ISSN 1992-6502 (Print)
Vol. 18, no. 5 (66), pp. 68-72, 2014

Вестник УГАТУ
Vestnik UGATU

ISSN 2225-2789 (Online)
http://journal.ugatu.ac.ru

UDC 004.4'23

CONCURRENT ERROR DETECTION FOR DISTRIBUTED SYSTEMS: A CASE STUDY

A. RASHIDOV[1], A. MOROZOV[2], K. JANSCHEK[3], N. YUSUPOVA[4]

[1] arseniy.rashidov@gmail.com, [2] andrey.morozov@tu-dresden.de,
[3] klaus.janschek@tu-dresden.de, [4] yussupova@ugatu.ac.ru

[1, 4] Ufa State Aviation Technical University, Russia;
[2, 3] Technische Universität Dresden, Germany

Submitted 2014, June 10

Abstract. Error detection is defined as observation of system operation in order to ensure its consistency with expected system behavior. According to IEEE stadards, error detection is one of the means to achieve fault tolerance. Hence, error detection is a necessary part of safety-critical systems design. Concurrent error detection for distributed systems is one of the problems researched in a scope of a joint project S3ARV (Small & Safe Space Autonomous Robotic Vehicles) of IfA, TU Dresden and iFR, Uni Stuttgart. In the presented article, we introduce a new safety monitoring framework, based on concurrent error detection. Our approach is focused on monitoring of distributed real-time control software. A prototype of the framework is applied on a real flight vehicle (an Octocopter).

Keywords: FDIR; error detection; safety monitoring; distributed systems.

1. INTRODUCTION

1.1. Motivation

Nowadays safety-critical systems such as automobiles, airplanes, and spacecrafts have become too complex to be developed without faults. Hence, their correct utilization in the presence of faults needs to be guaranteed. Some measures are necessary to be taken to achieve fault tolerance.

According to [1], activity for fault tolerance achievement is divided into error detection and recovery. On the other hand, according to [2] and [3], the authors at ESA (European Space Agency) split fault tolerance actions into fault detection, identification/isolation, and recovery.

Research results, presented in this paper, belong to the field of error detection.

1.2. State of the Art

The known works address a problem of fault detection based on probabilistic and other mathematical models [4–7]. Whereas this is applicable to many systems, there is a demand to develop simple and reliable methods of error detection in the condition of scarce system resources.

On the other hand, not many works can be found in the field of error detection for distributed systems and real-time systems. Again, the main focus in the known works [8, 9] is on the construction of fault models based on a comprehensive mathematical apparatus.

We have not found works stressing the problem of possible faults in the complementary parts of distributed systems, such as network.

1.3. Problem Definition

Concurrent error detection for distributed systems is one of the problems researched in a scope of a joint project S3ARV (Small & Safe Space Autonomous Robotic Vehicles) of IfA, TU Dresden and iFR, Uni Stuttgart. In the presented article, we introduce a new safety monitoring framework, based on concurrent error detection. Key functional requirements to the framework are listed below:

- Receive safety states of critical software components of the system;
- Estimate the whole system's safety state using the received information;
- Send the system's state to an automotive control unit that can switch the system to the hovering mode or to perform safety landing in case of error detection;
- Downstream relative safety information to a user.

1.4. Special Requirements

The error detection framework must be applicable to the S3ARV flight vehicle software, i. e.:

- Distributed software components on different hardware processing units.
- Data-driven framework.
- TCP/IP connection interface.
- Strict memory and central processing usage restrictions.

2. SAFETY MONITORING FRAMEWORK

2.1. Framework parts and their place in the system

The safety monitoring framework is a set of software tools for error detection that is developed in the scope of S3ARV project. It consists of four basic parts:

1. **Safety clients** at the system's **components** which send the error alerts generated inside a component to the Safety Monitor;
2. **Safety monitor** which gets all the alerts from the components and generates the system's state, considering system's real-time properties and a state machine (fig. 2). It then sends the system's state to the Error Handler and the LMD – logging/monitoring data to the Safety Monitoring Software.
3. **Error handler** which triggers recovery actions at the system's **Controller** component;
4. **Ground Control Safety Monitoring Software** which shows the user the state of all the components and the system's state and alerts in case of error.

