УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
январь–февраль 2019 Том 20 № 1 ISSN 2226-1494 http://ntv.itmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTCS
January–February 2020 Vol. 20 No 1 ISSN 2226-1494 http://ntv.itmo.ru/en/


**УДК 681.518.54** doi: 10.17586/2226-1494-2020-20-1-94-100

# АВТОМАТИЗАЦИЯ ПОСТРОЕНИЯ СРЕДСТВ ДИАГНОСТИРОВАНИЯ ДЛЯ ПОТОКОВОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ РЕАЛЬНОГО ВРЕМЕНИ

**Е.В. Лукоянов, А.М. Грузликов**

АО «Концерн «ЦНИИ «Электроприбор», Санкт-Петербург, 197046, Российская Федерация
Адрес для переписки: lukoyanov.egor@mail.ru

**Информация о статье**
Поступила в редакцию 06.12.19, принята к печати 30.12.19
Язык статьи — русский

**Ссылка для цитирования:** Лукоянов Е.В., Грузликов А.М. Автоматизация построения средств диагностирования для потоковой вычислительной системы реального времени // Научно-технический вестник информационных технологий, механики и оптики. 2020. Т. 1. № 1. С. 94–100. doi: 10.17586/2226-1494-2020-20-1-94-100

**Аннотация**

**Предмет исследования.** Рассмотрены вопросы проектирования средств диагностирования нарушений в адресации информационных обменов между программными модулями для потоковой вычислительной системы реального времени. Несмотря на декомпозицию процесса проектирования на основе иерархического подхода, он остается достаточно сложным, а значит, остается актуальной и проблема его автоматизации. **Метод.** Проблема автоматизации построения модели и тестов для потоковой вычислительной системы реального времени решается на основе декомпозиции и с привлечением дискретно-событийного моделирования. **Основные результаты.** Разработана инструментальная среда, автоматизирующая процедуру построения модели, генерации тестовых воздействий и эталонных выходных последовательностей. Приведено ее краткое описание. В основу функционала среды положены алгоритмы синтеза динамической модели системы и формирования теста для диагностирования нарушений обменов между программными модулями системы. **Практическая значимость.** Разработанная инструментальная среда позволяет существенно сокращать время проектирования средств диагностирования для потоковых вычислительных систем реального времени.

**Ключевые слова**
автоматизация построения модели, дискретно-событийная модель, потоковая вычислительная система, тестовое диагностирование, периодически нестационарная система

**Благодарности**
Работа проведена при поддержке гранта РФФИ № 19-08-00052.

doi: 10.17586/2226-1494-2020-20-1-94-100

# AUTOMATION OF DATAFLOW REAL-TIME COMPUTING SYSTEM DIAGNOSTICS

**E.V. Lukoyanov, A.M. Gruzlikov**

CSRI Elektropribor, JSC, Saint Petersburg, 197046, Russian Federation
Corresponding author: lukoyanov.egor@mail.ru

**Article info**
Received 06.12.19, accepted 30.12.19
Article in Russian

**For citation:** Lukoyanov E.V., Gruzlikov A.M. Automation of dataflow real-time computing system diagnostics. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2020, vol. 20, no. 1, pp. 94–100 (in Russian). doi: 10.17586/2226-1494-2020-20-1-94-100

**Abstract**

**Subject of Research.** The paper considers the design issues of fault diagnostic tools in information exchange addressing between program modules for a dataflow real-time computing system. Despite the decomposition of design process on the basis of a hierarchical approach, it is quite complicated, and the problem of its automation remains an urgent challenge. **Method.** The design automation of model and tests for a dataflow real-time computing system is performed on the basis of decomposition and by applying discrete-event modeling. **Main Results.** We have developed the instrumental environment automating procedure for model design, generation of test actions and reference output sequences. Its brief

description is given. The environment functional is based on the synthesis algorithms of the system dynamic model and the test formation for exchange diagnostics between the system software modules. **Practical Relevance.** The developed tool environment gives the possibility to reduce significantly the design time of diagnostic tools for dataflow real-time computing systems.

**Keywords**
model design automation, discrete-event model, dataflow computing system, test diagnostics, periodically non-stationary system

**Acknowledgements**
This work was supported by the project No. 19-08-00052 of the Russian Foundation for Basic Research, Russian Federation.

