An Integrated Model-driven Framework for Simulation, Analysis, and Testing Based on OMG Standards

Vitali Schneider, Anna Deitsch, Reinhard German
Friedrich-Alexander University of Erlangen-Nürnberg
Department of Computer Science 7
Martensstr. 3, D-91058 Erlangen, Germany
{vitali.schneider, anna.deitsch, reinhard.german}@fau.de

ABSTRACT

Model-driven engineering techniques are becoming increasingly popular for cost-effective and highly productive software development. Particularly, approaches and tools based on the standards provided by the Object Management Group (OMG), first of all on its famous Unified Modeling Language (UML), are already widespread throughout both the industry and the research community. Following this trend, the approach for Test-driven Agile Simulation (TAS) combines model-driven engineering, simulation, and testing techniques, based on OMG standards, to allow for advanced performance analysis and validation at early design stages. Among others, by deriving executable simulations from a system model and test specifications, this approach provides a cheap and agile technique for quantitative assessments and quality assurance of the system under design. In this paper, we present the current state of the SimTAny framework that provides a versatile, integrated, and standard-based tool chain to support the TAS approach. In particular, SimTAny facilitates the creation of model specifications, based on UML and several standardized profiles like SysML, MARTE, and UTP, the automated transformation to executable simulations, as well as testing and analysis activities. We demonstrate the main features of the SimTAny framework and illustrate the application of TAS on an example monocular vision system for the autonomous approach and landing that uses a low-budget micro aerial vehicle system.

CCS Concepts

•General and reference → Performance; Validation; Verification; •Computing methodologies → Modeling methodologies; Model verification and validation; *Discrete-event simulation;* **•Software and its engineering → Software design engineering;** *Traceability;*

VALUETOOLS 2016, October 25-28, Taormina, Italy

DOI 10.4108/eai.25-10-2016.2266601

Keywords

Test-driven agile simulation, model-driven engineering, performance analysis, UML, model-driven testing

1. INTRODUCTION

Model-driven engineering (MDE) [6] is a promising approach for a cost-effective and high-productive development of complex software-intensive systems. It focuses on models as a primary artifact during the engineering process and is essentially based on two technologies: modeling languages and transformation mechanisms [17]. Besides the benefits of an increasing productivity due to the automated code generation from specification models, MDE provides a lot of advantages. To mention just a few examples: it helps to overcome the complexity hurdle by increasing the abstraction level during system design, it supports precise documenting of software architectures, and leads to significant improvements in the area of quality assurance.

As recent studies show, the principals of MDE are already practiced across a wide range of industries [5, 20]. Thereby, the standards provided by the *Object Management Group* (OMG)[1] find a very wide application. This concerns in particular the OMG's *Unified Modeling Language* (UML) [15], especially for software development. Even if domain-specific modeling languages (DSML) are applied, they are often implemented as an extension profile for UML or used in combination with it. Based on UML, several standardized profiles are offered by OMG to cover specific modeling issues. Thus, for instance, the *System Modeling Language* (SysML) [13] allows modeling of system specifications and requirements, the profile for *Modeling and Analysis of Real-time and Embedded systems* (MARTE) [12] can be used to express non-functional properties, time and analysis aspects, while the *UML Testing Profile* (UTP) [9] considers testing issues. Furthermore, dedicated transformation languages MOFM2T [10] and QVT [11], respectively for model-to-text and model-to-model transformations, are also offered by OMG.

As a logical consequence of MDE, the approach of *Test-driven Agile Simulation* (TAS) [2] aims to combine model-driven engineering, simulation, and testing to a holistic approach by building upon the common modeling and transformation standards of OMG mentioned above. Using UML as a basic modeling language and coupling it with appropriate standardized extension profiles, the TAS approach effectively allows for seamless inspection of the involved models at early design stages. Thus, besides the static validation

[1] http://omg.org

of models, TAS also focuses on automated generation of executable simulations in order to analyze the performance and to validate the dynamic behavior of the designed system. This helps to improve the quality of the system under design as well as to reduce development and lifecycle costs potentially caused by insufficient designs.

Related MDE techniques based on OMG standards were partially approved in several European research projects, like COMPLEX [4] and MADES [16]. Quite similar ideas of coupling model-driven engineering, simulation, and testing are also embraced by some commercial development environments like SCADE[2] or IBM Rational Rhapsody[3]. However, in contrast to these projects, TAS is calling for a more holistic and solely standard-based approach, including not only system design but also simulation-based testing and performance analysis. A more detailed description of the TAS approach as well as a discussion about related work have been recently published in [18].

