

СИСТЕМЫ АВТОМАТИЗАЦИИ В НАУЧНЫХ ИССЛЕДОВАНИЯХ И ПРОМЫШЛЕННОСТИ

УДК 004.415.5 : 004.4'2

АВТОМАТИЧЕСКАЯ ВЕРИФИКАЦИЯ АЛГОРИТМОВ УПРАВЛЕНИЯ В КИБЕРФИЗИЧЕСКИХ СИСТЕМАХ НА ПРОГРАММНЫХ ИМИТАТОРАХ

© Т. В. Лях^{1,2}, В. Е. Зюбин^{1,2}, Н. О. Гаранина³

¹Институт автоматизации и электрометрии СО РАН,
630090, г. Новосибирск, просп. Академика Коптюга, 1

²Новосибирский государственный университет,
630090, г. Новосибирск, ул. Пирогова, 2

³Институт систем информатики СО РАН,
630090, г. Новосибирск, просп. Академика Лаврентьева, 6
E-mail: zzubin@iae.nsk.su

Решается задача автоматической верификации алгоритмов управления, созданных средствами процесс-ориентированного программирования для киберфизических систем. Предлагается метод на основе программных имитаторов объекта управления и описывается его реализация на базе пакета LabVIEW и транслятора языка Reflex.

Ключевые слова: верификация, алгоритмы управления, киберфизические системы, имитационное моделирование, процесс-ориентированное программирование.

DOI: 10.15372/AUT20190211

Введение. Алгоритмы управления в киберфизических системах, как и алгоритмы управления в области промышленной автоматизации, обладают специфическими свойствами, которые обуславливают необходимость использования проблемно ориентированных языков [1–4]:

— открытостью — наличием внешнего физического мира, с которым алгоритм управления (кибернетическая часть системы) взаимодействует через датчики и исполнительные органы;

— событийностью — зависимостью реакции системы управления от событий на объекте управления и управляющих команд оператора;

— неопределённой продолжительностью функционирования;

— синхронизмом — временной согласованностью реакции алгоритма управления и событий на объекте управления;

— логическим параллелизмом — наличием в алгоритме управления множества логически независимых потоков управления, отражающих множество физически параллельных процессов на объекте управления.

Реализация алгоритмов управления средствами объектно-ориентированных языков программирования общего назначения чревата чрезмерным усложнением программной архитектуры при росте сложности алгоритма [5]. Поэтому в области промышленной автоматизации применяются специализированные языковые средства для разработки алгоритмов управления: языки МЭК 61131-3, G (NI LabVIEW), Reflex [4, 6, 7].

Использование языков, входящих в спецификацию МЭК 61131-3, является трудоёмким из-за низкой выразительности при описании сложных алгоритмов управления в киберфизических системах, а в некоторых случаях и неприемлемым, например при необходимости интеграции кода в сторонние системы [4, 6–8].

Исследователи альтернативных лингвистических средств для описания алгоритмов управления в киберфизических системах [2, 4, 8–11] предлагают и практически обосновывают эффективность предметно-ориентированных языков, использующих модель конечного автомата, в частности эффективность процесс-ориентированных языков Reflex и IndustrialC [10, 11].

При применении языков на базе конечных автоматов в промышленной автоматизации [8, 9, 12] основную проблему представляет решение задач тестирования и верификации созданных алгоритмов, поскольку методы, разработанные для тестирования и верификации программного обеспечения в области объектно-ориентированного программирования, слабо применимы [5]. Свойство открытости означает, что управляющий алгоритм невозможно использовать автономно, в том числе в целях эмпирического исследования. Однако исследование алгоритма на реальном объекте управления может привести к критической ситуации (поломке оборудования, аварии и т. п.). Поэтому наиболее распространённый подход — ручная проверка на этапе пусконаладки. Специалист, проводящий ручную проверку, по сути, выступает в качестве имитатора объекта управления: сначала задаёт значения на входах алгоритма управления, а затем контролирует значения выходных сигналов. Такой способ в силу сложности реализации позволяет провести только контроль трассировки проводов и верификацию самых простых свойств алгоритма. Отсутствие контроля качества создаваемых алгоритмов управления серьёзно увеличивает риски проектов по созданию киберфизических систем. Поэтому разработка методов автоматической верификации алгоритмов управления актуальна не только с теоретической стороны, но и с практической [5, 13–17].

Методы, разрабатываемые в области статической верификации киберфизических систем, на текущем этапе сопряжены с серьёзными ограничениями на сложность верифицируемых алгоритмов, которые делают эти методы фактически неприменимыми для практических задач. Констатация этого факта является общей в работах по теме верификации киберфизических и гибридных систем [18–22].