## Введение

Рассмотрение вопросов диагностирования нарушений занимает важное место в процессе проектирования систем обработки информации и управления, поскольку от качества их решения зависит надежность и отказоустойчивость систем. Применяемые на практике процессы проектирования систем обработки информации и управления основываются на техниках функционального и тестового диагностирования [1–8]. Разработка средств диагностирования (СД) для сложной системы является непростой задачей из-за высокой размерности системы, параллельности вычислительного процесса и многообразия причин возникновения нарушений. Источником нарушений могут быть ошибки в программном обеспечении (ПО), сбои и отказы в работе аппаратуры, а также ошибки при проектировании информационного взаимодействия между задачами системы. С целью преодоления высокой сложности проблемы процесс разработки СД декомпозируется с использованием иерархического подхода [8], когда множество компонентов системы в соответствии с отношением включения распределяется по уровням сложности так, что компоненты более высокого уровня представляются композицией компонентов более низкого уровня. В настоящей работе рассматривается проблема автоматизации тестового диагностирования самого верхнего уровня для потоковой вычислительной системы (ПВС) реального времени. На этом уровне ПВС представляется композицией локальных подсистем, а в нашем случае — композицией программных модулей (ПМ), составляющих ПО системы и размещенных по локальным подсистемам. Это позволяет охватить рассмотрением как распределенные, так и локальные вычислительные системы. Далее будет рассмотрена распределенная вычислительная система, подразумевая, что формулируемые результаты справедливы также и для локальной системы, реализованной на одном процессоре. В данном случае под потоковой системой понимается система, в которой ПМ запускаются по готовности входных данных. Несмотря на декомпозицию, процесс проектирования СД для ПВС остается достаточно сложным, в том числе и для самого верхнего уровня, а значит, остается актуальной и проблема автоматизации этого процесса.

В статье приведено краткое содержание подхода к диагностированию ПВС, предложенного и исследованного при участии авторов. Дано описание используемой модели ПВС и процедуры ее синтеза. Показана процедура, используемая при автоматизации построения тестов. Приведено краткое описание разработанной инструментальной среды проектирования.

## Диагностическая модель потоковой вычислительной системы

Разработка СД всегда начинается с выбора диагностической модели. Особенностью рассматриваемого подхода [9–12] является то, что диагностическая модель встраивается в систему и исполняется параллельно с основным ПО системы, что позволяет упрощать процесс тестового диагностирования системы. Данную диагностическую модель с точки зрения рассматриваемого класса отказов можно отнести к дискретно-событийным системам [13], когда работа системы представляется на языке последовательностей некоторых событий. В данном случае — это события информационных обменов между ПМ. Подобные модели широко применяются при анализе и тестировании сложных систем [14]. С точки зрения математического описания используемая модель относится к конечно-автоматным, а точнее, к линейным двоичным динамическим системам. Известно, что конечно-автоматные модели применяются при решении диагностических задач как в отношении аппаратуры [8], так и в отношении ПО [5, 6].

Проиллюстрируем процедуру построения модели на простом примере (рис. 1). Здесь приведен пример системы $S$ реального времени, заданной графом межмодульных связей. В системе реализуются три функционально связанных ПМ: $ПМ_1$, $ПМ_2$ и $ПМ_3$, которые могут быть размещены как на разных процессорах системы, так и на одном. Каждый из ПМ на основе входных данных ($u_1$ — для $ПМ_1$; $u_2$ и $y_3$ — для $ПМ_2$; $y_1$ и $y_2$ — для $ПМ_3$) формирует выходные данные ($y_1$, $y_2$ и $y_3$ соответственно). Входные данные поступают и обрабатываются в реальном времени с некоторым заданным периодом. В системе все входные потоки конкретного ПМ являются аргументами реализуемой им функции, необходимыми для ее вычисления.

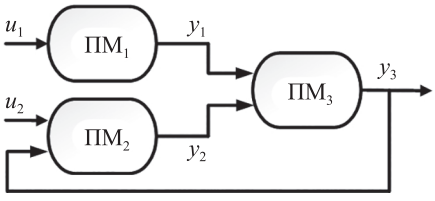


*Рис. 1.* Граф межмодульных связей системы

На первом этапе построения модели на основе известных алгоритмов [11, 12] формируется множество вычислительных путей, составляющих покрытие дуг графа межмодульных связей. При этом под вычислительным путем понимаем последовательность ПМ, соединяющую некоторый вход с выходом. Затем с каждым из полученных путей сопоставляется цепь из такого числа динамических звеньев, через сколько ПМ проходит данный путь. После описанных построений модель системы представляется совокупностью функционально независимых цепей, а задача диагностирования может быть сведена к диагностированию отдельных цепей.

