Electronic Communications of the EASST
Volume 77 (2019)

Interactive Workshop on the Industrial Application of Verification and Testing, ETAPS 2019 Workshop (InterAVT 2019)

Rigorous Design of FDIR Systems with BIP

Iulia Dragomir and Saddek Bensalem

7 pages

Guest Editors: Anila Mjeda, Stylianos Basagiannis, Goetz Botterweck
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Rigorous Design of FDIR Systems with BIP

Iulia Dragomir[1] **and Saddek Bensalem**[2]

[1] iulia.dragomir@univ-grenoble-alpes.fr [2] saddek.bensalem@univ-grenoble-alpes.fr
Univ. Grenoble Alpes, CNRS, Grenoble INP*, VERIMAG, 38000 Grenoble, France

Abstract: The correct design of autonomous systems is a challenge, due to the uncertainties arising at execution time. A special case of uncertainties are the faults and failures that break the system's requirements. Dealing with such situations requires to design fault detection, isolation and recovery (FDIR) components. The aim of FDIR components is to detect when a fault has occurred and to apply a recovery strategy that brings the system into a mode where the requirements are satisfied. In this paper we describe an approach based on the *Behavior, Interaction, Priority* (BIP) tools for the rigorous design of FDIR components. This approach leverages the scalability of statistical model-checking tool BIP-SMC to check for requirement satisfaction, and the code generation feature of the BIP compiler. Moreover, the generated code is executable with the BIP engine(s) and easily integrated with the original system. The approach has been used in the H2020 ESROCOS and ERGO projects for the development of (autonomous) robotics control systems, which have been validated through field trials.

Keywords: Fault detection isolation and recovery (FDIR), BIP (Behavior, Interaction, Priority), Statistical model checking, Autonomous robotics control systems

1 Introduction

The recent developments in machine learning and artificial intelligence have put the notions of autonomy and adaptation at the forefront of computer-based systems, with applications in many domains. For example, the PERASPERA Programme[1] aims to develop and demonstrate the technologies that will enable the next generation of autonomous space systems. Different efforts within this programme have already designed the basic building blocks for robotics control systems [MMW+17] and validated a proof-of-concept model-based design process for autonomous rovers and on-orbit robotic servicing platforms [OBC+18, OCE+18].

A crucial topic of study in the design of such applications is the systematic and robust handling of faults and failures that occur at execution time. The rationale is that uncertainties happening at execution can lead the system into situations where the desired requirements do not hold anymore, and therefore have critical consequences at many levels. An accepted solution is to consider within the system design fault detection, isolation and recovery (FDIR) components. The aim of such components is two-fold: (i) detect whether a fault has occurred at execution (i.e., *diagnoser*) and (ii) apply a strategy such that the system meets again the requirements (i.e., *recovery controller*).

* Institute of Engineering Univ. Grenoble Alpes
[1] https://www.h2020-peraspera.eu/

Recent work has leveraged the use of formal methods for designing FDIR components [WF12]. For example, synthesis algorithms are used for building the two parts of the FDIR components: the diagnoser for the fault detection and the controller for the recovery. Such algorithms are proposed and implemented in [BBC+14, BBC+16] for untimed systems and in [DIBB18] for real-time systems with partial observability. However, these methods have their limitations since they can only be applied for event-based safety properties and do not necessarily scale on real-life systems.

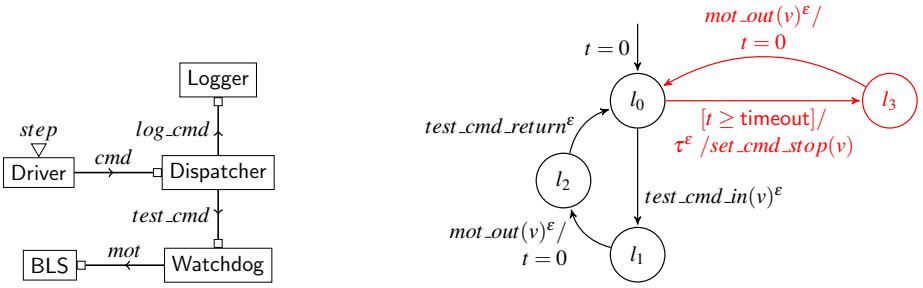
In this paper we propose an alternative approach for FDIR design with BIP (Behavior, Interaction, Priority) [BBS06]. BIP is a framework consisting of a modeling language and several static analysis tools. The language allows modeling complex heterogeneous systems with real-time and stochastic features. The validation and verification analysis tools range from simulation, to safety property checking and performance evaluation.

