Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. Вып. 53. Автоматное программирование. 2008, с. 3–23.

УДК 004.4`2

ПАРАДИГМА АВТОМАТНОГО ПРОГРАММИРОВАНИЯ

А. А. Шалыто

В статье приводятся основные положения автоматного программирования и обосновываются преимущества его использования при разработке программного обеспечения. Описываются инструментальные средства для поддержки автоматного программирования и приводятся примеры успешного применения предлагаемого подхода на практике. В статье описывается процедурное программирование с явным выделением состояний и объектно-ориентированное программирование с явным выделением состояний. Автоматные программы могут быть эффективно верифицированы, а для их построения в ряде случаев могут быть применены генетические алгоритмы.

Введение

Большинство программистов-практиков считают, что в программировании нет особых проблем. «Отсутствие» проблем приводит к тому, что на практике при создании программного обеспечения (ПО) в большинстве случаев используются частные (*ad hoc* – экспромт или спонтанное решение) подходы, основанные на опыте программистов. Если трудности при создании программ и возникают, то их смиренно считают «неизбежным злом профессии». Тот факт, что при таком подходе достаточно много проектов заканчиваются неудачно, не изменяет точку зрения большинства.

Принципиально другое мнение у теоретиков программирования, которые еще в 1968 году «открыто признали кризис программного обеспечения» [1]. Однако в настоящее время это иногда оспаривается. Так, например, профессора Н. Вирт и Ю. Гутхнехт на пресс-конференции, посвященной избранию в 2005 г. создателя «Паскаля» Почетным доктором СПбГУ ИТМО (http://is.ifmo.ru/belletristic/_wirth_poch.pdf), утверждали, что они не видят проблем в программировании, оговорившись, правда, что сказанное не относится к программированию драйверов, которые обладают сложным поведением (отметим, что именно вопросам реализации такого поведения и посвящена настоящая статья).

Несмотря на наличие у некоторых влиятельных ученых таких взглядов, многие теоретики считают, что указанный кризис продолжается, и они стали искать выход из него в переходе от «искусства программирования» [2] к *программной инженерии (Software Engineering)* [3, 4], которой, в частности, активно занимается «наследник» Н. Вирта по кафедре в *ETH* (Цюрих) – Б. Мейер (http://is.ifmo.ru/belletristic/_meyer.pdf).

Несмотря на большое число работ по методологиям разработки ПО [5], проводимых, в том числе, и в настоящее время [6], считается [7], что указанный кризис не миновал.

Он во многом связан с тем, что специалисты по программной инженерии «варятся в собственном соку» и почти не используют подходы, разработанные в других инженерных областях.

Это привело к созданию сообщества исследователей и практиков, озабоченных будущим программной инженерии, которые особое внимание уделяют междисциплинарным исследованиям и подходам, в особенности, тем из них, которые созданы в «не–ИТ»-

дисциплинах задолго до появления компьютеров (*Interdisciplinary Software Engineering Network – ISEN*). При этом, например, по аналогии с архитектурой, родились паттерны проектирования.

В ходе указанных исследований сформировалось мнение [8], что при разработке ПО, видимо, может быть полезен *опыт создания систем автоматического управления* и, возможно, целесообразно сделать и один шаг назад, и обратиться к трудам основоположников кибернетики, таких как, например, Н. Винер, Д. фон Нейман, У. Эшби.

В подтверждение сказанному родился термин **программная кибернетика** *(Software Cybernetics)* [8], а первый международный семинар в этой области (*The First International Workshop on Software Cybernetics*), который после этого *стал ежегодным*, прошел в *2004* году.

Ниже излагаются основы **автоматного программирования**, разрабатываемого автором с *1991* года [9]. **Исследования в области автоматного программирования относятся как программной инженерии, так и к программной кибернетике**, и базируется на идеях теории автоматов и теории автоматического управления – двух из трех составляющих, на которых, по мнению Н. Винера, базируется кибернетика [10]. При этом особо подчеркнем, что под автоматным программированием автором понимается *не* программирование с применением автоматов, а соответствующая технология программирования, направленная на создание систем со сложным поведением [11]. В этом смысле уместно провести параллель между автоматами и автоматным программированием, с одной стороны, и *UML* (*Unified Modeling Language*) и *RUP* (*Rational Unified Process*) – с другой. Так, автоматы и *UML* – это нотации, в то время как автоматное программирование и *RUP* – это процессы, использующие указанные нотации.

В заключение раздела отметим, что излагаемый подход близок к подходу Д. Харела [12], о котором Ф. Брукс в работе [5], сказал, что «он может оказаться революционным».

1. Автоматное программирование как стиль программирования

Автоматное программирование является одним из стилей программирования [13]. В работе [13] также отмечено, что этот термин был предложен в работе [11].

Упрощенная трактовка автоматного программирования состоит в том, что это стиль программирования, при использовании которого поведение программ предлагается описывать автоматами, которые в дальнейшем преобразуются в код.

При этом отметим, что автоматы давно и успешно применяются в программировании, например, при построении компиляторов [14, 15]. Автоматы используются также и при решении многих других задач, например, при реализации протоколов.

В работе [16] подход с использованием автоматов для описания поведения программ был определен как стиль программирования, названный «программирование от состояний». В этой работе отмечалось, что «программирование от состояний, пожалуй, самый старый стиль программирования, так как на него наталкивает само устройство существующих вычислительных машин, которые представляют собой гигантские конечные автоматы».

На выделение указанного подхода в качестве самостоятельного стиля программирования, на авторов работы [16] повлияла работа [11], на которую они ссылаются, как на современную методику программирования от состояний. В работе [11] было предложено применять автоматы в программировании не от случая к случаю, как одну из моделей дискретной математики, а как универсальный подход, который целесообразно использовать практически всегда, когда программа должна обладать достаточно сложным поведе-

нием, и, в особенности, реактивным [17] – реагировать по-разному на события в зависимости от состояний, в которых находится программа.

При этом отметим, что несмотря на работы Д. Харела (лауреата *ACM Software Systems Award 2007*), даже реактивные системы проектируются автоматно далеко не всегда. Об этом, в частности, свидетельствует появление только в 2007 году работ ведущего специалиста *IBM* Э. Принга [18, 19] о реализации медленно всплывающих подсказок с применением автоматов. Однако, даже в этом случае нельзя говорить о применении технологии автоматного программирования, так как переход от модели к программе выполнялся эвристически, что привело к их расхождению.

В заключение раздела отметим, что «программирование с автоматами» нельзя рассматривать как парадигму программирования, так как при этом остается не ясным как с использованием автоматов проектировать и реализовывать программы в целом.

2. Автоматное программирование как парадигма программирования

Очень многие системы, которые для внешнего наблюдателя ведут себя достаточно осмысленно, являются автоматизированными объектами управления.

Автоматизированный объект управления *представляет собой совокупность системы управления (СУ) и объекта управления (ОУ), охваченных обратными связями* (рис. 1).

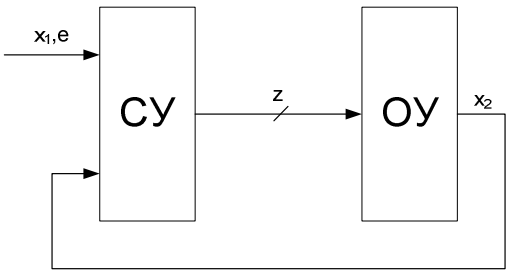

Рис. 1. Автоматизированный объект управления

Задача построения автоматизированных объектов управления рассматривается в любом курсе теории автоматического управления применительно к объектам различных типов.

Удивительно, что это почти никак не коснулось практики программирования, несмотря на то, что в теории алгоритмов в качестве одной из основных моделей используется машина Тьюринга (рис. 2).