The dataflow between components is shown on Fig. 1.

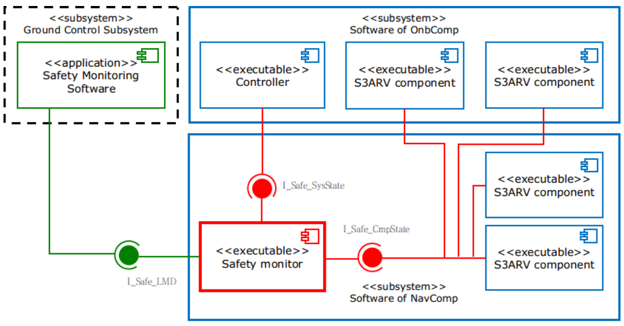
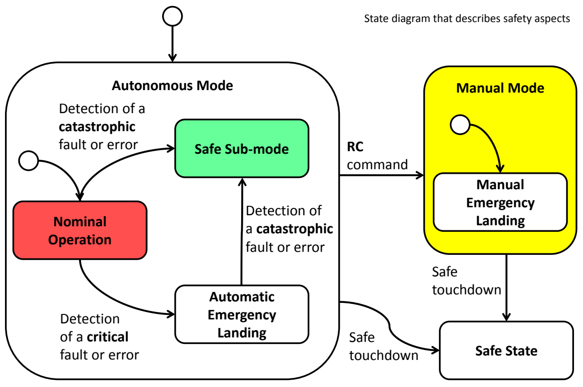


Fig 1. The place of the Safety monitor against the other S3ARV components

2.2. Safety monitor

The main part of the framework is Safety monitor. Its function is:

- To receive the system state messages from the components;
- To check the system safety parameters: timeouts between messages, component ID uniqueness,
- To estimate the system state using the information from (1) and (2) and the recovery actions state machine;
- To send the current system state messages with certain frequency, as well as error messages immediatly after new error registration.



State diagram that describes safety aspects
Autonomous Mode
Detection of a catastrophic fault or error
Safe Sub-mode
Nominal Operation
Detection of a catastrophic fault or error
Detection of a critical fault or error
Automatic Emergency Landing
RC command
Manual Mode
Manual Emergency Landing
Safe touchdown
Safe touchdown
Safe State

Fig 2. The state machine of the controller's recovery actions

One of the goals of the monitor design phase was to keep it as simple as possible. Simplicity improves reliability. The complete functional diagram is presented on Fig. 3.

3. IMPLEMENTATION

The part of system is developed for two embedded ARM platforms: Pandaboard ES[1] and Gumstix[2] running OpenEmbedded[3] GNU / Linux operating system. Ground control safety monitoring software is deployed on a PC with Ubuntu OS[4] running. The components that run on the flight vehicle are written in C++ with Boost 1.48 libraries[5] for TCP/IP network communication (Boost.ASIO), as well as threading and time management.

Ground Control Safety Monitoring Software is implemented in C++ with Qt[6] framework providing both the TCP/IP network communication and user interface.

The components are configured through configuration files.

