabilities.

3. SYSML AND UDP SYSTEM MODELING

The creation of a profiled SysML model is the first step of the analytical workflow as illustrated in Fig. 2. We used the UML and SysML modeling tool Papyrus Lanusse et al. (2009). The SysML Internal Block Diagram (IBD) in Fig. 3 shows how the instances of the top-level blocks are interconnected. The system consists of seven sub-components: two *SensorArray (UDP stereotype <<Sensor>>), two Mcu (UDP stereotype <<MCU>>), PowerSys (UDP stereotype <<PowerSystem>>)*, and two *Gps (UDP stereotype <<GPS>>*. The *PowerSys* part is powered by the external battery pack of the gondola and

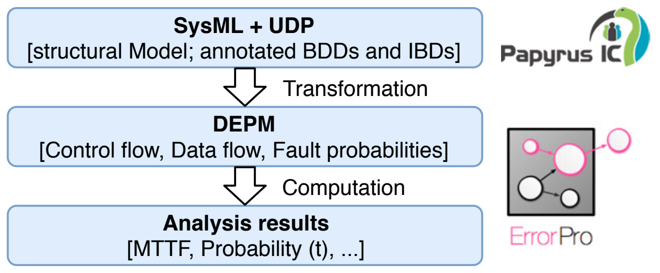

SysML + UDP
[structural Model; annotated BDDs and IBDs]
Transformation
DEPM
[Control flow, Data flow, Fault probabilities]
Computation
Analysis results
[MTTF, Probability (t), ...]
Papyrus IC
ErrorPro

Fig. 2. The method workflow.

provides 3.3 V to the *sa1*, *sa2*, *master*, and *slave* parts and additionally 5 V to the *master* and *slave* parts. The *master* and *slave* parts distribute the 3.3 V power supply to the sensors and the GPS receivers. The *sa1* and *sa2* parts send the measured data via analog and digital interfaces to *master* and *slave* respectively. Each *SensorArray* contains all sensors that are necessary for the computation of the current pose and position. To achieve higher accuracy the data of both *SensorArrays* are shared via Universal Asynchronous Receiver Transmitter (UART) connection between *master* and *slave* and merged in both. The *gps1* and *gps2* components share their data also with the *master* and the *slave* via UART. Similar IBDs have been created for all lower hierarchical levels.

The power system mainly contains two *TEN 8-2412WI* dc/dc converters, one *GMR100HTBFER015* high power low ohmic chip shunt resistor, one *InputCurrentVoltage* power measurement conditioning circuit, four *LM 2937 IMP-3.3* 3.3 V linear voltage regulators, and two *LM 2937 IMP-5.0* 5.0 V linear voltage regulators. Each *SensorArray* primarily consists of one *MPU-6050* combined 3-axis accelerometer and gyroscope, one *MS5607-02BA03-50* barometric pressure sensor, one *TMP116AIDRVR* digital pressure sensor, one *CiSPresSens* special for the prevailing conditions calibrated CiS pressure sensor, one *LIS3MDLTR* 3-axis magnetometer, and one *Pt1000* outside temperature sensor with the related conditioning circuit *OutsideTempCircuit*. The essential parts of the *Mcu* are the *STM32F429ZIT6* 32-bit *Arm Cortex-M4* based microcontroller with a frequency of up to 180 MHz, the surrounding Printed Circuit Board Assembly (PCBA) including two separated micro SD-card memories per controller and an ethernet interface. Only the instance *master* is connected to the ground station for the purpose of communication. The *Gps* part mainly consists of the *NEO-M8N* GPS-module and the *Taoglas Titan GPS RG-174* GPS-antenna.

All atomic (indivisible) components are annotated with update frequencies and failure rates, see Table 1. The related flow arcs are annotated with update frequencies. The update frequencies are defined using the component specifications and refer to the update of the values of the flow model components. The failure rates are specified using reliability guidelines such as the FIDES Guide et al. (2009), or the Nonelectronic Parts Reliability Data (NRPD) Denson et al. (1994).

