АВТОМАТНОЕ ПРОГРАММИРОВАНИЕ

Шалыто А.А.
Санкт-Петербургский государственный университет информационных технологий механики и оптики, г. Санкт-Петербург
shalyto@mail.ifmo.ru

Ключевые слова: программная инженерия, конечные автоматы, автоматное программирование, верификация, генетическое программирование.

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 году 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 систем с таким поведением и посвящена настоящая статья)**.**

Несмотря на наличие у некоторых известных 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 программ.

В ходе указанных исследований сформировалось мнение [8], что при разработке ПО может быть полезен *опыт создания систем автоматического управления* и, возможно, целесообразно сделать 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, по мнению автора, относятся как к программной инженерии, так и к программной кибернетике, и основаны 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 конечных автоматов, а технология программирования, направленная на создание систем со сложным поведением, которое реализуется автоматами [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 этой работе также отмечено, что термин «автоматное программирование» 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 поведением, 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

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).

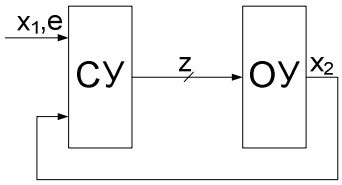

Рис. 1. Автоматизированный объект управления

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).

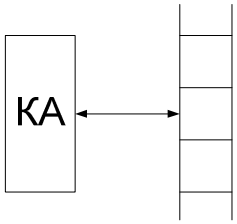

Рис. 2. Машина Тьюринга

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

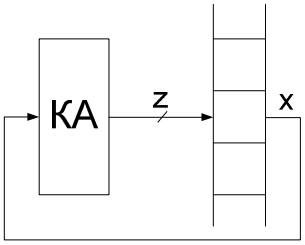

Рис. 3. Машина Тьюринга как автоматизированный объект управления

По мнению автора, 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 такие действия (операции), как запись и стирание 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 мало 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 для практического использования машины Тьюринга, которая и в исходном виде 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 разместить на экране монитора, позволяя человеку «охватить» весь граф одним взглядом, что резко упрощает его понимание.

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* 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 реализовывать 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 предлагаемого подхода – эффективность верификации автоматных программ на основе метода *Model Checking* (проверка на модели) [33]. Это объясняется тем, что модель для верификации программ этого класса может строиться по графам переходов автоматически и иметь относительно небольшой размер, так как в таких графах 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. 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 разработанными традиционным образом очень важно при построении ответственных систем (например, для 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 [35–38]. В частности, завершены 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».

6. 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 [39] предложено использовать *динамическое программирование для построения автоматов*. Однако такой подход имеет ограниченную область применения.

Значительно более универсально применение генетического программирования для генерации автоматов. По этой тематике в СПбГУ ИТМО активно ведутся работы. Так, в частности, выполнены исследования по государственному контракту «Технология генетического программирования для генерации автоматов управления системами со сложным поведением», которые проводились 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 работах [40–44] на основе генетического программирования выращивались автоматы, а в работе [45] 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, 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 [46].

При этом отметим, что поведение системы, построенной человеком, однако значительно более понятно, и в нее проще вносить изменения.

7. Технология автоматного программирования

На основании указанных выше работ была разработана технология автоматного программирования, которая охватывает все этапы жизненного цикла программ рассматриваемого класса, включая их верификацию. Эта технология описана в работе [47], а в работе [48] она изложена для массового читателя.

8. Инструментальные средства для поддержки автоматного программирования

Рассмотрим средства для поддержки автоматного программирования.

Для *процедурных автоматных программ* в работе [49] было разработано инструментальное средство, которое по графам переходов, представленным в нотации, предложенной в работе [50], и изображенным с помощью пакета *Visio*, генерирует код, изоморфный реализуемым графам переходов, который основан на конструкциях `switch` языка *C*.

Указанное средство применяется в настоящее время, например, при создании программного обеспечения одного класса ответственных систем реального времени [51]. 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 [52], 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 *MetaAuto*.

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 [53], 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 [54].

В результате было создано инструментальное средство *UniMod* [55, 56], 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* [57].

9. Апробация технологии автоматного программирования

Эта технология применялась не только при разработке программного обеспечения систем управления ответственными объектами, но и для ряда других задач, например, игр [58], информационных систем [59] и учебных программ [60].

Автоматы применялись также и в ходе выполнения исследований в области искусственного интеллекта. Так, например, в работах [61, 62] рассматривалось применение автоматов при построении мультиагентных систем, а в работе [63] – вопрос о совместном использовании автоматов и нейронных сетей.