[1] http://pandaboard.org/
[2] https://www.gumstix.com/
[3] http://www.openembedded.org/wiki/Main_Page
[4] http://www.ubuntu.com/
[5] http://www.boost.org/
[6] http://qt-project.org/

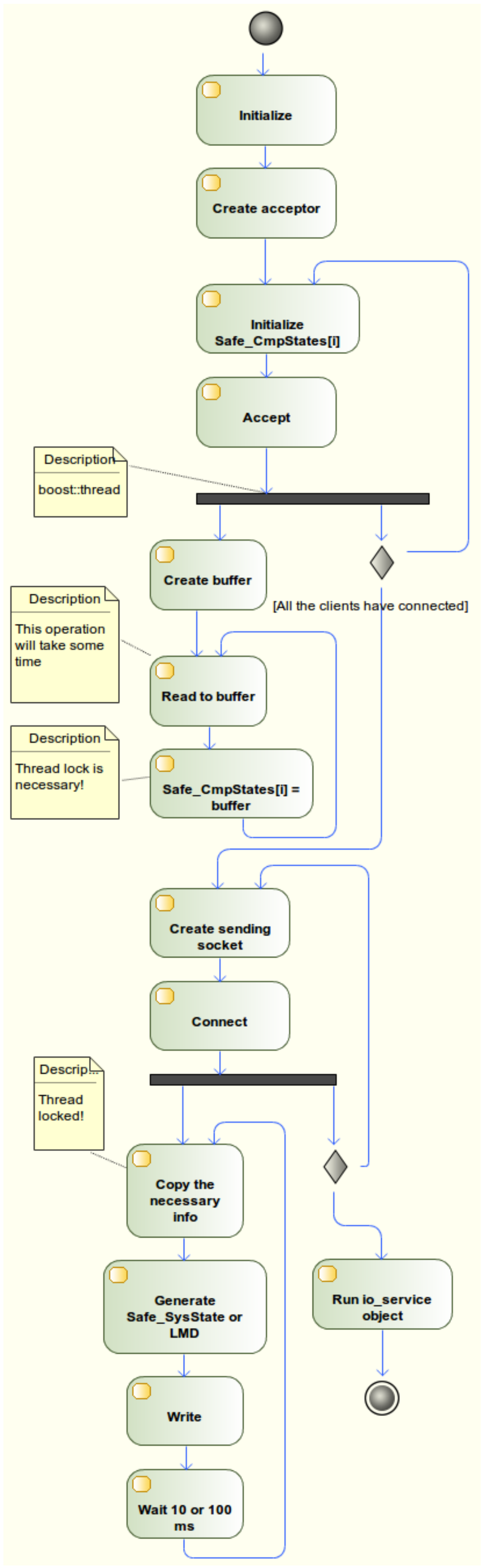
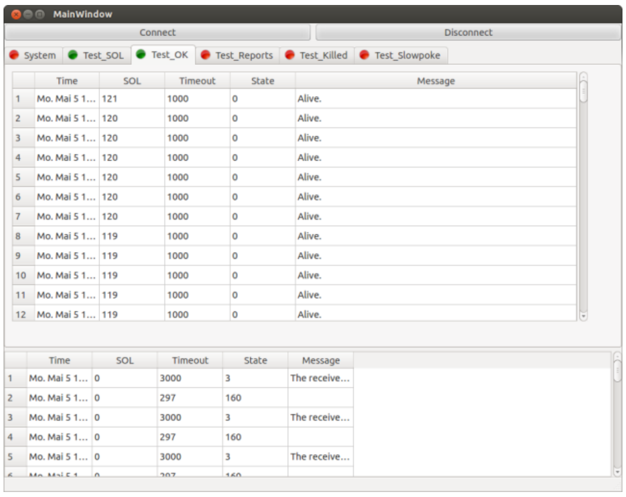


Fig 3. Safety monitor functional diagram

4. CASE STUDY

A case study for the approach includes detection of the following situations:

