ISSN 2658-5014 (Print)
2021. Т. 3, № 2 (6). С. 5–11
СИИТ
СИСТЕМНАЯ ИНЖЕНЕРИЯ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
ISSN 2686-7044 (Online)
http://siit.ugatu.su

УДК 004.021

NESTING ALGORITHM FOR DUAL-GRAPH ERROR PROPAGATION MODELS

F. ZHAO [1], A. MOROZOV [2], N. I. YUSUPOVA [3], K. JANSCHEK [4]

[1]fusheng.zhao@mailbox.tu-dresden.de, [2]andrey.morozov@tu-dresden.de, [3]yussupova@ugatu.ac.ru, [4]klaus.janschek@tu-dresden.de

[1] CAE Engineer Mechatronics, ESI ITI GmbH, Dresden, Germany
[2] Institute of Industrial Automation and Software Engineering, Faculty 05, University of Stuttgart, Germany
[3] Ufa State Aviation Technical University Ufa, Russia
[4] Technical University of Dresden, Dresden, Germany

Поступила в редакцию 14 мая 2021 г.

Abstract. Probabilistic error propagation analysis is a valuable part of dependable systems development process. It helps to estimate consequences of faults of particular system components to overall system reliability, which is required by industrial reliability methods such as FTA and FMEA. The dual-graph error propagation model is a recently presented stochastic model that captures system properties relevant for the error propagation analysis. It allows automatic generation of discrete time Markov chain models for quantitative estimation of system reliability e.g. a mean number of errors in critical system outputs. Likewise all Markov-based analytical approaches, our method is prone to the state space explosion problem: The number of states of the Markov model grows exponentially with the number of system components. This article introduces a new algorithm for automatic nesting of huge error propagation models that allows generation of a set of interconnected small Markov-chain models for each hierarchy level, instead of one large and incomputable Markov chain model.

Key word: model-based system; baseline models; elements, data storages, directed control; flow arcs; analytical software tool.

INTRODUCTIONMODEL-BASED SYSTEMS

Model-based system development approaches are popular nowadays in automotive, avionic, and aerospace industrial domains. UML/SysML [1, 2], AADL [3], and MATLAB Simulink/Stateflow [4, 5] are commonly used baseline models that allow generation of fully or semi-automatic toolchains for system design, implementation, integration, verification, and deployment. The model-based development approach also gives opportunities for a wide range of model-based system analysis techniques that can be applied in early stages of system development, including the error propagation analysis.1.2. ERROR PROPAGATION ANALYSIS.

Our research group for model-based system analysis is focused on stochastic error propagation analysis of heterogeneous systems developed with common baseline models like

MATLAB Simulink/Stateflow and UML/SysML. Recently, we have introduced, [6–9] a dual-graph error propagation model (DEPM) that captures key system design aspects relevant to error propagation processes. The DEPM helps to estimate the likelihood of error propagation to hazardous system parts and quantify the negative impact of a fault in a particular component on the overall system reliability. The DEPM consists of five sets: elements, data storages, directed control flow arcs extended with control flow decision probabilities, directed data flow arcs, and conditions of the elements. A simple DEPM is shown in Fig.1, a).

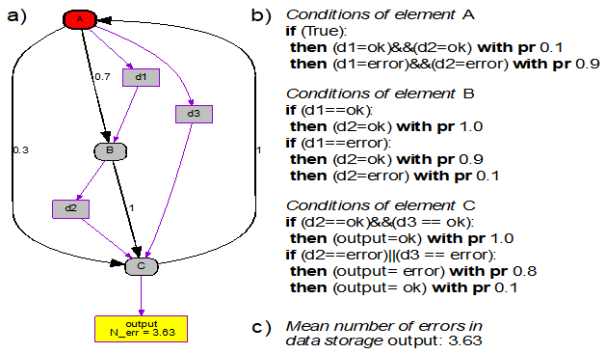


Fig. 1. An example of an error propagation model: *a – a graphical representation, b – probabilistic conditions of the elements that describe their reliability properties, c – data storage output. Image from* [10]

Elements *A*, *B*, and *C* represent executable parts of the system. Each element has data inputs and outputs. Data storages *d1*, *d2*, *d3*, and *output* represent variables, which can be read or written by the elements. During execution errors can occur on its outputs. The incoming errors can propagate from inputs to outputs depending on reliability properties of an element, defined with probabilistic conditions, (see Fig. 3, *b*).

