УДК 004.02:004.4’24

В.Н. Негода, А.В. Лылова

АВТОМАТИЗАЦИЯ ПРОЦЕССА ВЫБОРА ПРОЕКТНЫХ РЕШЕНИЙ ПРОГРАММНЫХ РЕАЛИЗАЦИЙ ЛОГИЧЕСКИХ ФУНКЦИЙ

Негода Виктор Николаевич, *доктор технических наук, окончил радиотехнический факультет Ульяновского политехнического института, профессор кафедры «Вычислительная техника» Ульяновского государственного технического университета. Имеет статьи, монографии и авторские свидетельства в области проектирования встроенных систем контроля и управления. Область научных интересов – автоматизация проектирования логического управления техническими системами. [e-mail: nvnulstu@gmail.com].*

Лылова Анна Вячеславовна, *окончила магистратуру факультета информационных систем и технологий УлГТУ, старший преподаватель кафедры «Вычислительная техника» УлГТУ. Имеет статьи в области автоматизации проектирования автоматизированных систем. Область научных интересов – автоматизация проектирования логического управления техническими системами. [e-mail: lylovaannav@gmail.com].*

Аннотация

В работе рассматривается сценарий выбора проектных решений программных реализаций логических функций, а также технологические аспекты автоматизации этого процесса. Согласно предлагаемому сценарию для каждого исходного представления функциональной зависимости строится множество реализационных, которые делятся на интерпретируемые и непосредственно исполняемые. Для каждого реализационного решения формируется аналитическая оценка значений критериальных параметров, основными из которых являются время вычисления и затраты памяти. Далее методом исключения пространство проектных решений сужается. Окончательный выбор проектных решений выполняет проектировщик, деятельность которого поддерживается приложением в среде веб-браузера. Спецификации сформированного множества проектных решений отображаются в множество объектов графического диалога системы автоматизированного проектирования. Средства графического диалога обеспечивают навигацию по объектам и управление запуском процессов уточнения значений критериальных параметров. Разработка диалогового приложения базируется на использовании технологии Data Driven Design и Javascript-библиотеки D3.JS. Технология позволяет достаточно простым способом отображать множества значений критериальных параметров проектных решений в графическое представление пространства вариантов выбора. Уточнение значений критериальных параметров базируется на прототипировании с использованием автоматической генерации программ и профилировании.

Ключевые слова: реализация логических функций, разработка на основе моделей, выбор проектных решений.

doi: 10.35752/1991-2927-2019-4-58-45-50

AUTOMATION OF CHOOSING THE DESIGN SOLUTIONS FOR SOFTWARE IMPLEMENTATION OF LOGICAL FUNCTIONS

Viktor NIkolaevich Negoda, *Doctor of Sciences in Engineering; graduated from Radioengineering Faculty of Ulyanovsk Technical Institute; Professor at the Department of Computer Engineering of Ulyanovsk State Technical University; an author of articles, monographs, certificates of authorship in the field of computer-aided design of embedded systems; research interests are in the field of computer-aided design of engineering systems with logical control. e-mail: nvn@ulstu.ru.*

Anna Viacheslavovna Lylova, *graduated from the Faculty of Information Systems and Technologies of Ulyanovsk State Technical University with the Master’s degree; Senior Lecturer at the Department of Computer Engineering of UlSTU, an author of articles in the field of computer-aided design of embedded systems; research interests are in the field of computer-aided design of engineering systems with logical control. e-mail: lylovaannav@gmail.com.*

Abstract
The article deals with the scenario of choosing design solutions for software implementation of logical functions as well as technological aspects of automation of this process. According to the proposed scenario, for each initial representation of the functional dependence, a lot of implementation dependencies are built, which are divided into interpreted and directly executed. For each implementation solution, an analytical assessment of the values of the criterial parameters is formed, the main of which are the calculation time and memory costs. Further, by the method of exclusion, the space of design decisions is narrowed. The final choice of design decisions is made by the designer, whose activities are supported by the application in a web browser environment. The specifications of the generated set of design decisions are mapped into the set of graphical dialogue objects of the computer-aided design system. The graphical dialogue tools provide navigation on objects and control the launch of processes for updating the values of criteria parameters. The development of the interactive application is based on the use of Data Driven Design Technology and the D3.JS Javascript Library. The technology allows a fairly simple way to display the set of values of the criterial parameters of design decisions in a graphical representation of the space of options. The refinement of the values of the criteria parameters is based on prototyping using automatic program generation and profiling.

