UDC 004.415.2:519.876.5

Aleksandr Dorensky
Kirovograd National Technical University

Method of the Models' Synthesis for Software Automated System Objects' States in Digital Images Processing

This article proposes a method for the synthesis of the behavior of software objects models (SOM) for the developed object-oriented software systems for automated digital image processing in order to avoid systemic and algorithmic errors in the design phase of a software system, as well as to reduce the time of its development. The process of constructing the SOM proceeding from its finite-state representation is viewed from the standpoint of abstract synthesis of an automata's finite state. Thus, the specialties of the synthesis of finite automaton SOM, the construction of the map defining a plurality of channels management class of objects, the order to bring it to an automata, the construction of the canonical set of events and their regular expressions to display defining a plurality of channels management of software objects class for object oriented software system are considered and justified.
software, a automated system, instance of the class, object-oriented model, finite automaton model, the behavior of the object, an object's state

А.П. Доренский, препод.
Кировоградский национальный технический университет
Метод синтеза моделей состояний объектов программного обеспечения автоматизированной системы обработки цифровых изображений

В статье предложен метод синтеза моделей поведения программных объектов (МСО) разрабатываемого объектно-ориентированного программного обеспечения системы автоматизированной обработки цифровых изображений с целью избежания системных и алгоритмических ошибок на этапе проектирования программной системы, а также сокращения времени разработки. Процесс построения МСО, исходя из ее конечно-автоматного представления, рассматривается с точки зрения абстрактного синтеза конечного автомата. Таким образом, изложены и обоснованы особенности синтеза конечно-автоматной МСО, построение отображения, определяющего множество каналов управления класса программных объектов, порядок приведения его к автоматному виду, построение канонического множества событий и их регулярных выражений для отображения, определяющего множество каналов управления класса программных объектов объектно-ориентированной программной системы.
программное обеспечение, автоматизированная система, экземпляр класса, объектно-ориентированная модель, конечно-автоматная модель, поведение объекта, состояние объекта

Introduction. Massive amounts of data circulate in modern information and telecommunication systems. The vast majority of the processed and transmitted information is graphic: digital images and videos, which are characterized by extra-large volumes. The volumes of this kind of data are constantly increasing. This leads to an overload of data channels, and as a consequence, to a significant increase in the time of information delivery [1]. Therefore, currently, not only the increase of the capacity of modern means of communication, but also the development of a software system to automatic processing of digital images for the purpose of compact representation (compression) is important.

Statement of the problem. Software (SW) of the automated digital imaging system (ADIS) must be reliable, high-performing, flexible, and should provide an opportunity for improving, scaling, and upbuilding its functionality [1, 2].

It is possible to ensure these requirements and characteristics through the use of object-oriented approach (OOA) [2]. Unlike the traditional approach to development [3], the OOA makes emphasis on both the information and the software objects' behavior. This leads to creating flexible SS that allow change of behavior and/or information.

However, even the use of the OOA does not prevent errors in the software. And the most important are the errors of software objects' interaction, most of which occur at the design stage [4]. The testing of models is carried out to identify them in the early stages of the life cycle of the system. Sufficient attention is given to the development of tests in the literature, while the creation of test models is not actually considered.

Thus, for the design of object-oriented software (OOSW) ADIS is necessary to provide automated conversion of object-oriented software models in the transition from one model to another. This lets testing them in terms of software models' behavior analysis. Thus, a scientific and technical problem of the synthesis of models states of the instances of classes OOSW ADIS is important.

Analysis of recent research and publications [1-2, 5-9] has shown that in OOSW designing an object decomposition and the techniques of presentation of logic (structure of classes and program objects) and physical (architecture of modules and processes), as well as of static and dynamic models of object-oriented software are used. Models are developed on the basis of objects and phenomena of the real world; describe the behavior of object-oriented software, developed within the domain objects associated with the states of objects.