На втором этапе формирования модели определяется вид динамических звеньев. При этом учитывается, что искомая динамическая модель системы далее используется для построения тестов, и что процедура построения тестов упрощается, если модель системы, во-первых, линейна, а во-вторых, управляема и наблюдаема [8]. Отсюда можно сформулировать требование к звеньям цепей модели. Они должны быть линейны, т. е.

$$\mathbf{x}_{i,j}(t+1) = \mathbf{f}_{i,j}\mathbf{x}_{i,j}(t) + \mathbf{g}_{i,j}\mathbf{u}_{i,j}(t),$$
$$\mathbf{y}_{i,j}(t) = \mathbf{h}_{i,j}\mathbf{x}_{i,j}(t),\ i = \overline{1,n_j},\ j = \overline{1,m},$$

где $\mathbf{f}_{i,j}$, $\mathbf{g}_{i,j}$, $\mathbf{h}_{i,j}$ — матрицы динамики, входа и выхода соответственно; $\mathbf{x}_{i,j}(t)$, $\mathbf{u}_{i,j}(t)$, $\mathbf{y}_{i,j}(t)$ — векторы состояния, входа и выхода соответственно для $i$-того звена модели $j$-той цепи; $n_j$, $m$ — число звеньев в $j$-той цепи и число цепей в модели системы соответственно. Кроме того, звенья должны быть таковы, чтобы модель системы была наблюдаемой и управляемой.

На рис. 2 система $S$ приведена совместно с СД. В систему уже введена избыточность, представленная алгоритмами $f_{d1}$, $f_{d2}$ и $f_{d3}$. СД состоят из генератора тестов (ГТ), генератора эталонных реакций (ГЭР) и компаратора (К). СД формируют для системы тестовые данные, дополняя ими входные данные для $ПМ_1$ и $ПМ_2$, и анализируют выходную реакцию $S$ ($ПМ_3$). В каждый ПМ — $ПМ_1$, $ПМ_2$ и $ПМ_3$ — по каналам обмена поступает информация, которая обрабатывается штатными алгоритмами. Параллельно с этим тестовые информационные слова обрабатываются специальными алгоритмами $f_{d1}$, $f_{d2}$ и $f_{d3}$, реагирующими на события приема/выдачи информации, а результаты их обработки выдаются в составе выходных данных. Поскольку механизм обмена реальными и тестовыми данными в системе является общим, возникает возможность по наблюдаемым в процессе работы тестовым результатам делать вывод о наличии или отсутствии нарушений в адресации обменов. Заметим, что искажения реальных данных в процессе обмена при сохранении графа информационных связей не входят в этот класс рассматриваемых нарушений.

На рис. 2 черным цветом обозначены связи между ПМ системы, зеленым цветом — связи между диагностическими алгоритмами. Соответственно, $u_1'$ и $u_2'$ являются входными (тестовыми) данными для $f_{d1}$ и $f_{d2}$, а $y_1'$ и $y_2'$ — выходными. Выходные данные $y_3'$ диагностического алгоритма $f_{d3}$, выделенные штрихпунктирной линией, поступают на обработку в СД. В результате добавления $f_{d1}$, $f_{d2}$ и $f_{d3}$ в системе образуются новые

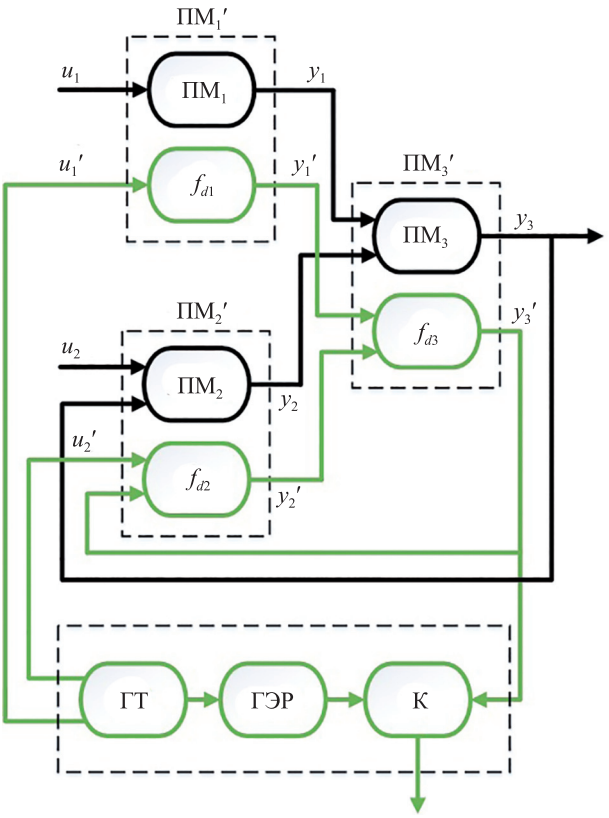


*Рис. 2.* Граф межмодульных связей избыточной системы и средств тестового диагностирования

ПМ: $ПМ_1'$, $ПМ_2'$ и $ПМ_3'$, которые также выделены штрихпунктирной линией.

