Гумунюк П. В.
P. V. Gumunyk

СРАВНЕНИЕ ЗАТРАТ ВРЕМЕНИ НА РАЗРАБОТКУ ПРОГРАММНОГО ПРОДУКТА РУЧНЫМ СПОСОБОМ И СРЕДСТВАМИ АВТОМАТНОГО ПРОГРАММИРОВАНИЯ

COMPARISON OF TIME COSTS FOR PROGRAM PRODUCT DEVELOPMENT BY MANUAL AND AUTOMATIC PROGRAMMING MEANS

Павел Васильевич Гумунюк – аспирант Института конструкторско-технологической информатики Российской академии наук (Россия, Москва); тел. 8(916)599-48-10. E-mail: pavelmailgpv@gmail.com.
Pavel Vasilievich Gumunyk – Graduate Student, Institute of Design and Technological Informatics of the Russian Academy of Sciences (Moscow, Russia); tel. 8(916)599-48-10. E-mail: pavelmailgpv@gmail.com.

Аннотация. Данная статья имеет цель популяризировать методику автоматного программирования. Для показа преимуществ этой методики в статье проводится сравнение времени, необходимого на разработку карточной игры UNO двумя способами, а именно: ручным программированием и средствами автоматного программирования. Приведено обоснование, почему в качестве программы для разработки системы управления выбрана карточная игра UNO, и поясняются правила этой игры. Процесс разработки для наглядности и анализа разбит на этапы, каждый из которых измерен. Произведено сравнение этапов разработки программного обеспечения для выявления самого затратного из них. Это сравнение проводилось путём замера промежутков времени, необходимых для выполнения каждого конкретного этапа процесса разработки. В связи с возможностью возникновения во время написания программного кода различных обстоятельств и непреодолимых сил, способствующих увеличению или уменьшению времени разработки, расчёт не может быть полностью достоверным, т. к. для каждого конкретного случая существует только конкретное решение. Однако для систематизации статистики по сравнению временных промежутков, затраченных на разработку программы как ручным способом, так и с помощью методики автоматного программирования, в статье была проведена разработка аналогичных карточных игр с получением результатов по временным затратам. В завершение статьи приведён анализ полученных программных кодов и на основании полученных результатов делается заключение о возможности дальнейшего применения автоматного программирования.

Summary. This article aims to popularize the Automatic Programming technique. To show the advantages of this technique, the article compares the time required to develop the UNO card game in 2 ways, namely: Manual programming and Automatic programming tools. A justification is given for why the UNO card game was chosen as a program for developing a control system and the rules of this game are explained. The development process is divided into stages for clarity and analysis, each of which is measured. A comparison was made of the stages of software development to identify the most expensive of them. This comparison was made by measuring the length of time required to complete each specific stage of the development process. Due to the possibility that various circumstances and force majeure may arise during the writing of program code, contributing to an increase or decrease in development time, the calculation cannot be completely reliable, since for each specific case there is only a specific solution. However, in order to systematize statistics comparing the time periods spent on developing the program, both manually and using the Automated Programming technique, the article developed similar card games with obtaining results on time costs. At the end of the article, an analysis of the obtained program codes is given and, based on the results obtained, a conclusion is made about the possibility of further application of automatic programming.

Ключевые слова: UNO; ручная разработка; Switch-технология; автоматное программирование; система управления; Индустрия 4.0.

Key words: UNO; Manual development; Switch technology; Automatic programming; Control system; Industry 4.0.

УДК 004.457

Для того чтобы рассмотреть карточную игру UNO как систему, необходимо разобрать правила её ведения. Эта игра имеет следующие карты: карты от 0 до 9, смена цвета, смена хода, противоположному игроку +4 карты, противоположному игроку +2 карты, карты пропуска хода, карты обмена картами.

Общее количество карт 108, из которых 18 зелёных номиналом от 1 до 9, одна зелёная карта номиналом 0, и столько же синих, красных и жёлтых карт. Цвет карт в игре UNO является аналогом масти в классических карточных играх. Восемь карт, предназначенных для вручения противнику (противникам) двух дополнительных карт (карты берутся из общей колоды) соответственно по две зелёного, красного, синего и жёлтого цветов. Восемь карт для смены хода, по две штуки каждого цвета. Восемь карт для пропуска хода, по две штуки каждого цвета. Четыре карты, предназначенные для вручения противнику (противникам) четырёх дополнительных карт (карты берутся из общей колоды). Четыре карты смены цвета.

Игрок, который первым избавился от всех карт в раунде, получает очки за все карты, которые остались на руках у остальных игроков по следующим правилам:

1. За все карты номиналом от 0 до 9 начисляются баллы, равные номиналам карт (за карту номиналом 0 даётся 0 очков, за карту номиналом 1 даётся 1 очко и т. д.).
2. За карты, дающие противникам две дополнительные карты, даётся 20 очков.
3. За карты, дающие возможность пропуска хода, даётся 20 очков.
4. За карты, меняющие направление игры (очерёдность, по которой игроки делают свои ходы), даётся 20 очков.
5. За карты, позволяющие поменять цвет (активную игровую масть), даётся 50 очков.
6. За карты, дающие противникам четыре дополнительные карты, даётся 50 очков.

