JOURNAL OF ENGINEERING SCIENCES
ЖУРНАЛ ІНЖЕНЕРНИХ НАУК
ЖУРНАЛ ИНЖЕНЕРНЫХ НАУК
Web site: http://jes.sumdu.edu.ua

DOI: 10.21272/jes.2017.4(2).h1 Volume 4, Issue 2 (2017)

UDC 004.42

Model of the management program for a means complex of the design works automation as a finite-state automaton

Zakharchenko V. P.[*], Marchenko A. V., Nenia V. H.

Sumy State University, 2 Rymskogo-Korsakova St., 40007, Sumy, Ukraine

Article info:

Paper received:	November 11, 2017
The final version of the paper received:	November 28, 2017
Paper accepted online:	December 1, 2017

***Corresponding Author's Address:**
victoriait@ukr.net

Abstract. For software development it is necessary to have its mathematical model. It is established that for a means complex of the design works automation a model of a finite automaton is the best choice. The automatic machine has been chosen with a single feedback state, which asynchronously initiates the execution of design procedures, on which there are Terms of References. For this an additional requested automaton is used. This automaton implements the selection of design work according to a status of the initiated design procedure. Commands of designers also are processed by a separate automaton. Situations arising in the automated design process and are associated with designers' commands, are divided into five groups.

Keywords: program model, finite-state automaton, automation means, design works.

1 Introduction

Design takes the important place in the life cycle of technical objects. It is so because at this stage efficient, reliable and durable their functioning is provided. Fast and ecological withdrawal of technical objects from the operation also takes place at this stage. Manufacturing of technical objects continues to grow nowadays. There is a feature of modern phase of its development which recently takes place [1]. It means that the volume of design works increases in ten times every ten years. At the same time, labor productivity in manufacturing grows up to 1000 %, while in design and construction it grows only up to 20 %.

The overcoming of this contradiction by means of application the systems of design works automation (SDWA) has led to success.

2 Statement of the problem

From the beginning of systems for design works (SDWA) automation the algorithms, which were adapted to specific objects, were offered for computerized execution of design procedures. The work [2] has generalized experience of the initial stage of SDWA development. There the proposed algorithm of design for a robotic complex has been presented. It is clear, that development and improvement of software for design in the conditions of the progressing development of element base in the presence of a rigid design algorithm is almost unpromising thing. This confirms the historical experience of development of systems for design works automation, and actually its absence.

Launching the logic schemes of design was the next step [3]. Introduction of a logical description of the design process became an important stage, the essence of which is to separate the description of the design process from the software of its implementation. The lack of a formal description of design processes and appropriate information technologies have not given practical results for the last four decades.

Substantially this has been promoted by commercialization of software and significant "advertising pressure". Under its influence, separate decisions of computerization the constructing process without design, without complex approach to the tasks solution inherent to a design stage as one of the most important stages in life cycle of a technical object have become widespread.

The biggest shortcoming is the lack of a computerization for the design process management. Methodological incorrect definition of the design works automation allows managers of industrial enterprises and design organizations to report about essential level of the design works automation. Although design automation itself is practically absent, and its mechanization is at extremely low level.

Therefore, the topical problem is to develop the scientific and methodical basis of mechanization and automation of design works in general and to develop models of the software which are considered in this paper.

3 Related works

It is necessary to develop the program according to the type of solvable tasks. Nowadays it is steady to divide programs into two types [4]. Programs which transform the data (transforming programs) belong to the first type. Programs which react to actions of the user (responding programs) belong to the second type. In the pure form, such programs are encountered extremely rarely. Usually there are programs of the combined type. Reference of the program to a concrete type is defined by the prevailing part of the performed functions. The most successful programs developments essentially take into account the peculiarities of their functioning and use appropriate formalism (theory, models, methods and algorithms).

The programs complex of systems for design works automation is developed as a combined complex with a clear distribution of performed functions between programs. Programs which are used in the design procedures are developed as transforming programs. They read the output data from files, perform the corresponding design calculation and write down results in the certain file. Programs, by means of which the design process and its management are organized, belong to type of responding programs. Exactly the last ones as more complex programs are considered in this work.

The first from eleven fragments of a flowchart of the technical object automated design is presented in Figure 1 from the work [2].