Таким образом, проблема состоит в построении алгоритмов обработки тестов в ПМ, а также в построении по виду этих алгоритмов самих тестов. Алгоритм обработки тестовых данных является событийной моделью системы и, как будет показано ниже, имеет вид линейной дискретной периодически нестационарной системы.

В терминах этой модели класс рассматриваемых нарушений определяется как всевозможные искажения матриц этой модели.

После описанных построений структура модели системы представляется совокупностью независимых цепей, а задача диагностирования может быть сведена к диагностированию отдельных цепей. Структура модели для рассматриваемого примера, состоящая из независимых цепей, приведена на рис. 3, *а*. Однако применение такой модели ПВС может в некоторых случаях потребовать передачи через каналы обмена большого количества диагностической информации, что не всегда допустимо. В подобных ситуациях целесообразно воспользоваться приемом, заключающимся в обработке нескольких массивов информации одним звеном (слияние вычислительных путей) [10].

Этот вариант иллюстрируется на рис. 3, *б*. Здесь выходные массивы информации звеньев $M_{11}$ и $M_{32}$ обрабатываются звеном $M_{21}$, а звено $M_{42}$ исключается. В результате сокращается размерность выходного вектора, а следовательно, и объем выдаваемой из ПМ диагностической информации в каждом такте.

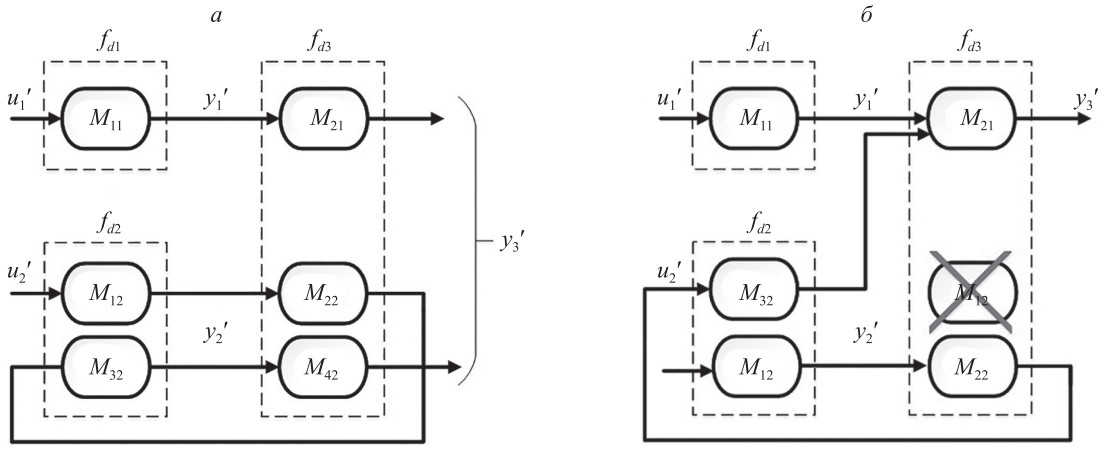


*Рис. 3*. Структура модели для системы: с независимыми цепями (*а*); со слиянием цепей (*б*)

Динамическое описание цепи или модели со слиянием цепей получается по следующим правилам. Вектор состояния $\mathbf{x}(t)$ модели формируется из векторов состояния звеньев $\mathbf{x}_{i,j}(t)$, входящих в эту модель. Предположим, что в каждый момент времени в системе происходит лишь один обмен, тогда перенос информации между ПМ и СД в каждом информационном обмене может быть описан с помощью матриц $\mathbf{F}(t)$, $\mathbf{G}(t)$, $\mathbf{H}(t)$. На практике это предположение не выполняется, однако, как показано в работах [11–14], это не является препятствием для использования таких моделей при построении тестов. Для удобства описания свяжем с каждой последовательностью матриц на интервале, равном периоду, свою последовательность индексов, полученных из множества $\{1, 2, \ldots, N\}$ в результате циклического сдвига. Например, при $N = 3$ получаем множество, состоящее из трех последовательностей индексов $\Gamma = \{1, 2, 3; 2, 3, 1; 3, 1, 2\}$. Обозначим за $\gamma_r$ элемент множества Г. Тогда

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{F}(\gamma_r(j))\mathbf{x}(t) + \mathbf{G}(\gamma_r(j))\mathbf{u}(t), \\ \mathbf{y}(t) &= \mathbf{H}(\gamma_r(j))\mathbf{x}(t), j = \overline{1,N}, \end{aligned} \qquad (1)$$

где $\mathbf{u}(t)$ — вектор входных воздействий; $\mathbf{y}(t)$ — вектор выходной реакции системы; $N$ — общее число обменов.

