УДК 622.276:681.5.09

Разработка и тестирование автоматизированных систем управления процессами добычи и подготовки нефти на основе временных автоматов

А.Г. Зебзеев (Томск, Россия)
zebzeevag@gmail.com

Инженер отдела автоматизации технологических процессов ОАО «ТомскНИПИнефть»

Д.В. Журавлев

Инженер отдела автоматизации технологических процессов ОАО «ТомскНИПИнефть»

Е.И. Громаков

К.т.н., доцент кафедры интегрированных компьютерных систем управления НИ ТПУ

М.А. Пушкарев

Начальник отдела автоматизации технологических процессов ОАО «ТомскНИПИнефть»

Предложен подход к разработке алгоритмов автоматизированных систем управления и тестированию программных реализаций на соответствие разработанным алгоритмам на основе временных автоматов. Выявлены преимущества автоматных моделей как языка алгоритмизации. Рассмотрены опыт и разработки ОАО «ТомскНИПИнефть» в применении автоматных моделей при проектировании автоматизированных систем управления процессами добычи и подготовки нефти. Показана необходимость тестирования программного кода с полным выявлением ошибок.

Материалы и методы
Теория автоматов, дискретная математика, теория графов.

Ключевые слова
временной автомат, автоматизированные системы управления, разработка алгоритмов, тестирование программной реализации

Automated control systems of oil extraction and treatment processes development and testing based on timed finite state machines

В условиях постоянного увеличения степени автоматизации процессов добычи и подготовки ископаемых углеводородов, ужесточения требования к качеству и безопасности этих процессов, вопросы оптимальной разработки, реализации и тестирования алгоритмов и программ управления с высокой степенью точности стоят наиболее остро. Зачастую разработчиками математического обеспечения (МО) автоматизированных систем управления технологическими процессами (АСУ ТП) и программного обеспечения (ПО) АСУ ТП являются различные организации. Данное обстоятельство определяет риск создания программных реализаций с логикой, не соответствующей разработанным алгоритмам МО.

Стоит отметить, что корректность работы всей системы управления напрямую зависит от того, насколько правильно написана программная реализация. При этом выполнение тестирования ПО на соответствие логике алгоритмов является зачастую творческим процессом, полнота которого не гарантируется проверяющим. В этих условиях желательно использовать единый научно-обоснованный метод и инструмент для разработки и тестирования ПО, и такая возможность предоставляется при использовании автоматных моделей.

Большинство процессов управления, происходящих в программируемых логических контроллерах (ПЛК) автоматизированных систем управления можно свести к преобразованию информации. Существует некоторый вход ПЛК, который представляет собой набор сигналов. Комбинация сигналов в каждый момент времени представляет собой символ x входного алфавита X. Алфавит X суть множество всех, определенных для данного автомата, символов (комбинаций входных сигналов). Преобразователь информации A реализует соответствие x→y, где y — выходной символ (комбинация выходных сигналов). Выходные символы, в свою очередь, составляют выходной алфавит Y преобразователя. Автоматный преобразователь характеризуется тем, что выход Y зависит не только от состояния входа X на данный момент, но и от истории состояний входа. При неопределенно долгом времени функционирования автомата размер его истории входа бесконечен. Для обеспечения реализуемости вводится ограничение, заключающееся в том, что автомат может находиться в одном из состояний конечного множества. Такой автомат называется конечным автоматным преобразователем информации или конечным автоматом [1, с. 87]. Большинство реальных алгоритмов управления содержит временные аспекты. В этом случае в каждом состоянии автомат может иметь временную задержку (таймаут). Если по истечении таймаута на автомат не поступило ни одного входного воздействия, то автомат переходит в другое (предписанное) состояние. Такие автоматы называются временными автоматами. Наиболее наглядным видом представления автоматной модели является диаграмма состояний (граф переходов), пример которой приведен на рисунке 1.

В зависимости от возможных входных воздействий (X) автомат может переходить из состояния в состояние с выдачей определенных выходных воздействий (Y) на исполнительное устройство. Для этого в автоматной модели есть переходы, помеченные вход-выходными действиями.

Пусть автомат и, соответственно, сам объект управления находятся в одном из следующих состояний: 1. Отключен. 2. Отключен по месту. 3. Включен по месту. 4. Команда на отключение. 5. Команда на отключение не прошла. 6. Сброс команды. 7. Внутренняя ошибка.

Каждое из входных и выходных воздействий в каждом из состояний представляет собой некоторую комбинацию сигналов. Входной и выходной алфавиты представляются подробно в отдельных таблицах.

