Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Model-Based Dependability Assessment of Phased-Mission Unmanned Aerial Vehicles

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:
Assessment of non-functional reliability and safety requirements in the early development phases helps to prevent conceptually wrong decisions and, as a consequence, significantly reduces overall development costs. The application of model-based system analysis techniques demonstrates promising results for complex avionics systems, especially software-intensive Unmanned Aerial Vehicles (UAV). Such systems are commonly designed to accomplish a specific mission consisting of multiple mission phases. The concept of phased mission systems enables the specification of individual requirements for different phases. For instance, the reliability requirements or system specifications are different for UAV flights over an agricultural field and a highway. Therefore, modern analytical methods have to distinguish between different mission phases and enable the analysis of phased missions. In this paper, we propose a new model-based method that allows system engineers to assess a conceptional design specification of the UAV concerning the fulfillment of phase-specific requirements. The proposed approach exploits modern probabilistic model checking techniques for the quantification of several dependability metrics. The method supports the systematic analysis of system specifications that contain both structural and behavioral system properties. A case study demonstrates the feasibility of the proposed method.

Keywords: Design methodologies, Flying robots, Error probability, Markov models, Reliability analysis, Safety analysis, Stochastic modeling, System analysis, Systems engineering

1. INTRODUCTION

UAVs are getting more complex with the increase in the number of various interacting components. The ascending complexity of such systems makes consistent development and communication between individual engineering disciplines more and more difficult. The Model-Based Systems Engineering (MBSE) concept implies that models are used as an integral part of the technical baseline [Bergenthal (2011)]. The Systems Modeling Language (SysML) is a powerful modeling paradigm that also supports the requirements engineering for a broad range of complex technical systems [OMG (2017)]. SysML is commonly used by engineers of various disciplines and offers the possibility to extend the semantics of the language with profiles.

UAV system failures can cause catastrophic consequences. Hence, UAVs are commonly treated as safety-critical systems and have to satisfy high dependability requirements. They shall be developed according to safety standards like the IEC 61508 [IEC (2010)], which concerns all systems based on electric, electronics, and programmable electronics, or the SAE ARP 4754 [SAE (2010)] and 4761 [SAE (1996)] for civil aircraft and systems.

These standards recommend reliability evaluation methods like the Fault Tree Analysis (FTA) [Vesely et al. (1981)] or Failure Mode and Effects Analysis (FMEA) [Stamatis (2003)]. The integration of safety analysis into MBSE is called Model-Based Safety Analysis (MBSA) [Mhenni et al. (2016)]. However, the mentioned methods are too simple for practical systems. They cannot model common dependability patterns, such as spare management or functional dependencies between system components, and complex dynamic behaviors. Methods like the Dynamic Fault Trees [Dugan et al. (1992)] partially support these features. However, they are also limited and not commonly used in the industry. Discrete-Time Markov Chain (DTMC) and Probabilistic Model Checking (PMC) [Baier and Katoen (2008)] methods are more flexible in general. We will focus on model-based DTMC techniques to assess the deterministic and probabilistic dependability requirements of UAVs. Using PMC provides the user with in-depth analysis capabilities.

The so-called Phased-Mission Systems (PMS) accomplish a specified task during multiple, consecutive, non-overlapping phases of operation [Alam and Al-Saggaf (1986)]. Thus, the system configuration, requirements, and/ or failure behavior are phase-specific [Xing and Dugan (2002)].

In this paper, we propose a new probabilistic dependability assessment approach that supports the systematic analysis of SysML system specification against non-functional dependability requirements of UAV-PMS. The remainder of this paper is structured as follows. Section 2 discusses the state of the art MBSA approaches and the dependability assessment of PMS. Section 3 introduces our method and shows a detailed insight into the subareas model-based requirements engineering, mission-definition, transformation, and assessment. In Section 4, a case study using our method is shown to clarify the principle. Finally, Section 5 summarizes the content of the paper.

2. STATE OF THE ART

Methods based on norms and standards: According to IEC 61508, the safety integrity level verification is a necessary procedure of the safety life cycle where the probability of failure on demand must be calculated. Since the IEC 61508 does not provide detailed information for the calculation, Guo and Yang (2007) investigate a method using a reliability block diagram to calculate the probability of failure on demand of the designed system. The works of Legendre et al. (2017) and Abdellatif and Holzapfel (2019) refer to the requirement fulfillment of the SAE ARP 4754 and 4761. The approach of Mhenni et al. (2016) studies the two compositional safety techniques recommended by all three standards mentioned above. Furthermore, they integrate both safety techniques into the MBSE using SysML baseline models.

Model-based fault tree analysis: Most of the works related to the MBSA are focused on the generation of Fault Trees or the FMEA. In the MBSE domain, multiple metamodels and languages support the modeling of the required system architecture, including composition and internal connections. The best-known representatives are the Architectural Analysis and Design Language (AADL), MATLAB/Simulink, Unified Modeling Language (UML), and SysML. In Joshi et al. (2007), the AADL, in combination with the Error Model Annex, is used for the generation of static fault trees, taking into account component redundancies. Tajarrod and Latif-Shabgahi (2008) present a methodology for the construction of static and dynamic fault trees from a MATLAB/Simulink system model. In Xiang et al. (2011) static fault trees are automatically derived from the SysML system model to bridge the gap between reliability and systems engineering. Yakymets et al. (2013) describes a safety modeling framework for fault tree generation that leverages features of SysML, including facilities for semantic connections with formal verification and FTA tools. The work of Castet et al. (2018) describes an approach that allows system engineers to capture failure-related information from a SysML model to automatically generate the FMECA and FTA and takes into account high-level mission phases. In Nordmann and Munk (2018), prominent approaches for the facilitation of SysML models with component fault trees to support the FTA are adapted to propose an integration of component fault trees with SysML Internal Block Diagrams (IBD) as well as Activity Diagrams (ACT).