### Алгоритм построения теста для сетединамической модели

Проверяющий тест для периодически нестационарной системы (1) состоит из $N$ фрагментов ($N$ — период системы): $U_T = U_{\gamma 1}U_{\gamma 2}\ldots U_{\gamma N}$. Каждый из фрагментов включает две характерные части. В первой части $U_{\gamma r}^{(1)}$ всех фрагментов система при последовательности индексов $\gamma_r$ проходит в пространстве состояний через состояния выбранного базиса $X = \{\mathbf{x}_i | i = 1,n\}$. Для каждого состояния $\mathbf{x}_i$ во фрагменте предусмотрены установочная последовательность $\mathbf{u}_{\gamma r}^{(1)}$ длиной кратной $N$, и интервал свободного движения при последовательности индексов $\gamma_r$, когда на входе системы подаются $nN$ нулей. Во второй части $U_{\gamma r}^{(2)}$ всех фрагментов на вход системы последовательно подаются вектора $\mathbf{u}_{\gamma r}^{(2)}$, принадлежащие некоторому базису пространства входных векторов. После каждого вектора система находится в свободном движении на $nN$ тактах при последовательности индексов $\gamma_r$.

Далее при построении теста используется следующий известный факт: для любой цепи $S^c$, которая описывается моделью (1) и имеет на периоде функционирования один сеанс приема информации от средств диагностирования, описываемый для некоторой последовательности индексов $\gamma_r$ матрицами $\mathbf{F}(\gamma_r(N))$, $\mathbf{G}(\gamma_r(N))$, $\mathbf{H}(\gamma_r(N))$, и один сеанс выдачи информации в средства диагностирования, описываемый матрицами $\mathbf{F}(\gamma_r(N-1))$, $\mathbf{G}(\gamma_r(N-1))$, $\mathbf{H}(\gamma_r(N-1))$, существует стационарная система $S^c$:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k),\ \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k), \qquad (2)$$

где

$$\mathbf{A} = \mathbf{F}_p(\gamma_r) = \prod_{i=1}^{N}\mathbf{F}(\gamma_r(N-i+1)),\ \mathbf{B} = \mathbf{G}(\gamma_r(N)),$$

$$\mathbf{C} = \mathbf{H}(\gamma_r(N-1))\mathbf{F}^{-1}(\gamma_r(N-1))\mathbf{F}^{-1}(\gamma_r(N))\mathbf{A} = \mathbf{H}(\gamma_r(N-1))\mathbf{A},$$

которая при любой входной последовательности формирует выходные последовательности, совпадающие с выходными последовательностями цепи $S^c$ на $\gamma_r$, при этом $N$ — периоды системы (1).

Известно [12, 15], что тест $\hat{U}_T$ для системы (2) состоит из двух фрагментов $\hat{U}_{T1}$ и $\hat{U}_{T2}$. При этом он является одновременно тестом для системы (1), но характеризуется существенно меньшей длиной, чем $\hat{U}_T$, и гарантированным списком обнаруживаемых отказов, совпадающим с соответствующим списком для $\hat{U}_T$.

### Инструментальная среда автоматизации построения модели и тестов

Авторами разработана инструментальная среда для автоматизации процедуры построения параллельной сетединамической модели ПВС реального времени, генерации тестовых воздействий и эталонных выходных реакций. Побудительными мотивами к созданию среды послужила высокая размерность проблемы диагностирования при распределенной обработки информации, когда ПВС может состоять из нескольких сотен ПМ.


*Рис. 4*. Этапы функционирования инструментальной среды синтеза средств диагностирования

Этапы функционирования инструментальной среды приведены на рис. 4.

На первом этапе пользователь с использованием графической среды вводит исходные данные об информационном взаимодействии ПВС. Далее осуществляется автоматическое построение сетединамической модели, тестов и эталонной реакции.

