А.А. Шалыто

Автоматное программирование

Программная реализация управляющих автоматов

Существуют различные подходы к программной реализации управляющих автоматов. Однако технология программной реализации, обеспечивающая упрощение взаимодействия заказчика, разработчика и программиста, требует обобщения имеющихся результатов, чему и посвящена настоящая работа.

Предлагаемый подход обеспечивает в значительной мере формализацию построения и проверки правильности программ, их однотипность, наглядность, структурированность, наблюдаемость и управляемость.

Технология предполагает пять стадий разработки автоматных программ, каждая из которых состоит из ряда этапов: системное проектирование (1-7); проектирование формальной спецификации автомата или системы взаимосвязанных автоматов (8-13); логическое проектирование автоматов (14-20); программирование (21,22); проверка правильности программ (23-27), применимая для задач небольшой размерности. Перечислим этапы разработки.

I. Системное проектирование.

1. Описание объекта управления с выделением источников и приемников информации и указанием его свойств.
2. Описание «алгоритма» управления, заданного заказчиком.
3. Дополнение «алгоритма», учитывающее опыт разработки.
4. Выбор органов управления и средств представления информации.
5. Построение схемы связей «органы управления – управляющий автомат – органы управления – средства представления информации».
6. Представление в случае необходимости управляющего автомата в виде взаимосвязанной системы управляющих автоматов.
7. Декомпозиция каждого управляющего автомата на автомат и функциональные элементы задержки.

II. Проектирование формальной спецификации автомата или композиции автоматов.

8. Выбор языка спецификаций, позволяющего в доступной форме согласовать с заказчиком поведение автомата и легко перейти к программе. Выполнено сравнение функциональных схем, схем алгоритмов и графов переходов и обоснован выбор последних в качестве языка спецификации.
9. Построение исходного графа переходов для каждого автомата с учетом взаимодействия автоматов.
10. Обеспечение полноты и непротиворечивости графа переходов для каждого автомата.
11. Согласование системы взаимосвязанных автоматов (формальной спецификации) с Заказчиком.
12. Построение формальной спецификации для функциональных элементов задержки.
13. Построение формальной спецификации для модели объектов управления.

III. Логическое проектирование автоматов.
14. Выбор структурной модели каждого автомата (автомат без выходного преобразователя, автомат Мура и т. д.) и разновидности, зависящей от расположения технологических элементов задержки в модели.
15. Выбор вида кодирования состояний каждого автомата (принудительное, логарифмическое, многозначное и т. д.). Обоснован выбор многозначного кодирования.
16. Выбор алгоритмической модели программной реализации каждого автомата (система булевых формул, таблица переходов, схема алгоритма, конструкция *switch*). Обоснован выбор конструкция *switch*.
17. Построение алгоритмической модели каждого автомата.
18. Оптимизация булевых формул, помечающих дуги графов переходов (раздельная или совместная).
19. Выбор и построение алгоритмической модели функциональных элементов задержки.
20. Выбор и построение алгоритмической модели объектов управления.

IV. Программирование
21. Выбор языка программирования.
22. Программирование взаимосвязанной системы автоматов, функциональных элементов задержки и моделей объектов управления.

V. Проверка правильности программ, применимая для задач небольшой размерности.
23. Построение единого графа проверки.
24. Построение графа функционирования.
25. Корректировка спецификаций и моделей в случае отсутствия изоморфизма графов функционирования и проверки.
26. Программирование графа проверки.
27. Замена программы, реализующей композицию автоматов, программой, реализующей граф проверки, что обеспечивает использование полностью проверенной программы, что практически недостижимо при других подходах.

Разработанная (при участии автора) программная оболочка позволяет выбрать требуемое число входов и выходов, а также используемое число графов переходов. С помощью клавиатуры на экран компьютера могут быть введены значения входных переменных. При этом программой на экран выводятся вычисленные значения выходных переменных и значения всех функциональных элементов задержки.

Новизна предлагаемого подхода состоит в том, что на экран также выводятся значения переменных состояний каждого графа переходов, *что делает построенную программу наблюдаемой*.

Изоморфизм графа переходов и конструкции *switch* обеспечивает управляемость (изменяемость) программной реализации.

В качестве примера для сравнения различных спецификаций и моделей в рамках предлагаемой технологии разработано 40 программ на языке СИ, реализующих алгоритм управления двумя клапанами с памятью с помощью двух кнопок без памяти.

Изложенный подход может быть использован также и для программной реализации вычислительных алгоритмов при замене объектов управления на операционные блоки, осуществляющие собственно вычисления, **что позволяет разделить управляющую и операционную части программы**.

https://is.ifmo.ru/works/switch_prr/. Первая публикация по автоматному программированию. Журнал «Судостроительная промышленность. Серия «Автоматика и телемеханика». **1991**. Вып.13, с. 41-42.

Об автоматизации стиральных машин

Недавно, на компьютерной выставке я пытался объяснить генеральному директору одной фирмы, занимающейся промышленной автоматизацией, что при создании программного обеспечения для описания поведения систем управления целесообразно использовать автоматный подход (http://is.ifmo.ru), основанный на применении графов переходов.

Это предложение было отвергнуто, и мне было сказано, что «у нас в этом вопросе и так все хорошо». Да и как может быть иначе, «так как обычно возражения исходят от тех, кто не владеет приемами, против которых они возражают» (**Грис Д.** Наука программирования. М.: Мир, 1984).

Через некоторое время мой собеседник, видимо, из вежливости, со мной согласился, добавив, что, возможно, применение такого подхода целесообразно при создании сложных систем, которыми он, якобы, не занимается. При этом, видимо, системы промышленной автоматизации он считает простыми!

Для убедительности своего высказывания мой собеседник добавил: «Вы, ведь не будете использовать графы переходов при автоматизации стиральных машин, правда?».

Я с ним не согласился, так как нашему с Грисом :-) мнению, «никто не сможет научиться хорошо составлять большие программы, пока он не научится хорошо составлять малые», но разговор исчерпал себя.

Однако, через минуту я вспомнил одну историю, которую и рассказал моему собеседнику: «Однажды, так же, как и Вас, я пытался убедить выдающегося студента (призера двух студенческих командных чемпионатов мира по программированию *ACM – Association for Computing Machinery*) в целесообразности использования автоматного подхода при программировании. Выслушав меня, не отрывая глаз от компьютера, студент сказал, что, когда он будет делать компилятор, он использует автоматы, а год назад по курсу информационных систем при автоматизации стиральной машины он, естественно, применял автоматы, а где еще применять их, он не знает.

После этого в глазах генерального директора появилась тоска: я, видимо, добился своей цели, так как **Антуан де Сент-Экзюпери** говорил: «Если хочешь построить лодку, не зови людей, не бей в барабаны, а всели в них тоску по бескрайнему синему морю».

И еще несколько замечаний.

В разговоре с генеральным директором я сказал, что использование автоматов, позволяет так документировать проект, что и через десять лет понять его логику можно будет весьма просто. На что директор ответил, что его не волнует, что будет через десять лет, как, впрочем, и подавляющее большинство его заказчиков! Интересно, не правда, ли? Особенно учитывая тот факт, что моему собеседнику было не более сорока лет! На вопрос, что же его волнует, он ответил: «Деньги или их экономия, и сегодня». Говоря другими словами, «после нас хоть потоп». Тут, как говорится, не до науки.

Слава Богу, так думают не все в мире, и поэтому техническая революция продолжается: «Экономия – не выход из кризиса. Чтобы достичь успеха в долгосрочной перспективе, технологические компании должны продолжать инвестиции в исследования в независимости от экономической ситуации» (**Крейг Баррет**, генеральный директор корпорации *Intel*).

Я, конечно, понимаю, что мой собеседник далек от мысли достичь уровня руководителя указанной корпорации, так как он нормальный человек. В то время как, по мнению председателя совета директоров», **Энди Гроува**, «**выживают только параноики**», о чем он и написал одноименную книгу.

Возвращаясь к общению с упомянутым выше выдающимся студентом, отмечу, что, каждый раз, показывая ему очередной завершенный проект, выполненный на основе автоматного подхода в рамках «Инициативы за открытую проектную документацию» (http://is.ifmo.ru), я спрашиваю: «**Это**

компилятор или стиральная машина?», и очень часто это оказывается ни тем, ни другим, а чем-то третьим. Это бывает, например, **визуализаторами алгоритмов, для построения логики которых он предложил на основе автоматного подхода прекрасный метод!** В общем, как говорится в одной пословице, «когда живешь, тогда доживаешь»!

И последнее. На днях я сделал открытие :-): многолетние успехи команд Санкт-Петербургского государственного университета информационных технологий, механики и оптики (ИТМО) на командных чемпионатах мира по программированию связаны с наличием аббревиатуры ИТМО в слове «**алгорИТМОв**».

Эта мистика продолжается и по сей день. 26 ноября 2003 г. в Санкт-Петербурге студенты университета ИТМО **Павел Маврин**, **Дмитрий Павлов** и **Сергей Оршанский** (тренер – **Андрей Станкевич**) победили, решив 10 задач из 11, в полуфинальных соревнованиях командного студенческого чемпионата мира по программированию *ACM* 2003/2004 гг., выиграв одновременно командный чемпионат России, а 31 марта 2004 г. в Праге они стали чемпионами мира.

Кроме указанной выше причины, успех команды ИТМО, видимо :-), связан также и с тем, что в начале 2003 г. **П. Маврин** и **Д. Павлов** были награждены Премией Президента РФ за выдающиеся **способности**, проявленные в ходе международной олимпиады по информатике среди школьников в 2002 г., а А. Станкевич является двукратным призером чемпионатов *ACM* и совместно с выдающимся студентом, упомянутым выше, также занимался автоматизацией стиральной машины! Интересно, кто готовил этот Указ Президента, в котором вместо выдающихся результатов указаны такие же способности, которые являются достижением не награждаемого, а лучшем случае его родителей.

23.12.2003. *PC Week/RE*. 2003. № 48 (414), с. 52. https://www.itweek.ru/themes/detail.php?ID=66414, http://is.ifmo.ru/works/washing/

P.S. 1. После того, как я опубликовал эту статью, мне сообщили, что стиральные машины, как я и предполагал, необходимо автоматизировать весьма деликатно, так как в противном случае **может наступить стиральная трагедия**. (**Лем С.** Из воспоминаний Ийона Тихого: V (Стиральная трагедия). http://lib.ru/LEM/memoirs5.txt).

2. Сегодня пришел очередной студент и, не зная о моем интересе к «стиральным машинам», принес … проект автоматизации стиральной машины, который выполнен на автоматах. Так что история со стиральными машинами продолжается.

https://vk.com/@1077823-ob-avtomatizacii-stiralnyh-mashin

Еще один разговор об автоматизации «стиральных» машин

Вчера один второкурсник начал разговор со мной, сказав, что не согласен с моим предложением о »тотальном» применении автоматов в программировании, отметив, что существует незначительное число областей, где, по его мнению, их применять целесообразно. При этом он упомянул компиляторы, системы управления и визуализаторы алгоритмов.

Ему повезло, что в это время на его лице не было улыбки собственного превосходства, которую я видел пару раз ранее, при разговоре со мной. Может быть, в улыбке и не было никакого превосходства, но так мне, почему-то, казалось.

Нельзя сказать, что этот разговор меня сильно обрадовал, но и не сильно расстроил, так как я слышал подобные высказывания весьма часто и привык к ним. Однако, некоторую радость я испытал: по сравнению с ситуацией, описанной в статье «Об автоматизации стиральных машин», приведенной выше, я за пару лет кое-чего добился – в указанный список добавились визуализаторы алгоритмов.

Даже, если бы я ничего больше в жизни не сделал – инициатива создания методов и технологий построения логики визуализаторов с применением автоматов, по моему мнению, весьма значительное достижение, так как только у нас в университете при изучении программирования и дискретной математики сотни сильных студентов эвристически строили визуализаторы вместо того, чтобы, по крайней мере, один из них создал формальный метод построения программ этого класса.

Я сказал об этом студенту, добавив, что уже сегодня можно говорить и о других областях, где целесообразно применять автоматы, а самая неожиданная из них – построение скинов для видеоплееров (http://is.ifmo.ru/projects/crystal/).

После этого я сказал, что буду очень признателен, если он или кто-то из его приятелей через некоторое время добавит еще что-нибудь в указанный «короткий» список.

После этого я отметил, что может быть у автоматов и узкая сфера применения, но известно, что все микроконтроллеры и 98% микропроцессоров, выпускаемых в мире (вполне возможно, выполняющих всего лишь два процента от общего числа вычислений, проводимых на планете), используются в системах управления, в которых студент признал целесообразным применение автоматов, и остается только два процента микропроцессоров (вполне возможно, выполняющих 98% от общего числа вычислений), используемых для других задач, и даже, если автоматы для них применимы не столь эффективно, то я это переживу.

После этого я пожелал студенту придумать что-либо в жизни, хотя бы для столь же «узкой» области применения, как у меня, и попросил на прощание дать мне номер его мобильного телефона.

Свой номер, как это принято сегодня у молодежи, он наизусть не помнил, и после весьма долгого поиска в памяти своего телефона, четко продиктовал мне номер … моего телефона.

Этот конфуз примирил нас, так как его уверенность в свое правоте после этого стала не столь явной, особенно учитывая тот факт, что разговор происходил при людях.
2003. http://is.ifmo.ru/belletristic/more_washing/

Предисловие авторов Н. Туккеля и А. Шалыто к статье «Автоматы и танки»

Статья (http://is.ifmo.ru/download/tanks_new.pdf) была написана почти полтора года назад – в ноябре 2001 г. После этого она была отправлена в издательство «Открытые системы» для публикации в одном из их журналов. Примерно через два месяца мы получили ответ, что статья безусловно интересная, но слишком большая и не подходит по стилю. Мы сократили статью примерно в два раза, перекомпоновали ее, а стиль... Ну что поделаешь, видимо, мы не умеем писать «легко и весело». Далее последовала работа по улучшению статьи, с паузами между ответами редакции в месяц-другой. В результате найти компромисс с редакцией нам не удалось. Наступил июль 2002 г.

Были ли эти восемь месяцев потеряны? Хочется верить, что не совсем, ведь благодаря замечаниям редактора «Открытых систем» **Руслана Богатырева** мы сократили и, во многом, улучшили статью, что позволило нам в конце концов ее опубликовать. А еще за это время мы успели закончить и опубликовать в Интернете полную проектную документацию (http://is.ifmo.ru/projects/tanks/) на рассмотренный в статье пример и начали работу по переводу статьи на английский.

После этого мы отправили уже сокращенную версию статьи в журнал «Программист». Примерно через месяц мы получили краткий ответ, гласящий, что «статья не представляет интереса для широкого круга читателей». Подняв с пола упавшие челюсти, переспросили: что совсем не представляет? В ответ получили довольно жесткую отповедь. Потом помирились, но статью не опубликовали.

До сих пор не совсем понимаем причину отказа. Как может быть неинтересна читателям профессионального программистского журнала статья, которая посвящена изложению новой технологии программирования и в качестве примера рассматривает находящуюся в тот момент на пике популярности программистскую игру *Robocode*, число скачанных копий которой уже давно перевалило за миллион?

Кстати, в данный момент наша статья «существует», а журнал «Программист» – уже нет.

Нас уже начали было терзать сомнения: а вдруг наша статья действительно неудачная и «дурная»? Но эти сомнения исчезли после появления статьи: ***Озеров А.А.*** Четыре танкиста и компьютер // Магия ПК. 2002. № 11 (http://is.ifmo.ru/aboutus/5/).

В ней созданный нами для игры *Robocode* танк отмечен как «вызвавший наибольший интерес», поскольку он хорошо сражается, построен «по науке» и является, по-видимому, единственным, для которого была выпущена проектная документация. Приятно было прочитать следующие слова в свой адрес: «точная математическая модель и использование современных методик программирования позволили российскому танку занять достойное место в лиге *Robocode*».

В результате у нас появилось шутливое (отчасти) подозрение, что в отечественных изданиях, посвященных информационным технологиям, существует негласный запрет на публикации с критикой *UML* – «священной коровы» современного программирования, о которой примерно раз в месяц выходит книга на английском или русском языке. При этом ни в одной из этих книг нет ни

одного проекта, выполненного целиком (постановка задачи – проектирование – реализация). Каждый раз чего-то не хватает: то пишут целые книги про одни случаи использования, то толстые книги про спецификации без примеров, то и вовсе нечто невнятное. Другой тип книг – про реализацию без проектирования.

Вспомнили, что в 47-м номере за 2001 г. еженедельника *PC Week/RE* мы читали статью **Андрея Колесова**, в которой он написал, что преподаватели вузов за полтора года его работы в журнале *BYTE/Россия* не представили ни одной статьи. Написав соответствующее письмо, мы отправили ему сокращенную версию статьи, попросив опубликовать ее в указанном журнале. Довольно быстро нам пришел ответ с согласием на публикацию и с уже до боли привычной оговоркой о необходимости ее дальнейшего сокращения и внесения некоторых изменений. К октябрю 2002 г. мы закончили очередную корректировку статьи и, наконец, окончательно ее согласовали.

А тем временем в январе 2003 г. мы получили рецензию из журнала «Программирование» на полную версию нашей статьи. Она **более года** находилась там на рецензировании. Полученная рецензия была отрицательной. При этом кроме вполне справедливых замечаний рецензента о, например, некоторой субъективности нашей критики в адрес *UML*, рецензия содержала и довольно странные замечания. Одно из них заключалось в том, что громоздкость диаграмм, применяемых в *UML*, является неустранимым недостатком любых графических способов спецификации, и поэтому как недостаток рассматриваться не может. Объяснялось это тем, что всегда найдется система достаточно сложная для того, чтобы ее графическое описание вышло за пределы физиологических способностей человека вне зависимости от используемой графической нотации. Странно, что рецензент упускает из виду тот факт, что при использовании, например, диаграмм *Statecharts* или *SDL* указанный физиологический предел достигается **гораздо** раньше, чем при использовании нотации, предложенной нами, которая в дополнение к этому является еще и более наглядной, и строгой.

Тем, кто в этом сомневается, в очередной раз предлагаем попробовать перерисовать с использованием *Statecharts* или *SDL* граф переходов автомата A0, приведенный в документации на систему управления дизель-генератором (http://is.ifmo.ru/projects/dg/). Вступать в дискуссию по вопросам применения *UML* не стали.

В итоге наша статья, сильно изменившаяся и сокращенная более чем в два раза, была опубликована в февральском номере журнала (*BYTE/Россия*. 2003. № 2, с. 69-73). Мы хотим выразить благодарность, в первую очередь, заместителю главного редактора этого журнала **Андрею Колесову**, а также всем, кто помог нам довести эту статью до «победного конца».

Любые совпадения названий журналов и издательств с реально существующими следует рассматривать как случайные :-).

Дополнение

Первоначально документация на танк была опубликована здесь: http://is.ifmo.ru/projects/tanks/, и мы получили по ней несколько замечаний. Основные замечания состояли в следующем:

- реализация является недостаточно объектной, а имеет место «оборачивание» автоматов классами. В частности, предлагалось реализовывать автоматы отдельно от объектов, которыми они управляют (разделить душу и тело :-));
- реализацию автоматов целесообразно выполнять не с помощью конструкции *Switch,* а применяя паттерн State;
- программа должна быть самодокументирующейся. Названия всех переменных должны быть смысловыми.

Для выяснения справедливости этих замечаний студент кафедры «Компьютерные технологии» Университета ИТМО **Денис Кузнецов** (двукратный призер студенческих командных чемпионатов мира по программированию *ACM* 2000 и 2001 гг.) выполнил рефакторинг нашего проекта (http://is.ifmo.ru/projects/robocode2/). Он изменил структуру кода без изменения его функциональности с целью учета указанных замечаний. Это ему сравнительно просто удалось, так как проект-прототип был полностью документирован, что позволило сохранить как схемы связей и графы переходов автоматов, так и практически все функции входных и выходных воздействий.

В приложении к статье (http://is.ifmo.ru/download/tanks_new.pdf) содержатся два варианта реализации класса «Стрелок» (на основе паттерна State и предложенный в статье, когда автоматы

являются методами классов), в которых обозначен путь от запуска автомата до завершения его работы. Авторы считают, что предложенный в статье подход является более целесообразным.

И последнее – про автоматы. Все больше программистов соглашаются с тем, что обработчики событий необходимо реализовывать не традиционным путем, «размазывая» по ним логику, а на основе автоматов. Это позволяет повысить централизацию логики, что, как и применение объектов, упрощает внесение изменений при повторном использовании кода. Поэтому совместное применение объектной и автоматной парадигм увеличивает эффективность каждой из них.

Как отмечалось выше, сокращенная версия статьи опубликована с названием «Танки и автоматы» в журнале *BYTE/Россия*. Вот ее аннотация: «При разработке программ процессу проектирования часто уделяется мало внимания. Во многом это связано с недостаточной эффективностью известных технологий проектирования. В настоящей работе делается попытка разработки технологии проектирования программ на основе минимального набора документов, описывающих как структурные, так и поведенческие стороны программ. При этом совместно применяются объектно-ориентированный и автоматный подходы, образующие технологию, которая может быть названа **«объектно-ориентированное программирование с явным выделением состояний»**. Эта технология иллюстрируется примером разработки системы управления виртуальным танком в игре *Robocode*.

Подход, излагаемый в настоящей работе, является продолжением работ авторов, направленных на широкое внедрение таких понятий, как «состояние» и «автомат», в инженерное программирование (*Software Engineering*). Он берет начало в теории автоматов и в проектировании систем управления и не ставит своей целью учесть все особенности объектных языков программирования. Настоящая работа является первой редакцией предлагаемого подхода для рассматриваемого класса задач и в дальнейшем может быть усовершенствована. В частности, автоматы могут реализовываться не в виде методов класса, как в настоящей работе, а отдельными классами. Кроме того, каждое состояние может рассматриваться как отдельный объект.

Объектно-ориентированное программирование с явным выделением состояний базируется на двух парадигмах: объектной и автоматной. Это соответствует идее одновременного использования нескольких парадигм, развиваемой в настоящее время в программировании. При этом, если объектный подход предоставляет программисту средства для решения задачи в ее терминах, то автоматный подход – средства для описания поведения объектов в терминах пространства состояний. Применение автоматов проясняет поведение программы, также, как применение объектов проясняет ее структуру.

Авторы надеются, что предложенный подход внесет определенный вклад в инженерное программирование, в необходимости которого, по словам **Г. Буча**, многие сомневаются, а **Б. Гейтс** публично заявляет, что не верит в диаграммы и не желает, чтобы его программисты занимались проектированием (Программы следующего десятилетия // Открытые системы. 2001. № 12). И это при наличии огромного числа ошибок в выпускаемом программном обеспечении, которые в системах, управляющих ответственными объектами, недопустимы вовсе.

Следует отметить, что несмотря на использование автоматов в некоторых известных методологиях объектной разработки программ, явное выделение состояний не характерно для объектной парадигмы. Например, одним из критериев объектно-ориентированного языка является «поддержание им объектов как абстракции данных с определенным интерфейсом поименованных операций и **скрытым** состоянием». Да и само определение объекта, предложенное одним из «трех друзей» (являющихся создателями объектно-ориентированного проектирования) **И. Джекобсоном** (другая нотация его фамилии – **И. Якобсон**), в этом смысле мало что проясняет: «Объект – это сущность, способная сохранять свое состояние (информацию) и обеспечивающая набор операций (поведение) для проверки и изменения этого состояния».

Определение поведения как набора операций является традиционным в объектно-ориентированном программировании. Оно сдерживает применение автоматов в рамках этой парадигмы программирования. Это определение не позволяет конструктивно использовать автоматы, так как при большом числе атрибутов число состояний объекта может быть огромным, что делает невозможным выделение состояний в явном виде.

Предлагаемый в настоящей работе подход устраняет указанные проблемы. Пример, рассматриваемый в статье, выбран из области создания виртуальных роботов, обладающих

средствами обнаружения и уничтожения противника. Каждая из игр этого класса имеет специфику, определяемую средой исполнения, правилами проведения боев и используемым языком программирования.

Исходя из целей настоящей работы, авторов интересовала игра, которая в качестве языка программирования использовала бы не специализированный язык, а один из широко распространенных объектно-ориентированных языков. Единственной известной авторам игрой (в 2001 г.), отвечающей этому требованию, являлась разработанная компанией *Alphaworks* **игра *Robocode*** (http://robocode.alphaworks.ibm.com). В 2002 г. эта ситуация несколько изменилась: другая версия этой игры была предложена в качестве «разминки» для участников командного студенческого чемпионата мира по программированию *ACM*. К играм этого класса относится также и разработанная фирмой *Microsoft* **игра *Terrarium*** (http://www.terrariumgame.net), которая также была реализована на основе предлагаемого подхода (http://is.ifmo.ru/projects/terrarium/).

В работе перечислены существующие, по мнению авторов, недостатки *UML*, являющегося в настоящее время наиболее известным и распространенным в мире графическим языком для визуализации, спецификации, конструирования и документирования программных систем.

Перечислены также особенности предлагаемой технологии, благодаря которым указанные недостатки в ней устранены. Например, вместо построения сценариев при создании программы, в предлагаемой технологии используется исчерпывающее **автоматическое** протоколирование, выполняемое в терминах автоматов. Такое протоколирование может быть полезно при построении **черных** (аварийных) **ящиков**, обеспечивая при необходимости фиксацию **всего** поведения программы.

Это решает одну из главных проблем проектирования, состоящую не в обеспечении безошибочности, которой достичь невозможно, а в создании подходящих способов выяснения, «что именно пошло не так и, как это исправить». Изложенное чрезвычайно важно, так как «протоколы (история вычислений) являются конструкциями, вскрывающими механизм работы программы и, поэтому, постепенно среди теоретиков программирования сложилось представление, что **множество протоколов лучше характеризует программу, нежели сам исходный программный текст**».

Это связано с тем, что «идея доказательства правильности программ в значительной мере исчерпала себя, так как выяснилось, что в общем случае невозможно установить свойства результата работы программы или процедуры, не исполнив ее до конца... В теории вычислений доказывается, что в общем случае распознать определенные свойства в программе можно только динамически, прослеживая весь процесс вычисления при соответствующих входных воздействиях».
2003. http://is.ifmo.ru/works/tanks_new/

Технология автоматного программирования

Излагаются основные положения новой технологии программирования, названной автором «автоматное программирование». При этом графы переходов конечных автоматов используются при спецификации, реализации, отладке и документировании программ. Автоматное программирование является разновидностью синхронного программирования, которое было создано и нашло применение в Европе для разработки систем управления ответственными объектами.

Если хочешь построить лодку, не зови
людей, не бей в барабаны, а всели в
них тоску по бескрайнему синему
морю.
Антуан де Сент-Экзюпери

Что такое автоматное программирование?

В последние годы большое внимание уделяется разработке технологий программирования для встроенных систем и систем реального времени, к которым предъявляются высокие требования по качеству программного обеспечения. Одним из наиболее известных подходов в этом направлении является синхронное программирование [1].

Параллельно с развитием в Европе синхронного программирования, в России создается подход к разработке программного обеспечения, названный «автоматное программирование» [2-4], который можно рассматривать в качестве разновидности синхронного программирования.

В настоящей работе описываются основные положения «автоматного программирования». Оно поддерживает такие этапы создания программного обеспечения как проектирование, реализация, отладка и документирование.

Если в программировании в последнее время все шире используется понятие «событие», то предлагаемый подход базируется на понятии «состояние». Добавляя к нему понятие «входное воздействие», которое может быть входной переменной или событием, вводится термин «автомат без выхода». Добавляя к последнему понятие «выходное воздействие», вводится термин «автомат» (конечный, детерминированный).

Поэтому область программирования, базирующаяся на этом понятии, была в работе [4] названа «автоматное программирование», а процесс создания таких программ – «автоматное проектирование программ».

Особенность рассматриваемого подхода состоит в том, что при его использовании автоматы задаются графами переходов, для различения вершин, в которых вводится понятие «кодирование состояний». При выборе «многозначного кодирования» с помощью одной переменной можно различить состояния, число которых совпадает со значностью выбранной переменной. Это позволило ввести в программирование такое понятие, как «наблюдаемость программ».

В рамках предлагаемого подхода программирование выполняется «через состояния», а не «через переменные» (флаги), что позволяет лучше понять и специфицировать задачу и ее составные части.

При этом необходимо отметить, что в автоматном программировании отладка проводится путем протоколирования в терминах автоматов.

В силу того, что в рамках этого подхода от графа переходов к тексту программы предлагается переходить формально и изоморфно, то при применении языков программирования высокого уровня это наиболее рационально выполнять с помощью конструкции *switch* языка Си либо ее аналогов в других языках программирования. Поэтому технологию автоматного программирования в работе [4] было решено назвать «*Switch*-технология».

В настоящее время эта технология разрабатывается в нескольких вариантах, различающихся как классом решаемых задач, так и типом вычислительных устройств, на которых осуществляется программирование.

Логическое управление

В 1996 г. Российский фонд фундаментальных исследований (РФФИ) в рамках издательского проекта № 96-01-14066 поддержал издание работы [4], в которой предлагаемая технология была изложена применительно к системам логического управления, в которых события отсутствуют, выходные воздействия являются двоичными переменными, а операционная система работает в режиме сканирования. Системы этого класса реализуются обычно на программируемых логических контроллерах, которые имеют относительно небольшой объем памяти, а их программирование выполняется на таких специфических языках, как, например, язык функциональных блоков [5]. В работе [4] предложены методы формального написания программ для таких языков при задании спецификации для разрабатываемого проекта системой взаимосвязанных графов переходов. Показаны преимущества использования языка графов переходов по сравнению с языком «Графсет».

Программирование с явным выделением состояний

В дальнейшем автоматный подход был распространен на событийные системы, которые называются также «реактивными» [6]. В них указанные выше ограничения сняты. Как следует из названия этих систем, в них среди входных воздействий используются события, в качестве выходных воздействий применяются произвольные процедуры, а в качестве операционных систем – любые операционные системы реального времени.

Для программирования событийных систем с применением автоматов был использован процедурный подход, и поэтому такое программирование было названо «программирование с явным выделением состояний» [7].

При этом выходные воздействия «привязаны» к дугам, петлям или вершинам графов переходов (применяются смешанные автоматы — автоматы Мура-Мили). Это позволяет в компактном виде представлять последовательности действий, которые являются реакциями на соответствующие входные воздействия.

Особенность предлагаемого подхода к программированию этого класса систем состоит в том, что в них повышается централизация логики за счет устранения ее в обработчиках событий и формирования системы взаимосвязанных автоматов, которые вызываются из обработчиков [8]. Автоматы между собой могут взаимодействовать по вложенности, вызываемости и за счет обмена номерами состояний.

Последний вид взаимодействия рассматривался также в работе [9], в которой утверждается, что «указанное взаимодействие может оказаться мощным средством при проверке программ».

Система взаимосвязанных автоматов образует системонезависимую часть программы, а системозависимая часть формируется из функций входных и выходных воздействий, обработчиков событий и т. д.

Другая важнейшая особенность описываемого подхода состоит в том, что при его применении автоматы используются триедино: при спецификации; при программировании (сохраняются в программном коде); при протоколировании, выполняемом, как указано выше, в терминах автоматов. Последнее позволяет контролировать правильность функционирования системы автоматов. Протоколирование выполняется автоматически по построенной программе и может использоваться для задач большой размерности при сложной логике программы.

При этом каждый построенный протокол может рассматриваться в качестве соответствующего сценария. Отметим, что для «больших» задач невозможно применение диаграмм последовательностей и диаграмм кооперации, входящих в состав языка *UML* [10], так как при использовании этого языка указанные диаграммы предлагается строить вручную на этапе проектирования в то время, как в автоматном программировании протоколы строятся автоматически при выполнении программы.

Протоколы позволяют наблюдать за ходом выполнения программы и демонстрируют тот факт, что автоматы являются не «картинками», а реально действующими сущностями.

Автоматный подход предлагается применять не только при создании системы управления, но и при моделировании объектов управления.

Этот подход был апробирован при разработке системы управления судовыми дизель-генераторами [11]. Система была специфицирована более чем тридцатью взаимодействующими автоматами. Для описания модели дизеля также использовались автоматы. При проектировании на каждый автомат выпускалось четыре документа: словесное описание («декларация о намерениях»), схема связей (поясняющая на русском языке символы, входящие в интерфейс автомата), граф переходов (с символьными обозначениями событий, входных переменных и выходных воздействий), текст программного модуля, который формально и изоморфно реализует граф переходов (также без использования смысловых идентификаторов и комментариев). Эти документы заменяют самодокументирующиеся программы, содержащие смысловые идентификаторы и комментарии, так как эти средства при сложной логике программы не обеспечивают должного понимания программ и их пригодности для дальнейшей модификации [12]. Эту проблему при сложной логике не решают и самодокументирующиеся графы переходов [10].

Выполненный проект подтвердил целесообразность использования протоколов для проверки корректности взаимодействия столь большого количества автоматов и каждого из них в отдельности.

Реализация указанного проекта была осуществлена на вычислительной системе с архитектурой *ix86*. В дальнейшем этот подход был развит **Н.И. Туккелем** применительно к созданию систем на микроконтроллерах. При этом все проектирование можно выполнять на персональном компьютере,

используя автоматное программирование, и переносить полученное программное обеспечение на микроконтроллер только на последней стадии разработки.

Объектно-ориентированное программирование с явным выделением состояний

Для решения широкого круга задач весьма эффективен подход, основанный на совместном использовании объектной и автоматной парадигм, который в работе [13] был назван «объектно-ориентированное программирование с явным выделением состояний». Особенности этого подхода состоят в следующем. Так же, как и в машине Тьюринга [14] явно выделены управляющие (автоматные) состояния объекта, число которых значительно меньше числа остальных состояний, например, «вычислительных». В программирование введено понятие «пространство состояний», под которым понимается множество управляющих состояний объекта. Это обеспечивает существенно более понятное поведение по сравнению со случаем, когда такое пространство или ориентация в нем отсутствует. Предложен минимальный набор документов, позволяющий наглядно и строго описывать как структурные (статические), так и поведенческие (динамические) стороны программ.

Как и при использовании любого другого подхода, применение предлагаемого связано с множеством эвристик, возвратов назад, уточнений и параллельно выполняемых работ. Однако после завершения создания программы предлагаемый подход может быть сформулирован (по крайней мере, для ее документирования) как «идеальная» технология, фиксирующая принятые решения [15].

1. На основе анализа предметной области выделяются классы, и строится диаграмма классов.
2. Для каждого класса разрабатывается словесное описание, по крайней мере, в форме перечня решаемых задач.
3. Для каждого класса создается структурная схема, отражающая его интерфейс и структуру. При этом атрибуты и методы разделены на автоматные и остальные.
4. При наличии в классе нескольких автоматов строится схема их взаимодействия.
5. Для каждого автомата разрабатываются словесное описание, схема связей, граф переходов.
6. Каждый класс реализуется соответствующим модулем программы. Его структура должна быть изоморфна структуре класса, а методы, соответствующие автоматам, реализованы по шаблону, приведенному в работе [8].
7. Для отладки системы и подтверждения правильности ее работы автоматически строятся протоколы, в которых функционирование объектов, содержащих автоматы, описывается в терминах состояний, переходов, событий, входных и выходных воздействий.
8. Выпускается проектная документация, составной частью которой является программная документация.

Из опыта применения изложенной технологии можно сделать вывод, что применение автоматов проясняет поведение программы также, как использование объектов делает прозрачной ее структуру.

Описанный подход был использован при создании системы управления «танком» для игры *Robocode* [15].

В отличие от систем управления сотнями других танков, на этот танк выпущена подробная проектная документация, содержащая, в частности, графы переходов и схемы связей автоматов, реализующих функциональность танка.

Детальные протоколы поведения танка позволяют проследить все течение боя. Метод построения протоколов, возможно, является основой для новой концепции построения «черных ящиков».

В описанной технологии автоматы применялись как методы классов. В рамках этой технологии могут быть использованы и другие подходы к объектной реализации автоматов, изложенные, например, в работах [16-18]. Автоматы могут выступать, в частности, как объекты-наследники определенного класса, реализующего базовую функциональность автоматов, обусловленную семантикой *Switch*-технологии.

Возможно также использование классов, реализующих понятия «состояние» или «группа состояний».

При проектировании автоматных программ может быть использован паттерн *State* [19] или другие паттерны.

Особенности автоматной реализации параллельных процессов на основе механизма обмена сообщениями рассмотрены в работе [20]. Еще один подход к автоматной реализации параллельных процессов описан в работе [21].

Наличие качественной проектной документации резко упрощает осуществление рефакторинга программы (изменение ее структуры при сохранении функциональности). Последнее подтверждается рефакторингом упомянутой выше системы управления танком, выполненным с целью повышения «объектности» программы [22].

Вычислительные алгоритмы

Автоматный подход используется в настоящее время и при реализации вычислительных алгоритмов [23-25].

Так, в частности, показано, что произвольный итеративный алгоритм может быть реализован конструкцией, эквивалентной циклу *do–while*, телом которого является оператор *switch*.

На основе автоматов предложен новый подход к построению визуализаторов алгоритмов, используемых на кафедре «Компьютерные технологии» СПбГУИТМО при обучении программированию и дискретной математике [26]. Подход позволяет представить логику работы визуализаторов системой взаимосвязанных конечных автоматов. Система состоит из пар автоматов, каждая из которых содержит «прямой» и «обратный» автоматы, обеспечивающие пошаговое выполнение алгоритма вперед и назад соответственно.

Движение за открытую проектную документацию

27 ноября 2002 г. на открытии полуфинальных соревнований командного чемпионата мира по программированию *ACM* (*Association for Computing Machinery*) в Северо-восточном Европейском регионе было объявлено об организации «Движения за открытую проектную документацию» [27, 28]. В рамках этого движения на сайте http://is.ifmo.ru создан раздел «Проекты», в котором размещено около 40 проектов разработки программного обеспечения на основе автоматного подхода. Перечислим некоторые из них:

- автоматная реализация интерактивных сценариев образовательной анимации с использованием *Macromedia Flash*;
- построение визуализаторов алгоритмов для обучения дискретной математике и программированию;
- обучающая и тестирующая программа с примером настройки для изучения английского языка;
- совместное использование теории построения компиляторов и *Switch*-технологии;
- скелетная анимация;
- управление различными технологическими процессами и объектами (упрощенная модель цеха холодной прокатки, упрощенная модель автомобиля, дизель-генератор, турникет, кодовый замок, светофор, кофеварка, телефон, банкомат, лифт, система безопасности банка и т.д);
- игры (*Terrarium*, *Robocode*, *CodeRally*, «Морской бой», *Lines* [27], *Bomber* [27], *Tron*, «Однорукий бандит», «Завалинка»);
- моделирование игры «Пятнашки» для роботов *LEGO*;
- *XML*-формат для описания внешнего вида видеопроигрывателя *CrystalPlayer* (www.crystalplayer.com);
- примеры клиент-серверных приложений;
- построение пользовательских интерфейсов;
- реализация сетевого протокола *SMTP*;
- классические «параллельные» задачи («Синхронизация цепи стрелков», «Обедающие философы»);
- управление в задачах логистики.

«Коллекция» проектов пополняется и будет пополнятся в дальнейшем.

Заключение

Одна из целей настоящей работы состоит в том, чтобы показать, что автоматы в программировании служат не только «для распознавания цепочек символов» [29] и управления «стиральными машинами». Кроме того, в работе показано, что автоматы являются не просто одной из

математических моделей дискретной математики, а могут применяться при реализации любых программ, обладающих сложным поведением.

Использование автоматов упрощает формализацию спецификации программы, определяющей ее поведение и играющей «ключевую роль в вопросе сдерживания программных ошибок» [30].

Предлагаемая технология должна ответить на вопрос, поставленный в [31]: «теорию конечных автоматов мы проходили, но, причем здесь программирование?»

Отметим также, что при решении задач логического управления используется стиль программирования «от состояний» (по классификации, предложенной в работе [32]). Программирование с явным выделением состояний основано на таких стилях, как «программирование от состояний» и «программирование от событий». Объектно-ориентированное программирование с явным выделением состояний объединяет объектно-ориентированный стиль программирования и два стиля, указанных выше.

Работа выполнена при поддержке РФФИ по гранту №02-07-90114 «Разработка технологии автоматного программирования».

Литература

1. ***Benveniste A., Caspi P., Edwards S. et al.*** The Synchronous Languages 12 Years Later // Proceedings of the IEEE. Vol. 91. 2003. № 1.

2. ***Шалыто А.А.*** Алгоритмизация и программирование для систем логического управления и «реактивных» систем. Обзор //Автоматика и телемеханика. 2001. № 1. https://www.mathnet.ru/links/5f04c69b3ec6e71c6a5bdd2ab43a31c0/at1715.pdf.

3. ***Шалыто А.А.*** Логическое управление. Методы аппаратной и программной реализации алгоритмов. СПб.: Наука, 2000. https://is.ifmo.ru/books/log_upr/1.

4. ***Шалыто А.А.*** *Switch*-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. https://is.ifmo.ru/books/switch/1.

5. ***Шалыто А.А.*** Реализация алгоритмов логического управления программами на языке функциональных блоков //Промышленные АСУ и контроллеры. 2000. № 4. https://www.avrorasystems.com/ru/Data/Pressroom/Files/asu2.pdf.

6. ***Harel D., Politi M.*** Modeling Reactive Systems with Statecharts. NY: McGraw-Hill, 1998.

7. ***Шалыто А., Туккель Н.*** Программирование с явным выделением состояний //Мир ПК. 2001. № 8, 9. https://is.ifmo.ru/works/mirk/.

8. ***Шалыто А.А., Туккель Н.И.*** *SWITCH*-технология – автоматный подход к созданию программного обеспечения «реактивных» систем //Программирование. 2001. № 5. http://is.ifmo.ru/download/switch.pdf.

9. ***Дейкстра Э.*** Взаимодействие последовательных процессов // Языки программирования. М.: Мир, 1972.

10. ***Буч Г., Рамбо Д., Джекобсон А.*** Язык UML. Руководство пользователя. М.: ДМК, 2000.

11. ***Шалыто А.А., Туккель Н.И.*** Проектирование программного обеспечения системы управления дизель-генераторами на основе автоматного подхода // Системы управления и обработки информации. 2003, вып. 5. http://is.ifmo.ru/works/diesel/.

12. ***Безруков Н.*** Повторный взгляд на «собор» и «базар» //BYTE/Россия. 2000. № 8.

13. ***Шалыто А.А., Туккель Н.И.*** Объектно-ориентированное программирование с явным выделением состояний //Материалы международной научно-технической конференции «Искусственный интеллект – 2002». Т.1. Таганрог-Донецк: ТГРУ-ДИПИИ, 2002.

14. ***Шалыто А., Туккель Н.*** От тьюрингова программирования к автоматному //Мир ПК. 2002. № 2. http://is.ifmo.ru/?i0=works&i1=turing.

15. ***Шалыто А.А., Туккель Н.И.*** Танки и автоматы //BYTE/Россия. 2003. № 2. http://is.ifmo.ru.

16. ***Гуров В.С., Нарвский А.С., Шалыто А.А.*** Автоматизация проектирования событийных объектно-ориентированных программ с явным выделением состояний //Труды X Всероссийской научно-методической конференции «Телематика–2003». Т.1. СПб.: СПбГИТМО (ТУ), 2003.

17. ***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». Т.1. СПб.: СПбГИТМО (ТУ), 2003.

18. ***Корнеев Г.А., Шалыто А.А.*** Реализация конечных автоматов с использованием объектно-ориентированного программирования //Труды X Всероссийской научно-методической конференции «Телематика-2003». Т.2. СПбГИТМО (ТУ), 2003.
19. ***Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж***. Приемы объектно-ориентированного программирования. Паттерны проектирования. СПб.: Питер, 2001.
20. ***Гуисов М.И., Кузнецов А.Б., Шалыто А.А.*** Интеграция механизма обмена сообщениями в *Switch*-технологию. **2003.** https://is.ifmo.ru/download/memech.pdf
21.. ***Любченко В.*** О бильярде с Microsoft Visual C++ 5.0 // Мир ПК 1998. № 1.
22. ***Кузнецов Д., Шалыто А.*** Система управления танком для игры «Robocode». Вариант 2. Проектная документация. 2003. https://is.ifmo.ru/projects/robocode2/.
23. ***Шалыто А., Туккель Н., Шамгунов Н.*** Ханойские башни и автоматы //Программист. 2002. № 8. http://is.ifmo.ru/works/hanoy/.
24.. ***Шалыто А., Туккель Н., Шамгунов Н.*** Задача о ходе коня //Мир ПК. 2003. № 1. http://is.ifmo.ru/works/knight/.
25. ***Шалыто А.А., Туккель Н.И.*** Преобразование итеративных алгоритмов в автоматные // Программирование. 2002. № 5. http://is.ifmo.ru/works/iter/.
26. ***Корнеев Г.А., Казаков М.А., Шалыто А.А.*** Построение логики работы визуализаторов алгоритмов на основе автоматного подхода //Труды X Всероссийской научно-методической конференции «Телематика-2003». Т.2. СПбГИТМО (ТУ), 2003.
27. ***Шалыто А.А.*** Новая инициатива в программировании – «Движение за открытую проектную документацию» //Мир ПК – Диск. 2003. № 8.
28. ***Шалыто А.А.*** Новая инициатива в программировании – «Движение за открытую проектную документацию» //Мир ПК. 2003. № 9. https://is.ifmo.ru/works/open_doc/.
29. ***Хопкрофт Д., Мотвани Р., Ульман Д.*** Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.
30. ***Аллен Э.*** Типичные ошибки проектирования. СПб.: Питер, 2003.
31. ***Чижов А.*** chizh@irk.ru.
32. ***Непейвода Н.Н., Скопин И.Н.*** Основания программирования. Москва-Ижевск: Институт компьютерных исследований, 2003.
https://is.ifmo.ru/works/tech_aut_prog. Статья опубликована в журнале «Мир ПК», 2003. №10, стр.74-78.

Скромное обаяние автоматного программирования

В 1998 г. мне исполнилось пятьдесят лет, и, как отмечено выше, вышла моя книга о *SWITCH*-технологии (http://is.ifmo.ru/books/switch/1). Из рецензии на эту книгу: «Недавно в редакции *PC Magazine/RE* появилась книга ***Шалыто А.А. SWITCH*-технология. Алгоритмизация и программирование задач логического управления.** СПб.: Наука, 1998. В монографии показывается, каким образом можно **пересмотреть концепции программирования**, если рассматривать программы как конечные автоматы. **Представлять современные системы, управляемые событиями, именно таким образом настолько естественно, что единственная мысль, которая возникла у меня после внимательного ознакомления с книгой, была: «*Почему до этого никто раньше не додумался?*» (*Герр Р.* Новый поворот** // *PC Magazine/RE*. 1998. № 10, с. 88-90. http://is.ifmo.ru/recensions/gerr/). Интересно, что Герр не написал, как любят писать и говорить некоторые: «Что тут такого – ведь так делают все» (https://vk.com/@1077823-ob-avtomatizacii-stiralnyh-mashin).

Когда люди «наезжают» на меня за слишком активную пропаганду автоматного программирования, то я им скромно сообщаю о том, что «более 99% всех микропроцессоров, проданных в мире в 1998 г., применялись во встроенных системах», в которых автоматы могут эффективно использоваться (***Ослэндер Д., Риджли Д., Ринггенберг Д.*** Управляющие программы для механических систем. Объектно-ориентированное проектирование систем реального времени. М.: БИНОМ, 2004). При этом разговор сразу сводится к тому, что сложность программ распределяется обратно пропорционально – в одном проценте оставшихся микропроцессоров выполняется 99% программ других типов.

Точность последнего утверждения, однако, обычно ничем не подтверждается. Более того, **Бертран Мейер**, выступая 2.06.2006 г. в СПбГУ ИТМО, сказал, что число микропроцессоров в одном современном автомобиле достигает 65, а объем программ в них уже превышает 10 млн строк.

Как вы думаете, какого типа программы используются в автомобиле?

Вчера один человек сказал мне: «**Хороший у Вас подход. Я на шесть лет увольнялся, а когда вернулся, полчаса посмотрел на графы переходов для дизель-генератора, и все вспомнил!**».

Вчера же, видимо, для того чтобы меня «уесть», другой человек сказал: «То, чем Вы занимаетесь, не наука, а методология или технология создания программного обеспечения». На это я ответил словами член-корреспондента РАН **Г.Г. Рябова**, который сказал мне, что «**технология – это выжимка из науки**». Моему собеседнику такой ответ не понравился.

Один молодой кандидат наук при мне спросил Бертрана Мейера: «Если поступить к Вам в постдокторантуру, то там можно будет заниматься чем-то, отличным от исследований в области языка ***Eiffel***?». Мейер ответил: «Нет. И не по тому, что я считаю, что нет других достойных тем для исследований в *Software Engineering* или в *Computer Science*, а просто **у меня жизнь одна и ресурсы ограничены**».

При этом я заметил, что поступаю аналогично применительно к автоматному программированию. На это Бертран Мейер по-русски сказал: «И правильно!».

Теперь приведу несколько слов академика РАН **В.Б. Кудрявцева**, заведующего кафедрой «Математической теории интеллектуальных систем» мехмата МГУ **об особой значимости автоматов в кибернетике**: «Из всех видов управляющих систем наиболее полно вобрала в себя черты управляющей системы модель автомата, поскольку она фактичекски получается путем лишь сужения множеств значений всех параметров управляющей системы до конечных множеств. При аппроксимационном подходе к изучению управляющих систем, состоящем в допущении только конечных множеств значений всех характеристик этих систем, по существу приводят к **модели автомата, которая тем самым обретает свойство универсальности и поэтому имеет особую значимость в кибернетике**» (http://www.intsys.msu.ru/staff/kudryavtsev/MaTIS.pdf).

Теперь фрагменты из выступлений на семинаре *Software 2000: a View of the Future* (10.04.1994). **Brian Randell:** «Я помню Дуга Росса из компании *SofTech*, который много лет назад говорил, что 80 или даже 90% информатики будет в будущем основываться на теории конечных автоматов. Большая часть теории автоматов была успешно использована в системных программах и текстовых фильтрах в *OS UNIX*. Это позволяет множеству людей работать на высоком уровне и разрабатывать очень эффективные программы». **Herve Gallaire**: «Я знаю людей из Боинга, занимающихся системами стабилизации самолетов с использованием чистой теории автоматов. **Даже трудно себе представить, что им удалось сделать с помощью этой теории**».

Кто-то может сказать, что прошло много лет, а широкого применения автоматов в программировании так и не наступило, но я, все-таки, верил и верю, что указанное время наступит.

2006. http://is.ifmo.ru/belletristic/obayanie/

P.S. 1. В 2021 г. мой аспирант **Ю.Ю. Янкин** защитил диссертацию в НПО «Аврора», в которой, в частности, описано внедрение технологии автоматного программирования в десятки систем, управляющих одним классом судового оборудования. Он предполагает продолжить внедрение этой технологии и в дальнейшем. После этого мне, как и режиссеру **М. Ромму**, ничего не остается сказать кроме «И все-таки я верю» (https://www.youtube.com/watch?v=dHIGtB13vYQ). И это не только мое мнение. Профессор **Валерий Вяткин** предполагает, что расцвет применения автоматов при создании ПО в промышленной автоматизации произойдет, если нефте-газовые компании мира будут проводить модернизацию своих систем управления на основе стандарта *IEC 61499* (***Pang C., Patil S., Yang C., Vyatkin V., Shalyto A.*** A Portability Study of IEC 61499: Semantiac and Tools / 12th IEEE International Conference on Industrial Informatics (INDIN 2014). 2014. Port Alegre, Brazil, pp. 440-445. https://ieeexplore.ieee.org/document/6945553).

