Emerging Methodologies and Applications in Modelling, Identification and Control

Series Editor: Quan Min Zhu

System Assurances

Modeling and Management

Edited by
Prashant Johri, Adarsh Anand, Jüri Vain,
Jagvinder Singh and Mohammad Tabrez Quasim

Contents

Contributors xxi
Preface xxv
Acknowledgement xxvii

1. **Statistical analysis approach for the quality assessment of open-source software**
Yoshinobu Tamura and Shigeru Yamada

1.1 Introduction 1
1.2 Correspondence analysis 1
1.3 Estimation procedure based on correspondence analysis 3
1.4 Numerical examples 3
1.5 Concluding remarks 7
Acknowledgment 8
References 8

2. **Analytical modeling and performance evaluation of SIP signaling protocol: Analytical modeling of SIP**
Nikesh Choudhary, Vandana Khaitan (nee Gupta), and Vaneeta Goel

2.1 Introduction 11
2.2 Background work 12
2.2.1 Motivation 13
2.2.2 Main contribution 14
2.3 SIP layered structure and its working 14
2.3.1 Types of responses of SIP 14
2.3.2 Layered structure and working of SIP 15
2.4 Proposed SRN model of SIP INVITE transaction 18
2.4.1 SRN and its attributes 18
2.4.2 Proposed SRN model of SIP INVITE transaction 20
2.5 Performance measures 31
2.5.1 Throughput 32
2.5.2 Latency 34
2.6 Numerical illustration 35
2.6.1 Results and discussion 35
2.7 Model validation 37
2.8 Conclusions 38
Acknowledgments 39
References 39

12.5 **The integrated efficiency model by using the dynamic programming method for the function** $c_j = b_j e^{\left[\frac{a_j}{(1-r_j)}\right]}$ 205
12.5.1 Dynamic programming solution 205
12.6 **Conclusions** 207
Further reading 208

13. **Modeling and availability assessment of smart building automation systems with multigoal maintenance**
Yuriy Ponochovniy, Vyacheslav Kharchenko, and Olga Morozova

13.1 **Introduction** 209
13.1.1 Motivation 209
13.1.2 Work-related analysis 210
13.1.3 Goals and contribution 211
13.2 **Concept of multigoal maintenance** 211
13.2.1 Principles of multigoal maintenance 212
13.2.2 Assessment of availability 213
13.3 **Development of models** 214
13.3.1 General model 214
13.3.2 Availability model considering reliability and cybersecurity 214
13.3.3 Availability model considering joint maintenance strategy 217
13.3.4 Availability model considering separate maintenance strategy 219
13.4 **Research of models** 219
13.5 **Conclusions** 223
References 226

14. **A study of bitcoin and Ethereum blockchains in the context of client types, transactions, and underlying network architecture**
Rohaila Naaz and Ashendra Kumar Saxena

14.1 **Blockchain: An insight** 230
14.2 **Blockchain architectures** 230
14.2.1 Infrastructure layer 232
14.2.2 Platform layer 234
14.3 **Blockchain adequacy** 236
14.3.1 Data integrity 238
14.3.2 Scalability 238

Chapter 13

Modeling and availability assessment of smart building automation systems with multigoal maintenance

Yuriy Ponochovnyi[a], Vyacheslav Kharchenko[b], and Olga Morozova[b]
[a]*Department of Information Systems and Technologies, Poltava State Agrarian University, Poltava, Ukraine,* [b]*Department of Computer Systems, Networks and Cybersecurity, National Aerospace University "KhAI", Kharkiv, Ukraine*

13.1 Introduction

13.1.1 Motivation

The rapid development of virtualization technologies and the cloud computing environment creation have led to the development and implementation of new versions of IT-systems architectures, for which it is necessary to take into account, evaluate, and provide indicators of computer systems and Web Services dependability. These include various systems, in particular smart building automation system (BAS).

Automation of houses is performed for efficient and safe maintenance of all technical systems of the residential or industrial building. Modern devices and control modules control (block, regulate) working of all equipment used for the automation of houses. Software control is carried out for preset conditions using the centralized system [1].

The evolution of BAS is interrelated with the development of the concept of the Internet of Things (IoT) [2], [3] and includes the following stages:

- "Building Automation" (lighting, climate and music equipment are connected to the data network, begin in 2010);
- "Connected Building" (cloud technologies and advanced capabilities over the Internet, begin in 2017);
- "Adaptive Building" (data analysis, event forecasting, adequate reactions, begin in 2021);

System Assurances. https://doi.org/10.1016/B978-0-323-90240-3.00013-8
Copyright © 2022 Elsevier Inc. All rights reserved.

- "Intellectual Building" (artificial intelligence, virtual reality, digital consciousness, predict begin in 2025).

The set of "smart building" subsystems that perform information and control functions is considered as building automation system [3]. The architecture of the BAS [4] includes controller levels (on microprocessor or programmable logic device), databases, and communications (wired and/or wireless).

An important component of information and control systems (ICS) (these include BAS) quality is dependability. Under dependability, in the general case, the complex property is the ability of the system to provide the necessary services that can be justifiably trusted [5,6]. There are a few taxonomic models of dependability for different applications [5–8]. The dependability combines groups of primary properties (reliability, availability, maintainability, high confidence, functional safety, survivability, integrity, confidentiality) and secondary properties (for example, information and cybersecurity, authenticity).

The complex nature of external factors in relation to the system is the rationale for choosing the concept of the dependability. In this case, the property of ICS and BAS dependability will be considered as the ability of the system to perform specified functions (provide the required services), maintaining over time the level of its defined indicators within specified limits and operating conditions, as well as changing requirements, environmental parameters, and the manifestation of unspecified failures.

The indicators of the BAS are determined by the properties (including the properties of the dependability) of its hardware and software components [9]. The factors that affect the dependability of the system [10,11] are divided into internal (reliability) and external (cybersecurity) factors. The failures and errors of the programmed logic automated means as well as the manifestation of production faults are considered as internal system factors. External factors include vandalism and attacks on software vulnerabilities (hidden intervention, cyberattacks).

13.1.2 Work-related analysis

The concept of dependability as complex property joining reliability, availability, security, and safety is a chronological and consistent development of the proposed [5]. The problem of integrating complex property of dependability definition has been revealed in Refs. [12–15], the issues of dependability web-services modeling in Refs. [8,16], and researched system dependability models of critical infrastructures revealed in Refs. [7,17]. The models of software components of dependability systems were developed in Refs. [18,19].

Different mathematical apparatus are used to assess the dependability indicators such as Monte Carlo method [18], stochastic reward nets Petri model based on package SHARPE [14,20], and models of multistate systems [21]. Issues in ensuring the reliability and cybersecurity of smart buildings were considered in Refs. [22–24] and others.

