Заключение

Изложенное в настоящей работе можно обобщить в виде технологии разработки функционального программного обеспечения для систем логического управления, базирующейся на теории конечных детерминированных автоматов и ее центральном понятии — «состояние».

При использовании языков программирования высокого уровня рассмотренный подход был назван SWITCH-технологией, являющейся одним из направлений широко развиваемой в настоящее время CASE-технологии [248—250].

Разработанная технология может быть также названа STATE-технология или более точно AUTOMATON-технология.

При этом отметим, что только в языке СИ применяемая в настоящей работе в качестве основной конструкция носит название «`switch`», в то время как в таких языках, как «Паскаль», «Модула-2», «Алгол-W», она называется «`case . . . of`», в языке ПЛ/М и в некоторых версиях «Фортрана» — «`do case`», в языках «Алгол-68», «Форт» и ST [260] — «`case`», в языке АДА — «`case. . .is`», а в языке «Бейсик» — «`on. . .goto`», причем, например, в языке «Турбо Паскаль» допускается использование вложенных конструкций этого типа [256].

Предлагаемая технология обеспечивает упрощение взаимодействия Заказчика, Технолога (Проектанта), Разработчика, Программиста, Пользователя (Оператора) и Контролера, а также помогает перераспределить их роли, вплоть до отказа, например, от услуг «функционального» Программиста.

Рассматриваемый подход позволяет в значительной мере формализовать построение и доказательство правильности программ, а кроме того, обеспечить их однотипность, наглядность, компактность, структурированность, иерархичность, наблюдаемость, управляемость, вызываемость и вложенность.

Технология предполагает четыре стадии разработки, каждая из которых состоит из ряда этапов: архитектурное (системное) проектирование (этапы *1*—7); проектирование формальных спецификаций (этапы *8*—17); построение и оптимизация алгоритмических моделей, реализующих формальные спецификации (этапы *18—20);* программирование (этапы *21—33)*. Перечислим этапы предлагаемой технологии.

Стадия 1:

1. Словесное описание объектов управления (ОУ) с перечислением их источников информации (ИИ) и исполнительных механизмов (ИМ);

2. Словесное описание «алгоритма» управления, заданного Заказчиком;

3. Выбор Разработчиком органов управления (дополнительные ИИ) и средств представления информации (СПИ);

4. Словесное описание «алгоритма» управления с учетом дополнений, предложенных Разработчиком;

5. Построение схемы связей «ИИ—УА (управляющий автомат)—СПИ—ИМ»;

6. Декомпозиция УА на автомат (А) и функциональные элементы задержки (ФЭЗ). Построение схемы связей «ИИ—А—ФЭЗ—СПИ—ИМ»;

7. Декомпозиция (при необходимости) автомата на систему взаимосвязанных автоматов (СВА). Построение схемы связей «ИИ—СВА—ФЭЗ—СПИ—ИМ», в которой с *i*-м автоматом связаны «свои» ФЭЗ, образуя *i*-й управляющий автомат.

Стадия 2:

8. Выбор графов переходов в качестве языка спецификаций;

9. Выбор структурной модели каждого автомата (например, модель автомата Мура);

10. Выбор вида кодирования состояний каждого автомата (например, многозначное кодирование);

11. Построение «автоматной» спецификации в общем случае в виде системы взаимосвязанных графов переходов (СВГП), структура каждого из которых однозначно соответствует выбранным структурной модели и виду кодирования состояний автомата;

12. Обеспечение корректности (полнота, непротиворечивость и отсутствие генерирующих контуров) в каждом графе переходов;

13. Построение по СВГП графа достижимых маркировок (ГДМ) и определение на основе его анализа правильности «автоматной» спецификации;

14. Корректировка (при необходимости) СВГП и построение нового ГДМ;

15. Согласование «автоматной» спецификации с заинтересованными Специалистами;

16. Построение корректных спецификаций ФЭЗ;

17. Построение корректных спецификаций моделей ОУ.

Стадия 3:

18. Выбор, построение и оптимизация алгоритмических моделей для реализации «автоматной» спецификации с учетом рода структурной модели каждого автомата с памятью (например, система булевых формул, реализующая граф переходов автомата Мура, являющегося автоматом этого класса второго рода);

19. Выбор, построение и оптимизация алгоритмических моделей для программной реализации ФЭЗ;

20. Выбор, построение и оптимизация алгоритмических моделей для программной реализации моделей ОУ.