In this paper, we primarily provide an updated overview about our framework SimTAny, which implements a versatile, integrated, and seamlessly standard-based tool environment to support the TAS approach. Particularly, in section 2 of this paper we focus on presenting the main features of this framework, like for instance modeling support, simulation code generation, traceability, and design of experiments. Subsequently, in section 3 we demonstrate how modeling, analysis, and validation issues can be performed with SimTAny on an example monocular vision system for the autonomous approach and landing using a low-cost micro aerial vehicle (MAV) system.

2. SIMTANY FRAMEWORK

In order to provide a technical support for the suggested TAS approach, the framework SimTAny (formerly introduced in [2] as 'VeriTAS') has been developed. This framework integrates relevant tools with newly developed components in a common environment based on a popular and open platform Eclipse RCP[4]. Among others, SimTAny integrates a UML modeling tool, a transformation framework, a simulation engine, and an analysis tool (see Fig. 1). Furthermore, due to a service-oriented architecture of the framework and mostly open source tools used, SimTAny provides a versatile environment that can be easily adapted for specific domains allowing integration in a distributed development environment. In the following sections, we describe the main features of our framework in some more details.

Modeling

As previously mentioned, our TAS approach utilizes UML as a basic modeling language that provides a general-purpose notation for object-oriented modeling. Using annotations from specialized extension profiles for UML, such as SysML, MARTE and UTP, it is further possible to express specific aspects like requirements, non-functional properties and tests (see examples in section 3). Thus, UML forms a fundamental basis for a holistic modeling methodology with only one common modeling language, as intended for TAS to cover system design, analysis and validation aspects.

In order to enable standard conform modeling using different profiles mentioned above, SimTAny integrates the open source modeling tool Papyrus[5]. We preferably use Papyrus since it most precisely implements several OMG standards and provides an advanced modeling support for UML including SysML and MARTE profiles. Nevertheless, all other modeling tools can also be used instead of Papyrus, provided that they are able to exchange models via the OMG' interchange format XMI consistently.

SimTAny also offers some extensions to Papyrus in order to improve the modeling efficiency. This primarily concerns the modeling of detailed behaviors and expressions with textual editors. For this purpose, we apply the standardized *Action Language for Foundational UML* (ALF) [14], which alternatively allows the specification of detailed behaviors by means of a higher programming language. Such textual representation is often much more compact, faster, and intuitive than standard UML behavior diagrams.

Static Validation and Verification

In order to achieve static validation and verification of modeled specifications, SimTAny provide several constraints to check for inconsistencies in each model separately and for incompatible relations between models. Since such checks are easily performable on very early design stages, this helps to eliminate the propagation of modeling errors and thus to increase the efficiency of the development process. This feature has been realized in SimTAny by means of the Eclipse EMF validation framework[6]. The detected defects are listed in the Eclipse' problems view and the affected elements are marked as erroneous in the model editor as well. It is further possible to navigate from problems view to corresponding elements in the model editor. Currently only a few constraints for most typical failures are implemented in SimTAny. Nevertheless, new constraints can be easily added.

Transformation to Simulation Code

One of the most important tasks of TAS, and thus also of SimTAny, is to enable for validation and analysis of the dynamic system behavior. For this purpose, our framework provides a transformation module for automated generation of the executable simulation code from system and test models. Since a solid methodology with extensive tool support is required to perform this task, we build our code generators upon the OMG's standard MOFM2T [10] and its reference implementation, i.e. the Eclipse Acceleo[7] framework.

Using Acceleo, we have implemented model-to-text transformation templates, as prescribed by MOFM2T. This enables us to derive executable simulations for OMNeT++[8] - an open source discrete-event simulator that is quite popular for simulation of communication networks. Furthermore, our transformation module is designed to be extendable for other simulation engines too.

During the execution of the generated simulation model, one can observe the behavior of system components, especially focusing on their communication patterns. Beside this, quantitative measurements are collected during the execution and can be analyzed afterwards.