2. Кроме пользы автоматного программирования для практики, оно сыграло очень важную роль для развития научной деятельности в области *Computer Science* на кафедре «Компьютерные технологии» Университета ИТМО. У нас обучалось много умных и образованных ребят, некоторые из которых хотели заниматься наукой, но кроме тем по физике, а иногда по математике, им ничего не предлагали. **Я решил переломить ситуацию** и сделать так, чтобы на кафедре с таким названием в этой области проходило не только обучение, но и научная деятельность студентов, а в дальнейшем и аспирантов. У меня в «загашнике» было только автоматное программирование, которое я создавал в НПО «Аврора» для совершенствования разработки программного обеспечения судовых систем управления. В ИТМО я читал на эту тему лекции и принудительно «внедрял» его через курсовые работы, тратя на это уйму времени (http://is.ifmo.ru/projects/). Еще больше времени на них тратили

студенты, и поэтому для многих из них было естественным развивать эту тему в бакалаврских работах и магистерских диссертациях (http://is.ifmo.ru/diploma-theses/). Потом некоторые из этих ребят захотели защищать кандидатские диссертации, и поэтому они продолжали исследования в области автоматного программирования.

Это позволило мне, как **Мальчишу-Кибальчишу**, «ночь простоять и день продержаться», и за это время некоторые ребята почувствовали вкус к науке, а после того как мы выиграли грант на тему «**Технология генетического программирования для генерации автоматов управления системами со сложным поведением**» (http://is.ifmo.ru/genalg/_2007_01_report-genetic.pdf, http://is.ifmo.ru/genalg/_2007_02_report-genetic.pdf, http://is.ifmo.ru/genalg/_2007_02_report-genetic.pdf, http://is.ifmo.ru/genalg/_2007_04_report-genetic.pdf) у них, в частности, возник интерес и к другим разновидностям искусственного интеллекта, например, к эволюционным алгоритмам. После этого мы выиграли грант на тему: «**Разработка методов автоматической генерации тестов на основе эволюционных алгоритмов**».

После моего знакомства с академиком РАН **К.Г. Скрябиным** ребята занялись биоинформатикой, а потом и системной биологией. Этому посвящена четырехчастная статья «Как биоинформатика и системная биология появились на кафедре «Компьютерные технологии» Университета ИТМО»:
2009-2011, https://vk.com/@1077823-kak-bioinformatika-i-sistemnaya-biologiya-poyavilis-na-na-ka;
2012-2016, https://vk.com/@1077823-chast-2-2012-2016-kak-bioinformatika-i-sistemnaya-biologiya;
2017, 2018, https://vk.com/@1077823-chast-3-2017-2018-gg-kak-bioinformatika-i-sistemnaya-biologi;
2019, 2020, https://vk.com/@1077823-chast-4-kak-bioinformatika-i-sistemnaya-biologiya-poyavilis).

В результате ребята стали заниматься тем, к чему у них больше лежит душа, и это чаще всего не относилось к автоматному программированию. При этом я всегда помнил, помню и сейчас, слова **П.Л. Капицы**: «Молодым надо не мешать работать». Однако я пошел дальше: считаю, что в этом высказывании слово «работать» – лишнее. Поэтому стараюсь молодым не мешать, а там, где могу – помогать.
3. Мой старинный знакомый **Руслан Богатырев**, который в свое время в журнале «Мир ПК» публиковал мои статьи про автоматное программирование (например, ***Шалыто А.А.*** Технология автоматного программирования // Мир ПК. 2003. № 10, с.74-78. http://is.ifmo.ru/works/tech_aut_prog), так прокомментировал этот текст: «**В программировании есть свой Калашников. И его фамилия – Шалыто**». Забавно…
https://vk.com/@1077823-skromnoe-obayanie-avtomatnogo-programmirovaniya

Применение конечных автоматов при программировании мобильных устройств
По материалу из книги ***Салмре И.*** Программирование мобильных устройств на платформе *.Net Compact Framework*. М.: Вильямс. 2006. Кооперайт на английском языке – *Pearson Education, Inc.*, 2005.

Аналогичные идеи в части применения автоматов в программировании развиваются мною (А.Ш.) **с 1991 года** (http://is.ifmo.ru/works/switch_prr/). В книге (возможно, только в переводе), почему-то, отсутствует список литературы.

Эта книга становится своего рода библией в быстро развивающемся мире мобильных вычислений. Ее автор работал в компании *Microsoft* в Редмонде на протяжении 12 лет. Сейчас он работает в Европейском Инновационном Центре компании *Microsoft* в городе Аахене, Германия, где занимается исследованиями в области разработки программных моделей для современных мобильных устройств.

В книге глава 5 (из 17) называется «**Наш друг конечный автомат**». Учитывая, что автор не является специалистом по конечным автоматам, наличие указанной главы в начале книги свидетельствует о значительной роли автоматов в проектировании мобильных устройств.

Введение
Несмотря на стремительную поступь технологий, надежные методы прикладного программирования и искусство программирования развиваются крайне медленно.

Путь, позволяющий преодолеть трудности, свойственные разработке программного обеспечения, пролегает через использование подходящих методологий.

В эпоху пакетной обработки безраздельно царствовали алгоритмы. Этот период можно назвать эрой «блок-схем».

При построении серверных приложений, отвечающих на запросы, большую роль играет «отсутствие состояния» – нет нужды сохранять состояния между двумя последовательными запросами.

При построении удачного интерактивного приложения, управляемого событиями, многое зависит от того, продумана ли модель управления состояниями.

Конечный автомат – весьма удобная концепция, которую целесообразно использовать для структурирования приложений.

Продуманное применение конечных автоматов облегчает организацию и сопровождение, как логики пользовательского интерфейса, так и логики приложения. Благодаря этому код будет более гибким и надежным, причем это относится к разработке не только мобильных, но и любых других приложений.

Поскольку мобильные приложения должны использовать пространство экрана и системные ресурсы эффективно, конечные автоматы оказываются особенно полезными при разработке ПО для таких приложений.

Ниже излагается, каким образом применение автоматов может способствовать написанию эффективного и надежного года.

«Состояние» Вашего приложения может быть определено как совокупность значений всех его переменных. Эти значения представляют уникальное состояние, которое изменяется каким-либо внешним событием. В любой момент времени приложение находится в каком-то определенном состоянии.

Конечный автомат, как отмечалось выше, является средством формальной структуризации приложения. Вместо того чтобы использовать все переменные приложения в качестве расширенного определения его состояния, конечный автомат имеет единственную переменную, в которой хранится информация о состоянии приложения. Такой переменной обычно является элемент перечисления некоторого множества состояний, определяемого глобально или на уровне класса.

Мощь подхода, использующего конечные автоматы, обусловлена тем, что он позволяет в явном виде определить действительные состояния для некоторого аспекта Вашего приложения и задать соответствующие варианты поведения при переходах приложения из одного состояния в другое.

В приложении может использоваться несколько конечных автоматов, для каждого из которых определяется собственный набор вариантов поведения приложения, подлежащих структуризации.

Приложение, для которого конечные автоматы не определены, в действительности является приложением с множеством конечных автоматов. При этом каждая переменная, определенная на уровне приложения, по сути дела сама является конечным автоматом, и любой код может получить доступ к состоянию и изменить его.

Конечные автоматы используются для формирования набора взаимосвязанных переменных или вариантов поведения и их логической организации, что облегчает обработку состояний.

Конечные автоматы – суть организованное поведение. Конечный автомат – это просто формальное описание того, как работает приложение. Он позволяет ясно представить различные состояния, в которых может находиться приложение.

Для определения текущего варианта изменения состояния удобно использовать блок операторов *switch/case*. Каждый оператор *case* соответствует одному из вариантов изменения состояния и должен содержать вызов функции, выполняющей всю необходимую для этого работу.

Подобного рода централизация и инкапсуляция управления состояниями является одним из наиболее мощных аспектов использования конечных автоматов. Все важные изменения состояний приложения определяются и обрабатываются централизованно в одном месте программы.

1. Явно и неявно определенные конечные автоматы
Для написания кода существуют два подхода. (В моих работах, в частности 2001 г., высказывается аналогичное мнение (http://is.ifmo.ru/works/mirk/, А.Ш.).

Подход 1. Зависящее от специфики конкретной ситуации, децентрализованное, **неявное управление состояниями** (**неудачный подход**).

Различные аспекты состояния приложения изменяются в разных местах приложения. Переменные, используемые для хранения ключевой информации о состоянии, изменяются непосредственно в тех строках кода, где это оказывается необходимым, а загрузка данных и освобождение памяти от них распределяются по всему приложению в зависимости от конкретной ситуации. Развитие событий напоминает «перетягивание каната» между различными частями приложения, поскольку каждая из функций делает все необходимое для выполнения возложенных на нее задач, не обращая никакого внимания на остальную часть приложения.

Подход 2. Плановое, централизованное, **явное управление состояниями** (**удачный подход**).

Явное управление состояниями – прямая противоположность предыдущему подходу. В этом случае любые переходы между состояниями реализуются в рамках одной главной функции. Например, в коде обработчика событий, ответственном за изменение некоторого аспекта состояния приложения, это обеспечивается вызовом единственной функции, которая вызывается в соответствии с логикой приложения. При этом используется инкапсуляция. Вместо того чтобы непосредственно изменять состояние интерфейса, коды обработки событий каждого из элементов пользовательского интерфейса вызывают главную функцию управления состояниями, которая и выполняет необходимую работу.

Этот процесс легко масштабируется при расширении или изменении приложения. Необходимые изменения любых аспектов функционирования приложения обеспечиваются за счет использования единственной главной функции. По мере возникновения потребности в дополнительных элементах управления или состояниях приложения, они могут быть без труда включаться в программную модель централизованным образом.

Использование конечных автоматов облегчает эффективную организацию кода, что, в свою очередь, способствует созданию качественных приложений.

Отдаете ли Вы себе в этом отчет или нет, но значительная доля кода нашего приложения всегда будет связана с управлением его состояниями. Как бы то ни было, Вам придется иметь дело с задачами подобного рода, и только Вам решать, выбрать ли явный подход к управлению состояниями приложения или же остановиться на модели неявного управления.

Явное управление состояниями имеет ряд существенных преимуществ, не самым последним из которых является то, что **такой подход заставляет Вас более тщательно продумывать функционирование Вашего приложения**.

При проектировании пользовательских интерфейсов мобильных приложений применение конечных автоматов, позволяющих централизованным образом экспериментировать с логикой размещения и масштабирования элементов управления, оказывается особенно полезным.

2. Сколько конечных автоматов должно быть в приложении?
Конечные автоматы должны быть центральной составляющей проекта мобильного приложения. Часто бывает удобно применять несколько конечных автоматов в одном приложении.

Конечный автомат, который управляет кэшированными данными, извлеченными из базы данных, может использоваться наряду с конечным автоматом, управляющим такими графическими объектами, как перья и кисти, кэшируемыми для их использования в обычных задачах рисования. Третий конечный автомат может управлять выполнением задач фоновыми потоками.

И хотя перечисленные автоматы можно было бы объединить в один «главный автомат», такой автомат представлял бы просто формальное объединение всех возможных случаев изменения состояния приложения и не дал бы ничего нового, поскольку состояния, которыми управляют отдельные конечные автоматы, слабо или вообще не коррелируют друг с другом.

Возможна крайность другого рода, когда вместо создания одного конечного автомата, выполняющего всю работу, предпочитают иметь десятки небольших конечных автоматов, каждый из которых управляет отдельной переменной приложения.

Роль ключа в нахождении правильного решения играет корреляция. Если можно выделить набор переменных, объектов или ресурсов, которые тем или иным образом коррелируют между собой, то управление этим набором удобно и целесообразно осуществлять при помощи одного конечного автомата.

3. Области применения конечных автоматов при программировании мобильных устройств

3.1. Конечные автоматы и пользовательские интерфейсы

Пользовательские интерфейсы – одна из наиболее очевидных областей, где конечные автоматы могут находить широкое применение. Поведение элементов управления, совместно использующих одну и ту же форму, часто является коррелированным. Некоторые элементы управления отображаются и скрываются в одно и то же время, некоторые элементы перемещаются при отображении других элементов и т. д. (Посмотрите, как предлагается создавать скины для видеоплейера (http://is.ifmo.ru/projects/crystal/, А.Ш.) С каждой из форм, которую вы конструируете, должен связываться конечный автомат, управляющий визуальным поведением формы. Если в самой форме содержится несколько страниц, например, диалоговые окна с вкладками, то, вероятно, имеет смысл предусмотреть для каждой из подстраниц собственный конечный автомат.

Если логика отображения, сокрытия и перемещения элементов управления будет растворена в логике приложения, то результирующая схема управления окажется весьма запутанной.

Существует настоятельная необходимость в главном арбитре, который бы управлял бы распределением пространства экрана, и конечный автомат великолепно подходит для этих целей.

Расположение всего кода, ответственного за параметры видимости, размеров и размещения, в одном месте позволяет значительно сократить сроки завершения этой части итеративного проектирования, и сделать ее более понятной по сравнению с вариантом, в котором упомянутый код располагается вразброс в различных местах кода, реализующего логику приложения.

Если код пользовательского интерфейса тесно переплетен с логикой приложения, то выполнить это будет очень трудно – внесение изменений в пользовательский интерфейс потребует кропотливого изучения всей логики приложения. Поэтому отыскать ошибки в коде интерфейса, насыщенного различного рода взаимосвязями, и устранить их, не нарушая работоспособности приложения, очень трудно, поскольку они разбросаны по всему приложению, а не сконцентрированы в одном месте и надежно инкапсулированы.

Поэтому логику приложения необходимо отделять от пользовательского интерфейса.

Конечный автомат подходит для управления всеми нуждами пользовательского интерфейса мобильного приложения. При этом необходимо следить за эффективным использованием экранного пространства в процессе того, как пользователь переводит приложение из одного состояния в другое. При наличии конечного автомата, который показывает, скрывает и перемещает элементы управления по экрану в соответствии с необходимостью, эту задачу можно решить весьма эффективно.

Абстрагирование всех режимов работы пользовательского интерфейса в одном автомате обеспечивает максимальную гибкость процесса внесения изменений в модель экранного дисплея, избавляя от необходимости просматривать и изменять множество кода, распределенного между различными функциями и обработчиками событий пользовательского интерфейса.

3.2. Конечные автоматы и управление памятью

Как и в случае пользовательских интерфейсов, существует два способа управления данными – явный и неявный. Для мобильных устройств крайне желательно использовать подход,

предполагающий явное управление использованием памяти. Если приходиться иметь дело с несколькими различными типами данных или системных ресурсов, то конечный автомат целесообразно построить для каждого из них.

3.3. Конечные автоматы и фоновые задачи

В мобильных приложениях часто встречаются участки кода, которые либо долго выполняются, либо требуют сетевого доступа. Операции подобного рода рекомендуется выполнять в асинхронном по отношению к потоку выполнения пользовательского интерфейса режиме. При реализации любой асинхронной операции конечный автомат, с помощью которого указанный поток может связываться с фоновым потоком и получать информацию о ходе его выполнения, играет важную роль, так как модель управления потоком на основе конечного автомата позволяет выполнить все требования.

Проектирование фоновых задач в виде классов, использующих конечные автоматы, упрощают получение соответствующих данных от потока пользовательского интерфейса.

Конечные автоматы значительно расширяют возможности управления выполнением фоновых задач. Их использование делает возможным предоставление фоновыми потоками информации о состоянии выполнения, а также обращение других потоков с запросами к фоновому потоку на выполнение определенных действий, например, с запросом на прекращение выполнения фоновой работы.

Приложения лучше всего проектировать на основе однопоточной модели, привлекая фоновую обработку лишь тогда, когда она необходима.

3.4. Конечные автоматы и игры

Конечные автоматы весьма удобно применять в играх (посмотрите игру «Космонавт» (http://is.ifmo.ru/projects/cosmo/, А.Ш.), в которых персонажи, перемещающиеся в области игры, могут действовать в различных режимах (например, персонаж бежит вперед или назад, взбирается по лестнице, падает и т. д.).

Использование автоматов с четко определенными правилами для каждого персонажа, задающими варианты его поведения в различных состояниях и допустимые переходы из одного состояния в другое, позволяет создавать сложные варианты поведения как персонажей, управляемых компьютером, так и персонажей, управляемых пользователем.

Выводы

Идея применения конечных автоматов является чрезвычайно полезной концепцией, плодотворность которой прошла проверку временем (но знает ли об этом народ, А.Ш.). Использование конечных автоматов позволяет разработчикам создавать хорошо организованные приложения с гибкими возможностями. Их применение позволяет создавать ясный, понятный и надежно функционирующий код.

Старайтесь избегать неявного управления состояниями приложения и не жалейте времени на разработку явно сформированной модели состояний.

Использование конечных автоматов стало уже обычной практикой (хотелось бы так думать, но я очень сомневаюсь, А.Ш.) при проектировании приложений для настольных компьютеров, серверов и мобильных устройств.

Преимущества, обеспечиваемые применением конечных автоматов, заметнее всего проявляются в случае приложений для мобильных устройств, требующих экономного расходования экранного пространства, памяти, вычислительной мощности и других ресурсов.

2006. https://is.ifmo.ru/automata/mobdev/

Автоматное программирование, водка и буква Ё

Как лодку назовешь,
так она и поплывет

Некоторое время назад при обсуждении содержания страницы «Автоматное программирование» в Интернет-энциклопедии *Wikipedia* меня пытались убедить, что автоматное программирование – это программирование с автоматами, которое давно и хорошо известно. Если придерживаться такой логики, то программирование с использованием интегралов – это интегральное программирование, дифференциалов – это дифференциальное программирование, а программирование с линейками – это линейное программирование и т. д. :-).

В моей трактовке автоматное программирование – это технология, которая охватывает спецификацию, проектирование, реализацию, верификацию и документирование программ с явно выделенными состояниями. При этом программы в целом предлагается строить как **системы автоматизированных объектов управления, каждый из которых состоит из объектов управления и системы управления, образованных системой взаимодействующих автоматов**.

Я думаю, что если посмотреть на список литературы в статье [1], то ситуация с автоматным программированием не столь очевидна, как кажется человеку, который критикует меня в указанном обсуждении. Авторы этой статьи со мной никак не были связаны, учились, в том числе и в МГУ, обладают учеными степенями в области физико-математических наук и, несомненно, знали до встречи со мной, что в программировании автоматы могут применяться, однако... Кстати, приведу цитату из переписки моего критика и его знакомого: «Довод Шалыто о том, что именно он оформил программирование от состояний в некую стройную систему, которую он назвал «автоматным программированием», и именно в этом его заслуга, вполне оправдан. Поэтому и говорят, что «автоматное программирование» предложил Шалыто. Это как с буквой Ё, мало кто знает, что её придумала **Екатерина Дашкова**, но все знают, что первым применять её начал **Николай Карамзин**. Поэтому зачастую говорят, что **буква Ё – буква Карамзина**» (http://alexott.livejournal.com/321730.html?thread=1717698#t1717698).

А еще есть история про исследования раствора ... из воды и спирта, которые проводил Менделеев, до открытия им Периодического закона. Представляете, как бы над ним некоторые сейчас смеялись, не открой он этот закон: «Водку на Руси давно пили, и зачем там что-то еще изучать, ха-ха-ха». А может быть, и тогда смеялись, но об этом просто ничего не известно. Зато теперь над ним никто не смеется, и то ладно. Победителей, ведь, не судят, правда?

Интересно, что мой критик признает, что нигде не нашел, чтобы термин «Автоматное программирование» применялся кем-либо до меня, но он человек молодой, и ему неважно кто предложил этот термин, так как, по его мнению, он совершенно естественен. Однако для некоторых людей существенно, кто первым предложил тот или иной термин. «**Определите значение слов, и Вы избавите человечество от половины его заблуждений**», – сказал **Декарт**.

Вот я и ввел во введении (http://is.ifmo.ru/books/switch_pdf/vvedenije.pdf) к книге [2] этот термин, который определил, как указано выше, так как определять его, как программирование с автоматами, в 1998 г. было бы просто глупо.

При этом отмечу, что рассматриваемый термин родился в результате обсуждения технологии с **Дмитрием Александровичем Поспеловым** (http://www.computer-museum.ru/galglory/pospelov.htm) на конференции по мультиагентным системам, проходившей в пригороде Санкт-Петербурга – в поселке Ольгино. **Вряд ли кто-то в то время в стране лучше его понимал, что такое автоматы и как их применять, но изложенная технология программирования его удивила, и он помог мне опубликовать статью [3] в журнале, в котором был заместителем главного редактора**. Вот что написано в аннотации к этой статье: «Описываемая технология может быть названа автоматной технологией, а соответствующая область программирования – автоматным программированием». Чуть позже эти термины были использованы в работах [4, 5], а в дальнейшем и в работе [6].

Это все по поводу термина, а саму технологию **я предложил в 1991 г.** [7]. Как это произошло, подробно (в форме пьесы) описано в разд. 19.2 (http://is.ifmo.ru/books/switch_pdf/_switch19.pdf) книги [1]. В качестве английских эквивалентов рассмотренного термина мною было предложено несколько терминов (например, ***Automaton Programming*** и ***Automata-Based Programming***). Окончательно же я остановился на термине ***Automata-Based Programming*** [8-11].

В заключение приведу слова Дмитрия Менделеева о науке: «Наука есть достояние общее, а потому **справедливость требует не тому отдать наибольшую научную славу, кто первым высказал**

истину, а тому, кто сумел убедить в ней других, показал ее достоверность и сделал ее применимой в науке» (http://is.ifmo.ru/foundation/mend_sc/).

Литература

1. ***Кузьмин Е.В., Соколов В.А.*** Моделирование, спецификация и верификация «автоматного» программирования // Программирование. 2008, № 1, с. 38-60. http://is.ifmo.ru/download/2008-03-12_verification.pdf.

2. ***Шалыто А.А.*** *Switch*-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998, 628 с. http://is.ifmo.ru/books/switch/1.

3. ***Шалыто А.А.*** Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления / /Известия РАН. Теория и системы управления. 2000. № 6, с. 63-81. http://is.ifmo.ru/works/app-aplu/1/.

4. ***Шалыто А.А.*** Алгоритмизация и программирование для систем логического управления и «реактивных» систем // Автоматика и телемеханика. 2001. № 1, с. 3-39. http://is.ifmo.ru/works/arew/1/.

5. ***Шалыто А.А., Туккель Н.И.*** SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001, № 5, с. 45-62. http://is.ifmo.ru/works/switch/1/.

6. ***Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А***. Инструментальное средство для поддержки автоматного программирования // Программирование. 2007, No 6, с. 65-80. http://is.ifmo.ru/works/_2008_01_27_gurov.pdf

7. ***Шалыто А.А.*** Программная реализация управляющих автоматов //Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып.13, с. 41, 42. http://is.ifmo.ru/works/switch_prr/.

Термин *Automaton Programming*, а затем термин *Automata-Based Programming*, видимо, первым предложил я. Перечисляю работы на английском языке, в которых эти термины используются.

8. ***Shalyto A.A.*** Software Automation Design: Algoritmization and Programming of Problems of Logical Control // Journal of Computer and Systems Sciences International. 2000. № 6, pp. 899-916. http://is.ifmo.ru/articles_en/2000/shalyto-switch-2000.pdf.

9. ***Shalyto A.A.*** Logic Control and «Reactive» Systems: Algorithmization and Programming // Automation and Remote Control. 2001. № 1, pp. 1-29. http://is.ifmo.ru/articles_en/_logic_control_and_reactive_systems.pdf.

10. ***Shalyto A.A., Tukkel N.I.*** SWITCH-technology – An Automated Approach to Developing Software for Reactive Systems // Programming and Computer Software. 2001, № 5. https://dl.acm.org/citation.cfm?id=597470.

11. ***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 Boston Section. 2003, pp. 65-70. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1245023, http://is.ifmo.ru/english/_aut_th.pdf.

2008. http://is.ifmo.ru/download/2008-03-17_automata.pdf

Тяжелый коврик и автоматное программирование

Я в разных аудиториях рассказывал, что понимаю под автоматным программированием и его парадигмой. При этом несмотря на то, что я говорю весьма громко, всегда находится те, кто меня не слышит :-).

Обычно я говорю, что автоматный подход применим для **надежного** построения управляющих программ, например, таких, что используются во встроенных системах – в лифте, принтере, фотоаппарате, турникете и т. п. При этом также говорю, что в книге (**Ослэндер Д., Риджли Д., Ринггенберг Д. Управляющие программы для механических систем. Объектно-ориентированное проектирование систем реального времени. М.: БИНОМ. Лаборатория знаний. 2004**) отмечается, что **99% всех микроконтроллеров и микропроцессоров мира используется в системах этого класса**. Подтверждение важности этого класса программ я получил недавно в представительстве корпорации *IBM* в Цюрихе, когда первое, что они с гордостью показали, были не сложные компьютеры или наносистемы, которыми они занимаются, а кредитные карты с чипом, так как они выпускаются огромными тиражами. Тут и до *Java-card* и автоматного программирования недалеко.

Я обычно рассказываю, что цель моей деятельности состоит в том, чтобы, например, в автомобиле все надежно открывалось и закрывалось, и он корректно разгонялся и тормозил. Чтобы не происходило того, что произошло несколько лет назад в центре Парижа, когда посол Филиппин в *Юнеско* из ошибок в программном обеспечении чуть было, не задохнулся в новом бронированном автомобиле.

Недавно, рассказав эту историю, я вдруг услышал, что для управления автомобилем автоматный подход, может быть, и подходит, а вот **для вычислительной геометрии не подходит совсем!** На это приходится отвечать, что это, возможно, и так, но …, и дальше все заново, как в сказке про белого бычка.

А еще слушатели любят, например, спрашивать, как применять автоматный подход в параллельных асинхронных системах, содержащих 10 со многими нулями состояний. Когда при этом я в ответ уточняю, что они понимают под «состоянием», народ «обычно безмолвствует».

После этого я обычно говорю, что цель предлагаемого подхода состоит в создании надежных (за счет кодогенерации и повышения уровня автоматизации верификации на основе метода *Model Checking*) программ, содержащих от единиц до сотен **управляющих** состояний, то в ответ часто слышу, что это слишком просто и неинтересно. Это особенно любят говорить программисты, которые никогда не разрабатывали ответственных систем и которые по образованию обычно являются математиками, многих из которых хоть «хлебом не корми», но дай порассуждать о сложных задачах.

А то, что люди часто погибают из-за «простых» ошибок в системах, их мало интересует.

В подтверждение, расскажу историю про **тяжелый коврик**!

«29.09.2009 г. Американское национальное управление по безопасности дорожного движения обратилось к 3,8 млн владельцев автомобилей *Toyota* и *Lexus* последних моделей с призывом немедленно избавиться от **резинового коврика** на полу со стороны водителя, который может способствовать залипанию педали газа и неконтролируемому ускорению автомобиля.

Вскоре в Интернете появился ролик с записью телефонного звонка в службу 911. Звонивший сообщил, что у него отказали тормоза, и он не может остановить разогнавшийся до 190 км/ч *Lexus*. В результате погиб звонивший и три его пассажира. По итогам расследования катастрофы ее причиной был назван резиновый **коврик**!

В ноябре 2009 г. корпорация отозвала еще миллионы автомобилей, так как помимо ковриков решила внести изменения в механизм педали газа и конструкцию пола, а также установить систему приоритета тормоза над газом. В январе 2010 стало известно, что с 1999 г. было зафиксировано 2274 случая внезапных ускорений автомобилей *Toyota*, которые привели к 75 авариям и 18 смертельным исходам. Тем временем сервисная кампания вышла за пределы Америки, и, в частности, было предложено отозвать 1,7 млн автомобилей в Европе. Общее число затронутых отзывами автомобилей **превысило 8.5 млн штук**, что соответствует годовому объему производства корпорации докризисной поры» (Газета «Ведомости». 22.03.2010. http://www.vedomosti.ru/newspaper/article/2010/03/22/228673).

Выслушав историю о том, как «простые» ошибки могут приводить к огромным затратам и катастрофам с человеческими жертвами, всегда находится слушатель, который, как глухой, продолжает бубнить: «А для решения такого-то класса задач автоматных подход неэффективен».

«Ну, что делать!», – обычно отвечаю я. На этом дискуссия обычно завершается, а недовольные автоматным программированием остаются, не понимая, что, пропустив даже одно состояние, последствия могут быть хуже, чем от коврика.

И на самом деле, в автомобильной промышленности пошли проблемы не только ковриками, но, и с системами управления, что связано с резким повышением уровня автоматизации автомобилей. Вот что пишет по этому поводу газета «Известия» от 20.05.2010 г.: «Корпорация *Toyota* отозвала 11,5 тыс. дорогих седанов *Lexus* по вине системы управления, которая призвана контролировать угол поворота руля при прохождении поворота. Иногда система не до конца «выходит из поворота» – даже при выправленном руле автомобиль продолжает поворачивать, что чревато авариями. После этого *Toyota* отозвала еще 34 тыс. внедорожников из-за неправильной работы системы курсовой устойчивости, приводящей к заносу автомобиля».

Я, естественно, не знаю в чем там проблема, но не исключаю, что эти системы не так программировались.

Каждый раз, когда начинается «бодяга» с обсуждением применимости автоматного программирования, так и хочется спросить: «А Вы когда-нибудь программировали ответственные системы?»

Многие умники сегодня слышали, что в реактивные системах следует использовать автоматы, и поэтому считают, что применение автоматного подхода естественно в системах этого класса, а я знаю многотысячные предприятия, в которых такой подход используют только те специалисты, которых я убедил в этом, а остальные продолжают программировать «как учили». У последних, как говорится, все хорошо, но только им, почему-то, очень не нравится, когда их спрашивают: «А как все-таки верифицировать ваши программы?».
2008. http://is.ifmo.ru/belletristic/kovrik

Парадигма автоматного программирования

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 г. «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 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** и **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 г.

Ниже излагаются основы автоматного программирования, разрабатываемого автором с **1991 г.** [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 **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 [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 [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 этой работе 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 г. 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.

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

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 идея 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.

Из изложенного следует, что **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 [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 не упорядочена 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. 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 неочевидна, так как даже при программировании логических контроллеров [35], практически никто не предлагал сначала проектировать автоматы, и только затем их реализовать на выбранном языке программирования. В работе [36] было показано, что плохого в неавтоматном программировании контроллеров, а в [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 [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: **1**. В качестве входных воздействий, наряду с входными переменными, используются события. **2.** Запуск программ осуществляется по событиям, а не циклически. **3**. В качестве выходных воздействий могут использоваться не только двоичные, но и другие функции, что позволяет называть автоматы, применяемые при этом, гибридными [44]. **4.** Автоматы могут содержать не только вложенные состояния [17], но и вложенные автоматы. **5.** Автоматы могут взаимодействовать не только за счет проверки номеров состояний, как было предложено для систем логического управления в [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 называется 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 «**программирование 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 [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 [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 [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 использованию 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 ООП 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 большие трудности [61]. Решению этого вопроса посвящена, например, работа [62]. Для устранения недостатков, присущих указанному паттерну, в [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 [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 разработана библиотека, реализованная на языке *C++*. При ее использовании 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 [71, 72]. Работы в указанном направлении продолжаются. При этом, например, в [73] предложена удобная графическая нотация

для отображения наследования в автоматных программах, а в [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* 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 [33, 75-78]. 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 годы».

Отдельно хочу сослаться на мою работу с **Владимиром Ульянцевым** «**О сложности верификации простых программ со сложным поведением**» (http://is.ifmo.ru/works/2013/ulyantsev-shalyto-verification.pdf), в которой показано, что даже очень простые автоматные программы бывает трудно верифицировать, хотя автоматные программы наиболее приспособлены для верификации методом *Model Checking*.

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 [79] 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 активно ведутся работы [80-87]. 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 указанных работах на основе генетического программирования выращивались автоматы, а в [88] одновременно использовались различные модели и методы искусственного интеллекта. В этой работе, 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 [89]. При этом отметим, что поведение системы, построенной человеком, однако, значительно более понятно, и в нее 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, включая их верификацию. Эта технология описана в [90-92], а в [93] – она изложена для массового читателя.

9. Инструментальные средства для поддержки автоматного программирования. Рассмотрим средства для поддержки автоматного программирования. Для процедурных автоматных программ в

[94] было разработано инструментальное средство, которое по графам переходов, представленным в нотации, предложенной в [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, например, 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 [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 ***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 [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 гг. [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**, **а 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 ответственными объектами, 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 [109, 110] рассматривалось применение автоматов при построении мультиагентных систем, а в [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 (в том числе вложенные), моделировать работу их в разных режимах и выполнять кодогенерацию на языке *C и не только*. В 2007 г. 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 описываемого в настоящей работе 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).

Применение автоматного подхода позволяет уменьшить число «прозрений», аналогичных произошедшим в 2007 г., когда ведущий специалист корпорации *IBM* (!) вдруг обнаружил, что автоматы целесообразно использовать и при разработке виджетов [18, 19]. **Автоматный подход, описанный в настоящей работе, поддержан парадигмой, технологией и инструментальными средствами**.

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]. 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.

Отмечу также, что применение предлагаемого подхода 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.

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. Если же 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. Тем самым 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]. С 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 настоящей работе, 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 г., использовались во встраиваемых системах, а в 2000 г. число микроконтроллеров в высококачественном автомобиле достигало 60**» [132].

Рассматриваемая технология 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].

Я предполагаю, 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 [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 [144, 145].

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 **новой парадигме управления**, названной автором «**автоматное управление**». Эта парадигма, как отмечалось выше, неоднократно апробировалась на практике, в том числе и при проектировании программного обеспечения сложных систем [45, 128, 129].

Интересно отметить, что в книге [146] есть глава, которая называется «Наш друг конечный автомат», которая, по моему мнению, является гимном применению автоматов в мобильных устройствах.

Новое направление в автоматном программировании – программирование ПЛИС (программируемых логических интегральных схем) [147]. Благодаря этому удается обеспечить возможность кодизайна, при котором поведение аппаратной программной частей устройства может описываться одинаково – с помощь **графов переходов**.

Отметим также, что использование автоматов при проектировании систем управления [148] до последнего времени в основном рассматривалось в рамках применения гибридных автоматов [149]. Дополнительный интерес к использованию автоматов в управлении возник у специалистов после пленарного доклада Р. Брокетта на конгрессе *IFAC* [150], в котором обсуждались вопросы упрощения проектирования сложных систем управления.

Автор признателен **Дмитрию Александровичу Поспелову**, в ходе беседы с которым в 1998 г. 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 [128], а парадигма автоматного программирования – в работе [129].

Литература

1. ***Дейкстра Э.*** Смиренный программист // Лекции лауреатов премии Тьюринга за первые двадцать лет 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 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, 2002.

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. https://is.ifmo.ru/download/turing.pdf.
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.С.*** 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 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. http://is.ifmo.ru/download/evsys.pdf.
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.*** Унифицированный 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. институт искусственного 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». 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 Technolo-gies 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. Учебно-методическое пособие. 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: НИИМВС 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: НИИМВС 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.*** Построение автоматов с помощью генетических алгоритмов для решения задачи о флибах / Сборник X международной конференции по мягким вычислениям и измерениям. СПбГУЭТУ «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. СПбГУЭТУ «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: НИИМВС 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 Applica-tion (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. https://www.kgeorgiy.info/papers/Korneev_GA__Shalyto_AA_--_State-Driven_Programming__slides.pdf.
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 // МирПК. 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., Noarvsky 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 и мотивации к творчеству**. 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. https://www.osp.ru/os/2004/03/184060/.
126. Разработка технологии верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода. СПбГУ ИТМО. 2007. 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., Рингенберг 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.
135. ***Кузнецов Б.П.*** Психология автоматного программирования // *BYTE*/Россия. 2000. № 11. http://www.softcraft.ru/design/ap/ap01.shtml1.
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.
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.

138. ***Лагунов И.А.*** Разработка текстового языка автоматного программирования и его реализация для инструментального средства 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 и императивных языков программирования / Материалы конференции «*Software Engineering Conference (Russia) – SEC(R)* 2007». М.: ТЕКАМА. 2007.
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.
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: ЦНИИРТК, 2008.
143. ***Клебан В.О., Новиков Ф.А.*** Применение конечных автоматов в документообороте // Научно-технический вестник СПбГУ ИТМО. № 8 (53). Автоматное программирование. 2008.
144. ***Shalyto A.A.*** Technology of Automata-Based Programming. 2004. http://www.codeproject.com/KB/architecture/abp.aspx?print=true.
145. ***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.
146. ***Салмре И.*** Программирование мобильных устройств на платформе .Net Compact Framework. М.: Вильямс. 2006. Кооперайт на английском языке – Pearson Education, Inc., 2005. http://is.ifmo.ru/automata/mobdev/.
147. ***Янкин Ю.Ю., Шалыто А.А.*** **Автоматное программирование ПЛИС** в задачах управления электроприводом // Информационно-управляющие системы. 2011. № 1. http://is.ifmo.ru/works/_automata_plis.pdf.
148. ***Гудвин Г.К., Гребе С.Ф., Сальгадо М.Э.*** Проектирование систем управления. М.: Бином, 2004.
149. ***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.
150. ***Brokett R.*** Reduced Complexity control systems / *Proceedings of the 17 th World Congress the International Federation of Automatic Control. Seoul*. 2008.
2008. http://is.ifmo.ru/works/_paradigma_automata.pdf, https://vk.com/@1077823-paradigma-avtomatnogo-programmirovaniya
Статья опубликована в «**Научно-техническом вестнике Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2008. № 8 (53). Автоматное программирование, с. 3-23**, https://ntv.ifmo.ru/ru/article/1288/paradigma_avtomatnogo_programmirovaniya.htm

Лучше, чем на телевизор

22.11.2010 г. я и Федор Царев были в *ETH* (Цюрих) на конференции, посвященной шестидесятилетию **Бертрана Мейера**, который в то время начинал работать у нас в Университете по совместительству. На фотографиях, размещенных по адресу: https://vk.com/id1077823?z=album1077823_122563144, изображены: **Никлаус Вирт, Эрих Гамма, Джозеф Сифакис, Давид Парнас, Юрий Гуревич**, **Андрей Терехов, Надя Поликарпова, Федя Царев** и другие участники конференции.

Отмечу, что в свое время я попал в очень хорошую компанию – в «**Bertrand Meyer's gallery of computer scientists**» (http://se.inf.ethz.ch/old/people/meyer/gallery/), в которой есть все упомянутые выше ученые, но потом «Бертранова любовь» ко мне вместе с моим портретом исчезла, как впрочем, и любовь к ИТМО. Многих из этих ученых можно найти и в галерее *Best Computer Science of All Time* (https://rankly.com/list/best-computer-scientist-off-all-time).

Там произошла интересная история, начало которой я описал в коротком тексте еще в 2003 г. и назвал «**Лучше, чем документация на телевизор**» (http://is.ifmo.ru/reflections/mystories/): «Один мой студент, увидев документацию на проект создания программы, выполненный по *Switch-*

технологии, сказал задумчиво: «**Это лучше, чем документация на телевизор. Это, видимо, как документация на системы управления подводной лодкой**».

Вот ее продолжение. Когда на конференции в Цюрихе, я показал одному из докладчиков – ***David Parnas*** (https://en.wikipedia.org/wiki/David_Parnas, https://vk.com/id1077823?z=photo1077823_457246347%2Fwall1077823_16528) – классно оформленную **проектную документацию** (http://is.ifmo.ru/projects/dg/) на программу (**ошибочно названную мною «программной документацией»**, http://is.ifmo.ru/download/short_dg.pdf), он незамедлительно определил, что я связан с военно-промышленным комплексом, так как, по его мнению, в иных местах документация так хорошо не оформляется.

Мое отрицание факта, связанного с работой, он всерьез не воспринял, так как сам был оттуда и этого не скрывал. Да и как я мог «не попасться», если даже в указанной выше статье в *Wikipedia* о нем, есть такие слова: «He is also noted for his advocacy of precise documentation» – он пропагандировал точную (четкую, аккуратную) документацию.

Еще одна история, произошедшая там, состояла в том, что я хотел рассказать **Джозефу Сифакису** (одному из создателей метода верификации ***Model checking***, за который они получили **премию Тьюринга**) о том, что их метод **классно работает на автоматных программах (*Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.*** Верификация автоматных программ. СПб.: Наука. 2011. 242 с., http://is.ifmo.ru/verification/velder_verification_posobie_nauka.pdf). Однако до его доклада я не смог его узнать – так сильно он изменился по сравнению с опубликованными ранее фотографиями, а после доклада – он сразу же исчез, так как, видимо, обиделся ... на юбиляра, который тоже его не узнал. Так Сифакис (https://vk.com/id1077823?z=photo1077823_457246346%2Fwall1077823_16528) остался в неведении об удобстве верификации их методом именно автоматных программ...

А еще мы там пили пиво в компании с Андреем Тереховым и с выдающимся ученым по *Computer Science* – Юрием Гуревичем (https://vk.com/id1077823?z=photo1077823_457246348%2Fwall1077823_16528).
19.06.2011. https://vk.com/@1077823-luchshe-chem-na-televizor

Конечный автомат многим не враг

Шестого мая 2020 г. мой ученик **Виталий Клебан** (соавтор одного из стандартов на «Интернет вещей» – на спецификацию *LoRaWAN*: https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf, сообщил мне, что не один я являюсь фанатом применения автоматов в программировании. При этом он написал, что 25, 26 мая 2020 г. «Российские интернет-технологии» проводят профессиональный онлайн-фестиваль для тех, кто создает Интернет (https://ritfest.ru/2020/abstracts/6425).

В программу в разделе «Новинки (!) и хайпы» включен доклад **Романа Омельницкого** из «Яндекса» на тему: «Стэйт-менеджмент на конечных автоматах» (https://ritfest.ru/2020/authors/13013). В тезисах доклада автор пишет: «Когда приложение растет и интерфейсы усложняются, классический подход к стейт-менеджменту показывает себя не так хорошо. В докладе я расскажу, что такое конечные автоматы и стейтчарты, и как они могут помочь нам писать более предсказуемую и прозрачную логику. Покажу, как их применять и какие готовые решения существуют».

Сначала я не понял, что такое «стэйт-менеджмент». Оказалось все очень просто: «стэйт» – это русская транскрипция английского слова «state» – «состояние», а потом очень удивился, что доклад попал в раздел конференции «Новинки», так как если в названии чего-то есть понятие «состояние» и число состояний не громадно (управляющих состояний, в отличие от вычислительных, много практически никогда не бывает), то что же еще использовать, как не «конечные автоматы»?

После того, как я посмотрел несколько лекций 2019 г. на *YouTube*, то оказалось, что, думая так, был неправ. Первое, что меня удивило – в их названиях понятие «состояние» используется в единственном числе, ну а далее удивление не покидало меня, так как авторы шести (!) лекций с названием «Управление состоянием» (https://www.youtube.com/watch?v=lwec8maPrrI) и лекции «Проблемы стэйт-менеджмента, и их решение с *Effector.js* « продолжительностью в один час шесть минут (https://www.youtube.com/watch?v=48XSmEIqbkI) ни разу (!) не упомянули про конечные автоматы! При этом они не догадывались, что, видимо, многие из указанных в лекциях проблем возникают именно потому, что они не используют автоматы, и хорошо, что Омельницкий, наконец-то, это понял и расскажет «народу», как их решать.

Я пишу о применении конечных автоматов в программировании уже почти тридцать лет – с 1991 г. (http://is.ifmo.ru/), и каждый раз, как мальчишка, удивляюсь, когда узнаю, что кто-то в очевидной, как в данном случае, ситуации для применения автоматов, не использует их.

Первый раз шок по этому поводу я испытал через пятнадцать (!) лет с начала «моей писанины» на эту тему, когда в 2007 г. увидел на сайте корпорации *IBM* (!), что их сотрудник **Эдвард Принг** «разразился» серией из трех статей на тему «Конечные автоматы на *Java Script*». Первая из них называлась «Разрабатываем виджет» (https://www.ibm.com/developerworks/ru/library/wa-finitemach1/index.html), вторая – «Реализация виджета» (https://www.ibm.com/developerworks/ru/library/wa-finitemach2/wa-finitemac_ru.html) и, наконец, третья – «Тестируем виджет (исходники)» (http://www.interface.ru/home.asp?artId=7867).

Интересно, что при поиске в *Google* непосредственно после второй статьи Принга выдавалась информация о моем большом тексте 2008 г. «**Автоматное программирование**», про который было написано «**читать бесплатно онлайн**». Кстати, этот текст (http://is.ifmo.ru/works/_2010_09_08_automata_progr.pdf) вошел в число победителей второго этапа Всероссийского конкурсного отбора обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы», введенному в «Перечень приоритетных направлений развития науки, технологий и техники в РФ», указом Президента РФ от 21.05.2006 г. № 843.

История с Прингом произошла уже после выхода книги ***Салмре И.*** **Программирование мобильных устройств на платформе *.Net Compact Framework***. М.: Вильямс. **2006** (На английском языке – в **2005** г.), в которой пятая глава называется «**Наш друг конечный автомат**» (http://is.ifmo.ru/automata/mobdev/), и, по моему мнению, **является гимном применения автоматов в мобильных устройствах**. Однако, оказывается, гимн гимну рознь: если Гимн СССР исполняли по радио каждое утро, и поэтому его узнавали все граждане нашей страны и не только ее, то указанный «гимн», как оказалось, не стал узнаваемым даже в области мобильных устройств, не говоря уже, о создании программного обеспечения (ПО) в других предметных областях.

Как мы видим, не только 2007 г., но и 2005 г. не стал переломным в рассматриваемом вопросе, но история этого вопроса идет еще глубже. Так сотрудники кафедры мехатроники Калифорнийского университета в Беркли **Ослэндер Д.М., Риджли Д.Р., Рингенберг Д.Д.** в книге «Управляющие программы для механических систем. Объектно-ориентированное проектирование систем реального времени. М.: БИНОМ. Лаборатория знаний. 2004) написали: «Для систем реального времени в 1997 г. 80% всех проектов не были вовремя внедрены из-за плохого качества ПО». Они считают, что для повышения качества ПО необходимо «**принять такую техническую политику, при которой *не только написание кода (программирование) рассматривается как творческая деятельность!***» (формулировка, сильно смягченная мною, А. Ш.).

Это может быть обеспечено, если специалисты в предметной области будут детально и формально описывать, **используя графы переходов конечных автоматов**, как управляющая часть ПО должна себя вести, а программисты – разрабатывать различные инструменты (включая, например, шаблоны) для преобразования таких описаний в программы на заданных языках программирования. Это, правда, делает собственно программирование значительно более рутинным, так как творческая часть в значительной мере переносится на этап проектирования и отдается на откуп другим людям.

При этом, однако, сам процесс создания ПО становится существенно более **упорядоченным**, и резко уменьшается число ошибок в нем. При таком подходе описание поведения программ выполняется достаточно просто и становится понятным широкому кругу заинтересованных лиц, а не только программистам, что также способствует повышению качества ПО, так как, чем больше людей будет понимать, что программисты собираются делать. то тем раньше их можно будет направить на «путь истины». Кстати, в автоматном программировании, которое было предложено значительно раньше, чем была написана эта книга (на английском – в 2002 г.), я высказывал эти мысли, «кощунственные» для традиционного программирования.

В связи с изложенным, казалось бы, автоматы сам Бог велел применять при программировании ПЛК (программируемых логических контроллеров), однако в стандарте *IEC 1131-3*, который потом стал именоваться *IEC 61131-3*, в описании языков программирования автоматы даже не упоминались, а

стандартизованные методы программирования на этих языках отсутствовали. Поэтому мне пришлось разрабатывать такие методы, основанные на применении конечных автоматов, и описывать их в одной из глав книги «***SWITCH*-технология. Алгоритмизация и программирование задач логического управления**». СПб.: Наука, 1998. 628 с. (http://is.ifmo.ru/books/switch/1). Мои «походы» в представительства компаний *Schneider Electric* и *Siemens* в Санкт-Петербурге по этому вопросу ни к чему не привели – да и что можно взять с «продавцов», которые единственное что и хотели взять, то это деньги с продаж. При этом всем было наплевать на простоту и удобство программирования, особенно **программистам-пользователям** ПЛК, так как **чем программирование сложнее и непонятнее, тем выше их зарплата – и так везде в программировании.**

Через какое-то время при разработке международного стандарта на сети распределенных ПЛК *IEC 61499* его авторы, видимо, внезапно поумнели и рекомендовали при программировании ПЛК в таких сетях для описания поведения управляющих устройств использовать конечные автоматы. Это, в частности, сподвигло **Валерия Вяткина** – профессора университетов Аалто (Финляндия) и Лулео (Швеция), одного из авторов книги «Distributed Control Application: Guidelines, Design Patterns, and Application Examples with the *IEC 61499* (Industrial Information Technology)» (https://www.amazon.com/Distributed-Control-Applications-Application-Information/dp/1482259052), в 2014 г. позвонить мне и предложить сотрудничество в рамках применения конечных автоматов при программировании промышленных систем управления, я согласился и ... «понеслось».

В результате Валерий стал профессором-исследователем в Университете ИТМО и соруководителем нашей международной научной лаборатории «Компьютерные технологии». Фотографии (http://is.ifmo.ru/photo/2014-05-15-Vyatkin/) запечатлели Валерия момент появления у нас. Через некоторое время он переоделся, и в Японии уже был одет правильно – в майку Университета ИТМО :-) (http://is.ifmo.ru/photo/2016-04-Vyatkin/index.html).

Из наших совместных публикаций укажу две: ***Chivilikhin D., Ulyantsev V., Shalyto A., Vyatkin V.*** Function Block Finite-State Model Identification Using *SAT* and *CSP* Solvers // IEEE Transactions on Industrial Informatics. 2019. № 8, pp. 4558-4568. IF: 5.43, SJR: 1.6 (https://ieeexplore.ieee.org/document/8606098) и ***Mukhutdinov D., Filchenkov A., Vyatkin V., Shalyto A.*** Multi-Agent Deep Learning for Simultaneous Optimization for Time and Energy in Distributed Routing System // Future Generation Computer Systems. 2019. Vol. 94, May, pp. 587-600. IF: 4.639, SJR: 1.151 (https://www.sciencedirect.com/science/article/abs/pii/S0167739X18309087).

Из изложенного следует: что ни пиши, что ни рассказывай, будь то я, будь ведущий разработчик *IBM*, всегда находятся люди, которые как в той области, о которой пишешь и говоришь, так и в смежных областях, ничего об этом не знают или не хотят знать, и программируют управляющую часть ПО, как Бог им на душу пошлет, и только внедрение международных стандартов и инструментальных средств для их поддержки, как на примере с ПЛК для распределенных сетей, приводит к тому, что это не будет через несколько лет или десятилетий переоткрываться заново.

Хотя и это часто не помогает. Например давно известно, но, видимо, не всем, что существует расширение **всемирно известного** пакета *MatLab* – инструментальное средство *Stateflow* (http://www.mathworks.com/products/stateflow/). Оно позволяет строить автоматы (в том числе, вложенные), моделировать их работу в разных режимах и выполнять кодогенерацию на языке *C* и не только.

Так, в частности, на его основе можно не только создавать автоматные программы в их традиционном понимании, но программировать аппаратуру – программируемые логические интегральные схемы (ПЛИС). Об этом можно прочесть здесь: ***Янкин Ю.Ю., Шалыто А.А.*** **Автоматное программирование ПЛИС в задачах управления электроприводом** // Информационно-управляющие системы. 2011. № 1, с. 50-56 (http://is.ifmo.ru/works/_automata_plis.pdf).