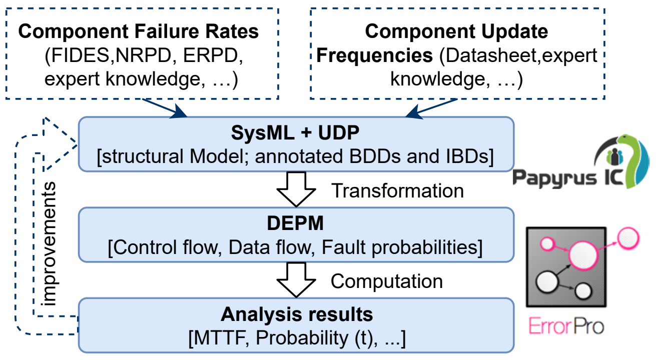
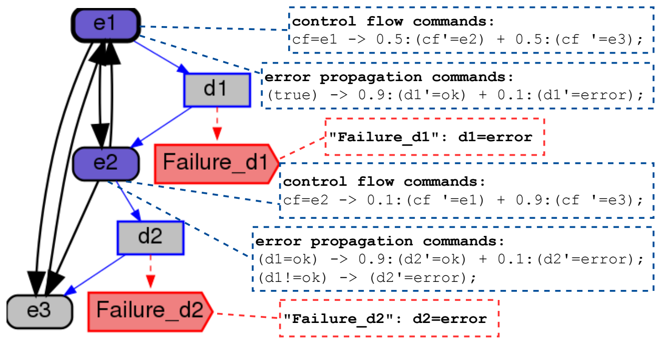

Component Failure Rates (FIDES,NRPD, ERPD, expert knowledge, ...)
Component Update Frequencies (Datasheet,expert knowledge, ...)
SysML + UDP [structural Model; annotated BDDs and IBDs]
Papyrus IC
Transformation
DEPM [Control flow, Data flow, Fault probabilities]
Computation
Analysis results [MTTF, Probability (t), ...]
ErrorPro
improvements

Fig. 1. Method workflow including the UAV dependability profile [adapted from Steurer et al. (2019)].

Model checking: However, UML/ SysML is not solely used as a basis for the generation of fault trees or FMEA. Machida et al. (2013) describes the generation of the hierarchical stochastic model, which is essentially the combination of state models and combinatorial models. Ghezzi and Sharifloo (2013) describes how to use probabilistic model checking techniques and tools to verify the non-functional properties of different configurations of a software product line. The product behavior is modeled with UML sequence diagrams. These diagrams are annotated with execution probability and energy and transformed into Markov models, which, in turn, are fed to a probabilistic model checker. Wang and Cai (2017) combines the probabilistic model checking and reliability analysis for flight control systems. The flight control system is described as a continuous-time Markov chain using the PRISM [Kwiatkowska et al. (2011)] language that represents the system behavior. They also provide rules for obtaining temporal logical formulas from reliability requirements to check quantitative properties using model checking.

Dependability analysis of phased-mission systems: A UAV mission often consists of several phases, in which different environmental influences prevail, and different requirements are set. This fact led to the establishment of PMS, which also require a dependability assessment. In Alam and Al-Saggaf (1986), a quantitative reliability model for a PMS using a Markov process is introduced. A mission profile table characterizes the whole mission. Within the mission profile table, a reliability block diagram describes the system logic configuration. The work of La Band and Andrews (2004) describes the use of FTA for PMS with non-repairable components. Binary decision diagrams are employed to quantify the likelihood of failure in each phase.

UAV-specific model-based dependability analysis: In our previous work [Steurer et al. (2018)] we introduced: (i) the new domain-specific SysML UAV Dependability Profile (UDP) that captures reliability-related properties of UAV components, and (ii) the transformation algorithm from profiled SysML models to the formal Dual-graph Error Propagation Model (DEPM) [Morozov and Janschek (2014)] for further extensive dependability analysis. The SysML was chosen because of the extensive modeling capabilities, inherent extensibility mechanisms (profiles), available modeling tools, and the XML metadata interchange storage and exchange format. We also evaluated



```
control flow commands:
cf=e1 -> 0.5:(cf'=e2) + 0.5:(cf '=e3);

error propagation commands:
(true) -> 0.9:(d1'=ok) + 0.1:(d1'=error);

"Failure_d1": d1=error

control flow commands:
cf=e2 -> 0.1:(cf '=e1) + 0.9:(cf '=e3);

error propagation commands:
(d1=ok) -> 0.9:(d2'=ok) + 0.1:(d2'=error);
(d1!=ok) -> (d2'=error);

"Failure_d2": d2=error
```

Fig. 2. Example dual-graph error propagation model.

the applicability of this method during the early development phases of an inertial navigation system [Steurer et al. (2019)]. The UDP method illustrated in Figure 1 allows adding reliability relevant information (component failure rates and update frequencies) to a structural model created using SysML Block Definition Diagrams (BDD) and IBD. The component failure rates are taken from guidelines such as FIDES [Guide et al. (2009)], ERPD [Denson et al. (1996)], or NRPD [Denson et al. (1994)]. The update frequencies of the time-discrete components can be found in the datasheets. Non-discrete components like motors are time-discretized to the highest frequency used in the model. Since adding the failure rates and frequencies is the user's task, experience, and expert knowledge can be integrated. To realize these features we used the UML and SysML modeling tool Papyrus [Lanusse et al. (2009)]. Afterward, the model, which is available in XML metadata interchange format, is converted into a DEPM. Once the DEPM is available, the required dependability metrics can be evaluated.