[2]http://www.esterel-technologies.com/products/
[3]http://www.ibm.com/software/awdtools/rhapsody/
[4]http://wiki.eclipse.org/Rich_Client_Platform
[5]http://eclipse.org/papyrus
[6]http://eclipse.org/modeling/emf/?project=validation
[7]http://eclipse.org/acceleo
[8]http://omnetpp.org

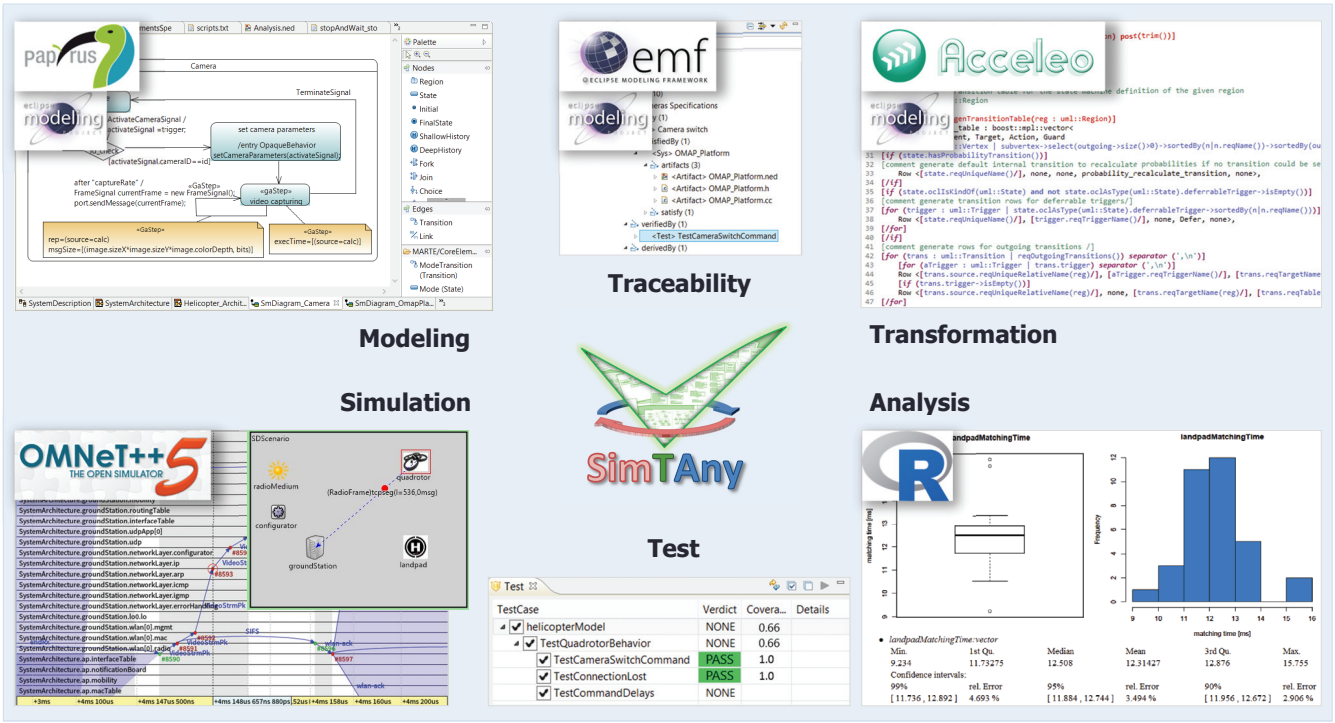


Figure 1: Main features of the SimTAny framework and integrated tools (Reproduced from [18])

Analysis

In order to simplify the access to output data collected during the simulation runs, SimTAny provides a dedicated perspective where the simulation results can be imported and visualized. It allows for calculation of a number of key statistical measures as well as creation of various graphical plots for the selected data. Therefore, it is possible to perform first analyses without leaving the SimTAny environment.

Test

Along with the generation of simulation code from the system model our framework also provides the generation of executable OMNeT++ simulations for each test case contained in the test model. Within a dedicated test view the user may obtain an overview of all test contexts and test cases defined in the model. Once the simulation code has been generated for the model, tests can be performed from this view. The test view also presents the verdict and eventual error reports of each completed test run.

Traceability

Further on, SimTAny provides a dedicated model for collecting of traceability information. Initially traceability links are created in the specification models. Thus, using the SysML profile, the designer has to assign test cases to requirements they are intended to verify or to assign system model elements to requirements they satisfy. These traceability links are automatically transformed by SimTAny to its own traceability meta-model utilizing the additional OMG standard for model-to-model transformations QVT and the Eclipse modeling framework EMF[9]. On the one side, a dedicated traceability model allows to put all relevant information together in a more compact form. On the other side, it can be enriched with additional information that is outside of the scope of specification models, like for example traceability links to code or simulation artifacts generated from models.