In [5], for software development using UML an object model is built, which reflects the basic abstractions of a domain, the variants of the software usage, its physical representation, as well as the information flows that function in the software. In [6], the object models of software systems built using UML are proposed. However the models that reflect the basic abstraction domain and the physical representation of the software are built. In fact, they are a set of models that characterize certain aspects of the software system. In turn, it does not provide an integrated behavior modeling of the software objects. In [7], a specification of the program management system, consisting of a logic model, patterns of use, implementation, processes, and deployment is performed. Each of these models describes a certain aspect of the system, and all together they make up a relatively complete model of the developed software. In [8], an object-oriented modeling approach of software systems is described, it proposes and describes a three-tier architecture of a conceptual model that reflects the development of reusable components of different nature (structures of domain models). In [9], the strategy of the synthesis of object-oriented models in terms of object-oriented approach is used for the synthesis of the complex structure of the object-oriented software, which consists of a static and dynamic component. Thus, the article proposes the construction of structures of both static and dynamic components of the complex model OOSW within the collective and individual behavior of the main conceptual units.

The wording of Article goals. Based on the analysis of recent research and publications it can be concluded that not enough attention is paid to the issue of constructing and using of the behavior models of the software objects. Thus, in developing of the software of ADIS, a problem of developing of models of instances of classes of the developed OOSW should be solved. *The purpose of the research* is to develop a method for the synthesis of a state of the models of software objects of OOSW automated system for processing of digital images that will provide: 1) the formalization of the process of determining the conditions and their relationships in the life cycle of an instance of OOSW; 2) reducing the complexity of the development of the dynamic component of an integrated model (IM) of OOSW [4, 9] in the process of the development of a software system at the logical level.

The main part. At the initial stage of synthesis of the structure of the dynamic

components of IM OOSW [4] while developing ADIS, the process of development of the states of software objects model (SOM) of OOSW based on its finite-state representation, viewed from the standpoint of abstract synthesis of a finite state automata [9]. As a result, it is necessary to get a formalized SOM provided with a marked transition table, on which a corresponding graph of a state transition to the Moore's initial partial finite automaton is built.

Let's give a precise statement of the problem of the SOM synthesis according to the method of synthesis of finite automata by alphabetical mappings they implement: for any class of objects of OOSW $sc \in SC$ is required by the injectivity mapping F_{SteSch} of the set of words over a finite alphabet of events in this class into a set of words (sequences of activities) over a finite alphabet of its activities, i.e. on multiple channels of the class management [4], to construct a finite-state model of behavior instances, given by the marked transition table.

Thus, the algorithm of the method consists of interconnected components:

– SOM formalized representation in terms of the Moore's initial finite automata [9];

– procedures of formalizing of a plurality of the class control channels [4] as the alphabetical display;

– of adapted method of the abstract synthesis of the Moore's finite automata by the implemented alphabetical mappings.

The procedure of the formalization of multiple channels of the class management in the form of alphabetical display includes the following stages:

1. Construction of the set of tracing of sequences of events the object class (TSEC) [9] STE_{sc} for $sc \in SC$ in the alphabet of events SE_{sc}: abstraction of objects involved in all models of message sequences of objects [4, 9], in the form of the corresponding classes of OOSW; abstracting of receiving messages by the objects in the form of the corresponding classes of events of OOSW.

2. Representation of every TSEC $ste \in STE_{sc}$ by a regular setting which can be written as a product of elementary events in the input alphabet SE_{sc} the events of the class $sc \in SC$ of OOSW: $rv_{ste} = se_{i1}se_{i2}...se_{im}$.

3. Presentation of the result of execution by the class instances of any sequence of activities $sch \in SCH_{sc}$ in the alphabet SH_{sc}, performed in response to a TSEC from the plurality STE_{sc}, as an injective mapping F_{SteSch}: $STE_{sc} \to SCH_{sc}$ (multiple channels of the class management class [4]).

Performing the first step of the adapted method of an abstract synthesis of finite automata is that for each class $sc \in SC$ of OOSW, a mapping $F_{SteSch} : STE_{sc} \to SCH_{sc}$, which defines a set of control channels, is transformed as follows:

1. Written as correspondence table where each row has the following form: $ste \to F_{SteSch}(ste)$ or $ste \to sch$, where $ste \in STE_{sc}$ and $sch \in SCH_{sc}$.