После подсчёта очков, если никто не набрал 500 очков, карты перетасовываются и игра продолжается. Победителем является тот, кто первым набрал 500 очков. Ещё один способ подсчёта очков – текущий подсчёт очков, которые остались на руках у игроков в конце каждого раунда. Тот игрок, у кого меньше всего очков по оставшимся на руках картам, становится победителем [11].

Карты перемешиваются и раздаются игрокам, по семь карт каждому игроку. Часть карт остаётся в колоде на тот случай, если игрокам понадобятся дополнительные карты [12].

Правила могут отличаться в зависимости от различных способов ведения игры, но можно выделить основные тезисы.

Данная программа будет разработана для игры между человеком и машиной (компьютерной программой, отыгрывающей роль противоположенного игрока).

Состояния системы игры зависят как от выбора игрока, так и от того, кто будет ходить первым.

Право первого хода предоставляется случайным образом.

Если первый ход за машиной, тогда она выбирает самую малую карту из имеющихся, т. к. задача машины выиграть, и ей необходимо в первую очередь избавиться от самых мелких карт, поскольку чем дальше идёт игра, тем труднее избавиться от карт с низким номиналом.

Карты можно бить только картами того же цвета, что и уже выложенные на игровой стол. В один момент времени игра возможна толька картами одного цвета. Если у машины нет карт нужного цвета или нужного номинала, но есть дополнительные карты для смены цвета, то машина может ими воспользоваться и выбрать новый активный цвет для дальнейшего хода игры. Цвет для замены компьютерной программой выбирается на основании подсчёта количества карт каждого цвета, и приоритет отдаётся картам того цвета, количество которых больше в колоде у машины. Если в колоде у машины оказывается равное количество карт разных цветов, то выбор производится случайным образом.

Если у противника машины, т. е. человека, осталось три карты или меньше, и если у машины есть карта на +4 / +2 карты противнику, то она может ей воспользоваться.

Если у машины нет карт требуемой масти и нет карт смены масти или карты, дающей игроку +4 / +2 карты, то она может воспользоваться картой смены хода, что освобождает её от выдачи карты.

Если у машины нет карт нужной масти или смены цвета, смены хода, противоположному игроку +4 / +2 карты, то ей необходимо самой взять из колоды 4 карты.

Если в колоде осталась только одна карта, тогда необходимо сообщить другим игрокам об этом, в противном случае нарушитель штрафуется двумя дополнительными картами из колоды.

На основании всего вышеперечисленного программа для игры в UNO должна в первую очередь состоять из двух блоков, где первый отвечает за рандомное тасование карт, выбор первого игрока, подсчёт очков для победы и определение победителя. Второй блок отвечает за то, чтобы обыграть игрока. В этом случае можно применить методы автоматного программирования для выделения состояний, в которых будет находиться игра, что позволит машине принимать решения для того, чтобы обыграть противника [3].

Первый автомат представляет собой работающий по триггерам счётчик, срабатывающий в начале и конце игры. В начале игры необходимо рандомно распределить карты и роли, а в конце – подсчитать баллы и определить победителя. Логические операции происходят во втором автомате, логика которого будет разобрана ниже.

Для увеличения количества игроков в игре достаточно продублировать второй блок. Каждое из условий, определённых правилами игры, будет соответствующим условием программы, а выбираться они будут оператором множественного выбора switch.

Важно не только время, необходимое на разработку, но и компетенция программиста. Согласно статистике, чтобы пройти путь от начинающего разработчика до опытного специалиста, требуется много лет упорной практики и развития своих навыков. Если учитывать этот фактор, то в условиях задачи поиска наименьшего времени на разработку программного обеспечения получается, что ручная разработка проигрывает автоматному программированию. Поэтому время, требуемое на обучение программиста, не будет учитываться в рамках данного эксперимента. Можно парировать этот аргумент тем, что разработка методов автоматного программирования является процессом не быстрым и равносильным обучению программиста до уровня профессионала. Однако методы и средства автоматного программирования делаются один раз и дают свои возможности по написанию кода всем специалистам, тогда как время, требуемое для обучения конкретного программиста, нужно умножить на количество всех специалистов. В данном случае автоматное программирование выигрывает у ручного в количественном соотношении по результатам общего затраченного времени. Сейчас в индустрии программирования квалифицированные программисты уже не справляются с поставленным объёмом задач, но при этом есть предпосылки для формирования технологии генерирования программного кода с помощью средств автоматного программирования и словаря понятий, направленных на обучение производственных специалистов азам разработки систем управления [6].

Для подсчёта времени разделим процесс разработки программного обеспечения на несколько ключевых этапов. Замеры каждого временного промежутка помогут в составлении статистики, которая выявит самый трудоёмкий этап и какой из методов лидирует.

В качестве снаряда для эксперимента выступит карточная игра UNO. Она была выбрана из-за большого количества состояний, возникающих в игре, и предсказуемости поведения искусственного интеллекта во время игровых ситуаций [10].