Современная тенденция при разработке алгоритмов управления (АУ) — проводить динамическую верификацию с использованием симуляторов — программных имитаторов объекта управления [12, 23–25]. Однако в настоящее время такая верификация характеризуется языковой гетерогенностью (использованием нескольких языков), высокой трудоёмкостью создания симуляторов и большим объёмом ручных операций, что мотивирует исследователей искать новые подходы. Исследования методов динамической верификации управляющих алгоритмов [26] показывают, что трудоёмкость динамической верификации может быть существенно снижена при применении средств процесс-ориентированного программирования.

В предлагаемой работе описываются общая схема итерационной разработки управляющих алгоритмов на основе симуляторов и метод автоматизированной верификации, формулируется основная задача, предлагается схема автоматической верификации управляющих алгоритмов, созданных средствами процесс-ориентированного программирования и демонстрируется использование предлагаемого подхода на практическом примере.

Постановка задачи. Общая схема верификации (рис. 1), предложенная в [12], включает последовательность шагов:

- верифицируемый алгоритм управления (его часть) реализуется программно и оформляется в обособленный алгоритмический блок (алгоблок);
- модель технологического объекта (его часть) реализуется программно и оформляется в обособленный алгоблок, называемый виртуальным объектом управления;
- осуществляются верификация путём создания тестовых ситуаций (сценариев) и контроль реакции алгоритма управления;

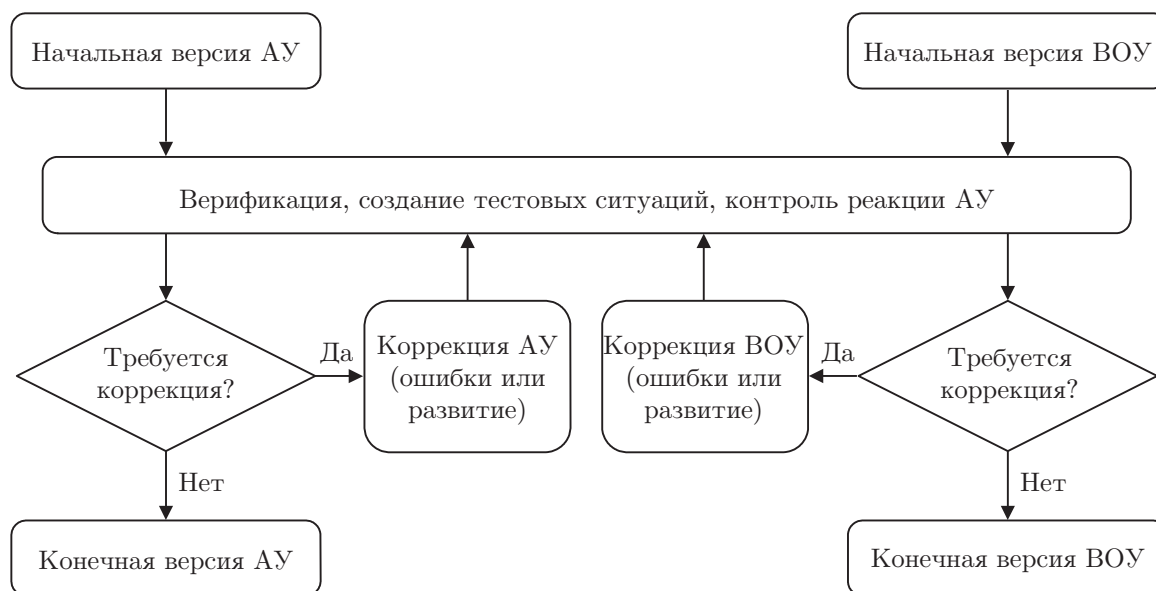


Рис. 1. Общая схема верификации алгоритмов управления технологическим объектом: АУ — алгоритм управления, ВОУ — виртуальный объект управления

— по результатам верификации либо корректируется код алгоритма управления и (или) виртуального объекта управления и проводится новая верификация, возможно, с дополнительными тестовыми ситуациями, либо верификация считается завершённой и фиксируется конечная версия алгоритма управления.

Схема позволяет применять итерационный подход к разработке промышленных алгоритмов управления сложными технологическими объектами. В упрощённом виде схема была опробована при создании набора виртуальных лабораторных стендов для обучения студентов, специализирующихся в области промышленной автоматизации [12]. Упрощение заключается в неизменности ВОУ. При этом ВОУ включает графическую анимацию для повышения наглядности, а корректность алгоритма контролируется визуально.

Несмотря на высокую эффективность использования метода в учебном процессе, он не нашёл практического применения в реальных проектах в силу относительно высокой трудоёмкости создания графических моделей ВОУ и их последующей модификации.

В работе [5] был предложен способ автоматизированной верификации алгоритмов управления, в котором управление сценариями работы и контроль реакции алгоритма проводятся оператором через универсальный графический интерфейс. Графический интерфейс предоставляет оператору следующие возможности: отправлять команды алгоритму управления, контролировать сообщения от алгоритма управления, управлять поведением ВОУ, контролировать сообщения от ВОУ.