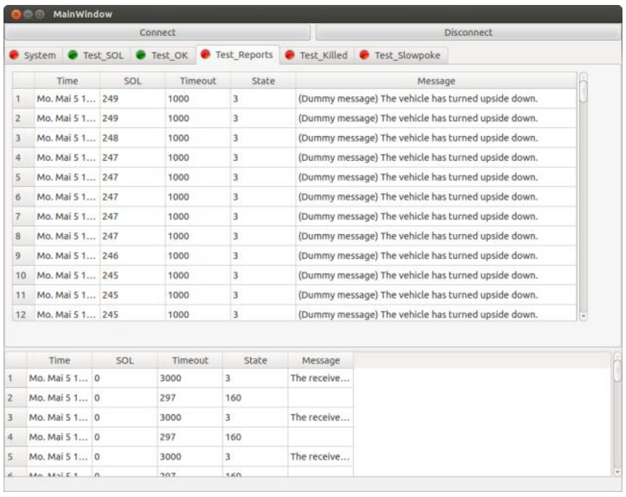
- A component works properly and reports its «alive» status on time (Fig. 4).
- A component sends an internally detected error notification (Fig. 5).
- A component is killed (Fig. 6).
- A component disconnects or delays transmission (Fig. 7).
- Component's messages order is wrong or a packet is lost.



	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	121	1000	0	Alive.
2	Mo. Mai 5 1...	120	1000	0	Alive.
3	Mo. Mai 5 1...	120	1000	0	Alive.
4	Mo. Mai 5 1...	120	1000	0	Alive.
5	Mo. Mai 5 1...	120	1000	0	Alive.
6	Mo. Mai 5 1...	120	1000	0	Alive.
7	Mo. Mai 5 1...	120	1000	0	Alive.
8	Mo. Mai 5 1...	119	1000	0	Alive.
9	Mo. Mai 5 1...	119	1000	0	Alive.
10	Mo. Mai 5 1...	119	1000	0	Alive.
11	Mo. Mai 5 1...	119	1000	0	Alive.
12	Mo. Mai 5 1...	119	1000	0	Alive.

	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	0	3000	3	The receive...
2	Mo. Mai 5 1...	0	297	160	
3	Mo. Mai 5 1...	0	3000	3	The receive...
4	Mo. Mai 5 1...	0	297	160	
5	Mo. Mai 5 1...	0	3000	3	The receive...

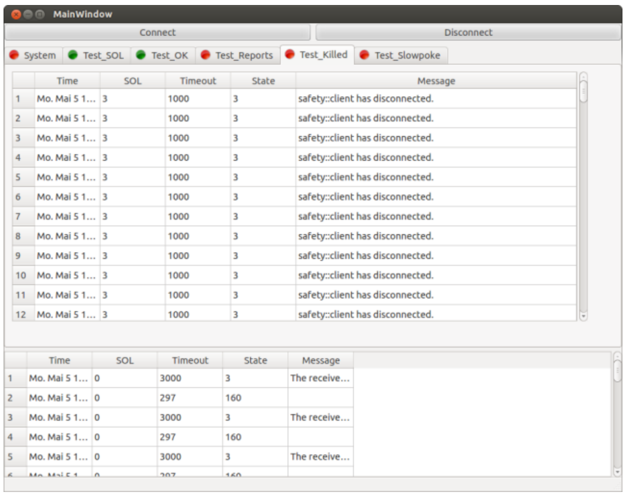
Fig 4. A component that works normally



	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	249	1000	3	(Dummy message) The vehicle has turned upside down.
2	Mo. Mai 5 1...	249	1000	3	(Dummy message) The vehicle has turned upside down.
3	Mo. Mai 5 1...	248	1000	3	(Dummy message) The vehicle has turned upside down.
4	Mo. Mai 5 1...	247	1000	3	(Dummy message) The vehicle has turned upside down.
5	Mo. Mai 5 1...	247	1000	3	(Dummy message) The vehicle has turned upside down.
6	Mo. Mai 5 1...	247	1000	3	(Dummy message) The vehicle has turned upside down.
7	Mo. Mai 5 1...	247	1000	3	(Dummy message) The vehicle has turned upside down.
8	Mo. Mai 5 1...	247	1000	3	(Dummy message) The vehicle has turned upside down.
9	Mo. Mai 5 1...	246	1000	3	(Dummy message) The vehicle has turned upside down.
10	Mo. Mai 5 1...	245	1000	3	(Dummy message) The vehicle has turned upside down.
11	Mo. Mai 5 1...	245	1000	3	(Dummy message) The vehicle has turned upside down.
12	Mo. Mai 5 1...	245	1000	3	(Dummy message) The vehicle has turned upside down.