Выделим следующие этапы разработки:

1. Изучение правил игры.
2. Поиск ключевых паттернов поведения программы в игре (в случае с ручным программированием это разработка структуры программы, в случае с автоматным программированием – выделение устойчивых состояний).
3. Разработка программного кода (в случае с ручной разработкой это написание программного кода вручную с последующей его компиляцией и исправлением ошибок, в случае же автоматного программирования – загрузка в программу для генерации кода).
4. Тестирование программы, поиск ошибок и их последующее устранение [7].

Для выявления количества времени была проведена разработка двух идентичных программ с целью замера времени, требуемого на создание программ разными путями. Рассмотрим сначала процесс автоматической разработки, а затем ручной, после чего проведём сравнение временных результатов. Начнём с автоматического способа.

Состояний в системе столько же, сколько карт и действий в игре:

1. Выдача карт от 0 до 9 по правилам игры (в приоритете находятся карты с меньшим числом).

2. Противоположному игроку +4 карты.
3. Противоположному игроку +2 карты.
4. Смена хода.
5. Карты запрета хода.
6. Смена цвета.
7. Карты пропуска хода.
8. Взять дополнительно карты.
9. Действие игрока.

После действий игрока цикл перезапускается, т. е. после последнего состояния система переходит в первое, но и в последнее состояние также можно попасть из всех предыдущих, минуя промежуточные.

Переменные системы управления:

int lastCard = 0; номинал последней карты, при старте игры он равен 0;

int firstMove; право первого хода в партии, выбирается в начале игры рандомно и может принимать два значения, где 0 – право хода достаётся компьютеру, а 1 – право хода достаётся игроку;

int lastMove; флаг, записывающий, кому принадлежал последний ход. Это необходимо для понимания программой, должна ли она сейчас ходить или она передаёт это право игроку, где 0 – ход игрока, а 1 – ход компьютера;

Arr [] cardArray = int []; массив карт, имеющихся у компьютера. При каждом праве хода карты пересчитываются массивом, и при отсутствии необходимой карты программа пользуется дополнительными картами.

int cardGame; состояние игрового процесса.

Игра UNO хорошо подходит как объект для реализации элементов искусственного интеллекта. Поведение машины достаточно просто прописать, следуя несложным правилам игры, в виду того, что карты имеют свою цену и иерархию. На основании этого можно составлять последовательность действий, рассчитывая их цену и основываясь на приоритете игровых карт [2].

Данная программа не является системой управления в привычном виде из-за следующей её особенности. Если переход между состояниями происходит автоматически, то переход с последнего на первое осуществляется уже вручную, при помощи действий игрока. Такие программы также необходимы и имеют свои области применения [9].

```
Switch cardGame:
Case 1: // Выдача карт от 0 до 9
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length; i++)
{ cardArray[i] = 0; }
Break;
Case 2: // противоположенному игроку +4 карты
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length; i++)
{ cardArray[i] = 10; }
Break;
Case 3: // противоположенному игроку +2 карты
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length; i++)
{ cardArray[i] = 11; }
Break;
Case 4: // смена цвета
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length; i++)
{ cardArray[i] = 12; }
Break;
```

```
Case 5: // смена хода
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length; i++)
{ cardArray[i] = 13; }
Break;
Case 6: // запрет хода
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length; i++)
{ cardArray[i] = 14; }
Break;
Case 7: // Взять дополнительно +4 карты
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length+3; i++)
{ cardArray[i] = 0; }
lastMove = 0 ;
Break;
Case 8: // действие игрока
lastMove = 1 ;
Break;
```

Для реализации кода методом ручного построения нужно ввести дополнительную переменную:

int CardForMove определяет значение требуемой карты, которое выбирается из готовых вариантов одним из методов, срабатывающим при действиях игрока. Переменная может принимать значение от 0 до 9 при срабатывании метода.

int playerMoveResult переменная, представляющая собой значение карты.

Для корректной реализации программы воспользуемся реализацией следующих методов:

CardSearch выдача карт от 0 до 9;

PlayerAddingCards противоположному игроку +4 карты и противоположному игроку +2 карты;

ChangeMove смена хода;

ChangeCards карты обмена картами;

ChangeColor смена цвета;

LeaveFromMove карты пропуска хода.

В коде программы необходимо реализовать все перечисленные методы. Они все берут переменную lastCard playerMoveResult, но при этом результат выдаёт только один из методов.

Далее представлен фрагмент ручной программы:

```
void CardSearch () // Выдача карт от 0 до 9
{
for (int i = 0; i <= cardArray.Length+3; i++)
{ cardArray[i] = 0 ; }
}
void ChangeMove () // Смена хода
{
for (int i = 0; i <= cardArray.Length+3; i++)
{ cardArray[i] = lastCard ; }
}
void ChangeCards () // Запрет хода
{
for (int i = 0; i <= cardArray.Length+3; i++)
{ cardArray[i] = lastCard ; }
}
```

```
void ChangeColor () // Смена цвета
{
for (int i = 0; i <= cardArray.Length+3; i++)
{ cardArray[i] = lastCard ; }
}
void LeaveFromMove () // Пропуск хода
{
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length+3; i++)
{ cardArray[i] = lastCard ; }
}
Void Main ()
{
CardSearch () ;
ChangeMove () ;
ChangeCards () ;
ChangeColor () ;
LeaveFromMove () ; // Поиск нужного метода
Arr [] cardArray = int [] ;
for (int i = 0; i <= cardArray.Length+3; i++)
{ cardArray[i] = 0 ; }
}
```

Оба программных кода написаны на языке C# и находятся в репозитории https://github.com/pavel-proger/CardGame. Внешний вид программы для игры в UNO представлен на рис. 1.

