Software design uses the set of tools for building and analyzing models which do not allow solving the complex problem of improving the quality of design solutions. This task includes not only the synthesis of the software model with parallelism, the detection and localization of errors for the correction of the model, but also the visual analysis of the model for the synthesis of new or corrected design solutions. The study object is the building and analyzing processes of software models with parallelism.

Techniques and a method's of synthesis and analysis of the software model with parallelism are proposed, which are based on the combined approach to simulation modelling of the systems with parallelism. The analysis of the software model with parallelism begins at the stage of creating and static analyzing component models. The proposed method provides dynamic analysis of component models and partial model in the process of assembling the complete software model.

Building of component models is carried out based on the rules for building PN-models, PN-templates and the principle of structural similarity, while their static properties are checked. The dynamic analysis of the component models of software is carried out using simulation modeling and the method of invariants. In process of the model assembling, the complete model is gradually formed by assembling it from the models of software components, and its dynamic properties are analyzed. In this case, the convolution method, the method of invariants, and simulation modelling are used.

Through in-depth dynamic analysis the presented method's provides an opportunity to check the static and dynamic properties of the studied models, which ensures an increase in the quality of project solutions at the software design stage. It can be used to reduce the cost of software development, as well as to analyze the developed software to improve the efficiency of support

Keywords: software with parallelism, Petri nets, critical properties, synthesis of software model, dynamic analysis

UDC 004:942

DOI: 10.15587/1729-4061.2023.283075

IMPROVING THE EFFICIENCY OF DYNAMIC ANALYSIS OF THE COMBINED SIMULATION MODEL FOR SOFTWARE WITH PARALLELISM

Oksana Suprunenko
Corresponding author
PhD, Associate Professor*
E-mail: ra-oks@vu.cdu.edu.ua
Borys Onyshchenko
PhD, Associate Professor*
Julia Grebenovich
Senior Lecturer*
*Department of Software for Automated Systems
The Bohdan Khmelnytsky National University of Cherkasy
Shevchenko blvd., 81, Cherkasy, Ukraine, 18031

Received date 03.04.2023
Accepted date 16.06.2023
Published date 30.06.2023

***How to Cite**: Suprunenko, O., Onyshchenko, B., Grebenovich, J. (2023). Improving the efficiency of dynamic analysis of the combined simulation model for software with parallelism. Eastern-European Journal of Enterprise Technologies, 3 (2 (123)), 26–34. doi: https://doi.org/10.15587/1729-4061.2023.283075*

1. Introduction

The high complexity of modern software requires a detailed analysis of project solutions, which calls for the development of software modeling approaches, methods, and tools [1]. In particular, the complexity of the software is due to the presence of explicit or hidden parallelism with numerous options for embedding or crossing parallel branches in the model structure. This can cause errors in software development, which makes the design, development, and maintenance of software systems more expensive. To ensure high performance indicators of software models, along with structural analysis at the stage of software design, it is necessary to conduct their dynamic analysis [2], which will reduce development time and reduce its costs.

To carry out these types of analysis, several models are used [3], which are built with the use of various means of describing models. Certain properties are analyzed on each of these models. The lack of a single model of the studied software causes difficulties, which are associated with the need to reconcile the results obtained on different models. With such an analysis, it is difficult to form a holistic view of the functioning of the software under the influence of various factors. Some models use classifiers built on the basis of statistical data to predict software defects [1]. The results of these studies are of an evaluative nature and do not allow obtaining specific data for correcting the models.

Conducting static and dynamic analysis on a single model is also relevant for already developed software, which requires analysis of implemented solutions to improve the effectiveness of its support. Such an analysis should be carried out based on the results of failures or non-standard functioning of software tools recorded during maintenance. The research approaches and tools should be adapted to the simulation of the implemented software, which involves a visual check of the model for compliance with the implemented code, and also allows automated analysis of its static and dynamic properties, determining the localization of defects. This will reduce software maintenance costs.

2. Literature review and problem statement

When designing software systems, various approaches and methods are offered, which are aimed at increasing the reliability of software operation. Thus, to reduce the complexity of the developed systems, aspect-oriented programming [1] is used, which also makes it possible to involve previously developed modules for their reuse. In this technological approach, modular functionality is separated from

end-to-end functionality of the developed software system. At the coding stage, evaluations of the metrics of linguistic means and topology complexity are used, a set of classifiers is used [1]. They make it possible to reduce the amount of code and improve functional decomposition. But this approach has only evaluative tools for project analysis, and also complicates the code and its analysis during testing [1]. The application of this approach covers the stages of coding and testing, which narrows the possibility of detecting hidden errors at the early stages of software design and determining the localization of these errors in the model. Therefore, the problem of synthesis of new or corrected project solutions cannot be solved using the considered approach.

To analyze project solutions at the software design stage, numerous models are used, designed to improve the quality of software product development. But several models are often built to represent different aspects of the software tool [3], or local models are formed that cover only certain characteristics of the software and do not provide means for visual analysis and correction of the model. In particular, great interest is attached to the algebra of processes, in which the technology of "block modeling" is used [4]. That makes it possible to determine the structure of the model and the nature of the interaction of its components at the upper level, and at the lower level to describe the algorithms for processing the workload by individual components of the model [4], which increases the adequacy of modeling. But for the dynamic and visual analysis of systems during their design, this tool is difficult since actions are modeled explicitly, and states are implicitly modeled [5]. The tasks of explicit representation of actions and states are solved when building models based on workflow networks (WF-networks) [5], but these networks are the most limited class of Petri nets. This does not allow them to be used for modeling software with parallelism. A more powerful class of models is used in [6], in which a marked system of transitions and a marked Kripke structure are used to build models from attribute-based code. This approach requires software code input, that is, it can be used to diagnose existing software and its components.