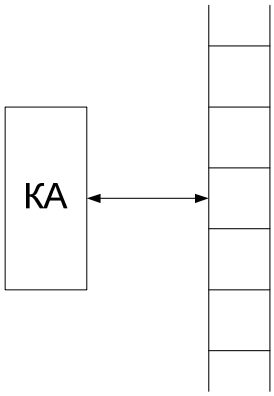

Рис. 2. Машина Тьюринга

Машина Тьюринга, по сути, является автоматизированным объектом управления [20], в котором система управления – конечный автомат, а объект управления – лента (ее ячейки памяти) (рис. 3).

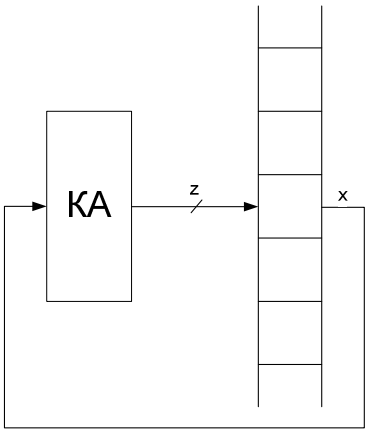

Рис. 3. Машина Тьюринга как автоматизированный объект управления

Сложность программирования на машине Тьюринга (например, умножение двух на три требует 66 команд [21]) определяется тем, что ней используются очень простые объекты управления (ячейки памяти), которые могут выполнять только простейшие действия (операции) по сдвигу головки и записи и стиранию отдельных символов. В этой ситуации вычисления, в некотором смысле, приходится выполнять конечному автомату, который для этой цели не приспособлен, так как его предназначение – управление. Другая особенность машины Тьюринга, которая может резко усложнять программы – это использование только одного автомата, которого достаточно для проведения теоретических исследований, но бывает недостаточно при практическом применении.

Переход от тьюрингова программирования к практическому (автоматному) программированию [20] осуществляется за счет усложнения объектов управления, которые могут выполнять сколь угодно сложные действия (операции), и применения в качестве системы управления системы взаимодействующих автоматов. Из изложенного следует, что универсальность предлагаемого подхода определяется тем, что он основан на расширении машины Тьюринга, которая позволяет реализовать произвольные алгоритмы.

При этом теоретические положения о том, что автоматы позволяют распознавать регулярные языки [22], а магазинные автоматы – языки с контекстно-свободными грамматиками, отходят на второй план, так как **в рассматриваемом подходе используются не**

автоматы, а автоматизированные объекты управления, в которых объекты управления и их сложность не фиксированы, как и число автоматов.

Учитывая изложенное, в работе [23] была сформулирована основная особенность автоматного программирования – *программы предлагается создавать так же, как производится автоматизация технологических (и не только) процессов*.

При этом на основе анализа предметной области выделяются источники входных воздействий и автоматизированные объекты управления, каждый из которых содержит систему управления (систему взаимодействующих конечных автоматов) и объекты управления. Эти объекты реализуют выходные воздействия и формируют значения еще одной разновидности входных воздействий, которые по обратным связям передаются системе управления.

Все перечисленные составляющие каждого автоматизированного объекта управления отображаются на *схеме связей*, которая может совмещаться со *схемой взаимодействия* автоматов. На схеме связей для каждого входного и выходного воздействия указывается **полное название** и краткое символьное обозначение, которое в дальнейшем и используется в качестве пометки в графах переходов автоматов и идентификатора соответствующей переменной в программе. Использование кратких символьных обозначений позволяет даже весьма сложные алгоритмы отражать так компактно, что часто граф переходов удается размещать на одном экране монитора, позволяя человеку «охватить» весь граф одним взглядом, что резко упрощает понимание таких графов.

Использование символьных, а не смысловых идентификаторов, являющихся обычно сокращенными английскими словами, смысл которых обычно забывается через некоторое время, не ухудшает понятности автоматных программ, так как в рамках рассматриваемого подхода их построение и внесение изменений в них должно производиться формально (вручную или автоматически) и только по графам переходов. При этом смысл переменных можно понять по схеме связей.

Объекты управления могут быть реальными или виртуальными – реализованными программно. В первом случае их логика изменена быть не может, а во втором – она (при необходимости) практически вся может быть вынесена в автоматы.

Парадигма автоматного программирования состоит в представлении и реализации программ как систем автоматизированных объектов управления [23].

3. Основные положения автоматного программирования

Основным понятием в автоматном программировании является *состояние* [11].

При написании автоматных программ предлагается (в отличие от работ [24, 25]) разделять состояния на два класса: *управляющие* и *вычислительные*. При этом с помощью небольшого числа управляющих состояний, как и в машине Тьюринга, можно управлять сколь угодно большим числом вычислительных состояний [26]. Во введенной классификации управляющие состояния могут быть названы *качественными*, а вычислительные – *количественными*. В рамках автоматного программирования основное внимание уделяется управляющим состояниям, которые, если это не оговаривается особо, и рассматриваются в дальнейшем.

При этом справедливо соотношение: «Состояния + входные воздействия = конечный автомат без выхода». Справедливо также: «Автомат без выхода + выходные воздействия = автомат».

Автоматы могут быть абстрактными (входные и выходные воздействия формируются последовательно) и структурными (входные и выходные воздействия формируются

«параллельно») [27]. В автоматном программировании, в отличие, например, от программирования компиляторов, обычно применяются структурные автоматы [28].

Время в автоматах в явном виде не используется. При необходимости применения элементов задержки, они рассматриваются как объекты управления. При этом задержки запускаются и сбрасываются из автоматов, а информация об истечении времени поступает в них в виде входных воздействий.

Автоматы могут задаваться в различном виде, однако при их проектировании и использовании человеком, они должны обладать *когнитивными свойствами* [29], что достигается при задании поведения автоматов в виде графов переходов (диаграмм состояний). В случае, если автоматы генерируются автоматически [30], то они могут задаваться иначе, например, в табличной форме. При этом отметим, что даже при сравнительно небольшом числе состояний и переходов, такое задание затрудняет понимание работы автоматов человеком, так как отражение переходов в них ненаглядно.

Понятность графов переходов достигается во многом за счет того, что *состояния декомпозируют множество всех входных воздействий* соответствующего автомата на группы, каждая из которых определяет переходы из рассматриваемого состояния.

4. Достоинства автоматного программирования

В рамках автоматного программирования предполагается, что собственно написание (генерация) программы начинается только после ее проектирования. При этом *в инженерной практике (в отличие от традиционного программирования) проект или его этап обязательно завершается выпуском проектной документации.* Поэтому при автоматном программировании, основной областью использования которого являются встроенные системы (чисто инженерная область), должна выпускаться **проектная** документация [31], а не только документация пользователя, как это обычно принято в программных проектах. При этом для автоматных программ необходимой компонентой, входящей в состав проектной документации, должны быть графы переходов.

Проектирование автоматов, описывающих логику программ, которая при традиционном программировании неупорядочена и поэтому сложна, а также *формальный и изоморфный* переход от автоматов к реализующим их программам, приводит к тому, что программы *либо сразу работают, либо требуют минимальной отладки*. Это также связано с тем, что реализация функций входных и выходных воздействий при излагаемом подходе, как отмечалось выше, почти не содержит логики.

При необходимости проведения отладки для автоматных программ могут генерироваться отладочные протоколы, которые отражают поведение программ в терминах автоматов (состояний, переходов, значений входных и выходных воздействий), так как в автоматном программировании автоматы являются не картинками [32], а частью программ, представленных в нетрадиционной для программистов визуальной, а не текстовой форме.

При этом отметим, что увеличение времени создания программ при использовании автоматного подхода компенсируется сокращением времени их отладки. Это приводит к тому, что для программ средней сложности трудоемкости разработки на основе автоматного и традиционного подходов практически совпадают. Однако в первом случае остаются диаграммы, понятные человеку, по которым программа была построена формально, а во втором – только программа, понимание логики которой для представителей заказчика или даже для ее автора через некоторое время часто представляет большую проблему.

Создание программ со сложным поведением без использования диаграмм приводит к трудностям на всех этапах жизненного цикла. Особенно сложно в этом случае реализо-

вывать программы, так как все особенности их поведения приходится держать в голове в течение всего времени их написания, вместо того чтобы отобразить их на диаграмме и на время забыть. Еще одним преимуществом автоматных программ является простота внесения изменений в них специалистами в предметной области, не являющимися профессиональными программистами.