4. SYSML TO DEPM TRANSFORMATION

A systematic approach for the transformation of the SysML model into a DEPM is the second step and described in Steurer et al. (2018). This method is partially reused in our case study. However, it was modified in order to exploit new features of the latest version of the DEPM-based tool OpenErrorPro Morozov et al. (2015).

1) Frequency classes: System components modeled with the SysML UDP blocks are classified according to their update frequencies. The case-study system contains the three frequency classes: f_1, f_2, and f_3 whereby the fastest class f_3 (1 MHz) contains 13 components, f_2 (1 KHz) contains 12 components, and the slowest class f_1 (10 Hz) contains 4 components. Analog components with a theoretically infinite frequency are classified into the highest frequency class.

2) DEPM elements and control flow arcs: For each atomic SysML UDP block, we create a DEPM element. Based on the defined frequency classes the elements are grouped into three hierarchical DEPMs. The top-level DEPM, see Fig. 5, contains the elements with the lowest update frequencies as well as the compound element f_2 that contains the DEPM of the second frequency class. The same structure is created for the second-level DEPM and so on. The number of repetitions represents the iterations of the sub model until the superordinate level is executed according to $rep_i = f_i/f_{i-1}$. Where, f_i is the frequency of the higher frequency class and f_{i-1} is the frequency of the lower frequency class. For instance, the number of repetitions of the compound element f_2 of the top-level DEPM is equal to $rep_2 = 1000Hz/10Hz = 100$. The elements of the top-level DEPM are connected into a loop structure with DEPM control flow arcs based on the attributes *informationSource* and *informationTarget* of the *UDP_Flows*. The elements of the other DEPMs are connected sequentially.

3) DEPM data storages and data flow arcs: Data storages are created and connected via DEPM data flow arcs with the corresponding DEPM elements, based on the *UDP_Flow* attributes such as *informationSource*, *informationTarget*, and *name*. Also, backward data flow arcs from a data storage to the source element are added in order to model permanent faults. Such data storages connected via incoming and outgoing data flow arcs with the same element are called state data. In Fig. 5, for example the state data *dGpsAnt1* models both the state and the output of the element *GpsAnt1*. Redundant elements require an additional data comparison element. For example, Fig. 4 shows two redundant voltage regulators *Reg12V1*, *Reg12V2*, and the following comparison element *Reg12Comp* with their associated error propagation commands. All state data of sub DEPMs f_3 and f_2 also appear in the top-level DEPM f_1 in order to model error propagation between the frequency classes, as shown in Fig. 5.

4) Probabilistic commands: Probabilistic commands in PRISM format are created to describe the fault activation and error propagation processes in the DEPM elements. The probabilities of errors in the outputs of the elements are calculated as follows:

$$EP_c = \frac{\lambda_c\ [FIT]}{f_c\ [Hz] \cdot 3600\ [\frac{s}{h}] \cdot 10^9} \quad (1)$$

Where λ_c is the component failure rate and f_c the update frequency of the component. The computed error

