DOI: 10.17277/amt.2019.02.pp.069-080

Architectural Design of Software Systems Implementing Models of Process Equipment

S.Yu. Alekseev

JSC "Roskhimzashchita", 19, Morshanskoe shosse, Tambov, 392000, Russia

Corresponding author. Tel.: +7 999 725 94 93. *E-mail*: isz@roshimzaschita.ru

Abstract

The paper describes the fundamentals of architectural design methodology of software systems implementing the tasks of mathematical modeling of technological processes / objects in the chemical industry. The existing software systems focused on solving engineering problems based on mathematical modeling of physical and chemical processes carried out in technological devices are analyzed. The authors show that all software systems are based on the use of numerical solutions of mathematical modeling tasks for technological processes / objects, which significantly limits their ability to organize the computation process and requires an individual approach to the development of each software system.

The urgency of developing a universal methodology for constructing software systems implementing models of the technological processes and equipment based on analytical solutions of mathematical physics tasks in partial derivatives is discussed. The developed methodology includes principles for formulating mathematical statements of tasks and methods for solving them ensuring their effective computer implementation; principles for building software systems as structural and dynamic analogues of technical systems. The application of the methodology is illustrated by the example of developing a software system implementing a model of non-stationary heat exchange processes in a shell-and-tube heat exchanger. The main emphasis in the example is put on the dynamic compliance of the software system and process equipment. It is shown that the calculation of the characteristics of the processes occurring in the heat exchanger simultaneously can be implemented in parallel in the software system, and the characteristics of successive processes can also be calculated sequentially.

Keywords

Computer model; software system; computation model; development model; mathematical modeling.

Introduction

Mathematical modeling is now widely used to improve the performance of complex technological equipment. Modeling is the main tool that provides additional information about the processes occurring in it and is used to solve design, optimization, management, decision-making and forecasting tasks. In this paper the process equipment is considered as a technical system – a set of related and interacting elements. The most complete information about the patterns and conditions of non-stationary processes occurring in technical systems can be obtained by mathematical modeling of the fields of their target characteristics.

In this case, solutions of mathematical modeling tasks are considered as a combination of mathematical operators and algorithms for their use, aimed at obtaining quantitative characteristics of the fields. Despite the fact that the solutions presented in this form fully provide the possibility of calculating the fields, it is impossible to use them manually without computing technologies in most cases. A one-time calculation involves a large number of intermediate calculations. In practice, such calculations are carried out repeatedly, for different values of the source data.

Thus, for modeling tasks which are solved in order to increase the quality of technical systems, it is not the solutions themselves that are of major importance, but their software implementation. Programs can represent separate information systems interacting with a person, or function as part of technical systems interacting with their component parts. Solutions are used as the basis of the program, but the program brings the main result. The user or other information systems interact with programs, but not with algorithms and mathematical operators. The form of the software implementation of algorithms and mathematical operators, as well as the decisions themselves, determine the quality of the simulation results obtained: the number of assumptions made, the accuracy, the time for obtaining the results and the further possibilities for its use. Therefore, the software implementation of solutions and its quality, i.e. such factors as the program structure, the complexity of its data flow and control, the efficiency of using computing resources, calculation time, reliability, resistance to incorrect source data, the ability to reuse code, are of fundamental importance for modeling tasks.

Materials and methods

Software implementing the tasks of modeling processes which occur in technical systems varies from narrowly specialized applications focused on the efficient solution of tasks within one application area to large general-purpose applications designed to solve a wide range of tasks in various application areas.

General-purpose systems

General-purpose applications are a multi-purpose analysis tool that can be used in various technical disciplines. Such applications as *ANSYS*, *SolidWorks Simulation*, *Autodesk Simulation*, *Abaqus FEA*, and *MOOSE* can be mentioned as an example. These are large applications focused on working with powerful desktops or specialized computing stations.

A characteristic feature of these applications is the unification of formulating modeling tasks, methods for their solution, and forms for presenting the results obtained [1 – 7].

The unification of statements is achieved due to the fact that the user tasks are constructed as special cases of general statements, which are formulated separately for each class of processes. Processes are usually classified according to the physical disciplines and engineering applications on which they are oriented, e.g. hydrodynamic processes, deformations of solids, heat and mass transfer, high and low frequency oscillations. This form of task statement can be used in various, unrelated applied fields.

Each common task chooses its own solution which is used to solve all private, custom tasks based on it. The main criteria for choosing a method are universality, wide dissemination, knowledge and predictability of the obtained results when it is used within the framework of the class of tasks which the general task refers to. To solve systems of differential and algebraic equations expressing the formulations of modeling tasks, numerical methods are currently the most widely used. This fact and the criterion of universality become the reason that the finite element method and its varieties are used in all the considered applications for solving modeling tasks [1 – 7].

The result of such applications is a one-time solution of a particular task with obtaining the fields of the target characteristics of the modeled processes in time and space. At the same time, the boundary conditions of the task and the values of the constants determining the course of the processes are formulated at the stage of task statement and are not changed during the calculation. When changing the boundary conditions, the task must be set and solved by the user again. This is the reason for the additional assumption that the boundary conditions and constants are permanent for the entire duration of the process.

In real technical systems, several interdependent processes take place simultaneously. The boundary conditions and the values of the constants determining the course of one of the processes may change under the influence of others. In this case, the search for process characteristics is possible with repeated, iterative repetition of calculations for short time intervals, when new source data are set for each interval.

In the considered applications, the functionality is not developed to ensure multiple calculations using the previously obtained results, which describe the state of the system at the time of the end of the previous time interval, as the source data. Difficulties in organizing such calculations are also determined by the form of presenting the simulation results, which in these systems, as well as the formulation and methods of solution, is focused on a universal and visual presentation of the results of solving a wide range of tasks in different, unrelated applied areas to the user. The results are presented visually in the form of a set of values, which quantitatively characterize the fields of the target characteristics, and graphical dependencies. They are poorly oriented for further, repeated use in computational procedures, providing an iterative algorithm, or transfer to other automated systems ensuring their operation. This makes it difficult to take into account the specifics of the task being solved and narrows the scope of application of the simulation results.

The options for determining the system of assumptions and uniqueness conditions for constructing a particular task are limited to the most common ones.

Therefore, it is impossible to take into account in detail the operation features of the simulated processes in the technical system.