Следующее преимущество автоматного подхода – эффективность верификации автоматных программ на основе метода *Model Checking* (верификация на модели) [33]. Это объясняется тем, что модель для верификации программ этого класса может строиться автоматически по графу переходов и иметь относительно небольшой размер, так как в графах переходов используются только управляющие состояния.

Для автоматных программ *отсутствует семантический разрыв* между требованиями к программе и к модели, так как он устраняется в ходе разработки графов переходов на этапе проектирования. Это позволяет считать автоматные программы приспособленными к верификации. Таким образом, при верификации программ имеет место ситуация, аналогичная с контролем схем со сложной логикой – эти схемы не удается проверить, если они не спроектированы специальным образом для обеспечения контролепригодности.

В заключение раздела отметим, что для автоматных программ естественен параллелизм, что особенно важно при применении многоядерных процессоров [34].

5. Разновидности автоматного программирования

Автоматное программирование развивается в трех основных направлениях: логическое управление, программирование с явным выделением состояний и объектно-ориентированное программирование с явным выделением состояний.

5.1. Логическое управление

Наиболее просто и естественно применять автоматы в системах логического управления, в которых входные и выходные переменные двоичны. Естественность применения автоматов при программировании этого класса систем объясняется тем, что программы в них заменяют логические схемы, проектирование которых с использованием автоматов широко распространено и развивается давно, начиная с релейно-контактных схем [28].

Однако простота и естественность применения автоматов даже для этого класса систем, видимо, далеко не очевидна, так как даже при программировании логических контроллеров [35], практически никто не предлагал сначала проектировать автоматы, а затем их реализовать на выбранном языке программирования.

В работе [36] было показано, что плохого в неавтоматном программировании контроллеров, а работах [37–40] описано применение технологии автоматного программирования для систем логического управления. Эта технология была использована при создании ряда систем управления, в том числе судовыми техническими средствами [11].

В работах [41–43] в качестве примеров показано, как применять предложенную технологию для языка функциональных блоков, широко используемого в программируемых логических контроллерах, а также в инструментальном средстве *LabVIEW*.

5.2. Программирование с явным выделением состояний

Значительно более широким классом по сравнению с системами логического управления являются реактивные системы. Для описания поведения таких систем, к которым относится, большинство встроенных систем, применение автоматов также естественно [17], как и для систем логического управления. Однако далеко не все программисты и для таких систем используют автоматы, что приводит к множеству проблем, о которых говорилось выше.

Реактивные системы являются более сложными по сравнению с системами логического управления. В них:

– в качестве входных воздействий, наряду с входными переменными, используются события;

– запуск программ осуществляется по событиям, а не циклически;

– в качестве выходных воздействий могут использоваться не только двоичные, но и другие функции, что позволяет называть автоматы, применяемые при этом, *гибридными* [44];

– автоматы могут содержать не только вложенные состояния [17], но и вложенные автоматы;

– автоматы могут взаимодействовать не только за счет проверки номеров состояний, как было предложено для систем логического управления [11], но также и за счет вложенности, вызываемости и обмена событиями (сообщениями).

Рассматриваемый класс систем обычно реализуется на процедурных языках программирования. Поэтому традиционно используемый процесс написания программ в этом случае может быть назван *процедурным программированием* или просто *программированием*. В таких программах состояния существуют, но они обычно явно не выделяются. Это отличает их от автоматных программ, создание которых в этом случае может быть названо *программированием с явным выделением состояний*.

Технология программирования с явным выделением состояний создавалось в ходе выполнения работ по разработке системы управления судовыми дизель-генераторами [45]. Эта технология подробно описана в работах [46–51].

5.3. Объектно-ориентированное программирование с явным выделением состояний

Уже два десятилетия объектно-ориентированное программирование (ООП) является наиболее широко используемым стилем программирования в мире. При применении объектной парадигмы программы строят из объектов, взаимодействующих за счет обмена сообщениями.

С развитием методов проектирования таких программ [52], в вопрос описания и реализации поведения объектов ясность не была внесена, а применять автоматы предлагалось лишь от случая к случаю, а не «как руководство» к действию, пригодное для использования во многих системах при описании сложного поведения. Даже в тех редких случаях, когда автоматы применялись для описания поведения программ (например, при проектировании программного обеспечения метеостанции в работе [24]), это делалось неубедительно и не способствовало их широкому применению в ООП.

Появление унифицированного языка моделирования *UML*, и даже языка *UML 2.0* [54], эту проблему не решило. Во-первых, в этом языке, кроме диаграмм состояний для описания поведения предлагается использовать и другие типы диаграмм и не говорится когда и какие диаграммы следует применять. Во-вторых, в рамках унифицированного процесса разработки программ [55], как впрочем, и при использовании других методологий их создания (например, описанной в работе [56]), не было предложено подходов для

совместного использования диаграмм, описывающих статические и динамические свойства программ. В-третьих, диаграммы для описания поведения в основном использовались как язык общения между участниками разработки и для документирования программ, в то время как для кодогенерации использовались только диаграммы классов. Лишь в последние годы в рамках концепции *исполняемого UML* [57] вопрос о кодогенерации был поставлен и для остальных диаграмм.

Для решения указанной проблемы под руководством автора были выполнены исследования по совместному использовании объектной и автоматной парадигм программирования. При этом такой стиль программирования был назван *объектно-ориентированное программирование с явным выделением состояний* [58].

Возможны различные подходы к решению этой проблемы. Автоматы можно использовать, например, как методы классов [59] или как классы [60].

Более «глубокое проникновение» автоматов в объектно-ориентированное программирование происходит при применении паттернов проектирования. При этом отметим, что применение паттерна *State*, предназначенного для реализации объектов, поведение которых зависит от состояния, как ни странно, обычно вызывает трудности [61]. Решению этого вопроса посвящена работа [62].

Для устранения недостатков, присущих указанному паттерну, в работах [63, 64] был предложен паттерн *State Machine,* на базе которого создан язык с тем же названием [65], являющийся расширением языка *Java* и позволяющий достаточно эффективно реализовывать автоматы.

В работе [66] был предложен еще один подход к реализации объектно-ориентированных программ с явным выделением состояний, который позволяет обеспечить повторное использование программных компонентов, параллельные вычисления и автоматическое протоколирование работы системы. В качестве основы для разработки «автоматной» части программ в этой работе была предложена библиотека, реализованная на языке *C++*. При использовании этой библиотеки остальная часть программ (контекст) разрабатывается традиционным образом.

Особенности проектирования и свойства автоматных программ позволяют отнести автоматное программирование к одной из разновидностей синхронного программирования [67, 68], которое активно развивается в Западной Европе для создания программного обеспечения ответственных систем.

Высокое качество автоматных программ может быть достигнуто не только за счет автоматного расширения универсальных языков программирования, но в тех случаях, когда для написания автоматных программ разрабатываются языки, учитывающие их специфику [69, 70].

Существуют и другие подходы к совместному использованию объектной и автоматной парадигм программирования. Классификация таких подходов приведена в работах [71, 72].

Работы в указанном направлении продолжаются. При этом, например, в работе [73] предложена удобная графическая нотация для отображения наследования в автоматных программах, а в работе [74] на основе использования понятия автоматизированный объект управления предложен новый подход к проектированию сложных объектно-ориентированных программ с явным выделением состояний.

6. Верификация автоматных программ

Выше отмечалось, что автоматные программы, в отличие от программ, написанных традиционно, сравнительно легко верифицируются на основе метода *Model Checking*, за разработку которого Э. Кларку, Э. Эмерсону и Д. Сифакису была присуждена *A.M. Turing Award 2007*. Упрощение формальной верификации для рассматриваемого класса программ по сравнению с построенными традиционным образом очень важно для построения ответственных систем (например, самолетов, вертолетов, ядерных реакторов и т. д.). Поэтому автор надеется, что со временем в технических заданиях на разработку программного обеспечения таких систем будет записываться требование использовать автоматное программирование.