2. Leads to an automaton, i.e. mapping F_{SteSch} is resulted in the display of the form F_{SteSch}, satisfying the four conditions of automaton.

If any alphabetic mapping satisfies the four conditions of the mapping automaton, the Moor's finite automaton inducing this mapping can be constructed [10]. As F_{SteSch} represents the mapping, it satisfies the first condition of the automaton by determination. The mapping F_{SteSch} satisfies the third condition of automaton based on the way of its construction during the conceptual analysis and determination of this surjective mapping. Thus the mapping F_{SteSch} does not satisfy the second and the fourth conditions of automation, as generally the domain of definition F_{SteSch} does not contain all the initial segments of all TSEC

and respectively, does not translate in the initial segments of TSEC into corresponding initial segments of activities [9].

Thus, in order to develop a finite-state model of behavior instances OOSW, which would induce a mapping F_{SteSch}, it is necessary to implement mapping cast F_{SteSch} to the automata form. As the mapping F_{SteSch} satisfies the third condition [10], then it is sufficient to apply to it the operation of completion only. However, in this case, the mapping F_{SteSch} in some cases, it may become ambiguous, since one and the same word may be included in an initial interval into several different words from the definition of the mapping F_{SteSch}.

In the case when the mapping is derived from an unambiguous alphabetical mapping as the result of a standard operation of the word length alignment, the completion of this mapping is unique and is an automaton mapping [10]. Thus, with respect to the mapping of a set of words over a finite alphabet of the event class in the set of words over a finite alphabet of OOSW class will perform the operation of the alignment of words length and operation of completion of this mapping.

The algorithm of performing of the standard operation of the alignment of words lengths is as follows:

1. To the input alphabet of the mapping F_{SteSch}, i.e. the multitude of events SE_{sc} of the class $sc \in SC$, a new event is added, which will be considered an empty one. Let's define this event through se^0.

2. To the output alphabet of the mapping F_{SteSch}, i.e. the multitude of events SH_{sc} of the class $sc \in SC$ a new event is added, which will be considered an empty one. Let's define this event through sh^0.

3. For each sequence of the class events or the tracing scenario $ste \in STE_{sc}$, n_{ste} of empty events se^0 is attributed to the right, thus n_{ste} is assumed equal to the word length (sequence of the class activity) $sch = F_{SteSch}(ste)$.

4. Each sequence of the class activities $sch \in SCH_{sc}$, m_{sch} of empty activities sh^0 is attributed to the left, thus m_{sch} is assumed to be equal to the word length (sequence of the class activity) ste.

5. A new mapping F_{SteSch}' of the multitude STE_{sc}' is built for the words in the alphabet $SE_{sc} \cup (se^0)$ into the multitude SCH_{sc}' of the words in the alphabet $SH_{sc} \cup (sh^0)$, which translates ste' and sch' into one another, the received as the result lengths of the words ste and sch respectively: $F_{SteSch}'(ste') = sch'$.

Thus the mapping F_{SteSch} is uniquely recovered as a result of the standard operation of alignment of word lengths mapping F_{SteSch}'. It can be assumed that a zero aligning operation is applied, in which no empty appending letters happens.

The next operation is the operation of replenishment. It may be applied only to the aligned mapping F_{SteSch}'. The essence of the operation of replenishment is to spread the mapping on the initial segments of words: if δ is the arbitrary initial segment of any word $ste' \in STE_{sc}'$, then $F_{SteSch}'(\delta)$ is equal to the initial segment of a word $F_{SteSch}'(ste')$, which has the length equal to the initial segment δ. As a result of applying the operation to replenishment to the aligned mapping F_{SteSch}' we will get a new mapping F_{SteSch}'', the domain STE_{sc}'' which satisfies the completeness.

As a result of the two operations described above, to the mapping of the multitude of

words over a finite alphabet of class event in the multitude of words over a finite alphabet of OOSW class of activities, and then the operation of replenishment, we get the automaton mapping F_{SteSch}'', satisfying all the conditions of automaton, i.e. the mapping F^{A}_{SteSch}.