In [7], statistical models of the reliability of complex technical systems are built, which make it possible to determine the time dependences of the time of the studied model in a state of failure, but these models do not allow detecting areas associated with failure. Models are also being developed [8], which are used to assess the quality of current software characteristics. But the evaluation characteristics do not allow their use for error localization and correction of models of the evaluated software.

Instead, in the models of sustainable computing [9], the criteria of stability of critical applications are considered, as well as the search for hidden errors is carried out. But in these studies, only part of the external influences is considered – from natural disturbances, which limits the options for analyzing the functioning of software models.

Study [10] considers the construction of a software model based on component-oriented modeling but does not propose a holistic approach that would allow solving not only the problem of synthesis but also the problem of analyzing the software model. Paper [11] proposes a combination of model-based engineering and component-oriented software engineering. It is aimed at overcoming the problem of the complexity of modern software systems, but the authors use several models, which causes difficulties in their combination.

Thus, in the reviewed studies, solutions are proposed to increase the reliability of software functioning, when using which it is impossible to comprehensively solve the problem of detecting dynamic errors in software models and their localization. Also, only part of the models can be used for visual analysis of project solutions, in particular models based on WF-networks [5]. But they do not have the necessary power of visual representation of software models with parallelism. This leads to the need for a complex and large-scale analysis of the model to make corrections, which cannot be applied under the conditions of the implementation of modern software projects. Some modern approaches consider the construction of software models based on actual software development methods [10] or on the basis of already written code [6], but they cover only part of the tasks – the tasks of model synthesis [10] or their analysis [6]. In other works, it is proposed to combine model-oriented methods with well-known software development methods [11], but they are based on multiple models, which significantly complicates the solution of the tasks.

Therefore, there is a need to develop methods that allow studying the behavior of software based on a single description (model) in any possible variants of functioning, taking into account obvious and hidden errors in the software. It is also important to solve the problems of localization of model areas for the analysis of detected errors and construction of options for their elimination.

3. The aim and objectives of the study

The purpose of this study is to improve the efficiency of the dynamic analysis of the combined simulation model of the software with parallelism. To achieve the goal of the current research, the following tasks were solved:

– development of a technique of localization of critical properties of the model in case of full coverage of all elements of the model with component invariants;

– determination of the features of the analysis of parallel processes when building simulation models, taking into account the context of the problems for which the model is being developed;

– development of the method's for the synthesis and analysis of a combined simulation model of a software system with parallelism based on a combined approach of simulation modeling of systems with parallelism and developed tools.

4. The study materials and methods

The object of our research is the process of building and analyzing models of software systems with parallelism. The subject of the study is the dynamic properties of software systems with parallelism and their components in the process of assembling component models into a complete model.

Research hypothesis: when applying the synthesis and analysis procedure of the combined simulation model of software systems with parallelism, it is possible to identify obvious and hidden errors in the functioning of software components and to determine their localization.

It is proposed to use a combined approach to simulation modeling of systems with parallelism [12]. But its linear use does not always lead to the desired result. Therefore, it is proposed to devise a technique for solving uncertain situa-

tions in the problem of localization of critical model properties, a model correction technique in the analysis of partially connected parallel processes, and the method's in which the proposed techniques will be used.

In order to increase the effectiveness of the combined approach to simulation modeling of software with parallelism [12], it is necessary to take into account the peculiarities of the construction and dynamic analysis of simulation models of software components when developing the method's for the synthesis and analysis of a combined simulation model of a software system with parallelism. Also, the method's should take into account the peculiarities of the study of static and dynamic properties of components on artificially closed models. The procedure must be adapted to the limitations of the dimensionality of the invariant method [13]. Our theoretical research used:

– the theory of formal languages of Petri nets [14], which formed the basis of tools for forming a simulation model of a system with parallelism, including PN-patterns [15], which increase the efficiency of building models;

– the architecture of the combined approach to simulation modeling of systems with parallelism [12];

– the principle of structural similarity [12], which allows the researcher to understand the logic of errors and changes when correcting the model;

– a technique for synthesizing software component models [15], which makes it possible to observe static properties [16] when building a component structure from PN-elements and PN-patterns, to display on this structure the peculiarities of the functioning of the model using dynamic labels;

– dynamic properties of liveliness, limitation, repeatability, preservation, conflictlessness, and controllability [2] of software models and their components, compliance with which will ensure the expected functioning of the models;

– the method of invariants, which makes it possible to calculate T- and S-invariants and determine by their elements whether the dynamic properties of the studied model are observed;

– a technique for assembling a software model from models of its components and using the convolution method [17], which is presented and illustrated in [18].

Convoluting the sections of component models makes it possible to reduce their dimensionality due to linear sections and sections without critical properties. This provides an opportunity to apply the invariant method not only for dynamic analysis of software component models but also for in-depth dynamic analysis of partial and full software models.

To build and conduct an in-depth analysis of software models, a combined approach to simulation modeling of systems with parallelism is proposed [12], which involves building a model of a software tool based on a Petri net. The specified model has an unambiguous mathematical notation, which allows analyzing static and dynamic properties analytically. Also, the model is built according to the principle of structural similarity, which allows for visual analysis during the construction of the model and its simulation modeling. This is important [12] both when designing software components and when assembling them into a complete software product, as well as when analyzing the implemented software.