Кроме этих работ, на сайте [64], 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.

10. 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. При этом был разработан ряд инструментальных средств, которые поддерживают программирование с автоматами [53]. 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).

Автоматный подход, описанный в настоящей работе, поддержан парадигмой, технологией и инструментальными средствами. Его применение практически не зависит от используемых программных [65] и аппаратных средств [66], в то время как в известных работах либо решается та или иная задача с применением автоматов, либо описывается язык или конкретное инструментальное средство для поддержки программирования с автоматами.

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 описаны выше.

Есть основание предполагать, что для ответственных объектов, автоматизация которых требуется верификации программ, применение автоматного программирования, как отмечалось выше, **может стать неизбежным**.

В заключение раздела отметим, что существенное влияние на создание автоматной парадигмы программирования оказали работы по программной реализации алгоритмов логического управления, которые проводились в лаборатории член-корреспондента АН СССР М.А. Гаврилова в Институте проблем управления [28, 67–71]. В первой главе работы [11] подробно изложены достоинства и недостатки этих и других подходов (сети Петри, язык «Графсет» и т.д.), и обоснована целесообразность создания технологии автоматного программирования, охватывающей все этапы жизненного цикла программ для систем со сложным поведением, которая может быть реализована на различных программных и аппаратных платформах.

Заключение

В ходе исследований по автоматному программированию были поставлены задачи в области проведения научных исследований, разработки технологии для его поддержки и образования, которые были успешно решены.

Со многими работами, указанными в списке литературы, можно ознакомиться на сайте http://is.ifmo.ru/. Там же приведены и 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, а также различных средств автоматизации и имитационного моделирования.

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» [72].

Изложенная выше парадигма программирования важна и для образования. В частности, на ее основе можно проводить *первоначальное обучение проектированию программ*, не только в университетах, но и в средних школах. При этом отметим, что, поскольку концепции автоматного программирования [73] существенно отличаются от традиционных, начинать обучение программированию в этом стиле следует как можно раньше.

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 автора ведутся работы по созданию текстовых языков автоматного программирования, декларативных методов описания автоматов на императивных языках программирования, методов динамической верификации автоматных программ, инструментальных средств на основе концепции предметно-ориентированных языков программирования. Также проводятся работы по применению автоматов при создании программного обеспечения для мобильных роботов и автоматизации документооборота [74].

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 2009*. 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 C++. 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. *Корнеев Г. А., Парфенов В. Г., Шалыто А. А.* Верификация автоматных программ /Тезисы докладов международной научной конференции «Компьютерные науки и технологии». Саратов: СГУ, 2007.
34. *Любченко В., Тяжлов Ю.* Осторожно: Многоядерный процессор //Открытые системы. 2007. № 6.
35. *Вельдер С. Э., Шалыто А. А.* О верификации простых автоматных программ на основе метода «Model Checking» //Информационно-управляющие системы. 2007. № 3.
36. *Кузьмин Е. В., Соколов В. А.* Моделирование, спецификация и верификация «автоматных» программ // Программирование. 2008. № 1.
37. *Егоров К. В.*, *Шалыто А. А.* Методика верификации автоматных программ //Информационно-управляющие системы //Информационно-управляющие системы. 2008. № 5.
38. *Вельдер С. Э., Шалыто А. А.* Верификация автоматных моделей методом редуцированного графа переходов // Научно-технический вестник СПбГУ ИТМО. 2009. № 6 (64).
39. *Оршанский С. А., Шалыто А. А.* Применение динамического программирования при решении задач на конечных автоматах // Компьютерные инструменты в образовании. 2007. № 4.
40. *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.