In general, the result of the first stage of the adapted method of abstract synthesis of finite automaton model of behavior of OOSW instances is the abbreviated table of correspondence automata display F^{A}_{SteSch}, into which an initial mapping F_{SteSch} has been transformed. A condensed correspondence table is called a correspondence table in which no input word is an initial segment of any other input word. When an automaton mapping F^{A}_{SteSch} is set by the condensed table, its extension to the initial segments of the input words that are included in the table is based on the fourth condition of automaton. Thus, based on the condensed lookup table the full correspondence table can always be recovered.

Performing the second stage of the adapted method of abstract synthesis of SOM of OOSW is to define a canonical set of multiple events corresponding to the mapping F^{A}_{SteSch}.

Let's define the canonic multitude of the events correspondent to the mapping F^{A}_{SteSch} through SFE^{A}. The event $SFE_{sh} \in SFE^{A}$ correspondent to the events of the class of OOSW sh may be developed following the next algorithm:

1. During the analysis of the condensed table of the compliance of the mapping F^{A}_{SteSch}, a multitude $\{\eta_j\}$ of all the initial segments of output words ending with the letter sh is defined.

2. The multitude $\{\delta_i\}$ of the initial segments of input words of the mapping F^{A}_{SteSch}, uniquely relevant to the elements of the multitude $\{\eta_j\}$, and will define the event $SFE_{sh} \in SFE^{A}$, i.e. $\forall SFE_{sh} \in SFE^{A}\left(\forall \delta \in SFE_{sh} \exists \eta = F^{A}_{SteSch}(\delta) = (\eta = (\eta_i sh))\right)$, where $SFE_{sh} \subseteq STE_{sh}$''; $\eta, \eta_i \in SCH_{sc}$'' and $\delta \in STE_{sc}$''.

At this stage the control of the first phase of operations may be also performed, i.e. the mapping F^{A}_{SteSch} is automated only when the point events are disjoint. Thus, the multitude SFE^{A} must satisfy the conditions of automation [10], according to formulas (1) and (2):

$$\left(SFE^{A} = \{SFE_i\}\right) \& \left(\forall SFE_i \in SFE^{A}\left(\forall ste \in SFE_i, ste = se_{s1}, se_{s2} \ldots, se_{sl(ste)}\left(\{se_{sc}\} \subseteq SE_{sc}\right)\right)\right) \&$$
$$\left(\forall SFE_i, SFE_j \in SFE^{A}, i \neq j, SFE_i \cap SFE_j = \varnothing\right) \& \left(\forall SFE_i \in SFE^{A}, SFE_i \cap \{ste^0\} = \varnothing\right), \quad (1)$$

where SFE^{A} – automata multitude of events in the input alphabet SE_{sc}; $s = 1,2,\ldots,n$; n – cardinality of the multitude SE_{sc}; i, j $= 1,2,\ldots,$ m; m – cardinality of the multitude SFE^{A}; $l(ste)$ – the length of a word ste.

That is, automata multitude of events SFE^{A} is made of the finite number of events SFE_j in one and the same input alphabet SE_{sc}, which are disjoint and do not contain the empty word ste^0:

$$\forall SFE_i \in SFE^{A}\left(\forall ste \in SFE_i, \text{i} \neq \text{j}, \left(\left(\forall \delta_{\text{ste}} = e_{sl} \ldots e_{sl(\delta ste)}\left(ste = e_{sl} \ldots e_{sl(\delta ste)} e_{s(l(\delta ste)+1)} \ldots e_{sl(ste)}\right)\right.\right.\right.$$
$$\left.\left.\left. \exists SFE_j \in SFE^{A}\left(\delta_{ste} \in SFE_j\right)\right) \& \left(\forall SFE_j \in SFE^{A}, SFE_j \cap \{\delta^0\} = \varnothing\right)\right)\right), \quad (2)$$

where $l(\delta_{ste})$ – is the length of the initial segment δ_{ste} of the word ste.