В настоящее время активно ведутся работы в указанном направлении [33, 75–78]. В частности, проводятся исследования по государственному контракту «Разработка технологии верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода», выполняемому в рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007–2012 годы».

7. Автоматизация построения автоматных программ

Из изложенного следует, что основная трудоемкость построения автоматных программ связана с проектированием автоматов. Однако существуют задачи, про которые известно, что они могут быть решены с использованием автоматов, но эвристически построить автоматы в этих задачах крайне трудно или даже невозможно.

В этих случаях можно пытаться применять формализованные методы. Например, в работе [79] предложено использовать *динамическое программирование для построения автоматов*. Однако такой подход имеет ограниченную область применения.

Значительно более универсально применение генетического программирования для генерации автоматов. В настоящее время активно ведутся работы по этой тематике [80-87], в частности, проводятся исследования по государственному контракту «Технология генетического программирования для генерации автоматов управления системами со сложным поведением», выполняемой в рамках указанной выше Федеральной целевой программы.

Генерация автоматов позволяет до 80% кода автоматных программ строить почти автоматически, так как в программах этого класса объем кода, порождаемого автоматами, может достигать указанной величины.

В указанных работах на основе генетического программирования выращивались автоматы, а в работе [88] одновременно использовались различные модели и методы искусственного интеллекта. Во-первых, проектировалась мультиагентная система, во-вторых, система управления каждым объектом состояла из нейронной сети и автомата, в-третьих, для того, чтобы не потерять информацию о взаимодействии объектов в качестве особи выбиралось две указанные системы управления, и, наконец, в-четвертых, к такой сложной особи применялось генетическое программирование. В результате была построена система управления, в которой автомат имеет шесть состояний и 48 переходов. Эксперименты показали, что автоматически построенная система управления обеспечивает более эффективную работу мультиагентной системы, по сравнению со случаем, когда в автоматы системы управления (их было семь) строились эвристически [89]. При этом отметим, что поведение системы, построенной человеком, однако значительно более понятно, и в нее существенно проще вносить изменения.

8. Технология автоматного программирования

На основании указанных выше работ была разработана технология автоматного программирования, которая охватывает все этапы жизненного цикла программ рассматриваемого класса. Эта технология описана в работах [90–92], а в работе [93] она изложена для массового читателя.

9. Инструментальные средства для поддержки автоматного программирования

Рассмотрим средства для поддержки автоматного программирования.

Для *процедурных автоматных программ* в работе [94] было разработано инструментальное средство, которое по графам переходов, представленным в нотации, предложенной в работе [48], и изображенным с помощью пакета *Visio*, генерирует код, изоморфный реализуемым графам переходов, который основан на конструкциях `switch` языка *C*.

Это средство применяется в настоящее время при создании программного обеспечения одного класса ответственных систем реального времени. При этом в качестве программ используются реализованные вручную на языке *C* функции входных и выходных воздействий, которые практически не содержат логики, а также графы переходов, по которым исходный код генерируется автоматически. По отзывам разработчиков этих систем, их уже долгое время не покидает удивление оттого, что в каждой новой системе программы, спроектированные указанным образом, работают практически без отладки, а расширение функциональности обычно обеспечивается «малой кровью», и все это достигается при использовании процедурного, а не объектного программирования.

Это направление исследований получило развитие в работе [95], в которой показано, что аналогичный подход может быть использован для реализации автоматов на любом наперед заданном языке программирования. Для поддержки такого подхода было создано инструментальное средство *MetoAuto*.

Переходя к инструментальному средству для поддержки построения *объектно-ориентированных автоматных программ*, отметим, что если для генерации программ по автоматам, кроме средств, рассмотренных выше, известны также и многие другие [96], то решение задачи об автоматизации построения объектно-ориентированных программ в целом в открытых источниках не излагалось.

Решение этой задачи было предложено в ходе выполнения работ по теме «Автоматное программирование: применение и инструментальные средства», которая выполнялась в рамках Федеральной целевой научно-технической программы «Исследования и разработки по приоритетным направлениям науки и техники» на 2002–2006 годы [97].

В результате было создано инструментальное средство *UniMod* [98–102], которое автоматизирует процесс построения объектно-ориентированных автоматных программ. При его использовании структура программ задается диаграммами классов, которые изображаются не традиционным путем, а в форме схемы связей автоматов с поставщиками событий и объектами управления. Динамика программ описывается с помощью диаграмм состояний в нотации языка *UML*, в которых могут использоваться не только вложенные состояния, но и вложенные автоматы. При этом имеется возможность проверить ряд свойств этих диаграмм, например, полноту и непротиворечивость. Функции входных и выходных воздействий, которые практически не содержат логики, пишутся на языке *Java* вручную. После этого автоматически может быть скомпилирован код программы в целом или программа может выполняться в режиме интерпретации. Описанное инструментальное средство находится в свободном доступе, и является единственным открытым и бесплатным средством http://unimod.sourceforge.net/intro.html, поддерживающим концепцию *исполняемого UML* [103].

10. Апробация технологии автоматного программирования

Эта технология применялась не только при разработке программного обеспечения систем управления, но и для ряда других задач, например, головоломок [104], игр [105, 106], информационных систем [107] и учебных программ [108].

Автоматы применялись также и в ходе выполнения исследований в области искусственного интеллекта. Так, например, в работах [109, 110] рассматривалось применение автоматов при построении мультиагентных систем, а в работе [111] – вопрос о совместном использовании автоматов и нейронных сетей.

Кроме этих работ, на сайте [112], посвященном автоматному программированию, постоянно публикуются проекты, для каждого из которых созданы программное обеспечение на основе автоматного подхода и проектная документация (раздел «Проекты»). На сайте также имеется раздел, посвященный *UniMod*-проектам.

11. Разработка визуализаторов алгоритмов дискретной математики

Технология автоматного программирования продемонстрировала свою эффективность при решении различных задач, но при реализации алгоритмов дискретной математики автоматы используются крайне редко.

Известны лишь несколько задач этого класса, в которых применять автоматы целесообразно. Это, например, поиск подстрок [113], подсчет длины слов в строке [114] и обход деревьев [115].

Однако оказалось, что если реализовывать алгоритмы дискретной математики на автоматах часто нецелесообразно, то строить их визуализаторы с применением автоматов крайне полезно всегда, так как при этом визуализацию можно проводить в состояниях. При решении задачи построения таких визуализаторов был выполнен ряд работ как теоретического [116–120], так и прикладного характера [121–124]. В результате было разработано инструментальное средство *VIZI* для автоматизированного построения визуализаторов рассматриваемого класса. Визуализаторы, построенные на основе автоматного подхода, и проектная документация к ним опубликованы на сайте [112] в разделе «Визуализаторы».

Несмотря на то, что лекции по курсу «Алгоритмы дискретной математики» читаются практически в каждом университете мира, где обучают информационным технологиям, а в некоторых из них в учебном процессе используются визуализаторы, эффективную методику их построения без применения автоматного подхода создать не удавалось.

12. Отличие предлагаемого подхода от известных

Автоматы все чаще применяются в программировании. Если раньше они обычно использовались при построении компиляторов и протоколов, то теперь их от случая к случаю применяют и в других областях [18, 19] При этом был разработан ряд инструментальных средств, которые поддерживают программирование с автоматами [96]. Одним из наиболее распространенных инструментальных средств в этой области является *Stateflow* (http://www.mathworks.com/products/stateflow/) – расширение пакета *MatLab*. Это средство позволяет строить автоматы, моделировать работу их и выполнять кодогенерацию на языке *СИ*. В 2007 году корпорация *Microsoft* разработала программный продукт *Windows Workflow Foundation* (http://itc.ua/node/23217), в котором конечные автоматы (State Machines) в форме диаграмм состояний используются в качестве языка программирования.