But with the linear application of the combined approach, it is not always possible to determine the localization of the critical elements of the model. Therefore, this paper proposes a procedure of synthesis and analysis of a combined simulation model of a software tool, which uses a technique that involves simulation modeling to detect "surface" errors and build a matrix description of the model. The matrix description allows for an in-depth analysis of the model based on the verification of dynamic properties: liveliness, safety (limitation), repeatability, preservation, conflictlessness, and controllability [2].

The proposed procedure is based on the architecture of a combined approach to simulation modeling of systems with parallelism [12], which involves the use of tools based on formal languages of L-type Petri nets with G-type language elements [14]. The choice of an L-type formal language is due to the fact that it is closed with respect to merging, parallel composition, and regular substitution, which is very important for software modeling problems.

Complementing the formal L-type language with elements of the G-type language allows working with intermediate markups when analyzing the model. Also, the use of G-type language elements provides an opportunity to analyze the local components of the model and the correctness of their merging into partial sub-models, which, upon further assembly, make up a complete model of the software tool.

5. The results of research on improving the efficiency of dynamic analysis of software system models with parallelism

5. 1. The technique for determining the localization of critical properties of the model in non-obvious cases

In the combined approach to simulation modeling of systems with parallelism, it is assumed to build a model based on Petri nets, which are described in the formal language of L-type Petri nets with G-type language elements [14]. To localize critical properties in systems models with parallelism, T- and S-invariants are used, as well as the rank of the incidence matrix W. The vertices associated with critical properties (violation of liveliness, limited, repeatability, preservation, conflictlessness) are indicated by the coverage of certain elements of T- and S-invariants only by zero values. Non-observance of the complete controllability property is indicated by a violation of the equation $\text{rang}(W)=\min(|T|,|S|)$ [2].

The rank of the incidence matrix W makes it possible, in addition to the dynamic properties, to determine the complete or incomplete controllability of the model, which in practice means the predictable or incompletely predictable functioning of the model. When determining the incomplete controllability of the model by the rank of the incidence matrix, in cases of complete coverage by the elements of the invariants of all components of the model [13], there is a need to determine the localization of logical errors in the elements of the model. The proposed technique makes it possible to determine the elements of the model that provoke incomplete controllability, guided by the markings specified in the S-invariants and simulation modeling of the studied model.

In the test model (Fig. 1) with the initial markup $\mu_0=[1\ 0\ 0\ 0]^T$, the properties of liveliness, limitation, and repeatability (according to the T-invariant), properties of preservation and conflictlessness (according to the S-invariant) are observed. But the rank of the incidence matrix $\text{rang}(W)<4$ [13] indicates the possibility of unpredictable functioning of the model.

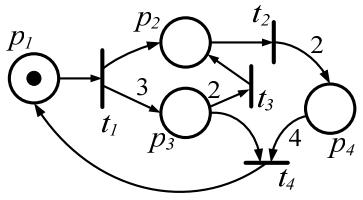

Fig. 1. Test model with hidden dead end [13]

To determine model vertices that cause unpredictable behavior, we use the following technique. When setting the initial markup indicated by the S-invariant $S=[5\ 2\ 1\ 1]^T$ and conducting a simulation experiment using sequences different from the counter of transitions vertices displayed by the T-invariant, we arrive at the markup $\mu_{53}=[0\ 0\ 0\ 31]^T$, which makes it possible to discover a hidden dead end.

According to this marking, the localization of the dead end is determined, which is obviously related to the vertices of the place p_4. In this vertex, there is an accumulation of labels, which leads to the appearance of a deadlock.

5. 2. Peculiarities of the analysis of parallel processes when building models of systems with parallelism

The general logic of the functioning of the modeled system should be supplemented by an analysis of the tasks solved by this system. Some features of the model cannot be detected when using a combined approach [12].

For example, in Fig. 2 shows a version of the functioning of the model, which reflects the execution of three tasks, which does not depend on the order of their execution. The model variant (Fig. 2) does not have critical properties that indicate the need to correct the model. But, when there are restrictions on the functioning of the modeled system, for example, it is necessary that some task *B* be performed only after the task *A* is performed, the analyzed model needs to be corrected. One of the variants of such a correction is shown in Fig. 3.

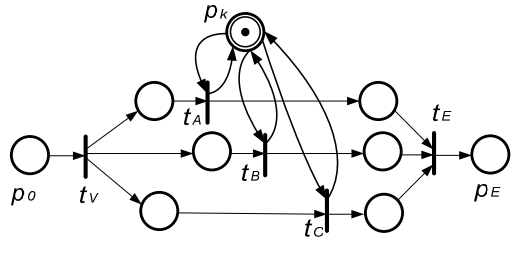

Fig. 2. Model with nondeterministic order of task processing

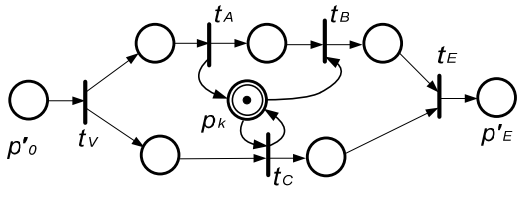

Fig. 3. A model with a partially ordered order of task processing

Thus, this analysis of the software system model should be carried out not only with the use of simulation modeling since formally the first model (Fig. 2) corresponds to dynamic characteristics. To find out the specifics of the task, it is necessary to analyze the business processes and their limitations reflected in the model. The example (Fig. 3) shows the process of taking into account the limitation in the order of tasks. To take it into account, it is necessary to display the partial dependence of the order of tasks in the model. In order to effectively solve such problems, a deeper analysis of the features of the model's functioning is required in terms of the implementation of the functioning of each of the business processes that are provided for by the conditions of the problem.