Dual-graph Error Propagation Model: The DEPM captures system control 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. Figure 2 shows a DEPM example. 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: e1, e2, e3) and data storages (rectangles with blue borders: d1, d2). Control flow arcs (black lines) model control flow transitions between the elements, e.g., after the execution of e1, we will execute either e2 or e3. Data flow arcs (blue lines) model data transfer between the elements and data storage. Elements connected to more than one outgoing control flow arc contain control flow commands that specify the triggering logic. Elements connected to data storages via data flow contain error propagation commands. The error propagation commands are probabilistic conditions that specify fault activation and error propagation of the elements. The reliability metrics are computed for the defined failures (highlighted in red) using automatically generated DTMC models. DTMC model states describe the current control flow state (which element will be executed next) and the states of all data storage. Technically a DEPM is transformed into one or several PRISM models for numerical evaluation with stochastic model checkers like PRISM or STORM [Dehnert et al. (2017)].

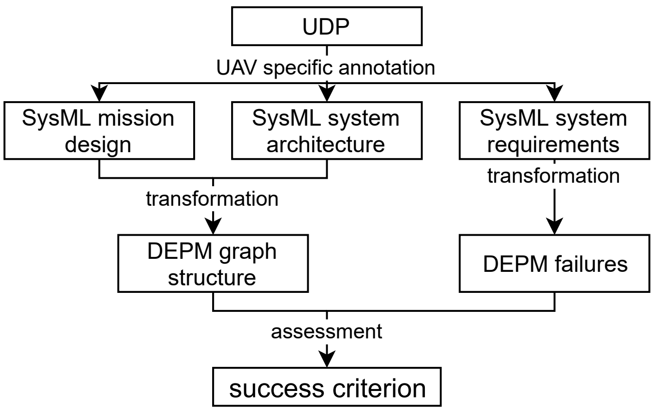


Fig. 3. Overview of the proposed approach.

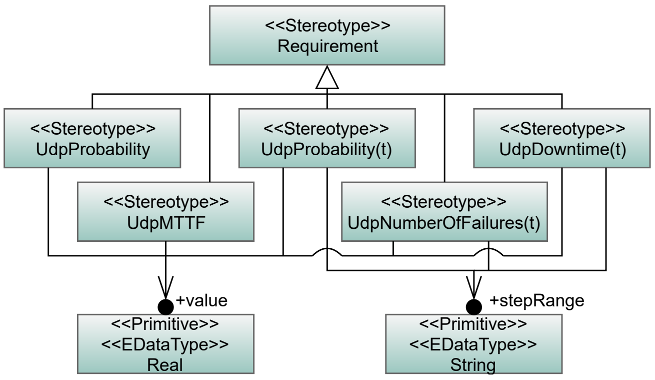


Fig. 4. Profile diagram with spezialized UDP requirements.

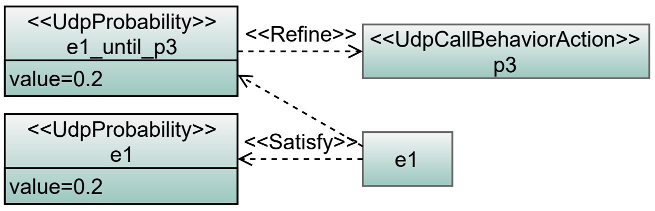


Fig. 5. Allocation of the profiled requirements to satisfying parts and associated mission phases.

Contribution: In this paper, we propose a new probabilistic dependability assessment approach. A SysML system specification is analyzed systematically regarding the non-functional dependability requirements of UAV-PMS. To assess mission phase varying requirements, we extend the UDP with the requirement and behavior modeling by the integration of SysML ACT and Requirement Diagrams (REQ). Furthermore, we extend the transformation method to DEPM, which is also discussed in this paper. Based on our case study, the development of a quadcopter PMS, the assessment is exemplified.

3. APPROACH

3.1 Overview

In most cases, UAVs are developed to accomplish specific missions that consist of individual mission phases. Different functionality and dependability requirements can be defined for different mission phases. Also, the UAV is subject to different thermal or mechanical loads during the various mission phases. With these information available, more realistic and trustworthy analysis results are achieved instead of mediating the requirements and loads throughout the mission. The structure of the proposed approach is shown in Figure 3. To apply the proposed approach:

- The requirements to be met shall be known.
- The mission to be accomplished shall be known.
- The system architecture concept shall be known.

As mentioned above, the basic method for the analysis of system architectures already exists. In this paper, we extend the UDP to enable the model-based assessment of phase-specific success criteria.

3.2 SysML model

System requirements: The tool OpenErrorPro [Morozov et al. (2019)] that supports the DEPM methods allows the user to compute the common reliability metrics *Probability of failure*, *MTTF*, *Number of failures*, and *Downtime*. As shown in Figure 4, the stereotype *requirement* is further specialized for the transformation to DEPM using the UDP. The specialized stereotypes are extended with the property *value*, which specifies the maximum allowed quantity. All time-dependent specializations e.g. *Probability (t)* are additionally enhanced with the property *stepRange* that is directly usable in DEPM. The proposed method also requires the formalization of the requirements engineering using the REQ. An REQ example is shown in Figure 5. Using specialized UDP stereotypes, recursively defines DEPM verification.

The SysML provides the *refine* and the *derive* relationship to relate the requirements to defined mission phases. The *refine* relationship describes how a model element, or set of elements, further refines a requirement. It differs from the *derive* relationship in that it can exist between a requirement and any other model element. A requirement is assessed for all associated phases. In the example, shown in Figure 5, *e1_before_p3* is related to *p3*, which means that the specified value shall be higher than the computed one before the start of *p3*. All requirements related to a phase are assessed.

