Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Факультет «Информационных технологий и программирования»
Кафедра “Компьютерные технологии”

Р. В. Наумов, А. В. Якушев, А. А. Шалыто

«Устройство» для карточной игры *Блэкджек*

Проектная документация

Проект создан в рамках «Движения за открытую проектную документацию»
http://is.ifmo.ru

Санкт-Петербург
2007

Оглавление

Введение

Цель данной работы – изучение автоматного подхода к программированию с использованием инструментального средства *Unimod.* В качестве примера рассматривается программная реализация карточной игры *Блэкджек* на основе автоматного подхода. Для данной системы требовалось обеспечить простоту понимания работы ядра, в котором реализована логика игры, и его взаимодействия с пользовательским интерфейсом. Так же требовалось разделить систему на максимальное число слабо зависимых друг от друга объектов. Это обеспечило бы хорошую "расширяемость" системы. Автоматный подход позволил изящно выполнить данные требования. Подробнее об автоматном подходе можно прочитать на сайте http://is.ifmo.ru.

Использование среды разработки *Unimod* позволяет с самого начала разработки программы использовать автоматный подход. Этот подход заключается в построении системы взаимосвязанных автоматов для формализации поведения системы. Благодаря использованию автоматов, программа разделяется на отдельные независимые блоки. Поэтому облегчается написание программы и снижается риск возникновения ошибок. Сайт инструментальной среды *UniMod* – http://unimod.sf.net.

1. Описание проекта

1.1. Правила игры

Цель игры *Блэкджек* заключается в том, чтобы собрать комбинацию карт с суммой очков, равной или близкой, но не превышающей 21. Существует много различных вариантов этой игры. В данной работе рассматриваются правила взятые, с сайта интернет-казино http://www.partycasino.com. В игре используется термин *Ценность руки* – это сумма номиналов, всех карт выданных игроку. Игрок – человек, который "использует" наше устройство, играя на нем в *Блэкджек.* Установлены следующие достоинства карт:

- Карты, начиная с двойки и заканчивая 10, ценятся по номиналу.
- Достоинство карт Валет, Дама и Король равняется 10 очкам.
- Туз равен 11 очкам или одному очку. Туз равен 11, если комбинация не превышает 21 очко, а в случае перебора Туз равен единице.

Игрок играет против "дилера казино" – нашего «устройства». Игрок выигрывает в том случае, когда он набирает больше очков, чем дилер и его сумма очков не превышает 21. Если игрок и казино набирают одинаковое количество очков, то игрок проигрывает. Если количество очков игрока превышает 21, то игрок проигрывает.

После того, как игрок решает больше не брать карт, ход переходит к дилеру. Дилер набирает себе карты до тех пор, пока ценность его руки не превысит ценность руки игрока. Если ценность его руки превысила 21, то он проигрывает.

1.2. Интерфейс игры

Перечислим элементы интерфейса.

- При запуске программы появляется окно, в котором пользователь может управлять своим начальным банком (рис. 1). Для того, чтобы покинуть игру, достаточно нажать на кнопку '*Exit*' (Выход).

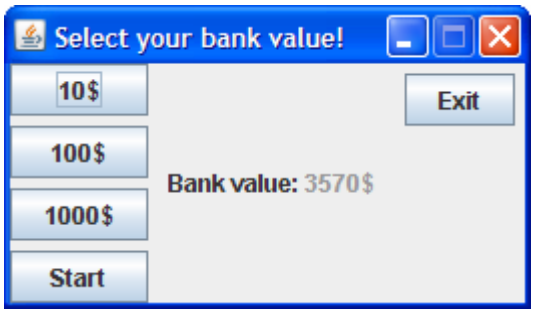

Рис. 1. Окно управлением начальным банком

- При нажатии на кнопки *10$*, *100$*, *100$* банк игрока увеличивается на соответствующую сумму.
- Если нажать кнопку *Start* (*Начать*), то появится окно, представляющее игровой стол (рис. 2).

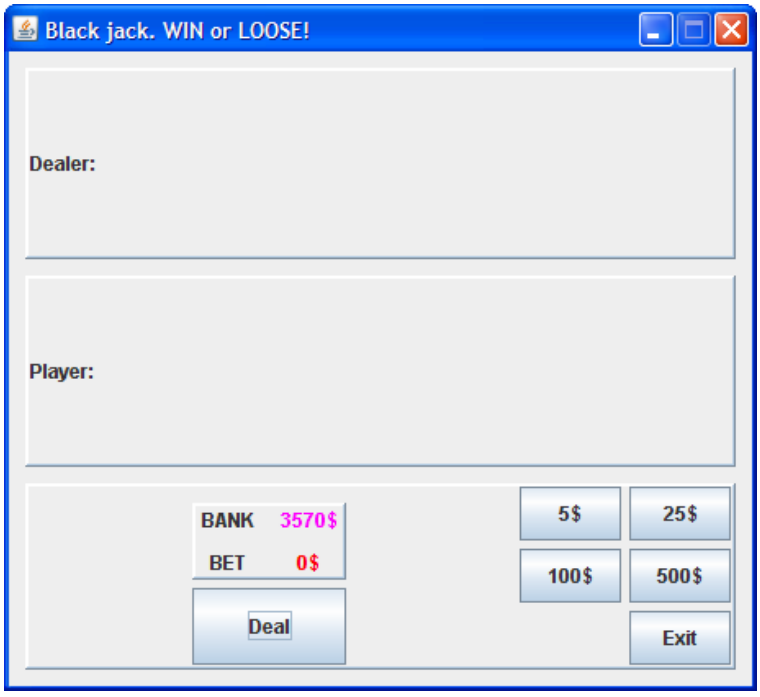

Рис. 2. Окно, представляющее игровой стол

- В правом углу стола находятся кнопки управления своей ставкой. Если игроку надоело играть или он проиграл все свои деньги, то он может нажать кнопку *Exit* (*Выход*) и покинуть игровой стол.
- Игрок может выбрать сумму, которую хочет поставить, нажав на кнопку с изображением необходимого ему номинала.
- После того, как ставка сделана, игрок нажимает кнопку *Deal* (*Раздать*). Для того, чтобы дилер раздал карты, ставка должна быть отлична от нуля.