Пусть состояния 1, 2, 3, 4, 6 имеют временную задержку. Для учета этого воздействия X0C и X15 включают в себя сигналы таймера (таймауты).

С использованием этих допущений при выполнении проектных работ логика управления различными устройствами автоматизированной системы может быть описана достаточно подробно. Кроме того, такая автоматная модель имеет ряд преимуществ по сравнению с другими языками алгоритмизации, например, схемами алгоритмов, функциональными схемами, циклограммами и др. Этими преимуществами являются: наглядность и компактность модели, отсутствие

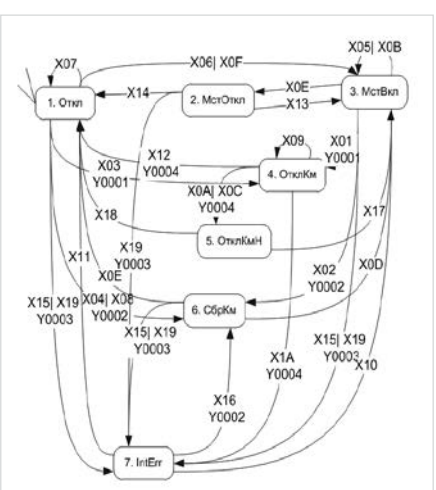

Рис. 1 — Диаграмма состояний работы насоса маслосборника

множества непонятных внутренних переменных — «флагов», а также возможность использования теории графов и дискретной математики для минимизации и формальной проверки на синтаксическую корректность программы, реализующей алгоритм управления.

Вот почему, специалистами ОАО «ТомскНИПИнефть» при проектировании алгоритмов управления широко применяются именно автоматные модели.

Во время проверки синтаксической корректности граф переходов проверяется на достижимость, полноту, непротиворечивость, реализуемость и отсутствие генерирующих контуров. Для этого выполняют минимизацию размерности модели. После минимизации количество узлов, в зависимости от степени различия поведения автомата в разных состояниях, уменьшается на порядок, что уменьшает количество операций сравнения в программе контроллера, где будет реализован алгоритм.

Для описания алгоритма в ОАО «ТомскНИПИнефть» разработана специальная программа FSM (рис. 2), которая реализована на основе теории конечных автоматов Мили. Для удобства работы в программе добавлены функции хранения и управления описательными данными разработанных алгоритмов (база данных MS Access) и вывод результата в виде описания алгоритма согласно нормативной документации [2, п. 7.1]. В соответствии с требованиями [2, п. 7.1] и с тем фактом, что наиболее используемым представлением алгоритмов на сегодняшний день остаются блок-схемы алгоритмов, в проектной документации приходится приводить также и модели в виде блок-схем, которые автоматически генерируются по соответствующему автоматному описанию. В результате работы с программой пользователь имеет связанный набор данных, который можно вывести либо в код программы контроллера, либо в описывающий разработанный алгоритм документ. В последнем случае программа, через OLE интерфейс, генерирует:

- описание используемой алгоритмом информации (MS Word);
- блок-схему функции переходов (MS Visio);
- таблицу переходов-выходов (MS Word);
- описание выходного алфавита (MS Word) [1, с. 89].

Следующей задачей создания МО проекта является разработка теста для проверки кода программы на соответствие алгоритму. Задача выявления ошибки в программной реализации алгоритма является очень ответственной и трудоемкой. Рассмотрим наглядный пример.

На рисунке 3 приведен фрагмент блок-схемы алгоритма, который определяет логику работы насоса во время аварии (S = 0x08). Фактически в данном состоянии в а горитме проверяется только одна входная переменная — «Break» (наличие аварии). Если приходит сигнал «Break», значение которого равно 1, то алгоритм остается в текущем состоянии. Если значение данного сигнала равно 0, то алгоритм переходит в состояние — отключен (S = 0x01).

На рисунке 4 приведен некорректный код программы, реализованной на языке Си. В коде присутствует лишняя строка, дополнительно проверяющая переменную «Command» (команда). Значение переменной «Command», равное 2, означает включить насос. В ответ на значение 2 переменной «Command» формируется соответствующий выходной сигнал — включить насос (On = 1).

Тривиальный вариант тестирования программы будет строиться на проверке логики на соответствие алгоритму (рис. 3), т.е. проверке только отклика программы на переменную «Break». При такой проверке ошибка с большой вероятностью не будет выявлена (в частности, в тех случаях, когда значение переменной «Command» не равно 2), и есть вероятность, что данная ошибка может и вовсе не выявиться в течение подготовки проекта к внедрению.