Отличие предлагаемого подхода от известных состоит в том, что *автором предлагается применять автоматы в программировании не от случая к случаю, а* ***везде и всегда, где требуется обеспечить сложное поведение*** (сложным считается поведение, при котором выбор выходного воздействия зависит не только от входного воздействия, но и от предыстории). Это позволяет уменьшить число «прозрений», аналогичных с произошедшим в 2007 году, когда ведущий специалист корпорации *IBM* (!) вдруг обнаружил, что автоматы целесообразно использовать и при разработке виджетов [18, 19]. Автоматный подход поддержан парадигмой, технологией и инструментальными средствами. Его применение практически не зависит от используемых программных [70] и аппаратных средств [42, 43], в то время как в известных работах либо решается та или иная задача с применением автоматов, либо описывается конкретное инструментальное средство для поддержки программирования с автоматами.

При этом отметим, что если при использовании традиционного подхода для реализации последовательностного поведения программы обычно применяется множество двоичных переменных, называемых флагами, то при автоматном программировании в процесс реализации вводится этап, который известен из теории автоматов и называется «кодирование состояний». Термин «кодирование» при традиционном подходе заменял термин «программирование», так как при его использовании состояния явно обычно не выделялись. Явное выделение состояний позволяет *кодировать (различать) состояния автомата с любым числом состояний с помощью одной многозначной переменной, что позволило ввести в программирование понятие «наблюдаемость»* [11]**,** которое широко применяется в теории автоматического управления. Это дает возможность наблюдать за поведением каждого автомата в пространстве его состояний по изменению значения только **одной переменной.**

При этом если для автоматных программ ввести ограничение, состоящее в том, что в каждом программном цикле или при каждом запуске программы в каждом автомате выполняется не более одного перехода, то все используемые значения переменной, кодирующей состояния автомата, становятся доступными для других автоматов системы и могут использоваться во взаимных блокировках, не требуя введения для этой цели дополнительных переменных.

Проектирование автоматов и многозначное кодирование их состояний позволяют *формально и изоморфно* реализовывать автоматы с помощью конструкции `switch` языка *C* или ее аналогов в других языках программирования. При этом в теле этих конструкций нецелесообразно реализовывать функции входных и выходных переменных, а достаточно указывать только сами эти переменные, которые осуществляют вызов соответствующих функций, которые практически не содержат логики и реализуются отдельно. Таким образом осуществляется отделение логики поведения (условий, определяющих необходимость выполнения тех или иных действий) от описания его семантики (смысла каждого из действий), что резко упрощает понимание программ.

Изложенный подход позволяет практически исключить отладку построенных таким образом программ. Использование предлагаемого подхода не всегда обеспечивает уменьшение времени создания программы по сравнению с традиционным подходом. Однако программы, построенные с использованием предлагаемого подхода, обладают многими достоинствами, которые были описаны выше.

Есть основание предполагать, что для ответственных объектов, автоматизация которых требуется верификации программ, применение автоматного программирования, как отмечалось выше, **может стать неизбежным**. Исследования в этом направлении активно проводятся [125, 126].

Заключение

В ходе исследований по автоматному программированию необходимо было решить задачи в области проведения научных исследований, разработки технологии для его поддержки и образования. Все эти задачи были решены в результате педагогического эксперимента, описанного в работе [127].

Со многими работами, указанными в списке литературы, можно ознакомиться на сайте http://is.ifmo.ru/. Там же приведены и некоторые результаты внедрения автоматного программирования в практику проектирования различных систем.

Идеи автоматного программирования, изложенные в статье, могут в том или ином виде использоваться не только для текстовых и визуальных языков программирования, как описано выше, но и для программируемых логических контроллеров [128, 129], а также различных средств автоматизации [43] и имитационного моделирования [44, 130, 131].

Автор предполагает, что области применения автоматного программирования будут еще расширяться, так как «более 99% всех микропроцессоров, проданных в 1998 году, использовались во встраиваемых системах, а в 2000 году число микроконтроллеров в высококачественном автомобиле достигало 60» [132].

Рассматриваемая парадигма важна и для образования. В частности, на ее основе можно проводить *первоначальное обучение проектированию программ*, не только в университетах [133], но и в средних школах [134]. При этом отметим, что, поскольку концепции автоматного программирования существенно отличаются от традиционных, начинать обучение программированию в этом стиле следует как можно раньше [135].

Автор выражает надежду, что технология использования автоматов при разработке программных систем со сложным поведением будет развиваться: появятся новые модели, нотации, инструментальные средства. Например, при участии авторов ведутся работы по созданию текстовых языков автоматного программирования [136–138], декларативных методов описания автоматов на императивных языках программирования [139], методов динамической верификации автоматных программ [140], инструментальных средств на основе концепции предметно-ориентированных языков программирования [141]. Также проводятся работы по применению автоматов при создании программного обеспечения для мобильных роботов [142] и автоматизации документооборота [143].

Автор надеется, что парадигма автоматного программирования, изложенная в этой статье, станет «каркасом» для дальнейших исследований в области использования автоматов в программировании.

В заключение работы отметим, что создание автоматных программ предполагает их проектирование, названное во введении к книге [11] «автоматным проектированием программ».

Так как при проектировании этого класса программ основное внимание уделяется управлению, то можно говорить о *парадигме управления*, названной автором в работе [11] «автоматное управление». Эта парадигма, как отмечалось выше, неоднократно апробировалась на практике, в том числе и при проектировании программного обеспечения сложных систем [45, 128, 129].

Отметим также, что использование автоматов при проектировании систем управления [144] до последнего времени в основном рассматривалось в рамках применения гибридных автоматов [145]. Дополнительный интерес к использованию автоматов в управлении возник у специалистов после пленарного доклада Р. Брокетта на конгрессе

IFAC [146], в котором обсуждались вопросы упрощения проектирования сложных систем управления.

Автор признателен **Дмитрию Александровичу Поспелову**, в ходе беседы с которым в 1998 году на конференции по мультиагентным системам, проходившей в поселке Ольгино под Санкт-Петербургом, родился термин (см. введение к работе [11]), который использован в названии настоящей статьи. На английский язык этот термин переводится как «Automata-Based Programming». Он был предложен в работе [147], а парадигма автоматного программирования – в работе [148].

Литература

1. Дейкстра Э. Смиренный программист // Лекции лауреатов премии Тьюринга за первые двадцать лет. 1966–1985. М.: Мир, 1993.
2. Кнут Д. Искусство программирования. Том. 1. Основные алгоритмы. М.: Вильямс, 2000.
3. Software Engineering. Germany: NATO Science Committee, 1968. http://www.europrog.ru/book/nato1968e.pdf
4. Software Engineering Techniques. Italy: NATO Science Committee, 1969. http://www.europrog.ru/book/nato1969e.pdf
5. Брукс Ф. Мифический человеко-месяц или как создаются программные системы. СПб.: Символ, 2000.
6. Software Engineering 2006. ETH Zurich. Chair of Software Engineering.
7. Черняк Л. Адаптируемость и адаптивность // Открытые системы. 2004. № 9.
8. Cai Kai-Yuan, Chen T. Y., Tse T. H. Towards Research on Software Cybernetics / Proceedings of 7th IEEE International on High-assurance Systems Engineering (HASE 2002). Los Alamitos. IEEE Computer Society Press, 2002.
9. Шалыто А. А. Программная реализация управляющих автоматов // Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып. 13.
10. Яковлев В. Б. Автоматика, кибернетика, информатика, синергетика / Труды конференции «Пятьдесят лет развития кибернетики». СПбГТУ, 1999.
11. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.
12. Harel D. Biting the Silver Bullet: Toward a Brighter Future for System Development // Computer. 1992. № 1.
13. Непейвода Н. Н. Стили и методы программирования. М.: Интернет-университет информационных технологий, 2005.
14. Ахо А., Сети Р., Ульман Д. Компиляторы. Принципы, технологии, инструменты. М.: Вильямс, 2001.
15. Хопкрофт Д., Мотвани Р., Ульман Д. Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.
16. Непейвода Н. Н., Скопин И. Н. Основания программирования. М.-Ижевск: Институт компьютерных исследований, 2003.
17. Harel D. et al. Statemate: A Working Environment for the Development of Complex Reactive Systems // IEEE Trans.Software Eng. 1990. № 4.
18. Принг Э. Конечные автоматы в JavaScript, Часть 1: Разработаем виджет. http://www.ibm.com/developerworks/ru/library/wa-finitemach1/index.html
19. Принг Э. Конечные автоматы в JavaScript, Часть 2: Реализация виджета. http://www.ibm.com/developerworks/ru/library/wa-finitemach2/wa-finitemac_ru.html