Достоинства решения — единый способ спецификации АУ и ВОУ средствами языка Reflex, независимость внутренней структуры программного комплекса и интерфейса от верифицируемого алгоритма и объекта управления, а также отсутствие ручных операций при получении исполняемого кода из верифицированной спецификации [27].

Решение практически апробировано в процессе автоматизации Большого солнечного вакуумного телескопа (пос. Листвянка Иркутской области) [5, 28]. В проекте был создан и верифицирован алгоритм управления системой вакуумирования. Верифицировались работа алгоритма при создании вакуума в трубе телескопа и развакуумировании перед техническими работами, реакция алгоритма на изменения температуры окружающей среды,

уровня воды в системе охлаждения и реакция алгоритма при разгерметизации трубы телескопа. При этом имитировались отказы отсечных клапанов, вакуумных заслонок, насосов, вытяжных вентиляторов, датчиков, исполнительных органов системы климат-контроля и т. д.

Верификация была проведена на территории разработчика и обеспечила значительное сокращение общей трудоёмкости работ, в частности продолжительности пусконаладки на целевом объекте.

Недостатки метода:

— визуальный контроль за реакцией алгоритма и сложность анализа отображаемой информации приводят к ошибкам верификации (вероятность пропустить ошибку в работе алгоритма);

— на каждой итерации должна быть заново проверена реакция алгоритма в соответствии со списком тестовых ситуаций, что означает большое количество рутинных действий и повышает вероятность ошибки оператора (пропуск тестовых ситуаций).

Таким образом, в работе была поставлена задача усовершенствовать существующий метод верификации алгоритмов управления в киберфизических системах на симуляторах и предложить решение, которое исключало бы рутинные операции по созданию тестовых ситуаций и их контролю.

Предлагаемое решение. В качестве решения данной задачи была предложена схема автоматической верификации алгоритмов управления в киберфизических системах на симуляторах (рис. 2, табл. 1). В отличие от автоматизированной верификации управление сценариями работы производится алгоритмическим блоком управления сценариями (БУС) 9, а контроль реакции алгоритма — алгоритмическим блоком верификации (БВ) 2. Через штатную очередь сообщений 8 БУС посылает команды для АУ 4, имитируя действия оператора, а также управляет поведением ВОУ 7 через очередь сообщений 10. При этом ВОУ эмулирует как штатную работу объекта управления, так и отказы его элементов. Параллельно БУС передаёт информацию о запускаемых сценариях в БВ через

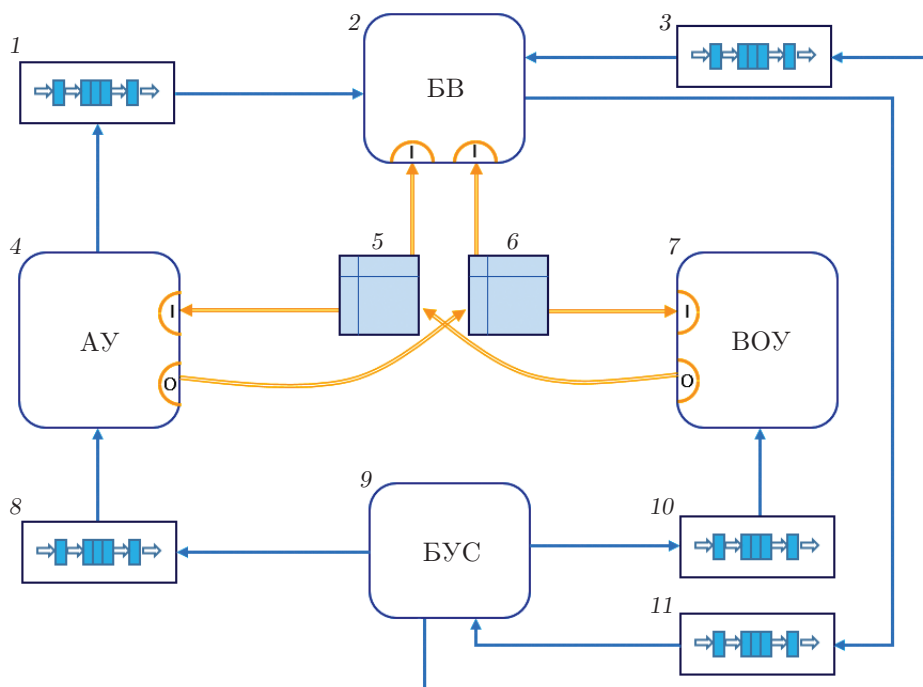



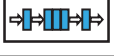




Рис. 2. Схема автоматической верификации алгоритмов управления

Таблица 1

Используемые условные обозначения	
	Алгоблок
	Цифровые порты алгоритмического блока (I — входные, O — выходные)
	Буфер значений дискретных цифровых сигналов
	Очередь сообщений
	Цифровые данные
	Сообщения