The analysis shows that this fragment contains four from forty states of the decision-making program regarding the directions of the continuation of the design. The states of control for reading and writing the design data are added to the mentioned states (ten cases on a fragment). Therefore, it is clear that the main program of the design process management has to belong to a type of the responding programs. According to the chosen type of programs, we use the corresponding model of the program, which represents the main software as a state machine [5].

In native practice it is accepted to call such machine as a finite-state automaton [6]. The theory of finite-state machines considers several problems, implementation of which guarantees that the proposed solution will match the necessary requirements in the best way. Firstly, the automaton must be minimal. The minimal automaton is an automatic machine which has the smallest possible quantity of states and implements the set function of outputs.

For any finite-state automaton the equivalent to it a finite-state automaton with the smallest number of states can be constructed [7].

The minimality of the automaton provides the minimal cost and the maximum reliability of work. Secondly, the automaton has to give an opportunity of the transition into any set state. This requirement provides the implementation of functional requirements completeness.

The authors' choice of a finite-state automaton as a model of software for automated design management is supported by the current tendency to develop the controlling programs of various directions [8–10].

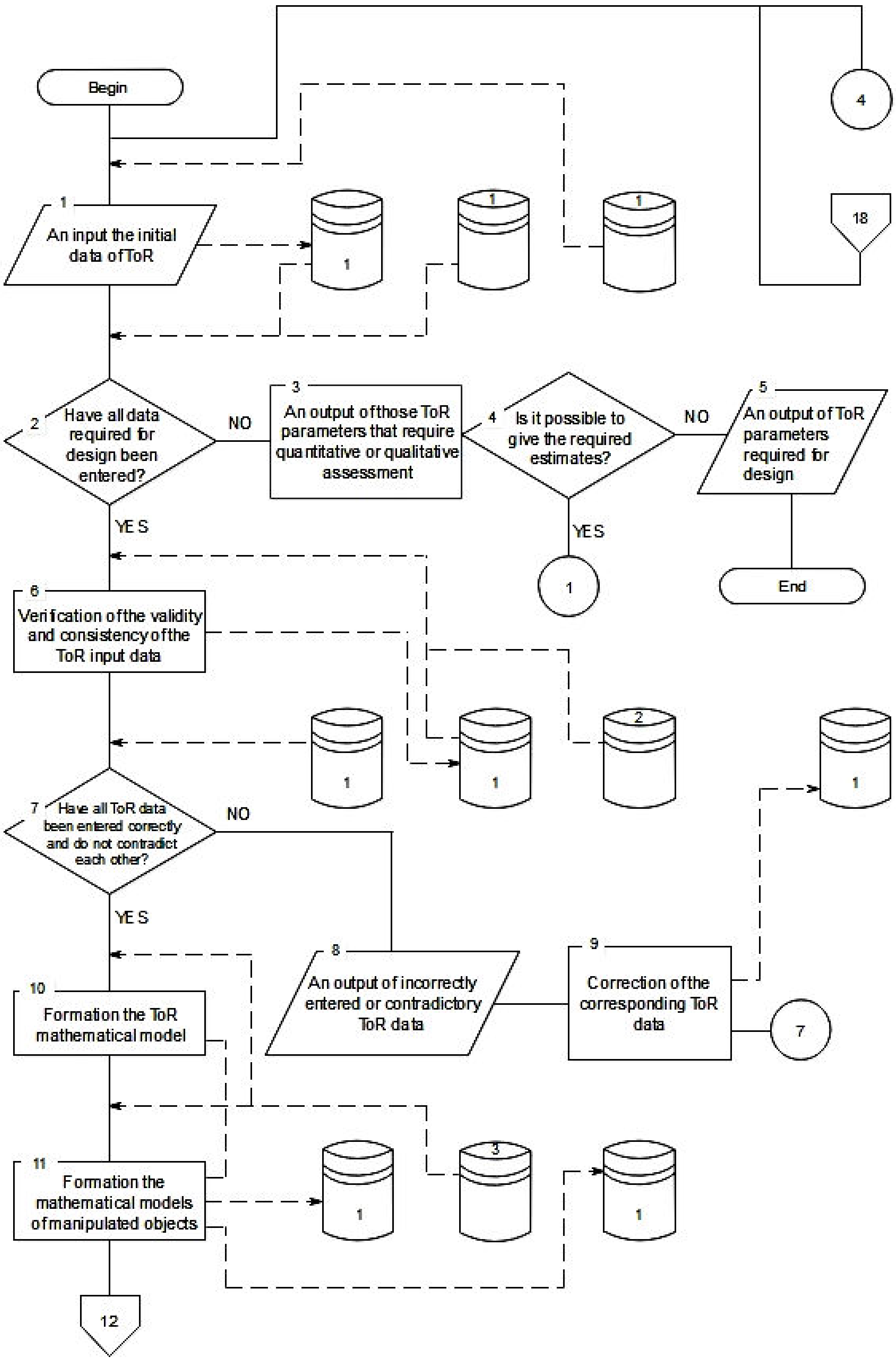
Application the modelling of the responding programs in the form of finite-state automatons continues to be investigated in various directions. They have been analysed below.