The control flow arcs (black lines) connect the elements. Each control flow arc is weighted with a transitions probability: after the execution of *A*, *B* will be executed with probability 0.7 and *C* with probability 0.3. The data flow arcs (purple lines) describe data transfer between the elements. Data flow arcs connect elements with data storages. The data flow arcs are considered to be the paths of data error propagation.

ANALYTICAL SOFTWARE TOOL ERRORPRO

ErrorPro [10] is our analytical software tool, which is based on the DEPM concept. This tool allows the creation of DEPMs and automatic computation of numerical reliability properties of a system using internal Markov chain models. For instance, the mean number of errors in the data storage output during 100 steps (execution of one element is one step) is equal to 3.63 as it is shown in Fig. 3, *c*). The PRISM model checker [11] is used as a computational backend for the Markov models. A DEPM model can be stored in a special XML-based format.

Our transformation methods allow the generation of DEPMs automatically from MATLAB Simulink models [12] and UML Activity diagrams [13]. Currently, we are working on the generation of DEPMs for plain C code using the LLVM technology. However, our method, similarly to the majority of Markov-based analytical approaches, is prone to the state space explosion problem: The number of states of the Markov model grows exponentially with the growth of the number of system components. This article introduces a new algorithm for automatic nesting of large and flat error propagation models. This allows generation of a set of interconnected small Markov-chain models for each hierarchy level instead of one large and incomputable Markov chain model.

AUTOMATIC NESTINGNESTED ERROR PROPAGATION MODELS

ErrorPro supports hierarchical DEPMs with compound elements. Fig. 2 shows an example of a flat *(a)* and a nested *(b)* DEPMs. Both models are identical from the functional point of view. The compound element *subtop1_level2* contains internal elements *e1*, *e2*, and *e3* and

data *d2*, *d3*, and *d5*. The control flow transfers to the initial element of the sub model *e1* at the moment when the compound element is executed. The sub model has external data input *d1* and output *d4*.

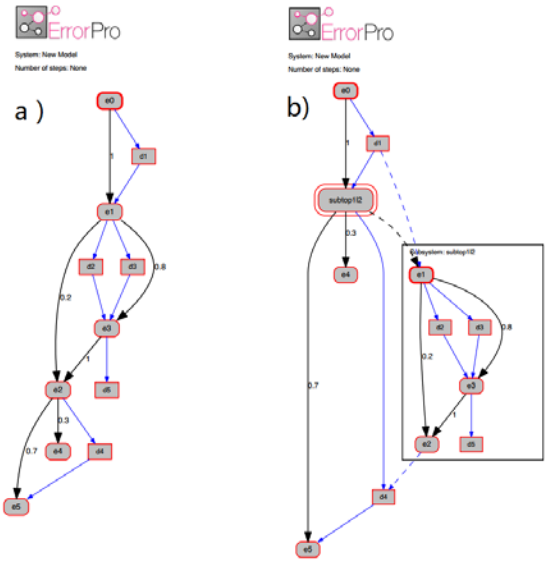


Fig. 2. An example of a flat *(a)* and nested *(b)* dual-graph error propagation models