Problem Statement. More precisely, we are interested in the correct design of FDIR components and, in this aim, we are using the BIP tools for system design and analysis. These tools allow to ensure that the obtained code is correct-by-construction and can therefore be deployed, while leveraging scalability. In the context of the ESROCOS and ERGO projects, our contribution is to model and validate FDIR components for system designs in TASTE [PCDT12] and to embed the generated code in the deployed system.

Case study. Our case study is an excerpt of a rover demonstrator control system developed for the validation of the ESROCOS environment [esrb]. This system, illustrated in Figure 1a as a TASTE design, covers the software chain of a robotics control system driving functionality. More precisely, it consists of a Driver that regularly sends motion commands *cmd* to the locomotion software for execution. The command sending is activated by the *step* trigger, which is set to 100*ms*. The *cmd* first travels through a Dispatcher that sends the request to two components: Logger for data logging and replaying, and Watchdog for data validity checking. Finally the Watchdog transfers the request as *test_cmd* to the locomotion component BLS for actual execution. The communication between components is asynchronous with queues that store at most one request and transfer it to the receiver with a periodicity of 50*ms*.

The Watchdog component, given in Figure 1b, models FDIR behavior related to its data validity checking. In the nominal scenario, the motion requests generated by the Driver are received before a timeout. If faults occur, these requests could be delayed or lost. In this case, in order to guarantee its safe operation, the rover must be stopped. Therefore, the detection part monitors the incoming *test_cmd* with respect to the timeout and the controller part sends the stop command to the BLS.

It is worth mentioning that the original system contains 20 components, and has been validated by 5 test cases in field trials (see Figure 2). The Watchdog component is extensively executed in all scenarios, as it transfers all communications between the control device and locomotion system and checks at the same time their validity. Moreover, one specific test case is dedicated to validate its behavior in faulty conditions. For further details the reader is referred to [MDNB18] with the model and implementation available in [Dra].



(a) System architecture (b) Behavior of Watchdog

Figure 1: Example of an autonomous robotics control system, where the Watchdog component, modeled with BIP, defines FDIR behavior (illustrated in red).

(a) Bridget rover (ESROCOS) (b) SherpaTT (ERGO)

Figure 2: Field tests of rovers running FDIR code designed, analyzed and generated with BIP (courtesy of DFKI).

2 FDIR Design and Analysis with BIP

The BIP Framework. BIP is a component-based real-time modeling formalism that allows one to design complex system models in a compositional manner. The language is based on the well-established theory of Timed Automata (TA) [AD94]. More specifically, components are modeled as TA extended with data, external code and urgencies. A system is given by the components composition through multi-party interactions (n-ary synchronizations among component actions). Additionally, the language accounts for the modeling of faults (through stereotypes) and uncertainties (through probability density functions on events, i.e., stochastic behavior [NMB+18]).

A BIP model can be obtained either through modeling or automated transformation from the supported language factory such as TASTE, DOL, etc. This model can be subject to multiple analyses at different granularity levels with the BIP tools[2]. The RTD-Finder tool checks the satisfaction of safety properties with a compositional invariant-based method. The BIP compiler generates C++ code from the BIP model, that can be executed with the BIP engines. Therefore

[2] http://www-verimag.imag.fr/RSD-Tools.html

the system requirements can be validated by different types of simulation: interactive, step-wise, real-time, etc. Thorough validation can be achieved with the BIP-SMC tool [MNB+18]. The statistical model-checking tool runs a relevant number of simulations (based on statistical confidence parameters) and computes/checks the probability of satisfaction of the formalized requirements.

Proposed Approach. The approach we use for the design and analysis of FDIR components is depicted in Figure 3. The method works on a fully specified BIP model including the designed FDIR component. Please note that the elements of this model can be obtained separately from different sources as showed below for the case study. Then the BIP compiler is called to assemble the elements into one full specification. The compiler checks the well-formedness of the obtained model and generates executable C++ code. The generated code is executed with the corresponding BIP engine and the obtained simulations are checked. The aim of this validation is to confirm that the model behaves as expected and possibly correct modeling errors as early as possible.

To provide better guarantees for the system properties, we use BIP-SMC next. Please note that for this analysis the stochastic (real-time) version of the BIP compiler and engine is used. The properties that can be checked could be divided in at least two categories: properties over the nominal behavior (where no faults are present), and properties of the FDIR behavior (where faults are exhibited at execution). The properties of FDIR behavior can cover not only the robustness of the detection and recovery mechanism, but also performance measurements. If the results of these analyses are satisfying, we deploy the generated code of interest (including the BIP engine) with the original system.