The main feature of general-purpose systems focused on the unification of solutions for a wide range of different tasks is the necessity to find a compromise between the costs on performing calculations and the accuracy of the results [8]. This is due to the large number (from thousands to hundreds of thousands) of finite elements into which the process areas are divided. For each element, the system of equations describing its state and behavior is repeatedly iteratively solved. The set of individual states of all elements determines the state of the object being modeled.

Designing a large number of particular tasks on the basis of one general task by setting the values of the coefficients in the system of equations may cause discrepancy in the obtained results, which do not correspond to the physical picture, though the problem is formally correct. For complex software systems, in which the software system implementing the modeling tasks is included as an integral part and ensures the operation of other systems, it is of fundamental importance. In this case, the simulation results are transmitted directly to other systems without the possibility of their analysis and adjustment by an expert.

Thus, despite the desire for universality of using and expanding the range of solved modeling problems, the general-purpose systems have their own specific application niche and the area of modeling tasks being solved. This is stipulated by the features listed above.

Specialized systems

Besides universal general-purpose software systems, there is a large number of specialized software systems. The difference consists in detailed considering only a narrow class of tasks within one application area allowing to obtain high-quality simulation results. The spread of computer modeling for solving applied problems and individual requirements for the quality of modeling imposed on tasks within each applied area determine the excess of the number of specialized software systems over the number of general-purpose systems.

Compared to universal systems, specialized ones allow for a more thorough formulation of tasks, taking into account a greater number of factors and their interrelations which determine the course of simulated processes during their formulation. The solution methods used in these systems allow to obtain more accurate results compared to the methods used by universal general-purpose systems. Increasing their accuracy is achieved by taking into account the specifics of the task determined by the applied area, the nature of the processes, the structure of equations and the way the simulation results are used.

A detailed account of the task features at the stages of its formulation and solution determines the improvement of the quality of simulation results when using specialized systems. The consequence of such opportunities is a narrow range of tasks. It is possible to note the tendency according to which the more specialized the software system is, the higher the quality of the obtained results [9 – 15].

Specialized programs do not require less computational resources for their work compared to universal ones. In most cases the obtained results are not transmitted to other subsystems and are presented to the user. The issues of improving the performance of calculations are considered, but there is no hard limit on the time for obtaining the results.

The detailed consideration of the process features complicates the task and requires additional computational resources when solving it. In general, both specialized and universal systems require the same number of simulation results, which must be as high as possible to increase their accuracy and quality.

Among the specialized modeling programs, we can single out those based on agent-based modeling (**ABMS**). The solution of modeling problems in these systems is based on the property of software agents to have conscious behavior – reactive, proactive, and social. Agent-based modeling is primarily used to study the behavior and interaction of individuals or organizations in specific situations or environments. ABMS can be used to study an existing system, which is analyzed to understand its behavior in response to specific changes in the environment or at the design stages, when creating new systems [16].

The main disadvantages of universal modeling systems are the limited ability to correctly formulate a modeling problem, taking into account the features of the domain and the physical system in which the simulated processes take place, the automatic addition of the task statement with expressions and constants in order to ensure the possibility of its solution when defining the task, which cannot be solved by the methods used in the program. This provides the possibility of obtaining plausible results with the obviously incorrect formulation of the task, but these results will not reflect the true picture of the process. These systems can be used to solve common typical problems, where there is no special need to take into account a large number of specific features. The simulation results must be presented to the expert so that he can assess their adequacy and, if necessary, correct the formulation of the task with available means, adjust the implementation of the solution method and repeat the calculation again.

The disadvantage of specialized programs is a small range of tasks for each of them. Taking into account the peculiarities of the processes occurring in them is so specific that it is very difficult to use them outside the subject area and the range of tasks for which they were developed. When new tasks arise, it becomes necessary to develop a new modeling program. There is no universal method for developing programs for modeling processes occurring in technical systems. The development of each system is a creative, iterative process that requires a lot of time, human resources and, consequently, finances.

A common feature of universal and part of specialized programs is that they implement the design and solution of a mathematical problem. The physics and nature of the simulated processes are considered only at the stage of its design. The resulting problem is solved further independently using software implementations of only mathematical operators. The solution results are the values of individual functions, each of which represents the target field characteristics of a single element of the technical system. With such an organization of modeling, the dynamics of interaction of the technical system elements is considered to be limited. There is a fundamental discrepancy between the time diagrams of the program and the time diagrams of the technical system – their statuses do not match at any equal time intervals from the beginning of the technical system and the program operation.

Now computer simulation is performed in several stages. First, assumptions are defined which simplifies the understanding of the technical system and the processes occurring in it. Then, for the obtained simplified representation of the technical system, a mathematical model is constructed. As a result, the technical system is replaced by a mathematical one, with new essential connections between them, which do not coincide with the essential connections of the original technical system. The next step is to solve the resulting system of equations and its software implementation. It also modifies the entities and relationships between them, which were present in the mathematical system, into linguistic elements of the programming language and fragments of the executed code representing continuous processes in discrete form. We can say that the representation of the technical system is distorted in the process of computer simulation. This happens more or less at all stages of computer simulation. The user interacting with the program sees the program representation of the technical system, which is the result of three transformations.

The modern level of the theory and methods of software systems engineering allows to take into account the specifics and to expand the possibilities of using analytical solutions. This implies the definition of software in terms of system links and functioning patterns of the technical system elements, where the simulated processes take place, but not in terms of the mathematical formulation of the task and algorithms for its solution. In this case, the software implements the calculation of the characteristics for the processes occurring in the technical systems on the basis of calculations which correspond to these processes structurally and dynamically. Wherein, the software itself is represented as a software system that structurally and dynamically corresponds to the technical system. The principles of designing the software systems and the principles of their functioning, which form the basis of the new architectural design methodology, have been developed for this purpose.

Construction and operation principles of software systems

The paper proposes a universal method of architectural design of software systems for process equipment based on the principle of structural and dynamic correspondence of calculations to the technical system. It covers two main phases of the software life cycle – the development phase and the implementation phase.

The general provisions of methodology are considered on the example of building a software system for modeling an industrial adsorber (Fig. 1) and involve decomposing a technical system, excluding those components that are not used in calculating the required characteristics from the point of the adopted system of assumptions, and determining for each of its elements a set of quantities characterizing its state, which are necessary for calculating the required characteristics. The decomposition of a technical system is performed to a level at which structural elements obtained by analytical solutions can be received.