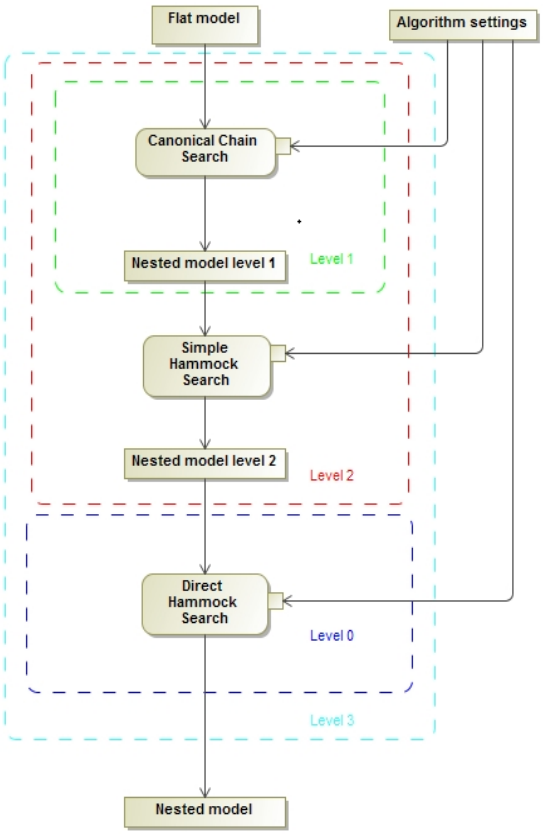


Fig. 3. UML activity diagram that describes the top-level architecture of the presented algorithm for automatic nesting of DEPM models

ARCHITECTURE OF THE NESTING ALGORITHM

The top-level structure of the presented algorithm is shown in Fig. 3 in a form of a UML activity diagram. *Flat model* and *Algorithm settings* are the inputs; *Nested model* is the output. The algorithm consists of three blocks: *Canonical Chain Search* (CCS), *Simple Hammock Search* (SHS) and *Direct Hammock Search* (DHS). The algorithm is also subdivided into four levels that can be used separately.

Level 0: The algorithm uses only DHS.

Level 1: The algorithm uses only CCS.

Level 2: The algorithm uses CCS and than SHS.

Level 3: The algorithm uses CCS, SHS, and then DHS.

The benchmarking results of the algorithm, described later in this article, will refer to these four levels.

DIRECT HAMMOCK SEARCH

A hammock is a region of a DEPM control flow graph (CFG) that has a single entry node and a single exit node [14]. The example is shown in Fig. 2. Elements *e1* (entry node), *e2, and e3 (exit node) form a hammock.*

The DHS is a greedy algorithm that finds hammocks in a DEPM. The DHS algorithm is parameterized with four parameters that define properties of desired submodels: min number of elements, max number of elements, min number of data, and max number of data. The DHS runs depth-first searches on the CFG subsequently for each element, starting with the initial element. The depth-first search always selects an element with the minimum number of

incoming control flow arcs among the control flow successors of the current element. In each step of the depth-first search, we add visited elements and data, connected to these elements, into a temporary submodel. If the temporary submodel is a correct hammock and if it satisfies the defined parameters, then the algorithm substitutes this region in the top-level model with a compound element that contains the generated submodel. If the temporary submodel has more elements or data than defined, then the DHS starts a new search, starting with the next element of the top-level DEPM according to the sequence of an internal array of the elements.

CANONICAL CHAIN SEARCH

A canonical chain contains two elements connected with a control flow arc. The first element has outgoing control flow arcs only to the second element or to itself. The second element has incoming control flow arcs only from the first element or from itself. The DEPM in Fig. 2 contains only one canonical chain (*e0, e1*).

The CCS algorithm itself is based on the depth-first search algorithm that finds and stores information about all canonical chains in the CFG of the flat DEPM. After that, it generates submodels, which consist of sequences of elements of the canonical chains and data slots, connected with these elements, that satisfy the given parameters. According to our analysis of a system described in [12], 88% of all software elements have only one outgoing control flow arc. This makes the CCS an effective and fast per filter before the SHS and DHS.