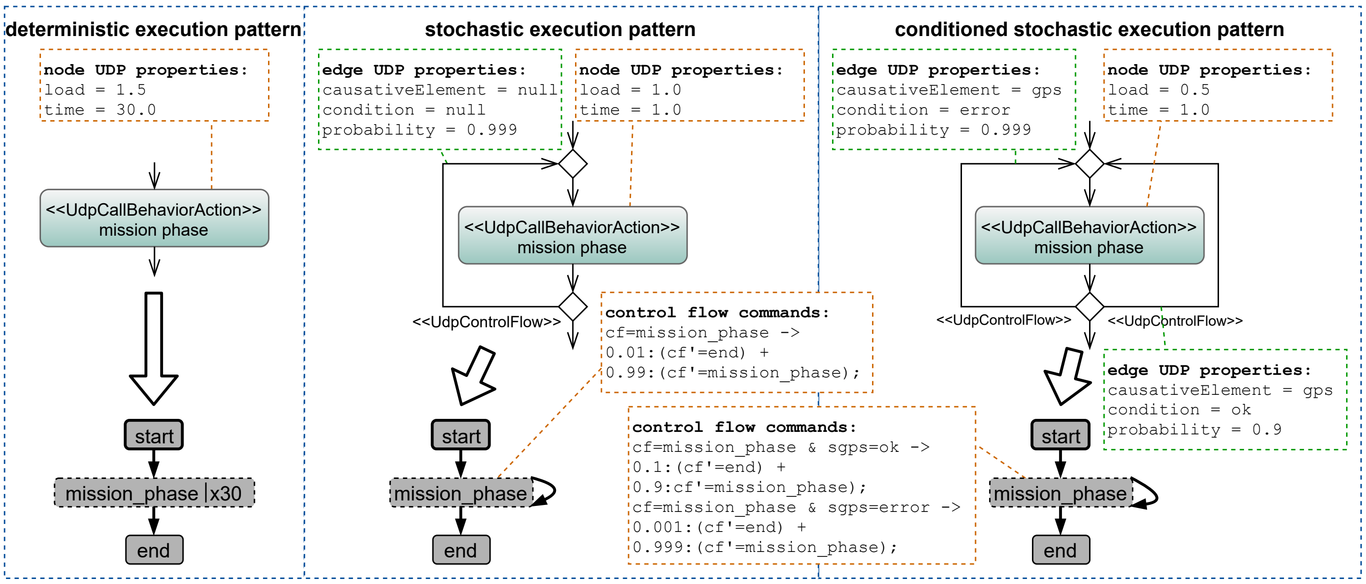
The SysML provides the *satisfy* relationship to relate requirements to the parts that shall satisfy these requirements. It describes how a model element satisfies one or more requirements specified by the system designer. These relationships are not formal inherently and can be interpreted in different ways. If one part is related to more than one requirement, this part shall satisfy all requirements. If one requirement is related to more than one part, all parts shall satisfy this requirement. The same applies to parts with a multiplicity higher than one.

Mission design: The SysML offers the activity, state machine, and sequence diagrams to model system behavior. We choose the ACT because the available nodes can be assigned to the individual mission phases and related to the defined requirements using the refine relationship. The UDP specializes the *CallBehaviorAction* stereotype to the *UdpCallBehaviorAction* stereotype with annotated *load:Real* and *time:Real* properties. Using the UDP, mission phase-specific environmental conditions or system parameters can also be taken into consideration. Adverse environmental conditions or high system loads, for example, because of highly dynamic flight maneuvers, increase the *load* property, which serves as a coefficient for the component failure rate specified in the structural model. The *time* property specifies the planned time of the corresponding phase if a deterministic phase length is to be modeled. Furthermore, the UDP extends the *ControlFlow* stereotype to the *UdpControlFlow* stereotype with annotated *causativeElement:Property*, *condition:Condition*, and *probability:Real* properties. These properties are annotated to formalize the modeling of stochastic and conditional stochastic processes for later transformation. Figure 6 shows a *UdpControlFlow* that connects the decision node after a mission phase with a merge node before the mission phase. If the *probability* property is defined, the control flow transition is activated only with this probability, which results in a stochastic mission phase time. A *causativeElement* together with the *condition* act as a control flow guard. The *causativeElement* determines the part and the *condition* a possible state of the part. The control flow transition is activated if the part is in the defined state, which results in a conditioned stochastic mission phase time.

System Architecture: In Steurer et al. (2018), we already described how the architectural properties are modeled using the BDD and IBD. An essential extension, however, is the use of the *boundReference* stereotype to model mission phase-specific system configurations. As shown in Figure 7, which represents the top-level BDD of the case study, the *boundReference* is a property of a general abstract block and its specializations. The type is equal to the block of a composite part, but the multiplicity specifies the quantity of the subpart. The allocation of the system configuration that shall be active in the specific phase is carried out by assigning the desired activity to the *operation* property of the block, describing the configuration.

3.3 Transformation and assessment

As shown in Figure 7, the designed mission is mapped to a top-level DEPM where each element represents one phase. The state represents a set of state data. These state data are the connection to the sub-levels that represent the system architecture active in this phase, where each element represents one system component. The load represents a set of mission phase loads specified in the SysML model that influences the error propagation commands of the component. The component error probability is calculated as described in Steurer et al. (2018) but additionally multiplied by the load coefficient depending on the phase. This extension results in different component error probabilities in different phases. From the behavior modeled in SysML, the mission to be executed is derived. As shown in Figure 6, the transformation follows three different patterns. A specified SysML mission phase *time* property is transformed into repetitions of the DEPM element that represents this phase. Using a *UdpControlFlow* with specified *probability* property the pattern is transformed to a DEPM with the corresponding control flow structure. The *probability* property influences the control flow commands of the corresponding element. As described above, the *UdpControlFlow* properties *causativeElement*