This approach answers the limitations of current approaches in FDIR design mentioned in Section 1. For instance, many of the properties to enforce with FDIR are value-based, e.g., if the battery level is below a certain threshold, then stop the rover and recharge. Moreover, we consider that any FDIR components can be tackled with this approach due to the expressivity of the BIP modeling language. One possible shortcoming of this approach is the use of statistical model-checking, and the BIP-SMC tool in particular, which is a non-exhaustive verification method. However this technique provides analysis results with a certain confidence level, while being usable on complex system models, as is the case for mission- and safety-critical applications.

Illustration on the Case Study. This paper's case study is originally designed with TASTE and describes only the nominal behavior excluding the Watchdog. The aim of our work is to provide our partners with correct-by-construction code for the Watchdog. First, we obtain the BIP model of the case study automatically with the TASTE2BIP tool[3]. Then, we model several faults described in the specification, such as non-sending of the *cmd* request for a certain time elapse (Driver component) or losing requests (Dispatcher component). Next, we fully model the Watchdog component with BIP based again on the specification, as illustrated in Figure 1b. This component stores in the variable *v* either the motion command sent by the Driver in the nominal case or the stop command in the fault case. This value is forwarded to the BLS on the *mot_out*

[3] https://gricad-gitlab.univ-grenoble-alpes.fr/verimag/bip/TASTE2BIP.git

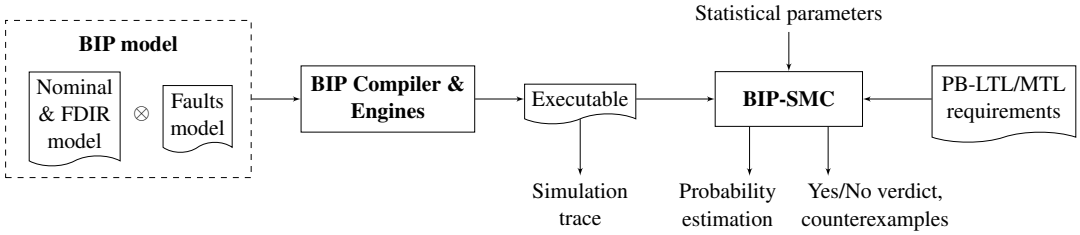


Figure 3: Approach for FDIR components design and analysis with the BIP framework.

interaction. To detect the fault occurrence, the time elapse between two $test\_cmd$ requests is monitored by the clock t with respect to a configurable timeout (e.g., 110ms).

The BIP compiler generates the code corresponding to the full model, and its execution shows that the Watchdog component performs as expected both for the nominal and faulty behavior. More specifically, several interactive simulations are run with the BIP real-time engine and checked with respect to the expected behavior. BIP-SMC is used to check multiple properties of the full system. For example, a nominal requirement specifies that all *cmd* requests sent by the Driver are received within 100ms by the BLS component. The FDIR robustness is specified by the consistent receiving of *mot* requests by the BLS every 110ms at most, either those sent by the Driver or the stop ones. Further details about the checked requirements and analysis results can be found in [MDNB18].

The obtained analysis results provide high confidence on the correctness of the Watchdog, and C++ code is generated for this component in particular. The generated code and the BIP engine (for scheduling one component) are integrated in the system design and finally deployed on the actual rover. The Watchdog component and the required wrappers total 2000 lines of C++ code, while the BIP real-time engine totals 6500 lines of C++ code. The models and source code of this case study can be found in [Dra], with the full application available in [esra].

3 Conclusion

In this paper we describe the approach we used in two H2020 projects, ESROCOS and ERGO, for designing FDIR components with BIP. FDIR components are modeled with BIP and multiple analyses are performed to ensure their correction with respect to the system and its properties. The BIP compiler is used to generate C++ code that is executed with the BIP engines and validated by simulation. The confidence on the system's correctness both at global and FDIR level is validated with BIP-SMC. Once the analysis results are satisfying, the generated code can be deployed with the system. This approach has several features as follows: (i) high usability as all used tools are integrated into the BIP framework (including a graphical user interfaces), (ii) good scalability on real-life applications due to the use of statistical model-checking and (iii) rigorous design due to the preservation of checked properties in the generated code.

This approach has been used on five case studies from the aforementioned projects. The code obtained with BIP for the FDIR components is successfully integrated in five (autonomous) robotics control systems. These systems are validated both through integration and field testing.

The feedback obtained from the industry partners is very positive. The FDIR components code is robust, succeeding in all types of situations, both foreseen by the tests but also unexpected ones. Also, the generated code shows to be more efficient (of an order of magnitude) in terms of resource consumption compared to other tools used for the design of FDIR components (e.g., TASTE).

Acknowledgements: This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement #730080 (ESROCOS) and #730086 (ERGO).