- Дилер сдает вам две карты лицом вверх и затем сдает две карты себе: одну лицом верх, а другую – лицом вниз (рис. 3).

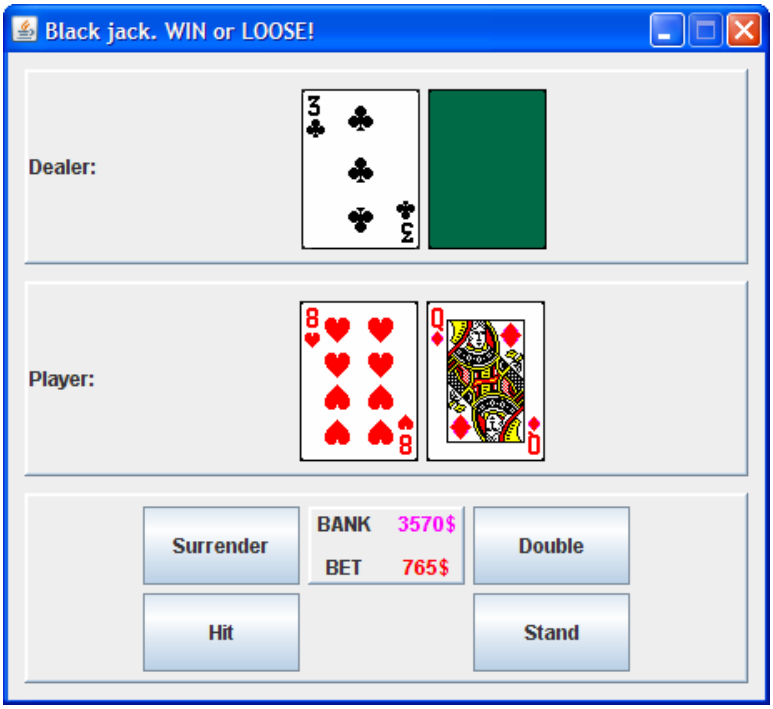

Рис. 3. Игровой стол после раздачи карт

- После раздачи первых двух карт и пока ценность руки не станет больше или равна 21 очкам, игрок может совершить четыре действия: *Surrender* (*Сдаться*), *Stand* (*Хватит*), *Hit* (*Взять*) или *Double* (*Удвоить*):
    - *Surrender.* Игрок может остановить партию, нажав на кнопку *Surrender* (*Сдаться*). Игра останавливается, и игроку возвращается половина его ставки.
    - *Stand.* Нажав на кнопку *Stand* (*Хватит*), игрок заканчивает свои действия в этой сдаче, и право хода передается дилеру.
    - *Hit.* Нажав на кнопку *Hit* (*Взять*), игрок получает еще одну карту. Игрок может продолжать брать карты до тех пор, пока не решит остановиться или пока ценность его руки не превысит 21.
    - *Double.* Нажав на кнопку *Double* (*Удвоить*), игрок удваивает сумму ставки, и ему выдается еще одна карта. После того, как сдается следующая карта, ход автоматически передается дилеру. Однако если баланс игрока окажется недостаточным для дополнительной ставки, игроку не будет доступна опция удвоения. Если количество очков после взятия карты превысит 21, то игрок сразу проигрывает.
- Когда право хода передается дилеру, он набирает себе карты до тех пор, пока ценность его руки не превысит ценность руки игрока. После этого определяется победитель в игре. Если игрок победил, то ему в банк добавляют сумму, равную его ставке, а если проиграл, то эта сумма снимается из банка.
- После сообщения о проигрыше или выигрыше, игроку предлагается сыграть еще раз. Для этого ему требуется снова сделать ставку. Если он хочет оставить ту же сумму, что была и в предшествующей сдаче, то он может нажать на кнопку *Rebet* (*Повторить ставку*). Если игрок хочет изменить ставку, то он должен нажать на кнопку *Clear* (*Очистить*).

2. Реализация с использованием автоматов

2.1. Схема связей автоматов

Устройство для игры в *Блэкджек* состоит из двух автоматов *A1* и *A2*. На схеме связей автоматов (рис. 4) каждому событию соответствует обозначение вида *E#* , где вместо # находится число. Входные переменные объектов управления обозначаются *x1*, *x2*, … Выходные воздействия объектов управления – *z0*, *z1*, ... Все они являются методами и реализуются вручную на языке *Java*.