Next, the image of the technical system is constructed in the address space of the computing complex. For each element of the technical system, its software abstraction is created. It is based on the methods of object-oriented design and presents a computational entity consisting of a set of variables describing its state and a set of functions that perform operations on them. Functions provide abstraction behavior.

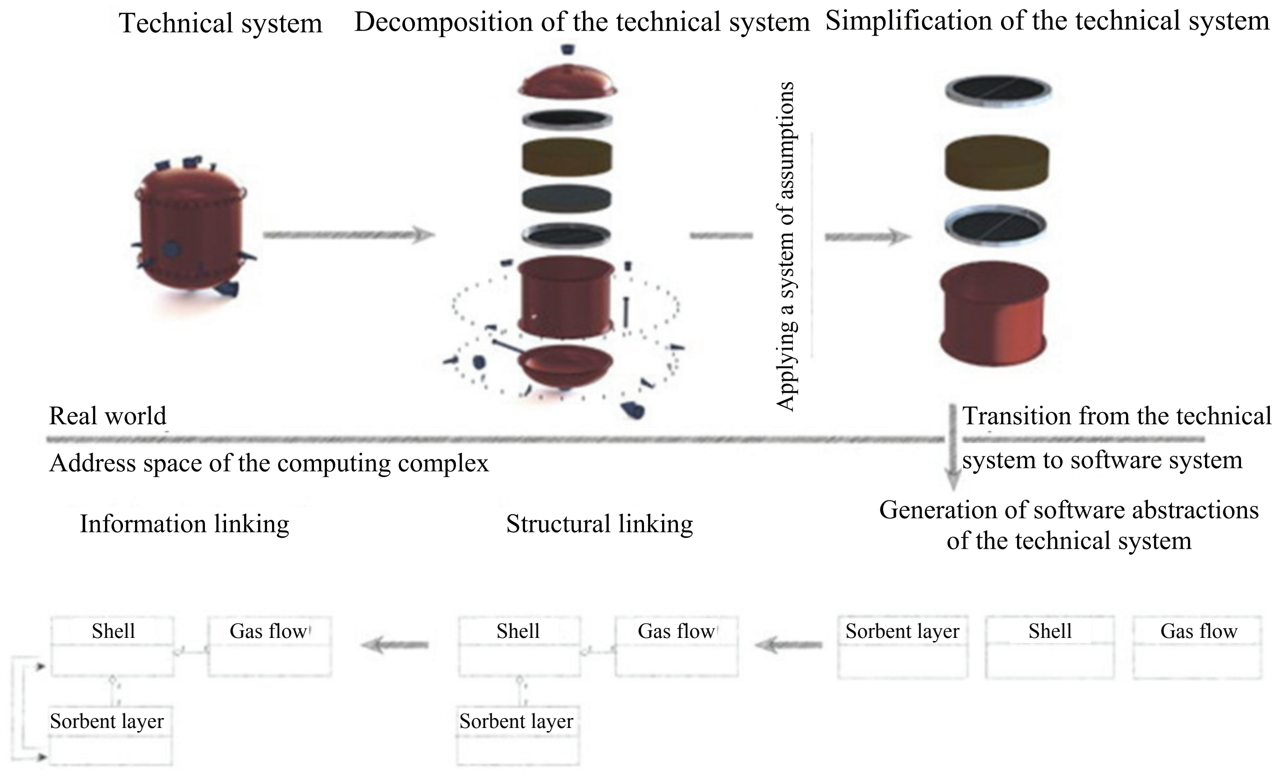
Technical system
Decomposition of the technical system
Simplification of the technical system
Applying a system of assumptions
Real world
Address space of the computing complex
Transition from the technical system to software system
Information linking
Structural linking
Generation of software abstractions of the technical system
Shell
Gas flow
Sorbent layer
Shell
Gas flow
Sorbent layer
Sorbent layer
Shell
Gas flow

Fig. 1. Building a software image of the technical system on the example of an industrial adsorber

Abstractions are interconnected so that they organize a structure that coincides with the structure of the technical system. These are structural links providing an image in the software system of the technical system design. Further, information links representing the interaction of the technical system elements are added between the abstractions.

Sets of variables equivalent to the previously defined sets of values that characterize the state of the technical system elements are implemented for abstractions. Within each abstraction, a local mathematical problem is constructed and solved to calculate the values of its variables. It is constructed considering certain interrelations of abstractions. Then functions ensuring the calculation of the values of state variables and implementing the calculation of mathematical operators that represent analytical solutions are determined.

The calculation of the processes characteristics is provided by the interrelated work of abstractions. Their interconnection in the calculation process is carried out through the exchange of messages and through channels built on the basis of established links representing the interaction of the technical system elements.

The main work of software abstractions is calculating the properties of the corresponding elements of the technical system. The structural conformity of the program and the technical system allows the use of a single method for describing their states. It is presented as a set of quantitative and qualitative characteristics of the processes occurring in its structural elements, a set of characteristics of the structural elements themselves and a set of characteristics describing the interaction of elements in the following form:

$$S = S_1\left(\left\{s_1^c,...,s_{a_1}^c\right\}, \left\{s_1^p,...,s_{b_1}^p\right\} \right),...,$$
$$S_n\left(\left\{s_1^c,...,s_{a_n}^c\right\}, \left\{s_1^p,...,s_{b_n}^p\right\} \right),$$
$$\left\{ I_1\left(i_1,...,i_{c_1}\right),..., I_m\left(i_1,...,i_{c_m}\right) \right\}, \qquad (1)$$

where n is the number of elements in the system, $S_i\left(\left\{s_1^c,...,s_{a_i}^c\right\}, \left\{s_1^p,...,s_{b_i}^p\right\}\right)$ is the state of the i-th element of the technical system, $\left\{s_1^c,...,s_{a_i}^c\right\}$ is the vector of the structural element characteristics, $\left\{s_1^p,...,s_{b_i}^p\right\}$ is the vector of the processes characteristics occurring in it, m is the number of links of the system elements, through which the elements interact, $I_1\left(i_1,...,i_{c_1}\right)$is the link of the system elements, $i_1,...,i_{c_1}$ is the vector of interaction characteristics for the i-th link.