Key words: implementation of logical functions, development based on models, choice of design solutions.

Введение

Программные реализации логических функций широко используются в системах логического управления (СЛУ) [1], в автоматизированных системах логического проектирования [2], в системах моделирования процессов со сложной логикой поведения [3]. Для логических функций характерно большое многообразие вариантов программной реализации, что предопределено большим количеством возможных представлений функций, широкими возможностями применения методов минимизации и многовариантностью отображения спецификаций функций в программный код.

Значения критериальных параметров программных реализаций существенно зависят от свойств целевых платформ, что часто не позволяет получить точные значения этих параметров до получения работающей программы [4]. По мере развития средств вычислительной техники свойства целевых платформ меняются, что заставляет возвращаться даже к давно проведенным исследованиям [5].

В этой связи актуальна задача автоматизации процесса выбора проектных решений. Для решения этой задачи предлагается излагаемый ниже сценарий процесса, основанный на использовании аналитических оценок, описанных в работе [6].

Особенности оценки значений критериальных параметров

Аналитические оценки затрат памяти являются детерминированными, и для их получения достаточно иметь множество процедур, состав которого определяется многообразием способов представления реализуемой функциональной зависимости в памяти программ и памяти данных. При этом следует различать интерпретируемые представления и непосредственно исполняемые. В первом случае спецификация функции отображается в наборы данных, а во втором – в код программ.

Поясним это на примере функции, для представления которой используется математический аппарат троичных матриц [2].

Пусть функция зависит от 8 булевых переменных и ее троичная матрица имеет вид:

10--11--
-111---0.

Интерпретируемое представление предполагает формирование набора данных, кодирующего матрицу, и создание интерпретатора, который для входного набора значений аргументов проверяет факт его покрытия одним из троичных векторов матрицы. Эффективным является представление каждого троичного вектора V парой бит-векторов, первый из которых задает небезразличные значения (обозначим $V.val$), а второй – набор масок для маскирования сравнения в позициях безразличных значений (обозначим $V.mask$). Для приведенной троичной матрицы получим 2 двухбайтовых вектора троичной матрицы M:

$$M = \left[\{val : 10001100,\ mask : 11001100\},\right.$$
$$\left.\{val : 01110000,\ mask : 01110000\}\right].$$

Установление факта, что набор значений аргументов X покрывается троичным вектором V, в этом случае обеспечивается формулой: $X \,\&\, V.mask == V.val$. Очевидно, что интерпретация троичной матрицы при этом сводится к выполнению такого цикла поиска с маскированием, в теле которого реализуется приведенная выше формула. Соответствующий интерпретатор пригоден для реализации любых функций от не более 8 переменных, и суммарные затраты памяти для реализации нескольких функций определяются выражением:

$$R = Rp + \sum\nolimits_i M[i].length,$$

где Rp – память процедуры интерпретации, $M[i].length$ – число троичных векторов в матрице i-й функции.

Непосредственно исполняемое представление на основе рассмотренной выше троичной матрицы M является продуктом трансляции следующего выражения, которое справедливо для многих языков программирования с Си-подобным синтаксисом:

$$y = X \,\&\, M[0].mask == \\ == M[0].val \,\|\, X \,\&\, M[1].mask == M[1].val.$$

Объем памяти, занимаемый таким непосредственно исполняемым представлением, выражается формулой $R = M.length * Rv$, где Rv – размер программного кода, который тратится на сопоставления набора значений аргументов X с одним троичным вектором. Здесь возникает потребность в калибровке оценки затрат памяти, поскольку величина Rv зависит от целевой платформы и используемого транслятора языка программирования. Калибровка заключается в выполнении трансляции, определении величины Rv и сохранении ее в базе данных САПР вместе со спецификаторами целевой платформы и транслятора.

Оценка затрат времени вычисления логической функции, приведенных в работе [6], имеет вид неравенств:

$$T\,max \le Tp + N * Ts,$$

где Tp – время подготовки итерационного процесса вычисления, Ts – затраты времени одной итерации, N – количество шагов, которое необходимо выполнить в худшем случае. Для рассмотренной выше троичной матрицы худший случай имеет место, когда ни один из ее троичных векторов не покрывает набор значений аргументов X. Тогда $N = 2$.