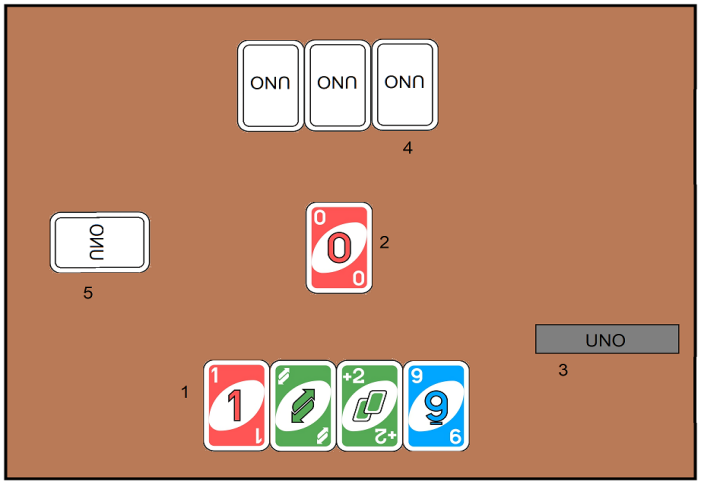

Рис. 1. Схематическое изображение программы для игры в UNO: 1 – карты игрока; 2 – текущая карта; 3 – кнопка для оповещения об окончании игры (становится активной только тогда, когда у игрока осталась одна карта); 4 – карты противника; 5 – стопка дополнительных карт

Рассмотрим фрагменты обоих программных кодов.

Оба программных кода выполняют свою функцию, но при этом они имеют разную структуру. Код, сгенерированный при помощи инструментов и методик автоматного программирования, имеет линейную, последовательную структуру набора условий, входящих в оператор switch, что является логичным и оправданным из-за второго названия данной методики – Switch-технология. Данная структура состоит из простых логических единиц, главным преимуществом которых является возможность без труда отследить связи между ними. По сути,

если взять условие как базовый логический элемент структуры Switch-технологии и рассмотреть его составные части в программах, в которых он применяется, можно систематизировать информацию и выделить общие правила его построения.

В итоге получается, что в каждом условии есть переменные, результаты которых заставляют программу перейти на это условие, переменные, по результатам которых программа переходит в другие условия, а также действия, выполняемые в самом условии, и реализуемые в нём же флаги [4]. Поверх этих базовых составляющих могут крепиться дополнительные конструкции, однако структура кода, сгенерированного при помощи технологии автоматного программирования, при этом не изменится. Таким образом, любая программа, которая была написана с применением Switch-технологии, всегда будет отличаться своей простотой и унификацией, что значительно помогает при её чтении. Эта простота может помочь при разработке аналогичных программ, применяемых для различных нужд, благодаря сведению процесса программирования к изучению технологического процесса и интерпретации его через состояния графов переходов автоматного программирования [1].