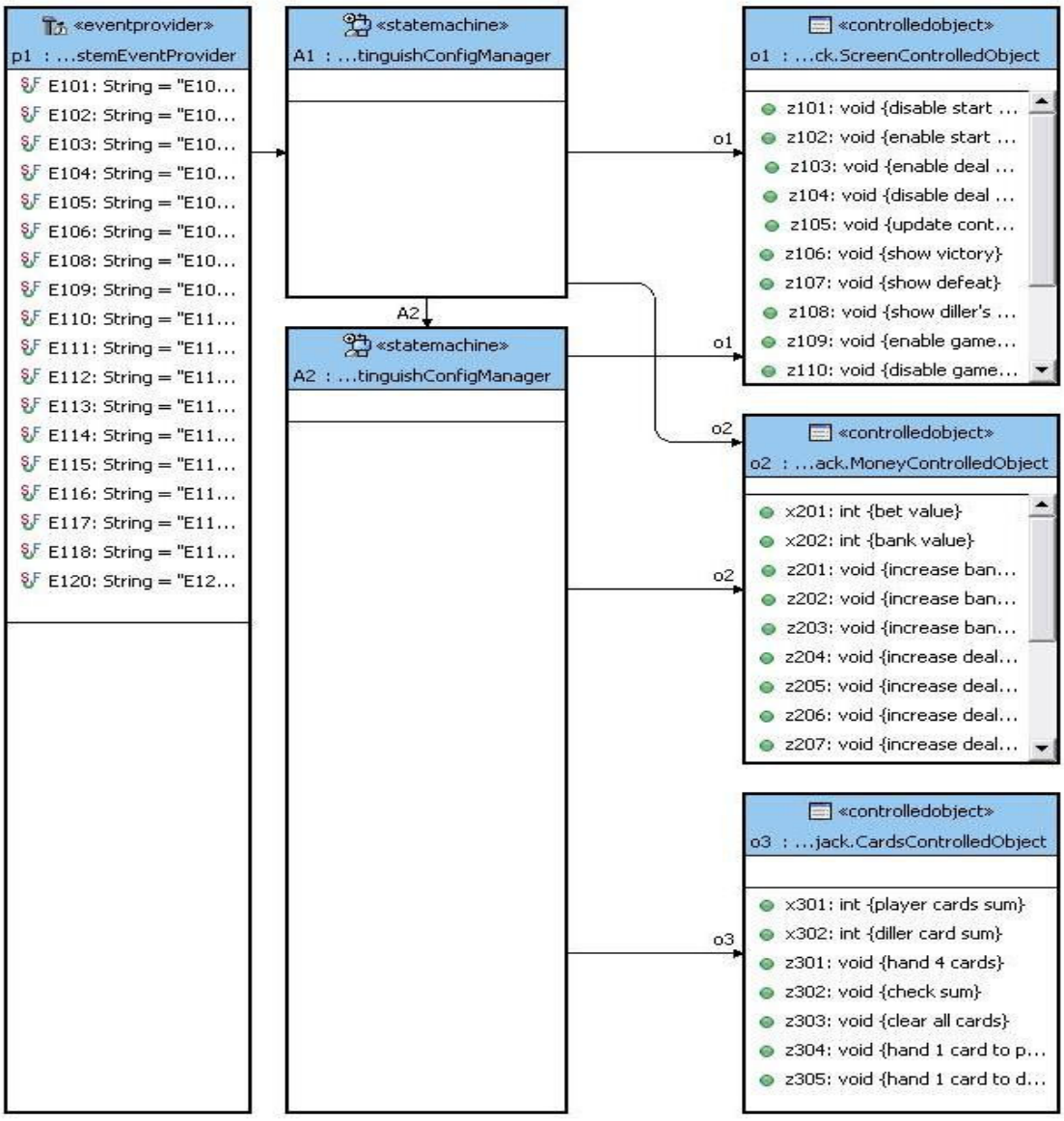

Рис. 4. Схема связей автоматов

На схеме связей автоматов каждая составляющая системы изображена в виде прямоугольника, в верхней части которого приведено краткое обозначение и название *Java*-класса, реализующего функциональность данного объекта. При этом для автоматов название *Java*-класса не указано, так как инструментальное средство *UniMod* автоматически генерирует их *XML*-описание.

Рассмотрим элементы схемы связей. В целях логического разделения функциональных возможностей системы создано два автомата:

- *A1* – инициализирующий автомат задает начальные данные (количество денег) для автомата *A2 (*игра *Блэкджек*) и запускает игру;
- *A2* – автомат, реализующий логику игры *Блэкджек.*

Приведем описание классов, используемых на схеме связей (табл. 1).

Таблица 1. Описание классов

Название класса	**Связь с *Unimod***	**Описание класса**
`SystemEventProvider.java`	Поставщик событий *p1*	Описывает все события, которые генерирует устройство
`ScreenControlledObject.java`	Объект управления *o1*	Отвечает за интерфейс устройства
`ScreenControlledObject.java`	Объект управления *o2*	Отвечает за все “денежные” операции в устройстве
`ScreenControlledObject.java`	Объект управления *o3*	Отвечает за все операции с картами (основная логика игры *Блэкджек*)

2.2. Источники событий

События, которые генерируются в системе, описаны в табл. 2

Таблица 2. Источник событий *p1*

Событие	*Описание*
`E101`	Добавить в банк 10$
`E102`	Добавить в банк 100$
`E103`	Добавить в банк 1000$
`E104`	Начать игру
`E105`	Закончить игру
`E106`	Раздать карты
`E108`	Увеличить ставку на 5$
`E109`	Увеличить ставку на 25$
`E110`	Увеличить ставку на 100$
`E111`	Увеличить ставку на 500$
`E112`	Сдаться
`E113`	Сбросить ставку
`E114`	Повторить туже ставку, что и в последней игре
`E115`	Взять еще одну карту
`E116`	Передать ход дилеру

E117	Удвоить ставку
E118	Забрать деньги и уйти из игры
E120	Проверить суммы карт

2.3. Объекты управления

2.3.1. Объект управления графическим интерфейсом

Входные и выходные воздействия объекта управления графическим интерфейсом *o1* представлены в табл. 3.