The work [11] is devoted to the development of logical control programs models and corresponding formalism.

The finite-state discrete automaton as a management program model is described by directed graphs as the use of a large number of the machine states is supposed. The problem of parallel processes management is solved by the decomposition into several graphs. The problem of coordination connected with the use of the synchronism concept is solved separately.

In the work [12] the synthesis of control systems, which use the microcontrollers, is presented. The offered method of automatic programming is considered below. This method is based on the formation of the program model according to the structural scheme of a control automaton new type. The new structural software model differs by introducing the multiplexer for selection one logical condition according to the code of the automaton state. This condition is checked at each step of the program model work. The offered method is focused on the structural organization of automatons with an input multiplexer. The result of this variant of automatic programming are high-speed programs with a minimal number of commands of the program code.

The problem of ambiguities elimination during the work of a finite-state automaton within the creation of control program systems is considered in work [13]. The structural scheme of the finite-state automaton model was created by the authors. It was done by taking into account the waiting time for the performance of functions and the result of their performance, the counting of the events number and stacks of states. It is concluded that the use of a finite-state automaton in the field of the program systems allows to determine the behavior of the program, to minimize the number of errors in the program logic and formalize the development process. The authors found it convenient to use the index-matrix approach to solve the problem of changing the automaton states.



Begin
1
An input the initial data of ToR
2
Have all data required for design been entered?
NO
YES
3
An output of those ToR parameters that require quantitative or qualitative assessment
4
Is it possible to give the required estimates?
NO
YES
5
An output of ToR parameters required for design
End
4
18
1
2
3
6
Verification of the validity and consistency of the ToR input data
7
Have all ToR data been entered correctly and do not contradict each other?
NO
YES
8
An output of incorrectly entered or contradictory ToR data
9
Correction of the corresponding ToR data
7
10
Formation the ToR mathematical model
11
Formation the mathematical models of manipulated objects
12

Figure1 – Fragment of the automated design flowchart

In the work [14] the models of programs for management of the parallel and distributed processes are presented by three groups: the generalized models, the models of data flows and the models of parallel processes with the interaction between them. Preference is given to models of the third group. At the same time it is specified their shortcomings. For example, the models do not allow a possibility of asynchronous interaction between processes (only via the rendezvous mechanism) and do not contain mechanisms for the creation of flexible templates of management modules. A multi-level model of distributed programs has been offered. It consists of the level of module templates (the highest level), the levels of distributed and embedded modules, and the structural and register level (the lowest level).

Along with this, it is recognized that such model can be suitable for systems with a predetermined set of tasks. However, it is not suitable for the models of programs for design works automation because a means complex of the design works automation has to be constantly extended and be improved [3]. Models of programs can be built both on the basis of one finite-state automaton, and on the basis of their combinations.

So in the work [15] the model on the basis of parallel automatic machines or one automatic machine, which has a set of descriptions of simultaneous partial states, is investigated. On the basis of sets of the atomic, partial or full state of the parallel automaton and parallel inputs the parallel functions of transitions and parallel outputs, which are set by matrixes of transitions and outputs, are defined. This way makes it possible to implement three types of the processes interaction: synchronous, ordinary and mixed. Conditions for the transformation of a parallel automaton into parallel-sequential compositions of simple automatons also have been formed.

Additionally to the peer automaton there is a possibility to create the hierarchical structures of automatic machines. So in the work [16] the set-theoretic description of such model has been investigated. There is a main automaton on top of the hierarchy. Embedded automatic machines perform subordinate roles and are located below in the hierarchical structure. Information communications on reception a condition of the management object and on giving to it certain management commands take place at all hierarchical levels of the structure of the management automaton. The efficiency of the proposed hierarchical model has been demonstrated both on Moore machines and on Mealy machines.