Technical system	*Software system*
State of the element	*State of abstraction*
Changes in the state of elements	*Changes in the state of abstraction*
Interaction of elements	*Information interchange between abstractions*
Impact of element I_1 on element I_2	*Sending a message by abstraction $A(I_1)$ to abstraction $A(I_2)$*
Impact type	*Message type*
Exposure time	*Message time*
Changes in the state	*Calculation of the state*
Time of the state changes	*Calculation time of the state*

Fig. 2. Compliance of the processes in the software and technical systems

The main work of each abstraction is aimed at calculating the values of the variables contained in it, describing its state. The software abstractions, as well as elements of the technical system during the working process, constantly interact with each other, as a result of which their state changes. This ensures the need to perform multiple calculations of the state of each abstraction. The initial data for calculating the state of each abstraction are the data of neighboring abstractions associated with it. This corresponds to the change in the state of the technical system elements under the influence of related elements.

Below (Fig. 2) the correspondence of terms peculiar to a technical system to computing operations occurring in a software system is shown.

This compliance is the basis of the set of computational operations allowed in software images of technical systems.

Temporal diagrams of abstractions are organized as they are organized for the relevant elements of the technical system. For example, during the flow of coolant in the annular space of the heat exchanger, the heat transfer occurs simultaneously to the tube walls and the shell. Accordingly, the calculation of their temperatures is carried out in parallel.

In this case, the coincidence of the states of the technical and software systems in time can be ensured. This can be both a coincidence of states in real time, and, in general, their indirect coincidence in time when the real-time axis is scaled up or down for the software system.

The main feature of the execution stage of the software system is the coincidence of its states and behavior with the states and behavior of the technical system.

The equivalence of the software and technical system structures ensures the presence in the software system of only abstractions of the technical system elements and the absence of others (Fig. 3). All elements performing service calculations, e.g. centralized data storage, message and transaction managers, etc., are excluded from the software system to ensure its operation. The use of software elements performing service functions is characterized by their association with all software system elements. Their exclusion greatly simplifies the software system structure.

Service calculations cannot be completely excluded. In this case, they are distributed between abstractions. When an abstraction approaches to the technical system element according to the correspondences shown in Fig. 2, it enhances its coherence, i.e. the degree of logical completion, and the focus of all its resources on solving tasks of calculating the values determining its state. The ideal coherence is the coherence of the technical system elements.

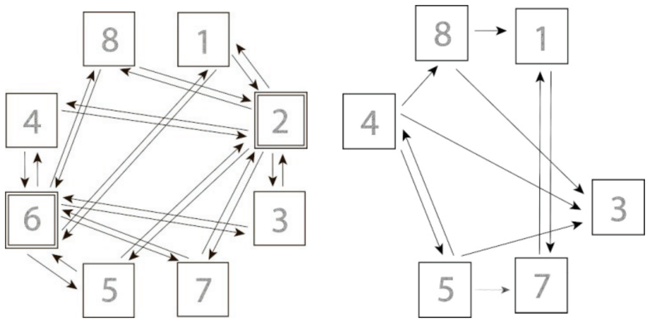

Fig. 3. Simplification of the software system structure with the exclusion of service components
(on the example of an eight-element software system with the exclusion of the service elements 6 and 2)

The total number of service calculations in the distribution of their abstractions decreases. The number of service calculations also decreases with increasing the abstraction coherence.

The equivalence of the dynamics of the software and technical systems, the organization of time diagrams for the operation of abstractions, as well as the time diagrams for the operation of the technical system elements allow to provide parallel computing (Fig. 4). It increases the efficiency of calculations and reduces the time spent on their performance.

The dynamic and structural correspondence of the software and technical systems allows to streamline and unify the information exchange between abstractions. The content of information flows in this case is defined as the parameters of the impact of the technical system element on the neighboring associated elements. Using analytical methods for calculating the state of abstractions always determines the content of information flows as a set of physical values only. The representation of information flows between abstractions according to the correspondence system shown in Fig. 2 reduces the number of message types that must be processed during the work of abstractions (Fig. 5). This allows to further improve the efficiency of the internal work of abstractions themselves, i.e. it leads to reducing the cost of computing resources for generating new abstractions, receiving and processing messages, and simplification of internal control flows.

To represent the properties of abstractions, elementary types or complex data structures are used. Elementary data represent single values describing the above listed properties of components: geometric parameters, physical characteristics, etc. If the physical

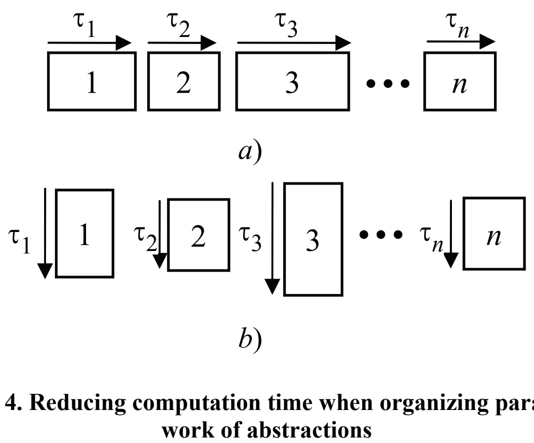

Fig. 4. Reducing computation time when organizing parallel work of abstractions

a – Consistent work of abstractions $\tau = \sum_{i=1}^{n} \tau_i$;

b – Parallel work of abstractions $\tau = \max \tau_i,\ i \in 1, ..., n$

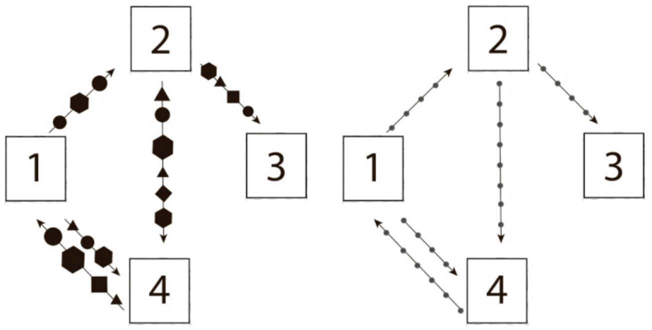

Fig. 5. Unification of the number of transmitted messages (on the example of a software system consisting of four abstractions; different types of messages are shown with different geometric shapes, and different sizes of the figures shows different sizes of messages)

characteristics of the media change, e.g. depending on the temperature, during the simulated processes, the corresponding properties of the component are represented by functions.