Таблица 3. Объект управления *o1*

Метод	*Описание*
x101	Убрать окно выбора начального банка
x102	Показать окно выбора начального банка
z103	Показать кнопки выбора ставки
z104	Убрать кнопки выбора ставки
z105	Перерисовать окно
z106	Показать победу игрока
z107	Показать проигрыш игрока
z108	Показать карты дилера
z109	Показать кнопки выбора действий игрока
z110	Убрать кнопки выбора действий игрока
z113	Показать кнопки выбора продолжения игры
z114	Убрать кнопки выбора продолжения игры
z115	Закончить игру
z116	Спрятать карты дилера

2.3.2. Объект управления деньгами

Входные и выходные воздействия объекта управления деньгами *o2* представлены в табл. 4.

Таблица 4. Объект управления *o2*

Метод	*Описание*
x201	Величина ставки игрока
x202	Величина банк игрока
z201	Увеличить банк на 10$
z202	Увеличить банк на 100$
z203	Увеличить банк на 1000$
z204	Увеличить ставку на 1$
z205	Увеличить ставку на 5$
z206	Увеличить ставку на 25$
z207	Увеличить ставку на 100$
z208	Увеличить ставку на 500$
z209	Вернуть игроку половину его ставки

z210	Сделать туже ставку, что и в последней игре
z211	Обнулить ставку
z212	Увеличить банк на величину ставки
z213	Уменьшить банк на величину ставки
z214	Удвоить ставку

2.3.3. Объект управления картами

Входные и выходные воздействия объекта управления картами *o3* представлены в табл. 5.

Таблица 5. Объект управления *o3*

Метод	*Описание*
x301	Ценность руки игрока
x302	Ценность руки дилера
z301	Раздать по две карты игроки и дилеру
z302	Проверить суммы карт
z303	Вернуть все карты в колоду
z304	Сдать игроку одну карту
z305	Сдать дилеру одну карту

2.4. Автоматы

Начальное состояние автомата обозначено черным кругом, а конечное — двойным кругом с закрашенной серединой. Каждый автомат имеет одно начальное и произвольное число конечных состояний. Состояния обозначены прямоугольниками со скругленными краями, переходы — линиями со стрелками, которые помечены следующим образом:

E# [логическое выражение]/список выходных воздействий.

Логическое выражение может состоять из:
- входных переменных в нотации *o#.x#* ;
- логических операций &&, ||, !;
- операций сравнения;
- скобок.

Допустим, что автомат находится в каком-либо состоянии. При появлении события для всех соответствующих ему переходов проверяется логическое выражение. Если найден переход, для которого оно истинно, то сначала выполняются все выходные воздействия, указанные на переходе после символа *«/»*, а затем — выходные воздействия, предусмотренные для исполнения в момент входа в состояние, которые указываются после синтаксической конструкции *«enter/»*.

Лексема *«include/A#»* соответствует тому, что в вершину вложен автомат. При этом после анализа логического выражения и выбора перехода в новое состояние реализуются:
1. Выходные воздействия, ассоциированные с переходом автомата в новое состояние.
2. Выходные воздействия, предусмотренные для исполнения в момент входа в состояние.
3. Передача управления вложенному автомату *A#*.

При проектировании графов переходов с помощью инструментальной средства *Unimod* выполняется автоматическая проверка условий переходов на полноту и непротиворечивость. Если у какого-то состояния система логических условий, соответствующая всем переходам из него, будет не полна или противоречива, то инструментальное средства *Unimod* автоматически подсветит все переходы по соответствующему событию, сигнализируя об ошибке (рис. 5). При этом на этом приведен лишь фрагмент автомата *A1*. Поэтому верхний ряд вершин в графе имеет недорисованные дуги. Подробнее проверка системы логических условий инструментальным средством *Unimod* описана в работе [1].