In addition to hierarchical, the models use regular structures in the form of cellular [17] and multitape structures [18].

Considerable attention continues to be paid to the minimization of finite-state automatons taking into account previously adduced arguments and which are supported by other researchers definitely. In the work [19] the solution of a problem of minimization for the automatic machine, behavior of which depends on the time when the input command or event occurs, and to which the automaton responds, is considered. It is also shown that there is a single minimal form for a fully determined automaton with temporary restrictions. The algorithm of minimization for a fully determined automatic machine with temporary restrictions is offered. It has been proposed on the basis of creation the splittings according to such equivalent states, at which if there is the identical number of identical commands at the outputs, we will have the identical results.

The algorithms needed simultaneously for solution of two problems of minimization of nondeterministic finite-state automatons, vertex and arc ones, are given in the work [20]. Taking into account the fact that the arcs correspond to the functions performed by the automatic machine, the reduction of the arcs number is not acceptable. It is so because the object that simulates such automaton will not conform to the specified requirements. A useful opportunity is the simultaneous construction of the functions for the states markup.

Fuzzy machines are considered in the work [20] from the position of using the minimax criterion for evaluating their functioning if there are fuzzy commands on the input. The results of this work can be used for processing the dialogue of designers and constructors based on the elements of their professional communication language.

4 Purpose of the work

The purpose of the work is to develop a model of the functioning of the system software for a means complex for the design works automation.

The research object is the organization of the technical objects design process.

The research subject is the model of program implementation of the design process organization.

5 Research results

The process of the functioning the management program of a means complex for the design works automation is the basis of the software tools functioning. It is impossible in advance to predict all possible aspects, structure and types of design works if it is necessary to solve new tasks [3]. Despite this still there is a need to develop and use the automation means. Therefore, it is expedient to solve this contradiction by using a different approach to the design of a finite-state automaton as a model of means for design works automation instead of its synthesis taking into account all performed functions and transitions into all necessary states. To implement this gained experience in theory of inventive problem solving is used, for example [21]. According to provisions of this theory the proposed solution is better when it gets closer to the ideal end result. At the same time the ideal end result is understood as that one, at which the required function, is performed and the object itself, which performs this function, is absent. It determines the provision for the development of the design object structure of the minimum complexity.

For implementation of this provision the following consecutive algorithm of development is used:

1) to offer the functional model of the investigated object;

2) to choose an element, which implements the functional model;

3) to improve an element for the implementation of a full design process;

4) to supplement the element with necessary components for the information technology implementation.

Under the condition of self-management and usage of own mechanisms of actions performance, the functional scheme of the work of a means complex for the design works automation takes the form, which is shown in Figure 2, where *ToR* – Terms of Reference.

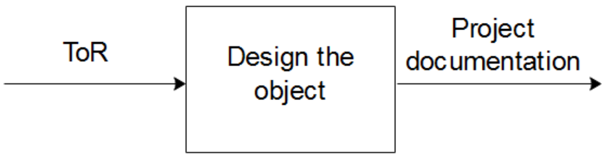


Figure 2 – Context diagram of the functioning of a means complex for the design works automation

This diagram should be considered as a model of the ideal tool, which in one step allows to perform the design in accordance with the requirements of Term of Reference (ToR).

Due to the design of a new mechanical engineering object (new structure and performed functions) every time, it is impossible in advance to predict all possible logic design schemes and corresponding processes of the work of the design management subsystem. It is also impossible to offer the fixed order of the design and its program implementation even for one object, its components of different degree of complexity and various structures of the construction, which are unknown in advance.

At the same time it is obvious that the management subsystem must pass through a certain number of fixed states, in which the analysis of the current situation is carried out. Also the corresponding decisions according to ways of the continuation of any process, both the manufacturing design process and the auxiliary process (code conversion, reformatting, etc.) are made.

Such behavior of a subsystem is implemented in the models of finite-state automatons. They are offered for use as information technology of processing of the current data both about the mechanical engineering objects design and about processes of its formation and interaction of the components of human, information and software complexes.