41. *Мандриков Е. А., Кулев В. А., Шалыто А. А.* Применение генетического программирования при решении задачи о флибах // Информационные технологии. 2007. № 12.
42. *Данилов В. Р., Шалыто А. А.* Метод генетического программирования для генерации автоматов, представленных деревьями решений /Материалы научно-технической конференции «Научное программное обеспечение в образовании и научных исследованиях». СПбГПУ, 2008.
43. *Davydov A., Sokolov D., Tsarev F.. Shalyto A.* Application of Genetic Programming for Generation of Controllers Represented by Automata /Preprints of 13th IFAC Symposium on Information Control Problems in Manufacturing. Moscow, 2009.
44. *Поликарпова Н. И., Точилин В. Н., Шалыто А. А.* Метод сокращенных таблиц для генерации автоматов с большим числом входных переменных на основе генетического программирования //Известия РАН. Теория и системы управления. 2010. № 2.
45. *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.
46. *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 (CIMSA-2006). Spain, 2006.
47. *Шалыто А. А.* Технология автоматного программирования / Труды первой Всероссийской научной конференции «Методы и средства обработки информации». М.: МГУ, 2003.
48. *Шалыто А. А.* Технология автоматного программирования // Мир ПК. 2003. № 10.
49. *Головешин А.* Использование конвертора *Visio2Switch*. http://is.ifmo.ru/progeny/visio2switch/
50. *Туккель Н. И., Шалыто А. А.* SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5.
51. *Волобуев В. Н., Калачинский А. В.* Опыт использования автоматного подхода при разработке программного обеспечения систем боевого управления //Системы управления и обработки информации. 2009. Вып. 18.
52. *Канжелев С. Ю., Шалыто А. А.* Автоматическая генерация автоматного кода // Информационно-управляющие системы. 2006. № 6.
53. *List of UML tools.* http://en.wikipedia.org/wiki/List_of_UML_tools
54. О проекте «Технология автоматного программирования: применение и инструментальные средства» // Информационные технологии. 2006. № 2.
55. *Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А. А.* Разработка UML. SWITCH-технология. Eclipse // Информационно-управляющие системы. 2004. № 6.
56. *Гуров В. С., Мазин М. А., Нарвский А. С., Шалыто А. А.* Инструментальное средство для поддержки автоматного программирования // Программирование. 2007. № 6.
57. *Гуров В. С., Нарвский А. С., Шалыто А. А.* Исполняемый UML из России //PC Week/RE. 2005. № 26.
58. *Беляев А. В., Суясов Д. И., Шалыто А. А.* Компьютерная игра «Космонавт». Проектирование и реализация // Компьютерные инструменты в образовании. 2004. № 4.
59. *Гуров В. С., Нарвский А. С., Шалыто А. А.* ICQ и автоматы // Технология «Клиент-Сервер». 2004. № 3.
60. *Мазин М. А., Парфенов В. Г., Шалыто А. А.* Анимация. FLASH-технология. Автоматы // Компьютерные инструменты в образовании. 2003. № 4.
61. *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.
62. *Yartsev B., Korneev G., Kotov V., Shalyto A.* Automata-Based Programming of the Reactive Multi-Agent Control Systems / International Conference on «Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering» (KIMAS-05). Boston: IEEE. 2005.
63. *Кретинин А. В., Солдатов Д. В., Шостак А. В., Шалыто А. А.* Ракеты. Автоматы. Нейронные сети // Нейрокомпьютеры: разработка и применение. 2005. № 5.
64. *Сайт по автоматному программированию*. http://is.ifmo.ru
65. *Степанов О. Г., Шопырин Д. Г., Шалыто А. А.* Предметно-ориентированный язык автоматного программирования на базе динамического языку RUBY // Информационно-управляющие системы. 2007. № 4.

66. *Альтерман И. З., Шалыто А. А.* Формальные методы программирования логических контроллеров // Промышленные АСУ и контроллеры. 2005. № 10.
67. *Кузнецов О. П., Макаревский А. Я., Марковский А. В., Шипилина Л. Б.* ЯРУС – язык описания работы сложных автоматов // Автоматика и телемеханика. 1972. № 6, 7.
68. *Амбарцумян А. А., Искра С. А., Кривандина Н. Ю. и др.* Проблемно-ориентированный язык описания поведения систем логического управления ФОРУМ-М /Проектирование устройств логического управления. М.: Наука, 1974.
69. *Девятков В. В., Чичковский А. Б.* Условие-82 – язык программно-логического управления //Автоматизация проектирования. Вып. 2. М.: Машиностроение, 1990.
70. *Иванов Н. Н.* О поведении взаимодействующих автоматов, моделирующих систему «объект управления – управляющее устройство» //Автоматика и телемеханика. 1979. № 7.
71. *Михайлов Г. И., Руднев В. В.* Простейшие системы взаимосвязанных графов и расширение практических возможностей конечных автоматов //Автоматика и телемеханика. 1979. № 10.
72. *Ослэндер Д.*, *Риджли Д., Рингенберг Д.* Управляющие программы для механических систем. Объектно-ориентированное проектирование систем реального времени. М.: Бином. Лаборатория знаний, 2004.
73. *Поликарпова Н. И., Шалыто А. А.* Автоматное программирование. СПб.: Питер, 2010.
74. *Научно-технический вестник СПбГУ ИТМО*. Вып. 53. Автоматное программирование. 2008. http://books.ifmo.ru/ntv/ntv/53/ntv_53.pdf