В результате в процессе эксплуатации при ложном срабатывании или ошибочном действии оператора насосный агрегат во время аварии может быть запущен в работу. Такая ситуация может привести к тяжелым последствиям. Задача тестирования еще более усложняется при проверке алгоритма с временными задержками. В этом случае при тестировании необходимо иметь дело уже с двумерными сигналами.

Данный пример показывает, что для тестирования программных реализаций на соответствие спецификации необходима полная

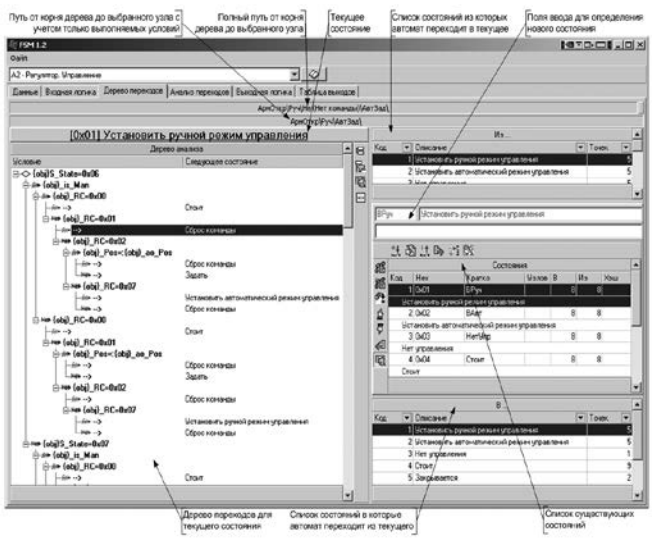

Рис. 2 — Интерфейс программного продукта FSM 1.2

Authors

Alexey G. Zebzeev (Tomsk, Russia)

academic degree is absent
workflow automation department engineer
of JSC «TomskNIPIneft»

Denis V. Zhuravlev

academic degree is absent
workflow automation department engineer
of JSC «TomskNIPIneft»

Evgeny I. Gromakov

PhD, National ResearchTomsk Polytechnic University, associate professor

Pushkarev A. Maxim

academic degree is absent
workflow automation department chief
of JSC «TomskNIPIneft»

Abstract

Approach to automated control system algorithms development and deriving conformance test suites for program implementations is proposed. Finite state machines using advantages as algorithm language are revealed. JSC "TomskNIPIneft" experience and developments of finite state machines using for automated control system of oil extraction and treatment project engineering are referred. Necessity of software testing with the guaranteed fault coverage is presented.

Materials and methods

Automata theory, discrete mathematics, graph theory.

Results

Software «FSM» for development of control algorithms based on automata theory is implemented. Software «FSM» using makes it possible to reduce time for algorithms development, test and optimization in the mathematical support engineering stage. Automata models using for development of program implementation conformance testing is introduced. This methodology makes it possible to test software with the guaranteed fault coverage.

Conclusions

This article is proposed the approach to timed finite state machines using for development of automated control system algorithms and deriving conformance test suites with the guaranteed fault coverage for program implementations. Necessity of software testing with the guaranteed fault coverage is cited as an example. Automata theory using makes it possible to reduce time for algorithms development, test and optimization in the mathematical support engineering stage. Science-based software testing methodology makes it possible to enhance quality and competitiveness of project documentation as well as makes it

possible to reduce time for software testing during starting-up and adjustment procedures.

Keywords
timed finite state machine, automated control systems, algorithm development, program implementation testing

References

1. *Marareskul S.I., Tutayev E.A. Finite automata theory and binary tree for method of formal designing and discrete control algorithms description // Oil industry. — 2008. — № 10. — P. 86-90.*
2. *Standardization management directive. RD- 50-34.698-90 "Automated systems. Records content requirement ". Group P87 — M.: 1992.*
3. *Zhigulin M.V., Dmitriev I.M., Yevtushenko N.V. Test synthesis with the guaranteed fault coverage for timed finite state machines // TPU news. — 2010. — T. 316. — №5. — P. 104-110.*
4. *Vasilevskiy M.P. About recognition of finite state machines faultiness // Кибернетика. — 1973. — № 4. — P. 93–108.*
5. *Hennie F.C. Fault detecting experiments for sequential circuits // Proc. of 5th Annual Symp. on Switching Theory and Logical Design. — USA, Nov. 1964. — P. 95–110.*

проверка всех переменных каждого из состояний алгоритма. Именно автоматный подход предлагает способы построения полного проверяющего теста, который на начальном этапе представляет собой поиск минимальной входной последовательности (различающей последовательности), формирующей все возможные входные сигналы в каждом из состояний алгоритма.