According to the accepted technology for the finite-state machines design [22, 23], it has been carried out the decomposition of the model from the Figure 2, which takes the following form (Fig. 3), where ToR_i – the state of the design object structure; SDP_i – the state of the corresponding design process in accordance with the implementation of manufacturing tasks.

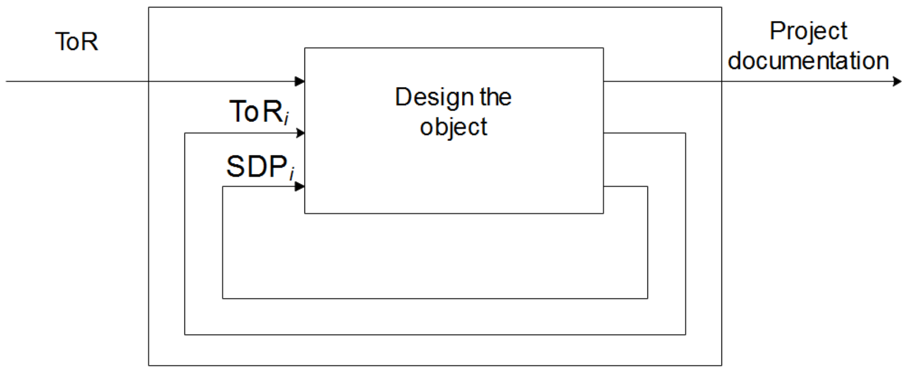
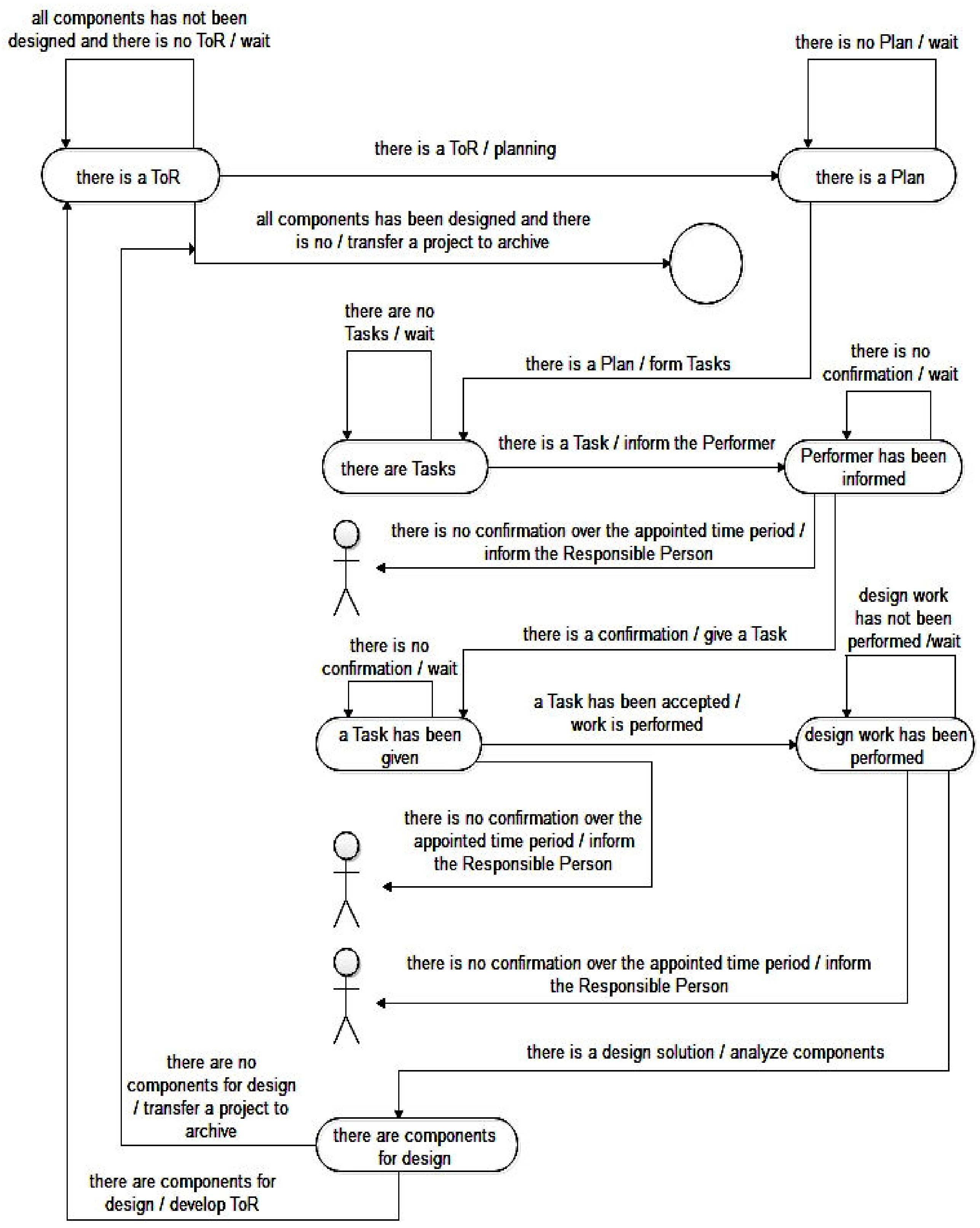