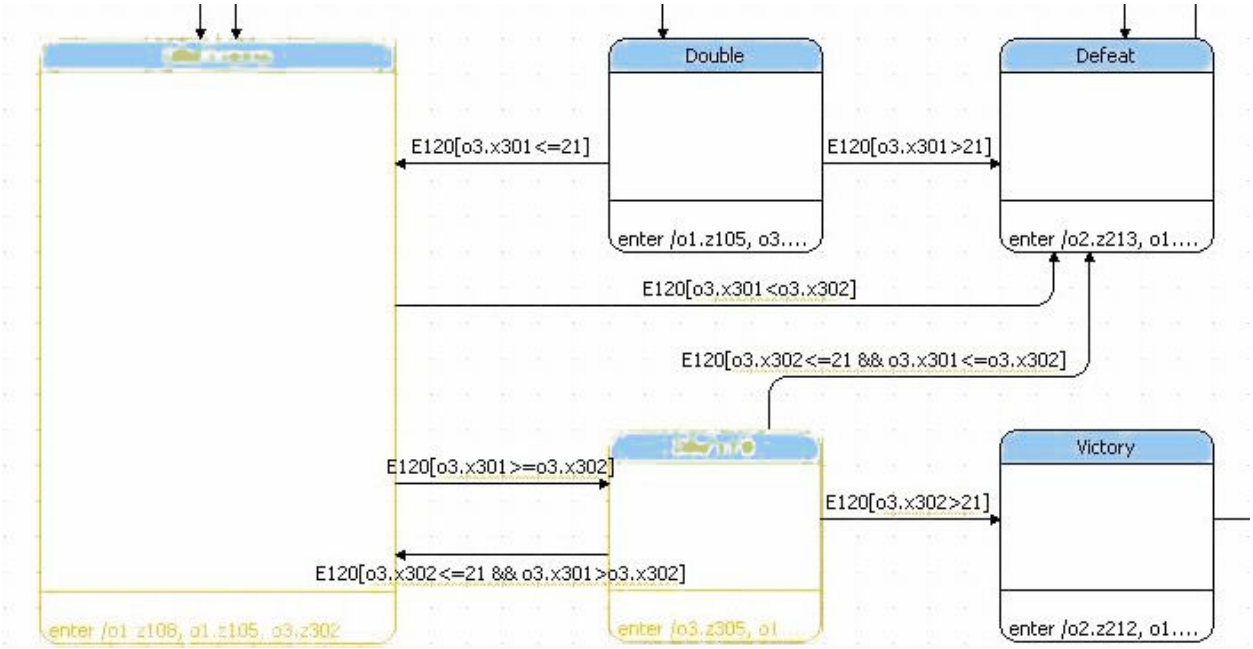

Рис. 5. Подсетка неполноты системы логических условий

Рассмотрим граф переходов автомата *A1* (рис. 6).

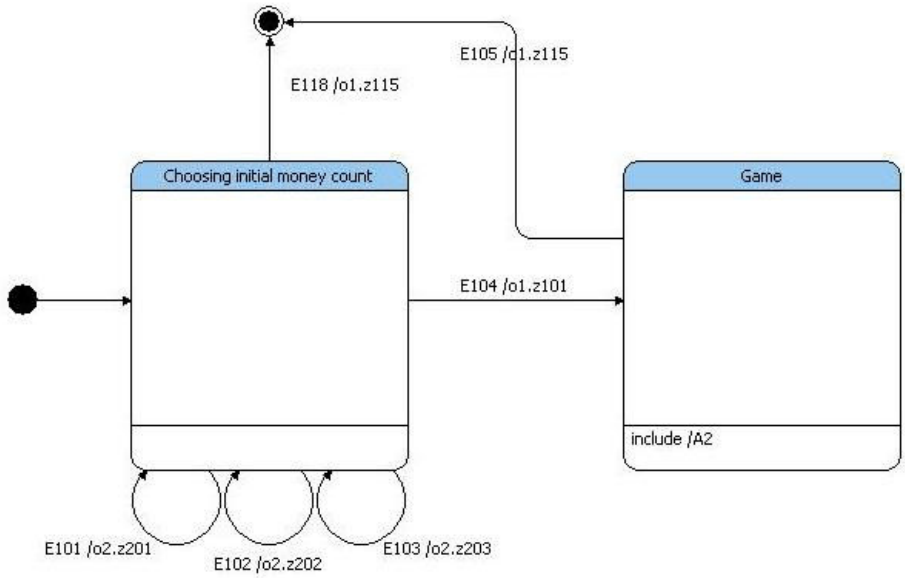

Рис. 6. Граф перехода автомата *A1*

Автомат *A1* может находиться в двух состояниях – инициализация или подготовка к игре, и собственно сама игра *Блэкждек*. В первом состоянии игрок задает начальное количество денег в

игре и может запустить игру или покинуть ее. Во второе состояние вложен автомат *A2*, отвечающий за логику игры. Данный автомат имеет девять состояний (табл. 6) и изображен ниже (рис. 7)

Таблица 6. Состояния автомата *A2*

Состояние автомата	*Описание состояния*
`Making a bet`	Игрок делает ставку – решает “на сколько денег” он будет играть
`Player's move`	Ход игрока, ставка сделана, карты дилеру и игроку розданы. Теперь игроку предстоит решать, что делать дальше. Он может забрать половину ставки, взять еще одну карту, остановится или удвоить ставку
`Double`	Игрок удвоил ставку и получил еще одну карту
`Player's hit`	Игрок взял еще одну карту
`Dealer's move`	Ход дилера. Игрок сделал ставку и взял необходимое число карт. Теперь дилер проверяет сумму своих карт и карт игрока и решает делать ему ход, объявить о поражении или победе игрока
`Dealer's hit`	Дилер решил сделать ход и взял еще одну карту
`Defeat`	Игрок проиграл игру
`Victory`	Игрок выиграл игру
`Choosing`	Конец игры, выбор новой ставки, повторение игры и выдача новых карт