После завершения построения модели ПВС проводится формирование входной тестовой и выходной (эталонной) последовательностей. Программа обеспечивает для каждого звена анализируемой цепи генерацию матриц динамики, входа и выхода, удовлетворяющих условиям управляемости и наблюдаемости. Далее формируются матрицы сеансов обмена для модели (1), а затем и матрицы для модели (2), в отношении которой в дальнейшем применяются процедуры синтеза тестов. Итогом работы инструментальной среды является исходный код параллельной модели ПВС со встроенными средствами диагностирования. После добавления кода функциональных задач, проведения компиляции и сборки — полученный проект становится законченным программным продуктом, реализующим распределенную потоковую обработку информации со встроенными средствами тестового диагностирования.

Функциональность среды не исчерпывается диагностической проблематикой и позволяет решать задачи распределения вычислительных ресурсов и планирования на основе графического представления обмена в ПВС.

Рассмотрим функционирование инструментальной среды на простом примере информационного обмена данными между двумя ПМ для первой цепи ($M_{11}$, $M_{21}$) на рис. 3, *а*. В данном случае выполняются три сеанса обмена (ввод, передача между ПМ, выдача), тогда матрицы модели (1) для всех сеансов обмена будут:

$$\mathbf{F}(1)=\begin{bmatrix}\mathbf{E} & 0\\ \mathbf{g}_2\mathbf{h}_1 & \mathbf{f}_2\end{bmatrix},\ \mathbf{F}(2)=\begin{bmatrix}\mathbf{E} & 0\\ 0 & \mathbf{E}\end{bmatrix},\ \mathbf{F}(3)=\begin{bmatrix}\mathbf{f}_1 & 0\\ 0 & \mathbf{E}\end{bmatrix},$$

$$\mathbf{G}(3)=\begin{bmatrix}\mathbf{g}_1\\ 0\end{bmatrix},\ \mathbf{H}(2)=[0\ \ \mathbf{h}_2],\ \mathbf{G}(j)=0,$$

$$j=\overline{1,2};\ \mathbf{H}(j)=0,\ j=\overline{1,3},$$

где $\mathbf{E}$ — единичная матрица; $\mathbf{f}_i$, $\mathbf{g}_i$, $\mathbf{h}_i$, $i=1, 2$ — матрицы динамики, входа и выхода соответственно, для звеньев $M_{11}$ и $M_{12}$ имеют вид:

$$\mathbf{f}_1=\begin{bmatrix}1 & 1\\ 0 & 1\end{bmatrix},\ \mathbf{f}_2=\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix},\ \mathbf{h}_1=\mathbf{h}_2=\mathbf{g}_1=\mathbf{g}_2=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}.$$

Далее для рассматриваемой цепи вычисляются матрицы стационарной системы $\mathbf{A}$, $\mathbf{B}$ и $\mathbf{C}$:

$$\mathbf{A}=\mathbf{F}(3)\mathbf{F}(2)\mathbf{F}(1)=\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 1 & 0 & 0\\ 1 & 0 & 0 & 1\\ 0 & 1 & 1 & 1\end{bmatrix},$$

$$\mathbf{B}=\mathbf{G}(3),\ \mathbf{C}=\mathbf{H}(2)\mathbf{A}=\begin{bmatrix}1 & 0 & 0 & 1\\ 0 & 1 & 1 & 1\end{bmatrix}.$$

С использованием вычисленных матриц $\mathbf{A}$, $\mathbf{B}$ и $\mathbf{C}$ формируются тестовые воздействия и эталонные выходные последовательности: $\hat{U}_{T1}$ (табл. 1) и $\hat{U}_{T2}$ (табл. 2).

Очевидно, что увеличение числа ПМ существенно увеличивает трудоемкость синтеза средств диагностирования и делает актуальным решение задачи с использованием предлагаемой инструментальной среды.

*Таблица 1.* Тестовые входные последовательности и эталонные выходные последовательности первого фрагмента теста $\hat{U}_{T1}$

| Тестовая входная последовательность | Эталонная выходная последовательность |
|---|---|
| $\mathbf{u}_{in}^1=\begin{bmatrix}0&1&0&0\\0&0&0&0\end{bmatrix}$ | $\mathbf{u}_{out}^1=\begin{bmatrix}0&0&1&1\\0&0&0&1\end{bmatrix}$ |
| $\mathbf{u}_{in}^2=\begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix}$ | $\mathbf{u}_{out}^2=\begin{bmatrix}0&0&0&0\\0&0&1&0\end{bmatrix}$ |
| $\mathbf{u}_{in}^3=\begin{bmatrix}1&1&0&0\\0&1&0&0\end{bmatrix}$ | $\mathbf{u}_{out}^3=\begin{bmatrix}0&1&0&1\\1&0&1&1\end{bmatrix}$ |
| $\mathbf{u}_{in}^4=\begin{bmatrix}0&0&0&0\\0&0&0&0\end{bmatrix}$ | $\mathbf{u}_{out}^4=\begin{bmatrix}1&0&1&1\\0&1&1&0\end{bmatrix}$ |