The analysis of research results provides an opportunity to identify specific problems that need to be addressed in terms of development and application of mathematical apparatus for dependability systems with different goals and strategies for maintenance and updates.

13.1.3 Goals and contribution

The goal of this study is to assess the availability functions (as an indicator of dependability) of maintained BAS in terms of their evolution as a result of occurrence of hardware and software failures and unspecified software failures (as attacks on vulnerabilities). To achieve this goal the following tasks are performed: development and analysis the principles of multigoal maintenance concept (Section 13.2), development of multifragment models for assessing the BAS availability (Section 13.3), study of simulation results for interval values of partial input parameters (Section 13.4), and conclusions and suggestions for further research (Section 13.5).

An approach to research is based on three main ideas and steps. First, we suggest principles of multigoal maintenance, in particular, two-goal maintenance of BAS and similar systems considering reliability- and cybersecurity-oriented maintenance activities performed separately (only for the reliability assurance or for cybersecurity assurance) or jointly (for reliability and security assurance).

Second idea is based on general model such as structural graph for describing system behavior in case of different failures caused by physical (hardware) and design (software) faults and attacks on vulnerabilities, and implementation of different maintenance strategies (separate or joint maintenance for the assurance of reliability and/or cybersecurity).

The next step is developing and researching Markov's models according to general model and maintenance strategies.

Hence, the contributions of the research are the development of a set of maintenance strategies; the models of BAS availability considering different failures and strategies; and recommendations concerning the choice of parameters of system, its components, and maintenance strategies.

13.2 Concept of multigoal maintenance

To ensure the normalized indicators of smart building BAS availability, it is proposed to use the concept of dependable ICS with multigoal maintenance [25]. It is obtained by developing the paradigm of von Neumann as follows: to build dependable (reliable and secure) systems with not dependable enough components and multigoal maintenance strategies in changing the requirements and the environment parameters. The proposed concept is based on the following conditions:

(1) The development of von Neumann's concept of "building reliable systems from unreliable components" [26] in the direction of ensuring the dependability [7,27], i.e., building dependable ICS from insufficiently reliable and secure/safe components.
(2) The intended use of ICS in changing conditions, changing environmental parameters, and the occurrence of unspecified failures due to physical and design faults and vulnerabilities prevents or significantly complicates the direct application of the von Neumann paradigm for nonmaintenance ICS components and requires updating and restoring the system, and/or giving it the properties of resilience [15,20].
(3) The use of multigoal maintenance with the use of strategies combined on various indicators, in particular, the goal of the maintenance (confirmation of requirements, changes in parameters, changes in functions, prevention of changes in parameters), maintenance processes (verification, update, patching), properties that are maintained through maintenance (reliability, functional safety, cybersecurity).

13.2.1 Principles of multigoal maintenance

The proposed conception is based on the concept of multigoal maintenance strategy, the formation of such strategies, and their choice to ensure dependability. In the multiple aspects the maintenance strategy will be defined as a set of identified components in the form of

$$\text{StrMaint} = \{\{\text{GM}_i\}, \{\text{TM}_i\}, \{\text{ProcM}_i\}, \{\text{PropM}_i\}, \{\text{ParM}_i\}\},$$

where GM is the goal of maintenance, TM is the type of maintenance, ProcM is the process of maintenance, PropM is the properties for which maintenance is performed, and ParM is the parameters of maintenance.

For coverage of the conceptual principles proposed in the work, it was decided to limit by two procedures, namely, patching and proof testing.

Patching (ProcM2) involves the replacement of blocks/modules or sections of program code with detected faults or vulnerabilities. In some cases, corrective actions are taken to disable defective blocks or code sections. At the same time the system degrades but retains a partially operational state.

The key point is to take corrective actions after the fault is detected or manifested as a failure or error.

Proof testing (ProcM4) aims to detect hidden failures and vulnerabilities. Such testing is characterized by parameter D (the probability of detecting hidden failures/vulnerabilities), the value of which may vary and must be justified.

The issues of complexing these maintenance procedures are interrelated with the domain of ICS use (building automation systems) and models of availability assessment. These are discussed in the following sections.

13.2.2 Assessment of availability

The function of ICS availability with multigoal maintenance has the following nature of changes: at the first stage the system availability is reduced to a minimum, then it asymptotically tends to a constant value.

Thus, in the further analysis of the results it is necessary to take into account three indicators:

- the minimum value of the availability function $A_M$min;
- the value of the availability function in the constant mode $A_M$const;
- time interval of transition of the availability function to the constant mode $T_M$const.

In the case of using the proposed models to compare service strategies, additional result indicators were used (Fig. 13.1):

- reduction of the system availability level with maintenance at the initial stage of operation relative to the coefficient of similar system availability of hardware and software system without maintenance $-\Delta Ai$;
- gain in the system availability with maintenance relatively similar in terms of hardware and software system without maintenance $+\Delta Ai$;
- time, after which there is a gain in the system availability with maintenance Ti_{up}.

To calculate these additional result indicators for a group of models with multigoal maintenance requires development and study of at least two additional supporting models.

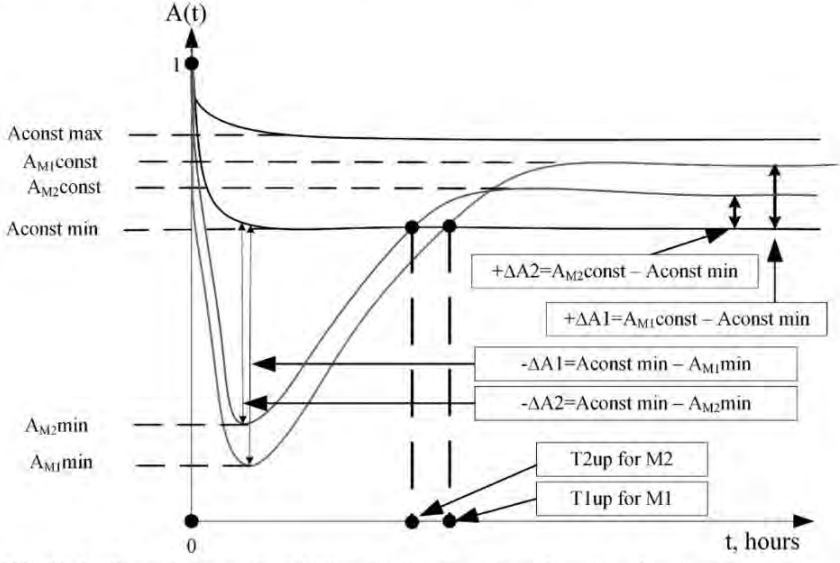

FIG. 13.1 Dependence graph of ICS availability function with multigoal maintenance.