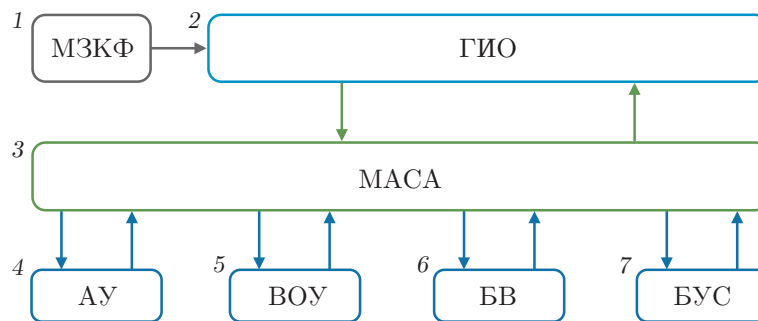


Рис. 3. Архитектура программного комплекса автоматической верификации алгоритмов управления

ряд сообщений 3. В свою очередь, БВ на основании информации о запускаемых сценариях, сообщений от АУ, получаемых через буфер сообщений 1, и состояния буферов входных/выходных сигналов АУ 5, 6 определяет корректность АУ и через очередь сообщений от БВ к БУС 11 передаёт информацию о готовности к началу верификации.

Программный комплекс автоматической верификации алгоритмов управления (рис. 3) реализован в виде LabVIEW-приложения. Комплекс включает графический интерфейс оператора (ГИО), который конфигурируется через модуль загрузки конфигурационных файлов (МЗКФ) и взаимодействует через модуль активации и синхронизации алгоблоков (МАСА) с целевыми модулями алгоблоков АУ, ВОУ, БУС и БВ. Модули загрузки конфигурационных файлов и ГИО реализованы на языке G LabVIEW, а МАСА — в виде DLL на языке C++. Исполняемые модули АУ, ВОУ, БУС и БВ генерируются из описания на языке Reflex и в виде DLL загружаются модулем МАСА.

Проект состоит из файлов описания алгоблоков АУ, ВОУ, БВ, БУС, графического файла со схемой объекта управления и XML-файла описания сигнальных элементов на схеме объекта управления. Файлы проекта располагаются в отдельной директории.

После запуска программного комплекса на исполнение открывается окно верификации (рис. 4), которое разделено на три области: меню управления проектом 1, панель управления верификацией 2 и область графического представления объекта управления 3.

Через меню управления проектом пользователь указывает директорию существующего проекта или создаёт новый проект. Через панель управления верификацией пользователю предоставляется возможность:

- запуска редактора алгоблоков на языке Reflex и их трансляции;
- запуска, прерывания и приостановки верификации;

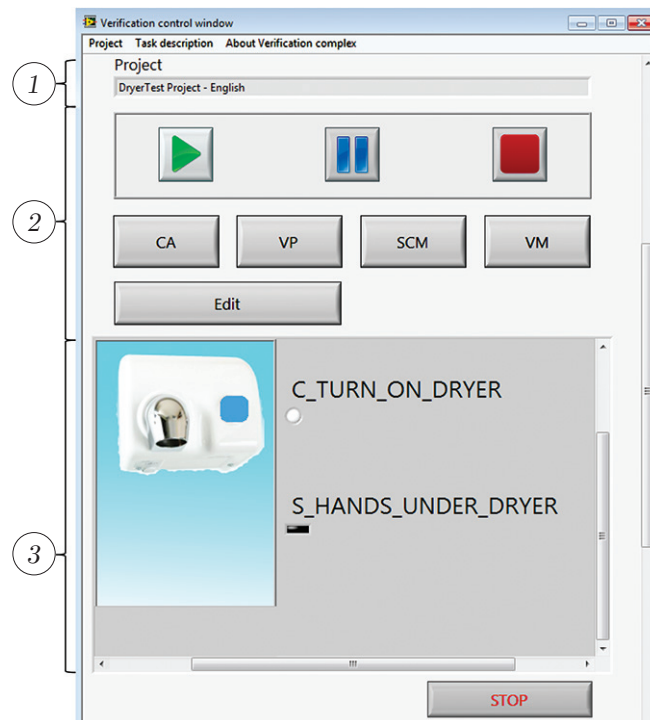


Рис. 4. Графический интерфейс оператора — окно верификации

— вызова панелей АУ, ВОУ, БУС и БВ для просмотра внутренних состояний алгоблоков.

На панели алгоблоков (рис. 5) отображаются:

- текущие значения внутренних переменных алгоблока 1;
- текущие значения входных/выходных сообщений алгоблока 2;
- текущие состояния процессов алгоблока 3.