SIMPLE HAMMOCK SEARCH

A simple hammock is a *hammock* that contains only one element and its control flow successors. The hammock in Fig. 2 is a simple hammock: it consists only of the element *e1* and its control flow successors *e2* and *e3*. The SHS algorithm is similar to CCS. It is based on the breadth-first search over the CFG nodes. It finds all simple hammocks that satisfy the algorithm parameters and generates submodels. This SHS is rather effective after the application of the CCS because all the chains of the original flat model are nested and we can perform the fast SHS before the more complex and hence slow DHS.

BENCHMARKINGDEPM GENERATOR

We have implemented an automatic generator of random large DEPM models in order to test and benchmark the introduced algorithm. The generator has three parameters: *number of elements*, *number of data*, and a parameter *p* that defines the control flow structure of the DEPM. For instance, *p* = [0, 0.85, 0.13, 002] defines that 85% of the elements of the generated DEPM will have only one control flow output, 13% will have two control flow outputs, and 2% will have three control flow outputs. The generation process consists of the next steps:

Step1: Create the set of elements.

Step2: Create the set of data.

Step3: Connect all elements sequentially using control flow arcs.

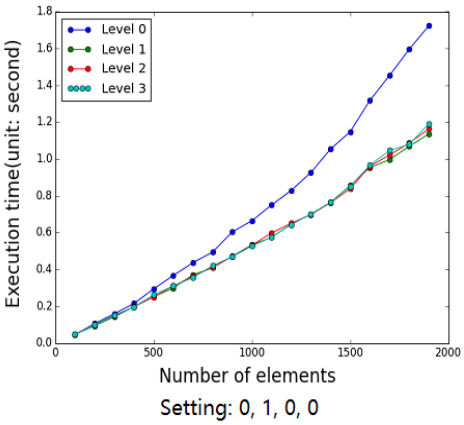
Step4: Randomly create additional control flow arcs according to the parameter *p* and specify equal control flow probabilities to all outgoing control flow arcs for all elements.

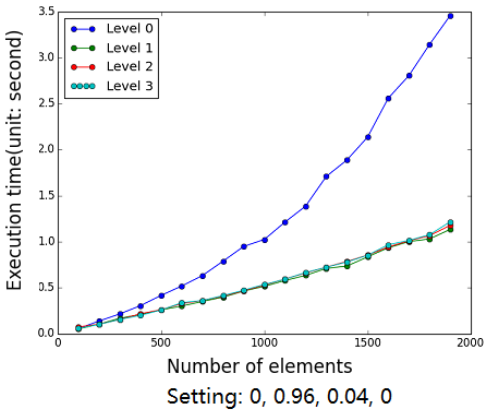
Step5: Randomly connect all the data slots with the elements with one incoming and one outgoing data flow arcs.

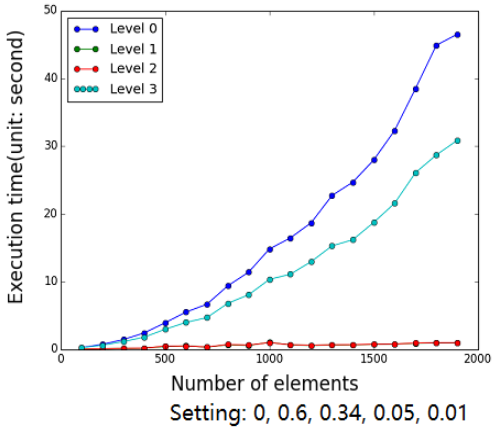
BENCHMARKING RESULTS

The speed and performance of the algorithm have been evaluated using the randomly generated sets of DEPM models. The goal of the first group of experiments is to measure the execution time for different types and sizes of the DEPMs. Four plots in the Fig. 4

demonstrate the numerical results. Each plot has been created for different settings of the parameter *p*. The number of data slots is equal to the number of elements and vary from 100 to 2000 with the step equals to 100. We have generated 100 random DEPMs of each size and compute the mean execution time (in seconds) for each of four levels of the nested algorithm.