20. Шалыто А. А., Туккель Н. И. От тьюрингова программирования к автоматному // Мир ПК. 2002. № 2.
21. Хопкрофт Д. Машины Тьюринга // В мире науки. 1984. № 7.
22. Карпов Ю. Г. Теория автоматов. СПб.: Питер, 2002.
23. Шалыто А. А. Автоматное программирование / Тезисы докладов Международной научной конференции памяти профессора А.М. Богомолова «Компьютерные науки и информационные технологии». Саратов: СГУ, 2007.
24. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++. М.: Бином, СПб.: Невский диалект, 1998.
25. Лавров С. Н. Программирование. Математические основы, средства, теория. СПб.: БХВ-Петербург, 2001.
26. Туккель Н. И., Шамгунов Н. Н., Шалыто А. А. Ханойские башни и автоматы // Программист. 2002. № 8.
27. Глушков В. М. Синтез цифровых автоматов. М.: Физматгиз, 1962.
28. Гаврилов М. А., Девятков В. В., Пупырев Е. И. Логическое проектирование дискретных автоматов. М.: Наука, 1977.
29. Shalyto A. A. Cognitive Properties of Hierarchical Representations of Complex Logical Structures / Proceeding of the 1995 International Symposium on Intelligent Control (ISIC). Workshop. 1995. Monterey. California.
30. Фогель Л., Оуэнс А., Уолш М. Искусственный интеллект и эволюционное моделирование. М.: Мир, 1969.
31. Шалыто А. А. Новая инициатива в программировании. Движение за открытую проектную документацию // Информационно-управляющие системы. 2003. № 4.
32. Зюбин В.Е. Программирование информационно-управляющих систем на основе конечных автоматов. Новосибирск: НГУ, 2006.
33. Кузьмин Е. В., Соколов В. А. Моделирование, спецификация и верификация «автоматных» программ // Программирование. 2008. № 1, с. 38–60.
34. Любченко В., Тяжлов Ю. Осторожно: многоядерный процессор // Открытые системы. 2007. № 6.
35. International Standard IEC 1131-3. Programmable controllers. Part 3. Programming languages. International Electrotechnical Commission. 1993.
36. Вавилов В. К., Шалыто А. А. Что плохого в неавтоматном подходе к программированию контроллеров? // Промышленные АСУ и контроллеры. 2007. № 1.
37. Шалыто А. А. Технология программной реализации алгоритмов логического управления как средство повышения живучести // Тезисы докладов научно-технической конференции «Проблемы обеспечения живучести кораблей и судов». СПб.: Судостроение, 1992.
38. Антипов В. В., Шалыто А. А. Алгоритмизация и программирование задач логического управления техническими средствами. СПб.: Моринтех, 1996.
39. Шалыто А. А. SWITCH-технология. Алгоритмизация и программирование задач логического управления // Промышленные АСУ и контроллеры. 1999. № 9.
40. Шалыто А. А. Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления // Известия РАН. Теория и системы управления. 2000. № 6.
41. Шалыто А. А. Реализация алгоритмов логического управления программами на языке функциональных блоков // Промышленные АСУ и контроллеры. 2000. № 4.
42. Альтерман И. З., Шалыто А. А. Формальные методы программирования логических контроллеров // Промышленные АСУ и контроллеры. 2005. № 10.

43. Вавилов В. К., Шалыто А. А. LabVIEW и SWITCH-технология // Промышленные АСУ и контроллеры. 2006. № 6.
44. Колесов Ю. Б., Сениченков Ю. Б. Моделирование систем. Динамические и гибридные системы. СПб.: БХВ–Петербург, 2006.
45. Туккель Н. И., Шалыто А. А. Проектирование программного обеспечения системы управления дизель-генераторами на основе автоматного подхода // Системы управления и обработки информации. 2003. Вып. 5.
46. Туккель Н. И., Шалыто А. А. Применение SWITCH-технологии для программирования в событийных системах / Труды международной научно-технической конференции «Пятьдесят лет развития кибернетики», СПб.: СПбГТУ, 1999.
47. Туккель Н. И., Шалыто А. А. SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Промышленные АСУ и контроллеры. 2000. № 10.
48. Туккель Н. И., Шалыто А. А. SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5.
49. Туккель Н. И., Шалыто А. А. Программирование с явным выделением состояний // Мир ПК. 2001. № 8, № 9.
50. Туккель Н. И., Шалыто А. А. Реализация автоматов при программировании событийных систем // Программист. 2002. № 4.
51. Шалыто А. А. Алгоритмизация и программирование для систем логического управления и «реактивных» систем (обзор) // Автоматика и телемеханика. 2001. № 1.
52. Грэхем И. Объектно-ориентированные методы. Принципы и практика. М.: Вильямс, 2004.
53. Рамбо Дж., Якобсон А., Буч Г. UML. Специальный справочник. СПб.: Питер, 2001.
54. Рамбо Дж., Блаха М. UML 2.0. Объектно-ориентированное моделирование и разработка. СПб.: Питер, 2007.
55. Рамбо Дж., Якобсон А., Буч Г. Унифицированные процесс разработки программного обеспечения. СПб.: Питер, 2002.
56. Ауер К., Миллер Р. Экстремальное программирование: постановка процесса. СПб.: Питер, 2003.
57. Mellor S. et al. Executabl UML: A Foundation for Model Driven Architecture. MA: Addison-Wesley, 2002.
58. Туккель Н. И., Шалыто А. А. Объектно-ориентированное программирование с явным выделением состояний / Материалы Международной научно-технической конференции «Искусственный интеллект». Т.1. Таганрог. ТГРУ; Донецк. Донецкий гос. ин-т искусств. интеллекта. 2002.
59. Туккель Н. И., Шалыто А. А. Танки и автоматы // BYTE/Россия. 2003. № 2.
60. Наумов Л. А., Шалыто А. А. Искусство программирования лифта. Объектно-ориентированное программирование с явным выделением состояний // Информационно-управляющие системы. 2003. № 6.
61. Гранд М. Шаблоны проектирования в Java. М.: Новое знание. 2004.
62. Шопырин Д. Г., Шалыто А. А. Применение класса STATE в объектно-ориентированном программировании с явным выделением состояний // Труды X Всероссийской научно-методической конференции «Телематика-2003». СПб.: СПбГИТМО (ТУ). 2003. Т.1.