A single abstraction is a complex program element. Representing the technical system element, its physical and structural characteristics, it implements the mechanism for calculating the states of the corresponding element of the technical system and the processes taking place therein. The main characteristics of the technical system processes are temporal and spatial temperature and concentration fields. They are represented by complex mathematical operators, which cannot be described using only elementary data types. For their representation, program elements, which include complex data structures describing the values of constants for mathematical operators and functions implementing the calculation of their values, are used. These elements are located inside abstractions; in relation to the software system, they are local and not visible outside the abstractions in which they are implemented. In such a way the software abstractions can save detailed information about the set of their unique states in the form of fields of target characteristics, which are functions of three coordinates and time.

Applied aspects of the methodology

The applied aspects of the basic methodological principle, i.e. the structural and dynamic correspondence of the program and technical systems, are illustrated by the example of building a software system for modeling a shell-and-tube heat exchanger. The main attention is paid to illustrating the features of constructing the dynamic correspondence and justifying the choice of those computational processes that will be performed in parallel and in consistency. Two different options are given for representing the dynamics of a technical system and the corresponding options for organizing the dynamics of a software system.

As an illustration, a computer model of a one-way shell-and-tube heat exchange equipment is used. The nonlinearity of the task for calculating non-stationary heat transfer modes is caused by changing the heat exchange process of thermophysical properties. One of the ways to take it into account is representing the heat exchange zone as a set of sections where the thermophysical properties of coolants can be considered constant.

The non-stationary cell is a fragment of the apparatus enclosed between two planes perpendicular to the longitudinal axis of the apparatus. It was assumed for each cell that the thermophysical characteristics of coolants along the cell length remain constant, corresponding to the temperature at the entrance to the cell, and change abruptly during the transition to the next cell. Temperatures of coolants along the cell length become solutions of the differential equation of heat transfer by a fluid moving in the mode of ideal displacement along the channel with allowance for heat transfer by thermal conductivity [17] (Fig. 6).

The calculation includes the solution of two non-stationary tasks of thermal conductivity:

– for a hollow unlimited cylinder (the calculation of the temperature field in the walls of the apparatus tubes);

– for a double-layer unlimited cylinder (the calculation of the temperature field in the heat-insulated apparatus wall).

First, the temperature of the inner and outer walls of the heat exchanger tube is calculated. To do this, the non-stationary thermal conductivity task in the cylinder is solved. The temperature field in the wall is described by solving a differential equation [17].

According to the obtained wall temperatures, the density of the heat flow from the hot coolant to the wall and from the wall to the cold coolant is calculated. Next, the amount of heat which hot and cold coolants have received is calculated. According to the known initial temperature of the coolants in the cell and the amount of heat which the coolant has received in the cell, it is possible to calculate the coolant temperature at the cell outlet.

The cell length is taken such that the temperature change along its length is sufficiently small, which entails a sufficiently small change in the thermophysical properties. Getting the temperature distribution in the apparatus involves the calculation of all the cells that make up the apparatus.

The way to solve the task is determined by the adopted system of assumptions, i.e. which processes are considered, and from what point of view they are considered. Below there are two different calculation options based on different views of the processes occurring in the coolant. Herewith, these options clearly illustrate the features of the considered method for designing software systems.

The structural conformity of the software and technical systems for these two described variants remains unchanged. These calculation variants differ in the ways of considering the dynamics of the technical system and, accordingly, their representation in the software system and calculations. The structure of the software system is shown in Fig. 7.

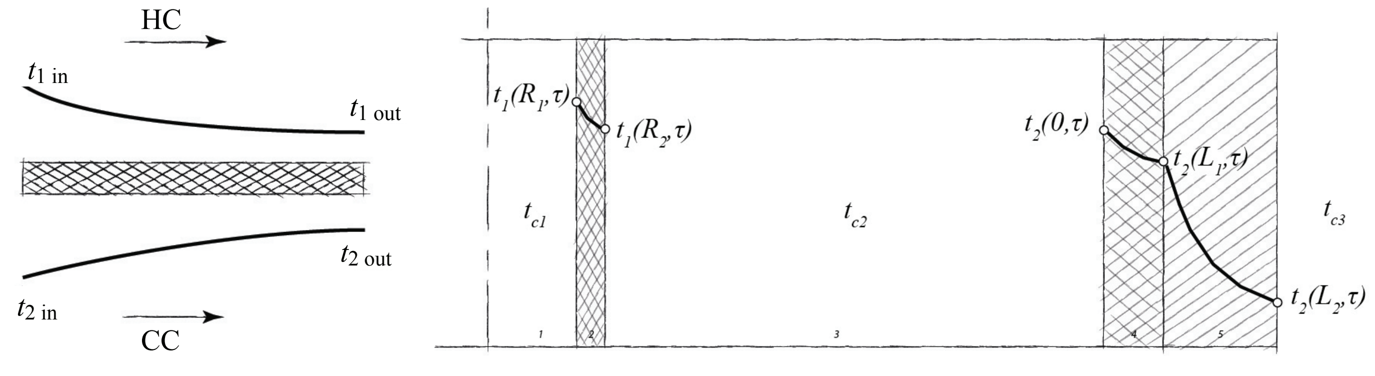

Fig. 6. Thermal scheme of a non-stationary cell:
HC – hot coolant; CC – cold coolant;
1 – tube space; *2* – tube wall; *3* – annular space; *4*, *5* – housing wall with an insulation layer; *6* – environment;
t_{c1} – hot coolant temperature; t_{c2} – cold coolant temperature; t_{c3} – ambient temperature