5. 3. Method's of synthesis and analysis of a combined software simulation model with parallelism

Our procedure of synthesis and analysis of the combined simulation model of software with parallelism is aimed at increasing the efficiency of dynamic error detection in models of software systems with parallelism and models of their components. It also makes it possible to determine the elements of the models with which the identified errors are associated. The method's devised has a number of stages (Fig. 4). At the initial stage of construction, the software model is represented as a "black box". This makes it possible to focus on the input parameters for the current simulation, as well as the output parameters and/or system properties that are expected to be obtained or need to be verified. Input parameters can be divided into initial parameters and control signals. Directly set initial parameters are used to run the model, this is the initial markup. Control signals or data from external systems, the arrival of which occurs under an asynchronous mode and is modeled by the corresponding parameters in the control vectors $\bar{X}_i$, are used to study external influences on the dynamic properties of the model. Therefore, the "default" mode is provided for the vertices of the model, where the control vectors can be configured. This mode at the initial stage of modeling makes it possible to build and debug a model without control influences or with deterministic (preset) control influences.

To form the structure of the model of the software tool with parallelism, the principle of "block modeling" is used, which is also used in process algebra [4]. This principle is the basis for component-oriented modeling [10], according to which an architectural solution is built. This decision reflects the division of the model into components and the determination of the general order of their interaction. At the lower level of the structural structure, the implementation model of each component is described in detail, which is close to the algorithmic description.

The proposed procedure of synthesis and analysis of the combined software simulation model allows for a step-by-step description [18] and automated analysis of such solutions [13] to check the dynamic properties of the model under study and its correction. This is the principle of block modeling, used in process algebra [4] and hierarchical Petri nets [16, 19]. It makes it possible to organize step-by-step detailing of the model in the modeling process, as well as the reverse process of convolution of the model components, which creates favorable conditions for the analysis of a partially assembled model by the invariant method.

The closed nature of formal languages of Petri nets to finite substitution [14] creates a favorable basis for the use of PN-patterns [15] and enlarged submodels in the synthesis of models. This speeds up the construction of the model and its visual inspection, for example, for the presence of "hanging" vertices. Solvability of analysis problems by the invariant method for restricted Petri nets without loops [2] and without inhibitory arcs [20, 21] allows for automated analysis of submodels built on their basis and partially connected models of software tools with parallelism.