13.3 Development of models

13.3.1 General model

The model of multigoal maintenance states in the context of patching and proof testing procedures is shown in Fig. 13.2. The oriented state graph of this model contains S_0 a fragment that includes operational states, S_1 a state that simulates patching, and S_2 a state that simulates the conduct of proof testing. Given the specifics of application domains, this model can be modified in the following scenarios:

(1) multifragment model, which describes only the patching procedures, after which the parameters of the ICS change (according to this scenario, the tests after the installation of patches are modeled not by a separate state S_1, but are taken into account when weighing the transition $S_1 \rightarrow S_0{}^*$);
(2) one-fragment model, which describes only the procedures of proof testing, because their implementation does not change the parameters of the ICS;
(3) multifragment model, which includes both procedures (patching and proof testing), and these procedures are carried out sequentially: after installing the patch is performed proof testing of ICS components;
(4) multifragment model, which includes patching and proof testing procedures that are performed independently of each other.

This section will specify the options for building models in the fourth and second scenarios.

Model $M_{DEP}2.2^{(i)}$ describes BAS in the event of hardware and software failures caused by physical faults (Fp), design faults (Fd), and vulnerabilities of components (Fai). It summarizes the BAS models developed by studies in Refs. [10,11,28] and expands them with space of hardware faults for database server architecture components and wireless modules.

13.3.2 Availability model considering reliability and cybersecurity

Marked graph of the model $M_{DEP}2.2^{(1)}$ is shown in Fig. 13.3.

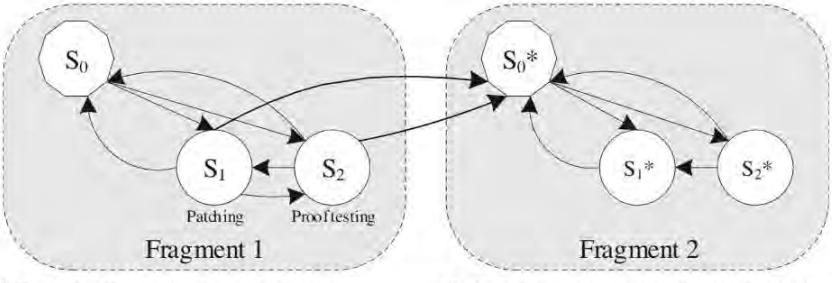

FIG. 13.2 Generalized model of patching and proof testing states of dependability ICS in the form of a two-fragment graph.

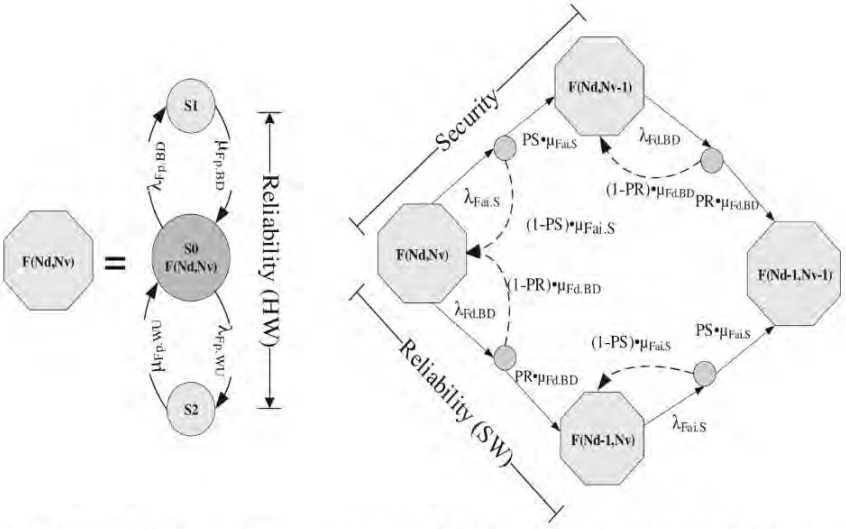

FIG. 13.3 Marked graph of the model $M_{DEP}2.2^{(1)}$ with the elimination of one SW fault and one vulnerability.

Dependability model $M_{DEP}2.2^{(1)}$ describes the operation of smart building BAS with a three-tier architecture:

(1) the level of sensors with controllers (sensor level);
(2) the level of wireless communications (wireless unit level);
(3) the level of data servers (BD Level).

BAS operates in the conditions of physical fault manifestation of components of the second and third levels (physical faults of components of the first level practically do not affect the system dependability as a whole), design faults of software components of the third level, and attacks on software components of the first level. On the marked graph these types of failures are illustrated by transitions with intensities $\lambda_{Fp.BD}$, $\lambda_{Fp.WU}$, $\lambda_{Fd.BD}$, $\lambda_{Fai.S}$.

The assumptions of the simplest flow of failures and recoveries that change the state of the system are accepted when developing the model. These assumptions are substantiated by studies in Refs. [12,14] and confirmed by the data of failures of real systems given in Refs. [11]. After the onset of hardware failure, the repair of the corresponding component is performed (in the graph of Fig. 13.3 it is shown by transitions with intensities $\mu_{Fp.BD}$, $\mu_{Fp.WU}$).

After the manifestation of the design fault, the system with the probability PR stops working until its complete elimination, or in the system with the probability (1-PR) the BAS returns to the previous working state due to the restart of the program. The same mechanism simulates the actions after an attack on the vulnerability of the component of the first level of the BAS architecture: with the probability PS the vulnerability will be eliminated, with the probability

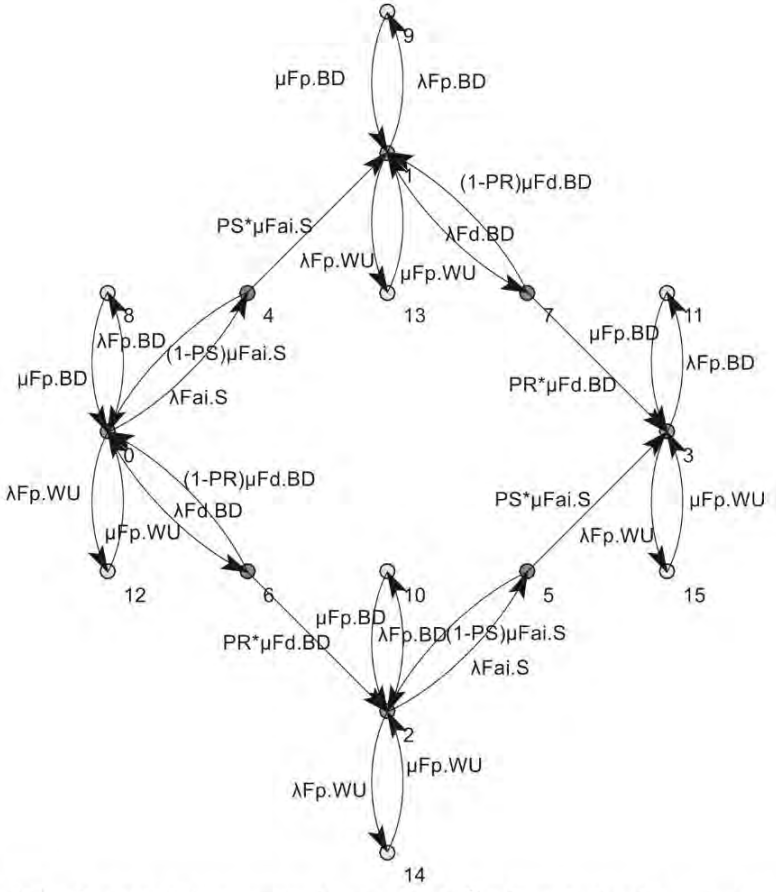

FIG. 13.4 The marked directed graph of model $M_{DEP}2.2^{(1)}$ with numbering of states, constructed by means of function grPlot_marker.