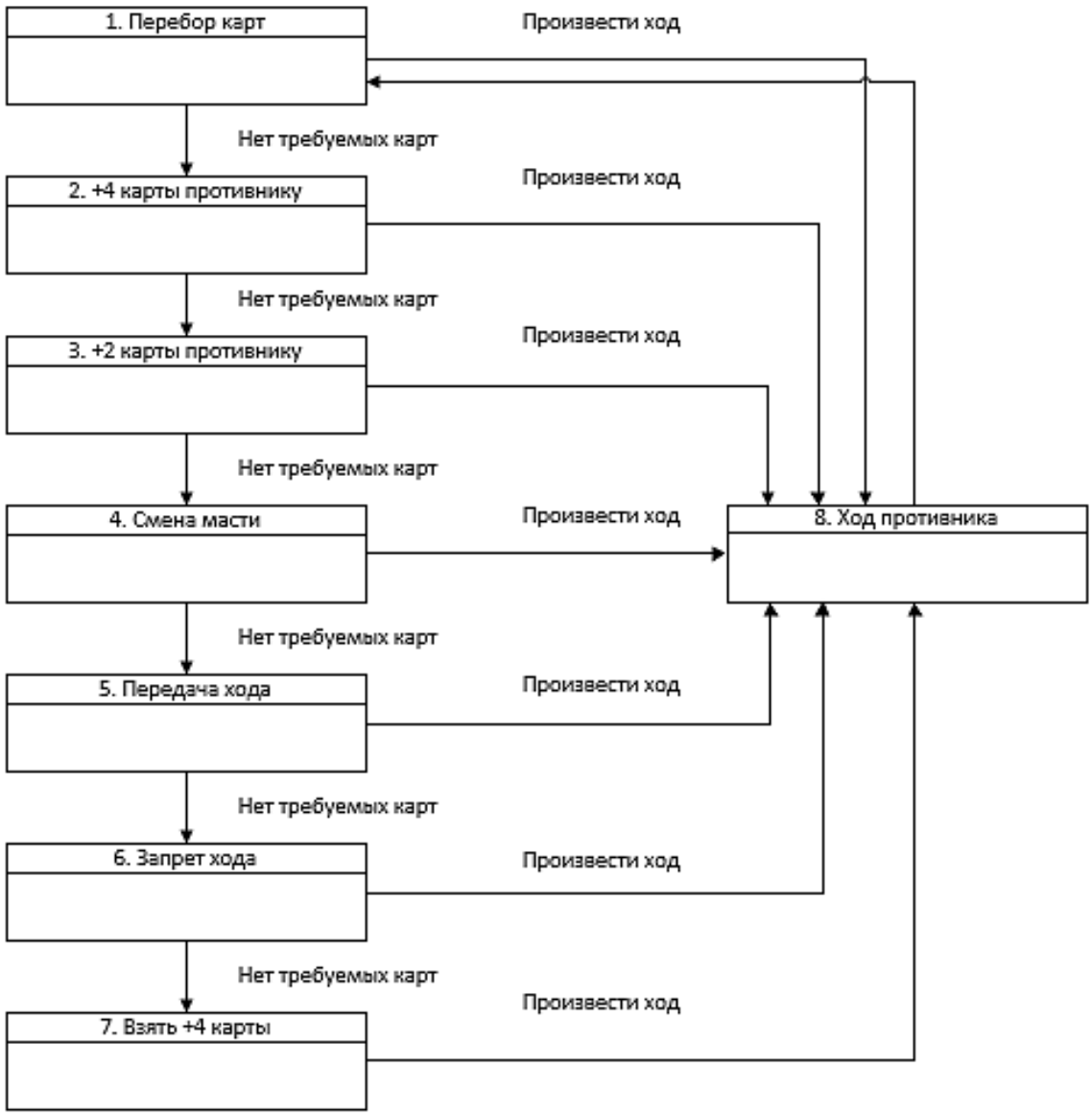

Рис. 2. Представление программного кода автоматного подхода в виде алгоритма (блок-схемы)

Программа, написанная вручную, имеет куда более сложное строение. Для её корректной работы потребовалось написать несколько методов. Каждый метод соответствует определённой операции, а именно: выдача карт от 0 до 9; противоположному игроку +4 карты; противоположному игроку +2 карты; смена хода; карты пропуска хода; смена цвета. В данном программном коде методы выступают в качестве аналога состояний из кода, сгенерированного технологией автоматного программирования. Исключение составляют только действия: «взять дополнительно карты» и «ход игрока», которые реализованы в виде простейших условий [8].

Программный код в виде блок-схемы, разработанной с помощью автоматного программирования, представлен на рис. 2.

Программный код в виде блок-схемы, разработанной с помощью ручного программирования, представлен на рис. 3.