Надеюсь, что Вы не думаете, что все, кто применяет ПЛИС, используют это замечательное средство, программируя с использованием **языка спецификации и программирования сверхвысокого уровня – графов переходов**, так как многих хлебом не корми – а дай покодить на *C*, а еще лучше – на ассемблере *VHDL*, и все это либо из-за безграмотности, либо из-за желания показать свою крутизну, либо для того, что кормиться не только хлебом, но и еще чем-то, так как отказавшись от применения автоматов в этих случаях, программист в некотором смысле становится незаменимым.

Я, конечно, понимаю, что ***Stateflow*** весьма дорогой пакет, но это не исключает того, что если Вы «крутой» программист, как думают о себе многие, то реализуйте аналог этого инструментального средства сами (как в свое время сделали мои ученики **В. Гуров и М. Мазин**, создав инструментальное средство ***UniMod*** – http://unimod.sourceforge.net/intro.html), или, по крайней мере, будете **проектировать** программы вручную, но не так, как Вам хочется, а идеологически так, как это сделано в указанных средствах!

При этом отмечу, что в 2012 г., когда мы практически перестали поддерживать ***UniMod***, стало известно, что он используется при бакалаврской подготовке в Италии. Предполагается его применение также, и при магистерской подготовке и при обучении аспирантов: ***Ricca F., Leotta M., Reggio G., Tiso A., Guerrini G., Torchiano M.*** Using *UniMod* for Maintenance Tasks: An Experimental Assessment in the Context of Model Driven Development (http://softeng.disi.unige.it/publications/2012-ricca-MiSE.pdf).

И в заключение сообщаю, что уже давно существует наша книга – ***Поликарпова Н., Шалыто А.*** **Автоматное программирование**. СПб.: Питер, 2008. 176 с., http://is.ifmo.ru/books/_book.pdf, которая, если у Вас найдется 39 рублей, «придет» к Вам на компьютер (https://www.piter.com/collection/all?q=автоматное+программирование).

У меня по этому вопросу пока все, но по рассматриваемой тематике совсем не всё – не надейтесь :-).
07.05.2020. http://is.ifmo.ru/belletristic/automata,
https://vk.com/@1077823-konechnyi-avtomat-mnogim-ne-vrag
P.S. 1. Оценка текста Валерием Вяткиным: «Браво», а Виталием Клебаном – «Класс!».
2. «Конечные автоматы – это даже не классика. Это почти античность. Проблема сия, увы, не только в ИТ. Так и хочется сказать: «Граждане-товарищи, изучайте античность! Серьёзно и глубоко. Там все гармонии идеальных пропорций давно найдены и доведены до совершенства» (Р. Богатырев).
3. Я ответил: «Спасибо, Руслан за классный совет! Теперь чуть-чуть из античности от меня. Всегда помните слова Аристотеля: «Известное известно немногим». Для тех, кто внезапно открывает для себя целесообразность применения автоматов в программировании после многолетней моей писанины об этом, я и написал этот текст».

Автоматное программирование

На эту тему я пишу с **1991 г.** (***Шалыто А.А.*** Программная реализация управляющих автоматов // Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып. 13, с. 41, 42, https://is.ifmo.ru/works/switch_prr/), а с конца **2002 г. существует «Сайт по автоматному программированию и мотивации к творчеству»** (http://is.ifmo.ru/), на котором, в частности, выложено большое число автоматно-спроектированных программ (http://is.ifmo.ru/projects/ и http://is.ifmo.ru/unimod-projects/).

В **1995 г.** я впервые выступил по этой теме на конференции за рубежом (***Shalyto A.A.*** Cognitive Properties of Hierarchical Representations of Complex Logical Structures / Proceedings of the 1995 International Symposium on Intelligent Control (ISIC). Workshop. 1995. Monterey. California, p. 391 (http://is.ifmo.ru/science/_cognitive_properties_of_hierarchical_representations_of_complex_logic_structures.pdf).

Здесь, в частности, я искал финансирование на издание написанной мною книги о программной реализации алгоритмов логического управления, которая базировалась на опыте работы по этой тематике в НПО «Аврора». Участвовавшие в работе конференции **Дмитрий Александрович Поспелов (**https://ru.wikipedia.org/wiki/Поспелов,_Дмитрий_Александрович**),** с которым я был знаком до этого, и **Вадим Николаевич Вагин (**https://naukarus.com/pozdravlenie-s-yubileem-k-70-letiyu-vadima-nikolaevicha-vagina**)** посоветовали мне подать заявку на **издательский грант** в Российский фонд фундаментальных исследований (РФФИ), что я и сделал. Кстати, среди афоризмов Поспелова наиболее известен такой: «**В науке первым часто оказывается не тот, кто сказал «А», а тот, кто сказал «Я»**. Это, в частности, относится и ко мне… К этому вопросу можно подойти и несколько иначе: ***Apple*** **не стала первым производителем *MP3*-плеера**. ***Она стала первой, кто сделал MP3*-плеер как надо**. Похоже, я так же поступил с автоматами в программировании.

В результате в **1995 г.** я выиграл большой грант РФФИ (проект № 96-01-14066) на издание книги объемом в 40 печатных листов и тиражом 1000 экземпляров, что по тем временам, да и сегодня, большая редкость. На его основе я заключил договор с издательством «Наука» на публикацию

книги с названием, которое указывал в заявке: «***Switch*-технология. Алгоритмизация и программирование задач логического управления**». Книга вышла из печати **в 1998 г.** (http://is.ifmo.ru/books/switch/1)**, ее выход совпал с моим пятидесятилетием**.

Вот очень короткая рецензия на эту книгу от **Кирилла Калишева:** «Я помню, когда еще был студентом в 90-х, работал в промышленной автоматизации, *real time control* и сложные состояния... ***Ваша книжка по Switch-технологии про то, что всю эту помойку нужно и можно генерировать из высокоуровневых описаний была откровением!*»**.

Таким образом, термин «***Switch*-технология» существует с 1995 г., а с 1998 г. он используется и не только мною** (https://ru.wikipedia.org/wiki/Switch-технология).

Термин «**автоматное программирование**», ***чтобы не говорили недоброжелатели о том, что он существовал «всегда»,*** родился в результате моей беседы с **Поспеловым** на конференции по мультиагентным системам, проходившей в **1997 г.** в поселке Ольгино под Санкт-Петербургом. Эта история описана в книге, посвященной 20-летию кафедры «Компьютерные технологии» Университета ИТМО (https://www.computer-museum.ru/books/shalyto_happy_years_new.pdf).

Приведу некоторые детали той встречи. После моего рассказа о том, как я предлагаю программировать, по крайней мере системы логического управления, **Дмитрий Александрович сказал**: «**Очень здоровый подход – крепко стоит на земле**. ***Назови его автоматным программированием*. Смотришь – привьется...**». Вот я и назвал!

Вряд ли кто-то в то время в стране лучше его понимал, что такое автоматы и как их применять, но изложенная технология программирования его удивила, и он помог мне опубликовать статью: ***Шалыто А.А.*** Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления // Известия РАН. Теория и системы управления. **2000**. № 6, с. 63-81, https://www.avrorasystems.com/ru/Data/Pressroom/Files/ran.pdf. В этом журнале Дмитрий Александрович в то время был заместителем главного редактора. Вот что написано в аннотации к этой статье: «**Описываемая технология может быть названа автоматной технологией, а соответствующая область программирования – автоматным программированием**». Этот термин на английский был переведен так: «**Automation Programming**» (http://is.ifmo.ru/articles_en/2000/shalyto-switch-2000.pdf).

При этом надо отметить, что после этого время от времени появлялись люди, которые в отличие от Поспелова, не удивлялись предложенной мной технологией, а с апломбом «поливали ее и меня», не зная того, что при этом «поливают» еще и Поспелова, отрицательных мнений о котором я в своей жизни не слышал. Но я-то помнил, что сказал Дмитрий Александрович, и поэтому, как тот караван, на которые лаяли собаки, шел и продолжает идти вперед…

Естественно, что и до моих работ в программировании использовались автоматы, **но ни парадигмы автоматного программирования** (http://is.ifmo.ru/works/2008/Vestnik/53/01-automata-based-programming.pdf), **ни такого термина, как «Автоматное программирование», ни на русском языке** (http://is.ifmo.ru/works/app-aplu/5), **ни на английском – Automata-Based Programming** (http://is.ifmo.ru/automata_en/_tech_aut_prog.pdf) **не было**. По второму термину доказательство приведено здесь: https://www.semanticscholar.org/topic/Automata-based-programming/2609355.

В **2023 г.** в аннотации на английском языке к статье ***Шалыто А.А.*** Валидация автоматных спецификаций // Научно-технический вестник информационных технологий, механики и оптики. 2023. № 2, с. 436-438 (https://ntv.ifmo.ru/file/article/21921.pdf) я использовал термин ***State Machine Program***, как перевод термина «автоматная программа». Возможен и такой перевод: **Automata-Based Program**.

Из рассмотрения этих работ следует, что термин «**Автоматное программирование**» был **предложен мною в статье «Автоматное проектирование программ**. Алгоритмизация и программирование задач логического управления» (https://www.avrorasystems.com/ru/Data/Pressroom/Files/ran.pdf) – **в 2000 г**., а термин «**Automata-Based Programming**» (http://is.ifmo.ru/science/automata_english) – **в 2003 г.** в названии статьи: **Technology of Automata-Based Programming** (https://is.ifmo.ru/automata_en/_tech_aut_prog.pdf).

В **2005 г.** этот термин был использован в нашей статье, опубликованной в США: ***Yartsev B., Korneev G., Shalyto A., Kotov V.*** **Automata-Based Programming** of the Reactive Multi-Agent Control Systems / 2005 International Conference on «Integration of Knowledge Intensive Multiagent Systems. KIMAS '05: Modeling, Exploration, and Engineering». USA, MA: IEEE, 2005, pp. 449-453. http://is.ifmo.ru/articles_en/_kimas05-2.pdf.

В последствии этот термин был использован в работе ***Micu A., Iftene A.*** **Communicative Automata-Based Programming. Society Framework** // Computer Science Journal of Moldova. Vol. 23. 2015. № 2(68), с. 189-208 (http://www.math.md/files/csjm/v23-n2/v23-n2-(pp189-208).pdf), в которой сказано: «In 2003 Russian scientist Anatoly Shalyto published an article about automata-based programming. This paper presents a **new way of programming** mechanisms for simulation of states, transitions and input/output operations».

В октябре 2023 г. в сети появился текст: **Automata-Based Programming** (https://www.devx.com/terms/automata-based-programming/), а в декабре: «**What Is Automata-Based Programming?**» (https://cellularnews.com/definitions/what-is-automata-based-programming/), в котором говорится: «Automata-Based Programming is a powerful technique that leverages automata theory to design efficient and reliable programs».

В **2006 г.** был предложен термин ***Automata-Based Design*** **(**https://web.archive.org/web/20090412002353/http://unimod.sourceforge.net/wiki/index.php/CSR2006_ABP_WORKSHOP), в **2007 г.** – термин «**Парадигма автоматного программирования**» (http://is.ifmo.ru/works/_2007_09_27_shalyto.pdf), а в **2008 г.** – термин «**Automata-Based Software**» (http://syrcose.ispras.ru/2008/files/11_talk.pdf).

А еще в **2009 г.** я предложил термин ***Automata-Based Control*** (***Shalyto A.A.*** Automata-Based Programming and Automata-Based Control. 2009. http://is.ifmo.ru/articles_en/_2009_10_07_automata_based_programming.pdf).

Интересно, что на русском языке термин «**Автоматное управление**» был предложен не мною, а в названии книги «**Автоматное управление асинхронными процессами в ЭВМ и дискретных системах**. Под редакцией В.И. Варшавского. М.: Наука, 1986. – 398 с.», однако в английском переводе этой книги была использована совсем другой термин: **Self-Timed Control of Concurrent Prosses**. Kluwer Academic Publishers. Editor: V.I. Varshavsky. 1990.

Уже много лет как в *Википедии* на английском языке есть статья «**Automata-based programming (Shalyto`s approah)**» (https://en.wikipedia.org/wiki/Automata-based_programming_(Shalyto%27s_approach)**)**.

Статья в этой энциклопедии под названием ***Automata-based programming*** исходно была написано мной, потом началась свара, которая закончилась тем, что статью полностью переписали, несмотря на то что в ней ссылки на работы, в названиях которых используется этот термин, кроме моих статей и статей и моих учеников, отсутствуют! На русском языке ситуация с переписыванием статьи аналогична. Статья о *Switch*-технологии существует в Википедии на русском языке бесконфликтно (https://ru.wikipedia.org/wiki/Switch-технология).

Моя последняя англоязычная статья на эту тему вышла **в 2017 г.**, она называется «**Why Design Programs: Anatoly Shalyto on Automata-Based Programming**» (http://news.ifmo.ru/en/science/it/news/6472/). Ее название на русском: «**Программа как инженерный проект, или зачем заказчику понимать структуру ПО изнутри**» (https://news.itmo.ru/ru/science/it/news/6472/). Этот текст начинается так: *«В газете «КоммерсантЪ» была опубликована статья «В «Росатоме» нашли проблемы с ядром. Сотрудники госкорпорации пожаловались на установленный на АЭС софт» (*https://www.kommersant.ru/doc/3196399*). В ней рассматривается вопрос о легитимности используемого на атомных электростанциях ПО. Там, в частности, написано, что имеющееся ПО не позволяет понимать, как программа будет себя вести в тех или иных ситуациях, как именно и какие в нее вносили изменения. Более того, в тексте статьи сказано, что на этот софт нет никакой документации. И это при том, что объект автоматизации – ядерный реактор (Таккер К. Как управлять ядерным реактором. М.: ДМК, 2022, 230 с.,* https://dmkpress.com/files/PDF/978-5-93700-132-0.pdf*), и почти никто, кроме, возможно, разработчика, которого, естественно, нет на объекте, а то уже и в живых, не понимает, как работает управляющая программа. И такой бардак с ПО творится почти везде в мире. Это нормально?».*

Вариант этой статьи на русском: «**Лекарство от болезни: автоматное программирование**» (https://habr.com/ru/company/spbifmo/blog/323122/).

Возможно, что применение автоматного программирования является «серебряной» пулей, о которой в 1975 г. ***Ф. Брукс****, говорил, что при создании ПО ее не существует, а через 25 лет – в 2010 г. (*https://nsu.ru/xmlui/bitstream/handle/nsu/8870/Frederick_Brooks.pdf*) с учетом работ* ***Д. Харела****, основанных на одной из разновидностей автоматного подхода, в ее отсутствии он уже был не так уверен. Вот базовая работа Харела: Harel D. Statechart: A Visual Formalism for Complex Systems // Science of Computer Programming. 1987. № 8, pp. 231-274.* www.inf.ed.ac.uk/teaching/courses/seoc/2005_2006/resources/statecharts.pdf.

Основным понятием автоматного программирования является понятие «состояние». Целесообразность применения автоматов состоит в том, что **их состояния декомпозируют все множество входных переменных на группы**, **выделяя с помощью каждого состояния только то подмножество входных переменных, которое определяет переходы из рассматриваемого состояния в соседние (смежные) состояния, в том числе и в самого себя**. При этом входные переменные, не входящие в группу, определенную некоторым состоянием, не влияют на переходы из этого состояния в другие состояния – переходы из рассматриваемого состояния несущественно зависят (не зависят) от этих переменных. **Это обеспечивает возможность реализации с помощью графов переходов задач большой размерности**. Такие задачи эффективно решаются также за счет того, что автоматы могут быть вложенными и вызываемыми.

Находясь в некотором состоянии, автомат с памятью превращается в соответствующий автомат без памяти (комбинационный автомат), который по значениям входных переменных, «выбранных» этим состоянием, осуществляет выбор одного из смежных состояний, в состав которых входит и рассматриваемое. Новое состояние «настраивает» автомат на реализацию в общем случае другого комбинационного автомата. Таким образом, автомат с памятью можно рассматривать в качестве многофункционального модуля, настраиваемого состояниями на реализацию в определенной последовательности различных ортогональных систем булевых формул, зависящих от различных групп входных переменных.

Еще о состояниях. А. Дж. Перлис в 1966 г. (***Перлис А. Дж.*** Синтез алгоритмических систем / Лекции лауреатов премии Тьюринга за первые двадцать лет. 1966-1985. М.: Мир, 1993) предложил в описания языка, среды и правил вычислений **включать состояния**, которые могут подвергаться мониторингу во время исполнения, позволяя диагностировать программы, не нарушая их целостности. В этом же году **Э. Дейкстра** (***Дейкстра Э.*** Взаимодействие последовательных процессов / **Языки программирования**. М.: Мир, 1972) предложил **ввести так называемые переменные состояния**, с помощью которых можно описывать состояния системы в любой момент времени. Он (как и я в автоматном программировании) **использовал для этих целей целочисленные (многозначные) переменные**. При этом им были поставлены вопросы о том, какие состояния должны вводиться, **как много значений должны иметь переменные состояния**, **и что эти значения должны означать**. **Он предложил сначала определять набор подходящих состояний (и я в автоматном программировании тоже), а лишь затем строить программу**.

По мнению Дейкстры, **диаграммы переходов между состояниями могут оказаться мощным средством для проверки программ**. Это обеспечивает поддержку его идеи о том, что ***программы должны быть с самого начала составлены правильно, а не отлаживаться до тех пор, пока они не станут правильными***. **Не появление ли автоматного программирования он предвещал?**

И еще одно высказывание про автоматы: «С тех пор, как разобрался с конечными автоматами, я уверен, что любой сложности задачу (в известных рамках) смогу реализовать быстро, правильно, а главное – с первого раза, и **мне не придется проводить бессонные ночи за отладчиком, тщетно пытаясь увеличить объем мозга для того, чтобы запомнить все**» (**А. Перро**).

Мне также близки слова **Б. Лисков** (Кавалли А. Сделать код понятным: как Барбара Лисков повлияла на современное программирование / Forbes Woman. 2022. https://www.forbes.ru/forbes-woman/473853-sdelat-kod-ponatnym-kak-barbara-liskov-povliala-na-sovremennoe-programmirovanie)**:** «**Моделируйте свои классы на основе поведения, а не свойств. Моделируйте свои данные на основе свойств, а не поведения**».

При этом всегда надо помнить: **«то, что не специфицировано формально, не может быть проверено, а то, что не может быть проверено, не может быть безошибочным**».

Наличие явно выделенных состояний **в автоматных программах** позволяет естественным образом (без дополнительных затрат) формировать протоколы, как для отладки, так и контроля их работы.

Приведу ряд «активностей» (в основном моих) в области автоматного программирования.

1. Введение в автоматное программирование

1.1. Зачем нужны автоматы? (http://is.ifmo.ru/download/airplane.pdf).

1.2. Программирование за ... 1 (одну) минуту (http://is.ifmo.ru/automata/1minute/).

1.3. Скромное обаяние автоматного программирования (http://is.ifmo.ru/belletristic/obayanie/).

1.4. Парадигма автоматного программирования (http://is.ifmo.ru/works/2008/Vestnik/53/01-automata-based-programming.pdf).

1.5. Применение автоматов при программировании мобильных устройств («Мой друг – конечный автомат»), (http://is.ifmo.ru/automata/mobdev/).

1.6. Лекция по автоматному программированию (https://www.youtube.com/watch?v=PPWTxceMutk). Вот один из комментариев к ней: «Написал пару скриптов по этой технологии на *Python*. Они просто железно работают». Потом появился еще один не менее интересный комментарий: «Вот уж действительно – гениальное просто. Пример с пятью и тридцати двумя состояниями при пяти двоичных флагах очень доступен для понимания. Спасибо, огромное. Теперь только автоматы :-)».

1.7. Презентации по автоматному программированию (https://is.ifmo.ru/download/shalyto_doklad_v_saratove.ppt, https://is.ifmo.ru/main/ap-intro.pdf, http://is.ifmo.ru/present/_1.ppt).

1.8. Что плохого в неавтоматном подходе к программированию контроллеров? (http://is.ifmo.ru/works/_Asu-2007-01.pdf).

1.9. Короткий материал «Зачем нужны автоматы?» (http://is.ifmo.ru/download/airplane.pdf).

1.10. Автоматное программирование ПЛИС (https://www.youtube.com/watch?v=YNWdmnwHZi8).

1.11. О верификации простых программ со сложным поведением (https://vk.com/@1077823-o-verifikacii-prostyh-programm-so-slozhnym-povedeniem).

1.12. Верификация автоматных программ (http://is.ifmo.ru/present/_verification_moscow.ppt).

1.13. Валидация автоматных спецификаций (https://ntv.ifmo.ru/file/article/21921.pdf).

Книги по автоматному программированию

2.1. ***Шалыто А.А. Антипов В.В.*** Алгоритмизация и программирование задач логического управления техническими средствами. СПб.: Моринтех, 1996. 90 с. Ее вариант опубликован в Интернете в 1998 г. (http://is.ifmo.ru/books/alg_log).

2.2. ***Шалыто А.А.*** *Switch*-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998. 628 с. (http://is.ifmo.ru/books/switch/1).

2.3. ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. Учебно-методическое пособие. СПбГУ ИТМО, 2007 (http://is.ifmo.ru/books/_umk.pdf).

Здесь впервые ***были сформулированы парадигма и миссия автоматного программирования***: «**Парадигма автоматного программирования состоит в представлении сущностей со сложным поведением в виде автоматизированных объектов управления**», а вот, как была определена его миссия: «**Нахождение компромисса между сложностью автомата и сложностью операций объекта управления, примирение тьюрингова программирования с традиционным – это и есть «миссия» автоматного подхода в мире разработки программного обеспечения**».

2.4. ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. Рукопись книги для издательства «Питер». 2008. (http://is.ifmo.ru/books/_book.pdf).

2.5. ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. СПб.: Питер. 2009. С издательскими неточностями (http://is.ifmo.ru/automata/shalytobook/).

2.6. ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. СПб.: Питер. Второе издание. 2010, 2011 (https://is.ifmo.ru/books/_book.pdf). В **2016** г. это **издание стало «вечным»**, так как книга стала цифровой (https://www.piter.com/product/avtomatnoe-programmirovanie).

3. Обзорные статьи по автоматному программированию

3.1. ***Шалыто А.А.*** Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления // Известия РАН. Теория и системы управления. 2000. № 6, с. 63-81. https://www.avrorasystems.com/ru/Data/Pressroom/Files/ran.pdf. (***Shalyto A.A.*** Software Automation Design: Algorithmization and Programming of Problems of Logical Control // Journal of Computer and

Systems Sciences International. 2000. Vol. 39. No. 6, pp. 899-916. http://is.ifmo.ru/articles_en/2000/shalyto-switch-2000.pdf).

3.2. ***Шалыто А.А.*** Алгоритмизация и программирование для систем логического управления и «реактивных» систем //Автоматика и телемеханика. 2001. № 1, с. 3-39. http://www.mathnet.ru/links/67df370047def9581c5d8713f122c865/at1715.pdf. (***Shalyto A.A.*** Logic Control and «Reactive» Systems: Algorithmization and Programming // Automation and Remote Control. 2001. Vol. 62. No. 1, pp. 1-29. http://is.ifmo.ru/articles_en/_log_control.pdf).

3.3. ***Туккель Н.И., Шалыто А.А.*** Реализация автоматов при программировании событийных систем // Программист. 2002. № 4, с. 74-80. http://is.ifmo.ru/download/evsys.pdf.

3.4. ***Шалыто А.А*** Технология автоматного программирования // Сборник научных статей «Современные технологии». СПбГУ ИТМО 2003, с. 18-26. https://www.elibrary.ru/item.asp?id=32370471.

3.5. ***Шалыто А.А.*** Технология автоматного программирования // Мир ПК. 2003. № 10, с.74-78. http://is.ifmo.ru/works/tech_aut_prog.

3.6. ***Шалыто А.А.*** Технология автоматного программирования / Материалы Всероссийской научно-технической конференции «Методы и средства обработки информации» (МСО-2003). М.: МГУ. 2003, с. 528-535. http://is.ifmo.ru/works/tech_aut_prog.

3.7. ***Шалыто А.А.*** Технология автоматного программирования // Мир ПК. 2003. № 10, с.74-78. http://is.ifmo.ru/works/tech_aut_prog. / Материалы Всероссийской научно-технической конференции «Методы и средства обработки информации» (МСО-2003). М.: МГУ. 2003, с. 528-535. http://is.ifmo.ru/works/tech_aut_prog.

3.8. ***Шалыто А.А.*** Автоматно-ориентированное программирование // Научно-технический вестник СПбГУ ИТМО. Актуальные проблемы современных оптико-информационных систем и технологий. 2005. № 5 (21), с. 35-41. https://ntv.ifmo.ru/ru/journal/102/journal_102.htm.

3.9. ***Шалыто А.А.*** Автоматно-ориентированное программирование / Материалы IX Всероссийской конференции по проблемам науки и высшей школы «Фундаментальные исследования в технических университетах». СПб.: Изд-во Политехнического университета. 2005, с. 44-52. http://is.ifmo.ru/works/_politeh.pdf.

3.10. ***Шалыто А.А.*** Автоматное программирование // Известия Уральского государственного университета. 2006. № 43. Компьютерные науки и информационные технологии. Вып. 1, с. 181-190. http://elar.urfu.ru/bitstream/10995/24543/1/iurm-2006-43-13.pdf.

3.11. ***Шалыто А.А.*** Парадигма автоматного программирования // Научно-технический вестник СПбГУ ИТМО. 2008. Выпуск 53. Автоматное программирование, с. 3-24. http://is.ifmo.ru/works/2008/Vestnik/53/01-automata-based-programming.pdf.

3.12. ***Шалыто А.А.*** Автоматное программирование. Всероссийский конкурс обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы».

3.13. ***Шалыто А.А.*** Автоматное программирование / Труды конференции «Технические и программные средства систем управления, контроля и измерения» М.: ИПУ РАН. 2010, с. 1213-1224. http://is.ifmo.ru/works/2010/UKI-shalyto-automata-programming.pdf.

3.14. ***Шалыто А.А.*** Автоматное программирование / Виртуальный компьютерный музей. https://computer-museum.ru/articles/tekhnologii-programmirovaniya/2433/?sphrase_id=557514.

3.15. ***Шалыто А.А.*** Автоматное программирование. https://vk.com/@1077823-vtomatnoe-programmirovanie.

3.16. ***Шалыто А.А.*** Еще об автоматном программировании. https://vk.com/@1077823-esche-ob-avtomatnom-programmirovanii.

3.17. Большое число статей по автоматному программированию на русском языке приведено здесь: https://is.ifmo.ru/works/.

20.02.2023 г. на «Хабр» появилась статья «**С чем едят автоматы**» (https://habr.com/ru/companies/timeweb/articles/717628/), в которой я оказался в хорошей компании... При этом отмечу, что в статье первый портрет не Мура, как должно быть по тексту, а Шеннона, но от этого компания только улучшается. В статье есть такие слова: «И так, мы показали, как конечные автоматы используются в математике и электронике. **Третье направление, где используются конечные автоматы – программирование.** Идея рассматривать программу в терминах **конечных автоматов** сама по себе не нова. Но наиболее активно свое развитие она получила в начале 90-х годов прошлого века. Одним из основоположников данной направления является **профессор Университета ИТМО Анатолий Абрамович Шалыто**. Его идея состоит в том, чтобы

программировать с использованием понятия «**состояние**». Сперва для названия этого подхода появился термин «***Switch*-технология»**, так как операторы множественного выбора в традиционных языках подходили для смены состояний программы больше всего. Позже, в конце 90-х термин «***Switch*-технология»** был заменен Шалыто на термин «**автоматное программирование**».

4. Статьи, в название которых входит термин «Automata-Based programming». Эти статьи из *Scholar articles for Automata-Based programming*

4.1. *Shalyto A.* Technology of Automata-Based Programming // PC World/Russia. 2003. № 10. http://is.ifmo.ru/automata_en/_tech_aut_prog.pdf.

4.2. *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 Boston Section. 2005, pp. 449-453. http://swan.kgeorgiy.info/papers/Yartsev_B__Korneev_G__Shalyto_A__Kotov_V_--_Automata-Based_Programming.pdf.

4.3. *Paraschenko D., Shalyto A., Tsarev F*. 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 Applications (CIMSA-2006). La Coruna. Spain. 2006, pp.15-20. https://www.academia.edu/31854706/Modeling_Technology_for_One_Class_of_Multi-Agent_Systems_with_Automata_Based_Programming.

4.4. *Gurov V.S., Mazin M.A., Narvsky A.S., Shalyto A.A.* Tools for Support of Automata-Based Programming // Programming and Computer Software. 2007. Vol. 33. No 6, pp. 343-355. https://www.academia.edu/31854647/Tools_for_support_of_automata-based_programming.

1.5. *Kurbatsky E.* Verification of Automata-Based Programs / Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. SPbSU. 2008. V. 2, pp. 15-17. http://is.ifmo.ru/verification/_kurbatsky_syrcse.pdf.

4.6. *Klebanov A.* *Automata*-Based Programming Technology Extension for Generation of JML Annotated Java Card Code / Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. SPbSU. 2008. V. 1, pp. 41-44. http://is.ifmo.ru/articles_en/_klebanov_spbsu.pdf.

4.7. *Kochelaev D., Khasanzyanov B., Yaminov B., Shalyto A.* Instrumental Tool for Automata-Based Software Development *UniMod-2* / Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. SPbSU.2008. https://cyberleninka.ru/article/n/instrumental-tool-for-automata-based-software-development-unimod-2.

4.8. *Shalyto A.A.* Automata-Based Programming and Automata-Based Control. 2009. http://is.ifmo.ru/articles_en/_2009_10_07_automata_based_programming.pdf.

4.9. *Zakonov A., Stepanov O., Shalyto A.* A GA-based approach for Test Generation for Automata-Based Programs / Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. SPbSU. 2010. http://syrcose.ispras.ru/2010/files/syrcose10_submission_12.pdf.

4.10. *Klebanov A.* On the Formal Specification of Automata-Based Programs via Specification Pattern / Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. SPb.: SPbSU. 2010. http://syrcose.ispras.ru/2010/files/syrcose10_submission_4.pdf.

4.11. *Ulyantsev V., Tsarev F.* Extended Finite-State Machine Induction Using SAT-Solver / 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM-2012). Bucharest, Romania, 2012, pp. 512-517. Термин «Automata-Based Programming» используется в тексте https://www.sciencedirect.com/science/article/pii/S1474667016331561.

4.12. *Chivilikhin D., Ulyantsev V.* Inferring Automata-Based Programs from Specification with Mutation-Based Ant Colony Optimization / Proceedings of the Sixteenth Genetic and Evolutionary Computation Conference Companion (GECCO 2014). ACM. NY, pp. 67, 68. https://dl.acm.org/doi/10.1145/2598394.2598446.

4.13. *Koumboulis F., Fragkoulis F., Kouvakas N.* Automata-Based Programming for the Development of a Web-based Application: A Case Study. https://dl.acm.org/doi/abs/10.1145/2984393.2984402. Первая не наша работа, в которой встречается термин *Automata-Based Programming*. **Она вышла в 2016 г. (**https://www.semanticscholar.org/topic/Automata-based-programming/2609355**). В ней ссылаются на нас.**

5. *Tutorial on Automata-Based Programming*

В июне 2006 г. мы провели ***Tutorial on Automata-Based Programming*** в рамках первой международной конференции *International Computer Symposium in Russia* (*CSR 2006*) (**ПОМИ им.**

В.А. Стеклова), в котором было заслушано 24 доклада по этой тематике (https://logic.pdmi.ras.ru/~csr2006/workshops.html, http://unimod.sourceforge.net/).

Одним из докладчиков был ***G. Berry*** (*Estrel Technology,* http://www.esterel-technologies.com), который участвовал в разработке программного обеспечения для *Airbus* (http://is.ifmo.ru/present/_berry-wabp.pdf). В семинаре принял участие и мой старинный знакомый **Михаил Кишиневский**, в то время работавший в корпорации *Intel*, который совместно с *G. Berry* разработал текстово-графический автоматный язык *Estrelv 7*. Вот перечень заслушанных докладов (https://web.archive.org/web/20090412002353/http://unimod.sourceforge.net/wiki/index.php/CSR2006_ABP_WORKSHOP):

1. ***Gurov V.S.*** (eVelopers) Automata-Based Programming Workshop organization.
2. ***Shalyto A.A.*** (SPbSU ITMO) Automata-Based Programming.
3. ***Berry G.*** (Esterel Technologies) Synchronous Programming Techniques for Embedded Systems (http://is.ifmo.ru/present/_berry-wabp.pdf).
4. ***Nepeyvoda N.N.*** (Udmurtia State University) Automata-Based Programming and its Role in Common Informatics Structure.
5. ***Gurov V.S., Mazin M.A., Narvskiy A.S.*** (eVelopers), ***Shalyto A.A.*** UniMod – a CASE-tool for Automata-Based Programming.
6. ***Gurov V.S., Mazine M.A.*** (eVelopers) Automata-Based design of applications for mobile devices.
7. ***Kuzmin E.V., Sokolov V.A.*** (State University of Yaroslavl) Modeling, Specification and Verification of «Automata» Programs.
8. ***Vasiljeva K.A., Kouzmin E.V., Sokolov V.A.*** (State University of Yaroslavl) LTL-based Verification of «Automata» Programs.
9. ***Lubchenko V.S.*** On physics of Automata-Based Concurrent Programming.
10. ***Shopyrin D.G***. (ZAO Tranzas Technologies) Graphical notation for automata objects inheritance.
11. ***Shamgunov N.N.*** (Microsoft), ***Korneev G.A.*** (SPbSU ITMO) State Machine design pattern.
12. ***Vavilov K.V.*** Programmable logic controller and Automata-Based Programming.
13. ***Kazakov M.A.*** (Flextronics) Discrete mathematics algorithms visualizers design with Automata-Based Programming.
14. ***Korneev G.A.*** (SPbSU ITMO) Automatized approach to visualizer`s design with finite state machines.
15. ***Babaev A.*** Automata-Based UI.
16. ***Korotkov M.A., Loukianova A.P.*** (eVelopers) Automata-Based Design of WEB-applications.
17. ***Polikarpova N.I.*** (SPbSU ITMO) A Notion of Subtyping for Types with State Dependent Behavior.
18. ***Stepanov O.G.*** (JetBrains) Automata-Based Programming Using Dynamic Programming Languages.
19. ***Astafurov A.A.*** (DataArt) Automata Objects Nesting and Inheritance Using Meta Information in Object Oriented Languages.
20. ***Yartsev B.M.*** (SPbSU ITMO) Automata-Based Design of the real-time systems.
21. ***Tsarev F.N., Paraschenko D.A.*** (SPbSU ITMO) Modeling Technology of One Class of Multi-Agent Systems with Automata-Based Programming.
22. ***Rudnev A.D.*** (SPbSU ITMO) Using Automata-Based Design in developing driver for floppy disk controller.
23. ***Kanzhelev S.Y.*** (AVIcode) Automatic automata's code generation.
24. ***Naumov A.S.*** (SPbSU ITMO) Virtual Machine for Automata-Based Programming.

6. Как правильно строить схемы алгоритмов

При необходимости использовать схемы алгоритмов (этот термин заменил термин «граф-схемы алгоритмов»), **предлагаю начинать их построение с дешифратора состояний, а не дешифратора входных воздействий, как это делается обычно**. Построенные таким образом схемы изоморфны конструкции *switch* в языках программирования, а схемы алгоритмов, построенные иначе – не изоморфны этой конструкции. **Если не знать в каком состоянии находится система управления, то какой смысл опрашивать входные переменные?** Однако большинство инженеров обращать на это внимание, почему-то, не хотят – видимо, потому что их так не учили программировать. Такие схемы названы мной – «**автоматными схемами алгоритмов**».

Этот подход описан в статье ***Шалыто А.А.*** Использование граф-схем и графов переходов при программной реализации алгоритмов логического управления. I, II // Автоматика и телемеханика. 1996. № 6, 7.

(http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=at&paperid=3235&option_lang=rus, http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=at&paperid=3251&option_lang=rus).

Эта статья существует и на английском языке, так как этот журнал переводится: ***Shalyto A.A.*** Algorithmic Graph Schemes and Transition Graphs: Their Use in Software Realization of Logical Control Algorithms. I. // Automation and Remote Control. 1996. Vol. 57. No 6, pp. 890-897 и ***Shalyto A.A.*** Algorithmic Graph Schemes and Transition Graphs: Their Use in Software Realization of Logical Control Algorithms. II. // Automation and Remote Control. 1996. Vol. 57. No 7, pp.1027-1045.

7. Автоматное программирование и язык «ДРАКОН»

Почти через 25 лет после опубликования моей статьи о «правильном» построении схем алгоритмов появилась работа ***Митькина С.Б.*** Автоматное программирование на языке ДРАКОН // Программная инженерия. 2019. №1, с. 3-13 (https://drakonhub.com/files/pe_drakon_automata_mitkin_2019.pdf), в которой **он ссылается на две мои книги**, указанные выше, но не обращает внимания на два важных момента: во-первых, на главу 13 (http://is.ifmo.ru/books/switch_pdf/_switch13.pdf) в первой из этих книг, опубликованной в **1998 г.,** которая базируется на указанной выше статье о схемах алгоритмов, а во-вторых, на то, что в этой книге приводятся примеры **применения автоматного программирования в базисе различных языков программирования**, в том числе и по стандарту *IEC 1131-3*. Естественно, что для этой цели могут использоваться и многие другие языки, например «ДРАКОН», что и было сделано Митькиным.

Как бы там ни было, после публикации статьи Митькина появился раздел **«9. Автоматное программирование на языке «ДРАКОН»** в статье в Википедии об этом языке (https://ru.wikipedia.org/wiki/ДРАКОН). Это произошло **не ранее 2019 г.** До этого я читал книги создателя ДРАКОНА **В.Д. Паранджонова**, ссылался на него, а он на меня – нет. Я ему безответно писал. В его работах не использовалось понятие «состояние», и **в этом было одно из принципиальных отличий автоматного программирования от «ДРАКОНА»**. У него правильно построенные граф-схемы – это всякое разное, а не граф-схемы, построенные изоморфно конструкции *Switch*, начиная с дешифратора состояний, как отмечено выше.

И еще один момент, на который Митькин не обратил внимание: графы переходов более обозримы, чем любые граф-схемы – хоть, с использованием «ДРАКОНА», хоть построенные иначе, так как первые обычно двумерны, а вторые имеют направленность сверху вниз.

До этого Митькин писал: «До недавнего времени у меня в голове был смысловой разрыв между иконами реального времени языка «ДРАКОН» (Вставка, Ввод) и конечными автоматами. И то, и другое моделирует динамику взаимодействия, но по-разному», а потом у него произошло ... «озарение»: «Взаимодействующие процессы весьма удобно реализовывать в виде конечных автоматов. «ДРАКОН»-схема преобразуется в конечный автомат, причем для каждой иконы «Вставка» и «Ввод» создается отдельное состояние. Выполнение процесса происходит в виде работы конечного автомата, который движется от одного состояния к другому» (https://forum.drakon.su/viewtopic.php?f=142&t=6631). Потом он пишет: «**Конечные автоматы на «ДРАКОНЕ» – это бомба. Жаль, что земляне этого не понимают**» (https://forum.drakon.su/viewtopic.php?f=142&t=6246). Я согласен с этим утверждением, но только без использования двух лишних слов: «**Конечные автоматы – это бомба. Жаль, что земляне этого не понимают**». **По моему мнению, иконы целесообразно применять для другой цели, а в программирование следует использовать графы переходов.**

Потом я узнал, что на сайте «Визуальный язык «ДРАКОН» (https://drakon.su/) с перечне форумов есть и такой: «Теоретические основы языка «ДРАКОН» (https://forum.drakon.su/viewforum.php?f=156), а в нем – **обсуждение на тему «Язык «ДРАКОН» и конечные автоматы»** (https://forum.drakon.su/viewforum.php?f=142), где существуют страницы, на которых рассматриваются вопросы, связанных с автоматным программированием и мною.

На странице «Язык «ДРАКОН». Метод Шалыто и важное предложение Игоря Мазницы» (https://forum.drakon.su/viewtopic.php?f=142&t=5950) мой старинный знакомый Игорь Мазница в **2016 г.** обратил внимание автора языка на то, что я не волен тем, что он не ссылается на меня. На это Пароджанов ответил: «**Анатолий Шалыто прав**. Отсутствие ссылок на его работы по

автоматному программированию, *Switch*-технологии – большое упущение. В следующей книге я постараюсь обязательно устранить этот недостаток». ***Он, видимо, постарался, но у него не получилось***: в его книге «Алгоритмы и жизненные ритмы на языке «ДРАКОН». Разработка алгоритмов» (https://drakon.su/_media/24_zhizneritm20.pdf), **датированной 2019 г**., нет ни слова ни обо мне, ни об автоматном программировании, ни о *Switch*-технологии.

В **2017 г.** Мазница сообщил драконовцам о существовании записи моей лекции про автоматное программирование (https://www.youtube.com/watch?v=tUo9ssPVa4c), и на их сайте появилась **страница «Лекция Анатолия Шалыто про автоматное программирование»** (https://forum.drakon.su/viewtopic.php?f=142&t=6133). О ней Степан Митькин написал «**Отличная лекция. Просто, понятно, а главное – с душой**». А еще им было сказано: «На днях я учинил у нас в офисе зачет по автоматам. Опросил нескольких программистов. Итог: все слышали это слово, но никто точно не знает, что это такое (**хорошо в этой ситуации применять в программировании автоматы, что мои недоброжелатели считают очевидным, А.Ш.**). Похоже, автоматы – это какое-то тайное знание, доступное только элите и аннукам (божествам, А.Ш.)».

В **2018 г.** у драконовцев «всплыла» упомянутая выше моя статья о правильном построении граф-схем (https://forum.drakon.su/viewtopic.php?f=142&t=6246). Пообсуждали...

В том же году на странице **«А. Шалыто. Проектный подход при обучении разработке программ»** (https://forum.drakon.su/viewtopic.php?f=142&t=6289) Пароджановым приведена половина (?) моей статьи «**Проектный подход при обучении разработке программ**» (http://is.ifmo.ru/award/_doklad_uch_sovet.pdf), которая была опубликована в журнале Компьютерные инструменты в образовании. 2009. № 4, с. 32-38 (http://ipo.spb.ru/journal/content/1074/Проектный_подход_обучении_разработке_программ.pdf). Этот текст на указанной странице никак не обсуждается...

В **2019 г.** лекцию про автоматное программирование я выложил на своем *YouTube*-канале в лучшем качестве, чем она была опубликована там раньше (https://www.youtube.com/watch?v=PPWTxceMutk).

На странице «**Язык «ДРАКОН», метод Шалыто, метод Ашкрофта-Манны**» (https://forum.drakon.su/viewtopic.php?f=142&t=5724) Пароджанов пишет: «Анатолий Абрамович Шалыто в статье 1996 г. сравнивает свой метод с методом Ашкрофта-Манны и делает вывод, что его метод удобнее. Он не сравнивает свой метод с языком ДРАКОН, так как **этот язык тогда был практически неизвестен**». Как говорится: «**Когда знаешь, как правильно, зачем делать иначе?**» – ниже будет показано, что **уже в 1991 г. при создании судовых систем управления мы успешно использовали автоматное программирование**. Конечно, последний довод слабый, так как «ДРАКОН» тоже использовали, да и мало чего только люди не применяют…

8. Сборники работ по автоматному программированию

8.1. В 2008 г. был издан первый в мире сборник по автоматному программированию: Научно-технический вестник СПбГУ ИТМО. 2008. № 8 (53). **Автоматное программирование** (https://ntv.ifmo.ru/ru/journal/61/journal_61.htm). Он содержит **28 наших статей** по этой теме (http://is.ifmo.ru/works/).

8.2. В **2011 г.** был издан Научно-технический вестник СПбГУ ИТМО. 2011. № 2 (72), https://ntv.ifmo.ru/ru/journal/28/journal_28.htm. Он содержит **17 наших статей** (http://is.ifmo.ru/works/) и имеет подзаголовок «Технологии автоматного программирования и искусственного интеллекта». Отмечу первую статью сборника: ***Александров А.В., Казаков С.В., Сергушичев А.А., Царев Ф.Н., Шалыто А.А***. Генерация конечных автоматов для управления моделью беспилотного самолета // Научно-технический вестник СПбГУ ИТМО. 2011. № 2 (72), с. 3-11. https://ntv.ifmo.ru/file/article/21970.pdf.

9. Автоматное программирование против классиков

9.1. Как надо программировать калькулятор (http://is.ifmo.ru/projects/calc/). Не делать, как предлагает **Б. Страуструп** – почетный доктор Университета ИТМО.

9.2. Как надо программировать лифт (http://is.ifmo.ru/projects/lift2/, http://is.ifmo.ru/works/lift2.pdf). Не делать, как предлагает **Д. Кнут**.
9.3. Как надо программировать систему сбора данных (http://is.ifmo.ru/projects/meteo/). Не делать, как предлагает **Г. Буч**.

10. Инструментальные средства поддержки автоматного программирования
10.1. Конвертер ***Visio2Switch*** (http://is.ifmo.ru/automata/visio2switch/).
10.2. Инструментальное средство ***MetaAuto*** для автоматической генерации автоматных программ на любом априори заданном языке программирования по графам переходов (http://is.ifmo.ru/projects/metaauto/). Это средство описано в статье: ***Канжелев С.Ю., Шалыто А.А.*** Автоматическая генерация автоматного кода // Информационно-управляющие системы. 2006. № 6, с. 35-42. http://is.ifmo.ru/works/_autogen.pdf. Презентация на эту тему размещена по адресу: http://www.myshared.ru/slide/128477/. Это средство используется моим аспирантом **А.В. Калачинским** в НПО «Аврора» в его технологии разработки программного обеспечения на основе автоматного подхода.
10.3. Инструментальное средство для объектно-ориентированного подхода к автоматному программированию ***UniMod*** (https://unimod.sourceforge.io, https://www.youtube.com/watch?v=Y4et51dz-HE). Оно описано в статье: ***Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А.*** Инструментальное средство для поддержки автоматного программирования // Программирование. 2007. № 6, с. 65-80. http://is.ifmo.ru/works/_2008_01_27_gurov.pdf. ***UniMod*** **использовался не только в учебном процессе в Университете ИТМО** (http://is.ifmo.ru/unimod-projects/), **но и в Италии** (https://sepl.dibris.unige.it/publications/2012-ricca-MiSE.pdf). Мы получили свидетельства о государственной регистрации программ для ЭВМ: **«Ядро автоматного программирования»** (http://is.ifmo.ru/unimod/_svid.pdf) **и «Встраиваемый модуль автоматного программирования для среды разработки *Eclipse*»** (http://is.ifmo.ru/unimod/_svid2.pdf). У некоторых пользователей *UniMod* стал любимым инструментом (https://biese.wordpress.com/2007/02/06/using-finite-state-machine-tools-to-solve-the-probelm/).
10.4. Мне кажется, что лучшее инструментальное средство для поддержки автоматного программирования – **пакет *Stateflow*** (https://en.wikipedia.org/wiki/Stateflow), который в последние годы активно использует другой мой аспирант в НПО «Аврора» **Ю.Ю. Янкин** при программировании ПЛИС для управления электроприводом регулирующих органов корабельных энергетических установок.
10.5. Для обеспечения импортозамещения **А.В. Калачинский** разработал инструментальное средство, которое стало основой технологии проектирования программного обеспечения на основе автоматного подхода **(*Калачинский А.В.*** Технология проектирования программного обеспечения систем дискретного управления на основе автоматного подхода // Системы управления и обработки информации. 2023. Вып. 62, с. 30-47. Она в некотором смысле импортозамещает *Stateflow.* Один из этапов технологии описан здесь: ***Калачинский А.В., Ященко И.Н.*** Генерация описания работы автоматных программ в документ формата *PDF* // Системы управления и обработки информации. 2019. Вып. 44, с. 93-98 (https://www.avrorasystems.com/upload/iblock/cf3/cf3801161b1ca1a2fa7ba969079c45ec.pdf).
10.6. В указанных инструментальных средствах автоматы строятся вручную. Для *Stateflow* нами впервые была сделана попытка с помощью машинного обучения повысить уровень автоматизации его применения за счет генерации по сценариям поведения графа переходов автомата, который и загружается в указанное средство. Для этого было разработано программное средство, позволяющее идентифицировать, загрузить и корректно отобразить граф переходов автомата в среде *Stateflow*: **1.** ***Ведерников Н.В., Демьянюк В.Ю., Кротков П.А., Ульянцев В.И., Шалыто А.А.*** Автоматизированное построение управляющих автоматов в среде *Stateflow* при помощи методов машинного обучения / Материалы научной конференции по проблемам информатики СПИСОК-2014. Матмех. СПбГУ. 2014, с. 411-417. http://spisok.math.spbu.ru/2014/txt/SPISOK-2014.pdf. **2.** ***Ведерников Н.В., Демьянюк В.Ю., Кротков П.А., Ульянцев В.И., Шалыто А.А.*** Применение методов машинного обучения для автоматизированного построения управляющих автоматов в высокоуровневых средствах проектирования систем / XII Всероссийское совещание по проблемам управления (ВСПУ-2014). ИПУ РАН, с. 3159-3166. http://is.ifmo.ru/works/2014/2014_VSPU_Vedernikov_et_al.pdf. Естественно, это было только первым шагом в указанном направлении. Второго шага с тех пор сделано не было...

11. Учебные автоматные проекты

11.1. Курсовые проекты по автоматному программированию, выполненные с **использованием процедурного подхода**, которые включают проектную документацию, размещены здесь: http://is.ifmo.ru/projects/.
11.2. Курсовые проекты по автоматному программированию, выполненные с **использованием объектно-ориентированного подхода,** которые включают проектную документацию, размещены здесь: http://is.ifmo.ru/unimod-projects/. Применяемое инструментальное средство – разработанный нами ***UniMod***. В качестве примера рекомендую посмотреть проект, расположенный по адресу: http://is.ifmo.ru/unimod-projects/camera/.

12. Генерация конечных автоматов для моделей самолетов
12.1. ***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, с. 100-117. http://is.ifmo.ru/works/_polikarpova_samolet.pdf.
12.2. ***Александров А.В., Казаков С.В., Сергушичев А.А., Царев Ф.Н., Шалыто А.А.*** Генерация конечных автоматов для управления моделью беспилотного самолета // Научно-технический вестник СПбГУ ИТМО. 2011. № 2 (72), с. 3-11. https://ntv.ifmo.ru/file/article/21970.pdf.
12.3. ***Казаков С.В., Царев Ф.Н., Шалыто А.А.*** Метод построения конечных автоматов верхнего уровня для управления моделью беспилотного самолета на основе обучающих примеров // Научно-технический вестник информационных технологий, механики и оптики. 2011. № 6 (76), с. 64-68. https://ntv.ifmo.ru/file/article/830.pdf.
12.4. ***Александров А.В., Казаков С.В., Сергушичев А.А., Царев Ф.Н., Шалыто А.А.*** Применение эволюционного программирования на основе обучающих примеров для генерации конечных автоматов, управляющих объектами со сложным поведением // Известия РАН. Теория и системы управления. 2013. № 3, с. 85-100. http://is.ifmo.ru/works/2013/alexandrov_samolet.pdf.
12.5. ***Бужинский И.П., Казаков С.В., Ульянцев В.И., Царев Ф.Н., Шалыто А.А.*** Модификация метода генерации управляющих конечных автоматов с непрерывными воздействиями по обучающим примерам // Известия РАН. Теория и системы управления. 2015. № 6, с. 17-30. http://is.ifmo.ru/works/2015/buzhinsky-tisu-2015.pdf.