63. Корнеев Г. А., Шамгунов Н. Н., Шалыто А. А. Паттерн State Machine для объектно-ориентированного проектирования автоматов // Информационно-управляющие системы. 2004. № 5.
64. Shamgunov N., Korneev G., Shalyto A. State Machine Design Pattern / .NET Technologies 2006 – Shot communication papers conference proceedings. 4-th International Conference in Central Europe on .Net Technologies. University of West Bohemia. 2006.
65. Корнеев Г. А., Шамгунов Н. Н., Шалыто А. А. Язык State Machine – расширение языка Java для эффективной реализации автоматов // Информационно-управляющие системы. 2005. № 1.
66. Шопырин Д. Г., Шалыто А. А. Объектно-ориентированный подход к автоматному программированию // Информационно-управляющие системы. 2003. № 5.
67. Туккель Н. И., Шалыто А. А. Автоматное и синхронное программирование // Искусственный интеллект. 2003. № 4.
68. Шопырин Д. Г., Шалыто А. А. Синхронное программирование // Информационно-управляющие системы. 2004. № 3.
69. Гуров В. С., Мазин М. А., Шалыто А. А. Текстовый язык для автоматного программирования // XIV Всероссийская научно-методическая конференция «Телематика-2007». СПб.: СПбГУ ИТМО. 2007. Т.2.
70. Степанов О. Г., Шопырин Д. Г., Шалыто А. А. Предметно-ориентированный язык автоматного программирования на базе динамического языку RUBY // Информационно-управляющие системы. 2007. № 4.
71. Наумов Л. А., Шалыто А. А. Методы объектно-ориентированной реализации реактивных агентов на основе конечных автоматов // Искусственный интеллект. 2004. № 4.
72. Naumov L., Korneev G., Shalyto A. Methods of Object-Oriented Reactive Agents Implementation on the Basis of Finite Automata / 2005 International Conference on «Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering» (KIMAS-05). Boston: IEEE. 2005.
73. Шопырин Д. Г., Шалыто А. А. Графическая нотация наследования автоматных классов // Программирование. 2007. № 4.
74. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. СПбГУ ИТМО, 2007. http://is.ifmo.ru/books/_umk.pdf
75. Вельдер С. Э., Шалыто А. А. О верификации простых автоматных программ на основе метода Model Checking // Информационно-управляющие системы. 2006. № 3.
76. Корнеев Г. А., Парфенов В. Г., Шалыто А. А. Верификация автоматных программ / Тезисы докладов Международной научной конференции, посвященной памяти профессора А.М. Богомолова «Компьютерные науки и технологии». Саратов: СГУ. 2007.
77. Корнеев Г.А., Шалыто А.А. Верификация управляющих программ со сложным поведением, построенных на основе автоматного подхода / Материалы международной научно-технической мультиконференции «Проблемы информационно-компьютерных технологий и мехатроники» (ИКТМ–2007). Таганрог: НИИ МВС ЮФУ. 2007.
78. Гуров В. С., Шалыто А. А., Яминов Б. Р. Технология верификации автоматных моделей программ без их трансляции во входной язык верификатора / Материалы международной научно-технической мультиконференции «Проблемы информационно-компьютерных технологий и мехатроники» (ИКТМ–2007). Таганрог: НИИ МВС ЮФУ. 2007.

79. Оршанский С. А., Шалыто А. А. Применение динамического программирования при решении задач на конечных автоматах // Компьютерные инструменты в образовании. 2007. № 4.
80. Лобанов П. Г., Шалыто А. А. Использование генетических алгоритмов для автоматического построения конечного автомата в задаче о флибах / 1-я Российская мультиконференция по проблемам управления. Сборник докладов четвертой научной конференции «Управление и информационные технологии». СПбГУ ЭТУ «ЛЭТИ». 2006.
81. Лобанов П. Г., Шалыто А. А. Использование генетических алгоритмов для автоматического построения конечных автоматов в задаче о флибах // Известия РАН. Теория и системы управления. 2007. № 5.
82. Мандриков Е. А., Кулев В. А., Шалыто А. А. Построения автоматов с помощью генетических алгоритмов для решения задачи о флибах / Сборник X международной конференции по мягким вычислениям и измерениям. СПбГУ ЭТУ «ЛЭТИ». 2007. Т.1.
83. Мандриков Е. А., Кулев В. А., Шалыто А. А. Применение генетического программирования при решении задачи о флибах // Информационные технологии. 2007. № 12.
84. Царев Ф. Н., Шалыто А. А. Применение генетического программирования для генерации автоматов в задаче об «умном муравье» / Сборник научных трудов. IV-я Международная научно-практическая конференция «Интегрированные модели и мягкие вычисления в искусственном интеллекте. Коломна: 2007.
85. Царев Ф. Н., Шалыто А. А. О построении автоматов с минимальным числом состояний для задачи об «умном муравье» / Сборник докладов X международной конференции по мягким вычислениям и измерениям. СПбГУ ЭТУ «ЛЭТИ». 2007. Т.2.
86. Поликарпова Н. И., Точилин В. Н., Шалыто А. А. Применение генетического программирования для реализации систем со сложным поведением / Сборник научных трудов. IV-я Международная научно-практическая конференция «Интегрированные модели и мягкие вычисления в искусственном интеллекте. Коломна. 2007.
87. Поликарпова Н. И., Точилин В. Н., Шалыто А. А. Разработка библиотеки для генерации автоматов методом генетического программирования / Сборник докладов X международной конференции по мягким вычислениям и измерениям. СПбГУ ЭТУ «ЛЭТИ». 2007. Т.2.
88. Царев Ф. Н., Шалыто А. А. Применение генетического программирования для построения мультиагентной системы одного класса / Материалы международной научно-технической мультиконференции «Проблемы информационно-компьютерных технологий и мехатроники» (ИКТМ–2007). Таганрог: НИИ МВС ЮФУ. 2007.
89. Paraschenko D., Tsarev F., Shalyto A. Modeling Technology for One Class of Multi-Agent Systems with Automata Based Programming / Proceedings of 2006 IEEE International Conference on Computational Intelligence for Measurement Systems and Application (IEEE CIMSA- 2006). Spain, 2006.
90. Шалыто А. А. Технология автоматного программирования / Труды первой Всероссийской научной конференции «Методы и средства обработки информации». М.: МГУ. 2003.

91. Korneev G. A., Shalyto A. A. State-Driven Programming / Материалы Евразийского научного симпозиума. Корея. Сеул. Политехнический университет. 2007. www.eurasia.re.kr
92. Шалыто А. А. Автоматное программирование / Тезисы докладов Международной научной конференции, посвященной памяти профессора А.М. Богомолова «Компьютерные науки и технологии». Саратов: СГУ. 2007.
93. Шалыто А. А. Технология автоматного программирования // Мир ПК. 2003. № 10.
94. Головешин А. Использование конвертора Visio2Switch. http://is.ifmo.ru
95. Канжелев С. Ю., Шалыто А. А. Автоматическая генерация автоматного кода // Информационно-управляющие системы. 2006. № 6.
96. List of UML tools. http://en.wikipedia.org/wiki/List_of_UML_tools
97. О проекте «Технология автоматного программирования: применение и инструментальные средства» // Информационные технологии. 2006. № 2.
98. Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А. А. Разработка UML. SWITCH-технология. Eclipse // Информационно-управляющие системы. 2004. № 6.
99. Gurov V. S., Mazin M. A., №arvsky A. S, Shalyto A. A. UniMod: Method and Development of Reactive Object-Oriented Programs with Explicit States Emphasis / Proceedings 2005 of St. Petersburg IEEE Chapters. International Conference «110 Anniversary of Radio Invention SPb ETU «LETI». 2005.
100. Гуров В. С., Мазин М. А., Шалыто А. А. Ядро автоматного программирования // Свидетельство об официальной регистрации программы для ЭВМ. № 2006 613249 от 14.09.2006.
101. Гуров В. С., Мазин М. А., Шалыто А. А. Встраиваемый модуль автоматного программирования для среды разработки // Свидетельство об официальной регистрации программы для ЭВМ. № 2006 613817 от 07.11.2006.
102. Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А.А. Инструментальное средство для поддержки автоматного программирования // Программирование. 2007. № 6.
103. Гуров В. С., Нарвский А. С., Шалыто А. А. Исполняемый UML из России // PC Week/RE. 2005. № 26.
104. Мазин М.А., Шалыто А.А. Преступники и автоматы // Мир ПК. 2004. № 9.
105. Беляев А. В., Суясов Д. И., Шалыто А. А. Компьютерная игра «Космонавт». Проектирование и реализация // Компьютерные инструменты в образовании. 2004. № 4.
106. Корнеев Г. А., Петрошенко П. А., Шалыто А. А. Реализация игры «Морской бой» на основе автоматного подхода // Компьютерные инструменты в образовании. 2005. № 6.
107. Гуров В. С., Нарвский А. С., Шалыто А. А. ICQ и автоматы // Технология «Клиент-Сервер». 2004. № 3.
108. Мазин М. А., Парфенов В. Г., Шалыто А. А. Анимация. FLASH-технология. Автоматы // Компьютерные инструменты в образовании. 2003. № 4.
109. Naumov L., Shalyto A. Automata Theory for Multi-Agent Systems Implementation / International Conference on «Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering». (KIMAS-03). Boston: IEEE. 2003.
110. Yartsev B., Korneev G., Kotov V., Shalyto A. Automata-Based Programming of the Reactive Multi-Agent Control Systems / 2005 International Conference on «Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering» (KIMAS-05). Boston: IEEE. 2005.
111. Кретинин А. В., Солдатов Д. В., Шостак А. В., Шалыто А. А. Ракеты. Автоматы. Нейронные сети // Нейрокомпьютеры: разработка и применение. 2005. № 5.