В случаях, когда рабочая нагрузка на процедуру вычисления логической функции представляет собой потоки наборов значений аргументов, практическое значение имеет среднее значение времени вычисления $Tavg$, которое зависит от характера рабочей нагрузки, который в общем случае специфицируется как случайный процесс. Калибровка выражения для $Tavg$ заключается в оценке среднего значения числа троичных векторов $Navg$, которые охватывает процесс реализации функции при определенной рабочей нагрузке.

Затраты времени одной итерации Ts для многих современных целевых платформ также являются случайной величиной. Это связано с такими свойствами целевой платформы, которые уменьшают определенность числа машинных тактов, затрачиваемых на исполнение кода. Прежде всего это относится к влиянию на время исполнения кода алгоритмов предсказания переходов в многостадийном конвейере и алгоритмов кэширования в многоуровневой памяти. Калибровка в этом случае требует разработки прототипа и его профилирования.

Приведенные выше рассуждения справедливы для большинства аналитических оценок, приведенных в работе [6], т. е. большинство способов задания булевых функций допускают интерпретируемые и непосредственно исполняемые представления, для которых требуется калибровка аналитических оценок. В случае, когда база данных САПР содержит результаты калибровки, достаточные для получения точных оценок значений критериальных параметров, соответствующее проектное решение сразу может включаться в множество проектных решений, предъявляемых проектировщику для выбора. Если точность оценок низка в силу недостатка калибровочных данных, то требуется выполнить их уточнение путем калибровки.

Сценарий управления выбором проектных решений

Изначально в процессе выбора проектных решений фигурируют спецификации всевозможных проектных решений. Этот набор спецификаций аккумулируется в стартовом множестве спецификаций $sdStart$. Большинство спецификаций этого множества формируется автоматически путем применения процедур трансформации представлений логических функций и автоматической генерации программ.

На первом этапе формируются аналитические оценки критериальных параметров, которые включают в себя возможные граничные и средние значения и становятся частью исходных спецификаций. Спецификации, снабженные оценками, переносятся либо в финишное множество $sdFinish$, либо в множество $sdRevision$, где аккумулируются спецификации, требующие уточнения. Выбор целевого множества определяется диапазонами между наихудшими и наилучшими значениями, наличием в базе данных таких данных калибровки аналитических моделей оценки значений критериальных параметров, которые обеспечивают достаточную точность для принятия решения, а также соотношением трудоемкости процесса уточнения с ожидаемым от этого эффектом. Оценка решения выполняется с использованием базы данных накопленных оценок и результатов калибровки аналитических оценок, приведенных в работе [6]. Если в базе данных имеется сохраненная оценка для подобной реализации функции, то используется именно она. Иначе оценка строится на основе результатов профилирования, которые порождают новые записи калибровки аналитических моделей.

На втором этапе выполняется уточнение оценок для проектных решений, спецификации которых находятся в множестве $sdRevision$. Процедурная поддержка этого процесса заключается в инициировании генераторов рабочей нагрузки, генерации сценариев профилирования, исполнении этих сценариев. Возможно диалоговое формирование оценок экспертом на основе анализа различий между условиями получения имеющихся в базе данных результатов профилирования и условиями оцениваемого проектного решения. По мере уточнения оценок спецификации переносятся в финишное

множество *sdFinish*. При этом возможно сохранение уточненных спецификаций в базе данных САПР для дальнейшего использования.

На третьем этапе выполняется сужение множества *sdFinish* путем отбрасывания заведомо нерациональных проектных решений. Исключению подвергаются те реализационные представления, для которых границы наилучших значений хуже границ с наихудшими значениями сохраняемых представлений. Эта операция подобна операции выбора подмножеств проектных решений для продолжения разбиения всего пространства проектных решений в случае применения метода ветвей и границ [7]. Это выполняется либо автоматически на основе зафиксированных в конфигурации ограничений, либо в диалоге с экспертом, который может учесть дополнительные критериальные параметры [8].

На последнем этапе над спецификациями суженного множества *sdFinish* работают только эксперты. При выборе одного из этих решений процесс прекращается, т. е. сценарий завершается. Если даже самое лучшее решение обладает недостатками и при этом существует потенциал порождения нового проектного решения с лучшими характеристиками, то выполняется инициирование соответствующей работы и возможно расширение множества *sdFinish*.

Предлагаемый сценарий управления выбором проектных решений представлен в формате диаграммы деятельности на рисунке, где фигурируют следующие предикаты:

P1 – в множестве *sdStart* больше нет необработанных элементов;

P2 – для достаточно точной оценки решения в базе данных САПР имеются результаты калибровки;

P3 – множество *sdFinish* сформировано;

P4 – целесообразно занести результаты калибровки в базу данных накопленных оценок с целью повторного использования;

P5 – требуется инициирование формирования нового проектного решения.