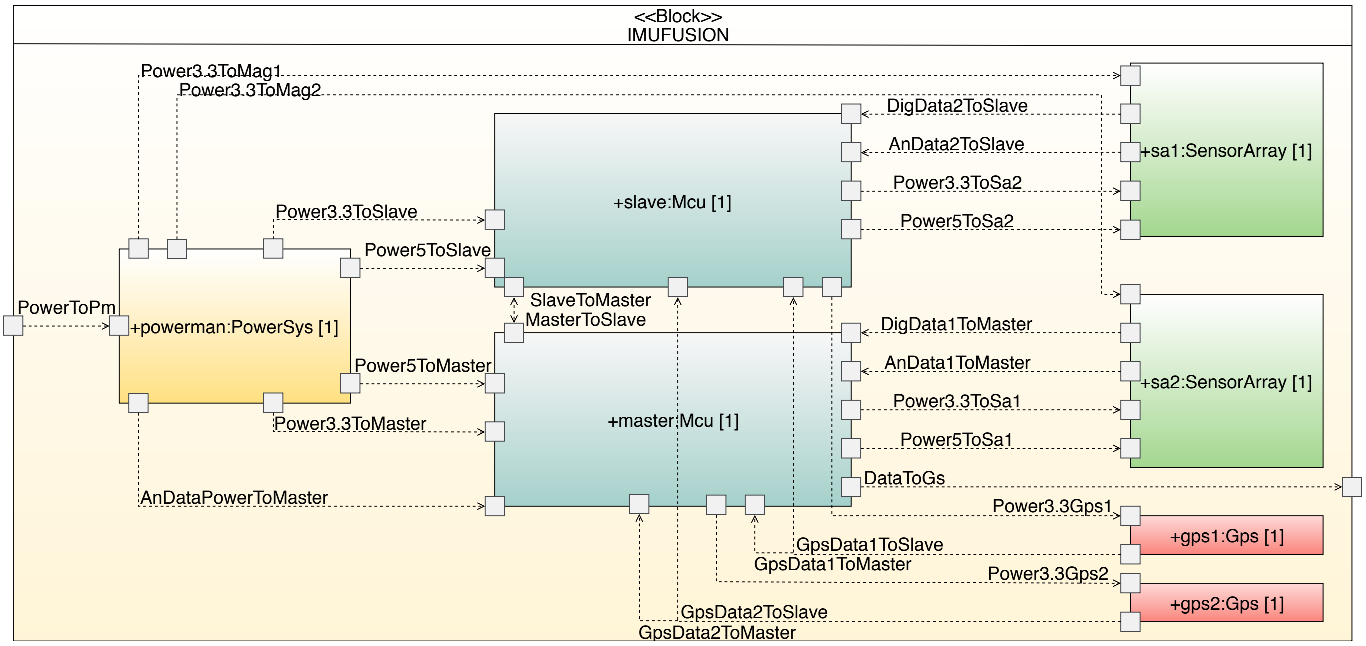
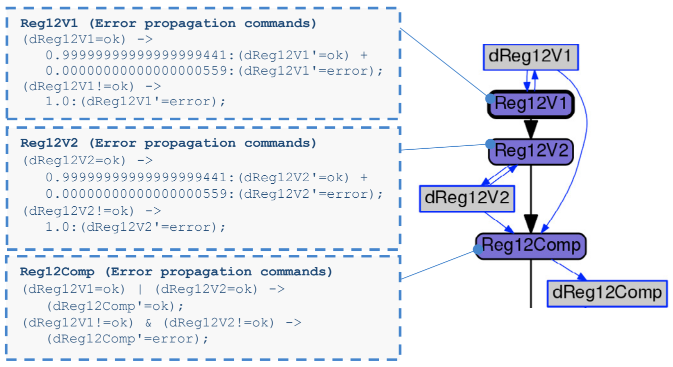


Fig. 3. The IMUFUSION top-level Internal Block Diagram.



```
Reg12V1 (Error propagation commands)
(dReg12V1=ok) ->
   0.99999999999999999441:(dReg12V1'=ok) +
   0.00000000000000000559:(dReg12V1'=error);
(dReg12V1!=ok) ->
   1.0:(dReg12V1'=error);
```

```
Reg12V2 (Error propagation commands)
(dReg12V2=ok) ->
   0.99999999999999999441:(dReg12V2'=ok) +
   0.00000000000000000559:(dReg12V2'=error);
(dReg12V2!=ok) ->
   1.0:(dReg12V2'=error);
```

```
Reg12Comp (Error propagation commands)
(dReg12V1=ok) | (dReg12V2=ok) ->
   (dReg12Comp'=ok);
(dReg12V1!=ok) & (dReg12V2!=ok) ->
   (dReg12Comp'=error);
```

Fig. 4. A fragment of the frequency class f_3 DEPM.