112. Сайт по автоматному программированию. http://is.ifmo.ru
113. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. М.: МЦНМО, 1999.
114. Лобанов П. Г., Шалыто А. А. Подсчет длины слов в строке // Мир ПК. 2005. № 7.
115. Корнеев Г. А., Шамгунов Н. Н., Шалыто А. А. Обход деревьев на основе автоматного подхода // Компьютерные инструменты в образовании. 2004. № 3.
116. Туккель Н. И., Шалыто А. А. Реализация вычислительных алгоритмов на основе автоматного подхода // Телекоммуникации и информатизация образования. 2001. № 6.
117. Туккель Н. И., Шалыто А. А. Преобразование итеративных алгоритмов в автоматные // Программирование. 2002. № 5.
118. Туккель Н. И., Шамгунов Н. Н., Шалыто А. А. Реализация рекурсивных алгоритмов на основе автоматного подхода // Телекоммуникации и информатизация образования. 2002. № 5.
119. Казаков М. А., Корнеев Г. А., Шалыто А. А. Метод построения логики работы визуализаторов алгоритмов на основе конечных автоматов // Телекоммуникации и информатизация образования. 2003. № 6.
120. Корнеев Г. А., Шалыто А. А. Преобразование программ в систему взаимодействующих конечных автоматов / Труды Второй Всероссийской научной конференции «Методы и средства обработки информации». М.: МГУ. 2005.
121. Казаков М. А., Шалыто А. А. Использование автоматного программирования для реализации визуализаторов // Компьютерные инструменты в образовании. 2004. № 2.
122. Казаков М. А., Шалыто А. А. Реализация анимации при построении визуализаторов алгоритмов на основе автоматного подхода // Информационно-управляющие системы. 2005. № 4.
123. Корнеев Г. А., Шалыто А. А. VIZI – язык описания визуализаторов алгоритмов // Научно-технический вестник СПбГУ ИТМО. Высокие технологии в оптических и информационных системах. 2005. Вып. 23.
124. Корнеев Г. А., Шалыто А. А. Построение визуализаторов алгоритмов дискретной математики // Научно-технический вестник СПбГУ ИТМО. Высокие технологии в оптических и информационных системах. 2005. Вып. 23.
125. Риган П., Хемилтон С. NASA: миссия надежна // Открытые системы. 2004. № 3.
126. Разработка технологии верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода. СПбГУ ИТМО. 2007. 2008. http://is.ifmo.ru/verification/_2007_01_report-verification.pdf
127. Шалыто А. А. Триединая задача одного педагогического эксперимента в области IT-образования // Инженерное образование. 2007. № 4.
128. Вавилов К. В. Программируемые логические контроллеры SIMATIC S7-200 (SIEMENS). Методика алгоритмизации и программирования задач логического управления. СПб.: 2005. http://is.ifmo.ru/progeny/_metod065.pdf
129. Вавилов К. В. Контроллеры SIMATIC S7-300 (SIEMENS). Организация взаимодействия локальных систем управления на основе автоматного подхода и функционального разделения автоматов управления. СПб.: 2005. http://is.ifmo.ru/progeny/_s7300.pdf
130. Карпов Ю. Г. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. БХВ-Петербург, 2006.
131. Дьяконов В. П. Simulink 5/6/7. Самоучитель. М.: ДМК Пресс, 2008.

132. Ослэндер Д., Риджли Д., Ринггенберг Д. Управляющие программы для механических систем. Объектно-ориентированное проектирование систем реального времени. М.: БИНОМ. Лаборатория знаний. 2004.
133. Васильев В. Н., Казаков М. А., Корнеев Г. А., Парфенов В. Г., Шалыто А. А. Применение проектного подхода на основе автоматного программирования при подготовке разработчиков программного обеспечения / Труды первого Санкт-Петербургского конгресса «Профессиональное образование, наука, инновации в XXI веке». СПбГУ ИТМО. 2007, с. 98–100. http://is.ifmo.ru/download/2008-02-25_comp_proekt.pdf
134. Красильников Н. Н., Парфенов В. Г., Царев Ф. Н., Шалыто А. А. Виртуальная лаборатория для первоначального обучения проектированию программ // Компьютерные инструменты в образовании. 2007. № 5, с. 62–67.
135. Кузнецов Б. П. Психология автоматного программирования // BYTE/Россия. 2000. № 11, с. 28–32. http://www.softcraft.ru/design/ap/ap01.shtml
136. Гуров В. С., Мазин М. А., Шалыто А. А. Текстовый язык автоматного программирования / Тезисы докладов Международной научной конференции, посвященной памяти профессора А. М. Богомолова «Компьютерные науки и технологии». Саратов: СГУ. 2007, с. 66–69.
137. Степанов О. Г., Шалыто А. А., Шопырин Д. Г. Предметно-ориентированный язык автоматного программирования на базе динамического языка Ruby // Информационно-управляющие системы. 2007. № 4, с. 22–27.
138. Лагунов И. А. Разработка текстового языка автоматного программирования и его реализация для инструментального средства UniMod на основе автоматного подхода. СПбГУ ИТМО, 2008. http://is.ifmo.ru/papers/fsml
139. Астафуров А. А., Шалыто А. А. Декларативный подход к вложению и наследованию автоматных классов при использовании императивных языков программирования / Материалы конференции «Software Engineering Confernece (Russia) – SEC(R) 2007». М.: TEKAMA. 2007, с. 230–238.
140. Stepanov O., Shalyto A. A Method for Automatic Runtime Verification of Automata-Based Programs / Proceeding of the Second Spring Young Researchers' Colloquium on Software Engineering (SYRCoSE'2008). SPbSU. 2008. Vol.2, pp.19–23.
141. Решетников Е. О. Инструментальное средство для визуального проектирования автоматных программ на основе Microsoft Domain-Specific Language Tools. СПбГУ ИТМО. 2007. http://is.ifmo.ru/papers/reshetnikov_bachelor
142. Клебан В. О., Шалыто А. А. Использование автоматного программирования для построения многоуровневых систем управления мобильными роботами / Сборник тезисов 19 Всероссийской научно-технической конференции «Экстремальная робототехника». СПб: ЦНИИ РТК, 2008, с. 85–87.
143. Клебан В. О., Новиков Ф.А. Применение конечных автоматов в документообороте // Научно-технический вестник СПбГУ ИТМО. Выпуск 53. Автоматное программирование. 2008, с. 286–294.
144. Гудвин Г. К., Гребе С. Ф., Сальгадо М. Э. Проектирование систем управления. М.: Бином, 2004.
145. Alur R., Courcoubetis C., Henzinger A., Ho P. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems //Lecture notes in computer science. 1993. V.736, pp. 209–229.
146. Brokett R. Reduced Complexity control systems /Proceedings of the 17 th World Congress The International Federation of Automatic Control. Seoul. 2008.

147. Shalyto A. A. Technology of Automata-Based Programming. 2004. http://www.codeproject.com/KB/architecture/abp.aspx?print=true
148. Шалыто А. А. Парадигма автоматного программирования / Международная научно-техническая мультиконференция «Проблемы информационно-компьютерных технологий и мехатроники». Материалы международной научно-технической конференции «Многопроцессорные вычислительные и управляющие системы» (МВУС`2007). Таганрог: НИИМВС. 2007. Т.1, с.191–194.