Thus, the result of the second phase is the canonic multitude of the events

$SFE^A = \{SFE_{sh}\}$ of the mapping F^A_{SteSch}.

Performing the third stage of the adapted method of the abstract synthesis of SOM of OOSW is to find the simplest possible regular expressions for each event found in the previous stage of the event $SFE_{sh} \in SFE^A$.

At this stage, as a rule, the identity transformations in the algebra of events are used, as well as the following method of automatic events' extension: the construction of the simplest regular expression for an automata event, an arbitrary set of forbidden words can be added to it.

All the words of any input alphabet are broken by the automaton conformity into two classes: the class of valid and the class of forbidden words. There are several ways to specify the multitudes of valid and forbidden words [10]. Valid will be assumed every word contained in only one of the set of regular events, and such that each of its non-empty initial segment is also contained in exactly the same of the specified events. All the other words, as well as the words for some of the letters of which the corresponding output letters are not defined, constitute the multitude of forbidden words. The totality of all the forbidden words for this automaton conformity is a domain ban.

When the domain of the original automaton mapping F^A_{SteSch} is ultimate, it is sufficient to use only identical transformations. In this regular expression found for the previous stage events in the most general case, to perform identical transformations represent the disjunction of the initial segments of sequences of events for OOSW classes, i.e. the disjunction singleton non-elementary events, which in turn are represented as a product of a finite alphabet letters in the events of OOSW class: $SER_{sh} = \bigvee_{j=1}^{n} \delta_j$, where SER_{sh} is the regular expression to automaton events $SFE_{sh} \in SFE^A$, and n is the power of automaton event SFE_{sh}.

Thus, the result of the third stage is the multitude of all built regular expressions $SER^A = \{SER_{sh}\}$ for the found on the second stage canonic multitude of events $SFE^A = \{SFE_{sh}\}$.

Performing the fourth stage of the adapted abstract synthesis method automaton model of behavior instances of OOSW class is its construction over the canonical multitude of events SFE^A, given by the appropriate set of regular expressions from the multitude SER^A.

There exists a single constructive technique that allows for any finite multitude of regular events, given their regular expressions to build the Moore's final automaton. Thus the number of various output signals does not exceed 2^n, and the number of states - not more than 2^{m+1}, where n is the general number of regular expressions, m – general number of the letters of input alphabet (including repetitions), which are included into defined regular expressions [10].

The synthesis process can be carried out in two main variants: with a natural area with the exception of the prohibition or with the exception one of the events. Let's use the first variant of the synthesis automaton model of the instances behavior of OOSW following the canonical multitude of events. This way the prohibition job coincides with the method described in the previous step. Thus, the synthesis algorithm of finite automaton model of behavior instances of OOSW by the canonical set of events SFE^A has the following form.

Step 1. The given multitude of regular events $SER^A = \{SER_{sh}\}$ is performed by the multitude $SER^A{}' = \{SER_{sh}{}'\}$ of written correctly regular expressions. The condition of the correct record of regular expressions is that all the original regular expression of the multitude SER^A, which are polynomials, are enclosed in brackets.

Step 2. In the multitude SER^{A}' of properly recorded regular expressions there produced a markup of places who share the signs of these expressions. The signs of any expressions SER_{sh}' are the letters of the alphabet SE_{sc}, the symbol of an empty word se^{0}, the disjunction sign and brackets. In this case let's consider three sets of locations:

– multitude of sharing locations $SPE_R = (spe_l)$, where every location spe_l divides any two signs of any regular expression SER_{sh}', $l = 1,...,r$ and r – the power of the multitude SPE_R;

– multitude of the initial locations $SPE_N = \{spe_j\}$, where every location spe_j is placed to the left of the very left sign corresponding to it expression SER_{sh}', $j = 1,...,n$ and n – the power of the multitude SPE_N, equal to the power of SER^{A}';

– multitude of the finite locations $SPE_C = \{spe_i\}$, where every spe_i is placed to the left of the very right sign corresponding to it expression SER_{sh}', $j = 1,...,c$ and c – the power of the multitude SPE_C, equal to the power of SER^{A}'.