Table 1. The component failure rates λ_c, the frequency f_c, and the specific error probabilities EP_c.

Name	λ_c [FIT]	f_c [Hz]	EP_c
Shunt	1.98	1000000	5.507e-19
Voltage regulator	20.14	1000000	5.594e-18
Accel/Gyro	3648.35	1000	1.013e-12
Temperature sensor	10.20	1000	2.833e-15
Magnetometer	3624.45	1000	1.007e-12
Pressure sensor	1728.95	1000	4.803e-13
Memory	12.53	1000	3.481e-15
PCBA	496.92	1000000	1.380e-16
Microcontroller	32.00	1000000	8.889e-18
GPS module	12.71	10	3.530e-13
GPS antenna	20.65	10	5.735e-13

probabilities P_F are listed in the fourth column of Table 1. The probabilistic commands for comparison elements are created separately, see the commands of the element *Reg12Comp* in Fig. 4.

5) Number of steps and element timing properties: The DEPM analysis is based on automatically generated DTMC models. Therefore, numerical results are related to the number of steps. Each step corresponds to a DTMC transition from one state to another that is equivalent to the execution of one DEPM element. The number of steps for the top-level DEPM that corresponds to a time unit should be computed in order to match analysis results to the system operation time. The number of steps per hour of the top-level DEPM is calculated as follows:

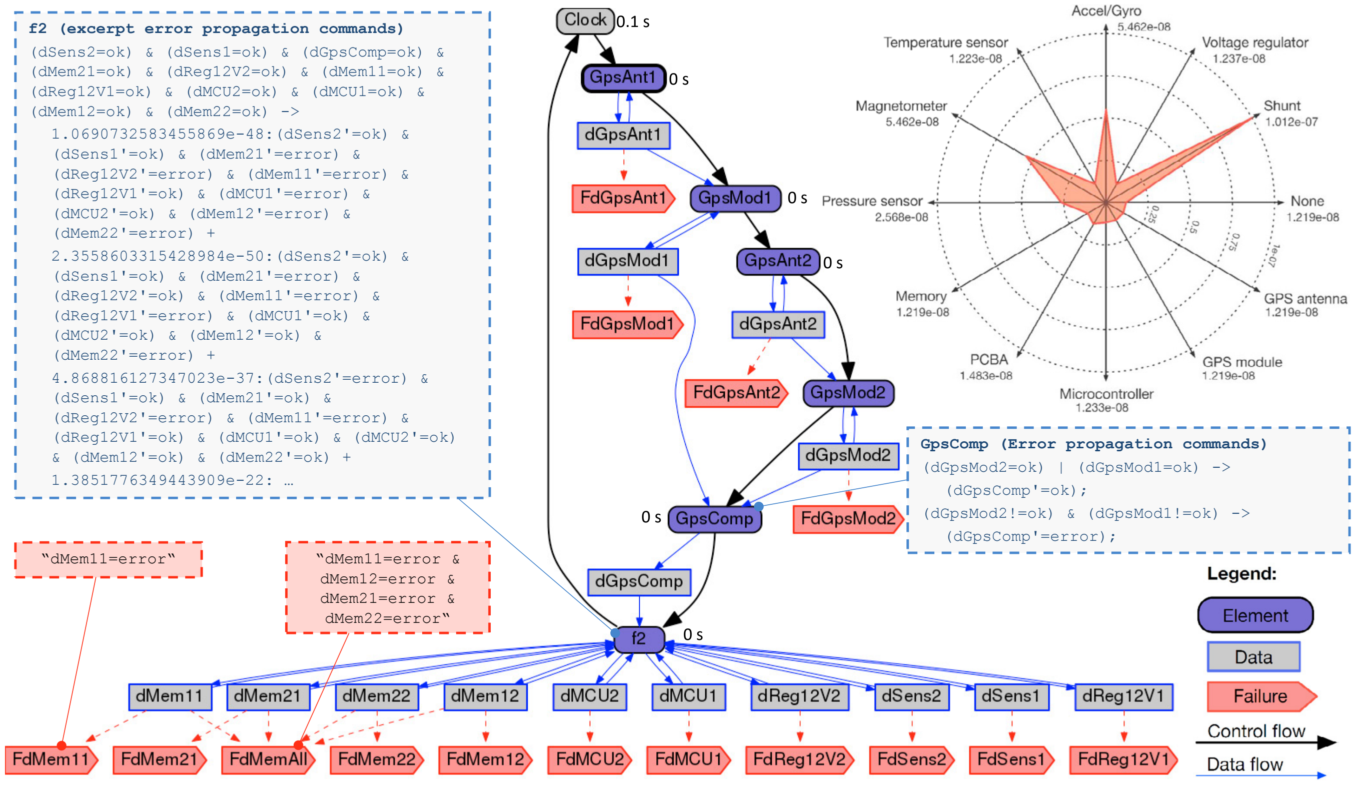
$$N_S\ [h^{-1}] = N_E \cdot f_i\ [Hz] \cdot 3600\ [\frac{s}{h}] = 252000\ h^{-1} \qquad (2)$$