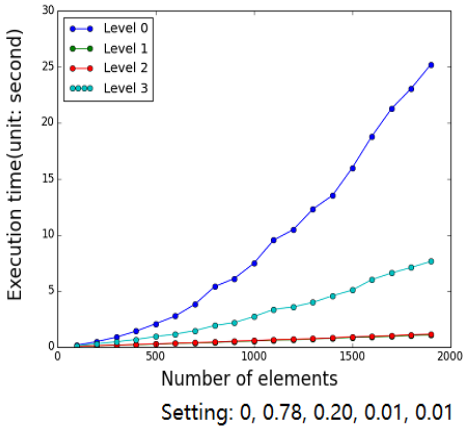


Fig. 4. Execution time of the four levels of the nesting algorithm for different types and sizes of DEPMs

The nested algorithm has been run with the next values of the parameters: *min number of elements = 2*, *max number of elements = 20*, *min number of data = 1*, and *max number of data* = 7.

The plots in Fig. 4 show that the execution time of the DHS (level 0) considerably grows with the control flow complexity. The level 3 (CCS + SHS + DHS) shows much better results. The results of the level 1 (CCS) and level 2 (CCS + SHS) are almost similar that defines that the SHS works very fast after the application of the CCS. The second group of the experiments has been carried out in order to estimate the performance of the algorithm. The main goal of the algorithm is to create a nested DEPM in such way that it will be converted into a set of small and computable Markov chain models. The maximum cumulative number of states of the Markov models (for the worst case) can be computed according to the next formula:

$$P = \prod_{i=1}^{k} N_i^E 2^{N_i^D},$$

where N_i^E is the number of elements and N_i^D is the number of data slots in the i^{th} submodel. We used *P* as the performance metric in the second group of the experiments. Fig. 5 demonstrates the comparison of the performance for DHS only (level 0) and CCS + SHS + DHS (level 3) according to the defined performance metric. The parameter *p* is constant and equals to [0, 0.96, 0.04, 0.0]. The number of data slots is equal to the number of elements and vary from 100 to 2000 with the step length 100. The

number of repetitions for each model size is also 100. This plot shows that even the application of only DHS results in linear growth of the cumulative number of states of the Markov chin models. The entire algorithm gives even better results. This allows solving the state space explosion problem of our approach to the stochastic error propagation analysis.

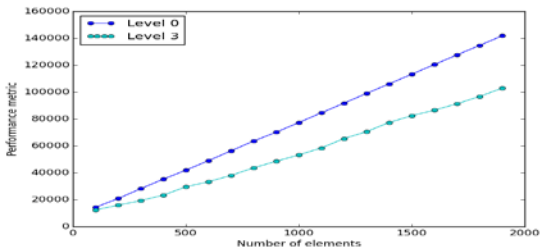


Fig. 5. Performance comparison of the DHS (level 0) and CCS + SHS + DHS (level 3) for different sizes of DEPM models

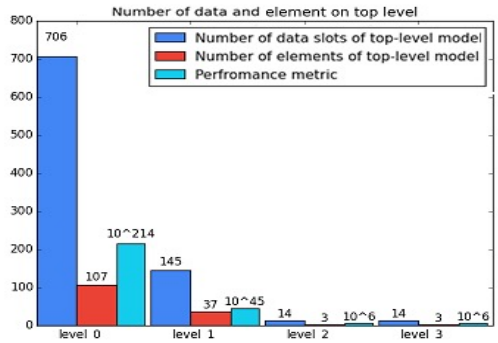


Fig. 6. Comparison of the performance of all four levels of the nesting algorithm

Fig. 6 demonstrates another peace of results. This time for the constant numbers of elements (1000) and data slots (4000), and p = [0, 0.96, 0.04, 0.0]. The plot basically shows that the performance of the DHS drops dramatically if the number of data is greater than the number of the elements. In the particular case, the top-level DEPM has 107 element and 706 data slots that will result in a huge and incomputable DEPM. However, the application of the entire algorithm (level 3) results in very good nesting: only 14 data slots and three elements in the top-level DEPM.