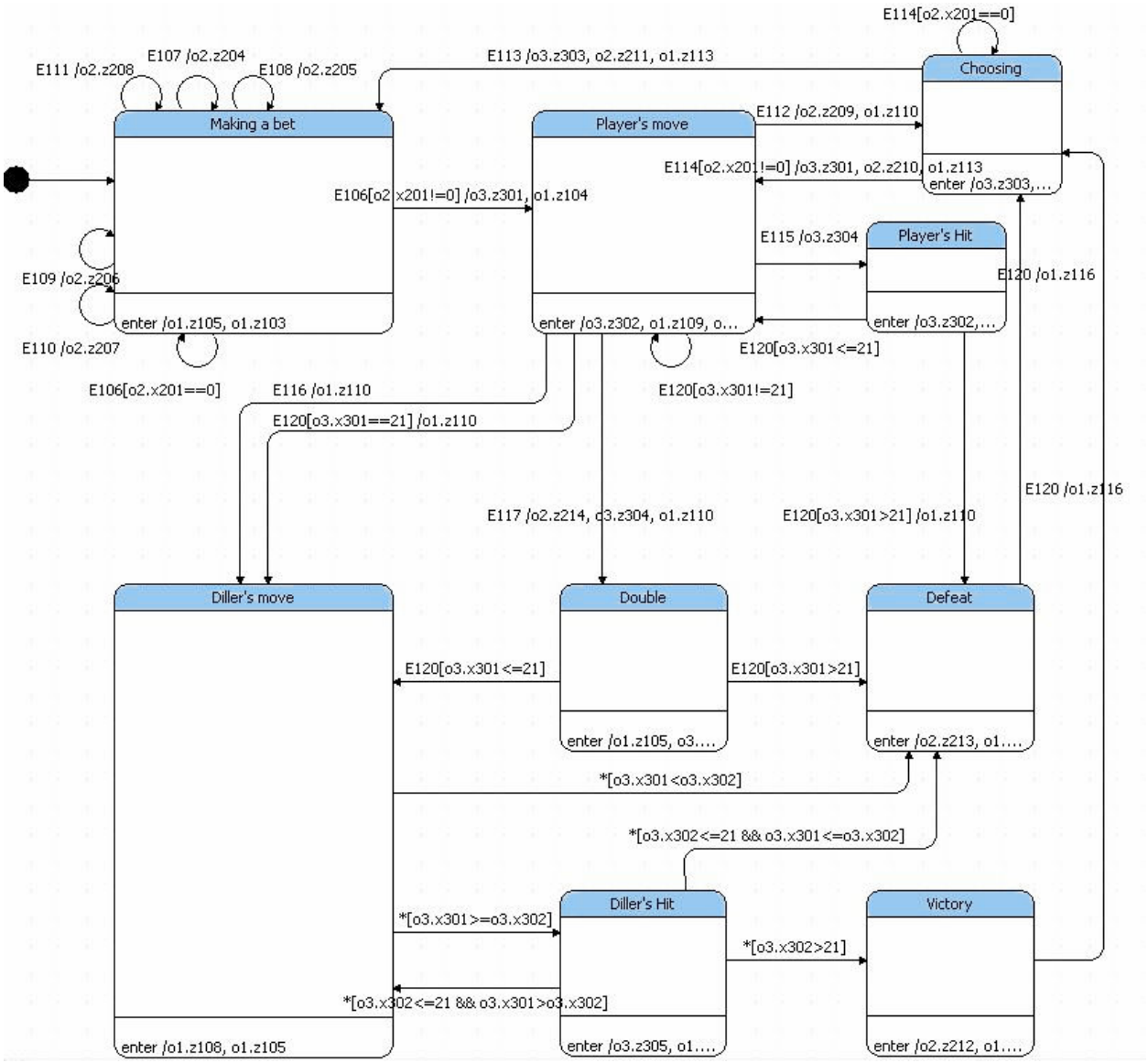

Рис. 7. Граф перехода автомата *A2*

3. Подходы к исполнению

3.1. Интерпретационный подход

При реализации программы на основе интерпретационного подхода автоматы не преобразуются в *Java*-код, а интерпретируются — сами запускаются, как исходный код. При этом классы, описывающие источники событий и объекты управления, компилируются в *Java* байт-код, а интерпретатор по *XML*-описанию автомата имитирует его работу, выводя в консоль протокол работы. В статье [2] акцентируются внимание на том, что триединое использование автоматов (при спецификации, программировании и протоколировании) является важнейшей особенностью *Switch*-технологии. По мнению авторов статьи: «протоколы позволяют наблюдать за ходом выполнения программы и демонстрируют тот факт, что автоматы являются не «картинками», а реально действующими сущностями».

Недостатками интерпретационного подхода являются низкое быстродействие и необходимость подключения многих библиотек. Для устранения этих недостатков был разработан компиляционный подход.

3.2. Компиляционный подход

При компиляционном подходе по *XML*-описанию автомата строится изоморфный ему *Java*-код, который потом будет скомпилирован в байт-код. Это позволяет (совместно с кодом входных и выходных воздействий) запускать программу без использования интерпретатора *UniMod*, уменьшая, таким образом, количество необходимых библиотек, а, следовательно, объем используемой памяти.

Компиляционный подход целесообразно применять для устройств с ограниченными ресурсами, например, для мобильных телефонов [3]. Заметим, что функционально приложения, построенные на базе обоих подходов, будут идентичны. Структурные схемы интерпретационного и компиляционного подходов приведены в статье [4].

Заключение

Выполненная работа показала, что автоматный подход к программированию устройств с “понятной” логикой и логических игр существенно упрощает процесс разработки и отладки. Применяя автоматный подход, проект можно представить в виде схем, которые, просты в понимании и логически выстроены. Представление в виде автоматов позволяет, не держать в голове весь проект целиком, а сосредотачиваться, на данном конкретном фрагменте (автомате). Это особенно важно для разработки больших и сложности программ. Используя же инструментальную среду *Unimod,* большая часть кода автоматически генерируется из схем связей и графов переходов.

Как развитие проекта, авторы видят реализацию электронного биллинга и использование данного проекта, как составляющей части Интернет-казино.