deterministic execution pattern
stochastic execution pattern
conditioned stochastic execution pattern
<<UdpCallBehaviorAction>> mission phase
<<UdpControlFlow>>
start
mission_phase |x30
mission_phase
end

```
node UDP properties:
load = 1.5
time = 30.0
```

```
edge UDP properties:
causativeElement = null
condition = null
probability = 0.999
```

```
node UDP properties:
load = 1.0
time = 1.0
```

```
control flow commands:
cf=mission_phase ->
0.01:(cf'=end) +
0.99:(cf'=mission_phase);
```

```
edge UDP properties:
causativeElement = gps
condition = error
probability = 0.999
```

```
node UDP properties:
load = 0.5
time = 1.0
```

```
edge UDP properties:
causativeElement = gps
condition = ok
probability = 0.9
```

```
control flow commands:
cf=mission_phase & sgps=ok ->
0.1:(cf'=end) +
0.9:cf'=mission_phase);
cf=mission_phase & sgps=error ->
0.001:(cf'=end) +
0.999:(cf'=mission_phase);
```

Fig. 6. Deterministic, stochastic, and conditioned stochastic activity diagram mission patterns and their DEPM counterpart.

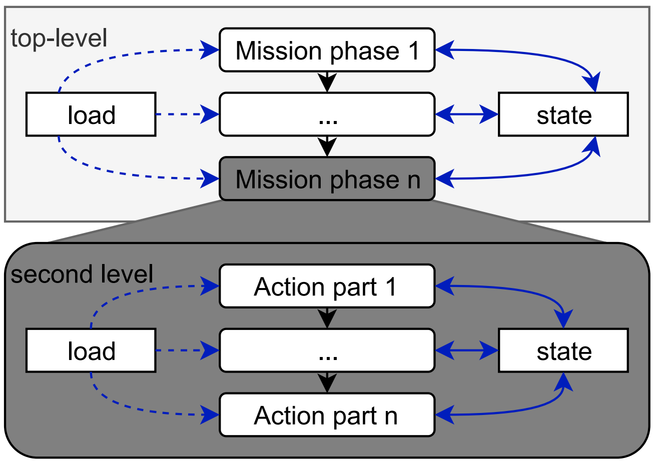
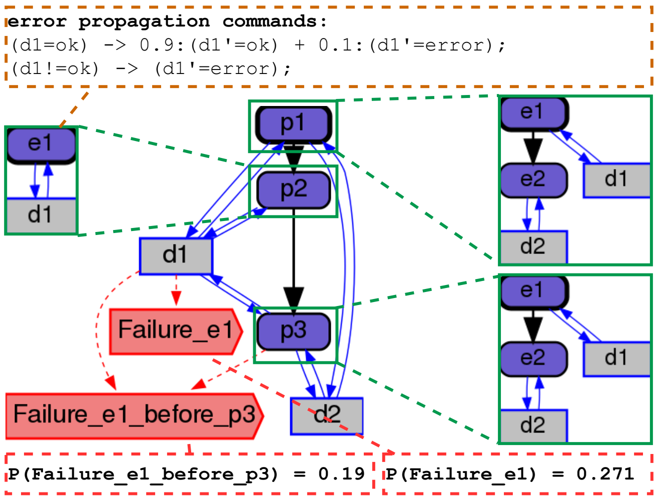


Fig. 7. The resulting structure of the DEPM.

and *condition* act like a guard. Together with the *probability* property, a conditioned stochastic process is the consequence.

After setting up the DEPM structure, the requirements are integrated. All UDP-specific requirements are transformed into failures in DEPM. All state data of each element are propagated to the top-level to ensure data exchange between mission phases. That top-level propagation allows the automatic component state storage between phases, even if not used in the meantime. A UDP-specific requirement in SysML can be related to one or more parts and one or more phases. That leads to DEPM failures belonging to the state data of the element related to the part and the control flow related to the phase. If we consider the REQ shown in Figure 5 the resulting DEPM is shown in Figure 8. The top-level shows three sequential executed phases. In mission phases one and three, the same components are used. In phase two, only the *e1* is in use. For simplification, the error propagation commands of all contained elements correspond to the illustration. The two requirements are transformed into the DEPM failures



```
error propagation commands:
(d1=ok) -> 0.9:(d1'=ok) + 0.1:(d1'=error);
(d1!=ok) -> (d1'=error);
```

```
P(Failure_e1_before_p3) = 0.19
P(Failure_e1) = 0.271
```

Fig. 8. Nested DEPM with SysML requirements transformed to DEPM failures.

Failure_e1 and *Failure_e1_before_p3*. *Failure_e1* is just depending on the state of element *e1*, *Failure_e1_before_p3* additionally on the outgoing control flow of *p2* which means the probability of this failure before the execution of *p3* is computed. The results are shown at the bottom of the figure. As expected, the probability of an error, before *p3* is performed, is lower.

4. CASE STUDY

4.1 SysML model

Mission design: The case study mission represents a realistic delivery mission. The UAV shall take off from a place outside and land inside a safety-critical area. After entering the safety-critical area, e.g., an urban area, the UAV shall search for a spot to land. The GPS module helps to find a landing spot. If the GPS module is already erroneous in the phase before entering the safety-critical area, the system shall not search for a landing spot

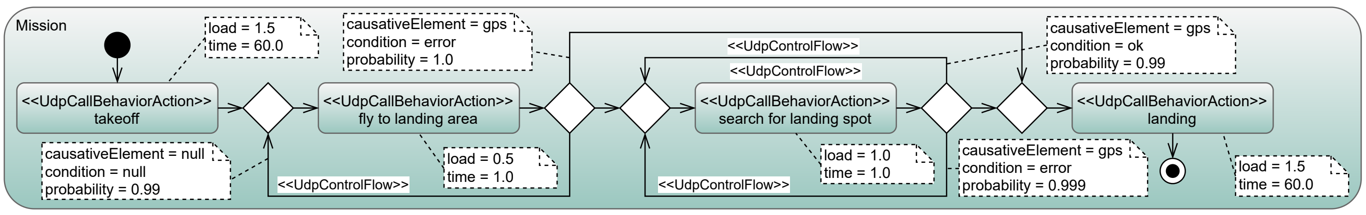


Fig. 9. Example delivery mission with four mission phases.