CONCLUSIONS

The new nesting algorithm for dual-graph error propagation models (DEPM) has been introduced in this paper. The algorithm copes with the state space explosion problem of the underlying Markov chain models. The algorithm consists of three separate blocks: *Canonical Chain Search* (CCS), *Simple Hammock Search* (SHS) and *Direct Hammock Search* (DHS). All three blocks can be used separately. The algorithm has been embedded into the new version of our analytical software tool ErrorPro. Also, the generator of random DEPMs has been implemented, which allowed us to perform an extensive benchmarking of the performance properties of the introduced nesting algorithm reported in this paper.

REFERENCES

1. **OMG - Object Management Group**, Unified Modeling Language (UML). Specification v2.5, June 2015. **[** *Unified Modeling Language (UML)*, OMG – Object Management Group, specification v2.5, June 2015. **]**

2. **OMG - Object Management Group**, Systems Modeling Language (SysML). Specification v.1.4, September 2015. **[** *Systems Modeling Language (SysML)*, OMG - Object Management Group, Specification v.1.4, September 2015. **]**

3. **Feiler P. H. and Gluch D. P.** Model-Based Engineering with AADL: An Introduction to the SAE. **[** P. H. Feiler and D. P. Gluch Model-Based Engineering with AADL: An Introduction to the SAE. 2015. **]**

4. **Architecture Analysis & Design Language** (1st ed.). Addison-Wesley Professional, 2012. **[Architecture Analysis & Design Language** (1st ed.). Addison-Wesley Professional, 2012.

5. **Mathworks,** "Matlab & Simulink: Simulink User's Guide R2015b". Retrieve, 2015. **[** Mathworks, "Matlab & Simulink: Simulink User's Guide R2015b". Retrieve, 2015. **]**

6. **Morozov A. and Janschek K.** Case study results for probabilistic error propagation analysis of a mechatronic system // Tagungsband Fachtagung Mechatronik. 2013, Pp. 229-234.

7. **Morozov A. and Janschek K.** Probabilistic error propagation model for mechatronic systems // Mechatronics. 2014. 24 (8). Pp.1189 - 1202.

8. **Janschek K. and Morozov A**. Dependability aspects of model-based systems design for mechatronic systems // Mechatronics (ICM), 2015 IEEE International Conference. Nagoya, Japan. 2015. Pp. 15-22.

9. **Morozov A., Tuk R., Janschek K**. ErrorPro: Software tool for stochastic error propagation analysis // 1st International Workshop on Resiliency in Embedded Electronic Systems, Amsterdam, The Netherlands. 2015. Pp. 59-60.

10. **Kwiatkowska M., Norman G., Parker D**. PRISM 4.0: Verification of probabilistic real-time systems / **G. Gopalakrishnan** and **S. Qadeer (**eds.) // Proceedings 23rd International Conference on Computer Aided Verification (CAV'11). 2021. Springer. Pp. 585-591. **[** M. Kwiatkowska, G. Norman, D. Parker, "Verification of probabilistic real-time systems", in *Proceedings 23rd International Conference on Computer Aided Verification (CAV'11)*. Springer. pp. 585-591, 2011. **]**

11. **Stochastic** Error Propagation Analysis of Model-driven Space Robotic Software Implemented in Simulink / A. Morozov, et al. // The proceedings to the Third Workshop on Model-driven Robot Software Engineering, Leipzig, Germany, 2016. **[** A. Morozov, *et al.* "Stochastic Error Propagation Analysis of Model-driven Space Robotic Software Implemented in Simulink", in *Proceedings to the Third Workshop on Model-driven Robot Software Engineering*", Leipzig,Germany,2016. **]**

12. **Automatic** Transformation of UML System Models for Model-based Error Propagation Analysis of Mechatronic Systems / K. Ding, et al. // The Proceeding of IFAC Mechatronics Conference, Loughborough, UK, 2016. **[** K. Ding, *et al,* "Automatic Transformation of UML System Models for Model-based Error Propagation Analysis of Mechatronic Systems", in *Proceeding of IFAC Mechatronics Conference*, Loughborough, UK, 2016. **]**

13. **Kasyanov V. N.** Distinguishing Hammocks in a Directed Graph // Soviet Math. Doklady. 1975. 16.5. Pp. 448-450.