(1-PS) the BAS will continue to function with vulnerability after system restart. The system stops functioning to completely eliminate the software design faults and vulnerabilities of the first level architecture components with PR = 1 and PS = 1. Thus, model $M_{DEP}2.2^{(1)}$ describes the procedures for patching the software code of the BAS architecture components. But with PR = 0 and PS = 0 the system operates only in the conditions of software restart without elimination of design faults and vulnerabilities of components. The marked directed graph of states and transitions (Fig. 13.4) includes end-to-end numbering of states, constructed using the modified function grPlot_marker [28].

Kolmogorov's system of differential equations, built on the graph of the model $M_{DEP}2.2^{(1)}$, are expressed by

$$
\begin{cases}
dP_0(t)/dt = -\left(\lambda_{\text{Fp.BD}} + \lambda_{\text{Fp.WU}} + \lambda_{\text{Fd.BD}} + \lambda_{\text{Fai.S}}\right)P_0(t) + \mu_{\text{Fp.BD}}P_8(t) + \mu_{\text{Fp.WU}}P_{12}(t) \\
\qquad + (1-PS)\mu_{\text{Fai.S}}P_4(t) + (1-PR)\mu_{\text{Fd.BD}}P_6(t), \\
dP_1(t)/dt = -\left(\lambda_{\text{Fp.BD}} + \lambda_{\text{Fp.WU}} + \lambda_{\text{Fd.BD}}\right)P_1(t) + \mu_{\text{Fp.BD}}P_9(t) + \mu_{\text{Fp.WU}}P_{13}(t) \\
\qquad + (1-PR)\mu_{\text{Fd.BD}}P_7(t) + PS\mu_{\text{Fai.S}}P_4(t), \\
dP_2(t)/dt = -\left(\lambda_{\text{Fp.BD}} + \lambda_{\text{Fp.WU}} + \lambda_{\text{Fai.S}}\right)P_2(t) + \mu_{\text{Fp.BD}}P_{10}(t) + \mu_{\text{Fp.WU}}P_{14}(t) \\
\qquad + (1-PS)\mu_{\text{Fai.S}}P_5(t) + PR\mu_{\text{Fd.BD}}P_6(t), \\
dP_3(t)/dt = -\left(\lambda_{\text{Fp.BD}} + \lambda_{\text{Fp.WU}}\right)P_3(t) + \mu_{\text{Fp.BD}}P_{11}(t) + \mu_{\text{Fp.WU}}P_{15}(t) \\
\qquad + PS\mu_{\text{Fai.S}}P_5(t) + PR\mu_{\text{Fd.BD}}P_7(t), \\
dP_4(t)/dt = -\mu_{\text{Fai.S}}P_4(t) + \lambda_{\text{Fai.S}}P_0(t), \\
dP_5(t)/dt = -\mu_{\text{Fai.S}}P_5(t) + \lambda_{\text{Fai.S}}P_2(t), \\
dP_6(t)/dt = -\mu_{\text{Fd.BD}}P_6(t) + \lambda_{\text{Fd.BD}}P_0(t), \\
dP_7(t)/dt = -\mu_{\text{Fd.BD}}P_7(t) + \lambda_{\text{Fd.BD}}P_1(t), \\
dP_{8..11}(t)/dt = -\mu_{\text{Fp.BD}}P_{8..11}(t) + \lambda_{\text{Fp.BD}}P_{0..3}(t), \\
dP_{12..15}(t)/dt = -\mu_{\text{Fp.WU}}P_{12..15}(t) + \lambda_{\text{Fp.WU}}P_{0..3}(t), \\
\sum_{i=0}^{20} P_i(t) = 1;
\end{cases}
$$

$$
P_0(0) = 1; \quad \forall i \in [1 \ldots 15] \Rightarrow P_i(0) = 0. \tag{13.1}
$$

The standby function is defined as

$$
A(t) = \sum_{i=0}^{(\text{Nd}+1)\cdot(\text{Nv}+1)-1} P_i(t). \tag{13.2}
$$

13.3.3 Availability model considering joint maintenance strategy

Availability model $M_{DEP}2.2^{(2)}$ expands the previous model with additional procedures of the general proof maintenance during which both detection and elimination of software faults of database servers and vulnerabilities of sensors of the first level of architecture of ICS are possible. The marked graph of the model is shown in Fig. 13.5. It is a generalization of the MBAS2.2 availability model proposed in Ref. [11] and takes into account hardware failures caused by physical faults.

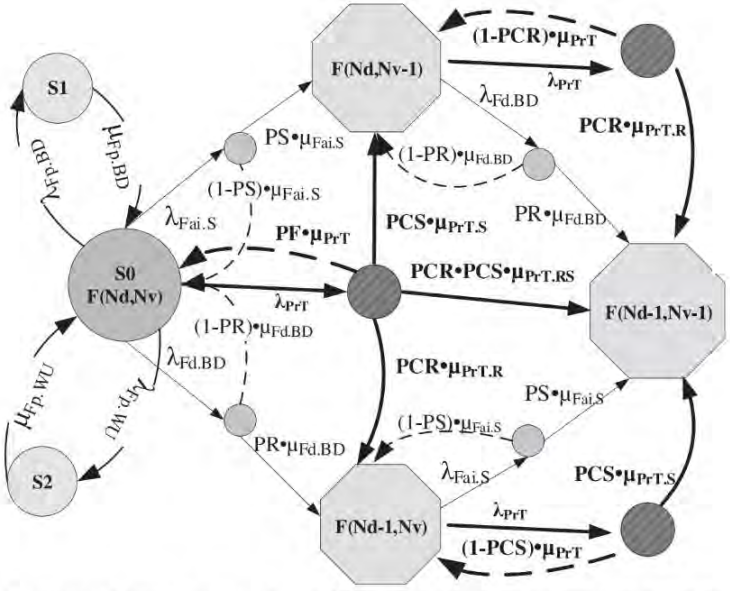

FIG. 13.5 Marked graph of model $M_{DEP}2.2^{(2)}$ with elimination of one software design faults and one vulnerability at two procedures of the general proof testing.