Thus for synthesis algorithm basic concepts of the main and pre-main locations are also very important. The main location is any location just to the left of which is the letter in the basic alphabet, as well as any initial place [10]. Let's denote the multitude of all the main locations of the multitude of regular expressions SER^{A}' through SPE_O, where $SPE_O \subseteq SPE_R \cup SPE_N \cup SPE_C$ and $SPE_O \cap SPE_N = SPE_N$.

The pre-main location spe is any location just to the right of which there is a letter of the main alphabet. Let's denote the multitude of all the pre-main locations of the multitude of the regular expressions SER^{A}' through SPE_P, where $SPE_P \subseteq SPE_R \cup SPE_N$. And the multitudes SPE_P and SPE_O may intersect.

Step 3. Every main location $spe \in SPE_O$ is attributed as and index a non-negative integer, with all initial locations are attributed to the same index 0. All other locations are numbered randomly with natural numbers 1, 2, ... This process may be formalized as the mapping $F_{SpeN} : SPE_O \to N$, where N is a multitude of natural numbers. At the same time the introduced indices $F_{SpeN}(spe)$ will be called the main indices.

Step 4. To the indexed as a result of the third step of the multitude of regular expressions SER^{A}' an operation of identification of the respective locations and identification operation of similar locations is applied consistently, step by step.

Step 5. Every main index $F_{SpeN}(spe)$ of every main location $spe \in SPE_O$ distributed as a minor index of all places (both core and non-core) that are subject to spe, but different from it. In addition, each subordinate position can receive a set of non-core indices.

Step 6. The construction of the required transition table of automaton model of behavior instances of OOSW is carried out. Wherein the input signals are the letters of the initial alphabet SE_{sc}, and the states are taken as in the general case of all subsets of a multitude of all the major indices. In this subset consisting of the major indexes $i_1,...,i_n$, where $n \geq 1$, will be denoted through the disjunction of indices $i_1 \vee ... \vee i_n$, and the empty multitude of the main indices will be denoted by the star (empty state).

An algorithm for constructing the transition table of the automata is as follows:

1. Table rows are designated by letters of the input alphabet in a given multitude of events.

2. The initial state is 0, and with the column corresponding to this state, the transition table construction begins.

3. The columns corresponding to the remaining states are issued only after designating their status has already appeared in the columns of the table discharged earlier.

4. At the intersection of the line se_i and the row ss_j of the table the state (many of the major indexes) is discharged, consisting of the main indices of all the main locations separated by the event se_i from the immediately preceding them pre-main locations, including which indexes (both core and non-core) there is at least one index belonging to the state ss_j. In the case of non-existence of the main locations with the required properties at the appropriate location in the table an empty state is issued.

Step 7. The marked transition table of the desired finite automaton model of behavior instances of OOSW is built. Every of the states $i_1 \vee ... \vee i_n$, where $n \geq 1$, which mark the columns in the transitions table, and is indicated by the subset $\left(SER_{sh_i}',...,SER_{sh_m}\right) \subseteq SER^A{}'$ of all the symbols of those and only those regular expressions, the finite locations of which contain among their indices (both core and non-core) at least one of the indices $i_1,...,i_n$. The empty state is denoted by the empty multitude of regular expressions.

Step 8. For a given area of prohibition or the multitude of admissible words of finite automaton model of behavior instances of the class are uncertain states of the model. Introducing the uncertainty of essence is marked in the transition table is as follows:

1. There stand uncertain outputs of finite automaton model of behavior instances of the class. Uncertain, according to the prohibition imposed by the output signals are all composed of two or more events, as well as an empty output signal.

2. Undefined states of the finite automaton model of behavior instances of the class are allocated. Undefined are all states that are marked with undefined output signals. Since the initial state is ranked as one of the elements in the transition table, it is not considered as undefined.

3. All occurrences of undefined states in the marked transition table automaton model of behavior instances are replaced by dashes, and certain of these states are excluded from the table columns.