	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	0	3000	3	The receive...
2	Mo. Mai 5 1...	0	297	160	
3	Mo. Mai 5 1...	0	3000	3	The receive...
4	Mo. Mai 5 1...	0	297	160	
5	Mo. Mai 5 1...	0	3000	3	The receive...

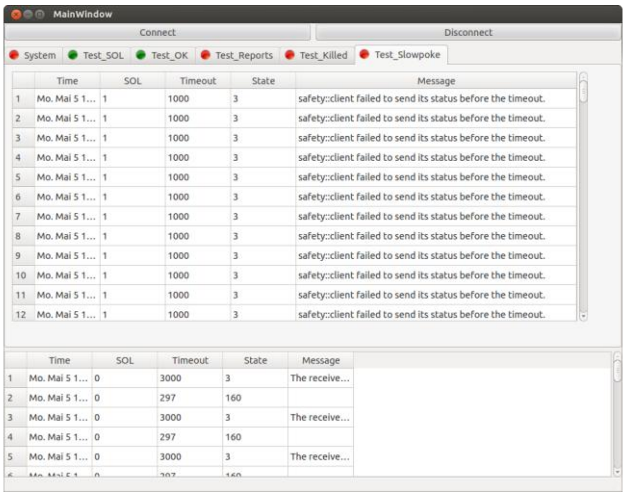
Fig 5. The component that reports error



	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
2	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
3	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
4	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
5	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
6	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
7	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
8	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
9	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
10	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
11	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.
12	Mo. Mai 5 1...	3	1000	3	safety::client has disconnected.

	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	0	3000	3	The receive...
2	Mo. Mai 5 1...	0	297	160	
3	Mo. Mai 5 1...	0	3000	3	The receive...
4	Mo. Mai 5 1...	0	297	160	
5	Mo. Mai 5 1...	0	3000	3	The receive...

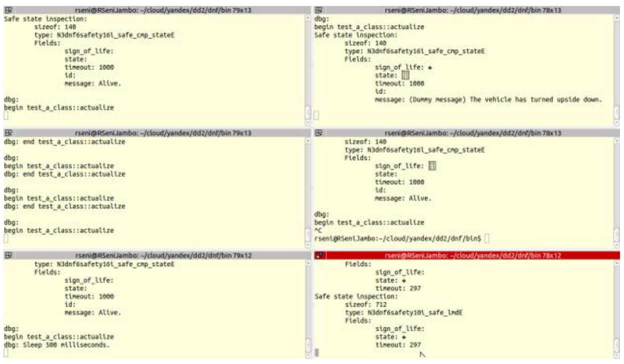
Fig 6. The disconnected component



	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
2	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
3	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
4	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
5	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
6	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
7	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
8	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
9	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
10	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
11	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.
12	Mo. Mai 5 1...	1	1000	3	safety::client failed to send its status before the timeout.

	Time	SOL	Timeout	State	Message
1	Mo. Mai 5 1...	0	3000	3	The receive...
2	Mo. Mai 5 1...	0	297	160	
3	Mo. Mai 5 1...	0	3000	3	The receive...
4	Mo. Mai 5 1...	0	297	160	
5	Mo. Mai 5 1...	0	3000	3	The receive...