Этот сценарий предполагает предельно высокую степень автоматизации процессов на первом этапе и снижение уровня автоматизации на втором и третьем. Производительность труда проектировщиков на втором этапе повышается за счет использования процедур управления процессами калибровки, а на третьем – за счет процедур трансформации одних представлений функций в другие и процедур автоматической генерации кода. Благодаря наличию в САПР именно этих процедур удается вовлечь в разработку программных реализаций функций существенно более широкий спектр проектных решений, нежели это возможно при традиционном процессе проектирования, для которого автоматизация проектирования фокусируется на поддержке формирования только одного проектного решения.

Технологические аспекты автоматизации выбора проектных решений

Базовые спецификации проектных решений представляют собой кортежи, значения которых хранятся в базе данных и отображаются в объекты программного обеспечения САПР. Связывание с процедурами оценки значений критериальных параметров на этом уровне возможно только с помощью спецификаторов типов проектных решений. Эти спецификаторы изначально представляют собой лексические единицы, никак не связанные с адресами процедур оценки и поддержки других манипуляций с проектными решениями.

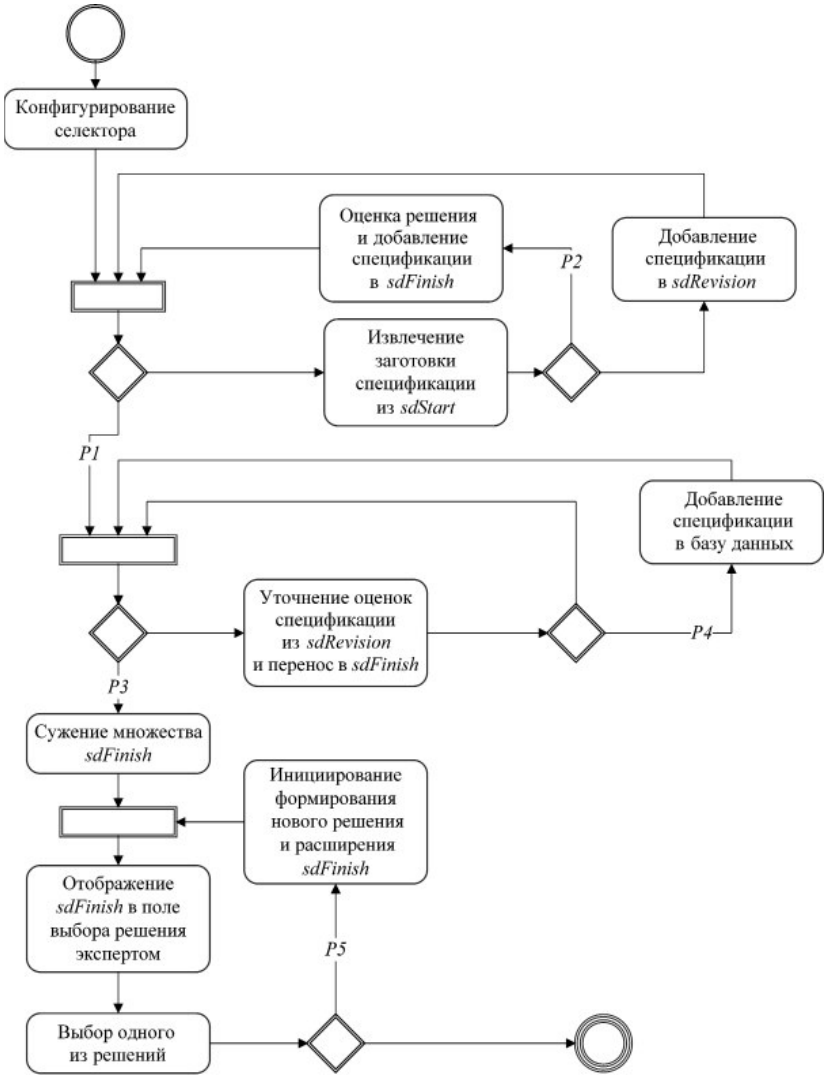

Рис. Обобщенный сценарий процесса выбора решений

В ходе ввода спецификаций из базы данных сначала порождаются объекты, в которые копируются элементы спецификаций. Затем на основе значений спецификаторов типов объекты расширяются набором ссылок на методы, основные из которых следующие:

• calcMem – ссылка на процедуру оценки затрат памяти;

• calcTime – ссылка на процедуру оценки затрат времени;

• calibrTime – ссылка на процедуру поддержки калибровки оценки затрат времени;

• calibrMem – ссылка на процедуру поддержки калибровки оценки затрат памяти;

• transformDS – набор ссылок на процедуры трансформации проектного решения в возможные альтернативные;

• showDS – ссылка на процедуру показа расширенной спецификации проектного решения.