Источники

1. *Шалыто А. А., Туккель Н. И.* SWITCH-технология — автоматный подход к созданию программного обеспечения "реактивных" систем //Программирование. 2001. № 5. http://is.ifmo.ru/works/switch/1/
2. *Шалыто А. А., Туккель Н. И.* Программирование с явным выделением состояний //Мир ПК. 2001. № 8, 9. http://is.ifmo.ru/works/mirk/
3. *Шалыто А. А.* Технология автоматного программирования //Мир ПК. 2003. № 10. http://is.ifmo.ru/works/tech_aut_prog/
4. *Гуров В. С., Мазин М. А., Шалыто А. А.* UniMod — инструментальное средство для автоматного программирования //Научно-технический вестник. Вып. 30. Фундаментальные и прикладные исследования информационных систем и технологий. 2006. http://is.ifmo.ru/works/_instrsr.pdf
5. *Колыхматов И. И., Рыбак О. О., Шалыто А. А*. Моделирование устройства для продажи газированной воды на инструментальном средстве *UniMod*. http://is.ifmo.ru/download/vending-machine-ru.pdf
6. *Кулагин Д. И., Суслов А. В.* Моделирование устройства для продажи проездных билетов http://is.ifmo.ru/download/bilets_documentation.pdf

Приложение 1. Сгенерированное *XML*-описание

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE model PUBLIC "-//eVelopers Corp.//DTD State
machine model V1.0//EN" "http://www.evelopers.com/dtd/unimod/statemachine.dtd">
<model name="Model1">
  <controlledObject class="ru.ifmo.blackjack.ScreenControlledObject" name="o1"/>
  <controlledObject class="ru.ifmo.blackjack.MoneyControlledObject" name="o2"/>
  <controlledObject class="ru.ifmo.blackjack.CardsControlledObject" name="o3"/>
  <eventProvider class="ru.ifmo.blackjack.SystemEventProvider" name="p1">
    <association clientRole="p1" targetRef="A1"/>
  </eventProvider>
  <rootStateMachine>
    <stateMachineRef name="A1"/>
  </rootStateMachine>
  <stateMachine name="A1">
    <configStore class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A1" supplierRole="o1" targetRef="o1"/>
    <association clientRole="A1" supplierRole="A2" targetRef="A2"/>
    <association clientRole="A1" supplierRole="o2" targetRef="o2"/>
    <state name="Top" type="NORMAL">
      <state name="s2" type="FINAL"/>
      <state name="Choosing initial money count" type="NORMAL"/>
      <state name="Game" type="NORMAL">
        <stateMachineRef name="A2"/>
      </state>
      <state name="s1" type="INITIAL"/>
    </state>
    <transition event="E118" sourceRef="Choosing initial money count" targetRef="s2">
      <outputAction ident="o1.z115"/>
    </transition>
    <transition event="E101" sourceRef="Choosing initial money count" targetRef="Choosing initial
money count">
      <outputAction ident="o2.z201"/>
    </transition>
    <transition event="E102" sourceRef="Choosing initial money count" targetRef="Choosing initial
money count">
      <outputAction ident="o2.z202"/>
    </transition>
    <transition event="E103" sourceRef="Choosing initial money count" targetRef="Choosing initial
money count">
      <outputAction ident="o2.z203"/>
    </transition>
    <transition event="E104" sourceRef="Choosing initial money count" targetRef="Game">
      <outputAction ident="o1.z101"/>
    </transition>
    <transition event="E105" sourceRef="Game" targetRef="s2">
      <outputAction ident="o1.z115"/>
    </transition>
    <transition sourceRef="s1" targetRef="Choosing initial money count"/>
  </stateMachine>
```

```
<stateMachine name="A2">
 <configStore class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
 <association clientRole="A2" supplierRole="o1" targetRef="o1"/>
 <association clientRole="A2" supplierRole="o2" targetRef="o2"/>
 <association clientRole="A2" supplierRole="o3" targetRef="o3"/>
 <state name="Top" type="NORMAL">
  <state name="Choosing" type="NORMAL">
   <outputAction ident="o3.z303"/>
   <outputAction ident="o1.z114"/>
   <outputAction ident="o1.z105"/>
  </state>
  <state name="Making a bet" type="NORMAL">
   <outputAction ident="o1.z105"/>
   <outputAction ident="o1.z103"/>
  </state>
  <state name="Player's move" type="NORMAL">
   <outputAction ident="o1.z105"/>
   <outputAction ident="o3.z302"/>
   <outputAction ident="o1.z109"/>
   <outputAction ident="o1.z116"/>
  </state>
  <state name="s1" type="INITIAL"/>
  <state name="Player's Hit" type="NORMAL">
   <outputAction ident="o1.z105"/>
   <outputAction ident="o3.z302"/>
  </state>
  <state name="Diller's move" type="NORMAL">
   <outputAction ident="o1.z108"/>
   <outputAction ident="o1.z105"/>
   <outputAction ident="o3.z302"/>
  </state>
  <state name="Double" type="NORMAL">
   <outputAction ident="o1.z105"/>
   <outputAction ident="o3.z302"/>
  </state>
  <state name="Defeat" type="NORMAL">
   <outputAction ident="o2.z213"/>
   <outputAction ident="o1.z107"/>
   <outputAction ident="o3.z302"/>
  </state>
  <state name="Diller's Hit" type="NORMAL">
   <outputAction ident="o3.z305"/>
   <outputAction ident="o1.z105"/>
   <outputAction ident="o3.z302"/>
  </state>
  <state name="Victory" type="NORMAL">
   <outputAction ident="o2.z212"/>
   <outputAction ident="o1.z106"/>
   <outputAction ident="o3.z302"/>
  </state>
 </state>
 <transition event="E114" guard="o2.x201==0" sourceRef="Choosing" targetRef="Choosing"/>
```