Figure 3 – Decomposition of the generalized scheme of the finite-state automaton

This decomposition considers that a finite-state machine changes the waiting state if there is any change in the state of the design process and the change of the state of arbitrary components in the design object structure. The automaton works asynchronously according to the cyclic scheme with a feedback. For the transitions performance (actions for design and management) additional automatic machines for the performance of own transitions are involved.

Such technique excludes a problem of the state uncertainty: a possibility of performance of all transitions and going through all necessary states. To exclude the need to remember all previous commands and states, the states vectors of the design object structure and the states of the design processes are introduced.

The model of a finite-state automaton, which processes the states of projects, has its structure shown in Figure 4. The model of a finite-state automaton, which processes the actions of designers according to the implementation of design processes, is presented in Figure 5.



all components has not been designed and there is no ToR / wait
there is no Plan / wait
there is a ToR / planning
there is a ToR
there is a Plan
all components has been designed and there is no / transfer a project to archive
there are no Tasks / wait
there is a Plan / form Tasks
there is no confirmation / wait
there are Tasks
there is a Task / inform the Performer
Performer has been informed
there is no confirmation over the appointed time period / inform the Responsible Person
design work has not been performed /wait
there is a confirmation / give a Task
there is no confirmation / wait
a Task has been accepted / work is performed
a Task has been given
design work has been performed
there is no confirmation over the appointed time period / inform the Responsible Person
there is no confirmation over the appointed time period / inform the Responsible Person
there is a design solution / analyze components
there are no components for design / transfer a project to archive
there are components for design
there are components for design / develop ToR

Figure 4 – Model of the finite-state automaton which processes the projects formation