Поскольку все процедуры являются частью объектов, отпадает необходимость в передаче параметров в эти процедуры, а унификация имен методов позволяет существенно упросить групповую обработку большого набора спецификаций и многое делать в автоматическом режиме. Для этого все объекты агрегируются в коллекцию спецификаций проектных решений DSC. Эта коллекция снабжается итератором, благодаря которому обеспечивается простой механизм обхода всех возможных решений при выполнении рассмотренного выше сценария. Например, автоматическая оценка затрат памяти всех проектных решений достигается одним оператором DSC.forEach(calcMem).

Для перевода части коллекции из чисто автоматического режима в автоматизированный с участием эксперта на этапе конфигурирования назначаются правила, позволяющие связывать методы оценивания с диалоговыми процедурами, сохраняя имена ссылок в объектах специфицирования и возможность упрощения обхода всех проектных решений.

Окончательный выбор проектных решений выполняет проектировщик, деятельность которого поддерживается приложением в среде веб-браузера. Спецификации сформированного множества проектных решений отображаются в множество объектов графического диалога системы автоматизированного проектирования. Используется два варианта представления проектных решений: упорядоченные по значениям критериальных параметров списки и двумерное пространство в координатах «время, память».

Средства графического диалога обеспечивают навигацию по объектам и управление запуском процессов уточнения значений критериальных параметров. При этом возможно задание наборов значений итератора, что поддерживает групповую обработку проектных решений с высоким уровнем автоматизации проектных процессов.

Разработка диалогового приложения базируется на использование технологий Model-Driven Development [9], Data Driven Documents [10] и Sample-Oriented Task-Driven Visualizations [11] с применением Javascript-библиотеки D3.JS. Совокупность этих технологий позволяет достаточно простым способом отображать множества значений критериальных параметров проектных решений в графическое представление пространства вариантов выбора. При этом все отображаемые в двумерном пространстве графические обозначения решений снабжаются ссылками на процедуру показа расширенных спецификаций showDS, что позволяет сочетать лаконичность, наглядность и полноту представления проектных решений в базовом диалоговом компоненте поддержки работы эксперта.

Заключение

Автоматизацию процесса выбора проектных решений целесообразно строить как процесс формирования спецификаций, содержащих описание реализуемых функций. Стартовый набор спецификаций, хранящихся в базе данных САПР, отображает коллекцию объектов, каждый из которых включает в себя ссылки на процедуры оценки, процедуры поддержки калибровки аналитических выражений оценки, процедуры поддержки трансформации одних проектных решений в другие. В ходе обработки элементов коллекции формируется финишное множество спецификаций, предъявляемых эксперту для принятия решения.

СПИСОК ЛИТЕРАТУРЫ

1. Шалыто А.А. Логическое управление. Методы аппаратной и программной реализации алгоритмов. – СПб. : Наука, 2000. – 780 с.

2. Закревский А.Д. Программирование вычислений в многомерном булевом пространстве // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. – 2008. – № 2 (3). – С. 94–105.

3. Stateflow and Stateflow Coder. For Complex Logic and State Diagram Modeling. Modeling, Simulation, Implementation. User Guide. The MathWorks. 2003. 896 p. – URL: http://fractale.gecif.net/si/logiciels/matlab/stateflow.pdf (дата обращения: 20.10.2019).

4. Прангишвили И.В. Проектирование устройств логического управления. – М. : Наука, 1984. – 239 с.

5. El-Bakry, Atwan A. Simplification and Implementation of Boolean Functions // International Journal of Universal Computer Sciences. 2010. Vol. 1, iss. 1. pp. 41–50.

6. Негода В.Н., Лылова А.В. Параметризация проектных решений в системе автоматизации проектирования программных реализаций функций логического управления // Радиотехника. – 2017. – № 6. – С. 99–106

7. Норенков И.П., Арутюнян И.М. Эволюционные методы в задачах выбора проектных решений // Наука и образование. Электронное издание МГТУ им. Н.Э. Баумана. – 2007. – № 9. – URL: http://engineering-science.ru/doc/68376.html (дата обращения: 11.09.2019).