13. Программирование логических контроллеров
Программируемые логические контроллеры (ПЛК) программировались и программируются в настоящее время в соответствии со стандартом ***IEC 1131-3***, в котором описаны пять языков программирования. Говорят, что наиболее используемым из этих языков является **язык «функциональных блоков»**. Однако, по моему мнению, он обычно применяется неэффективно. А как надо?

Эффективное программирование в этом случае происходит, если набор функциональных блоков содержит блок «**цифровой мультиплексор**» – **аналог конструкции *switch*** в языках программирования. Тогда граф переходов изоморфно реализуется с использованием такого мультиплексора. Реализация описана здесь: (http://is.ifmo.ru/books/switch_pdf/_switch4.pdf). Более 30 лет назад такие блоки уже входили в наборы блоков для программирования аппаратуры, например аппаратуры ***Selma*-2** фирмы ***ABB Stromberg*** (Финляндия).

Впервые этот подход мы использовали в 1991 г. при создании системы управления дизель-генератором ДГР-2А 500*500 судна проекта 15640 на базе аппаратуры *Selma-2*. Программирование выполнялось в **НПО «Аврора»** на языке функциональных блоков с использованием цифровых мультиплексоров.

При этом **по графам переходов строились изоморфные им функциональные схемы**, что до тех пор не делалось (*Project 15640. AS21. DG21. Control. АМИЕ. 95564.12M. St. Petersburg. ASS «Avrora», 1991*). **Это позволило нашему немолодому сотруднику, *который не умел программировать*, разработать и успешно сдать указанную систему на судне,** а также закрыть построечное удостоверение раньше проектировщиков других систем комплекса, которые были не только хорошими инженерами, но и **умели программировать, но делали это иначе...** Этот сотрудник с моей помощью создал графы переходов и научился, как от них изоморфно переходить к текстам программ – к функциональным схемам, ядром которых были цифровые мультиплексоры. Незначительные изменения в программе, созданной таким образом, необходимость внесения

которых возникла на судне, этот специалист в так спроектированной программе смог успешно произвести. Аналогичная работа была выполнена нами и для судна проекта 15967.

Первый текст про автоматное программирование, как отмечалось выше, также появился в 1991 г.: ***Шалыто А.А.*** Программная реализация управляющих автоматов // Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып. 13, с. 41, 42, https://is.ifmo.ru/works/switch_prr/.

В последние годы **в мире начинает использоваться стандарт *IEC 61499***, определяющий построение распределенных сетей на ПЛК, в соответствии с которым поведение функциональных блоков предлагается **описывать графами переходов**! (***Pang C., Patil S., Yang C., Vyatkin V., Shalyto A.*** A Portability Study of IEC 61499: Semantic and Tools / 12th IEEE International Conference on Industrial Informatics (INDIN 2014). 2014. Port Alegre, Brazil, pp. 440-445. https://ieeexplore.ieee.org/document/6945553).

14. Другие примеры применения автоматного программирования в промышленности

14.1. **В 1993 г.** использование графов переходов позволило при создании системы управления дизель-генератором ДГР-2А 500*500 для судна проекта 15760 найти общий язык при взаимодействии с сотрудниками фирмы ***Norcontrol*** **(Норвегия)**, для которых, как и для нас, английский язык не является родным. При этом мы научили наших партнеров программировать по шаблону на текстовом языке ПЛ/М графы переходов, которые нами выдавались им в качестве технического задания. Применение графов переходов резко упростило наше взаимодействие с представителями фирмы**,** и **они включили представленные нами графы переходов в документацию на систему, что до этого никогда не делали (*Functional Description. Warm-up & Prelubrication Logic. Generation Control Unit. Severnaya Hull no. 431. Norcontrol, 1993.*** http://is.ifmo.ru/progeny/_appl_doc2.pdf). Формализация на основе использования графов переходов автоматов при выдаче технического задания **позволила *разделить работу, а главное, ответственность* между нашими организациями**. Это также обеспечило возможность **проводить корректировку алгоритмов и программ не в терминах судовых устройств, как это делалось до сих пор, а в терминах автоматов, что для программистов значительно проще и понятнее**.

14.2. В 1999 г. графы переходов использовались при создании комплексной системы управления техническими средствами для судна проекта 17310 на базе ПЛК *Autolog*. Программирование было выполнено НПО «Аврора» на языке инструкций *ALPro*, который совместим с ассемблером микроконтроллеров *Intel 8051*. Для общесудовых систем оно выполнялось вручную, а для систем управления вспомогательными механизмами главного двигателя – с использованием транслятора «Ядро языка Си – язык *ALPro*», который был создан **Б.П. Кузнецовым** при участии автора. Во втором случае алгоритмы управления описывались графами переходов, после чего они изоморфно реализовывались программой на языке *C*, которая, в свою очередь, транслировалась в программу на языке *ALPro*.

14.3. *Туккель Н.И., Шалыто А.А.* Проектирование программного обеспечения системы управления дизель-генераторами на основе автоматного подхода // Системы управления и обработки информации. 2002. Вып. 5, с. 66-82. http://is.ifmo.ru/works/diesel/.

14.4. *Туккель Н.И., Шалыто А.А.* Система управления дизель-генератором (фрагмент). Программирование с явным выделением состояний. **Проектная документация** (http://is.ifmo.ru/projects/dg/). 2002.

14.5. Системы управления для «Водоканала»:

14.5.1. *Вавилов К.В.* Программируемые логические контроллеры *SIMATIC S7-200* (*SIEMENS*). Методика алгоритмизации и программирования задач логического управления. 2005. http://is.ifmo.ru/automata/_metod065.pdf;

14.5.2. *Вавилов К.В.* Контроллеры *SIMATIC S7-300 (SIEMENS)*. Организация взаимодействия независимых локальных систем управления на основе автоматного подхода и функционального разделения автоматов управления. 2005. http://is.ifmo.ru/automata/_s7300.pdf;

14.5.3. *Вавилов К.В.* *LabVIEW* и *Switch*-технология. Методика алгоритмизации и программирования задач логического управления. 2005. http://is.ifmo.ru/automata/_vavilov2.pdf.zip.

В свое время автор последних из указанных работ спросил у своих сотрудников, как они относятся к автоматному программированию. Их ответ был аналогичен **ответу Черчилля о демократии:** «Это наихудшая форма правления, если не считать всех остальных *опробованных».*

И еще. Обращаю Ваше внимание, что автоматный подход, как было показано на примерах в моей книге о Switch-технологии может быть применен при использовании аппаратных и программных средств разных типов. Как указано выше, ***Константин Вавилов*** *применял его для контроллеров фирмы Siemens*, которая о таком подходе к их программированию в своей документации **даже не заикалась**. В то время я неоднократно пытался объяснить различным представителям этой компании, чем такой подход может быть ей полезен. Однако все попытки заканчивались безрезультатно, так как эта компания великая, и без моих предложений жила, живет и, видимо, еще долго будет жить хорошо. Да и я без них живу тоже неплохо...

15. Что такое автоматное программирование?
*На онлайн-конференции по языку «Оберон» (*https://conf.oberon.org/schedule) в 2020 г. **Валерий Викторович Лаптев** из Астраханского государственного технического университета при обсуждении моего доклада сформулировал утверждение: «Автоматное программирование – это программирование для непрограммистов».

С этим утверждением не согласился участник той же конференции **главный эксперт АО «**Русатом – Автоматизированные системы управления» **(РАСУ) (**https://rasu.ru/company/**) Дмитрий Викторович Дагаев: «Можно точно сказать: «Автоматное программирование – это программирование, объяснимое и для специалистов-непрограммистов**. И, в отличие от других подходов, визуализируемое и масштабируемое**». Меня такое несогласие вполне устроило!**

После этого Дагаев написал статью о своем взгляде на автоматное программирование: «Исполняющая машина автоматных программ» (https://ntv.ifmo.ru/file/article/20577.pdf).

16. Применение автоматов при программировании мобильных устройств
16.1. ***Клебан В.О., Шалыто А.А.*** Использование автоматного программирования для построения систем управления мобильными роботами (http://is.ifmo.ru/present/_kleban_shalyto.ppt).
16.2. ***Клебан В.О., Шалыто А.А.*** Использование автоматного программирования во встраиваемых системах (http://is.ifmo.ru/present/_kleban.ppt).
16.3. ***Клебан В.О., Шалыто А.А.*** Разработка системы управления малоразмерным вертолетом (http://is.ifmo.ru/works/2011/Vestnik/72-2/02-Kleban-Shalyto.pdf).
16.4. ***Kleban V.*** Fully autonomous helicopter flight (http://www.youtube.com/watch?v=-LuVLH4cV0U). Более 10 000 просмотров.
16.5. ***Алексеев С.А., Калиниченко А.И., Клебан В.О., Шалыто А.А.*** Автоматический синтез системы управления мобильным роботом для решения задачи «Кегельринг» (http://is.ifmo.ru/works/2011/Vestnik/72-2/05-Alekseev-Kalinichenko-Kleban-Shalyto.pdf).
Виталий Клебан мне как-то рассказывал, что он сдавал на объекте управляющую систему, некоторые подсистемы которой были реализованы автоматно, а другие – традиционным путем. При этом автоматные подсистемы либо сразу правильно работали, либо не работали, но их работоспособность обеспечивалась достаточно просто. Уверенность в правильности работы остальных подсистем отсутствовала даже после их сдачи. В настоящее время Виталий соавтор стандарта *LoRaWAN*, член соответствующего технического комитета (https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf).

17. Программирование сложной игры
17.1. ***Озеров А.А.*** Четыре танкиста и компьютер (еще раз об игре *Robocod*e) // Магия ПК. 2002. № 11. (http://is.ifmo.ru/?i0=aboutus&i1=5).
17.2. ***Туккель Н.И., Шалыто А.А.*** Система управления танком для игры *Robocode*. Вариант 1. Объектно-ориентированное программирование с явным выделением состояний. Проектная документация. 2001. (http://is.ifmo.ru/projects/tanks/).
17.3. ***Кузнецов Д.В., Шалыто А.А.*** Система управления танком для игры *Robocode.* Вариант 2. 2003. (http://is.ifmo.ru/projects/robocode2/).

18. Визуализация на автоматах
18.1. Программирование виджетов (Часть 1 – https://www.ibm.com/developerworks/ru/library/wa-finitemach1/index.html. Часть 2 – https://www.ibm.com/developerworks/ru/library/wa-finitemach2/wa-finitemac_ru.html. Часть 3 – http://www.interface.ru/home.asp?artId=7867).
18.2. Программирование визуализаторов алгоритмов (http://is.ifmo.ru/works/vis/, http://is.ifmo.ru/works/art_vis.pdf, http://is.ifmo.ru/works/visanim.pdf).

19. Протоколы на автоматах
19.1. *SMTP* (http://is.ifmo.ru/projects/smtp/).
19.2. *TCP* (http://is.ifmo.ru/download/red4.pdf).

20. Верификация автоматных программ
Метод верификации ***Model Checking*** **классно работает при верификации автоматных моделей для простых задач, а на практике – при валидации автоматных спецификаций.** Это объясняется с тем, что описание парадигмы *Model Checking* начинается со слов: **»По программе строится модель»** со всеми вытекающими отсюда неприятными последствиями, в то время как при применении автоматного программирования **»по автоматной модели формально и изоморфно строится программа».** После этого все встает на свои места, и можно эффективно применять *Model Checking* к автоматной модели, с которой автоматное программирование должно начинаться.

Параллельно с нами исследования по формальной верификации автоматных программ стали проводиться в Ярославском государственном университете им. П.Г. Демидова, в котором сотрудники кафедры теоретической информатики к тому времени уже много лет занимались верификацией программ. Интерес к верификации именно **автоматных** программ у доктора физ.-мат. наук **В.А. Соколова** и кандидата (сейчас доктора) физ.-мат. наук **Е.В. Кузьмина инициировал я** на второй Всероссийской научной конференции «Методы и средства обработки информации», **которая проходила в МГУ в 2005 г.** Более того, я указал им на «полигон» для экспериментов, состоящий из курсовых проектов моих студентов (http://is.ifmo.ru/projects/). В 2007 г. мы выиграли грант «**Разработка технологии верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода**», в котором «ребята» из Ярославля были соисполнителями http://is.ifmo.ru/verification/_2007_01_report-verification.pdf, http://is.ifmo.ru/verification/_2007_02_report-verification.pdf, http://is.ifmo.ru/verification/_2007_03_report-verification.pdf, http://is.ifmo.ru/verification/_2007_04_report-verification.pdf.

Для этого класса программ интерес представляет также работа: «*NASA*: миссия надежна» (https://www.osp.ru/os/2004/03/184060/).

20.1. ***Вельдер С.Э., Шалыто А.А.*** (2006) Введение в верификацию автоматных программ на основе метода *Model Checking.* 52 с. http://is.ifmo.ru/download/modelchecking.pdf.
20.2. ***Вельдер С.*** (2006) Введение в верификацию автоматных программ на основе метода *Model Chcking* (http://is.ifmo.ru/papers/_velder_bachelor.pdf).
20.3. ***Кузьмин Е.В.*** (2006) Иерархическая модель автоматных программ (http://is.ifmo.ru/verification/_hamp.pdf).
20.4. ***Кузьмин Е.В., Соколов В.А.*** (2006) О верификации «автоматных программ» (http://is.ifmo.ru/verification/_verautpr.pdf).
20.5. ***Виноградов Р.А., Кузьмин Е.В. Соколов В.А.*** (2006) Верификация автоматных программ средствами *CPN/Tools* (http://is.ifmo.ru/verification/_cpnverif.pdf).
20.6. ***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 информационные технологии». СГУ. 2007, с. 66-69. http://is.ifmo.ru/verification/_KNIT-2007.pdf.
20.7. ***Кузьмин Е.В., Соколов В.А.*** (2007) «О дисциплине специализации «Верификация программ» (http://is.ifmo.ru/verification/_ver_prog.pdf).
20.8. ***Кузьмин Е.В,. Соколов В.А.*** (2007) О некоторых подходах к верификации автоматных программ (http://is.ifmo.ru/verification/_2007_10_01_verification.pdf).
20.9. ***Васильева К.А., Кузьмин Е.В.*** Верификация автоматных программ с использование *LTL* (http://is.ifmo.ru/verification/_LTL_for_Spin.pdf).
20.10. ***Кузьмин Е.В., Соколов В.А.*** (2008) Моделирование, спецификация и верификация «автоматных» программ (http://is.ifmo.ru/download/2008-03-12_verification.pdf).
20.11. ***Шалыто А.А., Царев Ф.Н.*** (2008) Технология верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода (http://is.ifmo.ru/present/_verification-itmo.ppt).

20.12. ***Лукин М.А., Шалыто А.А.*** (2008) Верификация автоматных программ с помощью верификатора *SPIN* (http://is.ifmo.ru/works/2008/Vestnik/53/13-verification-of-automata-based-programs-with-SPIN.pdf).
20.13. ***Вельдер С.Э., Шалыто А.А.*** (2008) Методы верификации автоматных программ (http://is.ifmo.ru/works/2008/Vestnik/53/11-verification-of-automata-models.pdf).
20.14. ***Егоров К.В., Шалыто А.А.*** Методика верификации автоматных программ // Информационно-управляющие системы. 2008. № 5, с. 15-21. http://www.i-us.ru/index.php/ius/article/view/14782.
20.15. ***Вельдер С.Э., Шалыто А.А.*** Верификация автоматных моделей методом редуцированного графа переходов // Научно-технический вестник СПбГУ ИТМО. 2009. № 6 (64), с. 66-77. https://ntv.ifmo.ru/file/article/91.pdf.
20.16. ***Кузьмин Е.В., Соколов В.А., Чалый Д.Ю.*** Применение метода формальных утверждений о трассах для спецификации, построения и верификации автоматных программ // Программирование. 2009. № 1, с. 61-77. http://is.ifmo.ru/disser/kuzmin_autoreferat.pdf.
20.17. ***Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.*** Верификация автоматных программ. СПб.: Наука, **2011**. 244 с. (http://is.ifmo.ru/verification/velder_verification_posobie_nauka.pdf).
20.18. ***Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.*** Верификация автоматных программ. Учебное пособие. СПбГУ ИТМО, 2011. 242 с. http://is.ifmo.ru/verification/velder_verification_posobie.pdf.
20.19. ***Ульянцев В.И., Шалыто А.А.*** (2013) О верификации простых программ со сложным поведением (http://is.ifmo.ru/works/2013/ulyantsev-shalyto-verification.pdf, https://vk.com/@1077823-o-verifikacii-prostyh-programm-so-slozhnym-povedeniem).
20.20. ***Лукин М.А., Шалыто А.А.*** Разработка и автоматическая верификация параллельных автоматных программ // Информационно-управляющие системы. 2013. № 5, с. 43-50. http://www.i-us.ru/index.php/ius/article/view/13674.
20.21. ***Лукин М.А.*** Верификация параллельных автоматных программ // Научно-технический вестник информационных технологий, механики и оптики. 2014. № 1 (89), с. 60-66. https://ntv.ifmo.ru/file/article/8323.pdf.

Особо хочу отметить работу ***Егоров К.В., Царев Ф.Н., Шалыто А.А.*** Совместное применение генетического программирования и верификации для построения автоматов управления системами со сложным поведением // Труды СПИИРАН. 2010. Вып. 15, с. 123-135. http://proceedings.spiiras.nw.ru/index.php/sp/article/view/1523/1386, **в которой предложено верификацию проводить не после построения автоматов, а в ходе их генерации!**

Этой работе предшествовала статья на ту же тему: ***Егоров К.В., Царев Ф.Н.*** Совместное применение генетического программирования и верификации моделей для построения автоматов управления системами со сложным поведением / Сборник докладов конференции молодых ученых и специалистов «Информационные технологии и системы» (ИТиС' 09). М.: ИППИ РАН. 2009, с. 77-82 (http://is.ifmo.ru/genalg/_2010_01_14_egorov_tsarev.pdf), которая **была восторженно принята двумя рецензентами – докторами наук!**

21. Валидация автоматных спецификаций
Этот вопрос изложен в статье *Шалыто А.А.* Валидация автоматных спецификаций // Научно-технический вестник информационных технологий, механики и оптики. 2023. № 2, с. 436-438. https://ntv.ifmo.ru/file/article/21921.pdf). ***С этой статьей также можно ознакомиться ниже***.

Вот как она начинается.

Под верификацией [1] обычно понимают проверку неформально построенной прикладной программы на предмет выполнения формальной спецификации. Этот процесс надо проводить заново для каждой вновь созданной прикладной программы.

Считается, что **верификация** позволяет установить, что «**мы создали продукт таким, каким и намеревались его сделать**», а **валидация** подтверждает, что «**мы создали правильный продукт**». **Поэтому ошибка в общем случае – не только то, что устанавливается формально (верификацией), но и неформально (валидацией)**. При этом **тестирование** может рассматриваться, как разновидность валидации, которая проводится с целью определения соответствия поведения системы её ожидаемому поведению на конечном наборе тестов.

Если формальная спецификация становится исполняемой, то вместо верификации каждой из генерируемых прикладных программ верификация может проводиться только один раз –

однократно верифицируется только программа-генератор прикладных программ. Если квалификации создателей генератора для его верификации не хватает, то он может быть **валидирован** – **проверен**, например статистически, на **правильность** построения с его помощью прикладных программ.

Поэтому если спецификация является исполняемой, а программа-генератор верифицирована или валидирована, то *проблема верификации прикладных программ исчезает!*

Я спросил мнение **А.М. Миронова с Мехмата МГУ** по поводу этой статьи. Он ответил: «Мое мнение – механизм верификации совсем не раскрыт. Дедуктивные методы (связанные с автоматическим построением доказательств в системах *Coq*, *Isabelle* и т. д.) не упомянуты вовсе». На это я ответил: «С этими методами я знаком, но они не применимы при создании сложных управляющих систем технологическими процессами, тем более что их делают инженеры и программисты, а не математики!». **«Понятно», – безрадостно ответил Александр Михайлович.**

22. Возможность обеспечения горячего резервирования
В ответственных системах для того, что избежать аварий, подобных произошедшей при прилунении израильского космического аппарата «Беришит» (https://habr.com/ru/post/448154/), **необходимо применять горячее резервирование**. Автоматы, особенно с многозначным кодированием состояний **(переменных, кодирующих состояния, столько, сколько автоматов, а не состояний в них!**), идеально подходят для этой цели, позволяя эффективно строить дублированные системы с синхронизацией состояний резервного и основного каналов, так как в этом случае из основного канала в резервный при необходимости надо передать значения лишь небольшого числа переменных, число которых равно числу автоматов в системе.

23. Автоматное программирование при использовании ПЛИС
Графы переходов автоматов (в том числе вложенные) с помощью пакета *Stateflow* (https://en.wikipedia.org/wiki/Stateflow) изображаются на экране. В пакете, в частности, имеется возможность по графам переходов осуществить генерацию программ на одном из ассемблеров ПЛИС, которые и загружаются в программируемую схему. Проводится отладка графов переходов (в пошаговом и/или автоматическом режимах), а не сгенерированных по ним программ. По этой технологии **Ю.Ю. Янкиным в НПО «Аврора»** реализовано программное обеспечение для модулей большого числа систем управления ответственными промышленными объектами.
23.1. Статьи (http://is.ifmo.ru/works/_automata_plis.pdf, http://is.ifmo.ru/works/2014/yankin-control-block.pdf).
23.2. Видео: https://www.youtube.com/watch?v=YNWdmnwHZi8.

24. Применение автоматов для обеспечения верификации *Smart Contracts*
Наши выпускники **Максим Коротков** (http://is.ifmo.ru/projects/sil/) и чемпион мира по программирования 2004 г. **Сергей Оршанский** (http://is.ifmo.ru/works/_2007_09_10_orshanskiy.pdf), которые в свое время прошли через мой курсовой проект по автоматному программированию, предложили **подход к применению конечных автоматов для *Ethereum Smart Contract* в *Blockchain*** (https://maximk.com/files/fa-draft.pdf). Контракты отличный объект для применения автоматов с целью обеспечения верификации, так как их код обычно небольшой и измеряется сотнями строк, а риски при традиционном написании контрактов очень велики.

Большинство контрактов пишется сейчас на тьюринг-полном языке *Solidity*. Авторы предполагали разработать язык автоматного программирования для *Ethereum* с названием *Etherel*, так как в нем должен быть синтаксис, похожий на используемый в языке *Esterel*. Код на таком языке может компилироваться в *Vyper/Solidity* и поддаваться формальной верификации.

Этой же тематике посвящена также работа ***Суворов Д.М., Ульянцев В.И.*** Примеры применения методов синтеза конечных автоматов для генерации моделей смарт-контрактов / Материалы Всероссийской научной конференции по проблемам информатики СПИСОК-2019. СПб.: ВВМ. СПбГУ. 2019, с. 209-214. http://spisok.math.spbu.ru/2019/txt/SPISOK-2019.pdf.

В заключение этой статьи приведу обзорные публицистические статьи по автоматному программированию (https://vk.com/@1077823-obzornye-publicisticheskie-stati-po-avtomatnomu-programmirov): **2003 г.** «Об автоматизации стиральных машин» (https://vk.com/@1077823-ob-avtomatizacii-stiralnyh-mashin), **2005 г.** «Зачем нужны автоматы?»

(https://is.ifmo.ru/download/airplane.pdf), **2005 г.** «Применение конечных автоматов при программировании мобильных устройств» (https://is.ifmo.ru/automata/mobdev/), **2006 г.** «Скромное обаяние автоматного программирования» (https://vk.com/@1077823-skromnoe-obayanie-avtomatnogo-programmirovaniya), **2008 г.** «Автоматное программирование» (https://is.ifmo.ru/works/_2010_09_08_automata_progr.pdf), **2008 г.** «Автоматное программирование, водка и буква Ё» (https://is.ifmo.ru/download/2008-03-17_automata.pdf), **2010 г.** «Тяжелый коврик и автоматное программирование» (https://is.ifmo.ru/belletristic/kovrik), **2011 г.** «Лучше, чем на телевизор» (https://vk.com/@1077823-luchshe-chem-na-televizor), **2013 г.** «О верификации простых программ со сложным поведением» (https://vk.com/@1077823-o-verifikacii-prostyh-programm-so-slozhnym-povedeniem), **2017 г.** «Программы – не стихи, их надо проектировать, а не писать» (https://is.ifmo.ru/main/article_ap.pdf), **2020 г.** «Конечный автомат многим не друг» (https://vk.com/@1077823-konechnyi-avtomat-mnogim-ne-drug), **2021 г.** «Автоматное программирование» (https://vk.com/@1077823-vtomatnoe-programmirovanie), ***2021 г.** «Автоматное программирование. Обзорная статья» (https://computer-museum.ru/articles/tekhnologii-programmirovaniya/4426/), **2021 г.** «Ещё об автоматном программировании» (https://vk.com/@1077823-esche-ob-avtomatnom-programmirovanii), **2022 г.** «О развитии автоматного программирования» (https://vk.com/@1077823-o-razvitii-avtomatnogo-programmirovaniya), **2022 г.** «О приоритете» (https://vk.com/@1077823-o-prioritete), **2023 г.** «Валидация автоматных спецификаций» (https://vk.com/@1077823-validaciya-avtomatnyh-specifikacii), **2023 г.** «Почему в эпоху нейронных сетей для управления ответственными технологическими объектам необходимо применять автоматное программирование» (https://vk.com/@1077823-pochemu-v-epohu-neironnyh-setei-dlya-upravleniya-otvetstvenn), **2023 г.** »Печальная история о применении конечных автоматов в программировании» (https://vk.com/@1077823-pechalnaya-istoriya-o-primenenii-konechnyh-avtomatov-v-progr), «Об объектно-ориентированном и автоматном программировании» (https://vk.com/@1077823-ob-obektno-orientirovannom-i-avtomatnom-programmirovanii), **2024 г.** «Возможно, что автоматное программирование еще не раз пригодится» (https://vk.com/@1077823-vozmozhno-chto-avtomatnoe-programmirovanie-esche-raz-prigodi), **2024 г.** «Автоматное программирование. Не задушишь, не убьешь!» (https://vk.com/@1077823-vtomatnoe-programmirovanie-ne-zadushish-ne-ubesh).

Продолжение этого текста «**Еще об автоматном программировании**» приведено здесь: https://vk.com/@1077823-esche-ob-avtomatnom-programmirovanii. Прочтите также: https://vk.com/@1077823-pochemu-v-epohu-neironnyh-setei-dlya-upravleniya-otvetstvenn.
16.02.2021. https://vk.com/@1077823-vtomatnoe-programmirovanie

Еще об автоматном программировании

Этот текст является дополнением к статье «Автоматное программирование» (https://vk.com/@1077823-vtomatnoe-programmirovanie) **и к приложению 11. В нем отражено развитие этой парадигмы программирования во времени.**

Первый текст про автоматное программирование появился в 1991 г.: ***Шалыто А.А.*** **Программная реализация управляющих автоматов** // Судостроительная промышленность. Серия «Автоматика и телемеханика». 1991. Вып. 13, с. 41, 42, https://is.ifmo.ru/works/switch_prr/.

В **1992 г.** – еще один текст: ***Шалыто А.А.*** Технология программной реализации алгоритмов логического управления как средство повышения живучести кораблей и судов / Тезисы докладов научно-технической конференции «Проблемы обеспечения живучести кораблей и судов». СПб.: Судостроение. 1992, с. 87-89, https://is.ifmo.ru/download/10_02_2008_shalyto1.pdf.

В **1995 г.** на эту тему было две мои публикации: **1**. ***Shalyto A.A.*** Cognitive Properties of Hierarchical Representations of Complex Logical Structures / Proceedings of the 1995 International Symposium on Intelligent Control (ISIC). Workshop. 1995. Monterey. California, p. 391. http://is.ifmo.ru/science/_cognitive_properties_of_hierarchical_representations_of_complex_logic_structures.pdf. **2.** ***Шалыто А.А., Антипов В.В.*** Технология алгоритмизации и программирования задач логического управления // Научно-производственное объединение «Аврора». Юбилейный научно-технический сборник. 1995, с. 162-164.

В **1996 г. я опубликовал двухчастную статью о том, как правильно строить схемы алгоритмов:** ***Шалыто А.А.*** Использование граф-схем и графов переходов при программной реализации алгоритмов логического управления. I, II // Автоматика и телемеханика. 1996. Vol. 57. № 6, с. 148-158, № 7, с. 144-169.
(http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=at&paperid=3235&option_lang=rus, http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=at&paperid=3251&option_lang=rus).

Эта статья опубликована и на английском языке, так как этот журнал переводится: ***Shalyto A.A.*** Algorithmic Graph Schemes and Transition Graphs: Their Use in Software Realization of Logical Control Algorithms. I, II. // Automation and Remote Control. 1996. Vol. 57. No 6, pp. 890-897, No 7, pp. 1027-1045.

В ней предлагается **строить схемы алгоритмов** (этот термин заменил термин «граф-схемы алгоритмов»), **начиная с дешифратора состояний, а не дешифратора входных воздействий, как это делается обычно**. Построенные таким образом схемы изоморфны конструкции *switch* в языках программирования, а схемы, построенные иначе – не изоморфны этой конструкции. ***Если не знать в каком состоянии находится система управления, то какой смысл опрашивать входные переменные?*** Однако многие программисты, почему-то, не хотят на это обращать внимания – видимо, потому что их так не учили. **Такие схемы названы мной «автоматными схемами алгоритмов»**.

В **1997 г.** была еще одна наша публикация: ***Shalyto A., Bagluk U.*** Switch-technology. Algorithmic and Programming Methods in Solution the Logic Control Problems of Shipping Equipment / Proceedings of International Conference on Informatics and Control (ICI&C 97). V.1. St. Petersburg. 1997, pp. 58-60.

Публиковался я тогда мало, так как писал толстую книгу на эту тему без какой-либо надежды издать ее в то мрачное время. Однако, как сказал поэт **Александр Кушнер: «Времена не выбирают, / В них живут и умирают»** (https://m.rupoem.ru/poets/kushner/vremena-ne-vybirayut). И поэтому несмотря ни на что я продолжал её писать...

Прошло несколько лет, и **Аркадий Ключев** – преподаватель кафедры «Вычислительная техника» Университета ИТМО – отметил: «У нас с литературой все плохо (**2000 г**., А.Ш.). Можно перечислить достойные упоминания книги по пальцам. Из особо идейных можно отметить Буча, Страуструпа, Брукса (их, видимо, по ошибке перевели и издали!), с некоторой натяжкой – Рихтера. Ну, еще вышли тощая книжка про *UML* и книжка Йордона... **Книжка Шалыто про автоматы единственная в своем роде за последние лет 15.** В общем – все запущено. Иногда создается впечатление, что у нас в стране только две категории компьютерщиков: чайники и начинающие администраторы сетей, с жутким трудом осваивающие *Perl* и *HTML*. Ах, да, есть еще мода на *Linux*...» (http://is.ifmo.ru/aboutus/1/).

Потом он продолжил: «**Толчком к применению нами конечных автоматов для программирования контроллеров собственного производства послужила брошюра Шалыто** (***Шалыто А.А., Антипов В.В.*** Алгоритмизация и программирование задач логического управления техническими средствами. СПб.: Моринтех, 1996. 90 с.), которую я купил в Доме книги (вариант этой брошюры, опубликованный в 1998 г., приведен здесь: http://is.ifmo.ru/books/alg_log). Она вышла до издания книги Шалыто (***Шалыто А.А.*** *Switch*-технология. Алгоритмизация и программирование задач логического управления**.** СПб.: Наука, 1998. 628 с., http://is.ifmo.ru/books/switch/1).

В классических (советских) книгах по конечным автоматам слабо рассмотрена прикладная область (если это не счетчик в аппаратуре или лексический анализатор в программном обеспечении, А.Ш.), в основном только теория. Несмотря на то, что я в свое время изучал курс «Прикладная теория цифровых автоматов», **мне раньше был не очень понятен способ применения автоматов в программировании** (*мои недоброжелатели и по сей день считают, что в этом вопросе всё всегда было ясно, А.Ш.)***. Никто вокруг не практиковал, да и я сам до этого не додумался**. В настоящее время «*Switch*-технология» практически единственная, доступная книга по автоматам (в программировании, А.Ш.), которую можно рекомендовать студентам...».

Грант Российского фонда фундаментальных исследований на издание этой книги объемом в 40 печатных листов и тиражом 1000 экземпляров я выиграл в **1995 г.** (проект № 96-01-14066).

В **1999 г.** была опубликована статья ***Шалыто А.А.*** *Switch*-технология. Алгоритмизация и программирование задач логического управления // Промышленные АСУ и контроллеры. 1999. № 9, с. 33-37.

В **2000 г.** Министерство образования РФ открыло в Университете ИТМО в качестве **государственного задания** на научно-исследовательскую работу «**Разработка технологии создания программного обеспечения систем управления на основе автоматного подхода»**, которая **продолжалась до 2011 г.** В ходе выполнения работы она изменила название: «**Разработка основных положений создания программных систем управления со сложным поведением на основе объектно-ориентированного и автоматного подходов**». Отчеты по нескольким ее этапам опубликованы по адресу: http://is.ifmo.ru/science/1/. Руководитель – А.А. Шалыто.

В этом же году я **опубликовал еще одну толстую книгу: *Шалыто А.А.* Логическое управление. Методы аппаратной и программной реализации алгоритмов.** СПб.: Наука. 2000, 780 с. http://is.ifmo.ru/books/log_upr/1, а потом еще и четыре статьи на эту тему: **1. *Шалыто А.А.*** Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления // Известия РАН. Теория и системы управления. 2000. № 6, с. 63-81. https://www.avrorasystems.com/ru/Data/Pressroom/Files/ran.pdf. Эта статья на английском: ***Shalyto A.A.*** Software Automation Design: Algorithmization and Programming of Problems of Logical Control // Journal of Computer and Systems Sciences International. 2000. Vol. 39. No 6, pp. 899-916. http://is.ifmo.ru/articles_en/2000/shalyto-switch-2000.pdf. **2. *Шалыто А.А.*** Реализация алгоритмов логического управления программами на языке функциональных блоков // Промышленные АСУ и контроллеры. 2000. № 4, с. 45-50. https://www.avrorasystems.com/ru/Data/Pressroom/Files/asu2.pdf. **3. *Шалыто А.А., Туккель Н.И.*** *Switch*-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Промышленные АСУ и контроллеры. 2000. № 10, с. 44-48. https://www.avrorasystems.com/ru/Data/Pressroom/Files/switch_tech.pdf. **4. *Шалыто А.А., Туккель Н.И.*** Автоматный подход к созданию программного обеспечения для систем логического управления и «реактивных» систем // Системы управления и обработки информации. 2000. Вып. 2, с. 165-173. http://is.ifmo.ru/works/_avtomatnij_podhod_k_sozdaniju_programmnogo_obespechenija.pdf.

В этом же году мой соавтор и коллега по НПО «Аврора» ***Борис Павлович Кузнецов*** опубликовал статью «Психология автоматного программирования» (http://www.softcraft.ru/design/ap/), а несколько позднее высказал свое мнение об этом подходе к программированию (http://is.ifmo.ru/automata/2012/kuznetsov-shalyto.pdf).

2001 г. был у меня весьма продуктивным: **1. *Шалыто А.А.*** Алгоритмизация и программирование для систем логического управления и «реактивных» систем //Автоматика и телемеханика. 2001. № 1, с. 3-39. http://www.mathnet.ru/links/67df370047def9581c5d8713f122c865/at1715.pdf. Эта статья на английском: ***Shalyto A.A.*** Logic Control and «Reactive» Systems: Algorithmization and Programming // Automation and Remote Control. 2001. Vol. 62. No 1, pp. 1-29. http://is.ifmo.ru/articles_en/_log_control.pdf. **2. *Шалыто А.А., Туккель Н.И.*** *Switch*-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5, с. 45-62. http://is.ifmo.ru/download/switch.pdf. Эта статья на английском: ***Shalyto A.A., Tukkel N.I.*** Switch-technology: An Automated Approach to Developing Software for Reactive Systems // Programming and Computer Software. 2001. Vol. 27. No 5, pp. 260-276. https://dl.acm.org/citation.cfm?id=597470. **3. *Шалыто А.А., Туккель Н.И.*** *Switch*-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Известия высших учебных заведений. Приборостроение. 2001. № 9, с. 28-35. **4. *Туккель Н.И., Шалыто А.А.*** Реализация вычислительных алгоритмов на основе автоматного подхода // Телекоммуникации и информатизация образования. 2001. № 6, с. 35-53. http://is.ifmo.ru/progeny/_2011_01_03_shalyto-tukkel.pdf. **5. *Шалыто А.А., Туккель Н.И.*** Программирование с явным выделением состояний. Части 1, 2 // Мир ПК. 2001. № 8, с. 116-121; № 9, с. 132-138. http://is.ifmo.ru/works/mirk/.

В этом же году я и **Никита Туккель** познакомились с **Сергеем Александровичем Вагановым** на «почве» введения автоматов в разработанную им среду ***FLORA/C+*** (http://is.ifmo.ru/automata/vaganov/):
Туккель Н.И., Шалыто А.А., Ваганов С.А. Использование *Switch*-технологии при разработке программ в среде *FLORA/C+* (модель технологического процесса в цехе холодной прокатки) (http://is.ifmo.ru/projects/cold/).

В **2001 г.** Никитой и мною был выполнен проект «**Система управления танком для игры *Robocode*. Вариант 1. Объектно-ориентированное программирование с явным выделением состояний. Проектная документация**» (http://is.ifmo.ru/projects/tanks/).

В этом же году вышла рецензия на мою последнюю книгу: ***Богатырев Р.*** Об автоматном и асинхронном программировании // Открытые системы. 2001. № 3, с. 68, 69 (http://is.ifmo.ru/recensions/bogatyrev/), в которой, в частности, сказано: «Книга затрагивает не просто область конечных автоматов, а куда более важные вопросы единства математических основ аппаратуры и программного обеспечения, а также проблемы доказательного и автоматического программирования, которое предусматривает автоматический синтез программ на основе формальной спецификации и базы знаний предметной области».

На следующий год мы выполнили проект: ***Туккель Н.И., Шалыто А.А.*** Система управления дизель-генератором (фрагмент). Программирование с явным выделением состояний. **Проектная документация** (http://is.ifmo.ru/projects/dg/). **2002**, а Российский фонд фундаментальных исследований в результате нашей победы на конкурсе открыл на **2002, 2003 гг.** тему: «**Разработка технологии автоматного программирования**». Руководитель – А.А. Шалыто.

В **2002 г.** в одиннадцатом номере журнала «Магия ПК» появилась статья ***Озерова А.А.*** Четыре танкиста и компьютер (еще раз об игре *Robocod*e), http://is.ifmo.ru/?i0=aboutus&i1=5, в которой есть такие слова: «В Интернете существует достаточно много сайтов, посвященных программированию и реализации своих собственных танков. Так, например, на сайте компании *IBM* представлен целый раздел с описанием различных тактик и методов ведения танковой войны. Среди них можно найти несколько полезных советов, помогающих реализовать систему движения или сканирования поля битвы на предмет других танков. Однако наибольший интерес вызвала российская реализация танка с использованием ***Switch*-технологии** – парадигмы ***программирования,* основанной на применении конечных автоматов** (уже тогда некоторые специалисты **называли предложенный мною подход парадигмой программирования**, А.Ш.). Эта реализация была предложена сотрудниками СПбГИТМО Н.И. Туккелем и А.А. Шалыто. Они на основе теории конечных автоматов создали модель танка, который запросто обыгрывает многих участников лиги роботов. Точная математическая модель и использование современных методик программирования, позволили российскому танку занять достойное место в лиге *Robocode*. Этот танк обладает **тремя особенностями, делающими его уникальным среди остальных: он построен «по науке», программа входит в состав проектной документации** (http://is.ifmo.ru/projects/tanks/, http://is.ifmo.ru/projects/robocode2/), **и при всем этом он еще и «неплохо» играет».**

А вот, что Озеров написал мне: «Спасибо за отличный пример использования *Switch*-технологии в таком нетривиальном контексте как компьютерная игра-симулятор. Я не мог не упомянуть Ваш труд в рамках своей статьи, так как **считаю предлагаемый подход очень толковым и оригинальным.** Также мне симпатичны и другие Ваши работы на http://is.ifmo.ru«.

Публикация статьи с Вагановым имела последействие – выпускник МИФИ **Александр Головешин в 2002 г.** написал мне, что разработал **под предложенную в нашей статье в журнале «Программирование» нотацию графов переходов** инструментальное средство для поддержки автоматного программирования ***Visio2Switch*,** но он прислал только *exe*-файл этого средства (http://is.ifmo.ru/progeny/visio2switch/). На мой вопрос «является ли он программистом», Александр ответил, что «написал в жизни только две программы – астрономическую и эту».

До этого **он использовал в программах флаги**, и ему «**надоело по-убогому ваять программы для железяк**. Каждый раз все держится на тонкой грани – достаточно что-то изменить или добавить, и все приходится переписывать» (http://is.ifmo.ru/aboutus/1/). Такие программы ему не нравились, так как они при практически каждом изменении падали. Он прочел одну из наших статей и разработал конвертер ***Visio2Switch***, который прислал мне, чтобы я выложил его в открытый доступ, что было весьма затруднительно, так как своего сайта у меня тогда еще не было, но потом усилиями **Саши Наумова** он появился, и мы выложили этот конвертер (http://is.ifmo.ru/automata/visio2switch/). Конвертор также опубликован по адресу: http://www.softcraft.ru/auto/switch/v2s/.

В дальнейшем **я стал сравнивать программы с флагами со «слонами на тонких ножках», изображенными С. Дали на картине «Искушение Святого Антония»** (http://www.arthistory.ru/dali2.htm). При этом я всегда отмечаю, что **такие слоны уникальны –**

встречаются только на этой картине, а программы с флагами применяются повсеместно, обладая устойчивостью :-) **этих слонов!** Однако, по мнению Вуди Аллена, тонкие ножки присущи не только программам и слонам: «Профессор спасался от здоровенного мохнатого неправильного глагола *tener* (иметь), гонявшегося за ним на длинных тонких ножках». В общем, «в тонких ножках» в отличие от состояний, нет ничего хорошего…

А вот, что **в том же году** написал еще один тогда неизвестный мне человек: ***Вавилов К.*** Программирование за... 1 (одну) минуту // Компьютер Price. 2002. № 31, с. 288-293 (http://is.ifmo.ru/automata/1minute/): «Ни с чем не сравнимое чувство возникает, когда ты точно и сразу знаешь (**на основе протоколов, автоматически строящихся в терминах автоматов**, А.Ш.) место и условия возникновения логической ошибки». Как потом выяснилось, что Вавилову тогда было 32 года, и ему, видимо, уже было с чем сравнивать :-). Он написал также: «Вы были у нас в «ТЯЖПРОМЭЛЕКТРОПРОЕКТЕ» и прочитали лекцию. Так, что **к автоматам я пришел только через Вас**». Казалось бы, легко было прийти к этому, как считают мои недоброжелатели, и без меня...

В то же время некто **Алексей Перро** прислал мне такое письмо: «**С тех пор, как разобрался с конечными автоматами, я уверен, что любой сложности задачу (в известных рамках) смогу реализовать быстро, правильно, а главное – с первого раза**, а не проводить бессонные ночи за отладчиком, **тщетно пытаясь увеличить объем мозга, для того чтобы запомнить все**».

Примерно тогда же я узнал и такую историю: «Я использовал *Switch*-технологию для автоматизации процесса безразборной мойки. Пока не изучил ее, не знал даже как подойти к реализации этой весьма сложной программы. Она написана под *PLC Sattcon OP45* на языке *DOX 5*. **Основываясь на Вашей технологии, я разработал схему преобразования автоматов в программу, и все пошло на ура**. Программа построена по 22 графам переходов автоматов, связанных по вложенности и обмену номерами состояний, которые изображены в *MS Visio* с помощью шаблона (http://is.ifmo.ru/progeny/visio2switch/), взятого на Вашем сайте. Автоматы имеют следующее число состояний – 3,3,4,4,4,4,4,4,5,5,5,5,5,6,6,7,8,8,8,10,14,15. Общее число состояний – 147. Когда я до этого **консультировался во многих фирмах, все утверждали, что *такого этапа как проектирование программ для PLC нет*, и мол «надо программы на них уметь писать», но меня это не устраивало.** О сроках выполнения проекта: полмесяца изучал технологический процесс и выбирал технологию построения программы; полмесяца изучал *Switch*-технологию (материалы Вашего сайта); полмесяца создавал первый проект автоматной программы; полмесяца изучал возможности *DOX5* для реализации и придумывал правила кодирования; один месяц экспериментальных пусков на установке, доработка самой установки, автоматов и программы; два месяца опытной эксплуатации, проводимой под моим контролем. И все! С уважением, инженер-электроник Группы предприятий *Parmalat* Россия ОАО «Белгородский молочный комбинат», **Магомедов Анатолий Анатольевич**, tolick-list@inbox.ru» (http://is.ifmo.ru/aboutus/33/).

В **2002 г.** было опубликовано пять статей с моим участием**: 1. *Шалыто А.А., Туккель Н.И.*** Преобразование итеративных алгоритмов в автоматные // Программирование. 2002. № 5, с. 12-26. http://is.ifmo.ru/works/iter/. Эта статья на английском: ***Shalyto A.A., Tukkel N.I.*** Translating Iterative Algorithms into Automation Ones // Programming and Computer Software. 2002. Vol. 28, No 5, pp. 250-260. https://link.springer.com/article/10.1023/A:1020208127964. **2. *Туккель Н.И., Шалыто А.А., Шамгунов Н.Н***. Реализация рекурсивных алгоритмов на основе автоматного подхода // Телекоммуникации и информатизация образования. 2002. № 5, с. 72-99. http://is.ifmo.ru/works/recurse/. **3. *Шалыто А.А., Туккель Н.И.*** От тьюрингова программирования к автоматному // Мир ПК. 2002. № 2, с. 144-149. https://is.ifmo.ru/download/turing.pdf. **4. *Туккель Н.И., Шалыто А.А.*** Проектирование программного обеспечения системы управления дизель-генераторами на основе автоматного подхода // Системы управления и обработки информации. 2002. Вып. 5, с. 66-82. http://is.ifmo.ru/works/diesel/. **5. *Туккель Н.И., Шалыто А.А.*** Реализация автоматов при программировании событийных систем // Программист. 2002. № 4, с. 74-80. http://is.ifmo.ru/download/evsys.pdf.

В **2003 г.** призер чемпионатов мира по программированию 2000 и 2001 гг. наш студент **Денис Кузнецов** вел кружок для старшеклассников во Дворце пионеров в Санкт-Петербурге. Летом он выбирал достаточно сложный проект, который было бы интересно делать детям. Для этой цели он выбрал игру *Robocod*e. Потом он увидел на моем сайте нашу с Туккелем работу про танки, и на ее основе, готовясь к преподаванию школьникам, сделал по моему курсу проект «**Система управления танком для игры *Robocod*e. Вариант 2**» (http://is.ifmo.ru/projects/robocode2/). В

аннотации к этому проекту **Денис пишет**: «Данная работа является развитием проекта ***Шалыто А.А., Туккеля Н.И.*** Система управления танком для игры *Robocode*. Вариант 1. Объектно-ориентированное программирование с явным выделением состояний (http://is.ifmo.ru/projects/tanks/). **Хорошая проектная документация по указанному проекту позволила весьма просто внести следующие изменения**: **1.** Выполнен переход от процедурного программирования с использованием классов к более полному применению объектно-ориентированного программирования. **2.** Осуществлено отделение автоматов управления от управляемых объектов в коде. **3.** Учтены последние изменения в интерфейсе среды *Robocode*, что, в свою очередь, позволило упростить логику программы и сократить число автоматов на один. **4.** Улучшена программная документация проекта, как за счет усовершенствования самодокументируемости кода, так и за счет использования средств *javadoc*. **Приведена *проектная документация*, которая, наряду с *программной документацией*, дает полное представление о решении задачи построения системы управления танком**».

В этом же году большим тиражем была опубликована моя статья «**Технология автоматного программирования**» (https://www.osp.ru/pcworld/2003/10/166609).

В 2003 г. в книге *Окулова С.М.* Конгнитивная информатика**.** Киров: ВятГГУ, 2003 было сказано: «... абсолютно неоднозначно то, что общепринятая на сегодня парадигма развития технологий программирования является наиболее эффективной. **Альтернативные варианты предлагаются в настоящее время, например, в работах Анатолия Абрамовича Шалыто** – доктора технических наук, профессора, **автора автоматной технологии программирования**».

В этом же году была опубликована книга ***Непейвода Н.Н., Скопин И.Н.*** Основания программирования. Ижевск-Москва: РХД, 2003, в которой было введено понятие «**стиль программирования**». Среди стилей программирования авторами был выделен и такой, который они с ссылкой на мою книгу назвали «**программирование от состояний**» (http://is.ifmo.ru/aboutus/_log_prog2.pdf).

В **2003 г.** этой тематике было опубликовано девять текстов с моим участием: **1.** ***Naumov L., Shalyto A.*** Automata Theory for Multi-Agent Systems Implementation / Proceedings of International Conference Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering. (KIMAS-03). Boston: IEEE Boston Section. 2003, pp. 65-70. https://www.academia.edu/31854746/Automata_theory_for_multi-agent_systems_implementation. **2.** ***Шопырин Д.Г., Шалыто А.А.*** Объектно-ориентированный подход к автоматному программированию // Информационно-управляющие системы. 2003. № 5, с. 29-39. http://www.i-us.ru/index.php/ius/article/view/14376. **3.** ***Наумов Л.А., Шалыто А.А.*** Искусство программирования лифта. Объектно-ориентированное программирование с явным выделением состояний // Информационно-управляющие системы. 2003. № 6, с. 38-49. http://www.i-us.ru/index.php/ius/article/view/14406. **4.** ***Шалыто А.А., Туккель Н.И.*** Автоматное и синхронное программирование // Искусственный интеллект. 2003. № 4, с. 82-88. http://iai.dn.ua/public/JournalAI_2003_4/Razdel1/12_Shalyto_Tukkel%27.pdf. **5.** ***Shalyto A.A., Naumov L.A.*** Automata Programming as a Synchronous Programming / Proceedings of the «East-West Design & Test Conference» (EWDTC-03). IEEE Ukrainian Department. Yalta: Kharkov National University of Radio-electronics. 2003, p. 140-143. https://cyberleninka.ru/article/n/automata-programming-as-a-sort-of-synchronous-programming/viewer. **6.** ***Шалыто А.А.*** **Технология автоматного программирования** // Сборник научных статей «Современные технологии». СПбГУ ИТМО 2003, с. 18-26. https://www.elibrary.ru/item.asp?id=32370471. **7.** ***Шалыто А.А***. Технология автоматного программирования // Мир ПК. 2003. № 10, с. 74-78. http://is.ifmo.ru/works/tech_aut_prog. **8.** ***Шалыто А.А.*** **Технология автоматного программирования** / Материалы Всероссийской научно-технической конференции «Методы и средства обработки информации» (МСО-2003). МГУ. 2003, с. 528-535. http://is.ifmo.ru/works/tech_aut_prog. **9.** ***Мазин М.А., Парфенов В.Г., Шалыто А.А.*** Анимация. *FLASH*-технология. Автоматы // Компьютерные инструменты в образовании. 2003. № 4, с. 39-47. http://is.ifmo.ru/download/flash.pdf.