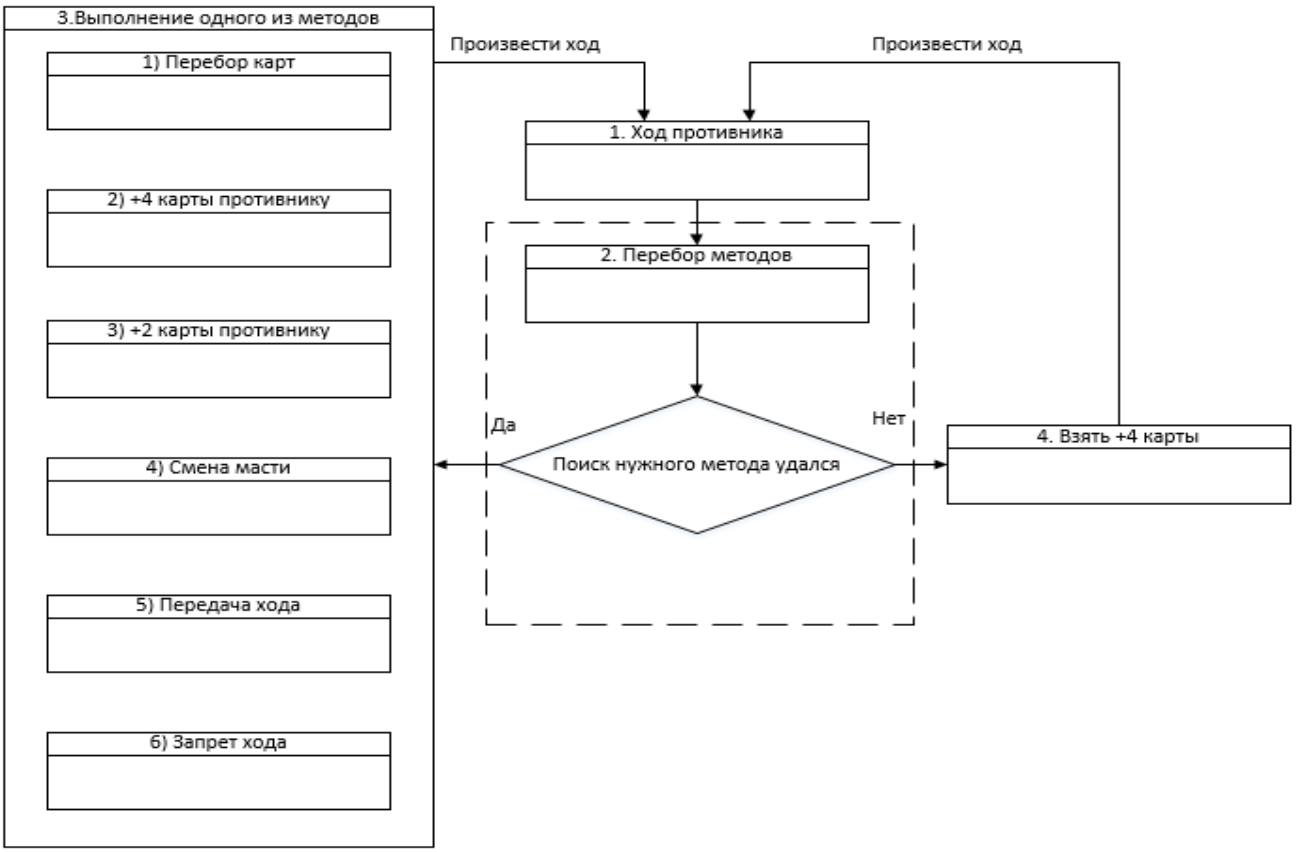

Рис. 3. Представление программного кода ручного подхода в виде алгоритма (блок-схемы)

Для достоверности полученного временного результата необходимо было убедиться в том, что данные не являются ошибочными. Для этого необходимо было провести аналогичные испытания на множестве примеров и только потом систематизировать результат. Игра UNO является карточной игрой, поэтому в качестве примера были взяты другие карточные игры и проанализированы их результаты для сравнения. Результаты по играм представлены в табл. 1.

Разбиение процесса автоматической разработки игры UNO по этапам представлено на рис. 4.

Разбиение процесса ручной разработки игры UNO по этапам представлено на рис. 5.

Сравнение времени, требуемого на разработку каждой программы, представлено на рис. 6.

Таблица 1

Сравнение временных промежутков, затрачиваемых на разработку карточных игр ручным и автоматическим способом

	Время (часы)	Игра				
Автоматическое генерирование	Этапы	UNO	Солитер	Пасьянс	Косынка	Покер
	1. Изучение	2	2	2	2	2
	2. Паттерны	1,5	1,5	1,5	1	2
	3. Разработка	0,25	0,25	0,25	0,25	0,25
	4. Тестирование	1	1,5	1	1	1,5
	Итого:	4,75	5,25	4,75	4,25	5,75
Ручная разработка	Время (часы)	Игра				
	Этап	UNO	Солитер	Пасьянс	Косынка	Покер
	1. Изучение	2	2	2	2	2
	2. Паттерны	1	1,5	1	1,5	2
	3. Разработка	6	6,5	6	7	8
	4. Тестирование	2	2,5	2	2	2,5
	Итого:	11	12,5	11	12,5	14,5