*Таблица 2.* Тестовые входные последовательности и эталонные выходные последовательности второго фрагмента теста $\hat{U}_{T2}$

| Тестовая входная последовательность | Эталонная выходная последовательность |
|---|---|
| $\mathbf{u}_{in}^1=\begin{bmatrix}1&0&0\\0&0&0\end{bmatrix}$ | $\mathbf{u}_{out}^1=\begin{bmatrix}1&1&0\\1&1&1\end{bmatrix}$ |
| $\mathbf{u}_{in}^2=\begin{bmatrix}0&0&0\\1&0&0\end{bmatrix}$ | $\mathbf{u}_{out}^2=\begin{bmatrix}1&1&0\\0&0&0\end{bmatrix}$ |

## Заключение

В статье рассмотрен вопрос диагностирования нарушений в процессе проектирования систем обработки информации и управления с использованием тестового подхода. Проведена процедура построения параллельной сетединамической модели потоковой вычислительной системы, которая используется для синтеза тестовых последовательностей для обнаружения нарушений в адресации информационных обменов между программными модулями.

В виду высокой сложности рассматриваемого класса систем, параллельности вычислительного процесса и многообразия причин возникновения нарушений разработана инструментальная среда, обеспечивающая автоматизацию процедуры построения модели, генерации тестовых воздействий и эталонных выходных последовательностей, и позволяющая сократить время проектирования средств диагностирования для потоковых вычислительных систем реального времени.

Приведено описание структуры инструментальной среды и результат апробации предложенных алгоритмов на простом примере.

Реализованный в инструментальной среде алгоритм синтеза диагностической модели не обладает достаточной общностью, и по-прежнему существуют вопросы, связанные с оптимизацией количества передаваемой в системе диагностической информацией, являясь темой дальнейших исследований и задачей расширения функциональности среды автоматизации построения средств диагностирования.

## Литература


1. Issues of Fault Diagnosis for Dynamic Systems / ed. by R.J. Patton, P.M. Frank, R.N. Clark. London: Springer-Verlag, 2000. 597 p.
2. Isermann R. Fault-Diagnosis Application. Heidelberg: Springer, 2011. 354 p.
3. Мироновский Л.А. Функциональное диагностирование динамических систем. М.-СПб.: Изд-во МГУ-ГРИФ, 1998. 256 с.
4. Шумский А.Е., Жирабок А.Н., Гаджиев Ч. Диагностирование и отказоустойчивое управление динамическими системами. Владивосток: Дальневосточный федеральный университет, 2016. 178 с.
5. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Теория соответствия для систем с блокировками и разрушениями. М.: Физматлит, 2008. 412 с.
6. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Использование конечных автоматов для тестирования программ // Программирование. 2000. № 2. С. 12–28.
7. Сперанский Д.В. Лекции по теории экспериментов с конечными автоматами. М.: Национальный Открытый Университет «ИНТУИТ», 2016. 354 с.
8. Колесов Н.В., Толмачева М.В., Юхта П.В. Системы реального времени. Планирование, анализ, диагностирование. СПб: ОАО «Концерн «ЦНИИ «Электроприбор», 2014. 180 с.
9. Грузликов А.М., Колесов Н.В. Дискретно-событийная диагностическая модель распределенной вычислительной системы. Независимые цепи // Автоматика и телемеханика. 2016. № 10. С. 140–155.
10. Грузликов А.М., Колесов Н.В. Дискретно-событийная диагностическая модель для распределенной вычислительной системы. Слияние цепей // Автоматика и телемеханика. 2017. № 4. С. 126–134.
11. Gruzlikov A.M., Kolesov N.V., Lukoyanov E.V., Tolmacheva M.V. Test-based diagnosis of distributed computer system using a time-varying model // IFAC-PapersOnLine. 2018. V. 51. N 24. P. 1075–1082. doi: 10.1016/j.ifacol.2018.09.724
12. Грузликов А.М., Колесов Н.В., Лукоянов Е.В. Тестовое диагностирование нарушений адресации информационных обменов в вычислительных системах с использованием параллельной модели // Известия РАН. Теория и системы управления. 2018. № 3. С. 76–89. doi: 10.7868/S0002338818030071
13. Cassandras C.G., Lafortune S. Introduction to Discrete Event Systems. 2nd ed. New York: Springer, 2008. 770 p.
14. Zaytoon J., Lafortune S. Overview of fault diagnosis methods for discrete event systems // Annual Reviews in Control. 2013. V. 37. N 2. P. 308–320. doi: 10.1016/j.arcontrol.2013.09.009
15. Гилл А. Линейные последовательностные машины: Анализ, синтез и применение. М.: Наука, 1974. 288 с.