При запуске верификации ГИО посылает МАСА запрос на подключение DLL-алгоблоков. Если DLL отсутствуют, выдаётся диагностическое сообщение об ошибке, иначе алгоблоки подключаются и через МЗКФ (см. рис. 3, 1) производится инициализация панелей АУ, ВОУ, БУС, БВ и области графического представления объекта управления (см. рис. 4, 3).

При подключении DLL МАСА передаёт им указатели на очереди сообщений, буферы входных и выходных переменных и состояния процессов. Оперативная память под процессы, порты и сообщения резервируется алгоблоками при запуске на исполнение. Во время верификации алгоблоки активируются циклически в порядке БУС, ВОУ, АУ, БВ. Результаты верификации отображаются на панели БВ. Итерационная разработка ведётся по стандартной схеме (см. рис. 1).

Предусмотрено два режима работы программы: первый предполагает тактирование цикла активации и соответствие модели динамическим характеристикам объекта, что позволяет визуально контролировать процесс верификации, во втором режиме задержка между циклами активации обнуляется, что даёт возможность радикально (до двух порядков) сократить продолжительность верификации.

Практическая апробация метода. Предложенный метод и реализованный инструментальный практически апробированы в проекте по модернизации виртуальных лабораторных стендов в Новосибирском государственном университете.

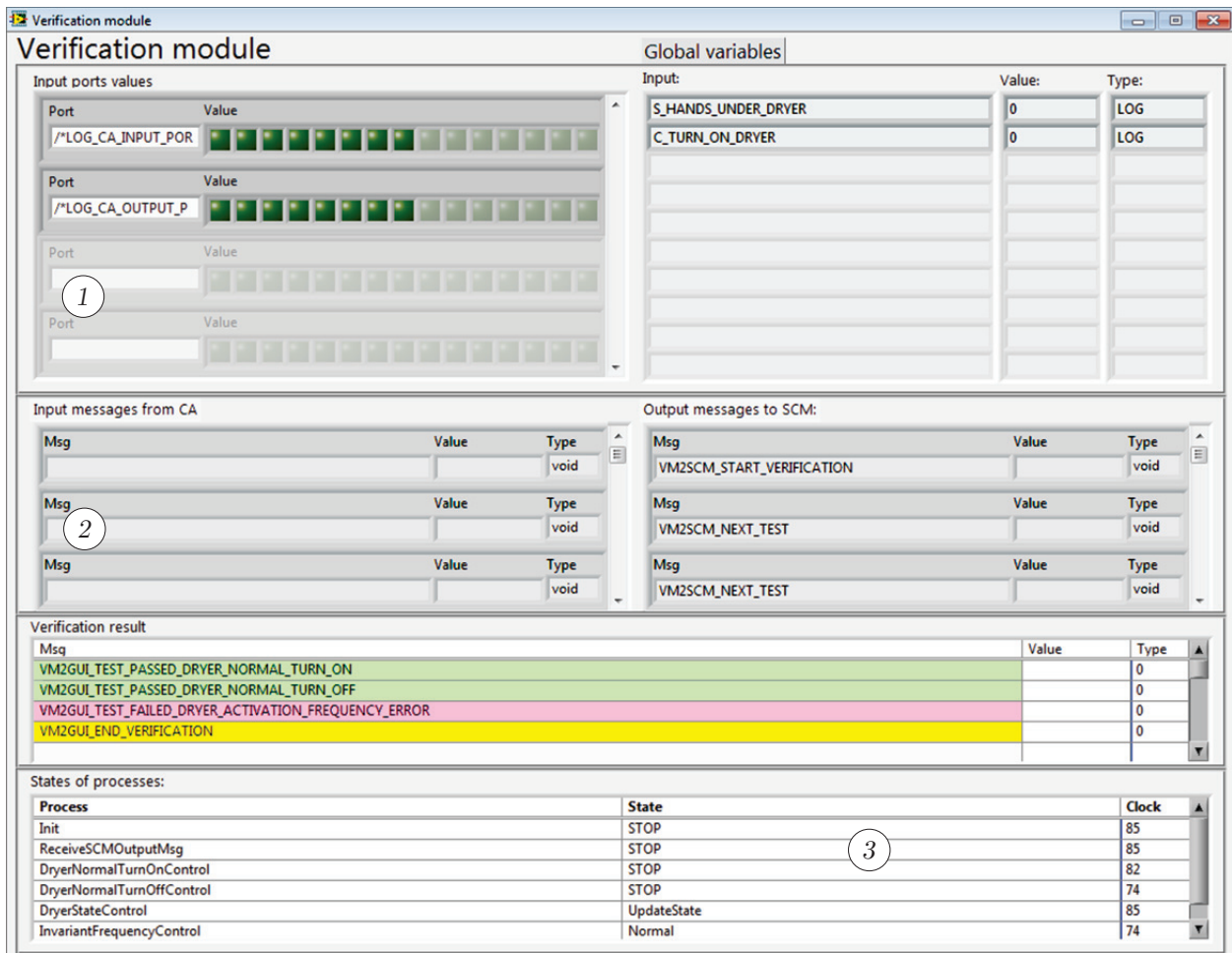


Рис. 5. Графический интерфейс оператора

Эмпирическая проверка работы комплекса верификации проводилась на алгоритме управления тепловентилятором для сушки свежескрашенных изделий.