Where f_i is the frequency of the i^{th} frequency class and N_E is the number of elements in the i^{th} frequency class. We set the execution time of all elements to zero and add the additional element *Clock* to the top-level DEPM and specify its execution time to 0.1 s that corresponds to the first frequency class.

5. DEPENDABILITY ANALYSIS

The next step is the numerical dependability analysis of the generated DEPM using OpenErrorPro. The tool automatically creates a PRISM DTMC model for the lowest hierarchical level, computes it using the built-in PRISM interface, and uses the results for the computation of the next hierarchical level. For instance, Fig. 5 shows the top-level DEPM with already computed second and third level DEPMs and defined system failures. The probabilistic commands of the element f_2 are automatically generated based on the submodels computation results. For example, the $FdMemAll$ = "$dMem11 = error$ & $dMem12 = error$ & $dMem21 = error$ & $dMem22 = error$" failure models simultaneous errors in all four memory modules. The computed probability of this failure for one hour mission time is equal to $2.07e^{-09}$.

After this verification, we performed an extensive DEPM-based sensitivity analysis of the *FdMemAll* failure to an increase of the failure rates of all component classes listed in Table 1. The results are shown in the last column of Table 1 and visualized with the diagram in Fig. 5. The axis "None" represents the originally computed *FdMemAll* probability.



```
f2 (excerpt error propagation commands)
(dSens2=ok) & (dSens1=ok) & (dGpsComp=ok) &
(dMem21=ok) & (dReg12V2=ok) & (dMem11=ok) &
(dReg12V1=ok) & (dMCU2=ok) & (dMCU1=ok) &
(dMem12=ok) & (dMem22=ok) ->
   1.0690732583455869e-48:(dSens2'=ok) &
   (dSens1'=ok) & (dMem21'=error) &
   (dReg12V2'=error) & (dMem11'=error) &
   (dReg12V1'=ok) & (dMCU1'=error) &
   (dMCU2'=ok) & (dMem12'=error) &
   (dMem22'=error) +
   2.3558603315428984e-50:(dSens2'=ok) &
   (dSens1'=ok) & (dMem21'=error) &
   (dReg12V2'=ok) & (dMem11'=error) &
   (dReg12V1'=error) & (dMCU1'=ok) &
   (dMCU2'=ok) & (dMem12'=ok) &
   (dMem22'=error) +
   4.868816127347023e-37:(dSens2'=error) &
   (dSens1'=ok) & (dMem21'=ok) &
   (dReg12V2'=error) & (dMem11'=error) &
   (dReg12V1'=ok) & (dMCU1'=ok) & (dMCU2'=ok)
   & (dMem12'=ok) & (dMem22'=ok) +
   1.3851776349443909e-22: …
```

```
GpsComp (Error propagation commands)
(dGpsMod2=ok) | (dGpsMod1=ok) ->
   (dGpsComp'=ok);
(dGpsMod2!=ok) & (dGpsMod1!=ok) ->
   (dGpsComp'=error);
```

Fig. 5. The top-level DEPM model for the frequency class f_1 with defined failures and the $P_{FdMemAll}(5h)$ with Individual tenfold increase of component failure rate.