The condition describing the procedure of general proof testing is shown by a shaded brown circle. The transition to the maintenance state is carried out with the intensity λ_{PrT}. The PCR and PCS parameters correspond to the probabilities of detecting a software fault or vulnerability during maintenance. The detection of software faults and vulnerabilities is carried out with the probability PCR*PCS, and detection of both the reasons for refusal is carried out with the probability PF, moreover.

$$\mathrm{PCS}^{*}\mathrm{PCR}+\mathrm{PCS}+\mathrm{PCR}+\mathrm{PF}=1.$$

The following transitions are possible from the state of proof testing:

(a) transition up with intensity $\mathrm{PCS}^{*}\mu_{\mathrm{PrT.S}}$ ($\mu_{\mathrm{PrT.S}}$ is the value inverse of the average time of detection and elimination of vulnerabilities [12]),

$$\mu_{\mathrm{PrT.S}}=1/\left(T_{\mathrm{det.V}}+T_{\mathrm{rem.V}}\right);$$

(b) transition down with the intensity $\mathrm{PCR}^{*}\mu_{\mathrm{PrT.R}}$ ($\mu_{\mathrm{PrT.R}}$ is the value inverse of the average time of detection and elimination of software design faults of database server [29]),

$$\mu_{\mathrm{PrT.R}}=1/(T\,\mathrm{det.D}+T\mathrm{rem.D});$$

(c) transition to the right with intensity PCS*PCR* $\mu_{PrT.RS}$,

$$\mu_{PrT.RS} = \frac{\mu_{PrT.R} \cdot \mu_{PrT.S}}{\mu_{PrT.R} + \mu_{PrT.S}} \tag{13.3}$$

(d) transition to the left with intensity PF*μ_{PrT}, where μ_{PrT} is the value inverse of the average time of the general proof testing procedure, $\mu_{PrT} = 1/T_{PrT}$.

After detection and elimination of all design faults, the proof testing procedure is aimed at finding and eliminating only vulnerabilities (provided that there are unresolved vulnerabilities), i.e., it is separate. Therefore, it has only two transitions weighted by the intensities PCR*$\mu_{PrT.R}$ and (1–PCR)*μ_{PrT}.

Similarly, for the case where all vulnerabilities have been eliminated, the proof testing procedure has transitions weighted by the intensities PCS*$\mu_{PrT.S}$ and (1–PCS)*μ_{PrT}.

After identifying and eliminating all design faults and vulnerabilities, the proof testing procedure has one transition with intensity μ_{PrT}.

13.3.4 Availability model considering separate maintenance strategy

Model $M_{DEP}2.2^{(3)}$ is a generalization of the availability model MBAS3.2 proposed in Ref. [11] and takes into account hardware failures caused by physical faults. It extends $M_{DEP}2.2^{(1)}$ model with additional separate proof maintenance procedures, during which it is possible to detect and eliminate only the software faults of the database servers (reliability maintenance) or only the vulnerabilities of the first level sensors of the ICS architecture (cybersecurity maintenance). The marked graph of the model is shown in Fig. 13.6.

13.4 Research of models

The values of the input parameters of model $M_{DEP}2.2$ are substantiated by studies in Refs. [11,12,29] and are given in Table 13.1.

Because models $M_{DEP}2.2^{(2)}$ and $M_{DEP}2.2^{(3)}$ describe two different maintenance strategies, each of which has certain parameters (intensity of conduction, average duration, probability of detecting faults, and vulnerabilities during maintenance), it is necessary to consider them separately and then compare.

Fig. 13.7 shows the results of models $M_{DEP}2.2$ (1, 2, and 3) for the basic values of the input parameters of Table 13.1. The following values of partial result indicators of dependability are received:

- the time when model $M_{DEP}2.2^{(2)}$ begins to give a gain in availability relative to model $M_{DEP}2.2^{(1)}$ Tup1.1 = 963.4 h;

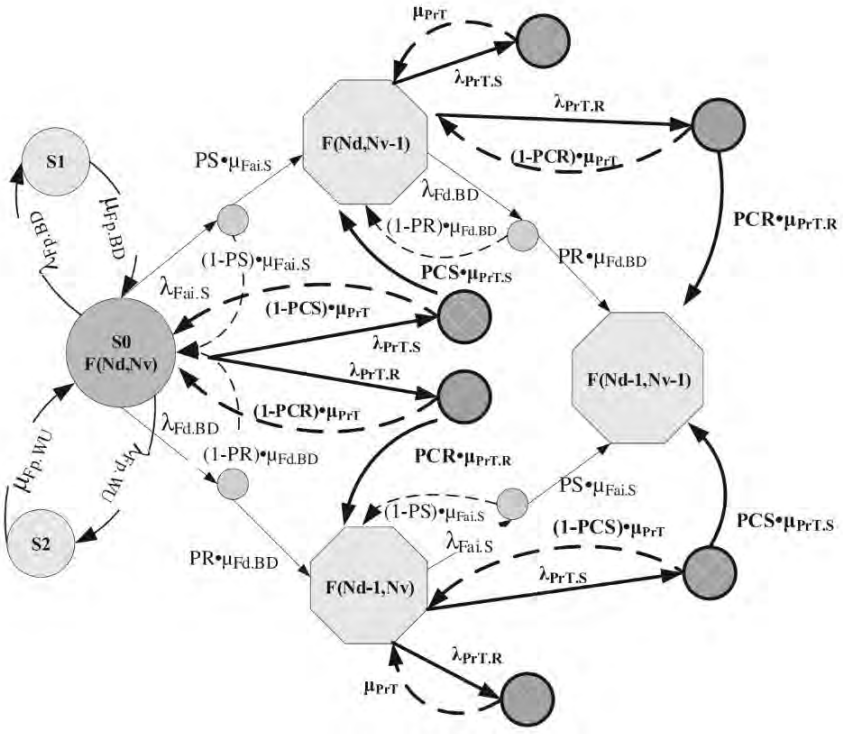

FIG. 13.6 Marked graph of model $M_{DEP}2.2^{(3)}$ with elimination of one software design fault and one vulnerability.

- the time when model $M_{DEP}2.2^{(3)}$ begins to give a gain in availability relative to model $M_{DEP}2.2^{(1)}$ Tup1.2 = 338.2 h;
- the time when models $M_{DEP}2.2^{(2)}$ and $M_{DEP}2.2^{(3)}$ begin to provide a requirement for the availability level of 0.999 Tup2 $\approx$ 1600 h;
- the largest decrease in the availability level of model $M_{DEP}2.2^{(2)}$ relative to model $M_{DEP}2.2^{(1)}$ $\Delta A = 6e-3$;
- the largest decrease in the availability level of model $M_{DEP}2.2^{(3)}$ relative to model $M_{DEP}2.2^{(1)}$ $\Delta A = 5{,}5e-3$.