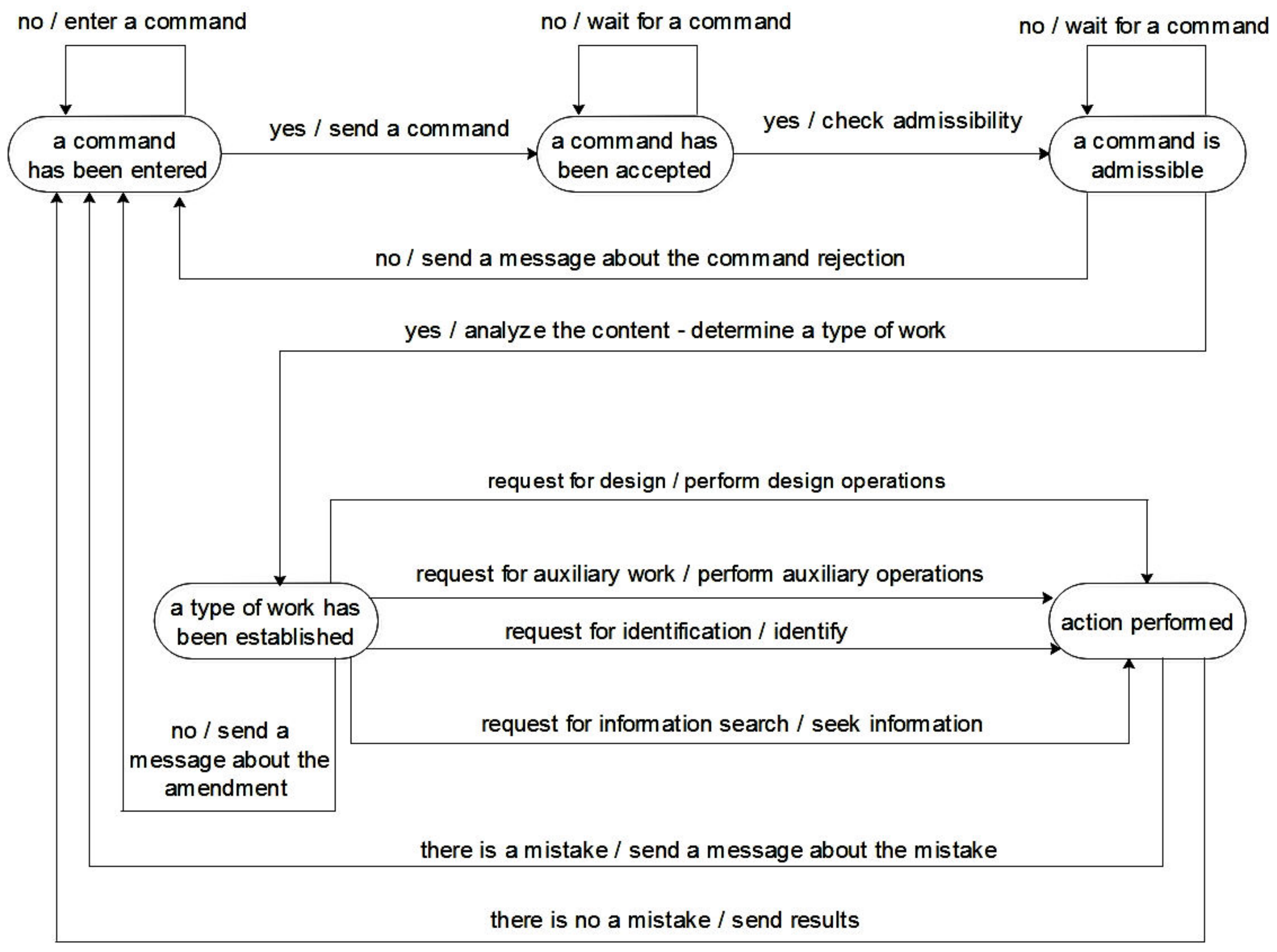

no / enter a command
no / wait for a command
no / wait for a command
a command has been entered
yes / send a command
a command has been accepted
yes / check admissibility
a command is admissible
no / send a message about the command rejection
yes / analyze the content - determine a type of work
request for design / perform design operations
request for auxiliary work / perform auxiliary operations
a type of work has been established
request for identification / identify
action performed
request for information search / seek information
no / send a message about the amendment
there is a mistake / send a message about the mistake
there is no a mistake / send results

Figure 5 – Model of the finite-state automaton which processes the commands of designers

For these models states are marked as ovals, transitions (performed actions) as lines. In inscriptions conditions are indicated firstly. The next there are actions performed according to them.

In the work [24] it is pointed out the complexity of application the model of the finite-state automaton to the distributed computer systems. The complexity exists because of the practical impossibility of fixation the structure of all its components. The use of such argument is justified in a case of the parallel performance of interconnected, and as a result controlled, calculations. If the asynchronous performance of the independent design works is applied to this case, the given argument is not significant. In this situation the state of each implemented projects, which is fixed in vectors sets of a state of each design procedure and each design object, is essential.

The offered linear sequence of states and actions guarantees performance of all actions and going through all states. The single-step type of the automatic machines work provides uniformity of their functioning and a possibility of their application to the design of objects with any structure.

6 Conclusions

The process of functioning the management program of a means complex for design works automation in the form of a finite-state automaton was presented during the performance of this research. The uniform description for functioning of the software means for the organization of automated design and its management was achieved.

7 Further work

If the mechanism of functioning of the software means for the organization of automated design and its management is in the presence, it demands the development the uniform information description of assignments for design.