## References


1. *Issues of Fault Diagnosis for Dynamic Systems*. Ed. by R.J. Patton, P.M. Frank, R.N. Clark. London, Springer-Verlag, 2000, 597 p.
2. Isermann R. *Fault-Diagnosis Application*. Heidelberg, Springer, 2011, 354 p.
3. Mironovskii L.A. *Functional Diagnosis of Dynamic-Systems*, Moscow, St. Petersburg, Moscow State University Publ., 1998, 256 p. (in Russian)
4. Shumsky A.Ye., Zhirabok A.N., Hadjiev C. *Diagnosis and Fault Tolerant Control in Dynamic Systems*. Vladivostok, Far Eastern Federal University, 2016, 178 p. (in Russian)
5. Burdonov I.B., Kosachev A.S., Kuliamin V.V. *Compliance Theory for Lock and Crash Systems*. Moscow, Fizmatlit Publ., 2008, 412 p. (in Russian)
6. Burdonov I.B., Kossatchev A.S., Kulyamin V.V. Application of finite automatons for program testing. *Programming and Computer Software*, 2000, vol. 26, no. 2, pp. 61–73. doi: 10.1007/BF02759192
7. Speranskii D.V. *Lectures on Theory of Experiments With Finite Automata*. Moscow, INTUIT Publ., 2016, 354 p. (in Russian)
8. Kolesov N.V., Tolmacheva M.V., Yukhta P.V. *Real-Time Systems. Planning, Analysis, Diagnostics*. St. Petersburg, Concern CSRI Elektropribor, 180 p. (in Russian)
9. Gruzlikov A.M., Kolesov N.V. Discrete-event diagnostic model for a distributed computational system. Independent chains. *Automation and Remote Control*, 2016, vol. 77, no. 10, pp. 1805–1817. doi: 10.1134/S0005117916100076
10. Gruzlikov A.M., Kolesov N.V. Discrete-event diagnostic model for a distributed computational system. Merging chains. *Automation and Remote Control*, 2017, vol. 78, no. 4, pp. 682–688. doi: 10.1134/S0005117917040099
11. Gruzlikov A.M., Kolesov N.V., Lukoyanov E.V., Tolmacheva M.V. Test-based diagnosis of distributed computer system using a time-varying model. *IFAC-PapersOnLine*, 2018, vol. 51, no. 24, pp. 1075–1082. doi: 10.1016/j.ifacol.2018.09.724
12. Gruzlikov A.M., Kolesov N.V., Lukoyanov E.V. Test-based diagnosis of faults in data exchange addressing in computer systems using parallel model. *Journal of Computer and Systems Sciences International*, 2018, vol. 57, no. 3, pp. 420–433. doi: 10.1134/S1064230718030024
13. Cassandras C.G., Lafortune S. *Introduction to Discrete Event Systems*. 2nd ed. New York, Springer, 2008, 770 p.
14. Zaytoon J., Lafortune S. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 2013, vol. 37, no. 2, pp. 308–320. doi: 10.1016/j.arcontrol.2013.09.009
15. Gill A. *Linear Sequential Circuits: Analysis, Synthesis, and Applications*. McGraw-Hill, 1966, 215 p.

**Авторы**

**Лукоянов Егор Васильевич** — младший научный сотрудник, АО «Концерн «ЦНИИ «Электроприбор», Санкт-Петербург, 197046, Российская Федерация, Scopus ID: 57193712278, ORCID ID: 0000-0003-3882-4315, lukoyanov.egor@mail.ru

**Грузликов Александр Михайлович** — кандидат технических наук, начальник отдела, АО «Концерн «ЦНИИ «Электроприбор», Санкт-Петербург, 197046, Российская Федерация, Scopus ID: 56037536900, ORCID ID: 0000-0001-8814-0726, agruzlikov@yandex.ru

**Authors**

**Egor V. Lukoyanov** — Junior Researcher, CSRI Elektropribor, JSC, Saint Petersburg, 197046, Russian Federation, Scopus ID: 57193712278, ORCID ID: 0000-0003-3882-4315, lukoyanov.egor@mail.ru

**Alexander M. Gruzlikov** — PhD, Department Head, CSRI Elektropribor, JSC, Saint Petersburg, 197046, Russian Federation, Scopus ID: 56037536900, ORCID ID: 0000-0001-8814-0726, agruzlikov@yandex.ru