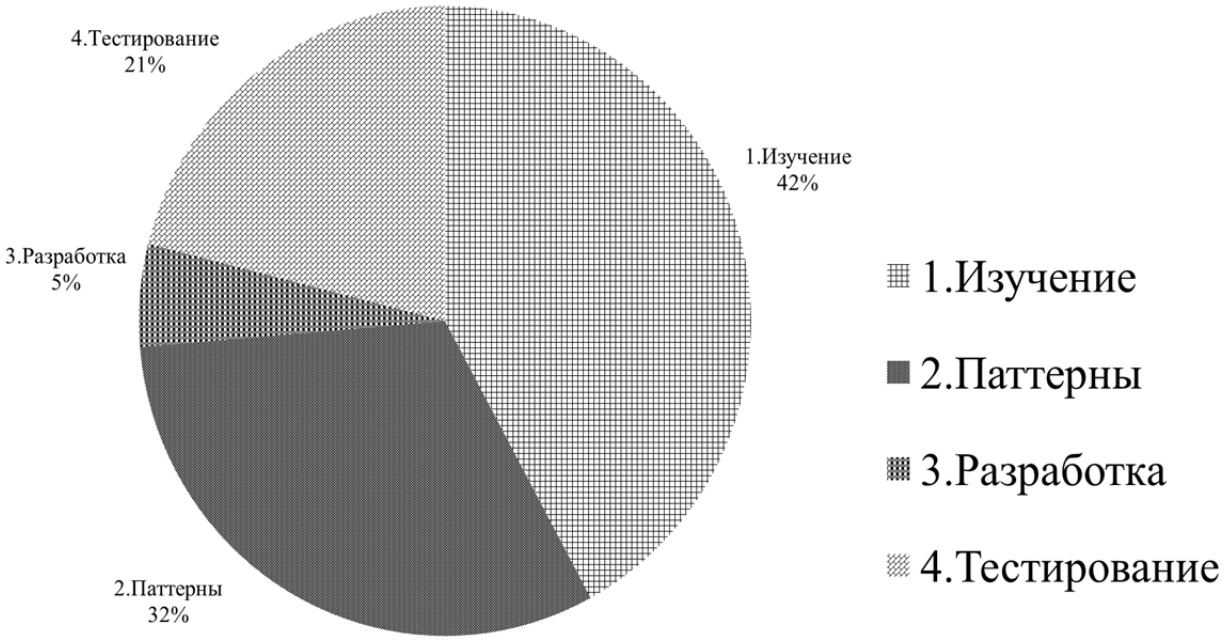

Рис. 4. Временное разбиение в часах по этапам процесса автоматической разработки

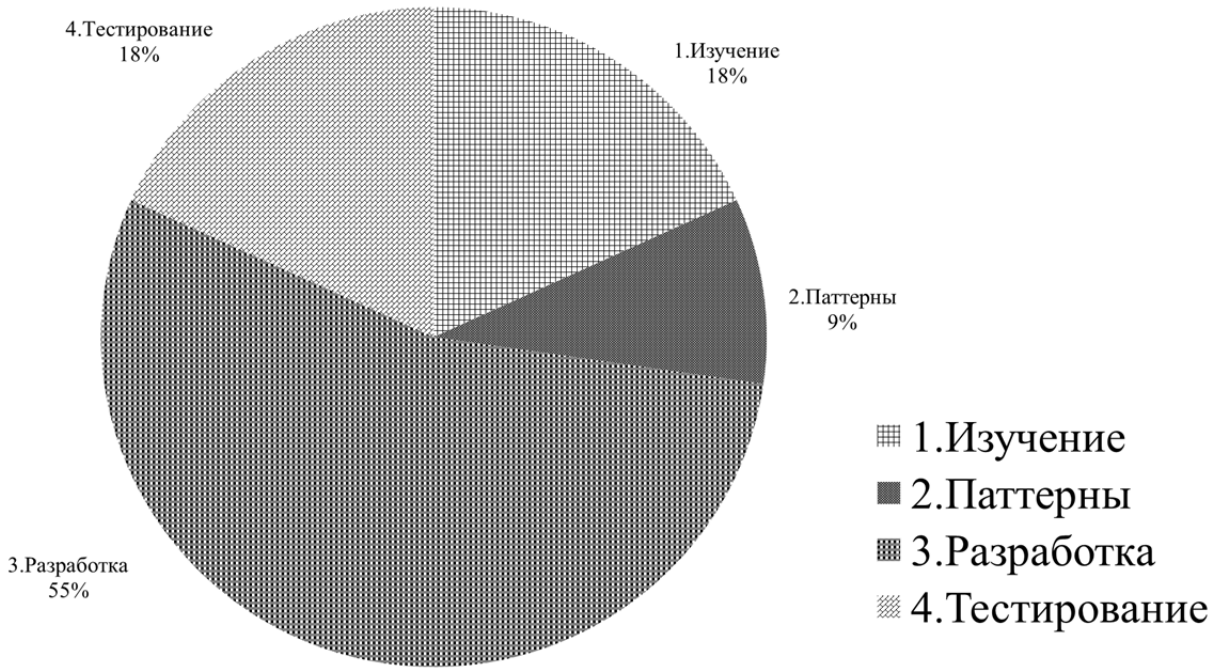

Рис. 5. Временное разбиение в часах по этапам процесса ручной разработки

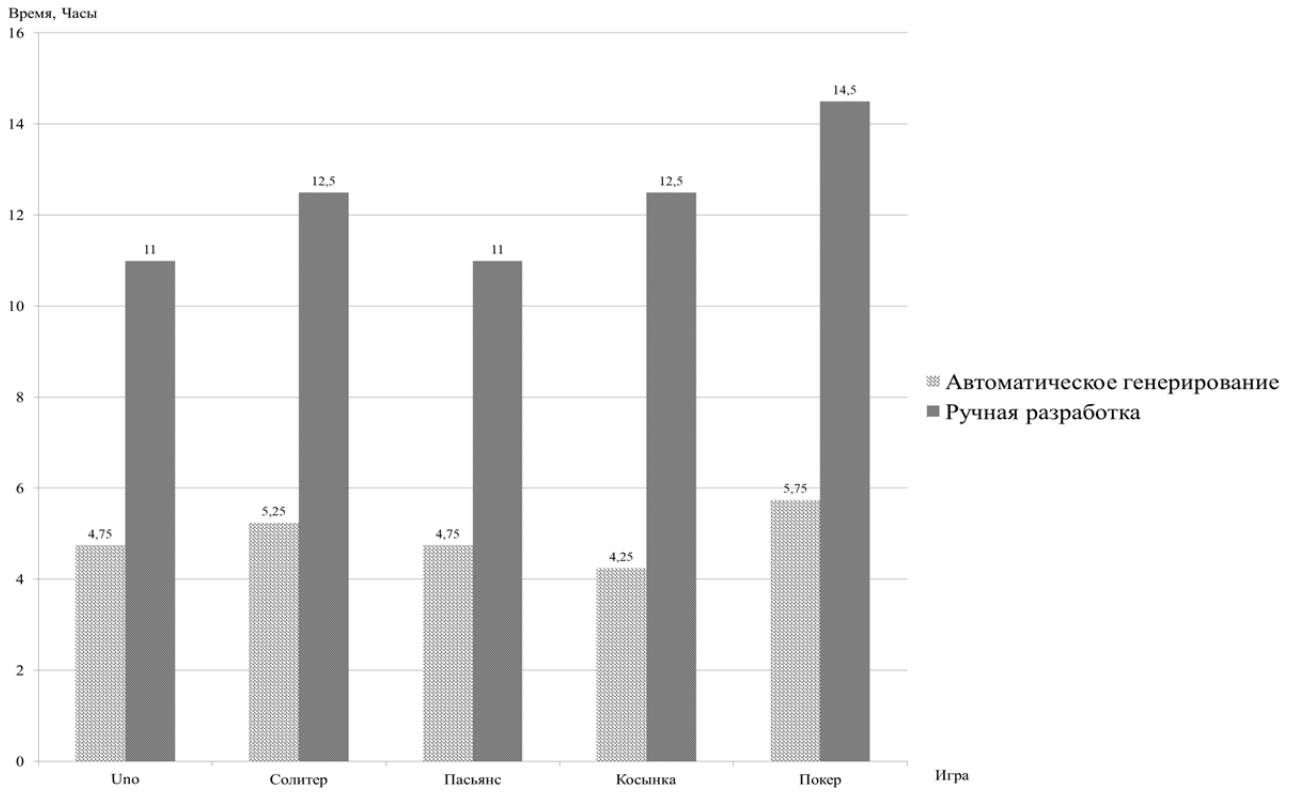

Рис. 6. Сравнение времени, требуемого на разработку каждой программы по часам

Обе программы вполне функциональны и безупречно справляются со своими функциями, несмотря на их различное написание и строение. Однако код, сгенерированный при помощи технологии автоматного программирования, значительно уменьшает время на его разработку (примерно с 6-8 часов до 15 минут), а общий результат уменьшает время разработки практически в два раза. Это свидетельствует о том, что автоматное программирование является эффективным средством для решения задач по написанию программных продуктов для тех систем, чей технологический процесс понятен и хорошо изучен [5].

ЛИТЕРАТУРА

1. Афанасьева, Ю. И. Организация деятельности сельскохозяйственных производителей в условиях информационно-технологической среды / Ю. И. Афанасьева, А. В. Рыбаков, А. Н. Шурпо // Учёные записки Комсомольского-на-Амуре государственного технического университета. Науки о природе и технике. – 2020. – № III-1 (43). – С. 4-8.
2. Афанасьева, Ю. И. О возможности производства пищевых продуктов с использованием «умных систем» / Ю. И. Афанасьева, А. В. Рыбаков, А. Н. Шурпо // Учёные записки Комсомольского-на-Амуре государственного технического университета. Науки о природе и технике. – 2020. – № V-1 (45). – С. 87-93.