Алгоритм использует входной сигнал от ИК-датчика, показывающий наличие изделия под тепловентилятором, и управляет нагревательным элементом с феном через выходной сигнал. Состоянию «изделие под тепловентилятором» соответствует значение ВКЛЮЧЕНО, состоянию «изделие убрано» — ВЫКЛЮЧЕНО. Основное требование к алгоритму управления — включить тепловентилятор при наличии изделия и автоматически выключать его после того, как изделие удалено. ИК-датчик время от времени показывает кратковременное отсутствие изделия. Эти кратковременные сигналы программа должна игнорировать, поскольку частые переключения не только неудобны пользователю, но и значительно снижают срок службы устройства. Однако программа должна своевременно выключать тепловентилятор, поскольку его работа вхолостую не только связана с лишним энергопотреблением, но также сокращает срок службы. Алгоритм может выполнить эти требования только через измерение и анализ продолжительности состояния ВЫКЛЮЧЕНО. В рассматриваемом случае граничная продолжительность принимается равной 1 секунде. Код АУ на языке Reflex, реализующий описанный алгоритм, представлен в листинге.

Модель объекта управления тепловентилятором реализуется на языке Reflex. Появление изделия под тепловентилятором имитируется установкой сигнала датчика с исчезнове-

```

PROGR HandDryerController {
  TACT 100;
  CONST ON 1;
  CONST OFF 0; /* direction, name, address, offset, size of the port */
  INPUT SENSOR_INPUT_PORT 0 0 8; /* IR sensor input port */
  OUTPUT ACTUATOR_OUTPUT_PORT 1 0 8; /* output to heater and fan */
  PROC Init {
    BOOL S_HANDS_UNDER_DRYER = {SENSOR_INPUT_PORT [1]} FOR ALL;
    BOOL C_TURN_ON_DRYER = {ACTUATOR_OUTPUT_PORT [1]} FOR ALL;
    STATE Waiting {
      IF (S_HANDS_UNDER_DRYER == ON) {
        C_TURN_ON_DRYER = ON;
        SET NEXT;
      } ELSE {
        C_TURN_ON_DRYER = OFF;
      }
    }
    STATE Drying {
      IF (S_HANDS_UNDER_DRYER == ON) RESET TIMEOUT;
      TIMEOUT 10 {SET STATE Waiting;}
    }
  } /* \PROC */
} /* \PROGRAM */

```

Таблица 2

№ п/п	MTL, clock = 100 ms	Пояснение
1	$(\text{hands} \rightarrow \diamond (1, 2) \text{ dryer})$	При появлении изделия тепловентилятор включится не позднее 0,2 с
2	$((\neg \text{hands} \wedge \text{dryer}) \mathbf{U} \{10\} \neg \text{dryer})$	Если изделие убрали, то через 1 с тепловентилятор выключится
3	$((\neg \text{hands} \wedge \neg \text{dryer}) \mathbf{W} \text{ hands})$	Тепловентилятор спонтанно не включается

нием сигнала продолжительностью от 100 до 900 мс. Запуск имитации и её прекращение осуществляется ВОУ по сообщению БУС. Сигнал датчика о наличии изделия является для объекта управления выходным.

К алгоритму управления тепловентилятором предъявляются требования, выраженные в терминах метрической темпоральной логики (табл. 2) [29].

Для проверки каждого из требований создаётся отдельный процесс, который запускается по соответствующей команде от БУС. Например, процесс проверки требования 1 (табл. 2) при появлении сигнала от датчика либо фиксирует включение тепловентилятора, либо по прошествии тайм-аута фиксирует ошибку в алгоритме. В первом случае БВ сначала посылает сообщение ГИО об успешном прохождении теста, а затем посылает сообщение БУС о запросе следующего теста и переходит в состояние нормального останова. В случае ошибки БВ сначала посылает сообщение ГИО об ошибке прохождения теста, а затем посылает сообщение БУС с запросом следующего теста и переходит в состояние нормального останова. Сигналы от датчика и тепловентилятора являются для БВ входными.

Действия БУС заключаются в последовательной выдаче команд ВОУ (о переводе ВОУ в соответствующие режимы симуляции действий пользователя) и БВ (о запуске соответствующего процесса проверки АУ) и последующем ожидании от БВ сообщения об окон-

чании теста. После запуска последнего теста из списка комплекс автоматической верификации завершает работу.