Fig 7. The component that fails to send its state in time



```
rseni@RSeniJambo: ~/cloud/yandex/dd2/dnf/bin 79x13
Safe state inspection:
        sizeof: 140
        type: N3dnf6safety16l_safe_cmp_stateE
        Fields:
                sign_of_life:
                state:
                timeout: 1000
                id:
                message: Alive.

dbg:
begin test_a_class::actualize
```

```
rseni@RSeniJambo: ~/cloud/yandex/dd2/dnf/bin 78x13
dbg:
begin test_a_class::actualize
Safe state inspection:
        sizeof: 140
        type: N3dnf6safety16l_safe_cmp_stateE
        Fields:
                sign_of_life: ◆
                state: ▯
                timeout: 1000
                id:
                message: (Dummy message) The vehicle has turned upside down.
```

```
rseni@RSeniJambo: ~/cloud/yandex/dd2/dnf/bin 79x13
dbg: end test_a_class::actualize

dbg:
begin test_a_class::actualize
dbg: end test_a_class::actualize

dbg:
begin test_a_class::actualize
dbg: end test_a_class::actualize

dbg:
begin test_a_class::actualize
```

```
rseni@RSeniJambo: ~/cloud/yandex/dd2/dnf/bin 78x13
        sizeof: 140
        type: N3dnf6safety16l_safe_cmp_stateE
        Fields:
                sign_of_life: ▯
                state:
                timeout: 1000
                id:
                message: Alive.

dbg:
begin test_a_class::actualize
^C
rseni@RSeniJambo:~/cloud/yandex/dd2/dnf/bin$
```

```
rseni@RSeniJambo: ~/cloud/yandex/dd2/dnf/bin 79x12
        type: N3dnf6safety16l_safe_cmp_stateE
        Fields:
                sign_of_life:
                state:
                timeout: 1000
                id:
                message: Alive.

dbg:
begin test_a_class::actualize
dbg: Sleep 500 milliseconds.
```

```
rseni@RSeniJambo: ~/cloud/yandex/dd2/dnf/bin 78x12
        Fields:
                sign_of_life:
                state: ◆
                timeout: 297
Safe state inspection:
        sizeof: 712
        type: N3dnf6safety10l_safe_lmdE
        Fields:
                sign_of_life:
                state: ◆
                timeout: 297
```

Fig 8. Six terminals with five running components and the Safety monitor

5. CONCLUSION. PROJECT STATUS

We designed and prototyped Safety monitoring framework for error detection in a particular distributed real-time system.

Future work includes:

- Framework testing in a joint project S3ARV of TU Dresden and Uni Stuttgart;
- Development of a scalable open-source version of the framework for some kinds of real-time systems in general;
- Development of an error identification / isolation model, e. g. back error propagation model and its connection with the framework.

REFERENCES

1. **Avižienis, Laprie, Randell** "Fundamental Concepts of Dependability", 2001.
2. **Andrea Guiotto, Andrea Martelli, Carlo Paccagnini, Michelle Lavagna** "SMART-FDIR: use of Artificial Intelligence in the implementation of a Satellite FDIR", 2003.
3. **Niklas Holsti, Matti Paakko** "Towards Advanced FDIR Components", 2001.
4. **Oliver Arafat, Andreas Bauer, Martin Leucker, and Christian Schallhart** "Runtime verification revisited". Technical Report TUM-I05, Technical University of Munich, 2005.
5. **Havelund K., Rosu G.** "Synthesizing monitors for safety properties." In: Tools and Algorithms for Construction and Analysis of Systems, pages 342–356, 2002.
6. **Barringer H., Rydeheard D., Havelund K.** "Rule systems for run-time monitoring: From eagle to ruler". In RV07: Proceedings of Runtime Verification 2007, number 4839 in Lecture Notes in Computer Science, pages 111–125. Springer-Verlag, 2007.
7. **Havelund K.** "Runtime verification of C programs." In TestCom/FATES, number 5047 in Lecture Notes in Computer Science. Springer-Verlag, 2008.
8. **D'Angelo B., Sankaranarayanan S., Sanchez C., Robinson W., Manna Z., Finkbeiner B., Spima H., Mehrotra S.** "LOLA: Runtime monitoring of synchronous systems". In 12th International Symposium on Temporal Representation and Reasoning, pages 166–174. IEEE Computer Society Press, 2005.
9. **Lee C.** "Monitoring and Timing Constraints and Streaming Events with Temporal Uncertainties." PhD thesis, University of Texas, 2005.