In order to provide an overview about all available traceability information, SimTAny implements a special traceability view. This view enables the user to inspect relationships between elements in all directions starting either from requirements, from system or test model elements, or even from artifacts. The traceability view also provides filters to detect potential deficiencies such as unsatisfied or untested requirements.

Design of Experiments

In order to inspect alternative system configurations, experiments with different combination of parameters have to be defined and performed. It is worth noting that using SimTAny an experiment can be defined in two ways: (1) Directly in the analysis model as a class annotated with MARTE's stereotype *GaAnalysisContext* and providing default values for each property that represent an experiment parameter; (2) Using our additional design of experiments module for more advanced and automated generation, management, execution control as well as result analysis of experiment studies. The module for method (2) has been initially set up in [21] applying model-driven development techniques based on the EMF framework. Among others, it allows designing parameter variation rules and automated generation of concrete experiment iterations according to various design-of-experiments (DOE) methods, like for instance *full factorial* or *latin hypercube* designs. More details about DOE can be found in [8] or [7].

[9]http://eclipse.org/modeling/emf

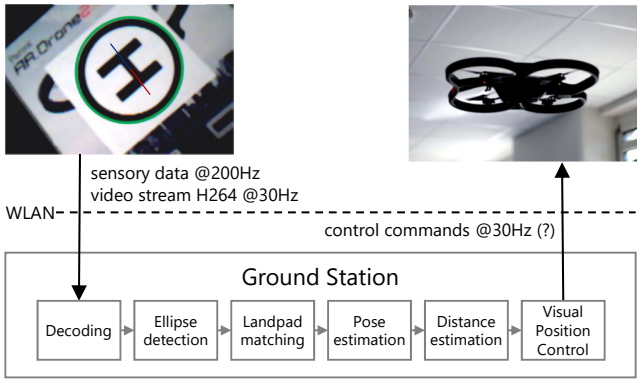


Figure 2: Concept of a monocular vision system for autonomous approach and landing using a low-budget quadrotor (Reproduced from [3])

3. CASE STUDY

In this section we introduce a system to exemplify a use case of the SimTAny framework. The system that is shown in Fig. 2 presents a monocular vision system for autonomous approach and landing using an off-the shelf Parrot A.R.Drone 2.0 quadrotor MAV. This system enables the quadrotor to autonomously detect a typical helicopter landpad consisting of a letter "H", approach it, and land on it. A detailed description as well as the implementation details of this system have been illustrated in [3].

In this work we selected only few factors that pose challenges for development of applications for this system. The detection of a landpad is performed by analyzing video stream frames captured using one of the two cameras installed on the quadrotor. Due to quadrotor's hardware restrictions the image processing pipeline cannot be performed directly onboard. Instead, the quadrotor must communicate with a ground station that receives sensory and video data, performs the computations, and generates steering commands. Quadrotors are generally very unstable systems that must react in a timely manner. To achieve smooth drone movements, steering commands have ideally to be sent every 30ms. However, delays between different kinds of sensors, delays introduced by the detection of the landpad, as well as delays arising during the communication between the quadrotor and the ground station can occasionally cause the quadrotor to drift while hovering over the landpad since the steering commands cannot be sent fast enough.

In order to support the development of this application, we first provide specification models which represent all relevant requirements, system's architecture and behavior aspects as well as formal tests for further validation. The application of the TAS approach using the SimTAny framework allows then to derive executable simulations. By means of simulation of the system behavior and communication aspects, we aim at resolving these problem statements where we investigate system's properties and compare alternative design solutions. For this case study in particular, we focus on the estimation of network latencies and detection of performance bottlenecks for variable system configurations regarding different camera resolutions and network transport protocols. The goal is therefore to determine which system configuration produces the best landpad detection rate and hence the steering rate close to 30Hz.

Modeling

Based on the modeling guidelines defined in our previous work [19, 1], we will now focus on modeling of some basic aspects like requirements, functional system behavior and test cases.

Requirements Modeling

Our design specification typically starts with the definition and modeling of system requirements by means of SysML. Some exemplary requirements, that need to be considered in order to achieve a satisfying control of the quadrotor, including sending steering commands and video capturing, are illustrated in Fig. 3. For example, the requirements on quadrotor's cameras include restrictions according activation and switching time between cameras.