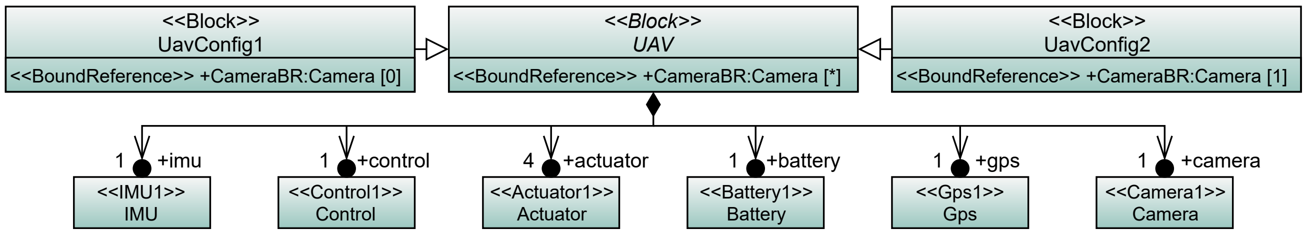


Fig. 10. Block definition diagram for modeling the mission phase altering configurations via bound reference.

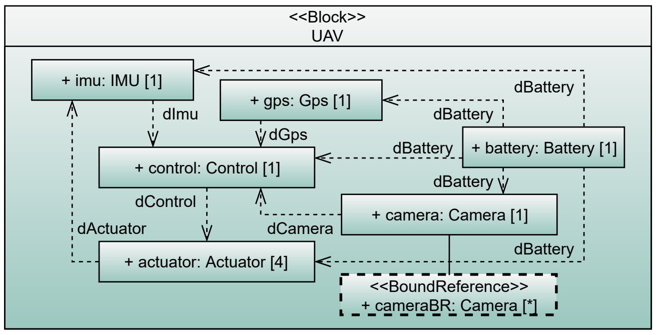


Fig. 11. Internal block diagram of abstract generalized block with binding connector to bound reference.

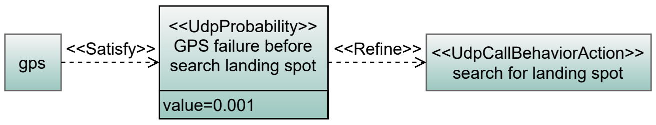


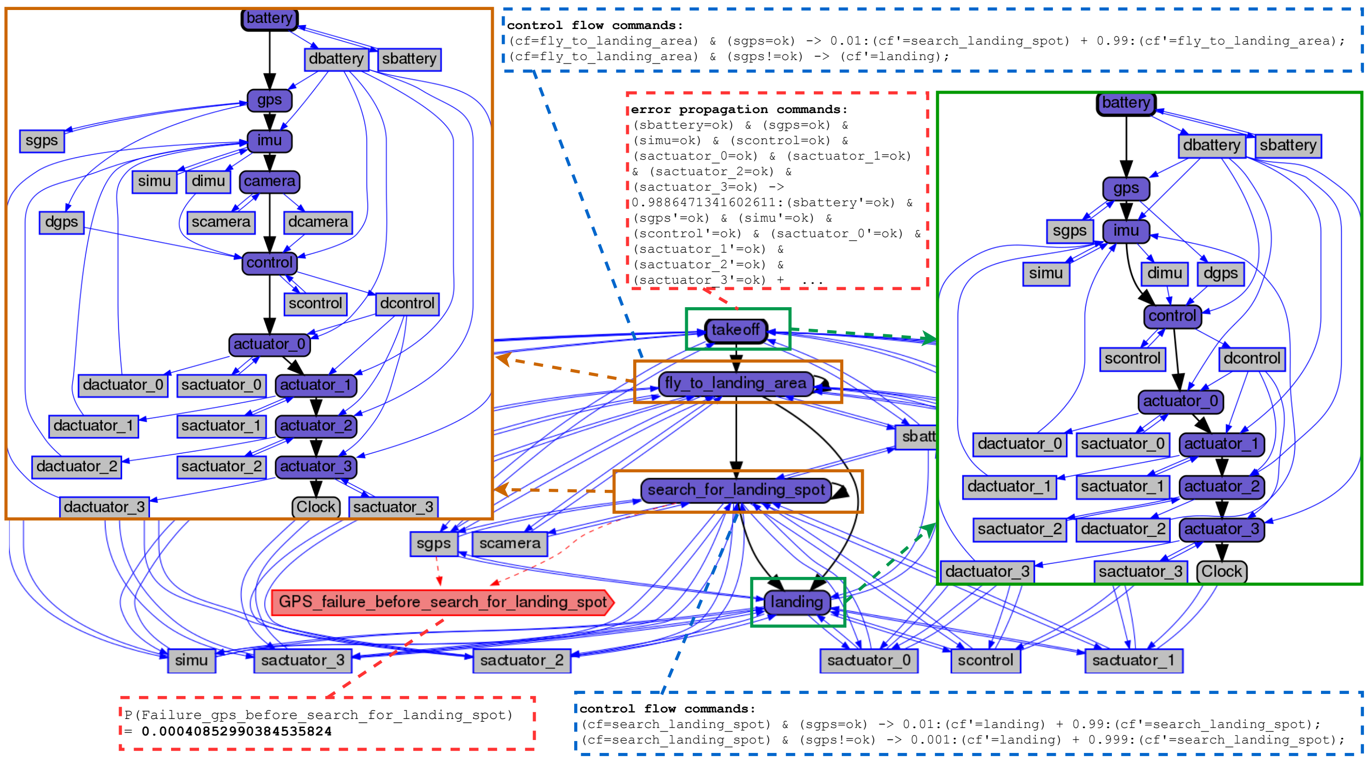
Fig. 12. Requirements diagram necessary to assess the representative requirement example.