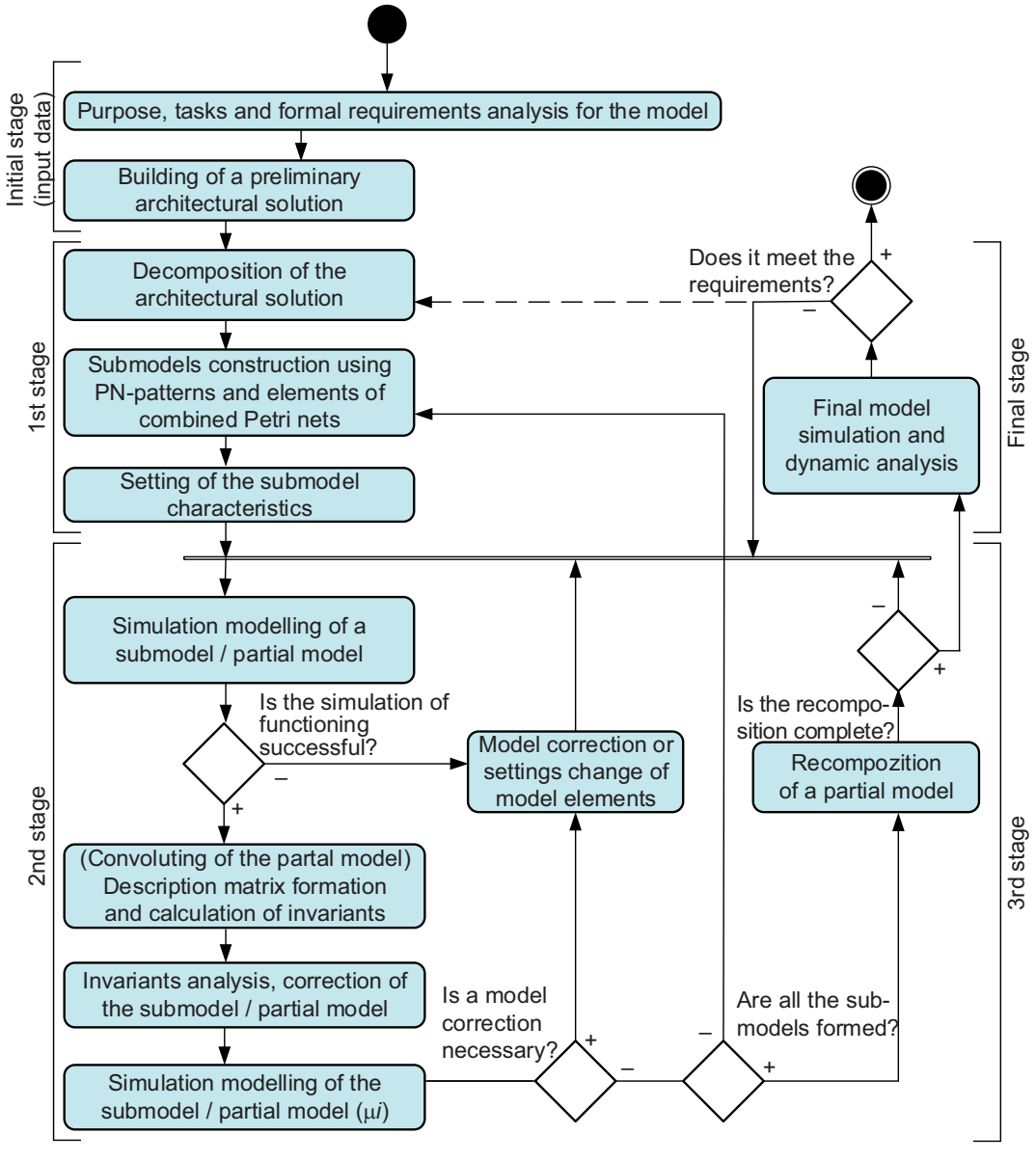

Fig. 4. Activity diagram of the method's of synthesis and analysis of the software tool model with parallelism

At the initial stage, input data is formed for the application of the proposed method's. The goals and objectives of model research, the formation of requirements for the model and the preliminary architectural solution are determined. Before starting the construction of the model, it is necessary to clearly formulate the purpose and tasks of the model research, analyze its features, and form an architectural solution:

0.1) conduct an analysis of the goal and tasks, determine the formalized requirements for the model, carry out their decomposition, the result of which will be a preliminary description of the components;

0.2) form a preliminary model of the architecture and/or select an appropriate architectural pattern in which to define the basic components and relationships between the components of the model.

At the first stage, a detailed representation of the model and debugging of its elements (Petri net elements and patterns) is built. When building submodels, it is necessary to step by step detail the solution, which involves:

1.1) step-by-step construction of submodels (model components) and, if necessary, their decomposition; when detailing submodels, it is worth determining the degree of decomposition, which depends on the complexity of the system, which is important for complex systems with parallelism; at this stage, we take into account the features of model refinement, considered in clause 5.2;

1.2) description of submodels of formed components using PN-patterns and elements of combined Petri nets; when building submodels, the static properties of Petri nets are checked, which contributes to the ongoing elimination of errors and inaccuracies;

1.3) correction the characteristics of the model elements and applied patterns, visual analysis of the logic displayed in the submodel and establishing the initial markup.

At the second stage, simulation modeling and invariant analysis of dynamic properties of submodels are carried out:

2.1) simulation modeling of the submodel with initial settings and initial marking; if necessary, making changes to the settings and markup of the submodel;

2.2) formation of a matrix description of the built submodel and calculation of invariants;

2.3) analysis of invariants, if necessary – making changes to the structure of the submodel/partial model and/or changing its initial settings;

2.4) repeated simulation modeling of the submodel and verification of its expected functioning under different work scenarios (using initial and intermediate markings)/simulation modeling of a partial model with different markings;

2.5) if there is a need to correct the submodel/partial model, return to the beginning of the second stage, i. e., step 2.1.

After the successful completion of the simulation modeling of the submodels, the system model is recomposed with parallelism [22, 23] and the intercomponent relationships are analyzed. To do this, we begin to assemble a model from the most complex submodel [24], to which, by sequential addition, we connect one adjacent submodel at a time and carry out their simulation modeling and analysis of invariants (Fig. 4).

If the architecture of the system is complex, then the recomposition of the model begins with the most complex submodel in each architectural component of the upper level [24]. At the next stage, we combine the formed and debugged components and, as a result, conduct a simulation of the entire model of the software tool. Thus, when all submodels are formed, assembling a software model with parallelism involves:

3.1) step-by-step recomposition of a partial model from submodels, taking into account inter-component connections in the system and external influences (in settings);

3.2) simulation modeling of assembled model components after each connection of a new submodel to a partial model;

3.3) in the event of conflicting situations during modeling (suspensions or clearly exceeded operating time of the model), preparation for the analysis of a partial model by its convolution to reduce the number of its elements p_i and t_i to 40–45 (the maximum dimension of the incidence matrix W) [13, 25] thanks to the convolution of the elements of the detailed representation of adjacent submodels [17] and the formation of a partial model for analysis;

3.4) formation of a matrix description of the constructed partial model and calculation of invariants;

3.5) analysis of invariants, if necessary, making changes to the structure of the partial model and/or changing its initial settings;

3.6) re-starting the simulation of the partial model and, in case of its successful completion, moving to the next stage of recomposition (3.1).

The final stage of model research is simulation modeling of it in a fully assembled form. During simulation, it is possible to control the main scenarios of use of the developed software model and the liveliness of its elements in these scenarios. A deeper analysis of the model is carried out on the basis of a matrix representation of its final recomposition. T- and S-invariants are calculated using the method of invariants, which determine the properties of reachability, repeatability, limitation, preservation, and conflictlessness. We also calculate the rank of the model's incidence matrix, which determines the controllability property. This property indicates the possibility of sequentially changing the markings from the initial one to the one determined in accordance with the researched tasks [26], and the subsequent correct (predictable) continuation or completion of the model.

This procedure is expected to be used under conditions where all elements of the invariants are covered by non-zero values, and the rank of the incidence matrix is $rang(W)<\min(|T|,|P|)$. In this case, we determine the difference between $\min(|T|,|P|)$ and r $rang(W)$, if $(rang(W) - \min(|T|,|P|)=1$, we apply the technique of determining critical properties described in clause 5.1. If the specified method indicates the vertices of the artificial closure of the model (for the calculation of invariants), then it is necessary to continue the assembly of the partial model. This is due to the peculiarities of the artificial closure of the partial model [18], which can cause a slight decrease in $rang(W)$.

