Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Факультет «Информационных технологий и программирования»

И. А. Балтийский, С. И. Гиндин

Моделирование работы банкомата

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

Санкт-Петербург
2008

Оглавление

1. Введение

Трудно представить себе современного человека, проживающего в городе, без множества разнообразных технических устройств, прочно вошедших в обиход. На сегодняшний день большинство подобных устройств являются программируемыми. Постоянно возрастающий спрос на все новые их виды ставит вопрос об эффективном подходе к их проектированию и реализации.

Одним из ключевых факторов в этой сфере является надежность, от которой часто зависит не только репутация марки или фирмы-изготовителя, но и человеческая жизнь. Для обеспечения надежности необходимо выполнить ряд требований к технологии программирования устройств. Так, немаловажными качествами того или иного подхода к проектированию являются простота и формализуемость, которые позволили бы с определенным уровнем строгости доказывать, например, корректность протокола, положенного в основу реализации. С другой стороны, существенное повышение надежности может быть достигнуто за счет частичной или полной автоматизации процесса создания программ, что не только уменьшит человеческие трудозатраты, но и позволит осуществлять автоматический контроль качества.

В данной работе предлагается использовать автоматный подход, отвечающий указанным выше критериям, и приводится пример программы для устройства, реализованного на основе такого метода. Основная идея рассматриваемого подхода заключается в том, что при создании ядра программы роль человека сводится к проектированию и созданию формальной спецификации, представляющей собой систему конечных автоматов, по которой при помощи специальных средств генерируется конкретная реализация на некотором языке программирования. Подробнее про автоматное программирование можно прочитать, например, в статье [2].

В настоящем проекте на основе автоматного подхода реализовано программное обеспечение для эмуляции системы банкомата – *Automated Teller Machine* (ATM). В качестве инструментального средства использован *Unimod* [1, 2], генерирующий код на языке *Java*.

Отметим, что существуют и другие варианты решения этой задачи со схожим подходом, такие, как, например, в работе [4]. Предлагаемое в этой работе решение отличается от предыдущих созданием более развитой и углубленной модели банковской системы. С одной стороны, возможности эмуляции банкомата с точки зрения клиента соответствуют полному набору возможностей реального банкомата. Так, например, имеется возможность совершать оплату мобильной связи, не реализованная в других работах. С другой стороны, и эмуляция работы сервера банка отражает один из вариантов реализации, применяемый на практике – в частности, выделена подсистема авторизации пользователя.

Используемый в данной работе подход применяется и к смежным задачам, что продемонстрировано в работах [5, 6].

2. Постановка задачи

Под банкоматом будем понимать автоматизированное устройство, позволяющее удаленно осуществлять операции, связанные с:

- аутентификацией пользователя (держателя счета в банке);
- просмотром текущего состояния счета;
- снятием денег со счета;
- осуществлением различных платежей.

Все операции со счетом могут сопровождаться распечаткой чека, содержащего информацию о произведенном действии. В случае с осуществлением платежей распечатка производится автоматически, во всех остальных – по желанию клиента.

Для выполнения вышеуказанных операций банкомат должен связываться с банком. При этом он отправляет серверу банка и получает от него сообщения согласно определенному протоколу.

В данной работе будет рассмотрена задача программной эмуляции работы банкомата, соответствующего приведенному выше описанию. Структура программы содержит две различные системы: протокол общения банка и банкомата, а также клиентский интерфейс.

С точки зрения клиента, взаимодействие с эмуляцией банкомата должно выглядеть так же, как и в реальности: он должен иметь возможность начать работу с конкретной картой, которая должна быть идентифицирована устройством, а затем при помощи меню совершать в любой последовательности перечисленные выше денежные операции. В реальном банкомате идентификация карты происходит путем автоматического считывания ее номера. В данной работе в качестве этой процедуры будет использоваться введение номера карты с клавиатуры. Аутентификация, как правило, производится на основе введения клиентом так называемого *PIN*-кода (*Personal Identification Number*), который проверяется сервером банка на соответствие считанному номеру карты. В данной работе также будет использовано введение пользователем *PIN*-кода его карты с клавиатуры.

Естественно реализовать меню на основе графического пользовательского интерфейса, имитирующего вид типичного банкомата. Эмуляция банкомата должна будет позволять проводить все вышеперечисленные операции со счетом. При этом в качестве операции платежа будет рассматриваться оплата телекоммуникационных услуг операторов мобильной связи.

Взаимодействие банкомата с сервером банка производится по специальному протоколу, который будет отражен в данном проекте. Детали протокола будут рассмотрены в разделе, посвященном сценарию общения пользователя с эмуляцией банкомата.