Based on the specified requirements model, the functional system model as well as the test model may be created independently from each other in order to ensure their utilization for mutual validation.

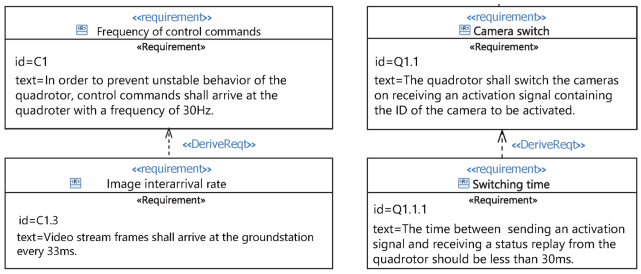


Figure 3: Example of system requirements model

Functional Modeling

The aim of a functional system model is to design the generic topology of the system architecture in terms of functional blocks. The structural and behavioral aspects of blocks can be expressed by means of SysML block definition, internal block, and state machine diagrams. For example, Fig. 4 presents the internal structure of the *Quadrotor* block from our example system. It consists of two cameras, a video encoder, controller, and a network component. By using the SysML concepts of flow ports and connectors, one is able to express communication links between these various components.

To model the behavior of system components, we primarily use UML state diagrams. An example state machine that represents the behavior of cameras is shown in Fig. 5.

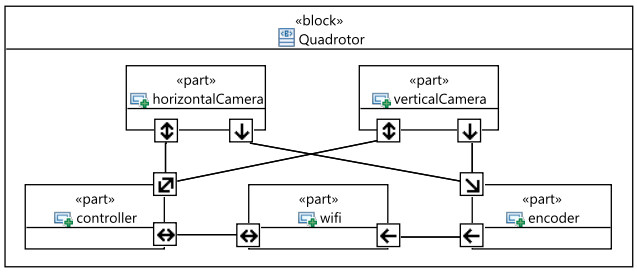


Figure 4: Example of a composite structure for quadrotor

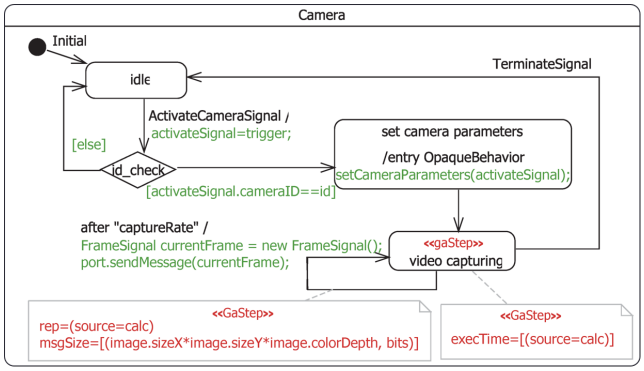


Figure 5: Example of modeling component behavior

In order to express non-functional performance attributes of behavior steps, we apply the MARTE's stereotype *GaStep*. For instance, we specify that the video frame size in bits can be measured as follows: resolution of frame × color depth. To collect the statistic of interest during the simulation, one just has to apply an additional MARTE expression *source=calc* on the corresponding property value.

Test Modeling

Similar modeling paradigm as used for functional system specification can also be applied for test specifications. Derived from common requirements the objective in this phase is to provide test cases for validation of the modeled system. As shown in Fig. 6, we utilize UML sequence diagrams to specify a test case. To identify several test aspects, i.e. test configurations, test cases, components under test, as well as test components, we apply the corresponding stereotypes of the UTP profile.

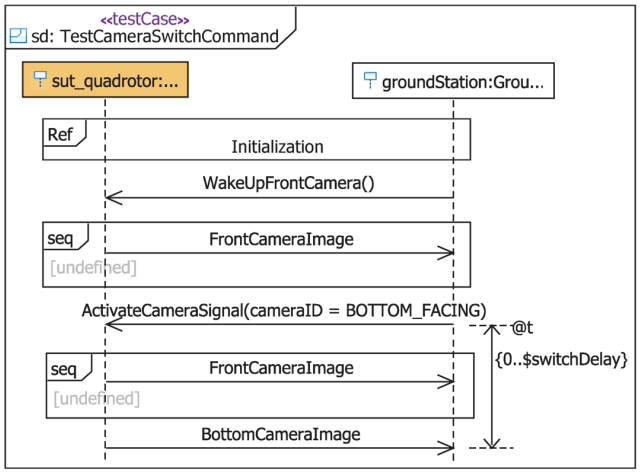


Figure 6: Example of modeling test case

Simulation Experiments and Results

The specification models, some parts of which have been presented in the previous sections, can finally be transformed with the help of SimTAny to executable simulation code for OMNeT++. Subsequently, simulation experiments and tests can be performed to examine and to validate these models.