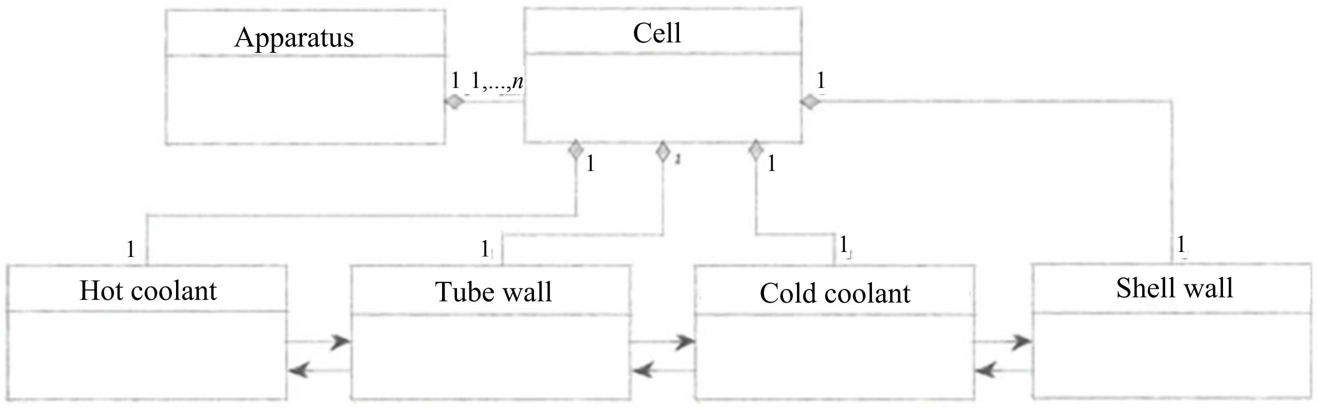

Fig. 7. Software system structure

The first option is based on the sequential calculation of all cells (Fig. 8). Here the calculation process is compared to the movement process of the hot coolant and the successive change in its temperature along the movement direction. For the forward flow, when the temperatures of the two coolants in the cell are known, it begins with the first one, into which the hot coolant enters and continues along its movement to the last cell. In the case of reverse flow, coolants are fed from opposite ends of the apparatus and, accordingly, the temperature of one coolant is known in the first cell, while the other – in the last cell. In such a situation, the above condition for the temperature awareness is not fulfilled for two coolants in the cell simultaneously, and the start of the sequential calculation of all cells is impossible. Therefore, an iterative algorithm is used. The calculation is carried out for a fixed time interval. For the next interval, another hot coolant inlet temperature is taken and the calculation is repeated.

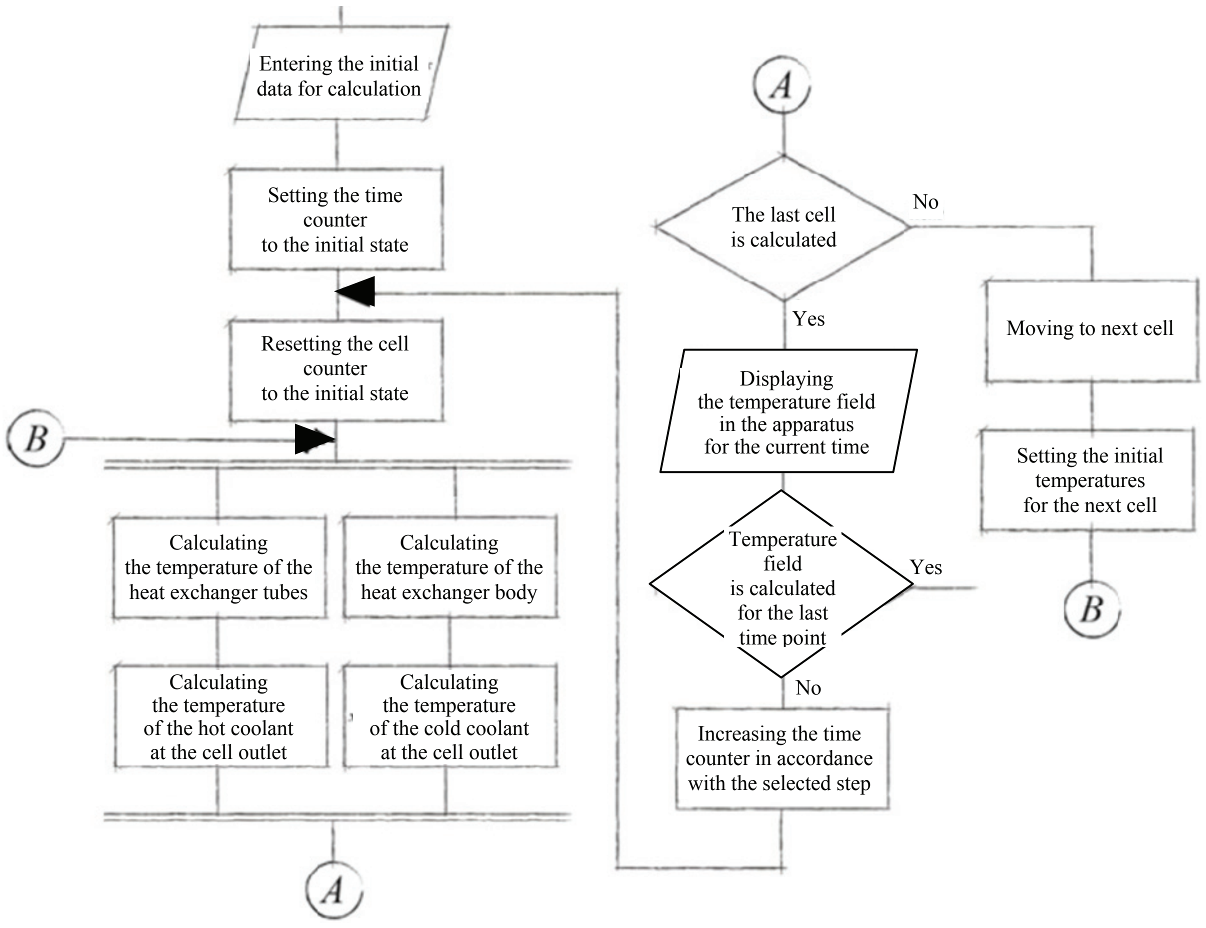
Entering the initial data for calculation
Setting the time counter to the initial state
Resetting the cell counter to the initial state
B
Calculating the temperature of the heat exchanger tubes
Calculating the temperature of the heat exchanger body
Calculating the temperature of the hot coolant at the cell outlet
Calculating the temperature of the cold coolant at the cell outlet
A
A
The last cell is calculated
No
Yes
Moving to next cell
Displaying the temperature field in the apparatus for the current time
Setting the initial temperatures for the next cell
Temperature field is calculated for the last time point
Yes
B
No
Increasing the time counter in accordance with the selected step

Fig. 8. Algorithm for calculating a non-stationary temperature field in the apparatus with sequential cell calculation

The second option considers the heat exchange process as a combination of two independent actions separated in time – the movement of coolants and the calculation of their temperatures in each cell. These two actions in the heat exchanger occur simultaneously, while in the software system they are separated in time. First, the coolants are moved, then they are positioned in space, and finally the temperatures are calculated. Here, the process of calculating temperatures is compared to the process of changing the coolant temperature over the entire length of the process area in the absence of coolant movement during the time interval when the calculation is carried out. After the coolant temperatures have been calculated, the transition to the next time interval is performed. This calculation option does not require various algorithms for the forward and reverse flow calculation. The calculation of these two options is implemented by a universal algorithm. This is achieved by separating the processes of moving coolants and heat exchangers in time (Fig. 9).