В **2004 г.** появился первый релиз инструментального средства для поддержки автоматного программирования ***UniMod*** (https://unimod.sourceforge.io, https://www.youtube.com/watch?v=Y4et51dz-HE). Публикации по этой тематике начались со статьи: ***Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто***

A.A. UML. *Switch*-технология. *Eclipse* // Информационно-управляющие системы. 2004. № 6, с. 12-17. http://www.i-us.ru/index.php/ius/article/view/14489.

Это средство в дальнейшем использовалось не только в учебном процессе в Университете ИТМО (http://is.ifmo.ru/unimod-projects/), но и в Италии (https://sepl.dibris.unige.it/publications/2012-ricca-MiSE.pdf). Потом мы получили свидетельства о государственной регистрации программ для ЭВМ с такими названиями: **«Ядро автоматного программирования»** (http://is.ifmo.ru/unimod/_svid.pdf) **и «Встраиваемый модуль автоматного программирования для среды разработки *Eclipse*»** (http://is.ifmo.ru/unimod/_svid2.pdf). У некоторых программистов *UniMod* стал любимым инструментом (https://biese.wordpress.com/2007/02/06/using-finite-state-machine-tools-to-solve-the-probelm/).

В **2004 г.** при моем участии были опубликованы еще две статьи по рассматриваемой тематике: **1. *Шопырин Д.Г., Шалыто А.А.*** Синхронное программирование // Информационно-управляющие системы. 2004. № 3, с. 35-42. http://www.i-us.ru/index.php/ius/article/view/14452. **2. *Корнеев Г.А., Шамгунов Н.Н., Шалыто А.А.*** Паттерн *State Machine* для объектно-ориентированного проектирования автоматов // Информационно-управляющие системы. 2004. № 5, с. 13-25. http://www.i-us.ru/index.php/ius/article/view/14476.

В этом же году ***Игорь Одинцов*** с матмеха СПбГУ в книге **Профессиональное программирование. Системный подход.** БХВ-Петербург. 2004 (https://www.studmed.ru/view/odincov-i-professionalnoe-programmirovanie-sistemnyy-podhod_cd4f159c838.html?page=1), рассказывая не о самых популярных методологиях программирования, **начинает с методологии автоматного программирования и ссылается на меня (с. 106)**. Я также упоминаюсь там и в связи с инициативой об открытой проектной документации (с. 597). Приведенные на моем сайте is.ifmo.ru примеры открытой проектной документации названы удачными (с. 598).

В **2005 г.** я опубликовал две статьи: **1. *Шалыто А.А.* Автоматно-ориентированное программирование** // Научно-технический вестник СПбГУ ИТМО. Актуальные проблемы современных оптико-информационных систем и технологий. 2005. № 5 (21) с. 35-41. https://ntv.ifmo.ru/ru/journal/102/journal_102.htm. **2. *Шалыто А.А.* Автоматно-ориентированное программирование** / Материалы IX Всероссийской конференции по проблемам науки и высшей школы «Фундаментальные исследования в технических университетах». СПб.: Политех. 2005, с. 44-52. http://is.ifmo.ru/works/_politeh.pdf.

В этом же году **Н.Н. Непейвода** выпустил книгу **Стили и методы программирования**. М.: Интернет-Университет Информационных технологий. 2005. 316 с., в которой ***автоматное программирование рассматривается как стиль программирования***. При этом **в качестве ключевых слов к главе «Автоматное программирование»** используются следующие термины: ***А.А. Шалыто****, таблица состояний и переходов, состояние, переход, автомат Мура, автомат Мили,* ***автоматное программирование****, блок-схема*.

Там же Николай Николаевич написал: «**Термин «автоматное программирование» принадлежит, насколько нам известно, А. Шалыто**. Во всяком случае, ***ему принадлежит заслуга в его развитии вопреки моде и мнению большинства***». Все это осталось и в изданиях книги 2012 и 2016 гг.

На основе этой книги на портале «ИНТУИТ. Национальный открытый университет» Непейвода опубликовал **четыре лекции** (с девятой по двенадцатую) **по автоматному программированию**, причем в аннотации к первой из них приведены указанные выше слова о мнении большинства (http://www.intuit.ru/studies/courses/40/40/lecture/1198).

В 2005, 2006 гг. в рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям науки и техники» на 2002-2006 годы мы выиграли государственный контракт на выполнение опытно-конструкторской работы на тему: «**Технология автоматного программирования: применение и инструментальные средства**». Руководитель – А.А. Шалыто. Работа вошла в число 15 наиболее перспективных научных проектов, которые находились в 2005/2006 гг. в распоряжении Федерального агентства по науке и инновациям (*Коммерсантъ Business Guide*. 2005. № 215, с. 36. http://www.kommersant.ru/doc/625381).

В 2005 г. ***Вавилов К.В.*** выполнил три работы: Программируемые логические контроллеры *SIMATIC S7-200* (*SIEMENS*). Методика алгоритмизации и программирования задач логического управления. 2005. http://is.ifmo.ru/automata/_metod065.pdf; Контроллеры *SIMATIC S7-300* (*SIEMENS*). Организация взаимодействия независимых локальных систем управления на основе автоматного подхода и функционального разделения автоматов управления. 2005. http://is.ifmo.ru/automata/_s7300.pdf; *LabVIEW* и *Switch*-технология. Методика алгоритмизации и программирования задач логического управления. 2005. http://is.ifmo.ru/automata/_vavilov2.pdf.zip.

В этом же году в сети появился текст «**Автоматное программирование для начинающих**» (https://tdocs.su/4199), а на странице «Олимпиадное программирование на Физтехе» (http://acm.mipt.ru/twiki/bin/view/Method/StylesSamples) **среди парадигм программирования приводится и «автоматная»**. Однако до всеобщего признания этой парадигмы еще далеко. Например, в учебном пособии ***Городняя Л.В.*** Пададигма программирования. СПб.: Лань. **2021**, 232 с., она даже не упомянута (https://e.lanbook.com/reader/book/151660/#1). Судя по названию :-), она и не должна быть упомянута, так как в названии слово «парадигма» используется в единственном числе, и вряд ли эта единственная парадигма – автоматная :-).

В **2005 г.** появилась статья ***Козаченко В.Ф.*** Эффективный метод программной реализации дискретных управляющих автоматов в встроенных системах управления (http://www.l-avt.ru/support/library/articles/state_mashine.pdf). Вот, что в это время Владимир Филиппович написал мне: «Мы с 1990 г. занимаемся разработкой микроконтроллерных систем управления для отечественных комплектных электроприводов с различными типами исполнительных двигателей, а также для преобразователей частоты (ПЧ) и различных источников питания. При этом управление как режимами работы собственно ПЧ, так и многонасосными рабочими станциями на их основе, реализовано на принципах, близких к Вашим.

Интересно, как мы впервые познакомились с Вашими работами в этой области. Я вел курсы повышения квалификации специалистов в области встраиваемых микроконтроллерных систем управления и читал лекцию о современных методах реализации программ для логических контроллеров и дискретных управляющих автоматов на сигнальных процессорах. Один из слушателей сказал, так это же **очень похоже на то, как делает А.А. Шалыто**, и показал Вашу статью. Мы были поражены схожестью подходов. Конечно, у Вас все более научно, а у нас приближено к нашим **практическим задачам с сотней прерываний** и прямым процессорным управлением силовыми элементами и прямым сопряжением с датчиками. **Для нас управляющие автоматы – это очень важная часть общего ПО**, которая должна работать предельно эффективно, организуя как бы верхний уровень программного обеспечения. Наши скромные возможности по технике отражены на сайте www.motorcontrol.ru. Директор Учебно-научно-консультационного центра «*Texas Instruments*-МЭИ», Генеральный директор «Научно-производственной фирмы «ВЕКТОР» ***Козаченко В.Ф.***».

В **2005 г.** в журнале «Радиолюбитель» **А. Черномырдин** опубликовал серию статей о применении автоматного программирования для микроконтроллеров (http://is.ifmo.ru/automata/_autmicroc.pdf, http://is.ifmo.ru/automata/_autmicroc2.pdf).

В этом же году в журнале *RSDN Magazine* была опубликована статья «**Реализация систем, управляемых событиями. Использование конечных автоматов**» (https://rsdn.org/article/patterns/Protocols.xml). Ее авторы, видимо, очень «гордые» – ни на кого не сослались...

В **2005 г. Сергеем Канжелевым** и мною было предложено инструментальное средство ***MetaAuto*** для автоматической генерации автоматных программ на любом априори заданном языке программирования по графам переходов (http://is.ifmo.ru/projects/metaauto/). Предложенный подход к генерации программ описан в статье: ***Канжелев С.Ю., Шалыто А.А.*** Автоматическая генерация автоматного кода // Информационно-управляющие системы. 2006. № 6, с. 35-42. http://is.ifmo.ru/works/_autogen.pdf. Презентация этого средства размещена по адресу: http://www.myshared.ru/slide/128477/. **Оно используется в АО «Концерн НПО «Аврора» моим аспирантом А.В. Калачинским в его технологии автоматного программирования судовых систем управления**.

А вот другие наши статьи, опубликованные по теме в этом году: **1.** ***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 Boston Section. 2005, pp. 449-453. http://swan.kgeorgiy.info/papers/Yartsev_B__Korneev_G__Shalyto_A__Kotov_V_--_Automata-Based_Programming.pdf. **2.** ***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 Boston Section. 2005, pp. 460-465. http://www.kgeorgiy.info/papers/Shalyto_A__Naumov_L__Korneev_G_--_OO_FSA.pdf. 3**.** ***Кретинин А.В., Солдатов Д.В., Шалыто А.А., Шостак А.В.*** Ракеты. Автоматы. Нейронные сети // Нейрокомпьютеры: разработка и применение. 2005. № 5, с. 50-59 http://is.ifmo.ru/works/_rocketaut.pdf. **4.** ***Кретинин А.В., Солдатов Д.В., Шалыто А.А., Шостак А.В.*** Использование нейросетевых конечных автоматов для моделирования функционирования агрегатов жидкостного ракетного двигателя // Информационные технологии. 2005. № 8, с. 47-53. **5.** ***Альтерман И.З., Шалыто А.А.*** Формальные методы программирования логических контроллеров // Промышленные АСУ и контроллеры. 2005. № 10, с. 49-52. http://is.ifmo.ru/works/formalcontroller.pdf. **6.** ***Корнеев Г.А., Шамгунов Н.Н., Шалыто А.А.*** *State Machine* – расширение языка *Java* для эффективной реализации автоматов // Информационно-управляющие системы. 2005. № 1, с. 16-24. http://www.i-us.ru/index.php/ius/article/view/14503. **7.** ***Шопырин Д.Г.*** Объектно-ориентированная реализация конечных автоматов на основе виртуальных методов // Информационно-управляющие системы. 2005. № 3, с. 36-40. http://www.i-us.ru/index.php/ius/article/view/14533.

В **2005 г.** автоматное программирование и *UniMod* обсуждаются в статье: ***Новиков Ф.А.*** Визуальное конструирование программ // Информационно-управляющие системы. 2005. № 6, с. 9-22. http://www.i-us.ru/index.php/ius/article/view/14557.

В этом же году был опубликован стандарт для построения распределенных систем управления и автоматизации *IEC 61499* (https://ru.wikipedia.org/wiki/IEC_61499), в котором поведение базовых блоков предложено описывать графами переходов, чего не было в дополняемым им стандарте *IEC 61131-3* (https://ru.wikipedia.org/wiki/IEC_61131-3), описывающем языки программирования программируемых логических контроллеров, первая редакция которого была разработана в 1993 г. При этом отмечу, что Виктор Николаевич Дубинин защитил докторскую диссертацию по этой тематике (https://science.pnzgu.ru/files/science.pnzgu.ru/science.pnzgu.ru/dissertaciya_dubinina_v_n_.pdf) в 2014 г. Книги на русском языке по этой теме размещены здесь: https://vt.pnzgu.ru/files/vt.pnzgu.ru/sotrudniki/dubinin/dlya_rezyume/monografiya/fb_monography.pdf, https://www.gov.kz/uploads/2020/10/7/37c2918c51d61ec3ab8cb6a849771402_original.13380269.pdf. Есть основания предполагать, что «новый» стандарт станет определяющим при автоматизации в промышленности (https://www.eclipse.org/4diac/), особенно в нефтяной, хотя нотация, применяемая в этом стандарте, по моему мнению, громоздка и не наглядна.

В **2006 г.** я опубликовал статью **Автоматное программирование** // Известия Уральского государственного университета. 2006. № 43. Компьютерные науки и информационные технологии. Вып. 1, с. 181-190. http://elar.urfu.ru/bitstream/10995/24543/1/iurm-2006-43-13.pdf.

Другие статьи, написанные при моем участии в этом году: **1.** ***Shalyto A., Shamgunov N., Korneev G.*** 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, pp. 51-58. http://is.ifmo.ru/articles_en/2006/shalyto-shamgunov-korneev-2006.pdf. **2.** ***Paraschenko D., Shalyto A., Tsarev F.*** 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 Applications (CIMSA-2006). La Coruna. Spain. 2006, pp. 15-20. https://www.academia.edu/31854706/Modeling_Technology_for_One_Class_of_Multi-Agent_Systems_with_Automata_Based_Programming. **3.** ***Шалыто А.А.*** О проекте «Технология автоматного программирования: применение и инструментальные средства» // Информационные технологии. 2006. № 2, с. 79. **4.** ***Вавилов К.В., Шалыто А.А.*** *LabVIEW* и *Switch*-технология // Промышленные АСУ и контроллеры. 2006. № 6, с. 43-45. http://is.ifmo.ru/works/_lv.pdf. **5.** ***Оршанский С.А., Шалыто А.А.*** Применение динамического программирования при решении

задач на конечных автоматах // Компьютерные инструменты в образовании. 2006. № 4, с. 26-35. http://is.ifmo.ru/works/_2007_09_10_orshanskiy.pdf.

В этом же году было издано учебно-методическое пособие по этой тематике: ***Зюбин В.*** Программирование информационно-управляющих систем на основе конечных автоматов. Новосибирский государственный университет, 2006 (http://reflex-language.narod.ru/articles/06ICSonFA.pdf, http://www.softcraft.ru/auto/other/mpz/mpz.pdf), в котором на странице 58 обсуждается *Switch*-технология. Мое письмо к Владимиру после прочтения его пособия приведено здесь: http://is.ifmo.ru/books/shalyto-zubinu.

В **2006 г.** вышла книга ***Салмре И.*** Программирование мобильных устройств на платформе *.Net Compact Framework***.** М.: Вильямс. 2006, в которой в главе «**Наш друг конечный автомат**» «звучит **гимн» применению конечных автоматов при программировании мобильных устройств** (http://is.ifmo.ru/automata/mobdev/).

В этом же году в журнале «Компоненты и технологии» № 8 была опубликована статья ***Татарчевского В.*** Некоторые мысли по поводу программирования встроенных систем (http://is.ifmo.ru/automata/_19_60.pdf), в которой **обсуждались наши публикации по применению автоматного программирования**. Потом появилась еще одна его статья по этой тематике: ***Татарчевский В.А. Switch*-технология в задачах логического управления** // Программные продукты и системы. 2006. № 4, с. 30-32 (http://is.ifmo.ru/works/_Tatarch.pdf, http://www.swsys.ru/index.php?page=article&id=441), в которой я получил благодарность от автора за помощь в ее подготовке. После этого вышла статья ***Татарчевский В.А.*** Применение *Switch*-технологии в задачах управления технологическими процессами // Надежность. 2007. № 1, с. 21-27 (http://is.ifmo.ru/works/_ALL.pdf). А еще он в 2006, 2007 гг. опубликовал в журнале «Компоненты и технологии» несколько статей под общим названием «Применение *Switch*-технологии при разработке прикладного программного обеспечения для микроконтроллеров» (https://disk.yandex.ru/d/89YWXdYIdMy42).

Затем со ссылками на Татарчевского появилась статья В. Васильева с важным для программистов словосочетанием: «**Конечные автоматы, как программировать без запарок**» (http://popayaem.ru/konechnye-avtomaty-kak-programmirovat-bez-zaparok.html). В ней автор пишет: «**Речь пойдет о таком интересном *стиле программирования* микроконтроллеров как автоматное программирование**. Точнее **это даже не стиль программирования, а целая концепция**, благодаря которой программист микроконтроллеров может значительно облегчить себе жизнь. Благодаря ее применения многие задачи, которые встают перед программистом, решаются гораздо легче и проще, **избавляя программиста от головной боли**. Кстати, **автоматное программирование часто называют *Switch*-технологией**».

В **2006, 2007 гг. на Физтехе** при подготовке к олимпиадам по программированию читался курс «Сравнительный анализ языков программирования» (http://acm.mipt.ru/twiki/bin/view/Method/WebHome), в программе которого в разделе «**Парадигмы программирования**» обсуждалось «**Программирование от состояний – автоматное программирование**» (http://acm.mipt.ru/twiki/bin/view/Method/MainProgram).

В **2007 г.** мы опубликовали в СПбГУ ИТМО учебно-методическое пособие: ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование (http://is.ifmo.ru/books/_umk.pdf), а ***Генельт А.Е.*** – учебное пособие «**Автоматизированные методы разработки архитектуры программного обеспечения**» (http://is.ifmo.ru/books/_henelt2.pdf), в котором есть раздел «Автоматное программирование».

А вот статьи с моим участием, опубликованные в **2007 г.**: **1.** ***Шопырин Д.Г., Шалыто А.А.*** Графическая нотация наследования автоматных классов // Программирование. 2007. № 5, с. 62-74. http://is.ifmo.ru/works/_12_12_2007_shopyrin.pdf. Эта же статья на английском: ***Shopyrin D., Shalyto A.*** Graphical Inheritance Notation for State-Based Classes // Programming and Computer Software. 2007. Vol. 33. No 5, pp. 283-292. http://is.ifmo.ru/articles_en/_2007_09_03_PCS283.pdf. **2.** ***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, с. 127-136. Эта статья на английском: ***Lobanov P.G., Shalyto A.A.*** Application of Genetic Algorithms for Automatic Construction of Finite-State Automata in the Problem of Flibs // Journal of

Computer and Systems Sciences International. 2007. Vol. 46. No 5, pp. 792-801. http://is.ifmo.ru/articles_en/_lobanov.pdf. **3. *Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А.*** Инструментальное средство для поддержки автоматного программирования // Программирование. 2007. № 6, с. 65-80. https://www.academia.edu/31854647/Tools_for_support_of_automata-based_programming, http://is.ifmo.ru/works/_2008_01_27_gurov.pdf. Эта статья на английском: ***Gurov V.S., Mazin M.A., Narvsky A.S., Shalyto A.A.*** Tools for Support of Automata-Based Programming // Programming and Computer Software. 2007. Vol. 33. No 6, pp. 343-355. https://www.researchgate.net/publication/220203978_Tools_for_support_of_automata-based_programming. **4. *Кретинин А.В., Солдатов Д.В., Шалыто А.А., Шостак А.В.*** Диагностирование аварийных состояний турбонасосного агрегата жидкостного ракетного двигателя // Нейрокомпьютеры: разработка, применение. 2007. № 9, с. 372-379. http://is.ifmo.ru/works/_kshh.pdf. **5. *Вельдер С.Э., Шалыто А.А.*** О верификации простых автоматных программ на основе метода *Model Checking* // Информационно-управляющие системы. 2007. № 3, с. 27-38. http://www.i-us.ru/index.php/ius/article/view/14670. **6. *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. http://www.i-us.ru/index.php/ius/article/view/14683. **7. *Вавилов К.В., Шалыто А.А.*** Что плохого в неавтоматном подходе к программированию контроллеров // Промышленные АСУ и контроллеры? 2007. № 1, с. 49-51. http://is.ifmo.ru/works/_Asu-2007-01.pdf.

В **2007 г.** во втором, третьем и четвертом номерах журнала «Компоненты и технологии» были опубликованы четвертая (с. 148-150), пятая (с. 180-182) и шестая (с. 202-204) части статьи ***Татарчевского В.*** Применение *Switch*-технологии при разработке программного обеспечения для микроконтроллеров.

В **2007, 2008 гг.** рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2012 годы». «Проведение проблемно-ориентированных поисковых исследований и создание научно-технического задела по перспективным технологиям в области информационно-телекоммуникационных систем» мы выиграли два конкурса на выполнение работ по темам: «**Технология генетического программирования для генерации автоматов управления системами со сложным поведением**» и «**Разработка технологии верификации управляющих программ со сложным поведением, построенных на основе автоматного подхода**». Руководитель – А.А. Шалыто.

В **2008 г.** мы подготовили и передали в издательство «Питер» рукопись книги ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование, оформленную по предложенному нам шаблону (http://is.ifmo.ru/books/_book.pdf).

В этом году мы впервые в мире опубликовали сборник по автоматному программированию: Научно-технический вестник СПбГУ ИТМО. 2008. № 8 (53). **Автоматное программирование** (http://bourabai.ru/library/automat53.pdf)**, содержащий 314 страниц, в котором приведены 28 (!) статей** (http://is.ifmo.ru/works/) **по различным аспектам автоматного программирования**. Сборник начинается с большой моей статьи, названной «**Парадигма автоматного программирования**». Мне кажется, что она хороша для первоначального **знакомства с этой парадигмой программирования.**

В 2008 г. мой обзор «**Автоматное программирование**» (https://is.ifmo.ru/works/_2010_09_08_automata_progr.pdf) стал одним из победителей Всероссийского конкурсного отбора обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы» (http://www.news.sfu-kras.ru/node/2860).

А вот еще две статьи на эту тему: ***Анисимов А.Е.*** Автоматное программирование**.** Часть 1 // Потенциал. 2008. № 3 (39), с. 27-35 (http://is.ifmo.ru/automata/_autonemi_3.pdf) и ***Анисимов А.Е.*** Автоматное программирование**.** Часть 2 // Потенциал. 2008. № 4 (40), с. 42-52 (http://is.ifmo.ru/automata/_autonemi_1.pdf). В этих статьях библиография – книги Непейводы и мой сайт. Андрей Евгеньевич совместно с Пупышевым В.В. является автором книги: Сборник заданий

по основам программирования. М.: Интуит, 2014 (https://www.ozon.ru/context/detail/id/2699626/), в которой есть задания и по автоматному программированию.

В этом же году разразилась склока с неким ***Dr Croco*** (А. Столяровым с ВМК МГУ) (https://www.wikiwand.com/ru/Обсуждение:Автоматное_программирование) по поводу статьи «**Автоматное программирование**» в русскоязычной Википедии (https://ru.wikipedia.org/wiki/Автоматное_программирование). Куратор этой статьи (физик-теоретик) принял сторону Столярова, который попытался свой текст перенести в англоязычную Википедию, что ему в полной мере осуществить не удалось (https://en.wikipedia.org/wiki/Automata-based_programming) – в начале англоязычной статьи есть такая ссылка: *For other uses, see* ***Automata-Based programming*** *(Shalyto's approach)*. Эта статья размещена по адресу: https://en.wikipedia.org/wiki/Automata-based_programming_(Shalyto's_approach). Она существует также и по адресу: https://en.academic.ru/dic.nsf/enwiki/9300552/Automata.

Все эти годы я утверждал и продолжаю утверждать, что оба термина «автоматное программирование» и *Automata-Based Programming* предложены мною (https://vk.com/@1077823-vtomatnoe-programmirovanie). По второму термину доказательство приведено здесь: https://www.semanticscholar.org/topic/Automata-based-programming/2609355. Эти термины очень важны, так как известно, что «**как лодку назовешь, так она и поплывет**»…

А вот, что по этому поводу написано обо мне в комментарии к тексту *DrCroco* (https://ru-cs.livejournal.com/7377.html): «Да, я тоже в свое время поразился этой ситуации... Но знаете, в чем забавная деталь? **Словосочетание «автоматное программирование» никто не догадался в свое время «закопирайтить». А г-н Шалыто догадался! И что теперь? Попробуйте найти этот термин в другом месте, без ссылок на Шалыто**» (*rg_software*). И это написал не мой знакомый или ученик...

А еще я предложил термин ***Automata-Based Control*** (***Shalyto A.A.*** Automata-Based Programming and Automata-Based Control. 2009. http://is.ifmo.ru/articles_en/_2009_10_07_automata_based_programming.pdf). Интересно, что на русском языке термин «**Автоматное управление**» был предложен не мною: он был использован в названии книги «**Автоматное управление асинхронными процессами в ЭВМ и дискретных системах**. Под редакцией В.И. Варшавский. М.: Наука, 1986. – 398 с.», однако в английском переводе этой книги была применена совсем другая терминология: ***Self-Timed Control of Concurrent Prosses****. Kluwer Academic Publishers. Editor: V.I. Varshavsky. 1990.*

Термин «**Автоматное управление**» использовал также и я, но значительно позднее: в **2009-2011** гг. в рамках «Аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы». «Проведение фундаментальных исследований в области естественных, технических и гуманитарных наук. Научно-методическое обеспечение развитие инфраструктуры вузовской науки» мы в Университете ИТМО выиграли грант на выполнение работ по теме: «**Адаптивное и автоматное управление мобильными роботами**». Руководители – **А.А. Бобцов и А.А. Шалыто**.

В 2008 г. мы познакомились с выдающимся учителем информатики из известного на всю страну московского лицея «Вторая школа» **Ильей Дединским**. Он привез к нам на кафедру трех школьников не старше восьмого класса, прослушав которых Георгий Корнеев, сказал, что если бы его так учили программированию в школе, то не знает каких бы высот он добился. Один из этих ребят восьмиклассник Леонид Столяров стал заниматься автоматным программированием, потом увлек еще парочку школьников. В результате были опубликованы следующие работы: 1. ***Столяров Л.В., Дединский И.Р., Шалыто А.А.*** Трансляция описаний автоматов, представленных в формате *Microsoft Visio*, в исходный код на языке C // Прикладная дискретная математика. Приложение. 2009. № 1, с. 81-83. http://www.lib.tsu.ru/mminfo/000349342/04-01/image/04-01-073.pdf. 2. ***Столяров Л.В.***Трансляция описаний автоматов, представленных в формате *Microsoft Visio* в исходный код на языке *C* // Компьютерные инструменты в образовании. 2009. № 5, с. 35-44. http://is.ifmo.ru/works/_2010_01_21_stolyarov.pdf. 3. ***Столяров Л.В., Петряйкин Ф.А., Уваров Н.С.*** Разработка платформы для автоматного моделирования и проведения соревнований автоматных интеллектов с трехмерной визуализацией /Материалы II Международной научно-практической конференции «Объектные системы-2010» (Зимняя сессия). Ростов-на-Дону. 2010, с. 75-81. http://is.ifmo.ru/works/_2010_12_25_stoljarov.pdf. А здесь (http://vimeo.com/9122399) опубликовано видео про созданную этими ребятами игровую платформу,

реализованную с использованием автоматного программирования. **Деятельность в этом направлении Илья продолжал до 2014 г.**: ***Пимкин А.*** Транслятор описания конечного автомата в исходный код на языке описания аппаратуры *Verilog* (http://ded32.net.ru/news/2014-02-25-75).

После общения с нами Дединский написал в газету для учителей информатики России статью (http://inf.1september.ru/view_article.php?ID=200900802, http://is.ifmo.ru/automata_school/_dedinskij.pdf): ***Дединский И.Р.*** **Почему мы стали заниматься автоматным программированием?** // Информатика. 2009. № 8, с. 8, 9. В ней, в частности, говорится: «Помню, будучи студентом, был удивлен одному из неформальных советов собирающимся заниматься научной работой: ***выбирай не тему, выбирай руководителя.*** Поэтому, **когда я узнал о том, что в Университете ИТМО** есть факультет, кафедра и люди, занимающиеся некой современной тематикой (автоматным программированием), доступной для понимания сильными школьниками, и не гнушающиеся с этими школьниками всерьез и напряженно работать – то, как говорят, я «сделал стойку». Почему я сказал «некой» программ– потому что не тема красит научный коллектив, а коллектив – тему, и **да простит меня Анатолий Абрамович Шалыто** за такие слова, **если бы он и его коллектив занимались чем-то другим, я все равно бы сделал эту «стойку», выбирая не тему – выбирая людей»**. Таким был выданный нам аванс, который, к сожалению, мы оправдали лишь частично.

В **2008 г.** мы получили премию Правительства РФ по образованию (https://rg.ru/2009/01/16/premii-obrazovanie-dok.html), которая имела такое название: «Инновационная система поиска и подготовки высококвалифицированных специалистов в области производства программного обеспечения на основе проектного и соревновательного подходов». Я был включен в авторский коллектив **за создания подхода к обучению студентов проектному подходу на основе автоматного программирования** (http://is.ifmo.ru/award/_award.pdf).

В этом году в Ярославском ГУ была защищена диссертация: ***Кубасов С.В.*** *Верификация автоматных программ в контексте синхронного программирования* (http://is.ifmo.ru/disser/kubasov_disser.pdf).

В 2009 г. был опубликовано первое издание книги *Поликарпова Н.И., Шалыто А.А.* Автоматное программирование. СПб.: Питер. 2009, содержащее **издательские** неточности из-за нашей невнимательности при чтении верстки книги (http://is.ifmo.ru/automata/shalytobook/).

Забавно, что какое-то время **одна из полок в Доме книги в Питере выглядела так: Д. Кнут, Н. Поликарпова и А.А. Шалыто, Б. Мейер** (http://is.ifmo.ru/books/book_dk). Вот рецензии на нашу книгу: http://is.ifmo.ru/books/_trudy_spiiran.pdf, http://is.ifmo.ru/books/Pages_30-31_from%20itn_125(2009-05).pdf, http://is.ifmo.ru/books/hard, а что написал на нашей книге **Джон Хопкрофт** можно посмотреть здесь: http://is.ifmo.ru/books/hopkroft_and_book.

В **2009, 2010 гг.** рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. «**Проведение научных исследований целевыми аспирантами**» мы выиграли открытый конкурс на выполнение НИР по направлению «Информатика» по теме «**Разработка методов машинного обучения на основе генетических алгоритмов для построения управляющих конечных автоматов**». Руководитель – Ф.Н. Царев.

В **2009-2011 гг.** в рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы. «**Проведение научных исследований молодыми учеными – кандидатами наук**» мы выиграли открытый конкурс на выполнение НИР по направлению «Информатика» по теме «**Разработка методов совместного применения генетического и автоматного программирования для построения систем управления беспилотными летательными объектами**». Руководитель – В.С. Гуров.

В те же годы в рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы. «**Проведение научных исследований научными группами под руководством кандидатов наук**» мы выиграли открытый конкурс на выполнение НИР по направлению «Механика», «**Информатика**», «Математика» по теме «**Методы повышения качества при разработке автоматных программ с использованием функциональных и объектно-ориентированных языков программирования**». Руководитель – Д.Г. Шопырин.

В **2010 г.** было опубликовано второе издание книги ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. СПб.: Питер, в котором **издательские неточности были устранены**. (https://www.piter.com/product/avtomatnoe-programmirovanie-2-e-izd).

18.10.2010 г. у меня был пленарный доклад «Автоматное программирование» на **Второй российской конференции с международным участием «Технические и программные средства систем управления, контроля и измерения»** в организации, которую в молодости я почти что боготворил – **Институте проблем управления РАН**. С текстом этого доклада можно ознакомиться по адресу: http://is.ifmo.ru/works/2010/UKI-shalyto-automata-programming.pdf.

22.11.2010 г. я и Федор Царев были в *ETH* (Цюрих) на конференции, посвященной шестидесятилетию **Бертрана Мейера**, который в то время начинал работать у нас в Университете по совместительству. На фотографиях (https://vk.com/id1077823?z=album1077823_122563144) изображены: **Никлаус Вирт, Эрих Гамма, Джозеф Сифакис, Давид Парнас, Юрий Гуревич**, **Андрей Терехов** и другие участники конференции.

Отмечу, что в свое время я попал в очень хорошую компанию – в «**Bertrand Meyer's gallery of computer scientists**» (http://se.inf.ethz.ch/old/people/meyer/gallery/), в которой есть все упомянутые выше ученые, но потом «Бертранова любовь» ко мне вместе с моим портретом исчезла, как впрочем, и любовь к ИТМО.

В Цюрихе произошла интересная история, начало которой я описал еще в 2003 г. и назвал «**Лучше, чем документация на телевизор»** (http://is.ifmo.ru/reflections/mystories/): «Один мой студент, увидев документацию на проект создания программы, выполненный по *Switch*-технологии, сказал задумчиво: «**Это лучше, чем документация на телевизор. Это, видимо, как документация на системы управления подводной лодкой**».

Вот ее продолжение. Когда на конференции в *ETH*, я показал одному из докладчиков – ***David Parnas*** (https://en.wikipedia.org/wiki/David_Parnas) – классно оформленную проектную документацию на программу (ошибочно названную мною «программной документацией») (http://is.ifmo.ru/download/short_dg.pdf), он незамедлительно предположил, что я связан с военно-промышленным комплексом, так как, по его мнению, в иных местах документацию так хорошо не оформляют.

Мое отрицание этого факта, он всерьез не воспринял, так как сам был оттуда и этого, в отличие от меня, не скрывал. Да и как я мог «не попасться», если даже в указанной выше статье в *Wikipedia* о *Parnas*, есть такие слова: «He is also noted for his advocacy of precise documentation» – он пропагандировал точную (четкую, аккуратную) документацию.

Еще одна история, произошедшая там, состояла в том, что я хотел рассказать **Джозефу Сифакису** (одному из создателей метода верификации ***Model Checking***, за который они получили премию Тьюринга) о том, что их метод **классно работает на автоматных программах.** Это объясняется тем, что описание парадигмы *Model Checking* начинается со слов: **»По программе строится модель»** со всеми вытекающими отсюда неприятными последствиями, в то время как при применении автоматного программирования **»по автоматной модели формально и изоморфно строится программа».** После этого все встает на свои места, и можно эффективно применять *Model Checking* к автоматной модели, с которой все начинается.

Однако до доклада я не смог его узнать – так сильно он изменился по сравнению с опубликованными ранее фотографиями, а после доклада – он сразу же исчез, так как обиделся ... на юбиляра, который тоже не узнал его. Так Сифакис остался в неведении об удобстве верификации их методом именно автоматных программ... Обе эти истории описаны здесь: https://vk.com/@1077823-deistvitelno-luchshe-chem-dokumentaciya-na-televizor.

Сегодня вместо термина «верификация автоматных программ» я использую (https://ntv.ifmo.ru/file/article/21921.pdf) термины «верификация автоматных моделей» (для простейших систем управления технологическими процессами) и «**валидацией автоматных спецификаций**» (систем управления, используемых на практике).

Ссылки на *Switch*-технологию, автоматное программирование и другие работы. выполненные **с 2003 по 2010 г.,** приведены здесь: http://is.ifmo.ru/aboutus/ssilki_switch/.

В **2011 г.** мы опубликовали два книжных издания:

– ***Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.*** Верификация автоматных программ**.** СПб.: Наука. 2011. 242 с. http://is.ifmo.ru/verification/velder_verification_posobie_nauka.pdf);

– ***Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.*** Верификация автоматных программ**.** Учебное пособие. СПбГУ ИТМО, 2011. 242 с. http://is.ifmo.ru/verification/velder_verification_posobie.pdf.

В **2011 г.** был опубликован второй тираж второго издания книги ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование (https://www.piter.com/product/avtomatnoe-programmirovanie-2-e-izd)**.**

В **2011-2013 гг.** рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы. «Проведение научных исследований научными **группами под руководством докторов наук** по направлениям «Механика», «**Информатика**», «Математика» мы выиграли конкурс на выполнение работ по теме «**Разработка метода машинного обучения на основе алгоритмов решения задачи о выполнимости булевой формулы для построения управляющих конечных автоматов**». **Руководитель – А.А. Шалыто. Ответственный исполнитель – В.И. Ульянцев.**

В **2011 г.** был опубликован «Научно-технический вестник СПбГУ ИТМО. 2011. № 2 (72)». http://is.ifmo.ru/works/. Он содержит **17 наших статей** и имеет подзаголовок «**Технологии автоматного программирования и искусственного интеллекта**».

В этом же году **В.И. Шелехов** опубликовал статью «**Язык и технология автоматного программирования**» (https://persons.iis.nsk.su/files/persons/pages/automatProg.pdf). При этом он, в частности, пишет, что **концепция автоматного программирования разработана Анатолем Шалыто**. **Наконец-то, не только Куракин из «Войны и мира», но и я стал Анатолем!**

В **2012 г.** была опубликована статья «**Автоматное программирование как новый способ создания автоматических торговых систем**» (https://www.mql5.com/ru/articles/446). Во введении к ней сказано: «**В России А.А. Шалыто (профессор, доктор технических наук, заведующий кафедрой «Технологии программирования» СПбГУ ИТМО) в 1991 г. разработал подход к программированию, названный им «автоматное программирование»**. Я думаю, что читателям будет интересно увидеть простоту и легкость автоматного программирования, на основе которого создана *Switch*-технология. ***Это настолько удобное программирование трейдерских систем, что лучше не придумаешь***. Такой стиль очень точно вписывается в систему принятия сложнейших решений».

А еще **в этой статье** приводятся мои слова: «Итак, **без исходных текстов плохо, но и с ними тоже бывает нехорошо. Чего же не хватает «для полного счастья»? Ответ прост: проектной документации**, выполненной весьма подробно и аккуратно, в которую программная документация входит как одна из составляющих. **Мосты, дороги и небоскребы без проектной документации обычно не строятся, а вот о программах несмотря на их в общем случае сложность такого не скажешь.** В программировании сложилась ситуация, определяемая так: «**Если бы строители строили дома так, как программисты пишут программы, достаточно было бы одного единственного дятла, чтобы разрушить цивилизацию**». Время идет, а в этом вопросе несмотря на все мои старания, к сожалению, ничего не меняется...

В 2012 г. рамках **Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России» на 2009-2013 гг. мы выиграли конкурс** (**первое место при 55 организациях**) на заключение государственного контракта на проведение научных исследований научной группой под руководством **доктора наук** в области **«Механика», «Информатика», «Математика»** по теме «**Разработка методов построения управляющих конечных автоматов по обучающим примерам на основе решения задачи удовлетворения ограничений**». **Руководитель – А.А. Шалыто. Ответственный исполнитель – В.И. Ульянцев.**

В **2013 г.** была опубликована статья: ***Большаков О.А., Рыбаков А.В.*** Автоматное моделирование систем автоматизации с реализацией на ПЛК // Автоматизация в промышленности. 2013. № 10, с. 61-64. http://is.ifmo.ru/works/2013/bolshakov-fsm-modeling.pdf. С ссылками там все нормально...

В **2014 г.** появился такой текст: «**Концепция автоматного программирования разработана *Анатолием Шалыто***, в том числе в интеграции с объектно-ориентированным программированием. Автоматная программа определяется как совокупность классических конечных автоматов. Используются графическое и текстовое представления программы. **Управляющие состояния** являются значениями переменной, соответствующий этим состояниям. При реализации автоматной программы применяется ***Switch*-технология**. ***Термин «автоматное программирование» и его аналог «Automata-Based Programming» применяется только в России*. Тем не менее, автоматные методы программирования заложены во многих языках. Автор благодарен *А.А. Шалыто* за его работы по автоматному программированию. Предлагаемое мною понятие автоматной программы** концептуально не отличается от введенного Анатолием Шалыто, однако различия в языке и технологии существенны» (**Шелехов В.И.**, http://persons.iis.nsk.su/files/persons/pages/automatProg.pdf).

Интересно, что и в следующей работе того же 2014 г. **Шелехов**, с которым я не знаком, еще раз упоминает меня: «**Автор благодарен А.А. Шалыто за работы по автоматному программированию, стимулировавшие мои исследования**» (https://persons.iis.nsk.su/files/persons/pages/req_tech.pdf).

В **2014, 2015 гг.** в рамках конкурса инициативных научных проектов РФФИ, выполняемых молодыми учеными «Мой первый грант», мы выиграли два гранта для выполнения работ по темам «**Разработка метода построения конечных автоматов для управления объектами со сложным поведением на основе обучающих примеров**» (***Казаков С.В., Бужинский И.П.*** (руководитель). и «**Разработка методов автоматизированного построения надежного программного обеспечения по обучающим примерам и темпоральным свойствам на основе автоматного подхода**» (***Чивилихин Д.С., Ульянцев В.И.*** (руководитель)).

В **2014-2016 гг.** в рамках конкурса РФФИ мы получили право на выполнение работ по теме «**Разработка муравьиных алгоритмов для построения конечных автоматов**», а в рамках государственного задания Министерства образования и науки РФ (раздел «Компьютерные и информационные науки») – по теме «**Технология разработки программного обеспечения систем управления ответственными объектами на основе методов машинного обучения и конечных автоматов**». Руководитель – А.А. Шалыто.

В **2014 г.** в сети появилась работа ***Карпов В.Э.* Автоматное программирование и робототехника** (http://raai.lgg.ru/about/persons/karpov/), а письмо Карпова ко мне размещено здесь: http://is.ifmo.ru/books/karpov_letter.

В 2015-2016 гг. появилась надежда на применение автоматного программирования при создании больших программных систем (*Шалыто А.А.* Может быть, при построении определенного класса больших программных систем лед тронулся? 29.06.2015. https://is.ifmo.ru/main/bank.pdf; ***Шалыто А.А.*** Автоматное программирование начинает новую жизнь. 15.01.2016. https://is.ifmo.ru/main/new-life.pdf). Внедрение состоялось в платформе *Corezoid*, но сегодня в 2024 г. все ссылки о внедрении в этих статьях не действуют.

В **2016 г.** наша книга ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. СПб.: Питер. 2011, 176 с. **стала вечной,** так как превратилась в цифровую (https://www.ozon.ru/context/detail/id/28260411/) и распространяется в электронном виде, в частности, с помощью портала «ЛитРес» (https://www.litres.ru/anatoliy-shalyto/avtomatnoe-programmirovanie-585075/otzivi/).

Вот отзыв, приведенный там: «Мой первый опыт работы программирования в данном стиле (можно назвать методом) был связан с разработкой ПО под микроконтроллер *STM32* для задачи управления несколькими двигателями с учетом показаний датчиков. **Коллега порекомендовал почитать Шалыто «Автоматное программирование»** и воспользоваться подходами, приведенными там. **Результат превзошел мои ожидания**, объем кода удалось сократить раза в два, читаемость улучшилась. Книга будет полезна в первую очередь начинающим разработчикам – примеры понятны, язык доступен. «Автоматное программирование» оказало существенное влияние на мой стиль разработки ПО не только для микроконтроллеров, но и десктопных систем» (*AlexisVaBel*). Теперь еще один отзыв: «Автоматное программирование позволяет единообразно осуществлять разработку программного обеспечения, предназначенного для управления логическими

контроллерами. ***Очень хорошо, что автор настойчиво продвигает свою идею***. **И хорошо то, что это отечественное ноу-хау**» (*Dastini*).

В **2016 г. Ф. Новиков** и **И. Афанасьева** в статье «**Кооперативное взаимодействие автоматных объектов** // Информационно-управляющие системы. 2016. № 6, с. 50-64» (http://www.i-us.ru/index.php/ius/article/view/4265) написали следующее: «Уже более четверти века развивается парадигма автоматного программирования – подход к описанию поведения, основанный на явном выделении состояний. ***Несравненные заслуги в развитии и продвижении этого подхода принадлежат профессору А.А. Шалыто***».

А вот, что пишет «моими словами» профессор **Е.М. Лаврищева** в учебно-методическом пособии **«Программная инженерия. Тема 1. Теория программирования. М.: МФТИ, 2016**, 48 с.» (http://www.computer-museum.ru/books/lavrischeva_1_programming.pdf): «**Автоматное программирование основано на применении конечных автоматов для описания поведения программ**. Автоматы задаются графами переходов, для различения вершин в которых вводится понятие «кодирование состояний». Особенность автоматного программирования состоит в том, что **графы переходов используются при спецификации, проектировании, реализации, отладке, документировании и сопровождении программ**. (Они могут применяться также и как язык программирования, А.Ш.).

Программирование выполняется «через состояния», а не «через события и переменные», что позволяет лучше понять и специфицировать задачу и ее составные части. Переход от графового представления к текстовому осуществляется формально и изоморфно с применением конструкции *switch* (в языке *C*) или ее аналогов (в других языках). Поэтому стиль автоматного программирования часто называют «***Switch*-технологией**». **В этом случае используется многозначное кодирование состояний.**

В настоящее время этот стиль развивается в нескольких вариантах, различающихся как классом решаемых задач, так и типом вычислительных устройств, на которых осуществляется программирование. Известны, например, его варианты для систем логического управления, в которых события отсутствуют, входные и выходные воздействия являются двоичными переменными, а операционная система работает в режиме сканирования.

Автоматный подход распространен и на событийные системы, называемые также реактивными. В них входные воздействия используют события, выходные воздействия – произвольные процедуры, а в качестве операционных систем – любые операционные системы реального времени. Для программирования событийных систем с применением автоматов применяется процедурный подход, такой стиль программирования называется «**программированием с явным выделением состояний**». Известен также подход, основанный на совместном использовании объектного и автоматного стилей и называемый «**объектно-ориентированным программированием с явным выделением состояний**». В контексте обеспечения качества применение автоматов проясняет поведение программы, а **наличие хорошей проектной документации упрощает ее изменение путем рефакторинга программы**».

С **2016 г.** можно считать, что автоматное программирование пошло в «народ» – ниже приводится две ссылки на выступления абсолютно незнакомых мне людей: вот видео об использовании автоматов в программных приложениях (https://www.youtube.com/watch?v=kBjqenUQvlU), а здесь – видео о применении конечных автоматов в платежных системах (https://www.youtube.com/watch?v=GEykpn6IgAA).

В **2017-2019 гг.** в рамках государственного задания Министерства образования и науки РФ. «**Инициативные научные проекты фундаментального характера**» мы получили право на выполнение работ по теме «**Технология разработки программного обеспечения систем управления ответственными объектами на основе глубокого обучения и конечных автоматов**». Руководитель – **А.А. Фильченков**.

20.01.2017 г. в газете «Коммерсант» появилась статья «**В Росатоме нашли проблемы с ядром**» (https://www.kommersant.ru/doc/3196399), в которой обсуждался вопрос о проблемах с программным обеспечением на некоторых атомных станциях России. В ней, в частности, отмечалось отсутствие документации на программную платформу.

27.02.2017 г. в ответ я написал статью «***Программа как инженерный продукт, или зачем заказчику понимать структуру ПО изнутри***» (http://news.ifmo.ru/ru/science/it/news/6472/), в которой отметил, что с **1991 г. я развиваю в России автоматный подход к проектированию программ**, применение которого уменьшило бы число проблем, указанных в статье, например в части проектной документации на программное обеспечение (ПО). Эта документация при использовании предлагаемого мною подхода в наглядной форме содержит алгоритмы управления, контроля и сигнализации. В моей статье, в частности, говорится: «Имеющееся ПО не позволяет понимать, как программа будет себя вести в тех или иных состояниях, как именно и какие в нее вносили изменения. Более того, в тексте статьи в «Коммерсанте» сказано, что на этот софт нет никакой документации, а не только проектной. И это при том, что объект автоматизации – ядерный реактор (***Таккер К.*** Как управлять ядерным реактором. М.: ДМЕ, 2022, 230 с., https://dmkpress.com/files/PDF/978-5-93700-132-0.pdf), и почти никто, кроме, возможно, разработчика, которого, естественно, нет на объекте, а то уже и в живых, не понимает, как работает программа, управляющая им. И такой бардак с ПО творится почти везде в мире. Нормально ли это?». В тот же день эта статья вышла и на английском: ***Why Design Programs: Anatoly Shalyto on Automata-Based Programming*** (http://news.ifmo.ru/en/science/it/news/6472/).

03.03.2017 г. на портале «Хабр» пресс-служба Университета ИТМО на базе этой моей статьи опубликовала текст «**Лекарство от болезни»: автоматное программирование**» (https://habr.com/ru/company/spbifmo/blog/323122/).

В это время стало известно, что разработана отечественная среда моделирования технических системами *SimInTech*, в которой, в частности, могут применяться конечные автоматы (https://help.simintech.ru/#metodika/konechnye_avtomaty/konechnye_avtomaty_v_simintech.html). Однако при ее использовании они применяются крайне редко. В НПО «Аврора» эта среда используется, а автоматы в ней – нет (видимо, из-за громоздкости их изображения, https://simintech.ru/). С этой средой можно ознакомиться не только по документации, но и здесь: ***Карташов Б.А., Шабаев Е.А., Козлов О.С., Щекатуров А.М.*** Среда динамического моделирования технических систем *SimInTech.* М.: ДМК Пресс. 2017. 424 с. (https://dmkpress.com/catalog/computer/handbooks/978-5-97060-482-3/).

В **2018-2020 гг.** в рамках гранта РФФИ мы получили право на выполнение работ по теме «**Разработка методов машинного обучения для синтеза автоматных моделей систем управления с учетом темпоральных свойств и временных отсечек на основе пропозиционального кодирования**». Руководитель – В.И. Ульянцев.

В эти же годы мы выиграли молодежный грант РФФИ на проведение работ по теме «**Разработка эффективных методов машинного обучения для построения детерминированных конечных автоматов на основе решения задачи выполнимости**». Руководитель – **И.Т. Закирзянов**.

В **2018 г.** сначала было опубликовано видео о стейт-машинах на службе у *MVP* (https://www.youtube.com/watch?v=U3StVUzqmzc), а потом – видео, в котором рассказывается о том, что применять конечные автоматы при разработке игр хорошо (https://www.youtube.com/watch?v=bhtKYbBbt50).

После публикации этого текста на последнюю ссылку неожиданно откликнулась докладчица – **Алена Пономаренко** из компании *Social Quantum*, в которая написала в сети: «Анатолий Абрамович Шалыто сослался на мой доклад на конференции *DevGAMM*». При этом наш общий знакомый ***Vlad. Vishnykov*** отметил, что упоминания доклада другими людьми у Алены такой радости не вызывали :)). Потом кто-то в комментариях к ее посту спросил, используется ли где-либо еще автоматное программирование, на что **Михаил Глухов** ответил: «Да, я не только видел использующих эту технологию, но и сам из них». После этого он написал мне: «**Благодарю Вас за автоматное программирование**, которое я неоднократно применял в своих проектах, начиная с той Вашей лекции, на которой впервые узнал о нем и получил книжку. Это было примерно 20 лет назад. Теперь уже у меня появились свои ученики, и я хотел бы попросить у Вас посоветовать, как проще научить их автоматному подходу». «Для начала прочесть предыдущую статью и эту».

В **2018 г.** мой давний знакомый **Вячеслав Любченко** выступил на тему «**Автоматное программирование: определение, модель, реализация**» (https://www.youtube.com/watch?v=SrnjMx2G2MM) в лаборатории **Александра Константиновича**

Петренко в Институте системного программирования РАН (http://sdat.ispras.ru/?p=802). Доклад начинался с меня :-). Он что-то говорит на эту тему – даже не критикует, а потом долго рассказывает о своем. При этом отмечу, что ***создание несколько десятков ответственных систем моими аспирантами*** **Юрием Янкиным** (видео ***Янкин Ю.Ю., Шалыто А.А.*** Автоматное программирование ПЛИС (http://is.ifmo.ru/present/2012/Yankin-Shalyto-PLIS.exe) и **Антоном Калачинским** (***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, с. 88-92. http://is.ifmo.ru/works/_volobuev.pdf) подтверждает правильность поведения Любченко в том смысле, что он меня даже не критикует! По-моему, в его лекции упоминаются только одна фамилия, одно имя и одно отчество – мои.