Using the example of the software model of the microservice of video calls from work [18], the static and dynamic analysis of the initial model is illustrated, which was divided into three submodels, as well as the process of gradual assembly of the model with verification of the dynamic properties of the partial and final models.

The initial microservice model had a dimensionality of $|T|=31, |P|=24$. During its construction and initial simulation modeling, the static properties of the model were checked. According to the results of the dynamic analysis, the vertices associated with the violation of the dynamic properties were detected. In two peaks of the transitions, a violation of the liveliness property is observed; in the two vertices of places – violation of the preservation property. For further analysis, the model was divided into three component models (submodels). The first submodel reflected the processes of preparing for a video communication session, the second – the processes of processing erroneous user requests, the third – the processes of making a video call.

During the analysis of the first submodel, vertices associated with the violation of the liveliness property were found. According to the results of re-analysis of the corrected first submodel, compliance with the dynamic properties of liveliness, repeatability, limitation, and preservation was confirmed.

In the second submodel, duplication of function was found. After eliminating the detected errors, the number of its elements decreased from 26 ($|T|=16, |P|=10$) to 22 ($|T|=14, |P|=8$), i.e., by 15 %. The invariant analysis of the corrected second submodel confirmed its compliance with the dynamic properties of liveliness, repeatability, limitation, and preservation.

The third submodel in the invariant analysis showed compliance with all dynamic properties. In the process of assembling the model from submodels, certain areas of the submodels were convoluted, which ultimately led to a reduction in the dimensions of the microservice model from 33 vertices to 55, i. e., by a third. It also corresponds to the properties of liveliness, repeatability, limitation, preservation and full controllability, which increases the quality of project solutions.

During the study of submodels, violations of the rank indicators of the incidence matrix were observed, that is, $rang(W)<\min(|T|,|P|)$, which was caused by the artificial closing of submodels for the purpose of calculating their T- and S-invariants. When assembling a partial model from submodels, the $rang(W)$ values approached the values of $\min(|T|,|P|)$ to which they should correspond. When analyzing a complete microservice model $rang(W)=\min(|T|,|P|)$, which corresponds to full controllability of the model. The solution obtained on the model [18] can be used to increase the reliability of the functioning of the studied software implementation of the microservice of video calls.

6. Discussion of the research results of synthesis and analysis of the combined simulation model of software systems with parallelism

When building and analyzing models of systems with parallelism, there are cases in which the application of a combined approach to simulation modeling of systems with parallelism [14] should be carried out with the involvement of a technique of solving the problem of error localization in uncertain cases. This happens when all elements of the model are covered by non-zero elements of T- and S-invariants, but the rank of the incidence matrix is less than the minimum power of one of the sets of vertices of the Petri net [2]. Such a case indicates the incomplete controllability of the built model, while it is impossible to clearly determine which vertices of the model lead to its incomplete controllability based on the invariants. To solve this problem, this paper proposes a technique that uses simulation modeling with a markup that corresponds to one of the S-invariants, but violates the sequences displayed in the T-invariant, as shown in Fig. 1. The final marking obtained during simulation modeling indicates the critical vertices.

It is also important to take into account partial dependences between processes and their permanent or temporary nature when building or correcting a system model with parallelism (within the proposed approach [12]). Therefore, in the process of building and preliminary analysis of the software model, it is necessary to take into account the context of the tasks it solves because it affects the way the model is built, as shown in Fig. 2, 3. Such an analysis is complexly formalized and is not performed under an automated mode, but the obtained solutions can be checked using a combined approach [14].

To combine the combined approach to simulation modeling of systems with parallelism and techniques of error localization in uncertain cases and construction (correctment) of the researched model, a procedure of synthesis and analysis of the combined simulation model of software with parallelism has been developed. Important when using this procedure is the formation and analysis of a simulation model on a single representation, which is displayed by a Petri net and uniquely described mathematically [12]. This simplifies the analysis of a partial and final model of a software tool [18] in comparison with procedures that use several models with different means of description to check certain groups of properties [3].

When assembling a software model from submodels of its components, the dimensionality of the resulting partial model increases rapidly. This complicates the application of the invariant method, as demonstrated by the example in [18]. To reduce the dimensionality of the model, the convolution method [17] is used, which makes it possible to leave essential elements related to parallelism in the model.

As a result of the complex solution of the task of researching the static and dynamic properties of the combined simulation model of software systems with parallelism, a procedure of synthesis and analysis of the model of the software tool with parallelism was developed (Fig. 4). The proposed procedure is based on a combined approach, which allows combining the advantages of simulation modeling and analytical research to identify explicit and hidden dynamic errors. It also uses additional proposed techniques of synthesis and analysis of software models, which make it possible to expand the options for applying the combined approach to simulation modeling of systems with parallelism.

The limitations of the proposed method's primarily include the dimensionality of the studied models, which, as in the combined approach to simulation modeling of systems with parallelism, can be no more than 40–45 elements of the largest set of sets of vertices P or T ($\max(|T|,|P|)$ [12].

Another important limiting aspect of the analysis of the model with parallelism is taking into account the peculiarities of the problems solved by the model. Thus, when analyzing the main conflict situations, the model should pay attention not only to the general logic of the simulated system but also to the specifics of the tasks solved by this system, for example, as shown in Fig. 2, 3. Such features cannot be detected automatically by the presented procedure.