Fig. 13.8 shows the results of the evaluation of the performance indicators obtained using a model with general proof testing $M_{\mathrm{DEP}}2.2^{(2)}$.

The influence of the values of the input parameter λ_{PrT} on the dynamics of change of partial dependability indicators was investigated.

A linear dependence was found for the partial Amin index, which decreases by 10.52% with a fourfold increase in λ_{PrT}. The nonlinear dependences were found for partial indicators Tup1 and Tup2, at the same time, with an increase in λ_{PrT} to 0.01 1/h and the time Tup2 for which the service system will take precedence over the maintenance system will be the same as the time Tup1

TABLE 13.1 The values of the input parameters of the model $M_{DEP}2.2$.

No.	Parameter	Matlab-variable	The value
1.	The hardware failure rate of the database server component	la_fp_bd	4e – 5 (1/h)
2.	The intensity of hardware recovery of the database server component	mu_fp_bd	1.5 (1/h)
3.	The hardware failure rate of the wireless communications	la_fp_wu	2e – 5 (1/h)
4.	The intensity of hardware recovery of the wireless communications	mu_fp_wu	0.66 (1/h)
5.	Intensity of manifestation of software design faults of the database server	la_fd_bd	[5e – 4 4.5e – 4] (1/h)
6.	Intensity of attack on sensors vulnerability	la_fai_s	[3e – 3 3.5e – 3] (1/h)
7.	Intensity of recovery after manifestation of software design faults of the database server	mu_fd_bd	[0.5 0.4] (1/h)
8.	Intensity of recovery after an attack on sensor vulnerabilities	mu_fai_s	[0.45 0.34] (1/h)
9.	The probability of eliminating the fault of the database server software during recovery	PR	0.87
10.	The probability of eliminating the vulnerability of sensors during recovery	PS	0.68
11.	The number of faults in the database server software in the system	Nd	2
12.	The number of sensor vulnerabilities in the system	Nv	2
13.	Intensity of BAS proof testing for faults and vulnerabilities detection (general maintenance)	la_prt	4e – 3 (1/h)
14.	Intensity of BAS proof testing for vulnerability detection (separate maintenance for vulnerabilities)	la_prt_s	5e – 3 (1/h)
15.	Intensity of BAS proof testing for fault detection (separate maintenance by faults)	la_prt_r	1e – 4 (1/h)
16.	The inverse of the average duration of general service activities	mu_prt	0.4 (1/h)

Continued

TABLE 13.1 The values of the input parameters of the model $M_{DEP}2.2$—cont'd

No.	Parameter	Matlab-variable	The value
17.	Intensity of detection and elimination of sensors vulnerability	mu_prt_s	1 (1/h)
18.	The intensity of detection and elimination of the database server fault	mu_prt_r	1.5 (1/h)
19.	The probability of detecting the vulnerability of sensors during proof testing in the model $M_{DEP}2.2^{(3)}$	PCS	0.9
20.	The probability of detecting a software fault of the database server during proof testing in the model $M_{DEP}2.2^{(3)}$	PCR	0.97
21.	The probability of detecting the vulnerability of sensors in general proof testing in the model $M_{DEP}2.2^{(2)}$	PCS	0.4409
22.	The probability of detecting a software fault of the database server during the general proof testing in the model $M_{DEP}2.2^{(2)}$	PCR	0.388
23.	Predicted number of vulnerability prevention in the model $M_{DEP}2.2^{(3)}$	Nvp	2
24.	The projected number of software design faults prevention in the model $M_{DEP}2.2^{(3)}$	Ndp	2
25.	The projected number of general prevention in the model $M_{DEP}2.2^{(2)}$	Np	4

(time for which a general maintenance system will take precedence over a patching system).

The results of model $M_{DEP}2.2^{(3)}$ research are presented in the form of a graphical dependence of the availability function on the system operation time and partial indicators Tup1, Tup2, Tup3, and Amin in Fig. 13.9.

In the previous study of model $M_{DEP}2.2^{(2)}$, linear dependences of the partial dependability index Amin on the input parameters λ_{PrTr} and λ_{PrTs} were revealed. At the same time, the graph Amin(λ_{PrTs}) has a larger angle of inclination relative

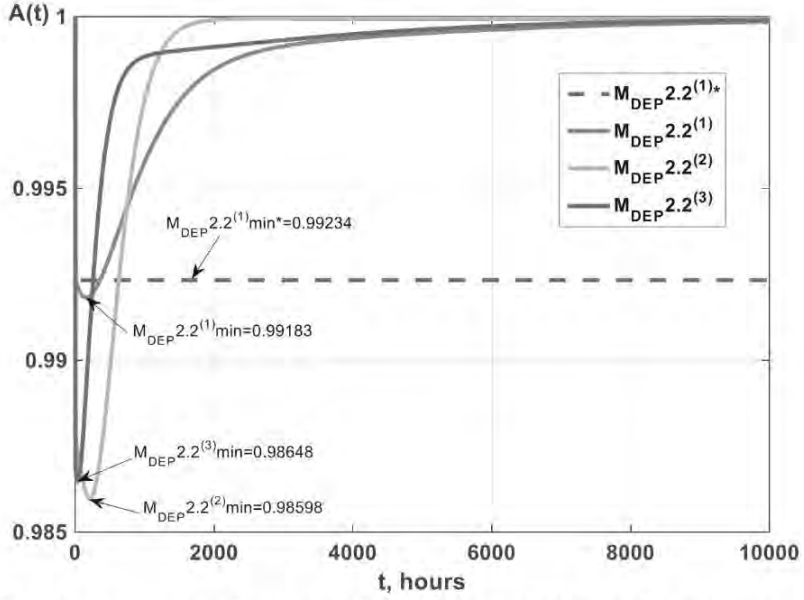

FIG. 13.7 Assessment results of the availability by models $M_{DEP}2.2$ under the baseline values of the parameters ($M_{DEP}2.2^{(1)}$*—model without patching procedures with PR = PS = 0).

to the horizontal axis. The dynamics of partial dependability indicators Tup1 and Tup2 change in the model with separate maintenance will differ significantly from the model with general maintenance (Figs. 13.8B and 13.9B).

As a result of the assessment of the dependability indicators for software patcherization procedures, an increase in accuracy by 0.00636 (Fig. 13.7, models $M_{DEP}2.2^{(2)}$ and $M_{DEP}2.2^{(3)}$) relative to model $M_{DEP}2.2^{(1)}$ was obtained. The results of the assessment of the dependability indicators Amin, Tup1, and Tup2 allowed to determine the impact of service parameters and their values, which meet the requirements for the level of the dependability, in particular to increase the partial Amin by 10.5% (Fig. 13.8, model $M_{DEP}2.2^{(2)}$ relative to the model $M_{DEP}2.2^{(1)}$).