References

1. Prohorov, A. F. (1987). Konstruktor i EVM. *Mashinostroenie* [in Russian].
2. Budya, A. P., Kononuk, A. E., et. al. (1988). Spravochnik po SAPR. *Tehnika* [in Russian].
3. Zhuk, K. D. & Timchenko, A. A. (1983). Postroenie sovremennyh system avtomatizacii proektirovaniya. *Naukova dumka* [in Russian].
4. Karpov, U. G. (2010). Model cheking. Verifikacija parallel'nyh i raspredelennyh programmnyh system. *BHV-Peterburg* [in Russian].
5. Fedotov, I. E. (2012). Modeli parallel’nogo programmirovaniy. *Solon-Press* [in Russian].
6. Amosov, N. M. & Artemenko I. A. (1974). Jenciklopedija kibernetiki. *Ukr.-sov. encikl.* [in Russian].
7. Gill, A. (1966). Vvedenie v teoriyu konechnuh avtomatov. *Nauka* [in Russian].
8. Zaboleeva-Zotova, A. V. & Orlova, U. A. (2010). Avtomatizaciya nachal’nuh etapov proektirovaniya programmnogo obespecheniya. *Izvestija Volgogradskogo gosudarstvennogo tehnicheskogo universiteta*, Vol. 6 (8), 121–124 [in Russian].
9. Filatov, V. A. & Kozyr’, O. F. (2013). Model’ povedeniya avtonomnogo scenariya v zadachah upravleniya raspredel’onnymi informacionnymi resursami. *Inzhenernyj vestnik Dona*, Vol. 26, No. 3 (26), 24–36 [in Russian].
10. Kozachenko, V. F. (2010). Effektivnyj metod programmnoj realizacii diskretnyh upravljajushhih avtomatov vo vstroennyh sistemah upravlenija [in Russian].
11. Novozhilov, B. M. (2015). Primenenie grafov v razrabotke programm dlja PLK. Vol. 2, p. 6 [in Russian].
12. Muchopad, U. F. & Muchopad, A.U. (2014). Analiz i sintez upravljajushhih avtomatov slozhnyh tehnicheskih sistem. *XII Vserossijskoe soveshhanie po problemam upravlenija VSPU-2014*, pp.7295–7306 [in Russian].
13. Smirnova, N. V. & Smirnov, V. V. (2014). Primenenie teorii konechnyh avtomatov v razrabotke programmnyh sistem. *Tehnika v sil's'kogospodars'komu virobnictvi, galuzeve mashinobuduvannja, avtomatizacija*, Vol. 27, 316–320 [in Russian].
14. Bolshakov, O. S., Petrov, A.V., et. al. (2015). Model' raspredelennyh programm dlja vstraivaemyh sistem. *Vestnik Rybinskoj gosudarstvennoj aviacionnoj tehnologicheskoj akademii im. P.A. Solovyova*, Vol. 1 (32), 165–171 [in Russian].
15. Vorobev, V. A. (2006). Model’ parallel’nogo avtomata. *Avtometrija*, Vol. 42 (3), 85–93 [in Russian].
16. Kyzmin, E. V. (2006). Ierarhicheskaja model’ avtomatnyh programm. *Modelirovanie i analiz informacionnyh sistem*, Vol. 13 (1), 27–34 [in Russian].
17. Schiff, J. L. (2011). Cellular automata: a discrete view of the world. *John Wiley & Sons*.
18. Furia, C. A. (2012). A survey of multi-tape automata. *arXiv preprint*, arXiv:1205.0178.
19. Tvardovskii, A. S. & Evtychenko, N. V. (2014). K minimizacii avtomatov s vremennymi ogranichenijami. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitel'naja tehnika i informatika*, Vol. 4, 77–83 [in Russian].
20. Melnikov, B. F. & Melnikova, A. A. (2011). Mnogoaspektnaja minimizacija nedeterminirovannyh konechnyh avtomatov. Chast’ I: Vspomogatel’nye fakty i algoritmy. *Izvestija vysshih uchebnyh zavedenij. Povolzhskij region. Fiziko-matematicheskie nauki. Matematika*, Vol. 4 (20), 59–69 [in Russian].
21. Orlov, M. A. (2006). Osnovy klassicheskoj TRIZ. Prakticheskoe rukovodstvo dlja izobretatel’nogo myshlenija. *Solon-Press* [in Russian].
22. Lee, E. A. & Varaiya, P. (2007). Structure and Interpretation of signals and Systems. *Lee & Seshia*.
23. Karpov, Y. G. (2003). Teorija avtomatov. *Piter* [in Russian].
24. Cheremisinov, D. I. (2011). Proektirovanie i analiz parallelizma v processah i programmah. Belarusskaja nauka [in Russian].