but initiate the landing process directly. The case study mission modeled with an ACT is shown in Figure 9. The *time* properties of *takeoff* and *landing* are equal to 60. Thus, these phases shall last 60 time units. The outgoing control flow of *fly to landing area* is connected to a decision node with two outgoing *UdpControlFlows* and one *ControlFlow*. The first *UdpControlFlow* returns to the merge node before phase *fly to landing area* and the property probability is equal to 0.99. In other words, this phase is executed 100 times on average. The duration of one execution is equal to one time unit. Thus, the average time required to *fly to landing area* is equal to 100 time units. The properties *causativeElement* and condition are not specified and ignored. The second *UdpControlFlow* leads to the merge node before the phase *landing* and the property *probability* is equal to one. The property *causativeElement* is *gps* and the *condition* is error what means that this flow is active with a probability of one if the part *gps* takes the state error.

Similar properties are defined for the two *UdpControlFlows* after phase *search for landing spot*. That the property *causativeElement* is defined as *gps* applies to both *UdpControlFlows*. However, the first *UdpControlFlow* specifies the property *condition* as ok and probability as 0.99 and the second specifies the *condition* as error and probability as 0.999. Generally spoken, this means that if the *gps* is working, the phase is completed faster than if the *gps* is erroneous. As mentioned above, phase-specific load coefficients are specified depending on the environmental conditions and the dynamic stress on the system. We specified the load coefficients as 1.5 for *takeoff*, 0.5 for *fly to landing area*, 1.0 for *search for landing spot*, and 1.5 for *landing*.

System Architecture: Figure 10 shows the BDD of the case study system. The block UAV is modeled as abstract and contains the *BoundReference CameraBR:Camera* with unspecified multiplicity. The non-abstract specializations *UavConfig1* and *UavConfig2*, shown in Figure 10, also contain the *CameraBR BoundReference* but with specified multiplicity. Figure 11 represents the IBD of the case study system where a *BindingConnector* between the camera part and the *BoundReference* is shown. This construct models different system configurations for different mission phases. Only if *UavConfig2* is active the *camera* sends data to the part *control:Control*. The allocation of the configuration to the intended mission phase is done by assigning the called activity to the *ownedOperation* of the configuration block. We have assigned the activities of *takeoff* and *landing* to *UavConfig1* and the activities of *fly to landing area* and *search for landing spot* to *UavConfig2*. This means that in phases *takeoff* and *landing* no camera is utilized.

System requirements: "The probability that the GPS module will fail before the beginning of the search for landing spot phase shall be less than 0.001" is our requirement in question. This requirement is interesting because a failure of the GPS module affects the execution time of *search for landing spot*. Figure 12 shows the REQ required for this purpose, that contains the *GPS failure before search for landing spot* modeled element of stereotype *UdpProbability*. The *refine* relationship, applied to the



```
control flow commands:
(cf=fly_to_landing_area) & (sgps=ok) -> 0.01:(cf'=search_landing_spot) + 0.99:(cf'=fly_to_landing_area);
(cf=fly_to_landing_area) & (sgps!=ok) -> (cf'=landing);
```

```
error propagation commands:
(sbattery=ok) & (sgps=ok) &
(simu=ok) & (scontrol=ok) &
(sactuator_0=ok) & (sactuator_1=ok)
& (sactuator_2=ok) &
(sactuator_3=ok) ->
0.9886471341602611:(sbattery'=ok) &
(sgps'=ok) & (simu'=ok) &
(scontrol'=ok) & (sactuator_0'=ok) &
(sactuator_1'=ok) &
(sactuator_2'=ok) &
(sactuator_3'=ok) +  ...
```

```
P(Failure_gps_before_search_for_landing_spot)
= 0.00040852990384535824
```

```
control flow commands:
(cf=search_landing_spot) & (sgps=ok) -> 0.01:(cf'=landing) + 0.99:(cf'=search_landing_spot);
(cf=search_landing_spot) & (sgps!=ok) -> 0.001:(cf'=landing) + 0.999:(cf'=search_landing_spot);
```

Fig. 13. Generated dual-graph error propagation model with top-level mission and phase varied nested system architectures.

GPS failure before search for landing spot requirement, defines that the requirement is valid before the *search for landing spot* mission phase. The part *gps* shall *satisfy* this requirement.

4.2 Transformation and assessment

The extended SysML diagrams discussed above contain all necessary information for the generation of a DEPM using the method explained in Section 3. The generated DEPM is shown in Figure 13. The DEPM top-level represent the mission and the lower levels of the system configurations of the respective mission phases. The elements corresponding to the phases *takeoff* and *landing* contain a system configuration without camera and the elements corresponding to the phases *fly to landing area* and *search for landing spot* contain a system configuration with camera. The phases *takeoff* and *landing* shall last 60 time units. Therefore, the corresponding DEPM elements are modeled using the repetition mechanism of the DEPM. The DEPM element is executed several times, successively before the control flow transition to the next element is activated. A subset of the calculated error propagation commands of *takeoff* is also shown in Figure 13. The shown control flow commands result from the information of the SysML mission ACT. The modeled requirements appear as DEPM failures. In this case study the *UdpProbability* requirement *Failure gps before search for landing spot* is transformed into the *Failure_gps_before_search_for_landing_spot*. The associated conditions were generated from the SysML model and are shown at the lower part of the figure. The computed probability (0.0004085) of this failure is shown at the lower part of the figure as well. The computed probability is less than the required 0.001. Thus, the success criterion of the case study is fulfilled.