Ограничений на использование разработанного метода для целей динамической верификации киберфизических систем не выявлено.

Определённые неудобства вызывает трансформация темпоральных требований в спецификацию БВ, поскольку она производится вручную.

Заключение. В работе предложен метод для автоматической верификации алгоритмов управления в киберфизических системах, который предполагает унифицированное описание на языке Reflex алгоритма управления, симулятора объекта управления, тестовых сценариев и процедуры контроля реакции алгоритма управления на внешние события.

Управление сценариями работы, имитация действий оператора системы управления и контроль реакции алгоритма производятся автоматически. Межблочное взаимодействие организовано через массивы и очереди сообщений. При этом эмулируется как штатная работа объекта управления, так и отказы его элементов. Предусмотрены средства визуального контроля поведения алгоблоков. Программный комплекс, реализующий метод, выполнен на базе пакета LabVIEW.

Кроме собственно автоматической верификации, предложенный метод обеспечивает отсутствие ручных операций при генерации исполняемого кода и позволяет использовать итерационные методики при создании киберфизических систем.

Практическая апробация метода проведена на тестовой задаче верификации алгоритма управления тепловентилятором для сушки свежеокрашенных изделий.

При видимой универсальности подхода зафиксированы определённые неудобства при преобразовании требований, выраженных в терминах темпоральной логики, в спецификацию БВ на языке Reflex. Далее планируется исследовать это обстоятельство и предложить решение по автоматизированному преобразованию пользовательских требований в спецификацию блока верификации.

Финансирование. Работа выполнена при финансовой поддержке Министерства высшего образования и науки РФ (государственная регистрация № АААА-А17-117060610006-6) и Российского фонда фундаментальных исследований (проект № 17-07-01600)

СПИСОК ЛИТЕРАТУРЫ

1. **Закревский А. Д.** Параллельные алгоритмы логического управления. М.: Эдиториал УРСС, 2003. 200 с.
2. **Wagner F., Schmuki R., Wagner T., Wolstenholme P.** Modeling Software With Finite State Machines: A Practical Approach. N. Y.: Auerbach Publications, 2006. 390 p.
3. **Harel D.** Statecharts: A visual formalism for complex systems // *Sci. Comput. Programm.* 1987. **8**, N 3. P. 231–274.
4. **Zyubin V. E.** Hyper-automaton: A model of control algorithms // *Proc. of the IEEE Intern. Siberian Conf. on Control and Communications (SIBCON 2007)*. Tomsk, Russia, 20–21 April, 2007. P. 51–57. DOI: 10.1109/SIBCON.2007.371297.
5. **Liakh T. V., Zyubin V. E.** The Reflex language usage to automate the Large Solar Vacuum Telescope // *Proc. of the 17th Intern. Conf. of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM-16)*. Erlagol, Russia, 30 June – 4 July, 2016. P. 137–139. DOI: 10.1109/EDM.2016.7538711.
6. **Basile F., Chiacchio P., Gerbasio D.** On the implementation of industrial automation systems based on PLC // *IEEE Trans. Automation Sci. Eng.* 2013. **10**, N 4. P. 990–1003.