The disadvantages of the proposed procedure include the sometimes rather complex construction of submodels. The construction process is partially simplified by the use of PN-patterns and models of reusable structures that have successfully passed the analysis of dynamic properties. It is also difficult in some non-linear cases to use the convolution method.

It is promising in the continuation of this study to simplify the construction of submodels, to expand the toolset for building models with elements of time [27] and color [28] Petri nets. Also among promising tasks is the representation of templated solutions for correcting models after detection of dynamic errors, automation of model construction during the analysis of implemented software components.

The proposed method's of synthesis and analysis of a combined simulation model of a software system with parallelism may be of interest to specialists such as Analyst, Solution Architect, Application Architect, Software Engineer, System Designer, Backend Developer, etc. The method's can also be used to diagnose implemented software; it can be used by Performance Engineers and Technical Support Engineers.

7. Conclusions

1. In order to identify the localization of errors in the software model in the case of full coverage of all model elements by invariants, the use of simulation modeling with markings indicated by the obtained S-invariants is implied. During the gradual assembly of components into a complete model, the use of both visual assessment and simulation modeling of Petri nets, and analysis based on the invariant method is implied. This makes it possible to check the proposed solutions, which ensure an increase in the quality of the obtained model of the software tool.

2. Building a software model with parallelism based on the toolkit of a combined approach to simulation modeling of systems with parallelism is complexly formalized. It can be partially automated by using PN-patterns and individual previously analyzed components. But when building a software model with parallelism, you need to take into account the context of the tasks it solves, as it affects the way the model is built. The models synthesized in this way can be checked using the proposed method's.

3. For a complex solution to the problem of synthesis and analysis of systems with parallelism, a method's of synthesis and analysis of software system models is proposed. It is built on the basis of a combination of error localization techniques in uncertain cases and construction (correction) of the researched model with a combined approach to simulation modeling of systems with parallelism. Its application makes it possible to implement an effective mechanism for detecting dynamic errors, including cases in which the invariant method does not give clear results. It also makes it possible to increase the efficiency of model correction, which is important in terms of software design. The devised method's of synthesis and analysis of the combined simulation model of software with parallelism can be used at the software design stage, as well as at the stage of diagnosing errors in the implemented software.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study and the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available in the main text of the manuscript.

References

1. Shakhovska, N., Yakovyna, V. (2021). Feature Selection and Software Defect Prediction by Different Ensemble Classifiers. Database and Expert Systems Applications, 307–313. doi: https://doi.org/10.1007/978-3-030-86472-9_28
2. Suprunenko, O. O., Onyshchenko, B. O., Grebenovych, J. E. (2022). Analysis of Hidden Errors in the Models of Software Systems Based on Petri Nets. Èlektronnoe Modelirovanie, 44 (2), 38–50. doi: https://doi.org/10.15407/emodel.44.02.038
3. Esparza, J. (1994). Model checking using net unfoldings. Science of Computer Programming, 23 (2-3), 151–195. doi: https://doi.org/10.1016/0167-6423(94)00019-0
4. Nesterenko, B. B., Novotarskiy, M. A. (2007). Algebra protsessov dlya modelirovaniya slozhnykh sistem s real'noy rabochey nagruzkoy. Reiestratsiya, zberihannia ta obrobka danykh, 9 (4), 49–59. Available at: http://dspace.nbuv.gov.ua/handle/123456789/50907
5. Will van der Aalst, Kees van Hee. (2002). Workflow management: Models, Methods, and Systems. MIT Press, 359. doi: https://doi.org/10.7551/mitpress/7301.001.0001
6. Duarte, L. M., Kramer, J., Uchitel, S. (2015). Using contexts to extract models from code. Software & Systems Modeling, 16 (2), 523–557. doi: https://doi.org/10.1007/s10270-015-0466-0
7. Yakovyna, V. S., Seniv, M. M., Symets, I. I., Sambir, N. B. (2020). Algorithms and software suite for reliability assessment of complex technical systems. Radio Electronics, Computer Science, Control, 4, 163–177. doi: https://doi.org/10.15588/1607-3274-2020-4-16
8. Kozina, Y., Maevskiy, D. (2015). Where and When Is Formed of Software Quality? Electrotechnic and Computer Systems, 18, 55–59. Available at: https://eltecs.op.edu.ua/index.php/journal/article/view/1597/803
9. Drozd, O., Kharchenko, V., Rucinski, A., Kochanski, T., Garbos, R., Maevsky, D. (2019). Development of Models in Resilient Computing. 2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT). doi: https://doi.org/10.1109/dessert.2019.8770035
10. Cherednichenko, O. Y., Hontar, Y. M., Ivashchenko, O. V., Vovk, M. A. (2018). Analysis of Component-oriented Methods of Software Developing for E-business Engineering. Visnyk of Vinnytsia Politechnical Institute, 2, 80–88. Available at: https://visnyk.vntu.edu.ua/index.php/visnyk/article/view/2218
11. Ciccozzi, F., Cicchetti, A., Wortmann, A. (2020). Editorial to theme section on interplay of model-driven and component-based software engineering. Software and Systems Modeling, 19 (6), 1461–1463. doi: https://doi.org/10.1007/s10270-020-00812-7
12. Suprunenko, O. (2021). Combined approach architecture development to simulation modeling of systems with parallelism. Eastern-European Journal of Enterprise Technologies, 4 (4 (112)), 74–82. doi: https://doi.org/10.15587/1729-4061.2021.239212
13. Suprunenko, O. O., Onyshchenko, B. O., Hrebenovych, Yu. Ye. (2020). Analitychnyi pidkhid pry doslidzhenni vlastyvostei hrafovoi modeli prohramnoi systemy. Pratsi mizhnarodnoi naukovo-praktychnoi konferentsiyi «Matematychne modeliuvannia protsesiv v ekonomitsi ta upravlinni proektamy i prohramamy», Koblevo-Kharkiv: KhNURE, 110–113. Available at: https://mmp-conf.org/documents/archive/proceedings2020.pdf
14. Suprunenko, O. O. (2019). Combined approach to simulation modeling of the dynamics of software systems based on interpretations of petri nets. KPI Science News, 5-6, 43–53. doi: https://doi.org/10.20535/kpi-sn.2019.5-6.174596
15. Suprunenko, O. O., Hrebenovych, Yu. Ye. (2022). Instrumentalni zasoby komponentno-orientovanoho modeliuvannia prohramnykh system ta PN-paterny. Cherkasy: Vydavets Chabanenko Yu.A., 82. Available at: https://drive.google.com/file/d/1uN-zDLjUfvlKUCD1lb4GZm2w6x1Hy0AF/view
16. Kuzmuk, V. V., Suprunenko, O. O. (2010). Modyfytsyrovannыe sety Petry y ustroistva modelyrovanyia parallelnыkh protsessov. Kyiv: Maklaut, 252.
17. Suprunenko, O., Grebenovych, J. (2022). Convolution method of graph models of software components for analysis of dynamic properties of software systems. Tezy dopovidei VI Mizhnarodnoi naukovo-praktychnoi konferentsiyi «Informatsiyni tekhnolohiyi v osviti, nautsi y tekhnitsi». Cherkasy: ChDTU, 15–18. Available at: https://itest.chdtu.edu.ua/Збірник_тез_ІТОНТ-2022_макет_26_06.pdf