Перекличку со мной Любченко ведет до сих пор. Вот примеры статей «обо мне»: «**Машина Тьюринга, как модель автоматных программ**» (https://habr.com/ru/post/481998/), «**Автоматы – вещь событийная?**» (https://habr.com/ru/post/483610/) и «**Автоматные рекурсивные вычисления**» (https://habr.com/ru/post/492958/). При этом отмечу, что **статьи по этой тематике я публиковал еще в начале двухтысячных.** Приведу два примера: ***Туккель Н.И., Шалыто А.А.*** От тьюрингова программирования к автоматному // Мир ПК. 2002. № 2, с. 144-149 (http://is.ifmo.ru/works/turing/) и ***Туккель Н.И., Шалыто А.А., Шамгунов Н.Н.*** Реализация рекурсивных алгоритмов на основе автоматного подхода // Телекоммуникации и информатизация образования. 2002. № 5, с. 72-99 (http://is.ifmo.ru/works/recurse/).

В **2019 г.** на своем *YouTube*-канале я выложил лекцию про автоматное программирование в лучшем качестве (https://www.youtube.com/watch?v=PPWTxceMutk), чем она была опубликована на том же канале на два года раньше (https://www.youtube.com/watch?v=tUo9ssPVa4c). Было несколько комментариев на нее: «Смотрел на одном дыхании, огромное спасибо за лекцию и привет родной Альма-матер!» (Александр Сальников), «Большое спасибо. Вспомнил родной МехМат» (Олег Мальсагов), «Купил Вашу книгу. Спасибо за лекцию. Пожалуйста, выкладывайте больше» (*Denys Bushulyak*), «Супер!» (Сергей Куксов), а Андрей Миронов из МГУ написал: «Шалыто, как всегда, гениален». Удивило... Более содержательным был пост от Кирилла Калишева «Я помню, когда еще был студентом в 90-х, работал в промышленной автоматизации, *real time control* и сложные состояния... **Ваша книжка по *Switch*-технологии про то, что *всю эту помойку нужно и можно генерировать из высокоуровневых описаний была откровением!*** Спасибо!».

А вот, что написал некто ***Vadim Gor***: «Анатолий Шалыто (хотя я с Вами и не знаком), хочу засвидетельствовать свое почтение за изобретение *Switch*-технологии! Чудесная вещь. Она близка к системной методологии, где имеет прямое отношение к моделированию доминант и детерминант в сложных системах. В молодости я с очень большим воодушевлением ее использовал в своей работе. Да и сейчас – тоже. **На всех программистских специальностях ее надо изучать сразу, чтобы мозги у программистов сразу вставали на место**. Потом их ставить труднее. В своей лекции Вы все очень правильно говорите, начиная от того, что «**графы должны быть по возможности планарны, а схемы красивы**». Автоматная модель – это несравнимо лучше превентивного кодерства».

Потом Вадим написал текст «**О спорах относительно парадигм программирования**», в котором, в частности, сказано: «**Желание написать такой материал навеяло автоматное программирование (*Switch*-технология) от Анатолия Шалыто, предложившего его**. Дело, конечно, не в открытии многоуровневых конечных автоматов для программирования, а в их удобном технологическом оформлении, сильно упрощающем как макетирование систем до непосредственного программирования на языках программирования, так и верификацию кода, вместе множеством достоинств, связанных с проблемами согласования алгоритмов с заказчиками программ. **Автоматное программирование вполне совместимо с объектно-ориентированным программированием** (ООП) и является средством построения «крепких» каркасов ООП-программ и, естественно, функционального кода тоже. Следовательно, хотелось бы раскрыть «**Настоящее ООП» = ООП + Каркасное программирование + Шаблоны проектирования + Автоматное программирование**», не противопоставляя его функциональному программированию, а просто показав приверженцам функционального подхода место последнего в указанной троице (каркасы и шаблоны по большому счету одно и то же, только на разных масштабах архитектуры)».

02.04.2019 г. Валерий Ракитин прочитал блестящую лекцию на тему **«Бесконечные возможности конечных автоматов»** (https://www.youtube.com/watch?v=6xzn78onzQk). **Автор предлагает писать**

программы до того, как начать кодировать. **По его мнению, должны создаваться «вечные» («бессмертные») программы за счет того, что в них легко вносить изменения**. Методологии, направленные на это, Ракитин назвал «**экологическим программированием**» (**eco-programming**). Это может, например, пригодиться во время тендера, когда его организаторы в последний момент меняют условия задачи с целью обеспечения преимуществ той команды, в которой организаторы тендера заинтересованы. Та команда, которая сможет быстро внести изменения в свой прототип, имеет шанс победить даже ангажированную команду. К одной из таких методологий автор относит **подход, основанный на применении конечных автоматов – автоматное программирование в моей терминологии**. Вот реакция одного из слушателей на эту лекцию: «**Не каждому дано сразу понять, что за конечными автоматами будущее!**». При этом, однако, автор понимает, что при использовании автоматного подхода только на первый взгляд «все понятно и все довольны. На деле, же не все так просто потому, что **это совершенно другой подход к программированию,** своя парадигма программирования **и этому надо учить и учиться**» (https://habr.com/ru/post/680160/).

13.12.2019 г. была опубликована лекция **А. Попцова** «Введение в автоматное программирование» (https://www.youtube.com/watch?v=7LwqznbApaY), которая имеет подзаголовок «**Автоматное программирование. Секретное оружие программиста при решении сложных задач**». Автор считает, что **программы – это способ выражения идей, и поэтому они должны быть написаны по-человечески. Это во многом обеспечивается применением автоматного программирования. Интересно, что в списке литературы (засечки с 1.27.56 по 1.29.44) только наши работы: книга с Поликарповой, названная канонической, моя книга о *Switch*-технологии, статья с Туккелем и сборник ИТМО (Выпуск 53 за 2008 г.), целиком посвященный автоматному программированию.**

В **2019-2021 гг.** мы провели исследования по гранту РФФИ по теме: «**Разработка методов машинного обучения на основе *SAT*-решателей для синтеза модульных логических контроллеров киберфизических систем**». Руководитель – Д.С. Чивилихин.

В **2020, 2001 гг.** нами был выигран конкурс «Научное наставничество» на лучшие проекты фундаментальных научных исследований, выполняемые молодыми учеными под руководством ведущего ученого – наставника, проводимого совместно Российским фондом фундаментальных исследований и образовательным фондом «Талант и успех» («Сириус») – https://www.rfbr.ru/rffi/ru/rffi_contest_results/o_2099426. Тема проекта «**Разработка методов синтеза конечно-автоматных алгоритмов управления для программируемых логических контроллеров в распределенных киберфизических системах**». Молодые ученые: Д.М. Суворов, А.Л. Павленко, К.И. Чухарев, П.А. Овсянникова, И.Т. Закирзянов, Д.С. Чивилихин и В.И. Ульянцев. Наставник – **В.В. Вяткин**.

24.06.2020 г. на портале «Хабр» была опубликована статья ***Александра Соловьева*** (Dr_Dash) «**Автоматное программирование – новая веха или миф? Часть 1**», которая популяризирует этот стиль программирования (https://habr.com/ru/post/331556/). Седьмого июля появилась вторая ее часть, в которой термин «миф» уже не используется: «**Автоматное программирование. Часть 2. Диаграммы состояний и переходов**» (https://habr.com/ru/post/332508/), а 11 ноября – третья. В ней было продолжено обсуждение диаграмм состояний и переходов (https://habr.com/ru/post/332664/). 18 ноября появилась четвертая часть, посвященная эффективности автоматно-спроектированных программ (https://habr.com/ru/post/341888/). 25 ноября и второго декабря Соловьев опубликовал двухчастную статью под названием «**Автоматный практикум**» (https://habr.com/ru/post/342048/, https://habr.com/ru/post/343736/).

Отмечу, что в этих статьях какая-либо библиография отсутствует, как будто в этом вопросе автор с Луны свалился, как, впрочем, и его читатели – на Луне, действительно, могут не знать, что этот подход к программированию на Земле, как отмечено выше, развивается более 30 лет (***Harel D.*** Statecharts: a Visual Formalism for Complex Systems // Science of Computer Programming. V. 8. 1987. Issue 3, pp. 231-274). Хотя в последних комментариях ко второй статье Соловьев, все-таки, написал: «**Стоит отдать должное Шалыто** (если считает нужным – пусть отдает, А.Ш.): **он популяризирует автоматное программирование судя по его книгам уже лет 30 – этакий Дон Кихот и первопроходец**. **Может он и увидит эру автоматного программирования**». В этот момент «проснулся» один читателей Соловьева: «Если еще не знакомы, возможно Вам будет интересно познакомиться с работами Анатолия Абрамовича Шалыто (http://www.softcraft.ru/auto/). Можно хотя бы сюда зайти: http://www.softcraft.ru/auto/switch/aptech/.

Ну, или совсем кратко: https://ru.wikipedia.org/wiki/Switch-технология«. Интересно, что что читатель **предлагает зайти на давно не поддерживаемый на сайт А. Легалова**, **а моего сайта** (http://is.ifmo.ru), **как будто, и не существует...** Неисповедимы пути Господни.

01.08. 2020 г. на портале «Хабр» опубликована статья «**Самые простые конечные автоматы или стейт-машины в три шага**» (https://habr.com/ru/post/509120/).

Среди курсов по обучению программированию на *JavaScript* на сайте онлайн-школы «Хекслет» я обнаружил курс «**Автоматное программирование**», который был сначала обновлен **23.08.2020 г**., а затем 23.09.2023 г. (https://ru.hexlet.io/courses/js-abp). Второго ноября 2023 г. сооснователь этой школы **Кирилл Мокевнин** прочел лекцию на тему «**Конечные автоматы как способ значительно упростить логику и понимание кода**» (https://www.youtube.com/watch?v=knoVv2ncwVI). В этом ролике на засечке 12.35 появляется обложка нашей книги с Надей Поликарповой книги «Автоматное программирование» (https://is.ifmo.ru/books/_book.pdf).

В этом же году под эгидой МФТИ я нашел сайт «Теория и реализация языков программирования» (http://trpl7.ru/), а в нем раздел «Конспекты по семинарам» (http://trpl7.ru/Conspectus/_trpl_2017.htm), в котором одно из приложений называется «**Что такое автоматное программирование» (введение)**». Оно начинается так: «В своей книге и в ряде статей профессор А. Шалыто из Университета ИТМО с сотрудниками проводит мысль о том, что потребность в автоматном подходе к программированию появляется тогда и в той мере, в какой программа или ее часть описывает **систему со сложным поведением** и сама является таковой» (http://trpl7.ru/Conspectus/Automat_progr_2011_09.htm).

Однако, до широкого признания автоматной парадигмы даже в России еще далеко, хотя отдельные проблески в этом отношении есть. Например, в книге *DocCroco*, который активно боролся со мной и с моим взглядом на автоматное программирование в *Wikipedia* (**Столяров А.В.** Программирование: ведение в профессию. III: системы м сети. М.: МАКС Пресс. **2017**. 400 с., http://www.stolyarov.info/books/pdf/progintro_vol3.pdf) имеется раздел 6.4.5, названный «**Сеанс работы как конечный автомат**». Этот раздел заканчивается так: «Иногда говорят, что состояние в обычных имперсивных (построенных на приказаниях) программах присутствует неявно – в отличие от случая, когда состояние явным образом определяется значениями переменных. Такой стиль написания программ называют ***программирование в терминах явных состояний***».

Эту тему автор продолжает в следующем томе книги: Столяров А.В. Программирование: ведение в профессию. IV: Парадигмы. М.: МАКС Пресс. **2020**. 656 с., http://www.stolyarov.info/books/pdf/progintro_vol4.pdf). При этом в разделе «9.4.2. Программирование в терминах явных состояний» он пишет, что «**такое программирование иногда называют автоматным программированием** поскольку необходимый при этом стиль мышления очень похож на тот, что приходится применять при работе с формальными автоматами. Можно сказать, что «***автоматное программирование как раз и состоит в переходе от неявных составляющих состояний выполнения программы к их явному указанию в виде значений переменных***». В этом же разделе: «**Парадигмы программирования – особенности мышления программиста. Они не в компьютере, а в голове программиста, определяя то под каким углом зрения программист рассматривает решение поставленной задачи**». После этого автор вскользь ***упоминает «автоматную парадигму».***

«Два» слова о склоке. Она не стоит и выеденного яйца, так как до тех пор, пока автоматное программирование не начнет широко внедряться на практике, это ни на что не влияет, *так как* ***по словам Аристотеля «даже известное известно немногим»***.

25.02.2021 г. на портале «Хабр» появилась статья **Ильи Казначеева «Автоматы на службе распределенных транзакций»** (https://habr.com/ru/post/544042/), в которой рассказывается о доменах, построенных на основе конечных автоматов, и распределенных транзакциях, реализованных с их помощью. При этом автор пишет: «**Термин «автоматизированный домен» был мною взят у Анатолия Шалыто** (http://is.ifmo.ru/books/_book.pdf) по аналогии с его «**автоматизированными объектами управления**» и «**автоматизированными классами**» («**автоматизированные объекты управления как классы**»).

В послесловии к статье автор пишет: «А вот отличная статья о Анатолии Шалыто (интервью **Анатолий Шалыто: «Если человек сомневается, заниматься ли ему наукой, ему стоит**

заняться чем-то другим» (https://habr.com/ru/company/dataart/blog/538580/). Я не знал про него, когда читал его книгу, а потом встретил такую историю, которая мне очень понравилась. Просто делюсь».

07.09.2021 г. была опубликована лекция «**Автоматное программирование с примерами *JavaScript***» (https://www.youtube.com/watch?v=mxz7_zcip0c), прочитанная на Украине, без ссылок на кого-нибудь. Лекция начинается словами: «**Мы должны в своем курсе покрыть все парадигмы программирования, и это одна из абсолютно фундаментальных вещей, так как вся вычислительная техника – это автоматы**».

01.09.2022 г. С. Пономарев сделал доклад на тему «State Machine: что это и зачем» (https://www.youtube.com/watch?v=vlqtNtTMdpk). По мнению автора большое число состояний осложняет поддержку проекта. Естественно, в докладе нет ссылок на кого-либо.

О работах В. Шелехова, М. Нейзова, а также одной из последних работ В. Зюбина, я написал в статье «О развитии автоматного программирования» (https://vk.com/@1077823-o-razvitii-avtomatnogo-programmirovaniya). Там, в частности, отмечено, что **автоматное программирование излагается Шелеховым в курсе лекций в Новосибирском государственном университете**.

Приведенный выше текст относится в основном к работам по автоматному программированию на русском языке. О предложенной мною терминологии на английском языке рассказано здесь: https://vk.com/@1077823-vtomatnoe-programmirovanie. Выше была упомянута **классическая работа Харела**, опубликованная в 1987 г., а еще я приведу книгу **2006** г., близкую по духу к тому, что делаю я: ***Wagner F., Schmuki R., Wagner Th., Wolstenholme P.*** Modeling Software with Finite State Machines. A Practical Approach (http://is.ifmo.ru/download/modelingsoftwarewithfinitestatemachinesapracticalapproach.pdf).

Работы по этой тематике и на Западе, похоже, не стихают ни на минуту. Так, например, **25.03.2021 г.** на *YouTube*-канале *MATLAB* можно было прослушать часовую лекцию на тему: *Modeling State Machines with Stateflow. What`s Your State? **(https://www.youtube.com/watch?v=rUeMUCrlxP0), а в другом месте: https://www.youtube.com/watch?v=W-jyNF3l84c – лекцию на тему: *How to Use the JSSM / FSL live editor (low quality draft)*, посвященную одному из инструментов для *FSL – Finite State Language* (https://fsl.tools/#videotable_at_top).

Интересно, что еще **в 2007 г.** историю о создании ***Statecharts***, которая в некотором смысле напоминает рассказываемую мною о создании автоматного программирования (https://vk.com/@1077823-vtomatnoe-programmirovanie), поведал ***Дэвид Харел***: Statecharts in the Making: A Personal Account (https://dl.acm.org/doi/10.1145/1467247.1467274). Он оказался предусмотрительнее :-) меня, так как сохранил и привел в статье черновики, посвященные созданию *Statecharts*. Когда он разрабатывал эти диаграммы, Интернета еще не было, а иностранные журналы мне были практически недоступны, но в своей книге 1998 г. на статью Харела 1987 г., указанную выше, я сослался. Может это и хорошо, что все, что я сделал в этой области, было сделано независимо – иначе либо «руки не поднялись», либо «опустились», а так многое удалось придумать самому и рассказать другим, в том числе и студентам.

Интересно, что в моих работах на английском языке, которые появлялись до выхода и после выхода моей книги о *Switch*-технологии, **рецензенты не указывали мне на работы Харела – возможно, они и сами тогда их еще не знали**. Этого даже не произошло и в **1995 г.**, когда я выступал в Monterey (California) с докладом на эту тему (http://is.ifmo.ru/science/_cognitive_properties_of_hierarchical_representations_of_complex_logic_structures.pdf) на *Workshop* «*Architectures for Semiotic Modeling and Situation Analysis in Large Complex Systems*», проходившем под руководством **Джима Альбуса** из Национального института стандартов и технологий США и **Дмитрия Поспелова**, которому предложенный мною автоматный подход к программированию понравился (https://vk.com/@1077823-vtomatnoe-programmirovanie).

Вопрос о приоритете в чем-либо (даже в достаточно узкой области) весьма сложный. Об этом, в частности, мой текст «**Автоматное программирование, водка и буква Ё**» (http://is.ifmo.ru/download/2008-03-17_automata.pdf), опубликованный еще в 2008 г., а также текст «**О приоритете**» (https://vk.com/@1077823-o-prioritete). Приведу примеры из них.

Первый пример: «**Довод Шалыто** о том, что именно он оформил программирование состояний в некую стройную систему, которую он назвал «автоматным программированием» и именно в этом

его заслуга, **вполне оправдан**. Поэтому и говорят, что автоматное программирование изобрел Шалыто. **Это как с буквой Ё. Мало кто знает что её придумала Екатерина Дашкова (1743-1810), но все знают что первым применять её начал Николай Карамзин (1766-1826), поэтому зачастую говорят, что буква Ё – буква Карамзина**» (https://alexott.livejournal.com/321730.html?thread=1717698#t1717698). Это написал человек, с которым я не был знаком ни тогда, ни теперь.

Второй пример: «Есть история про исследования раствора ... из воды и спирта, которые проводил **Дмитрий Менделеев (1834-1907)**, до открытия им Периодического закона. Представляете, как бы над ним некоторые смеялись сейчас, не открой он этот закон: «**Водку на Руси давно пили, и зачем там что-то еще изучать, ха-ха-ха**». А может быть, и тогда смеялись, но об этом ничего не известно. **Зато теперь над ним никто не смеется – победителей, ведь, не судят, правда?**».

Тот же Менделеев писал: «***Справедливость требует не тому отдать наибольшую научную славу, кто первым высказал истину, а тому, кто сумел убедить в ней других, показал ее достоверность и сделал ее применимой***», а **другой Дмитрий – Поспелов** говорил: «***В науке первым часто оказывается не тот, кто сказал «А», а тот, кто сказал «Я»***.

Третий пример. Вот что сказал в подтверждение этих слов Нобелевский лауреат **Жорес Алферов**: «**Герберт Кремер** (с ним Жорес Иванович разделил половину Нобелевской премии за разработку полупроводниковых гетероструктур, А.Ш.) **свои теоретические работы по гетероструктурам публиковал в 1950-е годы, намного раньше меня**. **Я стал заниматься ими только в конце 1962-го**. **Поэтому он пионер, но он теоретик, а мы пионеры тоже, но практики:** мы довольно рано поняли, что гетероструктуры в целом позволяют создавать принципиально новый класс материалов и на их основе можно иначе управлять потоками электронов и фотонов и создать новую электронику» (https://zen.yandex.ru/media/belrus/jores-alferov-ne-mog-je-ia-posle-etogo-skazat-chernomyrdinu-katis-so-svoim-domom-podalshe-604e3389011181447b9352f1).

Четвертый пример. Кто знает, мыли ли врачи руки (https://ru.wikipedia.org/wiki/Мытьё_рук) перед принятием родов – видимо, кто-то мыл, а кто-то и нет. В результате от родильной горячки в одной из лучших клиник Европы умирала каждая шестая женщина. **Игнац Земмельвайс (1818-1865)**, казалось бы, всего-то на всего, пытался **внедрить в медицину практику мытья рук и инструментов хлорной водой** (https://online812.ru/2012/04/12/011/). Его предложение опровергало сразу несколько догм, распространенных в медицине того времени, и поэтому большинство коллег категорически отказывались внедрять его практику. В конце концов, Земмельвайса без его согласия поместили в психиатрическую больницу, где он и умер (https://ru.wikipedia.org/wiki/Земмельвейс,_Игнац_Филипп), **но его предложение в конце концов победило!**

И, наконец, пятый пример. Писатель Людмила Улицкая как-то сказала: «**Понятие «ноосфера» придумал не Вернадский, но он создал учение о ноосфере**». Так и оказалось: «Понятие «ноосфера» было предложено профессором математики Сорбонны **Эдуардом Ле Руа (1870-1954)**, который трактовал её как «мыслящую» оболочку, формирующуюся человеческим сознанием. Э. Ле Руа подчеркивал, что пришел к этой идее совместно со своим другом – геологом и палеонтологом-эволюционистом, и католическим философом **Пьером Тейяром де Шарденом (1881-1955)**. При этом Леруа и Шарден основывались на лекциях по геохимии, которые в 1922/1923 гг. читал в Сорбонне **Владимир Иванович Вернадский (1863-1945)** (https://ru.wikipedia.org/wiki/Ноосфера), он же создал учение о ноосфере (https://ru.wikipedia.org/wiki/Вернадский,_Владимир_Иванович).

Еще раз повторю слова Аристотеля: «Известное известно немногим». Про тех, кто внезапно открывает для себя целесообразность применения автоматов в программировании после многолетней моей писанины об этом, я написал текст «**Конечный автомат многим не друг**» (https://vk.com/@1077823-konechnyi-avtomat-mnogim-ne-drug).

Далеко не все в автоматном мире стоит на месте. 25.11.2019 г. специалистами из Стэнфорда был введен новый класс нейронных сетей: *The Neural State Machine*. Это было сделано в статье: ***Hudson D., Manning C***. Learning by Abstraction: The Neural State Machine (https://arxiv.org/pdf/1907.03950.pdf).

А **08.10.2020 г.** в исследовательских недрах *Facebook* родился **текст о построении нейронных сетей на автоматах** – **Finite-State Transducers**. Статья про трансдьюсеры на русском размещена

здесь:
http://m.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ivm&paperid=6942&option_lang=rus.

Этот текст называется **«A New Open Framework for Automatic Differentiation with Graphs»** и **размещен по адресу**: https://ai.facebook.com/blog/a-new-open-source-framework-for-automatic-differentiation-with-graphs/. В нем есть ссылка на статью, опубликованную в *arXiv Cornel University*, которая называется «**Differentiable Weighted Finite-State Transdusers»** (https://arxiv.org/pdf/2010.01003.pdf), а сам *Framework* размещен по адресу: https://github.com/facebookresearch/gtn. Об этом мне рассказал наш выпускник 2006 г. **Артем Астафуров** (http://is.ifmo.ru/works/2008/Vestnik/53/20-declarative-nesting-and-inheritance-of-imperative-automata.pdf), работавший в то время в *Facebook*.

А вот работа про нейронные сети как деревья решений (https://arxiv.org/pdf/2210.05189.pdf).

Как говорится, будем наблюдать за применением автоматов в нейронных сетях и не только в них!

В декабре 2020 г. была опубликована статья ***Казакова Г.В., Корянова Вс.Вл., Чемирисова В.В., Уварова А.В.*** Методический подход к созданию универсального пользовательского интерфейса // Инженерный журнал: наука и инновации/ 2020. № 11, http://engjournal.ru/catalog/arse/itae/2034.html, в которой автоматное программирование используется для построения интерфейсов.

Ждать долго не пришлось. **29.06.2021 г.** *Microsoft* и *Open AI* представили *AI*-систему, названную *Github Copilot,* которая может давать советы по написанию кода программистам, что должно позволить сделать процесс программирования более доступным для освоения (https://habr.com/ru/news/t/565376/).

Инструмент использует исходный код, загруженный на сервис совместного использования кода *GitHub*, который *Microsoft* приобрела в 2018 г. В его реализации участвовал известнейший в мире стартап *Open AI*, занимающийся исследованиями в области искусственного интеллекта. Летом 2019 г. *Microsoft* инвестировала в него один миллиард долларов. *Github Copilot* является потомком мощной модели *GPT-3*, которую специалисты компании обучали на множестве терабайт общедоступного кода. Он способен практически полностью заменить программисту напарника, так как просматривает существующий код и комментарии к нему, а также местоположение курсора, и предлагает добавить одну или несколько строк в зависимости от контекста. По мере того, как программист принимает или отклоняет предложения, модель обучается и со временем становится все более сложной и умной. Уже сегодня сотни разработчиков на *GitHub* не отключают это средство в течение всего рабочего дня. В настоящее время инструмент лучше всего работает с *JavaScript*, *Python, TypeScript*, *Ruby* и *Go*.

А вот что, неожиданно для меня, в этот же день на своей странице в сети *Facebook* написал наш выпускник 2013 г. **Денис Чащин**: «*Github Copilot* – еще на один шаг к великому сокращению программистов. ***Кажется, добавь туда автоматное программирование, и готово***».

Так как Денис знаком, как с автоматным программированием, так и с *AI* со студенческих лет (http://is.ifmo.ru/genalg/labs_1/_chaschin.pdf), я попросил его пояснить сказанное. Вот, что он написал: «*Github Copilot* помогает дописывать код, когда понимает, что примерно нужно. Код, иногда, судя по скриншотам и анимациям на сайте, подставляется довольно большой, и выглядит примерно так же, как обычно сейчас пишут люди. При этом нет никакой гарантии ни по качеству кода, ни по тому, что он работает правильно, и даже неизвестно имеет ли он смысл (в документации прямо так и сказано).

Спрашивается, кто запрещает этому *AI* писать код не такой, как сейчас принято, а в разы лучше, скажем с использованием парадигмы автоматного программирования, чтобы бонусом получать верифицируемые программы и автоматически составлять для них тесты?

Приведу пример. Допустим, я описываю как визуально должна вести себя кнопка. *AI* понимает, что это кнопка, определяет какие у нее состояния (нажата, отпущена, длительное нажатие или что-нибудь еще и добавляет в код автомат с этими состояниями. После этого автоматически запускаются все автоматически построенные тесты, генерируются и отображаются скриншоты кнопки и ее поведения. Если где-то *AI* сомневается, то может запустить игру «Тиндер» (нравится/не нравится), что очень быстро развеет сомнения. Подобного рода «написание» кода может быть распространено

и на более сложные функциональности. По аналогии с автодополнением кода можно заодно и переписать и уже существующий».

При этом, правда, есть одна незадача: для того, чтобы *Github Copilot* использовала автоматы, она должна быть обучена на автоматных программах. А где много таких программ можно найти?

После предварительного знакомства с этим текстом упомянутый выше Константин Вавилов написал: «Лично для меня это Великая история с несомненным применением автоматов в различных разработках алгоритмов и ПО. На любую критику у меня есть конкретные примеры – автоматные программы, реально работающие на объектах Петербурга и России, эффективные, задокументированные, понятные (см. например, работы 2005 г. – http://is.ifmo.ru/automata/_metod065.pdf; http://is.ifmo.ru/automata/_s7300.pdf; http://is.ifmo.ru/automata/_vavilov2.pdf.zip)**». После этого я спросил Костю: «И сейчас применяешь?». Он ответил: «Да. По-другому уже не получается мыслить…».**

А вот, что по этому поводу пишет незнакомый мне человек: «**Большое Вам спасибо за Ваш труд, знания, которыми делитесь.** Когда я в начале 90-х учился по специальности «ЭВМ», нам рассказывали об аппаратной реализации автоматов. **Помню чувство восторга**, когда на втором курсе после долгого сидения в читальном зале над домашним заданием по электронике (надо было по входному импульсу изменяющейся длительности сформировать определенную последовательность импульсов, длительность которых должна была быть пропорциональна длительности входного импульса) я догадался, что это надо реализовать в виде автомата, разбив процесс на этапы – измерение входного импульса, запуск внутреннего генератора с нужной частотой, формирование выходного сигнала – выделив состояния автомата. Потом я догадался, что внутренний генератор надо делать не на регистре (для хранения длительности входного импульса) и счетчике, а просто на двух реверсивных счетчиках, считающих в противоположных направлениях и меняющих направление счета по достижению нуля, а формирователь выходного сигнала – это просто еще один автомат. **Как же мне тогда понравилась элегантность моего решения, как я им гордился :). А теперь, через 30 лет,** проработав все это время в связи, решил позаниматься ПЛК, управляющим несколькими контроллерами дизель-генераторной установки и выпрямительной системой с аккумуляторной батареей для автономного объекта, **и наткнулся на Ваши статьи и книги. Ещё раз – большое Вам спасибо**» (***Aleksander Bouksha***).

Недавно я узнал, что автоматы (https://docs.unrealengine.com/5.0/en-US/state-machines-in-unreal-engine/) применяются и в *Unreal Engine 5* (https://ru.wikipedia.org/wiki/Unreal_Engine) – игровом движке, разрабатываемом и поддерживаемом компанией *Epic Games*, на базе которого разработано немереное число игр (https://ru.wikipedia.org/wiki/Список_игр_на_движке_Unreal_Engine).

20.02.2023 г. на «Хабр» появилась статья «**С чем едят автоматы**» (https://habr.com/ru/companies/timeweb/articles/717628/), в которой я оказался в хорошей компании... При этом отмечу, что в статье первый портрет не Мура, как должно быть по тексту, а Шеннона, но от этого компания только улучшается. В статье есть такие слова: «И так, мы показали, как конечные автоматы используются в математике и электронике. Третье направление, где используются конечные автоматы – программирование. Идея рассматривать программы в терминах **конечных автоматов** сама по себе не нова. Но наиболее активно свое развитие она получила в начале 90-х годов прошлого века. Одним из основоположников данного подхода является **профессор Университета ИТМО Анатолий Абрамович Шалыто**. Подход основан на том, чтобы программировать с использованием понятия «состояние». Сперва для названия этого подхода появился термин «***Switch*-технология»**, так как операторы множественного выбора в традиционных языках программирования подходили для смены состояний программ больше всего. Позже, в конце 90-х термин «***Switch*-технологии»** был заменен на «**автоматное программирование**».

Интересно, что в статье ***Карпова В.Э., Воробьева В.В., Ровбо М.А.*** О некоторых аспектах применения автоматных моделей в групповом управлении // Мехатроника, автоматизация, управление. 2023. № 4, с. 171-180, https://mech.novtex.ru/jour/article/view/1349 **используется «мой термин» – «автоматное управление». Моя жизнь и жизнь автоматов продолжается…**

23.04.2024 г. я получил неожиданный подарок от неизвестного до этого времени мне **Сергея Муравьева** (smur@inbox.ru), который сообщил, что еще в 2019 г. на форуме «Робот с логикой на основе автоматного программирования» (http://roboforum.ru/forum10/topic18271.html) описал

процесс создания робота удаленного присутствия (https://vzikblog.wordpress.com/) **с использованием наших работ по автоматному программированию** (АП) на основе очень распространенных микроконтроллеров *ATmega 328* (обычно используются на платформе *Arduino*), которые можно купить на даже на «Озоне» и *Aliexpress*. В качестве ответа я написал текст «**Автоматное программирование. Не задушишь, не убьешь!**» (https://vk.com/@1077823-avtomatnoe-programmirovanie-ne-zadushish-ne-ubesh).

Продолжение этого текста в упомянутой выше работе «**О развитии автоматного программирования**» (https://vk.com/@1077823-o-razvitii-avtomatnogo-programmirovaniya, https://www.iae.nsk.su/ru/seminars-and-conferences/sem-isis/3035-220517-otklik-a-shalyto-na-seriyu-vebinarov-isis), а также здесь: https://vk.com/@1077823-pochemu-v-epohu-neironnyh-setei-dlya-upravleniya-otvetstvenn.
11.03.2021. https://vk.com/@1077823-esche-ob-avtomatnom-programmirovanii

О развитии автоматного программирования

Институт автоматики и электрометрии СО РАН и Институт систем информатики СО РАН им. А.П. Ершова проводят межинститутский семинар «**Инжиниринг современных информационных систем**» (https://www.iae.nsk.su/ru/seminars-and-conferences/sem-isis). Руководитель семинара – докт. техн. наук Владимир Евгеньевич Зюбин (https://www.iae.nsk.su/ru/laboratory-sites/lab-19, zyubin@iae.nsk.su).

15.03.2019 г. ***Владимир Иванович Шелехов*** (vshel@iis.nsk.su) из Института систем информатики сделал на этом семинаре доклад на тему: «**Язык требований в автоматном программировании**» (https://www.youtube.com/watch?v=IpZ9VlIcHYQ).

В начале докладчик рассказал о «первой производной» его исследований в области программирования – **предикатном программировании** (https://persons.iis.nsk.su/files/persons/pages/predbase.pdf).

Оно появилось как **отрицание теории схем-программ**, **структурного программирования** (избавление от оператора *go to* не привело к существенному улучшению процесса программирования) и **императивного программирования** (https://ru.wikipedia.org/wiki/Императивное_программирование). Функциональное программирование не заменяет императивное программирование, и это причина появления предикатного программирования. Однако в рамках последнего не все программы реализуются, так как **оно ограничено классом программ-функций**. Программа принадлежит классу **программ-функций**, если она не взаимодействует с внешним окружением. Точнее, если возможно перестроить программу таким образом, чтобы все операторы ввода данных находились в начале программы, а весь вывод собран в ее конце. *Класс программ-функций, по меньшей мере, содержит программы для задач дискретной и вычислительной математики*.

Если подобная перестройка программы принципиально невозможна, **ее следует определять как программу-процесс (реактивную систему),** для которой характерно аппаратно-программное окружение. К реактивным относятся системы управления. **Для реализации программ-процессов предназначено автоматное программирование**.

Указанные классы составляют более 90% всех программ. **Имеются другие более сложные типы программ**, например, **языковые процессоры** и **операционные системы.**

В начале двухтысячных **автоматное программирование стало** для Шелехова «**второй производной» его исследований**.

Я стал публиковаться по этой тематике на виду несколько раньше. Моя книга о *Switch*-технологии (http://is.ifmo.ru/books/switch/1) появилась в **1998 г.**, а в аннотации к моей статье: **Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления** // Известия РАН. Теория и системы управления. **2000**. № 6, с. 63-81 (https://www.avrorasystems.com/ru/Data/Pressroom/Files/ran.pdf) было сказано: «**Описываемая технология может быть названа автоматной технологией, а соответствующая область программирования – автоматным программированием**». Этот термин был предложен Дмитрием Александровичем Поспеловым, когда я рассказывал ему про этот подход **в 1997 г.** (https://vk.com/@1077823-vtomatnoe-programmirovanie).

Как, по мнению Шелехова, строится автоматное программирование (засечка 19.48)?
I. Автоматное программирование по Шалыто базируется на императивном программировании (языки *Java*, *C*, *C++*).
II. Автоматное программирование может быть основано функциональном программировании (язык *Erlang,* в котором в явном виде нет состояний, что приводит к тяжелому и противоестественному программированию, и язык «Рефлекс», http://reflex-language.narod.ru/)**.**

Мы пробовали этот вариант автоматного программирования, непосредственно используя функциональные языки программирования: **1.** ***Малаховски Я.М., Шалыто А.А.*** Конечные автоматы в чистых функциональных языках программирования. Автоматы и *Haskell* // *RSDN Magazin*e 2009. № 3, с. 20-26. https://rsdn.org/article/haskell/HaskellStateMachine.xml. **2.** ***Малаховски Я.М., Шалыто А.А.*** Реализация конечных автоматов на функциональных языках программирования // Информационно-управляющие системы. 2009. № 6, с. 30-33. http://www.i-us.ru/index.php/ius/article/view/14898). **3.** ***Малаховски Я.М., Корнеев Г.А.*** Валидация автоматов с переменными на функциональных языках программирования // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2010. № 6, с. 73-77. https://ntv.ifmo.ru/file/article/404.pdf. **4.** ***Малаховски Я.М., Шалыто А.А.*** Библиотека поддержки автоматного программирования для языка *Haskell* // Свидетельство о государственной регистрации программы для ЭВМ. № 2010 614196. Дата регистрации – 29.06.2010. **5.** ***Малаховски Я.М.*** Применение систем типов для валидации и верификации автоматных программ / Сборник «СПИСОК-2011». Материалы второй межвузовской научной конференции по проблемам информатики». СПб.: ВВМ. 2011, с. 368, 369. http://is.ifmo.ru/works/2011/SPISOK/13-Malahovski.pdf. Не мы, не докладчик это направление не развивали.
III. Шелехов, который не признает структурное программирование **и использует операторы перехода**, строит **автоматное программирование на базе предикатного программирования.** При этом «**третьей производной»** его исследований стал язык автоматного программирования. Перед ним встал вопрос: «Что такое спецификация автоматных программ?». Он пытался использовать теории процессов Хоара и Милнера. Оказалось, что для применения в технологии эти теории – тупик. Докладчик сказал, что попадал и в другие тупики. История науки – преодоление тупиков. Шелехов пришел к выводу: спецификация автоматных программ – требования к ним. Спецификация в виде требований переписывается в автоматную программу. Тем самым язык требований может быть использован в качестве языка автоматных программ.

При этом отмечу, что если в моем подходе спецификация – система графов переходов, то у Шелехова – это текст на языке требований, на котором, по словам автора, автоматная программа не всегда получается корректной, что требует трансформации требований – добавления новых ограничений, например, использующих темпоральную логику. Я считаю, что здесь в «полный рост» имеет место эвристика, а когда автор говорит, что его подход упрощает процесс построения программ, то это не очевидно.

Поясняя свой подход, Владимир Иванович говорит: «Картинка здесь тоже понятная», но картинок у него нет – только тексты на предложенном языке требований. При этом автоматная программа пишется, как говорит Шелехов, «**в стиле языка «Фортран**» и представляется в виде набора правил: **управляющее состояние *M*: <условие 1>, … <условие *n*> стрелка <действие 1>, … <действие *m*> #управляющее состояние *L*.** Правила выполняются последовательно.

Такие записи красивы, но Владимир Иванович, видимо, не знает, что Алиса говорила: «**Девочки любят картинки»**. Оказалось, что не только девочки, и особенно в тех случаях, когда картинки являются наглядными формальными спецификациями, по которым во многом видна динамика переходов автоматов. Шелехов, правда, отметил, что есть специалисты, которые более близки графические спецификации и в качестве примера привел не мой подход с графами переходов, а язык «Дракон» со схемами алгоритмов, построенными особым образом.

Еще один доклад на том же семинаре **Шелехов** сделал 01.04.2022 г. Его тема: «**Автоматное программирование на базе системы моделирования и верификации *Event-B***» (https://www.youtube.com/watch?v=la-DLQ4uf8M).

На первом слайде докладчик изложил, как, по его мнению, формировался понятийный базис программирования: «Становление теории программирования: **1.** Языки программирования.

Синтаксис. Семантика. **2.** Трансляторы. Операционные системы. Базы данных. **3.** Структурное программирование. **4.** Технологии программирования. **5.** Абстрактные типы данных. **6.** Объектно-ориентированное программирование. **7.** Понятие спецификации. Предусловие. Постусловие. **8.** Реактивные системы. **Автоматные методы**. **9. Автоматное программирование**. **10**. Формальные методы.

На засечке 1.44 Владимир Иванович говорит: «Термины «Реактивные системы» и «Автоматные методы» появились в девяностые годы. **Чуть позже возникло «автоматное программирование», причем появилось только у нас по инициативе Шалыто. Им было введено понятие «управляющее состояние»**. Шелехов тоже использует это понятие в «своем» автоматном программировании. **На засечке 8.28 снова появляется автоматное программирование по Шалыто**. На засечке 19.53 Шелехов среди способов кодирования автоматных программ рассматривает предложенную мною ***Switch***-технологию, **в которой не используется оператор *go to*, что позволяет реализовывать автоматные программы на языке *Java***, в частности.

Докладчик и в этом докладе отметил, что для автоматных программ методы Хоара и Флойда непригодны для дедуктивной верификации (https://ru.wikipedia.org/wiki/Верификация, https://ru.wikipedia.org/wiki/Формальная_верификация). Он рассмотрел **язык автоматного программирования**, построенный расширением языка спецификаций *Event-B* (http://www.event-b.org/), возникшего на базе модификации метода B (https://en.wikipedia.org/wiki/B-Method), предназначенного для разработки программного обеспечения критически важных систем безопасности и, в первую очередь, для метро и железнодорожного транспорта. Шелехов определил модель *Event-B* (***Abrial J.-R.*** Modeling in Event-B: System and Software Engineering. Cambridge University Press. 2010. 586 p.) для автоматного программирования. Докладчик описал технологию разработки и верификации автоматных программ на базе системы *Event-B*. Эта технология была апробирована на примере большой задачи управления движением на мосту из руководства по системе *Event-B*.

Свободно распространяемая платформа *Rodin*, созданная трудом многих (в основном европейских специалистов), является средой для разработки, анализа и верификации спецификаций при использовании метода *Event-B*. Платформа обеспечивает генерацию формул корректности инвариантов, поддерживает систему автоматического и интерактивного доказательства, содержит *Model Checker*, набор инструментов для доказательств, *SMT*-решатели (*SMT*-формула – это обобщение *SAT*-решателей, в которой переменные заменены предикатами из соответствующих теорий) и аниматор (https://www.ispras.ru/upload/iblock/5e5/5e5ac36633ead83d10476199d697be85.pdf).

Лекции и слайды к ним на русском языке по *Event-B* и *Rodin* приведены здесь: http://wasp.iis.nsk.su/page3.html.

По мнению докладчика, наличие таких мощных средств для рассматриваемого класса программ проведение верификации не гарантирует отсутствия ошибок (засечка 29.51).

К этому выводу мы пришли еще в 2013 г. Тогда **я и Владимир Ульянцев** написали статью «**О верификации простых программ со сложным поведением**» (http://is.ifmo.ru/works/2013/ulyantsev-shalyto-verification.pdf, https://vk.com/@1077823-o-verifikacii-prostyh-programm-so-slozhnym-povedeniem), в которой показали, что **качественно строить даже простые автоматные программы весьма сложно**. О программах других классов и говорить не приходится.

Автоматное программирование излагается Шелеховым на двух лекциях в курсе «**Формальные методы в программной инженерии**» (http://wasp.iis.nsk.su), который преподается на механико-математическом факультете Новосибирского государственного университета. На первой из них (видео, http://wasp.iis.nsk.su/video/fap_lection1.mp4; слайды, http://wasp.iis.nsk.su/pres/fap_lecture1.pdf) Шелехов утверждает, что для систем реального времени нет единого подхода к их программной реализации. По его мнению, автоматное программирование в этой области играет особую роль. **Он вновь подтведил, что оно впервые было предложено в России, и его родоначальником был А. Шалыто, когда лет двадцать назад предложил технологию автоматного программирования, названную им «*Switch*-технология»** (засечка 22.40).

В качестве одного из примеров создания автоматных программ (вторая лекция (http://wasp.iis.nsk.su/video/fap_lection2.mp4, http://wasp.iis.nsk.su/pres/fap_lecture2.pdf) целиком

посвящена примерам построения автоматных программ на основе его подхода) Шелехов использует электронные часы с будильником (засечка 39.58) **из моей с Н. Поликарповой книги** «Автоматное программирование» (http://is.ifmo.ru/books/_book.pdf).

При этом он по словесному описанию строит автоматную программу в предлагаемом им стиле языка «Фортран», а потом для сравнения – на основе *Switch*-технологии, и делается вывод, что его программа более проста и понятна, а **то, что моя программа построена формально и изоморфно по графу переходов,** Шелехов не учитывает, так как графы переходов в качестве языка спецификации программ, в отличие от меня, он не использует.

При этом отмечу, что при применении программного средства *Stateflow* (https://www.mathworks.com/products/stateflow.html) ***графы переходов являются не только языком спецификации, но и языком программирования автоматных программ. Это позволяет генерировать их текстовое представление, которое может быть автоматически (без участия человека) загружено в среду исполнения.*** Видеоролик приведен ниже.

Публикации В.И. Шелехова:

1. ***Шелехов В.И.*** Язык и технология автоматного программирования // Программная инженерия. 2014. № 4, с. 3-15 (https://persons.iis.nsk.su/files/persons/pages/automatProg.pdf). В этой работе есть ссылки на две мои книги (http://is.ifmo.ru/books/switch/1, http://is.ifmo.ru/books/_book.pdf) и такой фрагмент: «**Концепция автоматного программирования разработана *Анатолием Шалыто***, в том числе в интеграции с объектно-ориентированным программированием. Автоматная программа определяется как совокупность конечных автоматов. Используются графическое и текстовое представления программы. **Управляющие состояния** являются значениями переменной, соответствующей этим состояниям. При реализации автоматной программы применяется ***Switch*-технология**. ***Термин «автоматное программирование» и его аналог Automata-Based Programming применяется только в России*.** Тем не менее, автоматные методы программирования заложены во многих языках**. Автор благодарен *А.А. Шалыто* за его работы по автоматному программированию.** Предлагаемое мною понятие автоматной программы концептуально не отличается от введенного Анатолием Шалыто, однако различия в языке и технологии существенны».

2. ***Шелехов В.И.*** Разработка автоматных программ на базе определения требований // Системная информатика. 2014. № 4, с. 1-29 (http://persons.iis.nsk.su/files/persons/pages/req_tech.pdf. Есть ссылки на две указанные выше мои книги и на один наш курсовик, ссылка на который приведена ниже. Работа заканчивается так: «**Автор благодарен А.А. Шалыто за работы по автоматному программированию, стимулировавшие мои исследования**».

3. ***Шелехов В.И.*** Оптимизация автоматных программ методом трансформации требований // Программная инженерия. 2015. № 11, с. 3-13. http://persons.iis.nsk.su/files/persons/pages/req_k.pdf. Наши работы не упоминаются.

4. ***Шелехов В.И.*** Классификация программ, ориентированная на технологию программирования // Программная инженерия. 2016. № 12. 2016, с. 531-538. https://persons.iis.nsk.su/files/persons/pages/prog.pdf. Автоматные программы в статье упоминаются, а мы – нет.

5. ***Шелехов В.И., Тумуров Э.Г.*** Технология автоматного программирования на примере программы управления лифтом // Программная инженерия. 2017. № 3, с. 99-111 (https://persons.iis.nsk.su/files/persons/pages/lift1.pdf). В этой работе есть три ссылки на наши работы – на два курсовика наших студентов (*Наумов А.С., Шалыто А.А.* Система управления лифтом. 2003 (http://is.ifmo.ru/download/elevator_a.pdf), *Решетников Е.О., Смачных М.В.* Система управления пассажирским лифтом. 2006 (http://is.ifmo.ru/download/umlift.pdf) и мою книгу *Switch*-технология (http://is.ifmo.ru/books/switch/1), опубликованную почти двадцать лет до публикации этой статьи – в 1998 г. Интересно, что еще одну нашу работу про программирование лифта авторы статьи не заметили: ***Наумов Л.А., Шалыто А.А.*** Искусство программирования лифта. Объектно-ориентированное программирование с явным выделением состояний // Информационно-управляющие системы. 2003. № 6, с. 38-49. http://www.i-us.ru/index.php/ius/article/view/14406.

6. Шелехов В.И. Сравнение технологий автоматного программирования и *Event-B* // Системная информатика. 2021. № 18, с. 53-84. https://system-informatics.ru/files/article/bridgesi.pdf. Имеется ссылка на мою книгу: http://is.ifmo.ru/books/switch/1.

Платформу ***Rodin*** для верификации автоматных программ, специфицированных графами переходов, использует и специалист из Кургана **Максим Нейзов** (neyzov.max@gmail.com). 16 марта

2021 г. он на том же семинаре выступил с докладом на тему «**Опыт верификации автоматных программ на платформе *Rodin* (на примере задачи управления генератором эндогаза)**» (https://www.youtube.com/watch?v=Miqi4DWqS8o&t=303s).

Максим по этой теме опубликовал следующие работы:
1. *Нейзов М.* Формальный дедуктивный анализ автоматного алгоритма управления генератором эндогаза с помощью платформы *Rodin*. Часть 1. Определение требований надежности и безопасности работы генератора эндогаза // Современная электроника. 2020. № 9, с. 62, 63 (https://www.iae.nsk.su/images/stories/2_Science/5_Seminars-Conf/4_EngModernCompSys/210316-NeizovM/Neizov_part1.pdf). Формальный дедуктивный анализ представляет собой строгий математический подход к верификации алгоритмов: алгоритм описывается с помощью аксиом, а требуемые свойства доказываются как теоремы. Цель представленного анализа – доказать соответствие алгоритма управления предъявляемым требованиям надежности и безопасности. В статье описывается технологический процесс генерации эндогаза и определяются предъявляемые к нему требования надежности и безопасности.
2. *Нейзов М.* Формальный дедуктивный анализ автоматного алгоритма управления генератором эндогаза с помощью платформы *Rodin*. Часть 2. Алгоритм управления и платформа *Rodin* // Современная электроника. 2021. № 1, с. 44-47 (https://www.iae.nsk.su/images/stories/2_Science/5_Seminars-Conf/4_EngModernCompSys/210316-NeizovM/Neizov_part2.pdf). В статье представлен алгоритм управления **в виде системы взаимосвязанных автоматов, заданных некоторой разновидностью графов переходов, и платформа *Rodin*** – инструмент для формального анализа систем и автоматизации доказательства теорем. В этой части статьи есть ссылки на три наши работы. В работе используются и предложенные мною схемы связей автоматов с их окружением.
3. *Нейзов М.* Формальный дедуктивный анализ автоматного алгоритма управления генератором эндогаза с помощью платформы *Rodin*. Часть 3. Построение формальной теории для алгоритма управления // Современная электроника. 2021. № 2, с. 40-43 (https://www.iae.nsk.su/images/stories/2_Science/5_Seminars-Conf/4_EngModernCompSys/210316-NeizovM/Neizov_part3.pdf). В статье выполнено построение формальной теории для рассматриваемого алгоритма управления: проведена аксиоматизация алгоритма, формализованы требования, продемонстрировано доказательство теорем с помощью платформы *Rodin*.

11 ноября 2021 г. **Максим** выступал там же с докладом на тему «**Автоматно-функциональная парадигма программирования реактивных систем**» (https://www.youtube.com/watch?v=iyw98xpsJgw), где **он попытался недостатки автоматной парадигмы заменить достоинствами функциональной парадигмы и наоборот**.

Шестого мая 2022 г. Нейзов сделал доклад на тему «**Автоматно-функциональный подход к моделированию киберфизических систем (на примере *Event-B* модели для верификации свойств алгоритма управления движением автомобилей на мосту)**», в котором рассказал как с помощью его подхода строить модели для указанного класса систем корректных алгоритмов управления методом доказательства теорем (https://www.youtube.com/watch?v=891jZr-7Mos&t=6s).

Третьего июня Максим Нейзов сделал на семинар доклад на тему «Виды вложенности автоматов: формальная семантика (в рамках автоматно-функционального подхода)» (https://www.youtube.com/watch?v=RevSOB8_OBc). При этом на засечке 13.07 автор рассматривает как осуществляется вложенность в ***Switch*-технологии**.

А теперь несколько слов об одной из последних работ Зюбина. В 2020 г. под его руководством был разработан **текстовый язык** для спецификации управляющего программного обеспечения *poST* (процесс-ориентированное расширение языка *IEC 61131-3 Structured Text*). Язык сочетает преимущества **программирования на основе конечных автоматов** с синтаксисом языка структурированный текст для программируемых логических контроллеров.