ABOUT AUTHORS

ZHAO, Fusheng, CAE Engineer Mechatronics, ESI ITI GmbH, Dresden, Germany.

MOROZOV, Andrey, Junior Professor of Networked Automation, Institute of Industrial Automation and Software Engineering, Faculty 05, University of Stuttgart, Germany

YUSUPOVA, Nafisa, Professor, Dean of Computer Science and Robotics Faculty, Ufa State Aviation Technical University, Russia.

JANSCHEK, Kalus, Professor of Automation Engineering, Institute of Automation, Faculty of Electrical and Computer Engineering, Technische Universität Dresden, Germany.

МЕТАДАННЫЕ

Заголовок: Алгоритм вложенности для моделей распространения ошибок двойного графа.

Авторы: F. Zhao[1], A. Morozov[2], N. I. Yusupova[3], K. Janschek[4]

Принадлежность: [1] ESI ITI GmbH, Дрезден, Германия,

[2]Institute of Industrial Automation and Software Engineering, Faculty 05, University of Stuttgart, Germany, [3]Уфимский государственный авиационный технический университет, [4]Технический университет Дрездена Дрезден, Германия.

Почта: [1]fusheng.zhao@mailbox.tu-dresden.de, [2]andrey.morozov@tu-dresden.de, [3]yussupova@ugatu.ac.ru, [4]klaus.janschek@tu-dresden.de

Язык: English.

Источник: СИИТ (научный журнал Уфимского государственного авиационного технического университета), т. 3, № 2 (6), стр. 5–11, 2021. ISSN 2686-7044 (онлайн), ISSN 2658-5014 (печатный вариант).

Аннотация: Вероятностный анализ распространения ошибок – важная часть процесса разработки надежных систем. Это помогает оценить последствия отказов отдельных компонентов системы для общей надежности системы, что требуется для методов промышленной надежности, таких как FTA и FMEA. Модель распространения ошибок с двойным графом – это недавно представленная стохастическая модель, которая фиксирует свойства системы, относящиеся к анализу распространения ошибок. Это позволяет автоматически генерировать модели цепей Маркова с дискретным временем для количественной оценки надежности системы, например, среднее количество ошибок в критических выходных данных системы. Как и все аналитические подходы на основе Маркова, наш метод подвержен проблеме взрыва пространства состояний: число состояний модели Маркова растет экспоненциально с числом компонентов системы. В этой статье представлен новый алгоритм автоматического вложения моделей распространения огромных ошибок, который позволяет генерировать набор взаимосвязанных небольших моделей цепей Маркова для каждого уровня иерархии вместо одной большой и невычислимой модели цепей Маркова.

Ключевые слова: модельная система; базовые модели; элементы, хранилища данных, направленное управление; дуги потока; аналитический программный инструмент.

Об авторах:

ЧЖАО, Фушэн, CAE инженер мехатроники, ESI ITI GmbH, Дрезден, Германия.

МОРОЗОВ Андрей, младший профессор кафедры сетевой автоматизации. Институт промышленной автоматизации и программной инженерии, факультет 05, Штутгартский университет, Германия.

ЮСУПОВА Нафиса, профессор, декан факультета компьютерных наук и робототехники Уфимского государственного авиационного технического университета, Россия.

ЯНШЕК, Калус, профессор кафедры автоматизации, Институт автоматики, факультет электротехники и вычислительной техники, Технический университет Дрездена, Германия.