Для этого исходную автоматную модель предлагается представлять в виде полностью определенного автомата, т.е. такого автомата, для которого в каждом состоянии определен переход по любому из входных сигналов. Это достигается добавлением переходов с входными воздействиями, на которых поведение алгоритма не было описано. При этом автомат должен оставаться в том же состоянии (переход должен представлять собой петлю) и производить некоторый специальный выходной символ, например, Null.

На следующем шаге эта различающая последовательность подается на вход автоматной модели, после чего анализируется последовательность выходных реакций и дополнительно проверяется корректность работы самой автоматной модели. Затем та же входная последовательность подается на вход программной реализации.

Корректная реализация программы должна формировать выходную последовательность, совпадающую с выходом автоматной модели (задержки на каждом из переходов также должны совпадать).

В противном случае результатом теста будет выявление некорректной реализации и места возникновения ошибки.

Согласно классической теории автоматов, полный проверяющий тест можно построить только при условии, что множество рассматриваемых неисправностей ограничено. Тот факт, что проверяемая система должна выдавать только выходные последовательности, предписанные спецификацией (алгоритмом), формально описывается отношением f-эквивалентности между автоматами. В нашем случае система считается исправной, если программная реализация, описывающая ее поведение, является f-эквивалентной временному автомату-спецификации, и на каждую временную входную последовательность тестируемая система реагирует с быстротой, предписанной автоматом-спецификацией [3, с. 107]. Методики построения тестов предложены и достаточно подробно описаны в [3, с. 104–110; 4, с. 93–108; 5, с. 95–110], поэтому в настоящей статье мы не будем подробно останавливаться на этом. Поскольку в данных методиках проверяются все переходы в системе, то такие тесты позволяют выявлять все ошибки, не увеличивающие число состояний в системе.

Итоги

На основе теории автоматов реализована программа «FSM» для разработки алгоритмов управления. Использование программы уменьшает время на разработку, проверку и оптимизацию алгоритмов на стадии проектирования математического обеспечения. Модели автоматов также предлагается использовать для разработки тестов программных реализаций на соответствие алгоритмам. Данная методика позволяет гарантированно выявлять все ошибки в программном коде.

Выводы

В работе предложен подход к применению автоматных моделей для разработки алгоритмов работы автоматизированных систем управления и для полной проверки программных реализаций на соответствие разработанным алгоритмам. Необходимость такого тестирования проиллюстрирована на примере. Использование теории автоматов позволяет уменьшить время на разработку, проверку и оптимизацию алгоритмов на стадии проектирования математического обеспечения. Разработанная научно-обоснованная методика тестирования программного обеспечения позволяет повысить качество и конкурентоспособность проектной документации, а также позволяет сократить трудозатраты на проверку программного обеспечения автоматизированных систем управления технологическими процессами при пуско-наладочных работах.

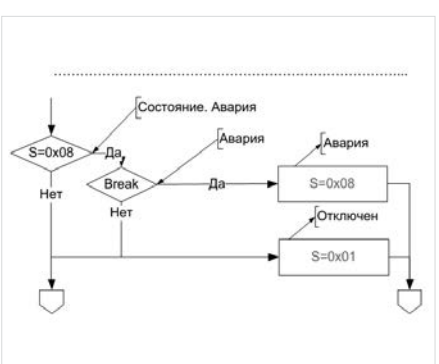

Рис. 3 — Алгоритма работы насоса

```
switch( S )
{.....
case 8:
  if ((Command==2))  {ON=1;  S=0x08}
 //Пришла команда «Включить»,
 //сработал выходной сигнал «Включить»
  else
   if ((Break))          {S=0x08}
  else
   if ((!Break))         {S=0x01}
break;
.....
```

Рис. 4 — Программный код работы насоса с ошибкой

Список использованной литературы

1. Марарескул С.И., Тютяев Е.А. Теория конечных автоматов и бинарные деревья в методе формального проектирования и описания алгоритма дискретного управления // Нефтяное хозяйство. - 2008. - № 10. - С. 86-90.
2. Руководящий документ по стандартизации. РД 50-34.698-90 «Автоматизированные системы. Требования к содержанию документов». Группа П87 – М.: 1992.
3. Жигулин М.В., Дмитриев И.М., Евтушенко Н.В. Синтез тестов с гарантированной полнотой для временных автоматов // Известия ТПУ. – 2010. – Т. 316. – №5. – С. 104-110.
4. Василевский М.П. О распознавании неисправности автоматов // Кибернетика. – 1973. – № 4. – С. 93–108.
5. Hennie F.C. Fault detecting experiments for sequential circuits //Proc. of 5th Annual Symp. on Switching Theory and Logical Design. – USA, Nov. 1964. – P. 95–110.