Вот публикации по этому вопросу:
1. *Bashev V., Anureev I., Zyubin V.* The Post Language: Process-Oriented Extension for IEC 61131-3 Structured Text / 2020 International Russian Automation Conference (RusAutoCon). Sochi. Russia. 2020, pp. 994-999, https://ieeexplore.ieee.org/document/9208049. Вот аннотация этой статьи: «This paper introduces a new programming language **for control software specification**. The language called poST is

a process-oriented extension of the IEC 61131-3 **Structured Text language** widely used in the PLC domain. The poST language enables control software specification as a set of interacting **FSM-based** processes that have event-driven behaviour and operate with time intervals. The language is intended to provide a possibility to use the process-oriented approach for IEC 61131-3 users and comparing to the other process-oriented languages poST is easy to learn for the IEC 61131-3 community. An IDE for poST was developed with Eclipse (Xtext) toolset. Paper illustrates the poST language using for a hand dryer control software: **we provide the source poST code and the generated C code for Arduino** (ATmega 168) platform».
2. ***Zyubin V., Rozov A, Anureev I., Garanina N., Vyatkin V.*** poST: A Process-Oriented Extension of the IEC 61131-3 Structured Text Language // IEEE Access. 2022. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9729833. Здесь есть две ссылки на наши работы.

Подходы, изложенные выше, развивает автоматное программирование и, возможно, кто-то из читателей настоящего текста захочет их опробовать и начать взаимодействовать с их авторами.

Кратко напомню о моем подходе к автоматному программированию. О нем можно прочитать в работах: «Автоматное программирование» (https://vk.com/@1077823-vtomatnoe-programmirovanie) и «Еще об автоматном программировании» (https://vk.com/@1077823-esche-ob-avtomatnom-programmirovanii).

В заключение расскажу о наиболее широко внедренном направлении «моей» технологии автоматного программирования.

Еще в 2009 г. «с моей подачи» в ОАО «НПО «Аврора» *Юрий Янкин* начал использовать **автоматное программирование** для написания программ, реализуемых программируемыми логическими интегральными схемами (ПЛИС) (http://is.ifmo.ru/works/_jankin.pdf).

Янкиным и мною на основе автоматного программирования была разработана технология создания программного обеспечения модулей, выполненных на основе ПЛИС (http://is.ifmo.ru/present/2012/Yankin-Shalyto-PLIS.exe). При этом отмечу, что перед запуском эта презентация должна быть скачана. А вот то же «кино», представленное иначе: https://www.youtube.com/watch?v=YNWdmnwHZi8&t=29s.

Предложенная технология описана в следующих статьях: **1.** ***Янкин Ю.Ю., Шалыто А.А.*** Автоматное программирование ПЛИС в задачах управления электроприводом // Информационно-управляющие системы. 2011. № 1, с. 50-56. http://www.i-us.ru/index.php/ius/article/view/13825. **2.** ***Янкин Ю.Ю., Шалыто А.А.*** Применение автоматного подхода при программировании модулей управления шаговыми двигателями, выполненными на основе ПЛИС // Системы управления и обработки информации. 2011. Вып. 22, с. 92-103. https://elibrary.ru/item.asp?id=29426972. **3.** ***Янкин Ю.Ю., Шалыто А.А.*** Метод создания программного обеспечения модулей, выполненных на основе программируемых логических интегральных схем // Системы управления и обработки информации. 2013. Вып. 26, с. 128-135. http://is.ifmo.ru/works/2013/yankin-ntkms.pdf. **4.** ***Янкин Ю.Ю., Шалыто А.А.*** Разработка резервированного блока управления электроприводом на основе автоматного подхода // Научно-технический вестник информационных технологий, механики и оптики. 2014. № 6 (94), с. 146-152. http://is.ifmo.ru/works/2014/yankin-control-block.pdf.

С момента начала разработки технологии прошло двенадцать лет, и 28.10.2021 г. в диссертационном совете при ОАО «Концерн «НПО «Аврора» мой аспирант ***Ю.Ю. Янкин*** защитил кандидатскую диссертацию на тему **«Технология разработки программного обеспечения автоматизированных систем управления электроприводом регулирующих органов судовых установок»**.

Несмотря на то, что эта технология базируется на покупном программном средстве ***Stateflow***, технологии создания ПО для рассматриваемого класса систем, реализуемых на ПЛИС, не было, а теперь она есть. Под моим руководством в Университете ИТМО уже защищена не одна диссертация про автоматное программирование (http://is.ifmo.ru/disser/), но, чтобы **внедрить этот подход в огромное число систем, такого даже близко не было**. Трудно себе представить то число комиссий, которому Юрию пришлось сдавать эти системы на различных заказах. Когда на защите ему задали вопрос о научности работы, то мне показалось, что спрашивающий забыл утверждение Маркса: «Практика – критерий истины», о чем я в своем выступлении сказать не забыл :-). Кроме того,

отвечая на вопросы, Юрий пообещал в ближайшие годы продолжить внедрять автоматный подход в судовые системы этого класса с той же интенсивностью. **В заключительном слове Юра благодарил судьбу за встречу со мной**.

В своем выступлении член Совета **Б.В. Грек** сказал, что ему однажды **два часа** сотрудник НПО «Аврора» **Сергей Ванюшкин восторженно рассказывал об опыте применении автоматного программирования при создании ПО для судовых дизель-генераторов** на программируемых логических контроллерах разных фирм ***при условии, что он не до не после программировать в классическом смысле этого слова не умел***. Другой член совета – **В.И. Поленин –** вспомнил, что автоматный подход уже много лет применяет еще один наш сотрудник – **Владимир Волобуев** при создании ПО для информационно-управляющих систем, реализуемых на управляющих вычислителях (http://is.ifmo.ru/works/_volobuev.pdf). Член совета **И.Р. Францев** сказал, что у него в отделе под руководством еще одного моего аспиранта **Антона Калачинского** разработано инструментальное средство для поддержки автоматного программирования, которое широко используется для систем управления газотурбинными установками (https://www.avrorasystems.com/upload/iblock/cf3/cf3801161b1ca1a2fa7ba969079c45ec.pdf).

«Автоматный подход применительно к разработке ПО для судовых систем управления был предложен мною еще в 1991 г. В том же году произошло его первое внедрение – при создании системы управления дизель-генератора ДГР 500*500 судна проекта 15760 на базе аппаратуры *Selma-2* фирмы *ABB Stromberg*. Программирование выполнялось на языке функциональных блоков, адаптированном под автоматное программирование» (http://is.ifmo.ru/books/switch/1). Несмотря на это, первая диссертация по автоматной тематике в НПО «Аврора» – диссертация Янкина – была защищена только через 30 (!) лет после этого события. **Не зря говорят, что в России надо жить долго!**

Перечень работ, посвященных формальной верификации «моих» автоматных программ, приведен в разделе 20 статьи «Автоматное программирование» (https://vk.com/@1077823-vtomatnoe-programmirovanie). При этом отмечу, что динамическую верификацию в форме тестирования для автоматных программ никто не отменял. При сдаче систем с автоматными программами они многократно верифицируются таким образом.

Продолжение этой статьи – «**Еще раз об автоматном программировании**» (https://vk.com/@1077823-esche-ob-avtomatnom-programmirovanii).

12.05.2022. https://vk.com/@1077823-o-razvitii-avtomatnogo-programmirovaniya, https://www.iae.nsk.su/ru/seminars-and-conferences/sem-isis/3035-220517-otklik-a-shalyto-na-seriyu-vebinarov-isis

Автоматы, нейронные сети и будущее программирования

25.11.2019 г. специалистами из Стэнфорда был введен новый класс нейронных сетей: *The Neural State Machine*. Это было сделано в статье: ***Hudson D., Manning C***. Learning by Abstraction: **The Neural State Machine** (https://arxiv.org/pdf/1907.03950.pdf).

А **08.10.2020 г.** в исследовательских недрах *Facebook* родился **текст о построении нейронных сетей на автоматах** – *Finite-State Transducers.* Статья про трансдьюсеры на русском размещена здесь: http://m.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ivm&paperid=6942&option_lang=rus.

Этот текст называется ***A New Open Framework for Automatic Differentiation with Graphs*** и **размещен по адресу***:* https://ai.facebook.com/blog/a-new-open-source-framework-for-automatic-differentiation-with-graphs/. В нем есть ссылка на статью, опубликованную в *arXiv Cornel University*, которая называется ***Differentiable Weighted Finite-State Transdusers*** (https://arxiv.org/pdf/2010.01003.pdf), а сам *Framework* размещен по адресу: https://github.com/facebookresearch/gtn. Об этом мне рассказал наш выпускник 2006 г. **Артем Астафуров** (http://is.ifmo.ru/works/2008/Vestnik/53/20-declarative-nesting-and-inheritance-of-imperative-automata.pdf), работающий в *Facebook*.

Как говорится, будем наблюдать за применением автоматов в нейронных сетях и не только в них!

Ждать долго не пришлось. **29.06.2021 г.** *Microsoft* и *Open AI* представили *AI*-систему, названную *Github Copilot,* которая может давать советы по написанию кода программистам, что должно

позволить сделать процесс программирования более доступным для освоения (https://habr.com/ru/news/t/565376/).

Инструмент использует исходный код, загруженный на сервис совместного использования кода *GitHub*, который *Microsoft* приобрела в 2018 г. В его реализации участвовал известнейший в мире стартап *Open AI*, занимающийся исследованиями в области искусственного интеллекта. Летом 2019 г. *Microsoft* инвестировала в него один миллиард долларов. *Github Copilot* является потомком мощной модели *GPT-3*, которую специалисты компании обучали на множестве терабайт общедоступного кода. Он способен практически полностью заменить программисту напарника, так как просматривает существующий код и комментарии к нему, а также местоположение курсора, и предлагает добавить одну или несколько строк в зависимости от контекста. По мере того, как программист принимает или отклоняет предложения, модель обучается и со временем становится все более сложной и умной. Уже сегодня сотни разработчиков на *GitHub* не отключают это средство в течение всего рабочего дня. В настоящее время инструмент лучше всего работает с *JavaScript*, *Python, TypeScript*, *Ruby* и *Go*.

А вот что, неожиданно для меня, в этот же день на своей странице в сети *Facebook* написал наш выпускник 2013 г. **Денис Чащин**: «*Github Copilot* – еще на один шаг к великому сокращению программистов. ***Кажется, добавь туда автоматное программирование, и готово***».

Так как Денис знаком, как с автоматным программированием, так и с *AI* со студенческих лет (http://is.ifmo.ru/genalg/labs_1/_chaschin.pdf), я попросил его пояснить сказанное. Вот, что он написал: «*Github Copilot* помогает дописывать код, когда понимает, что примерно нужно. Код, иногда, судя по скриншотам и анимациям на сайте, подставляется довольно большой, и выглядит примерно так же, как обычно сейчас пишут люди. При этом нет никакой гарантии ни по качеству кода, ни по тому, что он работает правильно, и даже неизвестно имеет ли он смысл (в документации прямо так и сказано).

Спрашивается, кто запрещает этому *AI* писать код не такой, как сейчас принято, а в разы лучше, скажем с использованием парадигмы автоматного программирования, чтобы бонусом получать верифицируемые программы и автоматически составлять для них тесты?

Приведу пример. Допустим, я описываю как визуально должна вести себя кнопка. *AI* понимает, что это кнопка, определяет какие у нее состояния (нажата, отпущена, длительное нажатие или что-нибудь еще и добавляет в код автомат с этими состояниями. После этого автоматически запускаются все автоматически построенные тесты, генерируются и отображаются скриншоты кнопки и ее поведения. Если где-то *AI* сомневается, то может запустить игру «Тиндер» (нравится/не нравится), что очень быстро развеет сомнения. Подобного рода «написание» кода может быть распространено и на более сложные функциональности. По аналогии с автодополнением кода можно заодно и переписать и уже существующий».

Я всегда знал, что с выпускниками классно дружить...

А теперь одна незадача: для того, чтобы ***Github Copilot использовала автоматы, она должна быть обучена на автоматных программах. А где их много можно найти?***

30.06.2021. https://vk.com/@1077823-avtomaty-neironnye-seti-i-buduschee-programmirovaniya

О приоритете

Известно, что лет семьдесят назад считалось, что все главное в мире создано или в СССР, или в России, а остальное подло украдено у нас. «Это должен был запомнить рядовой гражданин Империи, после чего – **немедленно лопнуть от гордости**» (https://u-96.livejournal.com/129636.html). Подтверждение сказанному книга ***Болховитинов В.Н., Буянов А.Ф., Захарченко В.Д., Орлов В.И.*** Рассказы о русском первенстве. М.: Молодая гвардия. 1950, 424 с. (https://www.ozon.ru/product/rasskazy-o-russkom-pervenstve-28708046/?sh=V-POzuP-). Кратко ее содержание можно сформулировать так: «Россия – родина слонов». Эту книгу можно прочесть целиком здесь: https://disk.yandex.ru/d/5BTBra7lSPEG3.

Вопрос о приоритете в чем либо (даже в достаточно узкой области) весьма сложный – об этом, в частности, и мой текст «**Автоматное программирование, водка и буква Ё**» (http://is.ifmo.ru/download/2008-03-17_automata.pdf), опубликованный в начале 2008 г. Приведу **два примера** из нее.

Первый: «***Довод Шалыто*** к тому, что именно он оформил программирование состояний в некую **стройную систему, которую он назвал «автоматным программированием»** и именно в этом его заслуга, ***вполне оправдан***. Поэтому и говорят, что автоматное программирование изобрел Шалыто. **Это как с буквой Ё. Мало кто знает что её придумала Екатерина Дашкова, но все знают что первым применять её начал Николай Карамзин, поэтому зачастую говорят, что буква Ё – буква Карамзина**» (https://alexott.livejournal.com/321730.html?thread=1717698#t1717698). Это написал человек, с которым я не был знаком ни тогда, ни теперь.

Второй пример: «Есть история про исследования раствора ... из воды и спирта, которые проводил **Дмитрий Менделеев**, до открытия им Периодического закона. Представляете, как бы над ним некоторые сейчас смеялись, не открой он этот закон: «**Водку на Руси давно пили, и зачем там что-то еще изучать, ха-ха-ха**». А может быть, и тогда смеялись, но об этом ничего не известно. **Зато теперь над ним никто не смеется – победителей, ведь, не судят, правда?**».

Тот же Менделеев писал: «***Справедливость требует не тому отдать наибольшую научную славу, кто первым высказал истину, а тому, кто сумел убедить в ней других, показал ее достоверность и сделал ее применимой***».

Теперь третий пример. Вот что сказал в подтверждение этих слов Нобелевский лауреат **Жорес Алферов**: «**Герберт Кремер** (с ним Жорес Иванович разделил половину Нобелевской премии за разработку полупроводниковых гетероструктур, А.Ш.) **свои теоретические работы по гетероструктурам публиковал в 1950-е годы, намного раньше меня**. **Я стал заниматься ими только в конце 1962-го**. **Поэтому он пионер, но он теоретик, а мы пионеры тоже, но практики:** мы довольно рано поняли, что гетероструктуры в целом позволяют создавать принципиально новый класс материалов и на их основе можно иначе управлять потоками электронов и фотонов и создать новую электронику» (https://dzen.ru/a/YE4ziQERgUR7k1Lx).

Четвертый пример. Кто знает, мыли ли врачи руки (https://ru.wikipedia.org/wiki/Мытьё_рук) перед принятием родов – видимо, кто-то мыл, а кто-то и нет. ***В результате от родильной горячки в Вене в одной из лучших клиник Европы умирала каждая шестая женщина***. **Игнац Земмельвайс**, казалось бы, всего-то на всего, пытался **внедрить в медицину практику мытья рук и инструментов хлорной водой** (https://online812.ru/2012/04/12/011/). Его предложение опровергало сразу несколько догм, распространенных в медицине того времени, и поэтому большинство коллег категорически отказывались внедрять его практику. Более того его подняли на смех и запретили публиковать результаты исследований. В конце концов, Земмельвайса без его согласия поместили в психиатрическую больницу, где он и умер (https://ru.wikipedia.org/wiki/Земмельвейс,_Игнац_Филипп), но его предложение в конце концов победило! Однако, как долго пришлось ждать, чтобы здравая идея стала нормой. **Земмельвайс сделал свое открытие в 1840-х годах, но только через 20 лет интуитивное предположение врача нашло научное подтверждение благодаря исследованиям Л. Пастера и Р. Коха**.

И, наконец, пятый. Писатель **Людмила Улицкая** как-то сказала: «**Понятие «ноосфера» придумал не Вернадский, он создал учение о ноосфере**». Так и оказалось: «Понятие «ноосфера» было предложено профессором математики Сорбонны **Эдуардом Леруа**, который трактовал её как «мыслящую» оболочку, формирующуюся человеческим сознанием. Э. Леруа подчеркивал, что пришел к этой идее совместно со своим другом – геологом и палеонтологом-эволюционистом, и католическим философом **Пьером Тейяром де Шарденом**. При этом Леруа и Шарден основывались на лекциях по геохимии, которые в 1922/1923 гг. читал в Сорбонне **Владимир Иванович Вернадский** (https://ru.wikipedia.org/wiki/Вернадский,_Владимир_Иванович), который создал учение о ноосфере (https://ru.wikipedia.org/wiki/Ноосфера).

Всегда помните слова Аристотеля: «Известное известно немногим». Про и для тех, кто внезапно открывает для себя целесообразность применения автоматов в программировании после многолетней моей писанины об этом, я написал текст «**Конечный автомат многим не друг**» (https://vk.com/@1077823-konechnyi-avtomat-mnogim-ne-drug).

Теперь авторитетное мнение о том, кому должна принадлежать слава в борьбе за постижение истины: ***«Справедливость требует отдать наибольшую научную славу, не тому, кто первым высказал истину, а тому, кто сумел убедить в ней других, показал ее достоверность и сделал ее применимой»*** **(Д. Менделеев).**

Еще одно мнение о приоритете. «***Не так важно кто нечто придумал первым – важно, кто начинает это использовать не от случая к случаю, а как технологию.*** Как делаются общественно-политические изобретения? Например, **само по себе изобретение паровой машины еще не вело к промышленной революции – нечто подобное создавалось и раньше в других странах Европы**. ***Заслуга Джеймса Уатта была не в том, что он изобрел паровую машину, а в том, что сумел продать первые сто экземпляров***. Это и есть **опыт цивилизации** – когда ***открытие, возникшее исторически случайно, начинает использоваться сознательно***, как технология, и воспроизводится в дальнейшем опыте» (В. Лейбин). Я считаю, что **такое постепенно происходит и с автоматным программированием.**

О первых. «**А первые оказываются первыми, как это ни странно, потому что они были первыми с самого начала. *Не потому, конечно, что родились ими, а потому, что наиболее непосредственно, наиболее впрямую, наиболее единственно были заняты своим делом – только своим и только этим.*** Такие люди, естественно, оказываются в выигрыше: во времени, в результативности, в производительности, – потому что с легкостью берут от себя все, на что способны. Никакими усилиями, никакой волей не заставит себя человек сделать что-либо себе чуждое и несвойственное лучше того, что он сделал бы сам, по собственному желанию и без принуждения. В первом случае он несвободен, во втором – свободен, а, как известно, самая низкая производительность – у раба. Таким образом, чемпион – это еще и категория свободы. (Недаром гладиаторам-победителям был приз – свобода.) Люди, понимающие и чувствующие живой и цельный организм своего дела, люди, всего лишь свободно и до конца занятые тем, чем они заняты, способны настолько полно выявить заложенные в них от природы возможности, что вдруг оказываются первыми, чем и вызывают всеобщее восхищение (или зависть, А.Ш)» (**А. Битов**, из главы «Колесо. Записки новичка» книги «Путешествия по Империи», АСТ, https://royallib.com/book/bitov_andrey/kniga_puteshestviy_po_imperii.html. Книга была подготовлена к изданию в 1991 г., однако увидела свет только в 2000 г.,). В **1968 г.** на занятиях по философии я узнал о существовании писателя Андрея Битова, книги которого »Дачная местность» и «Аптекарский остров», произвели на меня тогда очень сильное впечатление.

Еще о первых. «**В науке первым часто оказывается не тот, кто сказал «А», а тот, кто сказал «Я» (Д. Поспелов). Помните это…**

Теперь несколько дополнительных замечаний по рассмотренному вопросу.

1. Известный ученый **Николай Николаевич Непейвода** в книге **Стили и методы программирования**. М.: Интернет-Университет Информационных технологий. 2005. 316 с. написал: «**Термин «автоматное программирование» принадлежит, насколько нам известно, А. Шалыто**. Во всяком случае, ему принадлежит заслуга в его развитии **вопреки моде и мнению большинства**». Все это осталось и в изданиях этой книги, появившихся в 2012 и 2016 гг.

2. Вот комментарий на этот текст Родиона Юрьева: «Многие начинающие предприниматели тщательно оберегают свои идеи, ошибочно полагая, что они чего-то стоят. Идеи не стоят ничего без реализации (говорят они стоят процентов пять, если они успешно реализованы, А.Ш.), причем не реализации «вообще», а воплощения в виде бизнеса, да еще и успешного. Повсеместно, как только что-то оказывается успешным, начинаются пересуды: это-де изобрели в Древнем Египте, а это знали еще индийцы или индейцы. **Ну так если это все знали, кто ж вам мешал это реализовать?!** Поэтому рассуждать о том, кто там первым что-то упомянул об автоматном программировании бессмысленно, **его отцом уже является А.А. Шалыто и точка**, потому что именно он его развил и реализовал в новое направление».

3. 12.11.2019 г. я опубликовал текст «**Мертвые и живые**» (http://d-russia.ru/myortvye-i-zhivye.html). В ходе обсуждения этого текста **Вадим Гор** написал: «Пользуясь случаем, **хочу засвидетельствовать свое почтение Анатолию Шалыто за изобретение *Switch*-технологии! Чудесная вещь**. Она близка к системной методологии, где имеет прямое отношение к моделированию доминант и детерминант в сложных системах. В молодости я с очень большим воодушевлением ее использовал в своей работе. Да и сейчас – тоже. **На всех программистских специальностях ее надо изучать сразу, чтобы мозги у программистов сразу вставали на место**. Потом их ставить труднее».

После этого я сообщил Вадиму, что **выложил на *YouTube* лекцию по автоматному программированию** (https://www.youtube.com/watch?v=PPWTxceMutk). На это он написал мне: «Все очень правильно говорите, начиная от того, что «**графы должны быть по возможности планарны, а схемы красивы**». Автоматная модель – это несравнимо лучше превентивного кодерства».

18 ноября этого же года **Гор написал** текст «**О спорах относительно парадигм программирования**», в котором, в частности, сказано: «Желание написать такой материал навеяло автоматное программирование (*Switch*-технология) от Анатолия Шалыто, предложившего его. Дело, конечно, не в открытии многоуровневых конечных автоматов для программирования, а в их удобном технологическом оформлении, сильно упрощающем как макетирование систем до непосредственного программирования на языках программирования, так и верификацию кода, вместе множеством достоинств, связанных с проблемами согласования алгоритмов с заказчиками программ. **Автоматное программирование вполне совместимо с объектно-ориентированным программированием** (ООП) и является средством построения «крепких» каркасов ООП-программ и, естественно, функционального кода тоже. Следовательно, хотелось бы раскрыть «Настоящее ООП» = ООП + Каркасное программирование + Шаблоны проектирования + Автоматное программирование, не противопоставляя его функциональному программированию, а просто показав приверженцам функционального подхода место последнего в указанной троице (каркасы и шаблоны по большому счету одно и то же, только на разных масштабах архитектуры)». Этот текст можно найти по дате на моей странице https://www.facebook.com/anatoly.shalyto.

4. 16.02.2021 г. я опубликовал текст «Автоматное программирование» (https://vk.com/@1077823-vtomatnoe-programmirovanie), а 11 марта – его продолжение: «**Еще об автоматном программировании**» (https://vk.com/id1077823?z=article_edit1077823_62364). Упомянутый там **Константин Вавилов** после их прочтения написал: «**Лично для меня это Великая история** и несомненное применение в различных разработках алгоритмов и ПО. **На любую критику есть конкретные примеры – реально работающие на объектах Петербурга и России, эффективные, задокументированные, понятные** (см. например, работы 2005 г. – http://is.ifmo.ru/automata/_metod065.pdf; http://is.ifmo.ru/automata/_s7300.pdf; http://is.ifmo.ru/automata/_vavilov2.pdf.zip). После этого я спросил Костю: «**И сейчас применяешь?**». Он ответил: «**Да. По-другому уже не получается мыслить...**».

5. Мне пришло весьма лестное письмо с очень странной просьбой: «Здравствуйте, Анатолий Абрамович! **Мне очень нравится Ваша идея автоматного программирования**. Я закончил Киевский Политех по специальности «Вычислительная техника», которая имела профиль, связанный с компьютерным оборудованием. После этого я учился программировать. Мне показалось, что программирование на языках, подобных *Алголу*, странно. Про Вас я узнал на ярмарке в Ганновере от профессора В. Хачумова из РАН. Я в высшей степени уверен, что **направление Ваших исследований одно из важнейших в информатике**. ***Я скачал Вашу книгу про автоматное программирование и прошу Вашего разрешения прочесть ее*** (!). Желаю Вам успеха. **Я знаю, что Ваша идея велика**. **Веслав Пошевецки**, wiesiek@acm.org». Ну, что тут скажешь...

6. «**Благодарю Вас за автоматное программирование**, которое я неоднократно применял в своих проектах, начиная с той Вашей лекции, на которой впервые узнал о нем и получил книжку. Это было примерно 10 лет назад. Теперь уже у меня появились свои ученики, и я хотел бы попросить у Вас посоветовать, как проще научить их автоматному подходу» (**Михаил Глухов**).

7. «Приведу слова Шалыто из интервью (https://habr.com/ru/company/dataart/): «Программировать можно в разных терминах. Я предлагаю программировать с использованием понятия «состояние». Люди живут в состояниях. Жив, здоров, болен. Если болен, то в определенном состоянии находишься. Можно описать состояния и у технического объекта. Определили состояние: открыт-закрыт, открывается-закрывается. После этого формируются дуги, которые обеспечивают переходы из одного состояния в другое. Потом записываются условия переходов. После этого в вершинах и/или на дугах записываются выполняемые действия. И вот так, я считаю, должно начинаться создание любых более-менее сложных поведенческих программ. Это открывает такие возможности, что вы даже не представляете. Так спроектированные программы можно верифицировать, на них может быть выпущена удобная для понимания заказчика проектная документация». **Господи, насколько же это прекрасно! Полностью согласен**» (***gecube***).

8. «**Концепция автоматного программирования разработана *Анатолием Шалыто***, в том числе в интеграции с объектно-ориентированным программированием. Автоматная программа

определяется как совокупность классических конечных автоматов. Используются графическое и текстовое представления программы. Управляющие состояния являются значениями переменной, которая фактически является частью состояния программы. При реализации автоматной программы применяется ***Switch*-технология**. Использование объектно-ориентированной технологии позволяет «уплотнить» состояния автоматной программы, сократить ее объем и локализовать часть связей программы внутри классов. ***Термин «автоматное программирование» и его аналог «Automata-Based Programming» применяется только в России*. Автор благодарен *А.А. Шалыто* за его работы по автоматному программированию»** (**Шелехов В.И.**, http://persons.iis.nsk.su/files/persons/pages/automatProg.pdf).

9. 10.11.2021 г. поздравил нашего выпускника 2006 г. **Веселина Пенева** с днем рождения, который ответил мне так: «30 октября я на своем *YouTube*-канале *BitDust Team* разместил ролик *State Machines: How to «Draw» Behavior of Your Program* (https://www.youtube.com/watch?v=MnwnPFxh1Lk&feature=emb_logo). **Очень благодарен Вам за то, что когда-то этому меня научили!**».

Вот уж воистину: «***Известное известно немногим».***

10. И в заключение. А вот, что написано обо мне в комментарии к тексту *DrCroco* (А. Столяров с ВМК МГУ), с которым у меня в свое время была склока в Википедии (https://ru-cs.livejournal.com/7377.html): «Да, я тоже в свое время поразился этой ситуации... Но знаете, в чем забавная деталь? **Словосочетание «автоматное программирование» никто не догадался в свое время «закопирайтить». А г-н Шалыто догадался! И что теперь? Попробуйте найти этот термин в другом месте, без ссылок на Шалыто**» (***rg_software***). И это написал не мой знакомый или ученик...

10.01.2022. https://vk.com/@1077823-o-prioritete

Валидация автоматных спецификаций

В работе изложен новый взгляд на обеспечение качества автоматных программ. При этом вместо термина «верификация автоматных программ» предлагается использовать термины «верификация автоматных моделей» и «валидация автоматных спецификаций», первый которых термин применим при наличии формальной спецификации, а второй – при ее отсутствии, что более характерно для практики. Это позволяет более осмысленно подходить к пониманию того, как обеспечивать качество автоматных программ.

Под верификацией [1] обычно понимают проверку неформально построенной прикладной программы на предмет выполнения формальной спецификации**. Этот процесс надо проводить заново для каждой вновь созданной прикладной программы.**

Считается, что **верификация** позволяет установить, что «**мы создали продукт таким, каким и намеревались его сделать**», а **валидация** подтверждает, что «**мы создали правильный продукт**». **Поэтому ошибка в общем случае – не только то, что устанавливается формально (верификацией), но и не только формально (валидацией)**. При этом **тестирование** может рассматриваться, как разновидность валидации, которая проводится с целью определения соответствия поведения системы её ожидаемому поведению на конечном наборе тестов.

Если формальная спецификация становится исполняемой, то вместо верификации каждой из генерируемых прикладных программ верификация может проводиться только один раз – однократно верифицируется программа-генератор прикладных программ. Если квалификации создателей генератора для его верификации не хватает, то он может быть **валидирован** – **проверен**, например статистически, на **правильность** построения с его помощью прикладных программ.

Поэтому если спецификация является исполняемой, а программа-генератор верифицирована или валидирована, то *проблема верификации прикладных программ исчезает!*

Возможны два варианта (случая) создания программ: при наличии формальной спецификации в начале проектирования, и при ее отсутствии на этой стадии создания системы.

В первом случае при предлагаемом подходе можно проводить верификацию не самой программы, а неформально построенной модели, по которой она генерируется. **При использовании автоматного программирования такая модель – система графов переходов конечных автоматов**. Эта модель для программы является исполняемой спецификацией, которая должна быть

проверифицирована по исходной спецификации, так как предполагается, что она существует. В этом случае можно говорить о «**верификации автоматной модели**».

Однако формальная спецификация в начале проектирования существует крайне редко: либо для очень простых объектов, либо для невероятно ответственных объектов.

В работе [2] таким простым объектом являются часы с будильником, поведение системы управления которым **описывается всего лишь одним графом переходов автомата с тремя состояниями**. Показано, что даже такую модель трудно верифицировать: она успешно выполнилась для исходно построенной формальной спецификации в виде определенного числа темпоральных правил, но, когда в спецификацию были внесены изменения – увеличено число темпоральных правил, в графе переходов удалось обнаружить ошибку. Обратим внимание на то, что в данном случае **формальная спецификация строится неформально** – **ее не по чему верифицировать, а изменения, вносимые в неё, относятся к валидации**.

Для очень ответственных объектов *при наличии уникальных специалистов*, как это свое время имело место при автоматизации Лондонского метро, верификация проводится с дальнейшей валидацией в течение всего жизненного цикла.

Во втором случае при проектировании систем управления сложными технологическими объектами формальная спецификацию в начале проектирования отсутствует. **От Заказчика в лучшем случае можно получить** только разрозненные сведения и знания о том, как система должна работать, причем обычно в основном режиме. Все тонкости работы по разным причинам на ранних этапах создания системы Заказчик описать не может.

Какие-то знания (возможно, и основные) добавляет Разработчик системы управления, если он опытный специалист. По информации от Заказчика и Разработчика можно неформально писать программу (остаться без формальной спецификации) или неформально строить формальную спецификацию (зафиксировать указанную выше информацию на каком-либо формальном языке). Первый путь – традиционный [3], но, по-моему, мнению тупиковый, поэтому предлагаю использовать второй путь с генерацией программы по неформально построенной формальной спецификации.

При этом возникает вопрос, какой математический аппарат использовать при ее построении. **Мною в качестве формальной спецификации предлагается использовать систему графов переходов конечных автоматов. По этой спецификации программа должна строиться (генерироваться) формально и изоморфно. При этом она не только будет соответствовать спецификации, но и будет «внешне похожа» на неё.**

Правильность такой спецификации требует проверки – проведения валидации. **«Правильность» неформальное понятие,** оно должно **«устраивать»: оборудование** (при существующей на данный момент спецификации не должно происходить его аварий и поломок) и **всех специалистов**, участвующих в создании системы, **которые делают все возможное, чтобы и в дальнейшем оборудованию было «комфортно»**.

Любое испытание можно рассматривать как валидацию, проводимую с целью подтверждения правильности спецификации и уточнения ее при необходимости. Уточнение обеспечивается путем внесения соответствующих изменений в существующую к этому моменту систему графов переходов. Однако, так как в ходе корректировки могут быть внесены ошибки, то после этого необходимо снова провести испытания и т. д.

Эксплуатация системы (тем более опытная) также может рассматриваться как уточнение спецификации. Только в момент списания системы, если она существовала в единственном экземпляре, можно считать, что, наконец-то, получена спецификация, которая является окончательной, и то при условии, что на ее основе не будет создаваться модификация системы. Таким образом, в каждый момент времени **существует формальная спецификация, которая на любом этапе валидации может потребовать корректировки**.

Из изложенного следует, что уточнение формальной спецификации системы управления проводится путем валидации в течение всего её жизненного цикла, а при создании модификации, и после его завершения. К средствам валидации, в частности, относится тестирование, а также проверка выполнения темпоральных свойств, используемых в традиционной верификации [3].

Обращаю внимание, что в предлагаемом подходе **система графов переходов является не только исполняемой спецификацией, но и языком программирования**. ***При этом после валидации спецификации, верификация или даже валидация программы, построенной по ней, не требуется!*** **Предложенный подход использован в работе [4], который в динамике проиллюстрирован в видеоприложении [5].**

Таким образом можно сделать вывод, что, в свое время, назвав книгу «Верификация автоматных программ» [6], авторы поступили недостаточно корректно, так как уже при ее написании предполагали, что автоматные программы строятся не эвристически, а формально по спецификации. Поэтому она должна была называться **либо «Верификация автоматных моделей», либо «Валидация автоматных спецификаций», причем второе название является значительно более практичным**.

Исходя из изложенного, выражение «Верификация автоматных программ» [7] можно использовать как жаргон для понятия «Валидация автоматных спецификаций», так же как широко известное выражение «Минимизация булевых функций», которое повсеместно применяется вместо понятия «Минимизация булевых формул» [8]. Последнее связано с тем, что булеву функцию (таблицу истинности), если она существенно зависит от всех своих переменных, нельзя проминимизировать – в отличие от булевой формулы, для которой в большинстве случаев это возможно.

Литература

1. ***Кулямин В.В.*** Верификация программного обеспечения / Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы». 2008, 111 с. https://www.ispras.ru/publications/methods_of_software_verification.pdf.

2. ***Ульянцев В.И., Шалыто А.А.*** О верификации простых программ со сложным поведением (http://is.ifmo.ru/works/2013/ulyantsev-shalyto-verification.pdf).

3. ***Карпов Ю. Г.*** *Model Checking*. Верификация параллельных и распределенных программных систем. БХВ-Петербург. 2010, 560 с.

4. ***Янкин Ю.Ю., Шалыто А.А.*** Автоматное программирование ПЛИС в задачах управления электроприводом // Информационно-управляющие системы. 2011. № 1, с. 50-56. http://www.i-us.ru/index.php/ius/article/view/13825.

5. ***Янкин Ю.Ю., Шалыто А.А.*** Метод создания программного обеспечения модулей, выполненных на основе программируемых логических интегральных схем. Видеоприложение (https://www.youtube.com/watch?v=YNWdmnwHZi8).

6. ***Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.*** Верификация автоматных программ. СПб.: Наука. 2011, 244 с. (http://is.ifmo.ru/verification/velder_verification_posobie_nauka.pdf).

7. ***Кузьмин Е.В. Соколов В.А.*** Моделирование, спецификация и верификация «автоматных» программ // Программирование. № 1, с. 38-60 (http://is.ifmo.ru/download/2008-03-12_verification.pdf)

8. ***Шалыто А.А.*** Логическое управление. Методы аппаратной и программной реализации алгоритмов. СПб.: Наука. 2000, 780 с. (http://is.ifmo.ru/books/log_upr/1).

26.10.2022. https://ntv.ifmo.ru/file/article/21921.pdf

Почему в эпоху нейронных сетей для управления ответственными технологическими объектам необходимо применять автоматное программирование

Мир восхищен применением нейронных сетей во всех областях человеческой деятельности. Первой звоночек о том, что к их применению надо подходить очень ответственно прозвучал в связи с тем, что они могут порождать фейки, мало отличимые от реальных положений и событий.

Теперь и я решил «позвонить». Нейронные сети применяют, в том числе и в управлении техническими объектами, например комбайнами. Однако, по моему мнению, всё что создано **рукотворно** человеком, **должно быть понятно Разработчику, Заказчику и Пользователю** автоматизированного объекта управления. Каждый из них должен понимать, **как работает система управления. Это должно позволить сравнительно просто реализовать старинное свойство, предъявляемое к системам управления – обеспечить их «ремонтопригодность», понимаемую как приспособленность к внесению изменений в определенные документацией сроки.**

При применении же нейронной сети при неправильной ее работе остаются неясными два вопроса: правильно ли выбрана ее структура и достаточно ли она обучена. Вряд ли на эти вопросы можно

получить положительные ответы за короткие времена, отводимые на испытания сначала системы управления с имитаторами объекта управления (например, дизель-генератора) на заводе-изготовителе системы, затем при совместных испытаниях системы управления с объектом управления на заводе-изготовителе указанного объекта, потом при сдаче «системы управления совместно с объектом управления» на заводе-строителе ответственного технологического изделия (например, судна) и, наконец, в ходе испытаний изделия в «полевых» условиях.

Если же в процессе испытаний или в ходе эксплуатации изделия авария будет связана с человеческими жертвами, то **родственников погибших, конечно же, не удовлетворит естественный в этой ситуации ответ: «Недоучили!».**

Этого существенного недостатка для ответственных технологических объектов лишено автоматное программирование, которое я и мои ученики развивают и успешно внедряют в промышленности уже много лет. Кратко изложу его основы.

1. Основным понятием автоматного программирования является понятие «состояние». Мне кажется, что в программировании состояния также важны, как и в жизни человека: если у человека кружится голова, то состояние «кружится голова» отвратительно, и он не хочет находиться в нём долго, и старается побыстрее перейти в другое состояние, в котором понимает, что все «встало на свои места». Думаю, что программа, в которой с самого начала не выделены состояния, «испытывает» нечто похожее на головокружение человека, что не нормально.

Состояния могут быть связаны друг с другом дугами, по которым осуществляются переходы между состояниями, образуя конечный автомат. В теории алгоритмов под таким автоматом понимается абстрактная модель дискретного устройства, имеющего один вход и один выход. Такой автомат обычно называется абстрактным. На вход такого автомата подаются символы соответствующего алфавита, инициирующие переходы между состояниями. Автоматы этого типа часто используется в компиляторах. Абстрактный автомат на выходе может формировать символы из определённого алфавита.

Принципиальная особенность автоматного программирования состоит в том, что мною было предложено применять в нём другую модель автомата – структурный автомат, который имеет несколько входов, на которые поступают входные переменные, и несколько выходов, на которых формируются значения выходных переменных. Такие автоматы в программировании соответствуют схемам в аппаратуре. В дальнейшем под автоматами понимаются структурные автоматы.

2. Целесообразность применения автоматов состоит в том, что их состояния декомпозируют все множество входных переменных на группы, выделяя с помощью каждого состояния только то подмножество входных переменных, которое определяет переходы из рассматриваемого состояния в соседние (смежные) состояния, в том числе и в самого себя. При этом входные переменные, не входящие в группу, определенную некоторым состоянием, не влияют на переходы из этого состояния в другие состояния – переходы из рассматриваемого состояния несущественно зависят (не зависят) от этих переменных. Это способствует реализации с помощью графов переходов задач большой размерности. Такие задачи эффективно решаются также за счет того, что автоматы могут быть вложенными и вызываемыми.

3. Находясь в некотором состоянии, автомат с памятью превращается в соответствующий автомат без памяти (комбинационный автомат), который по значениям набора входных переменных, «выбранного» этим состоянием, осуществляет выбор одного из смежных состояний, в состав которых входит и рассматриваемое. Новое состояние «настраивает» автомат на реализацию в общем случае другого комбинационного автомата. Таким образом, автомат с памятью можно рассматривать в качестве многофункционального модуля, настраиваемого состояниями на реализацию в определенной последовательности различных ортогональных систем булевых формул, зависящих от различных наборов входных переменных.

4. Еще о состояниях. **А. Дж. Перлис** в 1966 г. [1] предложил в описания языка, среды и правил вычислений включать состояния, которые могут подвергаться мониторингу во время исполнения, позволяя диагностировать программы, не нарушая их целостности. В этом же году **Э. Дейкстра** [2] **предложил ввести так называемые переменные состояния**, **с помощью которых можно описывать состояния системы в любой момент времени**. Он (как и я в автоматном программировании) использовал для этих целей целочисленные (многозначные) переменные. При этом им были поставлены вопросы о том, какие состояния должны вводиться, как много значений

должны иметь переменные состояния, и что эти значения должны означать. Он предложил сначала определять набор подходящих состояний (и я в автоматном программировании предлагаю тоже), а лишь затем строить программу. Он также предложил сопоставлять процессы с переменными состояний и связывать процессы через эти переменные, что я делаю в автоматном программировании.

5. По мнению Дейкстры, диаграммы переходов (я их называю графы переходов) между состояниями могут оказаться мощным средством для проверки программ. Это обеспечивает поддержку его идеи о том, что «программы должны быть с самого начала составлены правильно, а не отлаживаться до тех пор, пока они не станут правильными». Не появление ли автоматного программирования он предвещал [3-16]?

6. А еще мне близки слова **Б. Лисков** (https://ru.wikipedia.org/wiki/Лисков,_Барбара): «Моделируйте свои классы на основе поведения, а не свойств. Моделируйте свои данные на основе свойств, а не поведения». При этом всегда надо помнить: «то, что не специфицировано формально, не может быть проверено, а то, что не может быть проверено, не может быть безошибочным».

7. Наличие явно выделенных состояний в автоматных программах позволяет естественным образом (без дополнительных затрат) формировать протоколы, как для отладки, так и контроля их работы. Вот, что по этому поводу пишет мой аспирант **А.В. Калачинский** [16], неоднократно применявший автоматное программирование на практике: «Одной из замечательных возможностей, заложенной в автоматное программирование, является автоматическое встраивание в генерируемый код функций документирования работы автоматных программ, что позволяет полностью контролировать процесс их выполнения. Это обеспечивает возможность значительно сократить время автономной и комплексной отладки систем». И еще от него: «Для систем управления объектами повышенной важности Заказчиком часто выдвигаются дополнительные требования к системе по ведению штатного документирования поведения объекта и работы системы. Таким образом, дополнительный, часто только отладочный функционал документирования работы автоматной программы, превращается в необходимую и обязательную часть функционирования системы – штатную функцию регистрации наблюдения за объектом управления и работой системы. Эта особенность автоматного программирования позволяет без дополнительных затрат превратить технологический (отладочный) режим в одну из основных функций системы, которая обеспечивает реализацию очень важного требования Заказчика».

8. При необходимости использовать схемы алгоритмов (этот термин заменил термин «граф-схемы алгоритмов») **предлагаю начинать их построение с дешифратора состояний, а не дешифратора входных воздействий, как это делается обычно**. Построенные таким образом схемы изоморфны конструкции *Switch*, входящей во многие языки программирования, а схемы алгоритмов, построенные иначе – не изоморфны этой конструкции. **Если не знать в каком состоянии находится система управления, то какой смысл опрашивать входные переменные, ведь не зная состояния, по значению входной переменной остается неясным куда системе переходить?** Однако большинство инженеров обращать на это внимание, почему-то, не хотят – видимо, потому что их так не учили программировать. Схемы с указанной выше структурой названы мной – «автоматными схемами алгоритмов».

9. В набор функциональных блоков, используемых для программирования некоторых типов контроллеров, входит блок «цифровой мультиплексор», который является аналогом конструкции *Switch* в языках программирования. Тогда граф переходов может быть изоморфно реализован на основе такого мультиплексора. Это позволило нашему немолодому сотруднику, ***который не умел программировать***, разработать и успешно сдать систему на судне, а также закрыть построечное удостоверение раньше проектировщиков других систем комплекса, которые были не только хорошими инженерами, но и умели программировать, однако делали это иначе... Этот сотрудник с моей помощью создал графы переходов и научился, как от них изоморфно переходить к текстам программ – к функциональным схемам, ядром которых были цифровые мультиплексоры. Незначительные изменения в программе, созданной таким образом, необходимость внесения которых возникла на судне, этот специалист в так спроектированной программе смог успешно произвести.

10. Формализация на основе использования графов переходов автоматов при выдаче технического задания позволила ***разделить работу, а главное, ответственность*** **между организациями Заказчика и Разработчика**. Это также обеспечило возможность проводить корректировку алгоритмов и программ не в терминах судовых устройств, как это делалось до сих пор, а в терминах автоматов, что для программистов значительно проще и понятнее.

11. В ответственных системах для того, что избежать аварий, подобных произошедшей при прилунении израильского космического аппарата «Беришит», необходимо применять горячее резервирование. Автоматы, особенно с многозначным кодированием состояний (в системе такими автоматами число переменных, кодирующих состояния, не зависит от числа состояний в автоматах и равно числу автоматов!), идеально подходят для этой цели, позволяя эффективно строить дублированные системы с синхронизацией состояний резервного и основного каналов, так как в этом случае из основного канала в резервный при необходимости надо передать значения лишь небольшого числа переменных, число которых равно числу автоматов в системе. Вот, что по этому поводу пишет **А.В. Калачинский**: «Другой замечательной особенностью автоматного программирования стала простота реализации «горячего» резервирования двухканальных систем управления. Для восстановления управления на резервном канале часто бывает достаточным передать в него данные только о состоянии автоматов основного канала управления. При этом система при переключении канала восстановит свою работу гарантированно в том состоянии, в котором она была в основном канале до переключения».

12. Графы переходов автоматов (в том числе вложенных), например, с помощью пакета *Stateflow* [11] изображаются на экране и отлаживаются в различных режимах (пошаговом и автоматическом). В пакете имеется возможность по графам переходов осуществить генерацию программы, в частности, на одном из ассемблеров ПЛИС, которая и загружается в схему. По этой технологии моим аспирантом **Ю.Ю. Янкиным** [11] реализовано программное обеспечение для модулей большого числа систем управления ответственными промышленными объектами. С динамикой работы указанного пакета можно ознакомиться здесь: https://www.youtube.com/watch?v=YNWdmnwHZi8.

13. Для поддержки автоматного программирования, в том числе и с участием автора, разработаны различные инструментальные средства, одним из которых является *UniMod* [8]. Сайт, посвященный этому средству, расположен по адресу https://unimod.sourceforge.io. Работа *UniMod* проиллюстрирована здесь: https://www.youtube.com/watch?v=Y4et51dz-HE.

14. Мой ученик **В.О. Клебан** [14] мне как-то рассказывал, что он сдавал на объекте управляющую систему, некоторые подсистемы которой были реализованы автоматно, а другие – традиционным путем. При этом автоматные подсистемы либо сразу правильно работали, либо не работали, и тогда их работоспособность обеспечивалась достаточно просто. Уверенность в правильности работы остальных подсистем отсутствовала даже после их сдачи.

15. Я предложил применять автоматные программы для управления ответственными технологическими объектами, так как обычно сравниваю традиционные программы с флагами со «слонами на тонких ножках», изображенными **С. Дали** на картине «Искушение Святого Антония». При этом я всегда отмечаю, что такие слоны уникальны – встречаются только на этой картине, а программы с флагами применяются повсеместно, обладая устойчивостью :-) этих слонов! Однако, по мнению **В. Аллена**, тонкие ножки присущи не только программам и слонам: «Профессор спасался от здоровенного мохнатого неправильного глагола *tener* (иметь), гонявшегося за ним на длинных тонких ножках». В общем, «в тонких ножках» в отличие от состояний, нет ничего хорошего…

16. Без исходных текстов плохо, но и с ними тоже бывает нехорошо. **Чего же не хватает Заказчику, Разработчику и Пользователю «для полного счастья»?** Ответ прост: проектной документации, выполненной весьма подробно и аккуратно, в которую программная документация входит только как одна из составляющих. Мосты, дороги и небоскребы без проектной документации обычно не строятся, а вот про программы несмотря на их в общем случае сложность этого не скажешь. В программировании сложилась ситуация, определяемая так: «Если бы строители строили дома так, как программисты пишут программы, достаточно было бы одного единственного дятла, чтобы разрушить цивилизацию». Время идет, а в этом вопросе несмотря на все мои старания, к сожалению, практически ничего не меняется...

17. Особенность автоматного программирования состоит в том, что при его применении графы переходов используются при спецификации, проектировании, реализации, отладке, документировании проекта и процесса работы. Они могут применяться также и как язык программирования.

18. Автоматные спецификации достаточно просто валидируются [15], а при необходимости и верифицируются [12, 13], так как **программы этого класса формально и изоморфно строятся по модели, в то время как для иных классов программ верификация выполняется в обратном порядке – по программе строится модель, что, по моему мнению, противоестественно**. **Применение автоматов *приближает к доказательному проектированию программ*.**

19. При алгоритмизации я выделяю **управляющие и вычислительные состояния**. Небольшое число управляющих состояний может, как например в головоломке «Ханойские башни», поддерживать процесс с экспоненциальным числом вычислительных состояний.
20. Парадигма автоматного программирования состоит в представлении сущностей со сложным поведением в виде автоматизированных объектов управления.
21. Нахождение компромисса между сложностью автомата и сложностью операций объекта управления, **примирение тьюрингова программирования с традиционным – это и есть «миссия» автоматного подхода в мире разработки программного обеспечения**.
22. Возможно, что применение автоматного программирования является «серебряной» пулей, о которой в 1975 г. **Ф. Брукс**, говорил, что при создании ПО ее не существует, а через 25 лет – в 2010 г. [17] с учетом работ **Д. Харела** [18], основанных на одной из разновидностей автоматного подхода, в ее отсутствии он уже был не так уверен.

Видимо, не случайно в международном стандарте ***IEC 61499*** [19], предназначенном для построения современных распределенных систем управления и автоматизации, в блоках, описывающих поведение программ, применяются не нейронные сети, а графы переходов конечных автоматов. Перспективные системы управления в нефтедобыче [20] и «интеллектуальной» энергетике [21] в настоящее время разрабатываются с использованием этого стандарта.

Обзорные статьи по автоматному программированию размещены здесь: https://vk.com/@1077823-vtomatnoe-programmirovanie, https://vk.com/@1077823-esche-ob-avtomatnom-programmirovanii.