As mentioned previously, one of the primary goals in this case study was to estimate the expected performance and network latency for the given application. In order to inspect alternative system configurations, different experiments have been defined and performed. Due to space limitation, we omit to show the complete overview of all experiments and just focus on a few of them.

Thus, among others, we analyzed the effects on the total image transmission delay and the detection rate by different frame resolutions and transport protocols for video streaming. In Fig. 7, we exemplarily show some averaged results of our simulation experiments. In particular, Fig. 7a displays varying proportions for image delays, i.e. the buffering and encoding delay on the quadrotor, network transmission delay, and decoding delay on the ground station. Thereby, two transmission protocols (TCP and UDP) and two resolutions of frames captured by quadrotor's cameras are considered in this example: SD with 640x360 pixels and HD with 1280x720 pixels. The distributions of time between arrivals of subsequent frames on the ground station are shown in Fig. 7b for different resolutions in case of using TCP as a transmission protocol. Furthermore, the resulting landpad detection rates for both SD and HD resolutions can be seen in Fig. 7c.

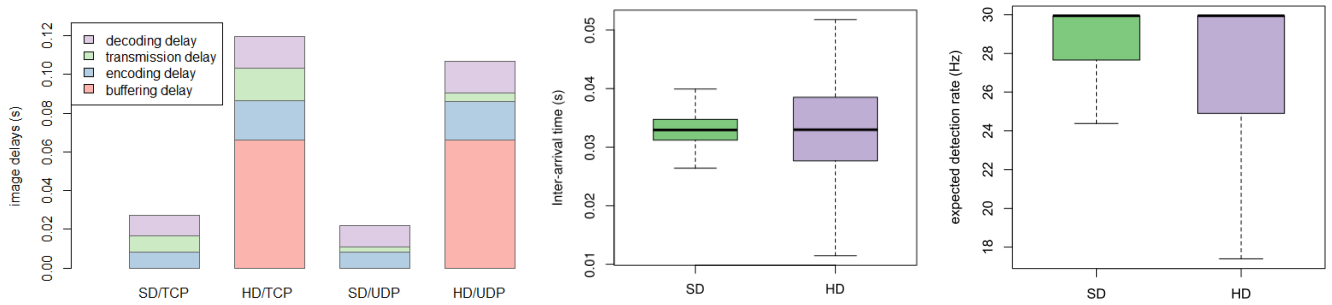
To summarize the results, we can exclude using of higher resolutions than SD because the required detection rate cannot be satisfied sufficiently in that case. Even if we could omit buffering on the quadrotor, which is automatically activated for HD encoding, the large remaining delay and rather big variations in inter-arrival times can lead to an unstable flight behavior of the quadrotor since steering commands cannot be sent on time. On the other side, the configuration with the SD resolution lets expect a sufficient detection rate above 24Hz in conjunction with an overall image delay of about 33ms. A further promising design solution would be to apply UDP as a transport protocol since the delay as well as the deviation of transmission times can be reduced significantly. Unfortunately, this option is not available for the Parrot AR.Drone 2.0 quadrotor.

The results obtained by the simulation for the SD/TCP configuration have already been successfully validated by the prototype implementation of the system. Furthermore, during the iterative modeling process we could repeatedly use validation and testing capabilities of SimTAny and could detect some modeling errors in the system model as well as in the test model itself.

4. CONCLUSIONS

One of the main objectives of the TAS approach and, by association, of its supporting framework SimTAny is to enable for efficient and qualitative development of complex systems. The combination of model-driven engineering, simulation and testing techniques based on common standards helps to improve the overall quality of the development process. On the one side, the models provide a good support in understanding, communicating and documenting of different engineering aspects. On the other side, simulation-based analysis and validation provide a cheap and agile possibility to investigate performance and to approve the functionality of the modeled system behavior.

In the presented work we offered an updated overview about main features of our framework SimTAny. Furthermore, on an example application for autonomous landpad



(a) Image delays (b) Inter-arrival times of frames (TCP) (c) Landpad detection rates (TCP)

Figure 7: Results of the simulation study

approaching of a quadrotor MAV, we demonstrated a practical application of SimTAny for modeling, performance analysis, and validation of a system under development.