7. **Thramboulidis K.** A cyber-physical system-based approach for industrial automation systems // *Comput. Industry*. 2015. **72**, Is. C. P. 92–102. URL: <http://dx.doi.org/10.1016/j.compind.2015.04.006> (дата обращения: 19.10.2018).
8. **Samek M., Montgomery P.** State oriented programming // *Embedded Syst. Programm.* 2000. **13**, N 8. P. 22–43.
9. **Шалыто А. А., Туккель Н. И.** SWITCH технология — автоматный подход к созданию программного обеспечения «реактивных» систем // *Программирование*. 2001. **27**, вып. 5. С. 45–62.
10. **Rozov A. S., Zyubin V. E.** Process-oriented programming language for MCU-based automation // *Proc. of the IEEE Intern. Siberian Conf. on Control and Communications (SIBCON 2013)*. Krasnoyarsk, Russia, 12–13 Sept., 2013. P. 1–4. DOI: 10.1109/SIBCON.2013.6693595.
11. **Liakh T. V., Rozov A. S., Zyubin V. E.** Reflex language: A practical notation for cyber-physical systems // *System Informatics*. 2018. **2**, N 12. P. 85–104.
12. **Zyubin V.** Using process-oriented programming in LabVIEW // *Proc. of the 2nd IASTED Intern. Multi-Conf. on Automation, Control, and Information Technology: Control, Diagnostics, and Automation*. Novosibirsk, Russia, 15–18 June, 2010. Vol. 1. P. 35–41.
13. **Garanina N., Zyubin V., Lyakh T., Gorlatch S.** An ontology of specification patterns for verification of concurrent systems // *Proc. of the 17th Intern. Conf. SoMeT-18. Ser.: Frontiers in Artificial Intell. and Appl.* Amsterdam: IOS Press, 2018. Vol. 303. P. 515–528. DOI: 10.3233/978-1-61499-900-3-515.
14. **Shilov N. V., Garanina N. O.** Combined logics of knowledge, time, and actions for reasoning about multi-agent systems // *Knowledge Process. and Data Analysis*. 2011. Is. 6581. P. 48–58.
15. **Шелехов В. И.** Верификация и синтез программ сложения на базе правил корректности операторов // *Моделирование и анализ информационных систем*. 2010. **17**, № 4. С. 101–110.
16. **Clarke E. M., Gao S.** Model checking hybrid systems // *Proc. of the 6th Intern. Symp. on Leveraging Applications of Formal Methods, Verification and Validation*. Corfu, Greece, 8–11 Oct., 2014. Is. 8803. P. 385–386.
17. **Drozdov D., Patil S., Dubinin V., Vyatkin V.** Towards formal verification for cyber-physically agnostic software: A case study // *Proc. of the 43rd Annual Conf. of the IEEE Industrial Electronics Society (IECON 2017)*. Beijing, China, 29 Oct. – 1 Nov. P. 5509–5514.
18. **Пакулин Н. В.** Динамическая верификация гибридных систем // *Науч.-техн. ведомости Санкт-Петербургского гос. политехн. ун-та. Информатика. Телекоммуникации. Управление*. 2014. **193**, № 2. С. 189–203.
19. **Степанов О. Г.** Метод автоматической динамической верификации автоматных программ // *Науч.-техн. вестн. Санкт-Петербургского гос. ун-та информац. технологий, механики и оптики*. 2008. **8**, вып. 53. Автоматное программирование. С. 221–229.
20. **Platzer A.** Logic and compositional verification of hybrid systems (invited tutorial) // *Proc. of the Intern. Conf. on Computer Aided Verification*. Snowbird, USA, 14–20 July, 2011. Vol. 1. P. 28–43.
21. **Cassez F., Larsen K. G.** The impressive power of stopwatches // *Proc. of the 11th Intern. Conf. on Concurrency Theory (CONCUR 2000)*. Pennsylvania, USA, 22–25 Aug., 2000. Vol. 1877. P. 138–152.
22. **Henzinger T. A.** The theory of hybrid automata // *Proc. of the 9th Annual IEEE Symp. on Logic in Computer Science*. New Brunswick, USA, 27–30 July, 1996. P. 278–292.
23. **Koziorek J., Ozana S., Srovnal V., Docekalin T.** Modeling and simulations in control software design // *Analytic methods in systems and software testing* /Eds. R. S. Kenett,

- F. Ruggeri, F. W. Faltin. 2018. Oxford: John Wiley & Sons Ltd. P. 287–326. DOI: 10.1002/9781119357056.ch12.
24. **Isermann R., Schaffnit J., Sinsel S.** Hardware-in-the-loop simulation for the design and testing of engine-control systems // Control Eng. Practice. 1999. **7**, N 5. P. 643–653.
 25. **Xiao B., Starke M., King D. et al.** Implementation of system level control and communications in a Hardware-in-the-Loop microgridtestbed // Proc. of the IEEE Power & Energy Society Innovative Smart Grid Technologies Conf. Minnesota, USA, 6–9 Sept., 2016. Vol. 1. P. 1–5.
 26. **Лях Т. В., Зюбин В. Е., Гаранина Н. О.** Автоматизированная верификация алгоритмов управления сложными технологическими объектами на программных имитаторах // Вестник НГУ. Сер. Информационные технологии. 2018. **16**, № 4. С. 85–94. DOI: 10.25205/1818-7900-2018-16-4-85-94.
 27. **Zyubin V. E.** Information complexity hypothesis: A conceptual framework for reasoning on pragmatics issues // Proc. of the IEEE Intern. Conf. on Computational Technologies in Electrical and Electronics Engineering. Novosibirsk, Russia, 21–25 July, 2008. Vol. 1. P. 272–275. DOI: 10.1109/SIBIRCON.2008.4602608.
 28. **Ковадло П. Г., Лубков А. А., Бевзов А. Н. и др.** Система автоматизации Большого солнечного вакуумного телескопа // Автометрия. 2016. **52**, № 2. С. 97–106.
 29. **Ouaknine J., Worrell J.** Some recent results in Metric Temporal Logic // Proc. of the 6th Intern. Conf. on Formal Modelling and Analysis of Timed Systems (FORMATS 2008). Saint Malo, France, 15–17 Sept., 2008. Vol. 1. P. 1–13. DOI: 10.1007/978-3-540-85778-5_1.

Поступила в редакцию 19.10.2018

После доработки 27.12.2018

Принята к публикации 17.01.2019