13.5 Conclusions

The multifragment models of dependability BAS in the event of failures caused by hardware physical faults, software design faults, and attacks on component vulnerabilities have been developed. Two strategies of maintenance have been analyzed: separate strategy for reliability and cybersecurity maintenance and assurance (RMA and CMA), and joint strategy for their maintenance and assurance (JMA). The function of BAS availability is used as a main indicator depending on component failure, recovery, and maintenance rates. Three models are developed and researched: the model of BAS without proactive testing and models of system with RMA-CMA and JMA maintenance strategies.

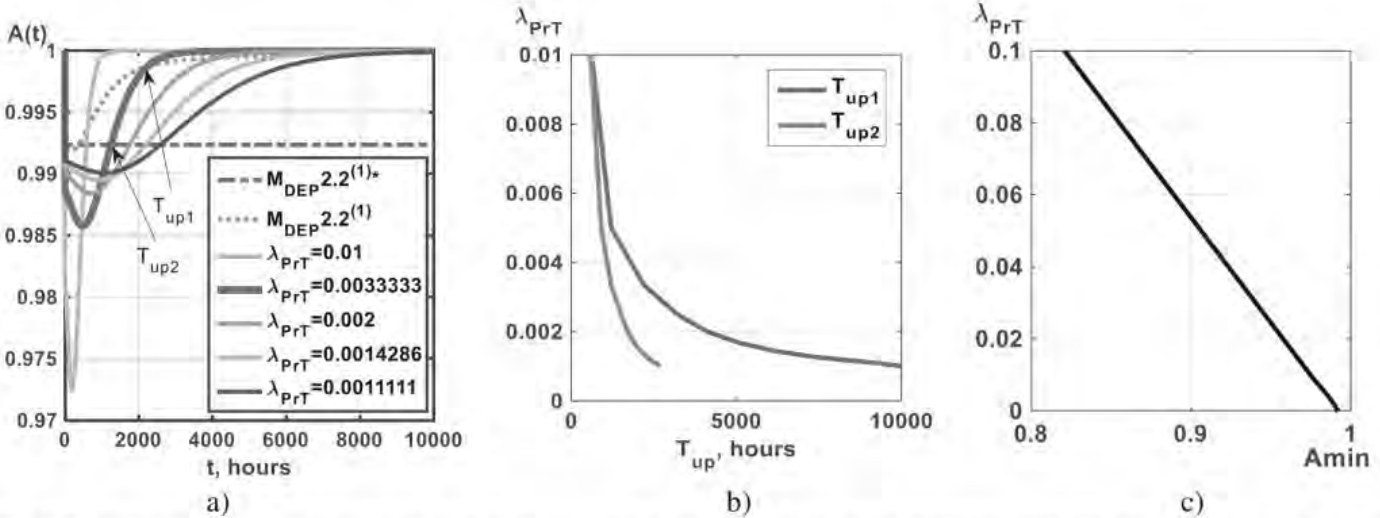

FIG. 13.8 Graphical dependences of availability function change on system operation time (A), partial dependability indicators Tup1 and Tup2 (B), and Amin (C) with different values λ_{PrT}, obtained using model $M_{DEP}2.2^{(2)}$.

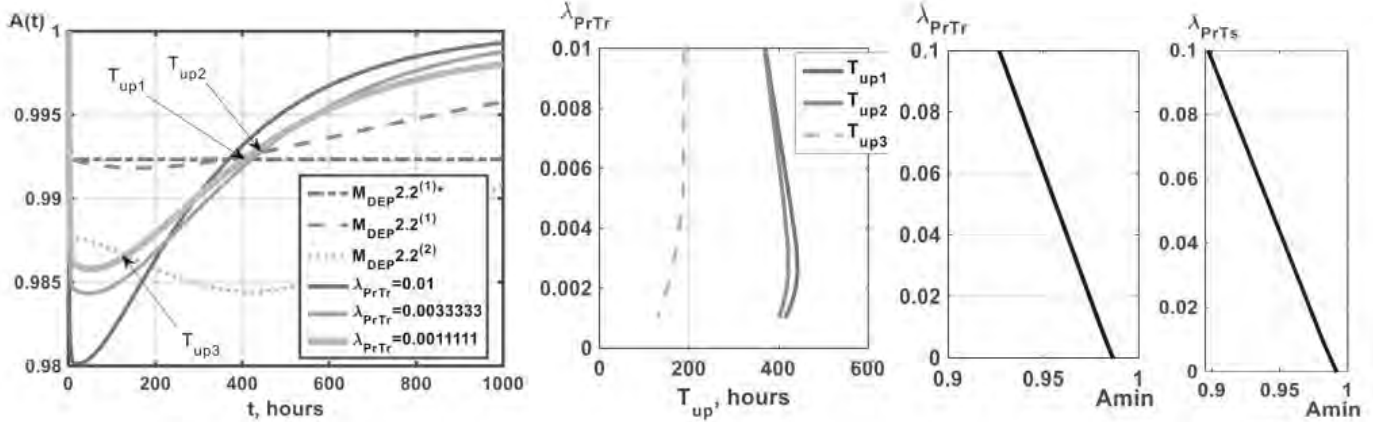

FIG. 13.9 Graphical dependences of availability function change on system operation time (A), partial dependability indicators Tup1, Tup2, and Tup3 (B), Amin with different values λ_{PrTr} (C), and λ_{PrTs} (D) obtained using the model $M_{DEP}2.2^{(3)}$.

Studies of $M_{DEP}2.2^{(i)}$ models have shown that patching procedures in conjunction with separate maintenance and cybersecurity procedures provide a more effective level of BAS availability in comparison with carrying out procedures with general maintenance. The increase in efficiency is to gain an increase in availability at the initial stage of operation by $1.5e-3$ with approximately the same time interval of bringing the system to the planned level of availability by 0.999 in 1600 h.

Further using the obtained results, development and research of multifragment models of smart building ICS should be carried out to create a single service to automate the process of assessing the BAS dependability, integration of developments with known packages for decision-making at all stages of ICS life cycle and their components that operate in changing requirements, environmental parameters, and unspecified failures including failures caused by attacks on vulnerabilities of IoT communication protocols [30].

References

[1] M. Froufe, C. Chinelli, A. Guedes, A. Haddad, A. Hammad, C. Soares, Smart buildings: systems and drivers, Buildings 10 (2020) 153, https://doi.org/10.3390/buildings10090153.

[2] D. Nicol, W. Sanders, K. Trivedi, Model-based evaluation: from dependability to security, IEEE Trans. Dependable Secure Comput. 1 (2004) 48–65, https://doi.org/10.1109/tdsc.2004.11.