4. In case of unreachable states, the indicated by them columns are also deleted (SOM performed minimization of OOSW).

Step 9. The redesignation of the output signals and synthesized finite-state automaton model signals and states are carried out. The essence of the execution is as follows: 1) Each output signal is designated by the relevant activity SFE_{sh}; 2) Every state is defined as ss_j, where $j = 0,1,2,...$. And the state ss^0 is defined as the initial one. From a content point of view, the state corresponds to an event class of OOSW, in which the instances of this class go to the appropriate state.

Conclusions. The process of the development of SOM of OOSW, based on its finite-state representation, is considered from the standpoint of the abstract synthesis of a finite state automate. Thus, the considered and justified are the specialties of the synthesis of finite automaton SOM of OOSW, construction of the mapping defining a plurality of channels management of the objects' class, and the order to bring it to an automata, the construction of the canonical multitude of events and their regular expressions for the mapping defining a plurality of channels management of the object class.

Based on the proposed adapted method of the abstract synthesis of the finite automaton model of instances' behavior of the software class, the method of synthesis of the models' states of the software objects (instances of classes) of object-oriented software is

developed. The proposed method provides the formalization of the process of determining the conditions and their relationships in the lifecycle of a software object of OOSW, and also reduces the complexity of developing an integrated dynamic of the component model of OOSW of ADIS in its design at the logical level.

Direction for future researches is developing a test SOM of OOSW with the help of the proposed method. This will ensure the practical implementation of testing of individual behavior of the software objects of OOSW to identify systemic and algorithmic errors in the process of the development of software systems.

References

1. Smirnov O.A. Research the impact of compression on images prompt delivery in the telecommunications system / Smirnov O.A., Dreyev O.M., Dorensky O.P. // Information processing systems. – 2013. – Issue 8(115). – P. 234-239.
2. Orlov S.A. Software engineering / S.A. Orlov. – SPb.: Piter, 2002. – 464 p.
3. Dudzianyi I.M. Object-oriented modeling of software systems / I.M. Dudzianyi. – Lviv: Publishing Center of Ivan Franko LNU, 2007. – 108 p.
4. Smirnov A.A. Mathematical formalization of the process of designing object oriented software information systems / A.A. Smirnov, A.P. Dorensky // Information technology and systems management, education, science: Monograph / Edited by Ponomarenko V.S. – Kh.: Publisher "Shchedra sadyba plus", 2014. – P. 22-36.
5. Yemelianov V.O. Object model software protect sensitive user data / V.O. Yemelianov // Information processing systems.– 2012. – Volume 2, Issue 3 (101). – P. 156-159.
6. Maievskii D.A. A priori estimate of the number of defects in software information systems / D.A. Maievskii, S.A. Yaremtchuk // Radio electronic and computer systems. – 2012. – Issue 5 (56). – P. 73-80.
7. Zhurakovskyi B.Y. Object-oriented technology of designing control systems // Zhurakovskyi B.Y., Varfolomeieva O.G., Gladkykh O.V., Khakhliuk O.A. / Bulletin SUICT. – 2013. – No. 1. – P. 49-53.
8. Veres O.M. The use of object-oriented approach to building models of DSS / O.M. Veres, Y.O. Veres // Bulletin of National University "Lviv Polytechnic". – 2013. – No. 770. – P. 30-36.
9. Dorensky O.P. Synthesis of the object-oriented software integrated model's structure / O.P. Dorensky // Information processing systems. – 2013. – Volume 2, Issue 2(118). – P. 68-72.
10. Glushkov V.M. Synthesis of digital automata / V.M. Glushkov. – M.: Fizmatlit, 1962. – 476 p.