Литература

1. ***Перлис А.*** Синтез алгоритмических систем / Лекции лауреатов премии Тьюринга за первые двадцать лет. 1966-1985. М.: Мир, 1993, с. 16-29.
2. ***Дейкстра Э.*** Взаимодействие последовательных процессов / Языки программирования. М.: Мир, 1972, с. 9-86.
3. ***Шалыто А.А.*** *Switch*-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука. 1998, 628 с.
4. ***Шалыто А.А.*** Автоматное проектирование программ. Алгоритмизация и программирование задач логического управления // Известия РАН. Теория и системы управления. 2000. № 6, с. 63-81.
5. ***Шалыто А.А.*** Алгоритмизация и программирование для систем логического управления и «реактивных» систем //Автоматика и телемеханика. 2001. № 1, с. 3-39.
6. ***Шалыто А.А., Туккель Н.И.*** *Switch*-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5, с. 45-62.
7. ***Канжелев С.Ю., Шалыто А.А.*** Автоматическая генерация автоматного кода // Информационно-управляющие системы. 2006. № 6, с. 35-42.
8. ***Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А.*** Инструментальное средство для поддержки автоматного программирования // Программирование. 2007. № 6, с. 65-80.
9. ***Шалыто А.А.*** Парадигма автоматного программирования // Научно-технический вестник информационных технологий, механики и оптики. 2008. № 8 (53). Автоматное программирование, с. 3-24.
10. ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. СПб.: «Питер». 2008, 168 с.
11. ***Янкин Ю.Ю., Шалыто А.А.*** Автоматное программирование ПЛИС в задачах управления электроприводом // Информационно-управляющие системы. 2011. № 1, с. 50-56.
12. ***Вельдер С.Э., Лукин М.А., Шалыто А.А., Яминов Б.Р.*** Верификация автоматных программ. СПб.: Наука. 2011, 244 с.
13. ***Кузьмин Е.В., Соколов В.А.*** Моделирование, спецификация и верификация «автоматных» программ // Программирование. 2008. № 1, с. 38-60.
14. ***Клебан В.О., Шалыто А.А.*** Разработка системы управления малоразмерным вертолетом // Научно-технический вестник информационных технологий, механики и оптики. 2011. № 2, с. 12-15.
15. ***Шалыто А.А.*** Валидация автоматных спецификаций // Научно-технический вестник информационных технологий, механики и оптики. 2023. № 2, с. 436-438.
16. ***Калачинский А.В.*** Технология проектирования программного обеспечения систем дискретного управления на основе автоматного подхода // Системы управления и обработки информации. 2023. № 3, с. 30-47.

17. ***Брукс Ф.*** Мифический человеко-месяц, или как создаются программные системы. СПб.: Символ, 2010. 298 с.
18. ***Harel D.*** Statechart: A Visual Formalism for Complex Systems // Science of Computer Programming. 1987. № 8, pp. 231-274.
19. ***Дубинин В.Н., Вяткин В.В.*** Модели функциональных блоков *IEC* 61499, их проверка и транформации в проектировании распределенных систем управления. Пенза. Изд-во ПГУ. 2012, 348 с.
20. ***Bartusiak R., Bitar S., DeBari D., Houk B., Stevens D., Fitzpatrick B., Sloan P.*** Open Process Automation: A Standards-Based, Open, Secure, Interoperable Process Control Architecture // Control Engineering Practice. 121 (2022). 105034.
21. ***Wang Y., Guo N., Yan G., Liu J.*** Design of Integrated Energy System Based on IEC 61499 and OPC UA / Proceedings of the 41st Chinese Control Conference, 2022, China, pp. 4270-4275.
20.08.2023. https://vk.com/@1077823-pochemu-v-epohu-neironnyh-setei-dlya-upravleniya-otvetstvenn, https://computer-museum.ru/articles/yazyki-i-sistemy-programmirovaniya/6442/
P.S. «Замечательный текст! Миссия (автоматного подхода в мире разработки программного обеспечения), как ее я вижу, совпадает с Вашим энергичном отстаиванием принципов точности и четкости конечных автоматов в мире обычного ПО, которое так легко проглатывает и прощает программисту до поры до времени его ошибки» (И. Дейнека, ИТМО).

Печальная история о применении конечных автоматов в программировании

Соискатель бакалаврской степени написал выпускную квалификационную работу по построению программ с использованием **стандарта *IEC 61499*** – открытого стандарта для построения распределенных систем управления и автоматизации https://vt.pnzgu.ru/files/vt.pnzgu.ru/sotrudniki/dubinin/dlya_rezyume/monografiya/fb_monography.pdf). В стандарте для описания поведения программ используются блоки, по сути **являющиеся графами переходов конечных автоматов**.

На защите молодой человек высказал следующую «мысль»: он в работе построил сверлильный станок. Это вызвало у меня большое сомнение, а вслух я предположил, что он на самом деле написал программу, управляющую станком. Молодой человек подтвердил мое предположение. После этого я спросил, построил ли он хотя бы один граф переходов, и получил утвердительный ответ.

Потом я поинтересовался у соискателя на каких языках он пишет программы на работе. Оказалось, что на *Kotlin* и *Java*. После этого я задал ему еще один вопрос: применял ли он графы переходов при написании (очень хотелось написать словосочетание «при проектировании», но я решил не пугать молодого человека) программ на указанных языках, за исключением каких-либо особых случаев, как, например, при поиске подстроки с помощью алгоритма **Ахо-Корасик** (https://neerc.ifmo.ru/wiki/index.php?title=Алгоритм_Ахо-Корасик), который рассказывается в рамках курса «Алгоритмы и структуры данных» во всех университетах мира, где этот курс читается. Молодой человек уверенно ответил: «Нет».

Потом я спросил, а слышал ли он что-либо *про автоматное программирование?* И снова молодой человек произнес уверенное «Нет». ***Сказать, что в этот момент мне стало немного обидно – это, значит, ничего не сказать.*** Это связано с тем, что молодой человек заканчивает бакалавриат кафедры, на которой я уже 25 лет развиваю указанный подход к программированию (https://is.ifmo.ru/). И это при том, что за указанное время по этой тематике было выполнено «немереное» число курсовых (https://is.ifmo.ru/projects/) и выпускных (https://is.ifmo.ru/diploma-theses/) работ, а также было защищено почти полтора десятка кандидатских диссертаций (https://is.ifmo.ru/disser/).

Конечно, я уже несколько лет не читаю этот курс, но отсюда не следует, что никто из других преподавателей кафедры и руководитель соискателя не могли хотя бы **упомянуть про указанный подход, который относят к одной из парадигм программирования** (https://znanierussia.ru/articles/Парадигмы_программирования), тем более, что я помню, что один из их основных преподавателей нынешних студентов был моим аспирантом и защищал диссертацию по этой тематике, не говоря уже о том, что некоторые из остальных преподавателей делали под моим руководством курсовики по автоматному программированию. Да и сам молодой человек, если бы не был лохом, мог бы в Интернете что-нибудь посмотреть (https://www.youtube.com/watch?v=PPWTxceMutk&t=149s) и почитать (https://is.ifmo.ru/books/_book.pdf)

по этому вопросу. Однако, **есть нюанс**: для этого надо знать какой сделать запрос, а он об указанном термине даже не слышал…

Я утерся и задал молодому человеку следующий вопрос: «Не показалось ли Вам странным, что в стандарте, описывающем будущее промышленной автоматики, автоматы являются единственным средством описания поведения программ, а Вы в своей практической деятельности никогда не применяли автоматы? И не надо говорить, что управлением в промышленности Вы не занимаетесь…». Молодой человек честно ответил, что он об этом даже не думал – самому в голову не пришло, а из преподавателей никто его не «подтолкнул» в этом направлении.

Для того, что приобщиться к автоматному программированию, я предложил защищающемуся в магистерской диссертации переписать на автоматах одну или несколько программ, написанных им на работе с использованием флагов, и сравнить полученные программы.

Помню, что еще в августе 2002 г. появилась статья об использовании предложенного мною автоматного подхода к программированию в весьма странном месте: ***Вавилов К.*** Программирование за… 1 (одну) минуту... // Компьютер Price. 2002. № 31 (http://is.ifmo.ru/automata/1minute/). Константин однажды услышал мою лекцию, и понеслось…

После того, как я опубликовал тексты «Автоматное программирование» (https://vk.com/@1077823-vtomatnoe-programmirovanie) и «**Еще об автоматном программировании**» (https://vk.com/id1077823?z=article_edit1077823_62364), Костя написал мне: «**Лично для меня это Великая история** и несомненное применение в различных разработках алгоритмов и ПО. **На любую критику есть конкретные примеры – реально работающие на объектах Петербурга и России, эффективные, задокументированные, понятные программы** (http://is.ifmo.ru/automata/_metod065.pdf; http://is.ifmo.ru/automata/_s7300.pdf; http://is.ifmo.ru/automata/_vavilov2.pdf.zip). После этого я спросил Костю: «**И сейчас применяешь?**». Он ответил: «**Да. По-другому уже не получается мыслить...**».

У меня есть и другие подобные примеры, и хочется, чтобы и указанный выше молодой человек понял, что человечество не зря собирается строить свое будущее в промышленной автоматике на автоматах (https://vk.com/@1077823-pochemu-v-epohu-neironnyh-setei-dlya-upravleniya-otvetstvenn).

С этой надеждой я и заканчиваю писать этот текст, но меня не покидает ощущение, что нечто подобное мне придется говорить и писать неоднократно…

21.06.2023. https://d-russia.ru/pechalnaja-istorija-o-primenenii-konechnyh-avtomatov-v-programmirovanii.html

P.S. 1. Этот текст был прокомментирован так: «Согласен с Костей: «При проектировании ПО, если ты хоть раз попробовал автоматный метод проектирования, то по-другому уже и не хочется проектировать :)». Спасибо Вам, Анатолий Абрамович, опираясь на Ваши труды, **я смог внедрить данный подход у нас на предприятии и проектировать качественное и надежное ПО**. Этот подход и в настоящее время мы активно применяем и продвигаем» (С. Августанович, закончил кафедру «Системы управления» факультета «Компьютерные технологии, управление и радиоэлектроника» Южно-Уральского государственного университета, https://vk.com/avgustanovich_s).

2. На сайте онлайн-школы «Хекслет» среди курсов по обучению программированию на *JavaScript* я **обнаружил курс «Автоматное программирование», который после 23.08.2020 г. еще раз был обновлен 23.09.2023 г.** (https://ru.hexlet.io/courses/js-abp). В обращении к слушателям сказано, что в результате изучения курса «Вы научитесь: видеть автоматы в происходящих вокруг процессах; применять автоматное программирование для решения типовых задач, использовать шаблоны проектирования *State* и ***State Machine***.

3. Особенно меня порадовало и удивило изучение второго паттерна, так как его разработали мы: ***Шамгунов Н.Н., Корнеев Г.А., Шалыто А.А***. *State Machine*. Новый паттерн объектно-ориентированного проектирования // Информационно-управляющие системы. 2004. № 5, с. 13-25 (http://is.ifmo.ru/works/pattern/); ***Шамгунов Н.Н., Корнеев Г.А., Шалыто А.А.*** Паттерн *State Machine*. Внедрение. Сравнение с другими подходами (http://is.ifmo.ru/works/patterninc/); ***Шамгунов Н.Н., Корнеев Г.А., Шалыто А.А.*** *State Machine* – расширение языка *Java* для эффективной реализации автоматов // Информационно-управляющие системы. 2005. № 1, с. 16-24 (http://is.ifmo.ru/works/sm_language); ***Shalyto A., Shamgunov N., Korneev G.*** 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, pp. 51-58. http://is.ifmo.ru/articles_en/2006/shalyto-shamgunov-korneev-2006.pdf.
4. После этого я написал «достаточно открытое» :-) письмо Георгию Александровичу Корнееву: «Дорогой Гоша! Люди за изучение паттерна *State Machine*, предложенного нами совместно с Шамгуновым платят деньги, а у нас автоматное программирование в курсах лекций по программирование даже не упоминают. Вот так... Настоящий тик-ток получается!».

Гоша ответил: «Уважаемый, Анатолий Абрамович! Если, по сути, то в этом (и даже большем) объеме рассказывается в курсе Андрея. А если рассказывать основательно и обо всем, то, как Вы знаете, это отдельный курс получается. Так и живем».

Естественно, что он без моего ответа не остался: «Дорогой Гоша, не держи меня за ... В курсе Андрея Станкевича рассказывается об автоматах, как в любых университетских курсах по этому вопросу, а совсем не об автоматном программировании, как его понимаю я и ты с Матвеем Казаковым на примере визуализаторов».
5. Второго ноября 2023 г. сооснователь этой школы **Кирилл Мокевнин** опубликовал ролик на тему «**Конечные автоматы как способ значительно упростить логику и понимание кода**» (https://www.youtube.com/watch?v=knoVv2ncwVI). В нем на засечке 12.35 на слайде весьма мелко изображена обложка моей с Надей Поликарповой книги «Автоматное программирование» (https://is.ifmo.ru/books/_book.pdf), но ни слова ни о ней, ни о нас не говорится. После этого я заинтересовался тем, что еще Мокевнин рассказывает свои слушателям про автоматы.
6. В сети обнаружил еще пять роликов, в которых он мог обсуждать автоматное программирование. При этом **четыре из них посвящены ментальному программированию**, названному так Моковниным: первый (https://www.youtube.com/watch?v=eEEHWQNuCLQ) был записан в **2013 г**., второй (https://www.youtube.com/watch?v=JnURhIf194s&t=1363s) – в **2014 г**., третий (https://www.youtube.com/watch?v=vkUTX1hruF8) – в **2018 г.**, а четвертый (https://www.youtube.com/watch?v=JnURhIf194s&t=1363s) – в **2021 г.** Пятый ролик, опубликованный в **2015 г.,** называется так: «Почему трудно программировать *UI*», и первый слайд в нем говорит сам за себя: «***UI*. State Machine**». Никаких откровений про автоматы по сравнению с изложенным ниже этот ролик не содержит.
7. В первом из четырех роликов Кирилл пояснил, что такое ментальные модели (https://rightrack.ru/109mental_models?ysclid=lok210unjo117131164). Он поведал, что **в психологии ментальной моделью** называют трудно формализуемую совокупность эмпирических знаний, которые формируется в сознании человека при взаимодействии с объектом. Проще говоря, это о том, **как мы представляем себе некий объект**.

При этом под **ментальным программированием** он понимает такое написание программ, при котором для каждой из них **по ее тексту можно понять не то, как она работает, а то, что программа делает – как решает задачу, для которой создавалась**. Ментально построенная программа, с одной стороны, работает и читается так, как о ней говорил Заказчик, а с другой – все программисты, участвующие в ее разработке и использовании, смогут единообразно понять, что реализует программа.

После этого Кирилл привел основные положения философии языка *Python*, которым должны соответствовать и ментальные программы, написанные не только на этом языке: **1. Явное лучше неявного**; **2.** Красивое лучше уродливого; **3.** Простое лучше сложного; **4.** Сложное лучше запутанного; **5.** Плоское лучше вложенного; **6.** Сложно объяснимая реализация – плоха; **7. Особые случаи не настолько особые, чтобы нарушать правила**.

Казалось бы, уже первое положение для обеспечения ментальности толкает к применению автоматного программирования, которое я назвал также «программирование с явным выделением состояний». **Однако время использования автоматов для построения ментальных программ в сознании, о котором Кирилл говорит выше, тогда еще у него не наступило – на этой лекции про автоматное программирование не было сказано ни слова**.
8. И это несмотря то, что книга ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование. СПб.: 176 с. к тому времени уже была опубликована трижды – в 2009 г. (с издательскими неточностями) и корректно в 2010 и 2011 годах (о «куче» статей и двух других толстых книгах на

эту тему, опубликованных мною к этому времени, я не говорю). При этом отмечу, что **в Интернете эта книга опубликована авторской редакции**: https://is.ifmo.ru/books/_book.pdf.

В 2014 г. эта книга стала «вечной» – в продаже появилось ее электронная копия (https://www.piter.com/product/avtomatnoe-programmirovanie).

Она была и на «Литрес» (https://www.litres.ru/book/anatoliy-shalyto/avtomatnoe-programmirovanie-585075/). Приведу некоторые комментариев оттуда: «Мой первый опыт программирования в данном стиле был связан с разработкой ПО под микроконтроллер *STM32* для задачи управления несколькими двигателями с учетом показаний датчиков. Коллега порекомендовал почитать книгу «Автоматное программирование» и воспользоваться методами, указанными в ней. **Результат превзошел мои ожидания: объем кода удалось сократить раза в два, а читаемость улучшилась**. Примеры в книге понятны, язык доступен. Автоматное программирование оказало существенное влияние на мой стиль разработки ПО не только для микроконтроллеров, но и десктопных систем» (***AlexisVaBel***, 12 лайков), «По-моему, это очень глубокая книга, и при этом краткая. Автоматное программирование – один из подходов, который должен помочь решить основное противоречие современного *IT*: чрезвычайная сложность и частая изменчивость логики программ при одновременном требовании к их высочайшей надежности (предсказуемость, однозначность поведения и корректная обработка исключительных ситуаций)» (***autoreg***, девять лайков), «Автоматное программирование позволяет единообразно осуществлять разработку программного обеспечения, предназначенного для управления логическими контроллерами. Очень хорошо, что Шалыто настойчиво продвигает свою идею и хорошо, что это отечественное ноу-хау» (***dastini***, четыре лайка).

9. На второй лекции, которая прошла через год, Моковнин про автоматное программирование не сказал ни слова. Видимо, еще не знал о существовании автоматного программирования – ну, что делать?

10. На третьей лекции, состоявшейся еще через четыре года, за 10 минут до ее окончания (засечка 1.00.12) **Кирилл вдруг стал отчаянно хвалить применение конечных автоматов и автоматного программирования применительно к построению ментальных программ**. Сначала он сказал, что автоматы применимы, если имеется процесс с дискретными состояниями. Поэтому **автоматы могут использоваться практически везде**: модель компьютера – совокупность автоматов, *User Interface* – то же самое, объект в объектно-ориентированном программировании – конечный автомат, в котором **надо различать управляющие и вычислительные состояния** (на их различие в свое время обратил внимание я) и т. д.

11. Как выглядит код без выделенных автоматов? Он похож на лапшу из условных переходов, в которой очень трудно разобраться и внести изменения. Когда такую программу переписывают с использованием автоматов, то по ней сразу видно, что она делает. Автомат отражает реальность. На самом деле переходы происходят именно так, как они описаны в автомате. Вместо набора переменных (флагов), косвенно реализующих состояния, при использовании автоматного программирования они с самого начала должны быть выделены явно, потом они также явно будут отражены в коде. Конечными автоматами, как отмечено выше, описывается практически все, с чем мы работаем.

Их достоинство состоит в том, что, по мнению Кирилла, при использовании автоматов **не надо разрабатывать никакую документацию** – просто открываете код и видите автомат, а этого достаточно, чтобы понять, что происходит. При таком подходе к программированию сам процесс как бы отделяется от кода, который его реализует. **Автомат описывает поведение декларативно** – как описываете его поведение, он так и работает. **Автоматы легко верифицировать** (по этому вопросу много лет назад с моим участием в издательстве «Наука» была опубликована книга, https://is.ifmo.ru/verification/velder_verification_posobie_nauka.pdf).

По смыслу автоматы близки к ментальному программированию, и остается только понять, как их использовать в коде. **Автоматное программирование, которое также называют программированием с явно выделением состояний** (кстати, оба эти термина предложил я), ***является одной из главнейших парадигм программирования*** (автоматный подход к написанию программ я в свое время назвал автоматной парадигмой программирования). «**Это, действительно, фундаментальная вещь, относительно которой никто не спорит»,** – сказал Кирилл.

Последний слайд этой лекции назывался «Автоматно программирование». **«Кто еще не использует автоматы, прочтите соответствующую книжку и начните их использовать»,** – в заключение сказал лектор.
12. Я в основном согласен с изложенным, тем более что в первом же комментарии к этому ролику, имевшему 33 лайка, была указана книга, на которую человек, призвавший все делать явно, сослался неявно.

Приведу этот комментарий: «Очень интересно и познавательно! Книга по программированию с использованием конечных автоматов, которую на лекции упомянул Кирилл – это ***Поликарпова Н.И., Шалыто А.А.*** Автоматное программирование». По моему мнению, этот комментарий было написать нетрудно, так как другой книги по этой тематике, по крайней мере на русском языке, нет!

После этого я прокомментировал утверждение Кирилла, с которым, якобы, никто не спорит: «О фундаментальности автоматного программирования никто не спорит только потому, что о нем практически никто даже не слышал, так как оно как парадигма программирования практически нигде не преподается, а даже там где оно преподавалось, последующие поколения студентов о нем тоже не слышали, хотя я 25 лет постоянно пишу и говорю о нем в русскоязычном Интернете. Но для того, чтобы автоматное программирование в нем обнаружить нужно, по крайней мере, сделать соответствующий запрос в поисковике, что практически никому даже в голову не приходит!».

А еще я не согласен с Кириллом в том, что при использовании автоматного программирования не нужна документация, так как она в этом случае из программной становится проектной, причем достаточно высокого качества (https://is.ifmo.ru/projects/).
13. К четвертой лекции про ментальное программирование, состоявшейся еще через три года, Кирилл, видимо, уже забыл о существовании автоматного программирования.
14. В этом уже, ставшем весьма длинным тексте, только в самом его начале были упомянуты **графы переходов, которые в части поведения могут быть не только языком спецификации, но и языком программирования**. При этом драйверы, реализующие входные и выходные переменные, обычно реализуются на другом языке.
О других особенностях автоматного программирования можно прочитать здесь: https://vk.com/@1077823-pochemu-v-epohu-neironnyh-setei-dlya-upravleniya-otvetstvenn
05.11.2023. https://vk.com/@1077823-pechalnaya-istoriya-o-primenenii-konechnyh-avtomatov-v-progr

Об объектно-ориентированном и автоматном программировании

Я уже много лет **развиваю парадигму автоматного программирования**: ***Шалыто А.А.*** Парадигма автоматного программирования // Вестник информационных технологий и оптики. **2008**. № 5 (53), с. 3-24 (https://is.ifmo.ru/works/2008/Vestnik/53/01-automata-based-programming.pdf).

В **2002 г.** при реализации автоматных программ мы использовали процедурный подход (https://is.ifmo.ru/projects/dg/). Чуть раньше (**2001 г.**) мы применяли реализацию, которая была названа «**оборачивание автоматов классами**» (https://is.ifmo.ru/projects/tanks/). После этого (**2003 г.**) у нас появилась работа (https://is.ifmo.ru/projects/robocode2/), в которой, частности, сказано: «**Выполнен переход от процедурного программирования с использованием классов к более полному использованию объектно-ориентированного программирования (ООП)**». В том же **2003 г.** мы опубликовали ещё один подход к реализации объектно-ориентированных систем с явным выделением состояний (http://www.i-us.ru/index.php/ius/article/view/14376). В **2004 г.** нами был **предложен новый паттерн для объектно-ориентированного проектирования *State Machine*** (http://www.i-us.ru/index.php/ius/article/view/14476).

В **2006 г.** было создано инструментальное средство ***MetaAuto*** для автоматической генерации кода автоматных программ на любом априори заданном языке программирования по графам переходов автоматов (http://www.i-us.ru/index.php/ius/article/view/14568). В **2007 г.** была сделана ещё одна попытка применения ООП для реализации автоматных программ, и было создано инструментальное средство ***UniMode*** (http://is.ifmo.ru/works/_2008_01_27_gurov.pdf, https://sepl.dibris.unige.it/publications/2012-ricca-MiSE.pdf, https://www.youtube.com/watch?v=Y4et51dz-HE).

В этом же **2007 г.** нами была опубликована работа о преобразовании диаграмм переходов, применяемых в автоматном программировании, в исполняемый код на основе использования

динамических языков программирования, возможности которых позволяют добиться изоморфности диаграммы и соответствующего программного кода (http://www.i-us.ru/index.php/ius/article/view/14683).

В **2008 г.** мы выпустили **первый в мире сборник по различным аспектам автоматного программирования, содержащий 28 статей** (https://ntv.ifmo.ru/ru/journal/61/journal_61.htm). По этому адресу приведены только их рефераты, но все они были опубликованы в сети. Например, статья, в которой предлагается декларативный подход к реализации автоматных объектов при использовании объектно-ориентированных императивных языков программирования со статической проверкой типов, размещена по адресу: https://is.ifmo.ru/works/_20_astafurov.pdf.

В том же **2008 г.** мы написали книгу про автоматное программирование (https://is.ifmo.ru/books/_book.pdf).

В **2009 г.** у нас появилась работа о наследовании автоматных классов с использованием динамических языков программирования (http://www.i-us.ru/index.php/ius/article/view/14860).

Работы по автоматному программированию мы продолжали и в дальнейшем. Так в **2012 г. в одном сборнике было опубликовано 17 статей по этой тематике** (https://ntv.ifmo.ru/ru/journal/28/journal_28.htm), однако этих работах рассматривались вопросы, отличные от указанных выше.

Автоматный подход мы использовали по-разному, в том числе и для программирования ПЛИС (http://www.i-us.ru/index.php/ius/article/view/13825, https://www.youtube.com/watch?v=YNWdmnwHZi8), однако несмотря столь большое число наших работ по автоматной тематике меня до сих пор не покидает неудовлетворенность тем, как взаимодействуют автоматная и объектно-ориентированная парадигмы программирования.

Недавно я познакомился с **Егором Бугаенко** (https://d-russia.ru/dobroe-slovo-ob-iskusstvennom-intellekte.html), который, кроме того, что является классным блогером (https://www.youtube.com/@yegor256), еще пытается ООП сделать по-настоящему объектным.

Для этого он, в частности, пишет книги. В предисловии к книге Элегантные объекты. *Java Edition*. СПб.: Питер. 2018, 240 с. (https://k0d.cc/storage/books/JAVA/Элегантные_объекты_Java_Edition_(Бугаенко_2018).pdf) Егором сказано: «Об ООП написано много книг. Зачем нужна еще одна? Затем, что мы в опасности. **Мы все дальше уходим от того, что было задумано создателями ООП, и у нас все меньше шансов вернуться**. **Все существующие ООП-языки предлагают рассматривать объекты как структуры данных с прикрепленными процедурами**, **что в корне неверно**. Появляются новые языки, но они делают так же или даже хуже. **Объектно-ориентированных программистов заставляют думать так, как процедурные программисты думали 40 лет назад – думать не как объекты, а как компьютеры»**.

Он продолжает: «Мне кажется, что **ООП было разработано для решения проблем процедурного программирования, особенно на языках вроде *C***. Процедурный стиль написания кода очень прост для понимания теми, кто знает, что процессор последовательно обрабатывает инструкции, манипулирующие данными в памяти. Фрагмент кода на C, также известный как функция, – это множество операторов, которые должны выполняться в хронологическом порядке, перемещая данные из одного места в памяти в другое и попутно проделывая над ними некоторые преобразования. Это работало много лет и работает до сих пор. Таким образом написана большая часть программного обеспечения, включая, к примеру, все основные *Unix*-подобные операционные системы. Такой подход технически работает – код компилируется и за пускается. Но при этом существует проблема с сопровождением. Автор кода более или менее понимает, как тот работает, пока пишет его. Но если заглянуть в него позже, то будет довольно трудно выяснить, что имел в виду его создатель. Иными словами, **код написан для компьютеров, а не для людей**. Лучший пример такого императивно-процедурного языка – ассемблер. Он ближе всего к процессору и очень далек от языка, на котором люди общаются в жизни. В ассемблере нет клиентов, файлов, прямоугольников и цен. Только регистры, байты, биты и указатели – то, что процессор понимает лучше всего.

Так было много лет назад, когда компьютеры были большими, медленными и повелевали всем. **Мы вынуждены были говорить на их языке, а не наоборот**. Так происходило преимущественно потому, что программное обеспечение должно было быть быстрым, чтобы стать полезным. Мы больше беспокоились о скорости и использовании памяти, чем о сопровождении кода. Важно отметить, что программисты тогда были на много дешевле компьютеров. Именно **поэтому приходилось делать то, что диктовали нам процессоры**».

Ещё от Егора: «**Через некоторое время ситуация изменилась, и проблема сопровождения стала более важна, чем скорость исполнения или расходование памяти**. Стало очевидно, что ассемблерный код не сможет пережить смену команды – новые люди предпочтут переписать код вместо того, чтобы разбираться длиной программе на ассемблере. **Так появились более высокоуровневые парадигмы программирования. Они перенесли фокус внимания с машин на людей**. Они позволили нам говорить на своем языке, а не на том, к которому привык процессор. Они помогли сделать код более читаемым и, как следствие, более простым для поддержки. Так было задумано.

Исторически ООП унаследовало многое от процедурного программирования. Под ООП здесь понимается не парадигма, а семейство популярных языков программирования, которые были названы объектно-ориентированными. Речь идет в основном о *C++* и *Java*. Из-за компромиссов в переходе от *C* к *C++* и от *C++* к *Java* ООП на сегодняшний день сильно напоминает процедурный *C*.

В этих языках есть классы и объекты, но все ещё остаются операторы, инструкции и их последовательное исполнение. Мы больше не работаем напрямую с указателями, памятью и регистрами процессора, но основной принцип остается неизменным – мы даем инструкции процессору и манипулируем данными в памяти. «Что с этим не так?» – можете спросить Вы. Все в порядке, если Вы хотите придерживаться процедурного подхода. Так же, как все было в порядке с ассемблером. Кроме того, что написанный на нем код было практически невозможно поддерживать. Точно такая же проблема сейчас и с **ПО, написанным, например, на *Java* – его невозможно поддерживать, поскольку оно никогда не было объектно-ориентированным. В этой книге я рассказываю об ООП, лишенном указанных недостатков**».

А ещё Егор написал: «**Императивное программирование «описывает вычисления в терминах операторов, изменяющих состояние программы**», а «**Декларативное программирование выражает логику вычисления, не описывая поток его выполнения**» (цитирую «Википедию»). Императивное программирование похоже на то, что делают компьютеры, обеспечивая последовательное выполнение инструкций. Декларативное программирование ближе к естественному образу мышления, в котором у нас есть сущности и отношения между ними. **Очевидно, что декларативное программирование – более мощный подход, но императивный подход понятнее процедурным программистам**».

Поэтому **я спросил Егора**: «**Первое определение интересное: операторы меняют состояния программы, но какие состояния, ведь их в традиционном программировании, в отличие от автоматного, никто не вводит и не определяет**. Второе определение и про автоматное программирование тоже? Тем более, что графы переходов автоматов можно рассматривать не только, как язык спецификаций, но и как основу языка программирования сверхвысокого уровня (https://ntv.ifmo.ru/file/article/21921.pdf)».

Ответ Егора: «Да, **автоматное программирование очень близко к тому ООП, что мы пытаемся проповедовать.** Поэтому меня очень заинтересовало автоматное программирование. В нашем ООП все операторы суть объекты – они также еще и функции. **Автоматное программирование – это, видимо, частный случай ООП, более строгий что ли**».

Я попросил Егора подумать на тему создания объектно-ориентированного автоматного программирования, которое может быть названо также объектно-ориентированным программированием с явным выделением состояний.
02.12.2023. https://vk.com/@1077823-ob-obektno-orientirovannom-i-avtomatnom-programmirovanii

Возможно, что автоматное программирование еще не раз пригодиться

В книге [1] я написал: «**В 2000 г. я и Никита Туккель познакомились с Сергеем Александровичем Вагановым**, который предложил новую среду программирования, **в которой объектно-ориентированная программа строилась на «дереве»**. Мы совместными усилиями добавили автоматный подход в эту среду (***Туккель Н.И., Шалыто А.А., Ваганов С.А.*** Использование *SWITCH*-технологии при разработке программ в среде *FLORA/C++* (модель технологического процесса в цехе холодной прокатки), http://is.ifmo.ru/projects/cold/). Она показалась нам весьма интересной, и мы решили показать ее второкурсникам кафедры «Компьютерные технологии» СПбГИТМО, на которой с 1998 г. я работал по совместительству. В обсуждении самыми активными, естественно, были **Георгий Корнеев** и **Андрей Станкевич**, с которыми я тогда не был еще знаком. Автоматное программирование их тогда не увлекло». Через несколько лет Корнеев под моим руководством защитил кандидатскую диссертацию по применению … автоматного программирования при построении визуализаторов алгоритмов дискретной математики [2].

В то время Сергей Александрович был начальником отдела систем реального времени в компания *Compass Plus* («Компас Плюс») (Магнитогорск). (https://compassplusonline.ru/about/about-company.html), которая существует с 1989 г. и разрабатывает программные продукты для процессинговых центров и розничных банков. Ее конкурентами, в частности, являются компании *OpenWay*, в которой работало много выпускников нашей кафедры, и *Solanteq*, организованной нашим выпускником 2009 г. **Дмитрием Кочалаевым** [3].

Ко времени нашего знакомства под руководством Сергея Александровича была разработана технология программирования ***FloraWare***, которая реализует объектный метод и содержит объектно-ориентированную среду разработки и исполнения приложений больших программных систем. Главная отличительная особенность этой технологии – реализация объектной модели непосредственно на объектной машине без создания промежуточной программы. В этом случае собственно написание программы можно исключить из цикла создания модели, которая будет собираться из стандартных готовых объектов. При этом для хранения объектов применялось дерево. Как и положено Флоре – богине цветов, расцвета, весны и полевых плодов – вагановская флора была прекрасна, но стать еще лучше ей мешала распределенная по всем обработчикам событий логика. Этот недостаток устранялся применением предложенной мною технологии автоматного программирования [4], одна из разновидностей которой была предназначена для программной реализации событийных объектно-ориентированных систем и была разработана совместно с Н. Туккелем [5]. При этом нами было предложено практически всю логику «вынести» из обработчиков событий в автоматы, к которым события только «обращались». Это позволило резко повысить централизацию логики программ этого класса.

На эту особенность нашего подхода понравилась Ваганову [6]. Вот, что он писал по этому поводу [7]: «Использование автоматного программирования органично вписалось во *FloraWare*. Это обусловлено тем, что в отличие от известных реализаций, где конечный автомат представляется текстом на языке программирования, во *FloraWare* конечный автомат представляет собой объект, реализованный на уровне ядра объектной машины. Разработка и внедрение объектов автоматного программирования – это качественно новая ступень в развитии технологии *FloraWare*. Для того чтобы это подчеркнуть, *FloraWare*, содержащая наряду с аппаратом шаблонов, объекты автоматного программирования и инструменты для работы с ними, получила рабочее название «*FloraWare 2*».

Эти события Сергей Александрович описывает также и более подробно: «Разработка концепции, архитектуры и реализация среды разработки *FloraWare* были выполнены в 1996-1998 гг. группой аналитиков и системных программистов компании «Компас Плюс». В 1998 и 1999 гг. *FloraWare* прошла апробацию в проекте по реализации программных продуктов для процессинговой системы электронных платежей ***TranzWare Online.*** В 2002 г. вышла вторая версия *FloraWare*, которая, в частности, была дополнена средствами автоматного программирования. Указанная система в настоящее время используется и, как положено программному обеспечению с четверть вековой историей, поддерживается в ряде отечественных и зарубежных банков (https://www.tadviser.ru/index.php/Компания:Compass_Plus).

Рассматриваемая технология в то время использовалась также в системах контроля и управления промышленными агрегатами. Ниже приведены ссылки на учебный пример использования

автоматного программирования при разработке программ в среде *Flora/C+*. Приведенная там программа реализует визуальную модель технологического процесса в цехе холодной прокатки и основана на аналогичном примере, поставляемом компанией «Компас Плюс» вместе со средой *Flora/C+* версии 2.1.4.0 [8, 9].

Обсуждаемая технология была создана большим коллективом молодых разработчиков под руководством **Анатолия Виленовича Капцана** и **Сергея Александровича Ваганова**, причем, как считает Ваганов, его вклад в создание *FloraWare* меньше вклада Анатолия Виленовича. По их мнению, затраты на ее разработку более 100 человеко-лет.

Подробный обзор рассматриваемой среды разработки приведен в [10].

Прошло 20 лет, и **я вновь услышал о построении объектно-ориентированных программ на деревьях**. Это произошло в ходе интервью, взятого **Егором Бугаенко** (https://www.youtube.com/@yegor256) у **Алексея Недоря** [11]. При этом было предложено назвать **такой подход к написанию программ »Архитектурным программированием»**. Я слушал это интервью, так как не так давно Егор брал интервью у меня [12]. После этого я позвонил Алексею, с котором был знаком, и сообщил ему о наших работах с Вагановым, о которых он, естественно, не знал. При этом я предположил, что применение автоматного программирования может помочь Алексею в его работе, как это произошло у Ваганова.

Алексей поблагодарил и пообещал посмотреть приводимые ниже публикации. Жду «приговора».

Литература

1. ***Шалыто А.А.*** Мои счастливые годы жизни на кафедре «Компьютерные технологии» Университета ИТМО. К двадцатилетию кафедры. СПб.: Мозаика-НК. 2012 (https://is.ifmo.ru/belletristic/Shalyto-moi-shastlivye-gody-na-CT.pdf).

2. ***Корнеев Г.А.*** Автоматизация построения визуализаторов алгоритмов дискретной математики на основе автоматного подхода // Диссертация на соискание ученой степени кандидата технических наук. СПбГУИТМО. 2006 (https://is.ifmo.ru/disser/korn_disser.pdf).

3. ***Кочелаев Д.Ю.*** Проектирование, спецификация и реализация автоматизированных классов // Магистерская работа. СПбГУИТМО. 2009 (https://is.ifmo.ru/diploma-theses/_2010_03_03_kochelaev.pdf).

4. ***Шалыто А.А.*** *SWITCH*-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998, 628 с. (https://is.ifmo.ru/books/switch/1).

5. ***Туккель Н.И., Шалыто А.А.*** *SWITCH*-технология – автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5, с. 42-62. (https://is.ifmo.ru/works/switch/1/).

6. ***Туккель Н.И., Шалыто А.А., Ваганов С.А.*** Повышение централизации управления при программировании «реактивных» систем / Труды международной научно-методической конференции «Телематика'2001». СПбГИТМО (ТУ), 2001, с. 176, 177 (https://www.avrorasystems.com/ru/Data/Pressroom/Files/gitmo2.pdf).

7. ***Ваганов С.А.*** Автоматное программирование во «Флоре». 2001 (https://is.ifmo.ru/automata/vaganov/).

8. ***Туккель Н.И., Шалыто А.А., Ваганов С.А.*** Использование *SWITCH*-технологии при разработке программ в среде *FLORA/C+* (Модель технологического процесса в цехе холодной прокатки). 2001 (https://is.ifmo.ru/projects/cold/, http://www.softcraft.ru/design/cold/).

9. ***Туккель Н.И., Шалыто А.А., Ваганов С.А.*** Использование *SWITCH*-технологии при разработке программ в среде *FLORA/C+* (Модель технологического процесса в цехе холодной прокатки). Проектная документация. 2001 (https://is.ifmo.ru/download/cold.pdf).

10. ***Ваганов С.А.*** *FloraWare* – ускоритель разработки приложений (подробный обзор среды программирования). 2004 (http://www.softcraft.ru/paradigm/oop/flora/, https://www.osp.ru/os/2004/06/184465).

11. ***Недоря А.Е.*** Никлаус Вирт, Архитектурное программирование, Компиляторы, Кронос, Тривиль (https://www.youtube.com/watch?v=FGXws8lrkoI).

12. ***Шалыто А.А.*** ИТМО, *ICPC*, *JetBrains*, лучший айти стендап в России! (https://www.youtube.com/watch?v=TlO6lPZrC3g).

12.02.2024. https://vk.com/@1077823-vozmozhno-chto-avtomatnoe-programmirovanie-esche-raz-prigodi

P.S. 1. После публикации этого текста получил вот такое письмо от моего выдающегося (по *ICPC*, в частности) ученика **Матвея Казакова**: «Слегка удивлен названием статьи. **Мне кажется, что автоматное программирование (АП) как инструментарий уже вошло в арсенал профессиональных разработчиков наравне с парадигмами процедурного и объектного программирования**. И уже перешло из экзотики в каждодневный инструмент. Как мне кажется, это бо́льшая победа, чем разрозненные исследования и разработки с использованием АП». Название «Возможно, что автоматное программирование еще раз пригодится» по совету Казакова исправил на более оптимистическое.
2. Дело пошло. Сначала отреагировал Ваганов: «Добрый день! Читаю с интересом. Хотелось бы пообщаться с Недоря: поговорить про *FloraWare*, и его деревья». После этого высказал пару замечаний, которые я учел.
3. Затем написал Недоря: «Анатолий Абрамович! Спасибо, очень интересно! На что сразу обратил внимание: «Главная отличительная особенность этой технологии – реализация объектной модели непосредственно на объектной машине без создания промежуточной программы. В этом случае собственно написание программы можно исключить из цикла создания модели, которая будет собираться из стандартных готовых объектов» и «Использование автоматного программирования органично вписалось во *FloraWare*. Это обусловлено тем, что в отличие от известных реализаций, где конечный автомат представляется текстом на языке программирования, во *FloraWare* конечный автомат представляет собой объект, реализованный на уровне ядра объектной машины». Буду читать и думать, как интегрировать. С уважением, Алексей». Я сообщил Алексею координаты Сергея Александровича. Возможно, что-то из этого получится…
4. После этого Ваганов написал: «Что касается самой идеи объектной машины и древовидной памяти с программой внутри, то их можно распространить на многоязыковые приложения с заменой существующей сейчас концепции исполняемых файлов в ОС. С некоторой периодичностью всплывают вопросы, в основном не сбывшегося, будущего *FloraWare*, и бередят былые раны. Это произошло, например, при чтении текста «ООП будущего: Барух Садогурский и Егор Бугаенко о том, как мы будем программировать через 20 лет». 2016 (https://habr.com/ru/companies/jugru/articles/308914/). На этот текст я давно хочу ответить что-то в духе «**Будущее ООП наступило четверть века назад**» и рассказать про *FloraWare*». В заключение Сергей Александрович предложил переименовать «Архитектурное программирование» в «Древовидное программирование». Я же, в свою очередь, посоветовал Егору, чтобы Ваганов сделал это в его блоге.
5. В указанном выше тексте Бугаенко говорит: «**Я вижу, какой код пишут люди, как тяжело этот код потом понимать, поддерживать, как вообще неприятно программировать, и считаю, что с этим надо что-то делать**. Специалисты говорят, что ООП противная концепция в принципе и предлагают её не использовать, потому что **она помогает создавать *spaghetti code***.
6. Мой ответ на этот вопрос прост: «Используйте в программах вместо спагетти автоматы, и такие программы будут нравиться Вам больше».

Автоматное программирование. Не задушишь, не убьешь!

Для успеха необходимо знать секрет мира – его обычно никто не знает, считает неправильным или нецелесообразным
П. Тиль

23.02.2024 г. в качестве праздничного подарка я получил от неизвестного мне до этого времени **Сергея Муравьева** (smur@inbox.ru) письмо, в котором он сообщил, что еще в 2019 г. на форуме «**Робот с логикой на основе автоматного программирования**» (http://roboforum.ru/forum10/topic18271.html) описал процесс создания робота удаленного присутствия (https://vzikblog.wordpress.com/) **с использованием наших работ по автоматному программированию** (АП) применительно к очень распространенным микроконтроллерам *ATmega 328* (обычно используются на платформе *Arduino*), которые можно купить на даже на «Озоне» и *Aliexpress*.

На указанном форуме в 2023 г. **Александр Капульский**, которой как мне стало известно из переписки с ним, также использует наши работы по автоматному программированию (https://www.youtube.com/watch?v=NxaWKFe5sP8), написал следующее: «Если на АП смотреть с точки зрения теории Тьюринга, то его машина, наглядно подтверждает уникальность АП. Недавно

мне на глаза попалась лекция бакалавра сиднейского технологического университета, причем в такой лаконичной подаче, что мне захотелось перевести материал («Автоматное программирование, машина Тьюринга или как возникла информатика», https://www.youtube.com/watch?v=lQGw1b0ILGw) и представить его в тематических обществах».

После этого **Муравьев написал Капульскому**: «Вы говорите, что парадигма автоматного программирования не стала магистральным трендом, но это не так. Это доказывают многочисленные источники и диссертации, а также многочисленные проекты. **Используйте приводимые ниже источники, когда пишете об автоматном программировании**: Университет ИТМО. Кафедра «Технологии программирования» (https://is.ifmo.ru/); Шалыто Анатолий Абрамович, профессор факультета информационных технологий и программирования (https://itmo.ru/ru/viewperson/94/shalyto_anatoliy_abramovich.htm); Что такое автоматное программирование? (http://is.ifmo.ru/works/tech_aut_prog, http://is.ifmo.ru/works/diesel/, http://is.ifmo.ru/projects_en/).

Капульский внял совету Муравьева и записал ролик и на русском языке (https://www.youtube.com/watch?v=hFoMia3s4NU), в котором рассказано о моей роли в обсуждаемом вопросе.

А тем временем обсуждение на форуме перекинулось к рассмотрению еще одного аспекта автоматного программирования: «Интересной является тема визуального программирования, и, в частности, один из инструментов преобразования графов переходов, представленных в формате *MS Visio*, в код.

20.08.2023 г. на «Форуме пользователей *Visio*» (https://visio.getbb.ru/viewtopic.php?f=18&t=792&st=0&sk=t&sd=a&start=40) участник под ником ***gtfox*** написал: «*Visio* довольно давно, минимум с 2002 г. (https://is.ifmo.ru/automata/visio2switch/), используется для решения задач «Автоматизации программирования задач логического управления» (приводится ссылка на нашу с **Н. Туккелем** статью в журнале «Программирование», размещенную по адресу https://is.ifmo.ru/download/switch.pdf). Соответствующая технология была названа «*Switch*-технология» (https://ru.wikipedia.org/wiki/Switch-технология). В 2005 г. (приводится ссылка на нашу с **С. Канжелевым** работу, размещенную по адресу https://is.ifmo.ru/projects/metaauto/) был создан конвертер ***MetaAuto***, а в 2008 г. он был преобразован мною для программирования ПЛК на языке *ST* стандарта МЭК 61131-3. После этого я пиарил *Switch*-технологию и это средство на разных ресурсах: Owen.ru (https://owen.ru/forum/showthread.php?t=11839), Segnetics.com (https://forum.segnetics.com/showthread.php?t=223), Asutpforum.ru (https://asutpforum.ru/viewtopic.php?start=75&t=1450&sid=9014ba00448175949e46a01fc2217f46)».

Интересно, что на Owen.ru приведены также результаты голосования 64 участников, проведенного в 2008 г. опроса на тему «Автоматы в логическом управлении». Вот они: **использую – 40,63%, буду использовать – 23,13%**, не использую – 9,38%, о чем речь – 21,88%.

Суть подхода пользователя ***gtfox*** состоит в следующем: по графу переходов конечного автомата получить исходный текст программы на целевом языке программирования (*ST* или *C++*). Этот конвертер *METAAUTO_VISIO_ST_CPP* (https://disk.yandex.ru/d/QkHH8ReftIzVUw) хоть и с открытыми исходниками, но получился сложным и неудобным в использовании. Автор предполагает написать всё это на чистом *Visual Basic for Application (VBA)* в *Visio*.

Далее ***gtfox*** пишет, что ему недавно (17.08.2023 г.) на Хабре попалась статья «Графическое программирование конечных автоматов для *Arduino*. Часть 1» (https://habr.com/ru/articles/755292/), в которой автор с ником ***DonEgen*** выполнил реализацию соответствующего конвертора в *VBA*, правда не для ПЛК, а для микроконтроллеров.

Вот что пишет ***DonEgen*** во введении к этой статье: «Представление программного кода в виде конечного автомата подкупает надежностью и предсказуемостью результатов. Действительно, *finite-state machine* всегда находится только в одном состоянии, что значительно облегчает поиск ошибок. Однако ее код обычно весьма объемен, и поэтому его часто невозможно охватить взглядом. Поэтому фреймворки, помогающие создавать только код, не пользуются популярностью. Другое дело, графическое представление конечного автомата, например, *Stateflow* в *Matlab,* но *Matlab* – это очень дорого. Куда больше мне понравился продукт от *Quantum Leaps* (https://www.state-machine.com/).

Это именно то, что хотелось бы видеть у себя в лаборатории! Однако конские цены полностью отбивают всё желание (https://www.state-machine.com/licensing#Commercial).

Выход нашелся – работы Анатолия Шалыто под общим названием «Автоматное программирование» (https://is.ifmo.ru/books/_umk.pdf). Там очень много полезной теории, но наибольший интерес для меня представляла его ранняя разработка: программа трансляции графов переходов автоматов, изображенных в *Microsoft Visio*, в код на языке *C* (https://is.ifmo.ru/works/_autogen.pdf). Это было именно то, что нужно. Увы, Анатолий Абрамович углубился в академическую деятельность (всецело одобряю), и полезная программа оказалась заброшена (не совсем: в НПО «Аврора» ее по сей день использует **А. Калачинский** (https://is.ifmo.ru/works/_volobuev.pdf), А.Ш.).

Пришлось повторять уже сделанное, используя его концепцию автоматного программирования в качестве стартовой точки. В результате получился программный комплекс ***V2S*** (https://github.com/donchenko-egen/StateMachines). За время эксплуатации этот комплекс позволил разработать эффективный код в ряде коммерческих проектов. Из самых крупных – система контроля высева пропашных культур «Топаз» на базе микроконтроллера *STM32F412*».

Вторая часть этой статьи посвящена описанию использования этого программного комплекса при создании проекта «Светофор» на аппаратной платформе *Arduino* (https://habr.com/ru/articles/756512/).

Из обсуждения этих статей:

– «Программа на языке программирования с приличным функционалом вообще не воспринимается посторонними, а в графическом варианте ее можно обсуждать хоть с руководством, хоть с заказчиком»;

– «Автоматное программирование оперирует состояниями конечного автомата (алгоритма поведения системы) и переходами между ними. При этом чем сложнее алгоритм, тем легче его понимание в графическом виде по сравнению с пониманием традиционно написанной программы»;

– «С 2008 г. применяем *Switch*-технологию в программировании промышленных ПЛК для автоматизации задач логического управления. Для этого используем конвертер *MetaAuto*, переделанный для программирования ПЛК под язык ***ST*** стандарта МЭК 61131-3»;

– «Программный комплекс *V2S* опирается на *Switch*-технологию, автор которой тоже А. Шалыто»;

– «Эта технология у Вас применяется для управления с использованием ПЛК, а у меня – для создания программ для встраиваемых микроконтроллеров»;

– «Приведенный пример «Светофор» демонстрирует преимущества автоматного программирования. Я, не глядя в код, смог понять алгоритм работы, смог легко его изменить и получить исходный текст программы, избежав синтаксических ошибок в ней»;

– «Еще немаловажно, что все это можно легко повторить и через год, и через два, если вдруг понадобиться модернизировать алгоритм. А еще графическое представление в виде автоматов легко обсуждать с заказчиком/руководством, которые зачастую в программировании ничего не понимает, но в визуальном виде хотя бы воспринимают что там происходит. Это тоже очень важно, так как много раз сталкивался, что заказчик сам не знает чего хочет, откуда и вытекают многочисленные доработки. Дорабатывать программу в визуальном виде тоже весьма просто».

Этот текст начинался с письма С. Муравьева, этим письмом я хочу его и закончить. Вот что он еще написал: «Нашлась неточность в Ваших материалах на https://is.ifmo.ru/projects/metaauto/. Для тех, кто хочет использовать данный инструмент без глубокого погружения в тематику – это может быть важно. Данная неточность не позволяет синтезировать исходный код из шаблона для случаев задания выходных функций на переходе между состояниями. Далее указано изменение кода, опубликованное по адресу, указанному ниже

Там есть еще несколько неточностей, которые я увидел в процессе использования данного инструмента, но это один из неочевидных, и поэтому его не так быстро найти, не зная логику работы программы».

Я поблагодарил Сергея и написал этот текст.

23.02.2024. https://vk.com/@1077823-avtomatnoe-programmirovanie-ne-zadushish-ne-ubesh