```
<transition event="E113" sourceRef="Choosing" targetRef="Making a bet">
  <outputAction ident="o3.z303"/>
  <outputAction ident="o2.z211"/>
  <outputAction ident="o1.z113"/>
</transition>
<transition event="E114" guard="o2.x201!=0" sourceRef="Choosing" targetRef="Player's move">
  <outputAction ident="o3.z301"/>
  <outputAction ident="o2.z210"/>
  <outputAction ident="o1.z113"/>
</transition>
<transition event="E111" sourceRef="Making a bet" targetRef="Making a bet">
  <outputAction ident="o2.z208"/>
</transition>
<transition event="E107" sourceRef="Making a bet" targetRef="Making a bet">
  <outputAction ident="o2.z204"/>
</transition>
<transition event="E108" sourceRef="Making a bet" targetRef="Making a bet">
  <outputAction ident="o2.z205"/>
</transition>
<transition event="E109" sourceRef="Making a bet" targetRef="Making a bet">
  <outputAction ident="o2.z206"/>
</transition>
<transition event="E110" sourceRef="Making a bet" targetRef="Making a bet">
  <outputAction ident="o2.z207"/>
</transition>
<transition event="E106" guard="o2.x201==0" sourceRef="Making a bet" targetRef="Making a
bet"/>
<transition event="E106" guard="o2.x201!=0" sourceRef="Making a bet" targetRef="Player's
move">
  <outputAction ident="o3.z301"/>
  <outputAction ident="o1.z104"/>
</transition>
<transition event="E112" sourceRef="Player's move" targetRef="Choosing">
  <outputAction ident="o2.z209"/>
  <outputAction ident="o1.z110"/>
</transition>
<transition event="E120" guard="o3.x301!=21" sourceRef="Player's move" targetRef="Player's
move"/>
<transition event="E115" sourceRef="Player's move" targetRef="Player's Hit">
  <outputAction ident="o3.z304"/>
</transition>
<transition event="E120" guard="o3.x301==21" sourceRef="Player's move" targetRef="Diller's
move">
  <outputAction ident="o1.z110"/>
</transition>
<transition event="E116" sourceRef="Player's move" targetRef="Diller's move">
  <outputAction ident="o1.z110"/>
</transition>
<transition event="E117" sourceRef="Player's move" targetRef="Double">
  <outputAction ident="o2.z214"/>
  <outputAction ident="o3.z304"/>
  <outputAction ident="o1.z110"/>
```

```
    <outputAction ident="o1.z108"/>
   </transition>
   <transition sourceRef="s1" targetRef="Making a bet"/>
   <transition event="E120" guard="o3.x301&lt;=21" sourceRef="Player's Hit" targetRef="Player's
move"/>
   <transition event="E120" guard="o3.x301&gt;21" sourceRef="Player's Hit" targetRef="Defeat">
    <outputAction ident="o1.z110"/>
   </transition>
   <transition event="*" guard="o3.x301&lt;o3.x302" sourceRef="Diller's move" targetRef="Defeat"/>
   <transition event="*" guard="o3.x301&gt;=o3.x302" sourceRef="Diller's move" targetRef="Diller's
Hit"/>
   <transition event="E120" guard="o3.x301&lt;=21" sourceRef="Double" targetRef="Diller's
move"/>
   <transition event="E120" guard="o3.x301&gt;21" sourceRef="Double" targetRef="Defeat"/>
   <transition event="E120" sourceRef="Defeat" targetRef="Choosing">
    <outputAction ident="o1.z116"/>
   </transition>
   <transition event="*" guard="o3.x302&lt;=21 &amp;&amp; o3.x301&gt;o3.x302"
sourceRef="Diller's Hit" targetRef="Diller's move"/>
   <transition event="*" guard="o3.x302&lt;=21 &amp;&amp; o3.x301&lt;=o3.x302"
sourceRef="Diller's Hit" targetRef="Defeat"/>
   <transition event="*" guard="o3.x302&gt;21" sourceRef="Diller's Hit" targetRef="Victory"/>
   <transition event="E120" sourceRef="Victory" targetRef="Choosing">
    <outputAction ident="o1.z116"/>
   </transition>
  </stateMachine>
</model>
```

Приложение 2. Исходные тексты программы, написанные вручную

// Класс, реализующий запуск приложения, при компиляционном подходе
Main.java

Основные классы программы

// Класс, поставщик событий
SystemEventProvider.java

// Класс, реализующий объект управления, отвечающий за деньги
// MoneyControlledObject.java

// Класс, реализующий объект управления, отвечающий за интерфейс
ScreenControlledObject.java

// Класс, реализующий объект управления, отвечающий за карты
CardsControlledObject.java

Вспомогательные классы программы

// Класс, представляющий игральную карту
Card.java

// Класс, представляющий внешний образ карты
CardComponent.java

// Класс, представляющий внешний образ карт
JCardsList.java

// Класс, представляющий окно выбора начального банка игрока
StartFrame.java

// Класс, представляющий окно игры
BlackjackFrame.java