О.П. Доренський, викл.
Кіровоградський національний технічний університет

Метод синтезу моделей станів об'єктів програмного забезпечення автоматизованої системи обробки цифрових зображень

Переважна більшість даних, які обробляються сучасними інфокомунікаційними системами, є графічними. Істотна частка з них – цифрові зображення, які характеризуються великими обсягами. Таким чином, виникає потреба їх представлення у компактному вигляді, що забезпечить зменшення навантаження на канали зв'язку, підвищення оперативності доставки та скорочення обсягів пам'яті, необхідної для зберігання даних. Вирішенням цієї проблеми є розроблення з використанням об'єктно-орієнтованої технології автоматизованої системи обробки цифрових зображень (АСОЗ), на етапі проектування якої постає актуальна задача побудови моделей поведінки екземплярів класів об'єктно-орієнтованого програмного забезпечення (ООПЗ) задля уникнення системних та алгоритмічних помилок, а також скорочення часу розроблення. Тож, мета роботи полягає в розробленні метода синтезу моделей станів програмних об'єктів об'єктно-орієнтованого програмного забезпечення АСОЗ.

Процес побудови моделі станів програмних об'єктів (МСО) ООПЗ, виходячи з її скінченно-автоматного представлення, розглянуто з погляду абстрактного синтезу скінченного автомата. У роботі викладені й обгрунтовані особливості синтезу скінченно-автоматної МСО ООПЗ, побудова відображення, яке визначає множину каналів керування класу об'єктів, та порядок приведення його до автоматного виду, а також побудова канонічної множини подій і їх регулярних виразів для відображення,

яке визначає множину каналів керування класу програмних об’єктів. На основі отриманих результатів дослідження запропоновано метод синтезу моделей станів програмних об’єктів ООПЗ.

Запропонований метод забезпечує формалізацію процесу визначення станів та їх взаємозв’язків у життєвому циклі екземпляра класу ООПЗ, а також дозволяє зменшити трудомісткість процесу розробки динамічної компоненти комплексної моделі ООПЗ під час її проектування на логічному рівні.
програмне забезпечення, автоматизована система, екземпляр класу, об’єктно-орієнтована модель, скінченно-автоматна модель, поведінка об’єкта, стан об’єкта

Одержано 27.02.14

УДК 681.513.5

Б.М. Гончаренко, д-р техн. наук
Національний університет харчових технологій
Л.Г. Віхрова, канд. техн. наук
Кіровоградський національний технічний університет

Алгоритм синтезу оптимальних робастних регуляторів

Розглядається задача побудови оптимального робастного керування у вигляді зворотного зв'язку від стану лінійної динамічної системи, яке мінімізує інтегрально-квадратичний функціонал при найбільш несприятливих збуреннях системи. Отримано однопараметричне сімейство мінімаксних регуляторів, при яких заданий критерій не перевищує деякого граничного значення. Оптимальне мінімаксне керування знаходиться шляхом пошуку мінімально допустимого порогового значення функціоналу за допомогою чисельних ітераційних методів.
оптимізаційна задача, робастність, синтез робастного регулятора, рівняння Ріккаті, функція Гамільтона, сполучена система, критеріальна задача

Б.Н. Гончаренко, д-р техн. наук
Национальный университет пищевых технологий
Л.Г. Вихрова, канд. техн. наук
Кировоградский национальный технический университет
Алгоритм синтеза оптимальных робастных регуляторов

Рассматривается задача построения оптимального робастного управления в виде обратной связи от состояния линейной динамической системы, которое минимизирует интегрально-квадратичный функционал при наиболее неблагоприятных возмущениях системы. Получено однопараметрическое семейство минимаксных регуляторов, при которых заданный критерий не превышает некоторого граничного значения. Оптимальное минимаксное управление находится путём поиска минимально допустимого порогового значения функционала при помощи численных итерационных методов.
оптимизационная задача, робастность, синтез робастного регулятора, уравнение Риккати, функция Гамильтона, соединённая система, критериальная задача

Вступ. Більшість реальних систем або об'єктів керування функціонує [1] в умовах невизначеності, пов'язаної з недостатньою інформацією про об'єкт керування, неточністю його математичної моделі, вихідних даних і т.д. Тому завданням керування об'єктами, що функціонують в умовах невизначеності, приділялася і продовжує приділятися велика увага [2]. У даній роботі розглядається і пропонується розв’язок