Стадия 4:

21. Выбор языка программирования;

22. Выбор, построение и оптимизация программных моделей для реализации алгоритмических моделей, соответствующих «автоматной»

спецификации (например, две конструкции `switch` языка СИ, изоморфно реализующие такую алгоритмическую модель, как граф переходов автомата Мура, являющегося автоматом этого класса второго рода);

23. Выбор, построение и оптимизация программных моделей для реализации ФЭЗ;

24. Выбор, построение и оптимизация программных моделей объектов управления;

25. Формальная и желательно изоморфная реализация с помощью управляющей программы алгоритмических моделей, выбранных для «автоматной» спецификации и спецификации функциональных элементов задержки;

26. Сертификация управляющей программы с помощью ГДМ;

27. Корректировка (в случае необходимости) управляющей программы;

28. Построение комплексной программы «управляющая программа-программа, моделирующая объекты управления»;

29. Сертификация комплексной программы с помощью ГДМ;

30. Корректировка (в случае необходимости) «автоматной» спецификации и построение нового ГДМ;

31. Повторное (в случае необходимости) выполнение этапов *25—30* с целью получения корректной управляющей программы;

32. Корректировка (в случае необходимости) «автоматной» спецификации, ГДМ и управляющей программы по результатам испытаний с использованием физических моделей объектов управления (при их наличии);

33. Корректировка (в случае необходимости) «автоматной» спецификации, ГДМ и управляющей программы по результатам испытаний и эксплуатации на реальных объектах управления.

Обратим внимание, что построение корректной спецификации за один этап (этап *17*) возможно только для «простых» объектов управления. Для сложных объектов создание их спецификации может быть не проще, чем построение «автоматной» спецификации.

Методика построения спецификации в виде одного графа переходов изложена в разд. 4.4.9, а пример ее использования приведен в Приложении 12.

Для повышения качества разработки управляющих программ создана «оболочка», которая позволяет назначить требуемое число входов, выходов и функциональных элементов задержки (включая уставки последних), а также используемое число графов переходов. С помощью клавиатуры на экран ПЭВМ могут быть введены значения входных переменных. При этом программой на экран выводятся вычисленные значения выходных переменных и текущие значения всех ФЭЗ.

Новизна предлагаемого подхода состоит в том, что на экран также выводятся значения N многозначных переменных, кодирующих состояния N автоматов системы, что делает построенную программу полностью наблюдаемой не только по входам и выходам, но и по внутренним переменным, превращая ее в «белый ящик» с минимальным числом внутренних переменных.

Изоморфизм графа переходов и конструкции `switch` языка СИ обеспечивает управляемость (изменяемость) программы.

Другая разработанная «оболочка» позволяет на различных «экранах» ПЭВМ изобразить систему взаимосвязанных графов переходов и выполнить ее моделирование. Особенность этой «оболочки» состоит в том, что на каждом «экране» можно наблюдать не только функционирование изображенного на нем графа переходов или его фрагмента, но и информацию о состояниях всех остальных графов системы, невидимых на этом экране. Это позволяет и в этом случае сохранить свойство «наблюдаемости». В этой оболочке имеется возможность как автоматического, так и принудительного перехода от одного «экрана» к другому.

Отметим также, что если выбранный язык программирования отличается от используемого в «бортовом» вычислителе, то перечисленные выше этапы (за исключением этапа *33*) являются только этапами алгоритмизации и моделирования формальных спецификаций, после завершения которых должно выполняться собственно программирование — трансляция откорректированной системы взаимосвязанных графов переходов или окончательного текста управляющей программы на выбранном на этапе *22* языке программирования в текст программы на языке программирования «бортового» вычислителя. При этом трансляция должна выполняться формально (автоматически или вручную) и желательно изоморфно.

В качестве примеров для сравнения различных видов спецификаций и моделей в рамках предлагаемой технологии разработано 40 программ на языке СИ, реализующих три алгоритма управления двумя клапанами с памятью с помощью двух кнопок без памяти (Приложение 5). В Приложении 7 для этих алгоритмов приведены примеры применения функций этого языка, а Приложение 8 содержит примеры этих алгоритмов, реализованных объектно-ориентированными программами. С. И. Ушаковым (НПО «Аврора») предлагаемый подход был использован при разработке программы логического управления шаговым двигателем.

Разработанная технология, в частности, применялась еще в 1991 году при программной реализации алгоритмов логического управления дизель-генератором лесовоза проекта 15640 для системы «Selma-2» [269].