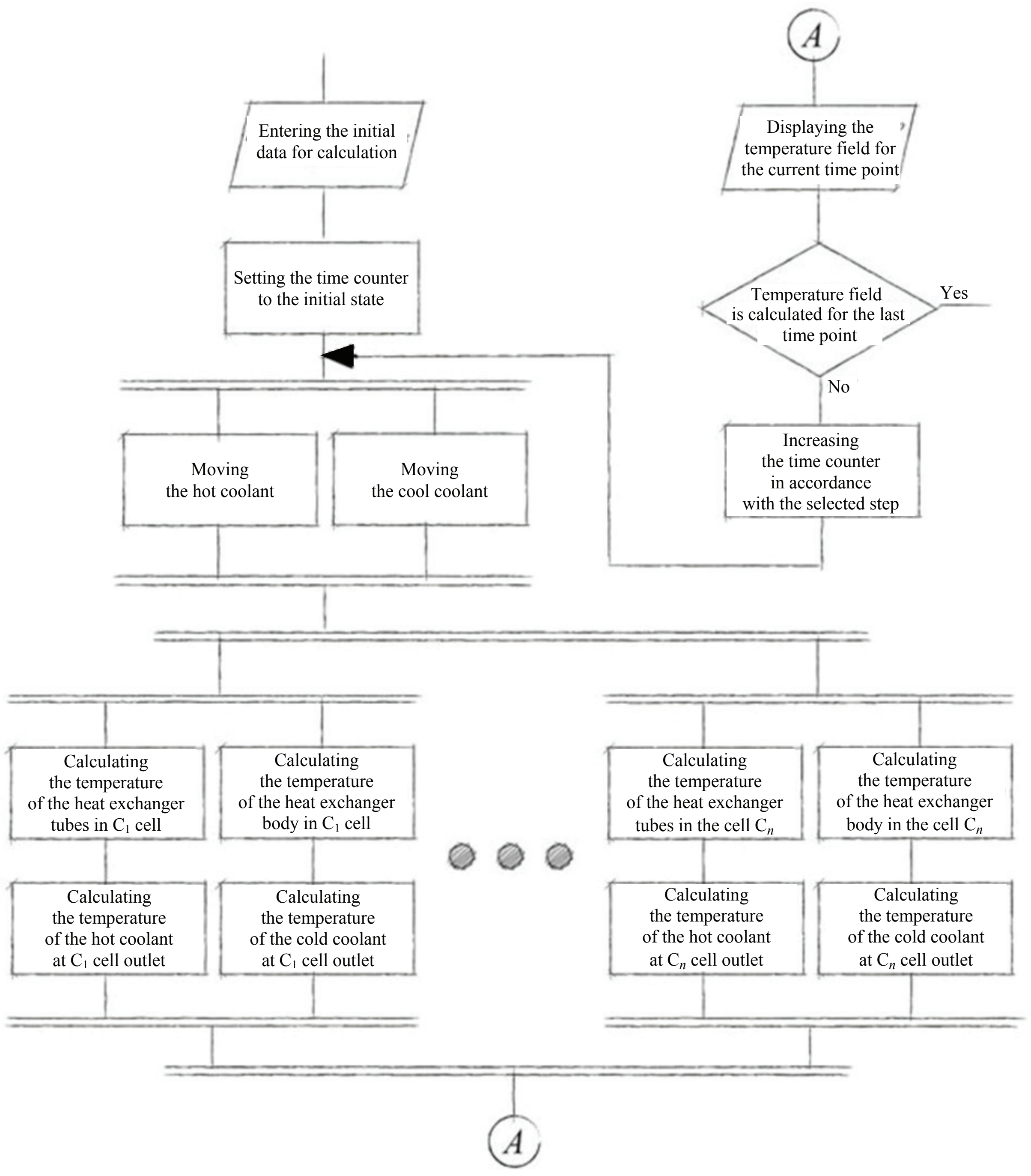
A
Entering the initial data for calculation
Setting the time counter to the initial state
Moving the hot coolant
Moving the cool coolant
Displaying the temperature field for the current time point
Temperature field is calculated for the last time point
Yes
No
Increasing the time counter in accordance with the selected step
Calculating the temperature of the heat exchanger tubes in C_1 cell
Calculating the temperature of the heat exchanger body in C_1 cell
Calculating the temperature of the hot coolant at C_1 cell outlet
Calculating the temperature of the cold coolant at C_1 cell outlet
Calculating the temperature of the heat exchanger tubes in the cell C_n
Calculating the temperature of the heat exchanger body in the cell C_n
Calculating the temperature of the hot coolant at C_n cell outlet
Calculating the temperature of the cold coolant at C_n cell outlet
A

Fig. 9. Algorithm for calculating the temperature field in the apparatus with the parallel cell calculation

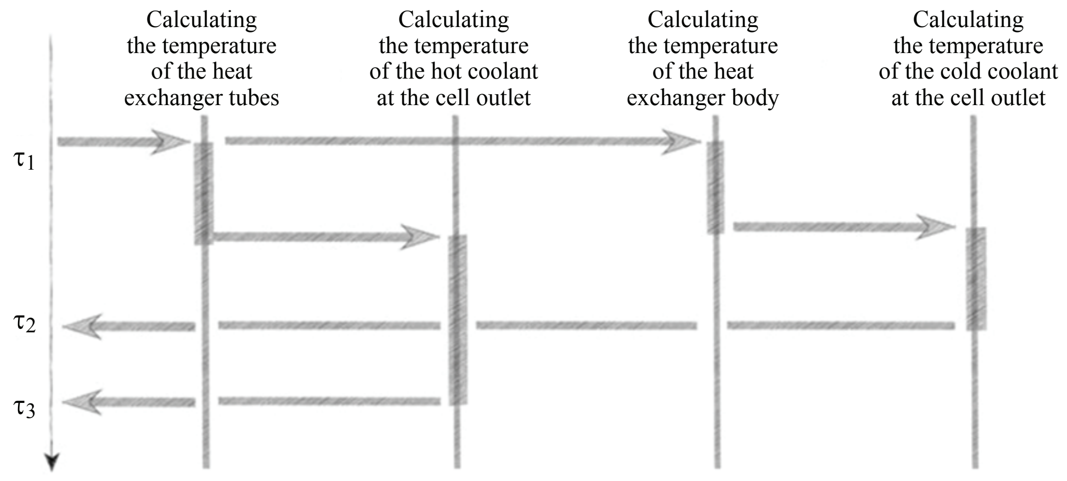
Calculating the temperature of the heat exchanger tubes
Calculating the temperature of the hot coolant at the cell outlet
Calculating the temperature of the heat exchanger body
Calculating the temperature of the cold coolant at the cell outlet
τ1
τ2
τ3

Fig. 10. Time line of calculating a non-stationary cell

The algorithm for calculating a single cell in the first and second variants is implemented in parallel, in accordance with the processes occurring in it. Fig. 10 shows this algorithm in the time line.