The other axes show the probability of the *FdMemAll* failure after a tenfold increase of the corresponding component failure rate. Three significant deteriorations of the $P_{FdMemAll}(5h)$ appear in the component classes *Shunt*, *Accel/Gyro*, and *Magnetometer*. This indicates that these components shall not be replaced by less reliable ones. Vice versa, all other component classes are insensitive to a tenfold component failure rate increase. This type of sensitivity analysis is very useful for further development because it helps to reduce costs while preserving the required reliability level.

6. CHALLENGES

Challenge 1: We noticed that the PRISM model checker (v4.4) used by OpenErroPro can not process correctly the probabilities that are close to one (double-precision floating-point format). Extremely low probabilities of errors that are less than e^{-15} are processed correctly, but the opposite probabilities with more than 15 decimal places of nines are rounded to one. We have encountered this problem only for the third-level DEPM because the probabilities of error in the frequency class f_3 (1000000 Hz) are in the magnitude of e^{-19}. Therefore, the original component failure rates of the third-level DEPM components are multiplied by 10^5 and 10^6. The result of the computation of a sub DEPM is basically a number of probabilities for possible combinations of 'ok' and 'error' values of the external data outputs that are used on higher DEPM levels. After that, we compute factors for each combination as the ratio of the x10^6 and x10^5 results and use this factor for the computation of the real probabilities in the last column. Note that this method can be applied only if the relations between the component-level error probabilities and the computed probabilities are linear. This was checked and appeared to be true for the IMUFUSION system.

Challenge 2: The State Space Explosion (SSE) is the problem of all Markov chain based methods. OpenErrorPro and PRISM support several inherent optimizations. However, additional DEPM-level optimizations based on expert knowledge are required. The maximum possible number of DTMC states for a DEPM model is

$$S = E \cdot \prod_{i=1}^{D} V_i \tag{3}$$

Where E is the number of DEPM elements, D is the number of data storages, and V_i is the number of modeled values of the data storage i. The number of data storages is a crucial part of the equation because it causes exponential growth of the state space. For the computation of the DEPM shown in Fig. 5 with 7 elements, 15 data storages that can take values from {'ok','error'}, in the worst case, $7 \cdot 2^{15}$ DTMC states have to be generated. To cope with this problem the practical solution was to reduce the number of state data that are not taking part in failure definitions and are not directly relevant for the analysis. In particular, we identified sequential control flow structures where the state data of a preceding element is used only as input for a single succeeding element. After that, we removed the backward data flow arc from the state data to its parent element for all elements in the sequential structure except the last one. By doing this we reduced the number of state data preserving the same behavior of the model.

7. CONCLUSION

The article presents the practitioner experiences and lessons learned after the application of a recently introduced method for the model-based dependability analysis to an inertial navigation system for near-space vehicles. The case study system was developed within the context of the Balloon Experiments for University Students (BEXUS) campaign. The SysML UAV Dependability Profile (UDP) and the transformation algorithm from SysML to the Dual-graph Error Propagation Model (DEPM) are two major parts of this method. With the optimization against the state-space explosion of underlying Markov chain models and the introduction of a new computation algorithm for DEPMs with extremely low fault activation probabilities, it was demonstrated that the method copes with this type of mechatronic system and is appropriate for prospective tasks of dependability analysis in early development phases. The analytical results help to evaluate overall system reliability, probabilities of particular failures, and to identify critical components. The presented approach is systematic and partially automated. A future goal is the complete automation of the method, especially of the transformation step which will also increase the consistency. The DEPM capabilities allow more complex and realistic system analysis, that can be fully exploited with further investigations on possible extensions of the UDP.