ABOUT AUTHORS

RASHIDOV, Arseniy, Post-graduate student in the Dept. of Computer Science and Robotics, Ufa State Aviation Technical University (USATU). Dipl.-Inf. degree in Math and CS (USATU, 2014).

MOROZOV, Andrey, Postdoc researcher at Institute of Automation (IfA), ET/IT, TU Dresden, Dipl-Inf. degree in Math and CS (USATU 2007), Dr.-Ing. degree in the field of system dependability (TUD 2012).

JANSCHEK, Klaus, Professor in the Institute of Automation, Technische Universität Dresden. Dipl.-Ing. degree in Electrical Engineering (Technische Universität Graz, Austria, 1978). Dr.techn. degree (with distinction) in Electrical Engineering (Automatic Control, Prof. G. Schneider, Technische Universität Graz, Austria, 1982).

YUSUPOVA, Nafisa, Professor, Dr.-Eng. Dean of the Faculty of Computer Science and Robotics, Ufa State Aviation Technical University (USATU), Head of the Dept. of Computational Mathematics and Cybernetics. Diploma in radiophysics (Voronezh State University, 1975). Dr.-Eng. (USATU, 1998).

МЕТАДАННЫЕ

Название: Сопутствующее обнаружение ошибок в распределённых системах: практическое исследование.

Авторы: А. О. Рашидов, А. М. Морозов, К. Янчек, Н. И. Юсупова

Организации: Технический университет Дрездена, Германия. ФГБОУ ВПО «Уфимский государственный авиационный технический университет», Россия.

Email: arseniy.rashidov@gmail.com, andrey.morozov@tu-dresden.de, klaus.janschek@tu-dresden.de, yussupova@ugatu.ac.ru.

Язык: анлийский.

Источник: Вестник УГАТУ. 2014. Т. 18, № 5 (66). С. 68–72. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

Аннотация: Обнаружение ошибок определено как наблюдение за работой системы, чтобы удостовериться в соответствии её функционирования с ожидаемым поведением. Согласно стандартам IEEE обнаружение ошибок — одно из средств достижения отказоустойчивости. Следовательно, обнаружение ошибок — необходимая часть процесса разработки чувствительных к отказам систем. Сопутствующее обнаружение ошибок в распределённых системах — одна из проблем, исследуемых в рамках совместного проекта S3ARV (Small & Safe Space Autonomous Robotic Vehicles) IfA, TU Dresden и iFR, Uni Stuttgart. В представленной статье авторы вводят новый фреймворк обеспечения безопасности, основанный на сопутствующем обнаружении ошибок. Наш подход направлен на наблюдение за распределённым управляющим программным обеспечением реального времени. Прототип фреймворка применён на летательном аппарате (октокоптере).

Кючевые слова: FDIR; обнаружение ошибок; монитор безопасности; распределённые системы.

Об авторах:

РАШИДОВ Арсений Олегович, асп. каф. выч. мат. и кибернетики. Дипл. мат.-программист (УГАТУ, 2014).

МОРОЗОВ Андрей Михайлович, исследователь (Postdoc) Института автоматизации. Дипл. мат.-программиста (УГАТУ, 2007), Dr.-Ing. в области надёжности систем (Техн. ун-т Дрездена, 2012).

ЯНЧЕК Клаус, проф. Института автоматизации. Дипл. инж. в обл. электроинженерии (Техн. ун-т г. Грац, Австрия, 1978). PhD (с отличием) в обл. электроинженерии (там же, 1982).

ЮСУПОВА Нафиса Исламовна, декан ф-та информатики и робототехники, зав. каф. выч. мат. и кибернетики. Дипл. радиофизик (Воронежск. гос. ун-т, 1975). Д-р техн. наук по упр. в техн. системах (УГАТУ, 1998). Иссл. в обл. ситуац. управления и информатики.