In our ongoing efforts to improve the modeling efficiency we are working on the clarification of the holistic modeling methodology by introducing the concept of viewpoints and implementing a view-specific wizard modeling support. Furthermore, we intend to offer a domain specific modeling language and modeling library for the special domain of image processing systems.

5. REFERENCES

[1] A. Deitsch, V. Schneider, J. Kane, W. Dulz, and R. German. Towards an Efficient High-Level Modeling of Heterogeneous Image Processing Systems. In *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative (DEVS '16)*, Pasadena, CA, USA, Apr. 2016.

[2] A. Djanatliev, W. Dulz, R. German, and V. Schneider. VeriTAS - a Versatile Modeling Environment for Test-driven Agile Simulation. In *Proceedings of the Winter Simulation Conference*, WSC '11, pages 3662–3671, Phoenix, AZ, USA, Dec. 2011.

[3] S. Dotenco, F. Gallwitz, and E. Angelopoulou. Autonomous Approach and Landing for a Low-Cost Quadrotor Using Monocular Cameras. In L. Agapito, M. M. Bronstein, and C. Rother, editors, *Computer Vision - ECCV 2014 Workshops*, pages 209–222. Springer International Publishing, Cham, 2015.

[4] F. Herrera, H. Posadas, P. Peñil, E. Villar, F. Ferrero, R. Valencia, and G. Palermo. The COMPLEX methodology for UML/MARTE Modeling and design space exploration of embedded systems. *Journal of Systems Architecture*, 60(1):55–78, Feb. 2014.

[5] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen. Empirical Assessment of MDE in Industry. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 471–480, New York, USA, 2011. ACM.

[6] S. Kent. Model Driven Engineering. In M. Butler, L. Petre, and K. Sere, editors, *Integrated Formal Methods*, Lecture Notes in Computer Science, pages 286–298. Springer Berlin Heidelberg, 2002.

[7] J. P. C. Kleijnen. *Design and Analysis of Simulation Experiments*, volume 111. Springer-Verlag, 2008.

[8] A. Law and W. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, 2000.

[9] Object Management Group (OMG). UTP: UML Testing Profile 1.2. http://omg.org/spec/UTP.

[10] OMG. MOF Model to Text Transformation Language (MOFM2T). http://omg.org/spec/MOFM2T, 2008.

[11] OMG. MOF Query/View/Transformation (QVT). http://omg.org/spec/QVT, 2011.

[12] OMG. UML Profile for MARTE Modeling and Analysis of Real-Time and Embedded Systems. http://omg.org/spec/MARTE, 2011.

[13] OMG. Systems Modeling Language (SysML). http://omg.org/spec/SysML, 2012.

[14] OMG. Action Language for Foundational UML (ALF). http://omg.org/spec/ALF/, 2013.

[15] OMG. UML Unified Modeling Language. http://omg.org/spec/UML, 2015.

[16] I. R. Quadri, A. Bagnato, and A. Sadovykh. MADES EU FP7 Project: Model-Driven Methodology for Real Time Embedded Systems. In M. A. Khan, S. Saeed, A. Darwish, and A. Abraham, editors, *Embedded and Real Time System Development: A Software Engineering Perspective*, pages 57–89. Springer Berlin Heidelberg, 2014.

[17] D. C. Schmidt. Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, Feb. 2006.

[18] V. Schneider, A. Deitsch, W. Dulz, and R. German. Combined Simulation and Testing Based on Standard UML Models. In L. Fiondella and A. Puliafito, editors, *Principles of Performance and Reliability Modeling and Evaluation*, Springer Series in Reliability Engineering. Springer International Publishing, 2016.

[19] V. Schneider, A. Yupatova, W. Dulz, and R. German. How to Avoid Model Interferences for Test-driven Agile Simulation based on Standardized UML Profiles. In *Proceedings of the Symposium on Theory of Modeling and Simulation - DEVS Integrative (TMS/DEVS '14)*, Tampa, FL, USA, Apr. 2014.

[20] J. Whittle, J. Hutchinson, and M. Rouncefield. The State of Practice in Model-Driven Engineering. *IEEE Software*, 31:79–85, 2014.

[21] T. Zangl. Management and Control of Experiments in the Test-driven Agile Simulation. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2015.