[3] M. Alam, M. Reaz, M. Ali, A review of smart homes—past, present, and future, IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. 42 (2012) 1190–1203, https://doi.org/10.1109/tsmcc.2012.2189204.

[4] S. Budijono, J. Andrianto, M.A.N. Noor, Design and implementation of modular home security system with short messaging system, EPJ Web Conf. 68 (2014), https://doi.org/10.1051/epjconf/20146800025, 00025.

[5] A. Avizienis, J. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Trans. Dependable Secure Comput. 1 (2004) 11–33, https://doi.org/10.1109/tdsc.2004.2.

[6] International Electrotechnical Commission, IEC 60050–192. International Electrotechnical Vocabulary—Part 192: Dependability, 2015. https://webstore.iec.ch/publication/21886.

[7] O. Drozd, V. Kharchenko, A. Rucinski, T. Kochanski, R. Garbos, D. Maevsky, Development of models in resilient computing, in: 10th International Conference on Dependable Systems, Services and Technologies (DESSERT), 2019, pp. 1–6, https://doi.org/10.1109/DESSERT.2019.8770035.

[8] A. Gorbenko, V. Kharchenko, P. Popov, A. Romanovsky, Dependable composite web services with components upgraded online, in: Architecting Dependable Systems III, 2005, pp. 92–121, https://doi.org/10.1007/11556169_5.

[9] International Organization for Standardization, ISO 16484-1:2010. Building Automation and Control Systems (BACS)—Part 1: Project Specification And Implementation, 2010. https://www.iso.org/standard/37300.html.

[10] V. Kharchenko, Y. Ponochovnyi, A. Abdulmunem, A. Andrashov, Availability models and maintenance strategies for smart building automation systems considering attacks on component vulnerabilities, in: Advances in Dependability Engineering of Complex Systems, 2017, pp. 186–195, https://doi.org/10.1007/978-3-319-59415-6_18.

[11] V. Kharchenko, Y. Ponochovnyi, A.-S.M.Q. Abdulmunem, A. Ivasiuk, O. Ivanchenko, Model of information and control systems in smart buildings with separate maintenance by reliability and security, in: 14th Int. Conf. on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer, Kyiv, Ukraine, May 14–17, 2018, pp. 583–595.

[12] A. Gorbenko, V. Kharchenko, O. Tarasyuk, Y. Chen, A. Romanovsky, The threat of uncertainty in service-oriented architecture, in: SERENE 2008, RISE/EFTS Joint International Workshop on Software Engineering for REsilient SystEms, Newcastle Upon Tyne, UK, November 17–19, 2008, https://doi.org/10.1145/1479772.1479781.

[13] M. Bhuiyan, S. Kuo, D. Lyons, Z. Shao, Dependability in cyber-physical systems and applications, ACM Trans. Cyber-Phys. Syst. 3 (2019) 1–4, https://doi.org/10.1145/3271432.

[14] K. Trivedi, A. Bobbio, Reliability and Availability Engineering, Modeling, Analysis, and Applications, Cambridge University Press, 2017, https://doi.org/10.1017/9781316163047.

[15] S. Lysenko, V. Kharchenko, K. Bobrovnikova, R. Shchuka, Computer systems resilience in the presence of cyber threats: taxonomy and ontology, Radioelectron. Comput. Syst. 1 (2020) 17–28, https://doi.org/10.32620/reks.2020.1.02.

[16] A. Gorbenko, A. Romanovsky, O. Tarasyuk, Fault tolerant internet computing: benchmarking and modelling trade-offs between availability, latency and consistency, J. Netw. Comput. Appl. 146 (2019) 102412, https://doi.org/10.1016/j.jnca.2019.102412.

[17] A. Rucinski, I. Kovalev, M. Drozd, O. Drozd, V. Antoniuk, Y. Sulima, Development of computer system components in critical applications: problems, their origins and solutions, Herald Adv. Inf. Technol. 3 (2020) 252–262, https://doi.org/10.15276/hait.04.2020.4.

[18] E. Zio, The Monte Carlo Simulation Method for System Reliability and Risk Analysis, Springer Series in Reliability Engineering, 2013, https://doi.org/10.1007/978-1-4471-4588-2.

[19] R. Pietrantuono, P. Popov, S. Russo, Reliability assessment of service-based software under operational profile uncertainty, Reliab. Eng. Syst. Saf. 204 (2020) 107193, https://doi.org/10.1016/j.ress.2020.107193.

[20] K. Trivedi, D. Kim, R. Ghosh, Resilience in computer systems and networks, in: Proceedings of the 2009 International Conference on Computer-Aided Design—ICCAD'09, 2009, https://doi.org/10.1145/1687399.1687415.

[21] E. Zaitseva, V. Levashenko, J. Rabcan, M. Kvassay, P. Rusnak, Reliability evaluation of multi-state system based on incompletely specified data and structure function, in: 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2019, https://doi.org/10.1109/idaacs.2019.8924454.

[22] J. Batalla, F. Gonciarz, Deployment of smart home management system at the edge: mechanisms and protocols, Neural Comput. Applic. 31 (2018) 1301–1315, https://doi.org/10.1007/s00521-018-3545-7.

[23] K. Milanovic, Smart home security: how safe is your data? [opinion], IEEE Technol. Soc. Mag. 39 (2020) 26–29, https://doi.org/10.1109/mts.2020.2967490.

[24] S. Zheng, N. Apthorpe, M. Chetty, N. Feamster, User perceptions of smart home IoT privacy, Proc. ACM Hum.-Comput. Interact. 2 (2018) 1–20, https://doi.org/10.1145/3274469.

[25] Y. Ponochovnyi, V. Kharchenko, Dependability assurance methodology of information and control systems using multipurpose service strategies, Radioelectron. Comput. Syst. 3 (2020) 43–58, https://doi.org/10.32620/reks.2020.3.05.

[26] J. Neumann, C.E. Shannon, J. McCarthy, Probabilistic logics and the synthesis of reliable organisms from unreliable components, in: Automata Studies, Princeton Univ. Press, 1956, pp. 43–98.

[27] J. Henke, Dependable software for undependable hardware, in: 7th IEEE International Symposium on Industrial Embedded Systems (SIES'12), 2012, p. 1, https://doi.org/10.1109/SIES.2012.6356614.

[28] S. Iglin, grPlot—A Function for Drawing Graphs and Digraphs Using MATLAB, 2020. http://www.iglin.epizy.com/All/GrMatlab/grPlot.html.

[29] M. Ge, H. Kim, D. Kim, Evaluating security and availability of multiple redundancy designs when applying security patches, in: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2017, https://doi.org/10.1109/dsn-w.2017.37.

[30] M. Kolisnyk, Vulnerability analysis and method of selection of communication protocols for information transfer in Internet of Things systems, Radioelectron. Comput. Syst. 1 (2021) 133–149, https://doi.org/10.32620/reks.2021.1.12.