REFERENCES

Ali, S., Basit-Ur-Rahim, M.A., and Arif, F. (2015). Formal verification of internal block diagram of sysml for modeling real-time system. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*, 1–6. IEEE.

Avižienis, A., Laprie, J.C., and Randell, B. (2004). Dependability and its threats: a taxonomy. In *Building the Information Society*, 91–120. Springer.

Baier, C. and Katoen, J.P. (2008). *Principles of model checking*. MIT press.

Baouya, A., Bennouar, D., Mohamed, O.A., and Ouchani, S. (2015). A probabilistic and timed verification approach of sysml state machine diagram. In *Programming and Systems (ISPS), 2015 12th International Symposium on*, 1–9. IEEE.

BEXUS (2018). Balloon experiments for university students. URL `http://rexusbexus.net/bexus/`.

Debbabi, M., Hassaïne, F., Jarraya, Y., Soeanu, A., and Alawneh, L. (2010). *Verification and Validation in Systems Engineering - Assessing UML / SysML Design Models.*

Dehnert, C., Junges, S., Katoen, J.P., and Volk, M. (2017). A storm is coming: A modern probabilistic model checker. In *International Conference on Computer Aided Verification*, 592–600. Springer.

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

Feiler, P.H., Gluch, D.P., and Hudak, J.J. (2006). The architecture analysis & design language (aadl): An introduction. Technical report, Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

Guide, F. et al. (2009). Reliability methodology for electronic systems. *FIDES group.*

Jarraya, Y., Soeanu, A., Debbabi, M., and Hassaine, F. (2007). Automatic verification and performance analysis of time-constrained sysml activity diagrams. In *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, 515–522. IEEE.

Kim, M.C. (2011). Reliability block diagram with general gates and its application to system reliability analysis. *Annals of Nuclear Energy*, 38(11), 2456–2461.

Kwiatkowska, M., Norman, G., and Parker, D. (2002). Prism: Probabilistic symbolic model checker. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, 200–204. Springer.

Lanusse, A., Tanguy, Y., Espinoza, H., Mraidha, C., Gerard, S., Tessier, P., Schnekenburger, R., Dubois, H., and Terrier, F. (2009). Papyrus uml: an open source toolset for mda. In *Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009)*, 1–4.

Machida, F., Xiang, J., Tadano, K., and Maeno, Y. (2013). Composing hierarchical stochastic model from sysml for system availability analysis. In *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, 51–60. IEEE.

Mhenni, F., Nguyen, N., and Choley, J.Y. (2014). Automatic fault tree generation from sysml system models. In *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*, 715–720. IEEE.

Morozov, A., Tuk, R., and Janschek, K. (2015). Errorpro: Software tool for stochastic error propagation analysis. In *1st International Workshop on Resiliency in Embedded Electronic Systems, Amsterdam, The Netherlands*, 59–60.

Morozov, A. and Janschek, K. (2014). Probabilistic error propagation model for mechatronic systems. *Mechatronics*, 24(8), 1189–1202.

OMG, H. (2008). Omg systems modeling language (omg sysml)-version 1.4. *Juni-2015.*

Ong, C.M. (1998). *Dynamic simulation of electric machinery: using MATLAB/SIMULINK*, volume 5. Prentice hall PTR Upper Saddle River, NJ.

Ouchani, S., Mohamed, O.A., and Debbabi, M. (2014). A formal verification framework for sysml activity diagrams. *Expert Systems with Applications*, 41(6), 2713–2728.

REXUS/BEXUS (2018). Rexus/bexus rocket and balloon experiments for university students. URL `http://rexusbexus.net/`.

Ruijters, E. and Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer science review*, 15, 29–62.

Steurer, M., Morozov, A., Janschek, K., and Neitzke, K.P. (2018). Sysml-based profile for dependable uav design. *IFAC-PapersOnLine*, 51(24), 1067–1074.

UML, O. (2001). Unified modeling language. *Object Management Group.*