5. CONCLUSION

In this paper, we proposed a probabilistic dependability assessment approach that supports the systematic analysis of system specifications. The focus was to assess mission phase varying non-functional dependability requirements of phased-mission Unmanned Aerial Vehicles (UAV) under development. Therefore, we extended the UAV Dependability Profile, presented in our previous work, by configuration, requirement, and behavioral modeling. As a consequence, the approach is not limited to the static system topology, provided by SysML Block Definition Diagrams and Internal Block Diagrams, but also supports dynamic mission aspects by the integration SysML Activity Diagrams and Requirements Diagrams. Furthermore, we extended the transformation method to the Dual-graph Error Propagation Model and assessed quantitative and probabilistic requirements for deterministic, stochastic, or conditionally executable mission-phase times. To illustrate the feasibility and benefit of our approach, we assessed a representative requirement within a case study.

REFERENCES

Abdellatif, A.A. and Holzapfel, F. (2019). New methodology for model-based safety analysis. In *2019 IEEE Aerospace Conference*, 1–7. IEEE.

Alam, M. and Al-Saggaf, U.M. (1986). Quantitative reliability evaluation of repairable phased-mission systems using markov approach. *IEEE Transactions on Reliability*, 35(5), 498–503.

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

Bergenthal, J. (2011). Final report model based engineering (mbe) subcommittee. *National Defense Industrial Association, Arlington, VA.*

Castet, J.F., Bareh, M., Nunes, J., Okon, S., Garner, L., Chacko, E., and Izygon, M. (2018). Failure analysis and products in a model-based environment. In *2018 IEEE Aerospace Conference*, 1–13. IEEE.

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.

Denson, W., Jaworski, P., Crowell, W., and Mahar, D. (1996). Electronic parts reliability data 1997.

Dugan, J.B., Bavuso, S.J., and Boyd, M.A. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on reliability*, 41(3), 363–377.

Ghezzi, C. and Sharifloo, A.M. (2013). Model-based verification of quantitative non-functional properties for software product lines. *Information and Software Technology*, 55(3), 508–524.

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

Guo, H. and Yang, X. (2007). A simple reliability block diagram method for safety integrity verification. *Reliability Engineering & System Safety*, 92(9), 1267–1273.

IEC (2010). Functional safety of electrical/electronic/programmable electronic safety-related systems. URL `https://www.iec.ch/functionalsafety/standards/`.

Joshi, A., Vestal, S., and Binns, P. (2007). Automatic generation of static fault trees from aadl models. In *DSN Workshop on Architecting Dependable Systems.* Springer Berlin.

Kwiatkowska, M., Norman, G., and Parker, D. (2011). Prism 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*, 585–591. Springer.

La Band, R.A. and Andrews, J.D. (2004). Phased mission modelling using fault tree analysis. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, 218(2), 83–91.

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.

Legendre, A., Lanusse, A., and Rauzy, A. (2017). Toward model synchronization between safety analysis and system architecture design in industrial contexts. In *International Symposium on Model-Based Safety and Assessment*, 35–49. Springer.

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. (2016). Safesyse: A safety analysis integration in systems engineering approach. *IEEE Systems Journal*, 12(1), 161–172.

Morozov, A., Ding, K., Steurer, M., and Janschek, K. (2019). Openerrorpro: A new tool for stochastic model-based reliability and resilience analysis.

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

Nordmann, A. and Munk, P. (2018). Lessons learned from model-based safety assessment with sysml and component fault trees. In *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 134–143. ACM.

OMG (2017). Omg systems modeling language (omg sysml)- version 1.5. *May-2017*. URL `http://www.omg.org/spec/SysML/1.5/`.

SAE (1996). Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment. URL `https://www.sae.org/standards/content/arp4761/`.

SAE (2010). Guidelines for development of civil aircraft and systems. URL `https://www.sae.org/standards/content/arp4754a/`.

Stamatis, D.H. (2003). *Failure mode and effect analysis: FMEA from theory to execution.* ASQ Quality Press.

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

Steurer, M., Morozov, A., Janschek, K., and Neitzke, K.P. (2019). Model-based dependability analysis of fault-tolerant inertial navigation system: A practical experience report. *IFAC PapersOnLine.*

Tajarrod, F. and Latif-Shabgahi, G. (2008). A novel methodology for synthesis of fault trees from matlab-simulink model. *World Academy of Science, Engineering and Technology*, 41, 630–636.

Vesely, W.E., Goldberg, F.F., Roberts, N.H., and Haasl, D.F. (1981). Fault tree handbook. Technical report, Nuclear Regulatory Commission Washington dc.

Wang, L. and Cai, F. (2017). Reliability analysis for flight control systems using probabilistic model checking. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 161–164. IEEE.

Xiang, J., Yanoo, K., Maeno, Y., and Tadano, K. (2011). Automatic synthesis of static fault trees from system models. In *2011 Fifth International Conference on Secure Software Integration and Reliability Improvement*, 127–136. IEEE.

Xing, L. and Dugan, J.B. (2002). Analysis of generalized phased-mission system reliability, performance, and sensitivity. *IEEE Transactions on Reliability*, 51(2), 199–211.

Yakymets, N., Jaber, H., and Lanusse, A. (2013). Model-based system engineering for fault tree generation and analysis.