3. СЦЕНАРИИ РАБОТЫ С ЭМУЛЯЦИЕЙ БАНКОМАТА

3.1. Неформальное описание

Модель общения банкомата с сервером банка построена на основе транзакций – в ходе взаимодействия с пользователем устройство банкомата накапливает вводимую информацию в специальном внутреннем списке, отправляя серверу по требованию совершения операции все описание транзакции. Так, в начале работы пользователь вводит номер карты. После этого банкомат запрашивает у него *PIN*-код. Как номер карты, так и введенный код запоминаются во внутреннем списке. Банкомат запрашивает у сервера авторизацию для карты. В случае неверного ввода *PIN*-кода он запрашивается повторно. В случае неверного ввода *PIN*-кода три раза подряд работа с банкоматом принудительно завершается.

После успешной проверки *PIN*-кода пользователю становится доступно основное меню, в котором он может выбрать одно из следующих действий:

- снять деньги со счета;
- просмотреть остаток счета;
- осуществить платеж;
- забрать карту.

При выборе последнего пункта пользователю возвращается карта, и работа с банкоматом завершается. При выборе любого другого пункта банкомат добавляет к внутреннему списку вид выбранной пользователем операции. Для совершения операции клиент должен пройти через ряд меню, в которых ему предлагается выбрать один из возможных вариантов или ввести число с клавиатуры. Эту информацию автомат также добавляет во внутренний список.

Когда все необходимые для совершения транзакции данные накоплены, автомат отправляет серверу все содержимое внутреннего списка, очищая его для дальнейшей работы. При этом клиенту отображается информация о результате операции. После этого банкомат вновь отображает главное меню.

В случае выбора пользователем еще одной операции всю информацию о карте необходимо ввести заново. В реальном банкомате повторное считывание номера производится автоматически (пользователь не должен заново вставлять карту). При эмуляции банкомата номер будет запоминаться, так как он имитирует карту.

3.1.1. Снятие денег со счета (cash withdrawal)

При снятии денег со счета банкомат запрашивает у пользователя ввод необходимой ему суммы. В реальном банкомате для этого доступны только цифровые клавиши от 0 до 9, эмуляция же будет проверять, что считанное с клавиатуры значение является неотрицательным целым числом в десятичной системе счисления. После того, как ввод запрашиваемой суммы подтвержден, банкомат добавляет введенную сумму к накопленной информации и отправляет ее на сервер банка.

Теперь происходит проверка возможности снять со счета запрашиваемую сумму – должно быть достаточно денег на счете.

В случае нарушения этого условия банкомату отправляется сообщение об ошибке, которое он отображает пользователю.

Если сервер банка подтверждает транзакцию, то на сервере происходит ее учет (непосредственное снятие денег со счета), а банкомату отправляется сообщение о выдаче денег пользователю. Банкомат отображает пользователю меню, в котором он может выбрать распечатку чека с информацией о транзакции.

После удачного или неудачного завершения операции банкомат отображает пользователю главное меню, возвращаясь в исходное состояние.

3.1.2. Состояние счета (balance inquiry)

После выбора этой операции банкомат отправляет серверу список с информацией о требуемой операции. После этого устройство банкомата получает сообщение от сервера, содержащее текущее состояние счета. Пользователю предоставляется возможность распечатать эту информацию, и в случае утвердительного ответа выдается чек. После этого на экран выводится состояние счета. На данном этапе клиент может забрать карту либо продолжить работу. В последнем случае банкомат отображает главное меню и возвращается в исходное состояние.

3.1.3. Осуществление платежей (payment)

Пользователь выбирает сначала оператора услуг мобильной связи, затем вводит номер телефона и количество денег, которое он хочет перевести (все эти данные сохраняются во внутреннем списке). В целях устранения ошибок, связанных с введением номера (10 цифр), банкомат предлагает пользователю подтвердить введенную информацию или ввести ее заново. В случае подтверждения информации внутренний список, содержащий описание операции, отправляется на сервер.

После проверки возможности осуществления платежа, подобного тому, который был описан выше, сервер либо фиксирует факт оплаты мобильных услуг и отправляет банкомату извещение об успешном завершении транзакции, либо отправляет банкомату сообщение об ошибке. Во втором случае пользователю отображается сообщение об ошибке, а в первом – распечатывается чек с информацией о совершенном платеже.

3.2. Формальное описание

Для реализации вышеописанного сценария необходимо выделить сначала *объекты управления*, над которыми будут производиться операции. Во-первых, это то, что непосредственно видит клиент, который работает с банкоматом, – *пользовательский интерфейс*, который будет рассматриваться как единая система, позволяющая отображать на экране банкомата информацию. Во-вторых, это *система удаленного взаимодействия* банкомата с сервером. Далее, объектом управления будет являться и *сервер банка*, обрабатывающий запросы банкомата. Наконец, выделим также специальную подсистему банка, отвечающую за авторизацию пользователей, – с ней работает сервер банка, проверяя такую информацию, как, например, введенный *PIN*-код.