18. Suprunenko, O., Onyshchenko, B., Grebenovych, J., Nedonosko, P. (2023). Applying a Combined Approach to Modeling of Software Functioning. Lecture Notes on Data Engineering and Communications Technologies, 30–48. doi: https://doi.org/10.1007/978-3-031-35467-0_3
19. van der Aalst, W. M. P. (2013). Business Process Management: A Comprehensive Survey. ISRN Software Engineering, 2013, 1–37. doi: https://doi.org/10.1155/2013/507984
20. Peterson, J. L. (1981). Petri Net Theory and the Modeling of Systems. Prentice-Hall. Available at: https://dl.icdst.org/pdfs/files3/2bf95f7fde49a09814231bbcbe592526.pdf
21. Hlomodza, D. K. (2016). Zastosuvannia metodu invariantiv do analizu kolorovykh merezh Petri iz dedlokamy. Visnyk NTUU KPI. Informatyka, upravlinnia ta obchysliuvalna tekhnika, 64, 38–46. Available at: http://ekmair.ukma.edu.ua/bitstream/handle/123456789/10870/Hlomozda_Zastosuvannia_metodu_invariantiv.pdf
22. Mannel, L. L., van der Aalst, W. M. P. (2019). Finding Complex Process-Structures by Exploiting the Token-Game. Lecture Notes in Computer Science, 258–278. doi: https://doi.org/10.1007/978-3-030-21571-2_15
23. Meyer, B., Kogtenkov, A., Akhi, A. (2012). Processors and Their Collection. Lecture Notes in Computer Science, 1–15. doi: https://doi.org/10.1007/978-3-642-31202-1_1
24. Lavrischeva, E. M. (2014). Software Engineering komp'yuternykh sistem. Paradigmy, Tekhnologii, CASE-sredstva programmirovaniya. Kyiv: Naukova dumka, 283.
25. Kovalenko, A. S., Kuz'muk, V. V., Taranenko, E. A., Suprunenko, O. A., Eremeev, B. N. (2011). Description of the algorithm of interaction of information flow during tissue-cell therapy by ATM equipment. Eastern-European Journal of Enterprise Technologies, 6 (9 (54)), 43–47. Available at: http://journals.uran.ua/eejet/article/view/2343/2147
26. Zaytsev, D. A., Sleptsov, A. I. (1997). Uravneniya sostoyaniy i ekvivalentnye preobrazovaniya vremennykh setey Petri. Kibernetika i sistemniy analiz, 5, 59–76. Available at: https://www.researchgate.net/profile/Anatolii-Sleptsov/publication/315047362_Zajcev_DA_Slepcov_AI_Uravnenie_sostoanij_i_ekvivalentnye_preobrazovania_vremennyh_setej_Petri_Kibernetika_i_sistemnyj_analiz_No_5_1997_s_59-76/links/58c9056f92851c2b9d56412a/Zajcev-DA-Slepcov-AI-Uravnenie-sostoanij-i-ekvivalentnye-preobrazovania-vremennyh-setej-Petri-Kibernetika-i-sistemnyj-analiz-No-5-1997-s-59-76.pdf
27. Kryvyi, S., Grinenko, E. (2020). Ecosystems of Software Engineering. Cybernetics and Systems Analysis, 56 (4), 628–640. doi: https://doi.org/10.1007/s10559-020-00280-3
28. Hlomozda, D. K., Glybovets, M. M., Maksymets, O. M. (2018). Automating the Conversion of Colored Petri Nets with Qualitative Tokens Into Colored Petri Nets with Quantitative Tokens. Cybernetics and Systems Analysis, 54 (4), 650–661. doi: https://doi.org/10.1007/s10559-018-0066-4