Bibliography

[AD94] R. Alur, D. L. Dill. A Theory of Timed Automata. *Theor. Comput. Sci.* 126(2):183–235, Apr. 1994.

[BBC+14] B. Bittner, M. Bozzano, A. Cimatti, R. D. Ferluc, M. Gario, A. Guiotto, Y. Yushtein. An Integrated Process for FDIR Design in Aerospace. In *IMBSA 2014*. Pp. 82–95. 2014.

[BBC+16] B. Bittner, M. Bozzano, R. Cavada, A. Cimatti, M. Gario, A. Griggio, C. Mattarei, A. Micheli, G. Zampedri. The xSAP Safety Analysis Platform. In *TACAS 2016*. Pp. 533–539. 2016.

[BBS06] A. Basu, M. Bozga, J. Sifakis. Modeling Heterogeneous Real-time Components in BIP. In *SEFM 2006*. Pp. 3–12. 2006.

[DIBB18] I. Dragomir, S. Iosti, M. Bozga, S. Bensalem. Designing Systems with Detection and Reconfiguration Capabilities: A Formal Approach. In Steffen and Margaria (eds.), *Leveraging Applications of Formal Methods, Verification and Validation - 8th International Symposium, ISoLA 2018, Lymassol, Cyprus, November 5-9, 2018*. Lecture Notes in Computer Science. Springer, november 2018.

[Dra] I. Dragomir. ESROCOS Planetary Exploration Demonstrator: the Watchdog component in TASTE and BIP. https://github.com/ESROCOS/control-mc_watchdog.

[esra] ESROCOS Planetary Exploration Demonstrator. https://github.com/ESROCOS/plex-demonstrator-record.

[esrb] ESROCOS Project Github Repository. https://github.com/ESROCOS.

[MDNB18] B. L. Mediouni, I. Dragomir, A. Nouri, S. Bensalem. Quantitative Risk Assessment in the Design of Resilient Systems. Technical report TR-2018-10, VERIMAG, 2018. http://www-verimag.imag.fr/TR/TR-2018-10.pdf.

[MMW+17] M. Munoz, G. Montano, M. Wirkus, K. Hoeflinger, D. Silveira, N. Tsiogkas, J. Hugues, H. Bruyninckx, I. Dragomir, A. Muhammad. ESROCOS: a Robotic

Operating System for Space and Terrestrial Applications. In *Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA) 2017, Leiden, Netherlands, June 20-22, 2017*. june 2017.

[MNB+18] B. L. Mediouni, A. Nouri, M. Bozga, M. Dellabani, A. Legay, S. Bensalem. $\mathscr{S}$*BIP* 2.0: Statistical Model Checking Stochastic Real-Time Systems. In Lahiri and Wang (eds.), *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*. Lecture Notes in Computer Science 11138, pp. 536–542. Springer, 2018.
doi:10.1007/978-3-030-01090-4_33
https://doi.org/10.1007/978-3-030-01090-4_33

[NMB+18] A. Nouri, B. L. Mediouni, M. Bozga, J. Combaz, A. Legay, S. Bensalem. Performance Evaluation of Stochastic Real-Time Systems with the SBIP Framework. *International Journal of Critical Computer-Based Systems (IJCCBS)* 8(3), 2018.

[OBC+18] J. Ocon, K. Buckley, F. Colemenero, S. Bensalem, I. Dragomir, S. Karachalios, M. Woods, F. Pommerening, T. Keller. Using the ERGO framework in a Planetary and an Orbital Scenario. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS) 2018, Madrid, Spain, June 4-6, 2018*. june 2018.

[OCE+18] J. Ocon, F. Colemenero, J. Estremera, K. Buckley, M. Alonso, E. Heredia, J. Garcia, A. Coles, A. Coles, M. Martinez, E. Savas, F. Pommerening, T. Keller, S. Karachalios, M. Woods, I. Dragomir, S. Bensalem, P. Dissaux, A. Schach, R. Marc, P. Weclewski. The ERGO framework and its use in planetary/orbital scenarios. In *International Astronautical Congress (IAC) 2018, Bremen, Germany, October 1-5, 2018*. october 2018.

[PCDT12] M. Perrotin, E. Conquet, J. Delange, T. Tsiodras. TASTE-An open-source toolchain for embedded system and software development. In *Proceedings of the Embedded Real Time Software and Systems Conference (ERTS), Toulouse, France*. 2012.

[WF12] A. Wander, R. Forstner. Innovative Fault Detection, Isolation and Recovery Strategies On-board Spacecraft: State of the Art and Research Challenges. Deutscher Luft- und Raumfahrtkongress, 2012.