The two shown options fully implement the principle of the considered method on the dynamic correspondence of the program and technical systems. The first calculation option is based on the representation of the process when the coolant temperature changes sequentially as it moves through the apparatus. Accordingly, the coolant temperature in the cells is calculated sequentially. Parallel temperature calculation in all cells in this case is impossible.

The second option is based on the idea of placing coolants in the heat exchanger so that the heat exchange process takes place simultaneously along the entire length of the process area. The initial temperatures of coolants in each cell are known. Herewith, the calculation of cells is organized in parallel.

Fig. 7 and 8 show that the number of conditional statements, branching and transitions decrease as the degree of parallelism increases. It results in a reduction in the number of decisions that need to be made during the program execution.

Conclusion

The developed methodology makes it possible to build software systems in terms of a technical system, the area of its use and purpose, as well as to take into account the peculiarities of the software system functioning at the structural level. Besides, the generated abstractions and semantics based on the objects of the technical complex and the subject area allow developers and, especially, users to perceive themselves as directly working with the subject area elements. This condition determines the main advantages of the considered methodology, i.e. improving the efficiency of the application development process, the possibility of involving experts from the applied area into the development process, improving the efficiency of the software system and computing, the ability to build applications of a wide range of uses, e.g. desktop, distributed and embedded systems, web services, mobile applications, etc.

In the course of the performed calculations, the algorithm for solving a general mathematical problem is not used directly; instead, the script of the software system based on the functioning of its individual components and the connections between them is implemented. Each component separately solves a local math problem describing the state of an individual element of a technical system.

Further development of the methodology is possible in the direction of increasing the degree of the design process formalization and the software systems operation to a level where it is possible to use automated development tools, rapid design and prototyping. With such a degree of the design process formalization, it is possible to develop a specialized problem-oriented programming language to create software abstractions of technical systems.

Acknowledgments

The work was supported by the Ministry of Education and Science of the Russian Federation in the framework of the project part (Project 8.2906.2017/PCh).

References

1. Stolarski T., Nakasone Y., Yoshimoto S. *Engineering Analysis with ANSYS Software.* Butterworth-Heinemann, 2018.

2. Shih R. *Introduction to Finite Element Analysis Using Solid Works Simulation*. SDC publications, 2014.

3. Systèmes D. ABAQUS Unified FEA: Complete Solutions for Realistic Simulation. *Dassault Systemes*, 2014. Available at: https://www.3ds.com/productsservices/simulia/products/abaqus/ (accessed 9 March 2018).

4.Tonks M., Gaston D., Millett P., Andrs D., Talbot P. An Object-oriented Finite Element Framework for Multiphysics Phase Field Simulations. *Comp. Mat. Sci.*, 2012. vol. 51 (1), pp. 20–29. doi:10.1016/j.commatsci.2011.07.028.

5. Prof. Sham Tickoo Purdue Univ. *Autodesk Simulation Mechanical 2016 for Designers*. CADCIM Technologies, 2015.

6. Giełżecki J., Jakubowski T. The Simulation of Temperature Distribution in a Ground Heat Exchanger – GHE Usingthe Autodesk CFD Simulation Program. *Renewable Energy Sources: Engineering, Technology, Innovation*. Springer, Cham, 2018, pp. 333-343.

7. *Finite Element Analysis Software (Fea Software)*. Available at: www.autodesk.com/solutions/finite-element-analysis (accessed 9 March 2018).

8. Madenci E., Guven I. *The Finite Element Method and Applications in Engineering Using ANSYS*®. Springer, 2015.

9. Dubbeldam D., et al. RASPA: Molecular Simulation Software for Adsorption and Diffusion in Flexible Nanoporous Materials. *Molecular Simulation,* 2016, vol. 42.2, pp. 81-101.

10. Borgenstam A., et al. DICTRA, a Tool for Simulation of Diffusional Transformations in Alloys. *Journal of Phase Equilibria*, 2000, vol. 21, issue 3, pp. 269.

11. Sinha S., Chandel S.S. Review of Software Tools for Hybrid Renewable Energy Systems. *Renewable and Sustainable Energy Reviews*, 2014, vol. 32, pp. 192-205.

12. Choudhari C.M., Narkhede B.E., Mahajan S.K. Casting Design and Simulation of Cover Plate Using AutoCAST-X Software for Defect Minimization with Experimental Validation. *Procedia Materials Science*, 2014, vol. 6, pp. 786-797.

13. Rabold F., Kuna M. Automated finite element simulation of Fatigue Crack Growth in Three-dimensional Structures with the Software System ProCrackProcedia *Materials Science*, 2014, vol. 3, pp. 1099-1104.

14. Agapito G., Puglisi A., Esposito S. PASSATA: Object Oriented Numerical Simulation Software for Adaptive Optics. *Adaptive Optics Systems V. International Society for Optics and Photonics*, 2016, vol. 9909, pp. 99097E.

15. Pal P., et al. A Visual Basic Simulation Software Tool for Performance Analysis of a Membrane-based Advanced Water Treatment Plant. *Environmental Science and Pollution Research*, 2014, vol. 21, issue 3, pp. 1833-1849.

16. Whitaker E.T. Agent-based Simulation. *Modeling and Simulation in the Systems Engineering Life Cycle*. Springer, London, 2015, pp. 139-155.

17. Tugolukov E.N. Reshenie zadach teplo-provodnosti metodom konechnyh integral'nyh preobrazovanij [The solution of heat conductivity problems by the method of finite integral transformations]. Tambov, 2005, 116 p. (Rus)