8. Антипова Е.В. Влияние способа преобразования автоматных диаграмм на параметры сгенерированно-

го программного кода // Вестник Волжского университета им. В.Н. Татищева. Сер. Информатика. – 2012. – Вып. 20. – С. 111–122.

9. Parviainen P., Takalo J., Teppola S., Tihinen M. Model-Driven Development. Process and practices. – URL: http://www.vtt.fi/inf/pdf/workingpapers/2009/ W114.pdf (дата обращения: 11.09.2019).

10. Ferreira N., Fisher D., König A. Sample-oriented task-driven visualizations: Allowing users to make better, more confident decisions // Conference on Human Factors in Computing Systems. Proceedings. Toronto, Ontario, Canada, 2014. pp. 571–580.

11. Alferd D., Tauro C. Web 3D Data Visualization of Spatio Temporal Data using Data Driven Document (D3js) // International Journal of Computer Applications. 2015. 111(4). pp. 42–46.

REFERENCES

1. Shalyto A.A. *Logicheskoe upravlenie. Metody apparatnoi i programmnoi realizatsii algoritmov* [Logic Control. Procedures of Hardware and Software Implementation of Algorithms]. St. Petersburg. Nauka Publ., 2000. 780 p.

2. Zakrevskii A.D. Programmirovanie vychislenii v mnogomernom bulevom prostranstve [Manipulations on Large Variables Boolean Functions]. *Vestnik Tomskogo gosudarstvennogo universiteta. Upravlenie, vychislitelnaia tekhnika i informatika* [Tomsk State University Journal of Control and Computer Science], 2008, no. 2 (3), pp. 94–105.

3. *Stateflow and Stateflow Coder. For Complex Logic and State Diagram Modeling. Modeling, Simulation, Implementation. User Guide. The MathWorks.* 2003. 896 p. Available at: http://fractale.gecif.net/si/logiciels/matlab/stateflow.pdf (accessed: 20.10.2019).

4. Prangishvili I.V. *Proektirovanie ustroistv logicheskogo upravleniia* [Design of Logic Control Devices]. Moscow, Nauka Publ., 1984. 239 p.

5. El-Bakry, Atwan A. Simplification and Implementation of Boolean Functions. *International Journal of Universal Computer Sciences*, 2010, vol. 1, iss. 1, pp. 41–50.

6. Negoda V.N., Lylova A.V. Parametrizatsiia proektnykh reshenii v sisteme avtomatizatsii proektirovaniia programmnykh realizatsii funktsii logicheskogo upravleniia [Parameterization of Design Solutions in the Automation System for the Design of Software Implementations of Logical Control Functions]. *Radiotekhnika* [Journal of Radioengineering], 2017, no. 6, pp. 99–106.

7. Norenkov I.P., Arutuiunian I.M. Evoliutsionnye metody v zadachakh vybora proektnykh reshenii [Evolutionary Techniques in Design Selection Tasks]. *Nauka i obrazovanie. Elektronnoe izdanie MGTU im. N.E. Baumana* [Scientific Edition of Bauman MSTU. Science and Education], 2007, no. 9. Available at: http://engineering-science.ru/doc/68376.html (accessed: 11.09.2019).

8. Antipova E.V. Vliianie sposoba preobrazovaniia avtomatnykh diagramm na parametry sgenerirovannogo programmnogo koda [The Influence of the Method of Converting Automaton Diagrams on the Parameters of the Generated Program Code]. *Vestnik Volzhskogo universiteta im. V.N. Tatishcheva. Ser. Informatika* [Vestnik of Volzhsky University after V.N. Tatischev. Informatics], 2012, iss. 20, pp. 111–122.

9. Parviainen P., Takalo J., Teppola S., Tihinen M. *Model-Driven Development. Process and Practices.* Available at: http://www.vtt.fi/inf/pdf/workingpapers/2009/ W114.pdf (accessed: 11.09.2019).

10. Ferreira N., Fisher D., König A. Sample-Oriented Task-Driven Visualizations: Allowing Users to Make Better, More Confident Decisions. *Conference on Human Factors in Computing Systems. Proceedings.* Toronto, Ontario, Canada, 2014, pp. 571–580.

11. Alferd D., Tauro C. Web 3D Data Visualization of Spatio Temporal Data using Data Driven Document (D3js). *International Journal of Computer Applications*, 2015, iss. 111 (4), pp. 42–46.