3. Канжелев, С. Ю. Преобразование графов переходов, представленных в формате MSVisio, в исходные коды программ для различных языков программирования (инструментальное средство MetaAuto) / С. Ю. Канжелев, А. А. Шалыто; Национальный исследовательский университет институт точной механики и оптики. – СПб.: ИТМО, 2005. – 102 с.
4. Липкин, Е. ИНДУСТРИЯ 4.0: умные технологии – ключевой элемент в промышленной конкуренции / Е. Липкин. – М.: ООО «Остек–СМТ», 2017. – 224 с.
5. Поликарпова, Н. И. Автоматное программирование: учебно-методическое пособие / Н. И. Поликарпова, А. А. Шалыто. – СПб.: СПбГУ ИТМО, 2007. – 108 с.
6. Шалыто, А. А. Методы аппаратной и программной реализации алгоритмов / А. А. Шалыто; Национальный исследовательский университет институт точной механики и оптики. – СПб.: ИТМО, 2005. – 779 с.
7. Татарчевский, В. А. Switch-технология в задачах логического управления / В. А. Татарчевский // Программные продукты и системы. – 2006. – № 5. – С. 30-32.
8. Шалыто, А. А. Алгоритмизация и программирование задач логического управления / А. А. Шалыто. – СПб.: СПбГУ ИТМО, 1998. – 55 с.
9. Шалыто, А. А. Switch-технология – автоматный подход к созданию программного обеспечения «реактивных» систем / А. А. Шалыто, Н. И. Туккель // Программирование. – 2001. – № 5. – С. 45-62.
10. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition, Wiley, 2015.
11. Правила игры UNO // Игронафтик.ру. – URL: https://igronaftik.ru/files/cms/common/pravila_igry_uno.pdf (дата обращения: 17.03.2024). – Текст: электронный.
12. UNO Настольная игра [Электронный ресурс]. – URL: https://static.onlinetrade.ru/docs/470367/nastolnaya_igra_mattel_uno_1581150012_1.pdf (дата обращения: 17.03.2024). – Текст: электронный.