Изложенный подход может быть использован и для программной реализации логико-вычислительных алгоритмов при замене объектов управления на операционные блоки, осуществляющие собственно вычисления, что позволяет разделить управляющую и операционную части программ и первоначально проводить их независимую отладку. Пример реализации алгоритма этого класса приведен в Приложении 10.

Предложенная технология может быть применена также и для проектирования других классов управляющих программ, например операционных систем, тем более что язык СИ был разработан специально для написания этого класса программных систем.

Предлагаемый подход содержит элементы как непроцедурных, так и процедурных подходов [133]. С одной стороны, он ориентирован на детальное и компактное описание решаемой задачи, которое изоморфно реализуется программой, а с другой — содержит «элементы» процедуры ее решения, например пометки, указывающие связи между графами переходов. В отличие от традиционных непроцедурных подходов при использовании предлагаемого метода априорно определяются все реше-

ния поставленной задачи, в том числе и за счет доопределения при исходно неопределенных входных воздействиях, что, во-первых, обычно возможно в задачах логического управления, а во-вторых, и необходимо в связи с их большой ответственностью.

В заключение хочется отметить, что побудительным мотивом к написанию данной книги было желание изложить путь автора к решению проблемы, состоящей в том, как понятно и наглядно описывать и корректно и изоморфно программно реализовывать алгоритмы логического управления, в которые достаточно легко могли бы быть внесены изменения.

Как следует из содержания книги, этот путь пролегал через таблицы истинности и таблицы решений, системы булевых функций и булевых формул, лестничные (контактные) и функциональные схемы, секвенции и формулы переходов, матрицы и таблицы переходов, графы переходов и переключений, логические и графические схемы алгоритмов, временные диаграммы и циклограммы, *P*-схемы и диаграммы «Графсет», SFC- и SDL-диаграммы, сети Петри и графы операций, языки инструкций и ассемблеры, проблемно-ориентированные языки и алгоритмические языки высокого уровня и т. д.

На выходе из этого, казалось бы, бесконечного лабиринта я убедил себя в том, что применение пентады «состояние—независимость от «глубокой» предыстории—система взаимосвязанных графов переходов—многозначное кодирование—конструкция `switch`» наиболее полно и просто позволяет решить указанную проблему.

Из изложенного следует, что теперь дело осталось за «немногим»: убедить и других в преимуществах разработанного подхода. Казалось бы, одно то, что, например, для понимания алгоритма по функциональной схеме требуется `считать`, а граф переходов без флагов и умолчаний значений выходных переменных можно `читать`, должно мгновенно убеждать Собеседника в преимуществах предлагаемого подхода. Однако, как показывает опыт, так получается далеко не всегда.

Такая же ситуация имеет место и для ряда других вопросов, рассмотренных в книге. Так, например, из ее содержания видно, сколько усилий было затрачено автором для определения структуры хорошо понимаемых и легко программируемых граф-схем алгоритмов. Сегодня я уже знаю, что такие граф-схемы должны начинаться с дешифратора состояний и не иметь флагов и умолчаний значений выходных переменных. Однако, несмотря на это, даже у моих знакомых программистов число граф-схем с такой структурой увеличилось незначительно.

Поэтому остается только надеяться, что имеет место нормальная ситуация, описываемая одним из законов Мэрфи, по которому каждая идея проходит три стадии: «Это бред!». «Мы об этом догадывались». «Как же может быть иначе!»

На первый взгляд кажется странным, что в конце двадцатого века, когда Человечество достигло невероятных успехов в области вычислительной техники и сетей на ее основе, в том числе и Глобальных, в этой книге столь пристальное внимание уделяется управлению «клапанами». Это направление, казалось бы, досконально изучено к настоящему времени и нечего вновь поднимать этот вопрос.

Однако, по мнению автора, это далеко не так, ввиду того, как показывает опыт, что в настоящее время молодежь считает работу в области вычислительной техники значительно более престижной, чем «управление клапанами». Это при рыночной экономике существенно сказывается на финансировании, подготовке в институтах и университетах, на тематике издаваемых книг и проводимых конференций, оплате труда. Это еще более усугубляет действие принципа Питера, в соответствии с которым самую ответственную работу выполняют самые некомпетентные люди. При этом необходимо учесть также и тот факт, что системы управления ответственными объектами уникальны и для них практически отсутствует возможность выявления и устранения ошибок производителем вследствие применения продукта многими пользователями, как это имеет место для коммерческих программ.

Анализ крупнейших катастроф на «опасных» [324] технологических объектах показывает, что одно неправильное действие оператора или одно неправильное открытие или закрытие «клапана», возникающее из-за ошибки в алгоритмическом и (или) программном обеспечении даже при многократном резервировании аппаратуры, может привести к ужасным последствиям.

Именно неоткрытие клапана подачи химического реагента (или нештатное срабатывание клапана, обеспечивающего подачу горючего) привело к тому, что двигатель космического аппарата в нужный момент времени не запустился и произошел срыв Международных программ и перенос сроков их выполнения на неопределенное время.

Изложенное, а также неудовлетворенность автора руководствами по программированию контроллеров, выпускаемыми крупнейшими в мире (в области автоматизации) фирмами, и книгами по программированию и теории автоматов привели к созданию этой работы.

Завершив написание книги, я прочел удивительную работу [301], которая изменила многие мои представления о поведении динамических систем и о математике вообще. Однако в одном вопросе, несколько раз поднимаемом в настоящей книге, я в этой работе нашел дополнительную опору: чрезвычайно простое формульное рекуррентное нелинейное описание динамической системы может порождать сколь угодно сложное ее поведение вплоть до «хаоса», и поэтому без досконального изучения поведения отдельных автоматов с памятью, каждый из которых также описывается рекуррентными соотношениями, или систем таких автоматов не может быть и речи о возможности корректного использования этих описаний или соответствующих им схем в качестве формальных спецификаций алгоритмов, предназначенных для управления ответственными объектами.

И последнее. «Знания добываются в борьбе ради того, чтобы отыскать существенное и представить его в „двух словах"» [301].

Столь высокого совершенства в рассматриваемом вопросе мне добиться не удалось, но, по-моему, в пяти «словах» — пентаде, приведенной выше, удается сформулировать основную идею исследований, описанных в настоящей работе.

Появление в 1993 г. программного продукта «Modicon State Language» для ПЛК фирмы «Модикон» [299], а в 1996 г. программного продукта

«S7—HiGraph technology software» для ПЛК фирмы «Сименс» [305], в котором в качестве языка программирования используются диаграммы состояний (графы переходов), еще более укрепило уверенность автора в правильности предложенной в 1991 г. [167, 269] и описанной в настоящей работе методологии, которая может стать основой для повышения безопасности программного обеспечения [338] систем рассматриваемого класса.

Без использования технологии алгоритмизации и программирования, направленной на создание «качественных» программ, невозможно решить проблему полноты их проверки, так же как нельзя обеспечить контролепригодность аппаратуры, если это свойство не закладывать при ее проектировании.

Предложенный подход для систем логического управления, не исключая возможности применения других методов построения программного обеспечения «без ошибок» [339], существенно более конструктивен, так как позволяет начинать «борьбу с ошибками» еще на стадии алгоритмизации.

Автор надеется, что после прочтения этой книги у читателя сложится впечатление о том, что существующее в настоящее время мнение, изложенное, например, в [306], о смерти теории автоматов является сильно преувеличенным, а предлагаемая в настоящей работе парадигма, базирующаяся на этой теории, сможет наконец позволить навести «порядок» в алгоритмизации и программировании задач логического управления ответственными промышленными объектами и вдохнуть новую жизнь в эту теорию.

Автор предполагает, что автоматное программирование может стать основным не только для задач логического управления, но и для существенно более широкого класса задач, решаемых на ПЭВМ.

Заканчивая эту весьма толстую книгу, автор понимает, что в ней не удалось решить все вопросы алгоритмизации и программирования задач логического управления, возникающие при автоматизации сложных и ответственных технологических объектов.

Было бы наивно даже предполагать, что можно написать книгу, которая позволит «закрыть» столь важную для практики проблему.

Более того, автор понимает, что, «высветив» ряд вопросов столь сложной проблемы, еще большее число вопросов, требующих своего решения, остались не рассмотренными, ведь недаром древние мудрецы сравнивали знания с кругом света от фонаря: «Чем шире круг, тем больше граница с темнотой».

Подходы, изложенные в этой книге, позволили автору и его коллегам решить ряд весьма важных практических задач логического управления ответственными технологическими объектами более «красиво», чем они это делали до сих пор.

Если книга поможет хоть частично сделать это и читателю, то автор будет считать, что цель написания книги достигнута.