Управление перечисленными объектами инициируется *источниками событий,* в качестве которых рассматриваются система, регистрирующая номер карты клиента, клавиатура банкомата, собственно банкомат, отправляющий запросы серверу, и сервер, отвечающий на эти запросы.

Управление осуществляется *системой взаимодействующих автоматов.*

Формальное описание, содержащее в себе объекты управления, источники событий и систему автоматов, приведено ниже.

3.2.1. Схема связей автоматов с поставщиками событий и объектами управления

На рис. 1 приведена схема связи *системы взаимодействующих автоматов*, *поставщиков событий* и *объектов управления*.

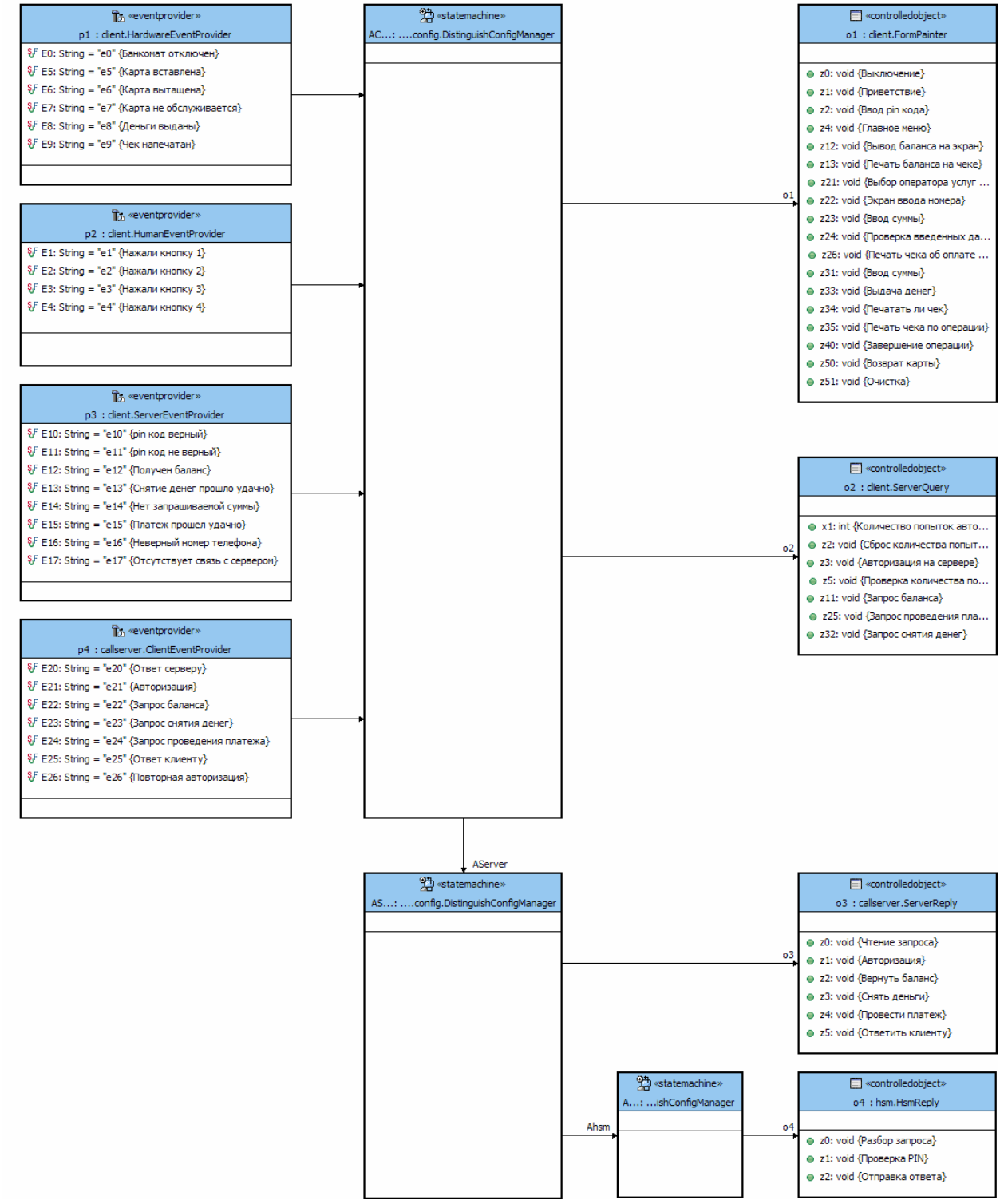

Рис. 1. Схема связей

3.2.2. Автомат, отвечающий за работу банкомата

На рис. 2 приведена схема автомата, отвечающего за работу банкомата. Объектами управления, как можно видеть из схемы на рис. 1, являются *пользовательский интерфейс* и *система удаленного взаимодействия* банкомата с сервером.

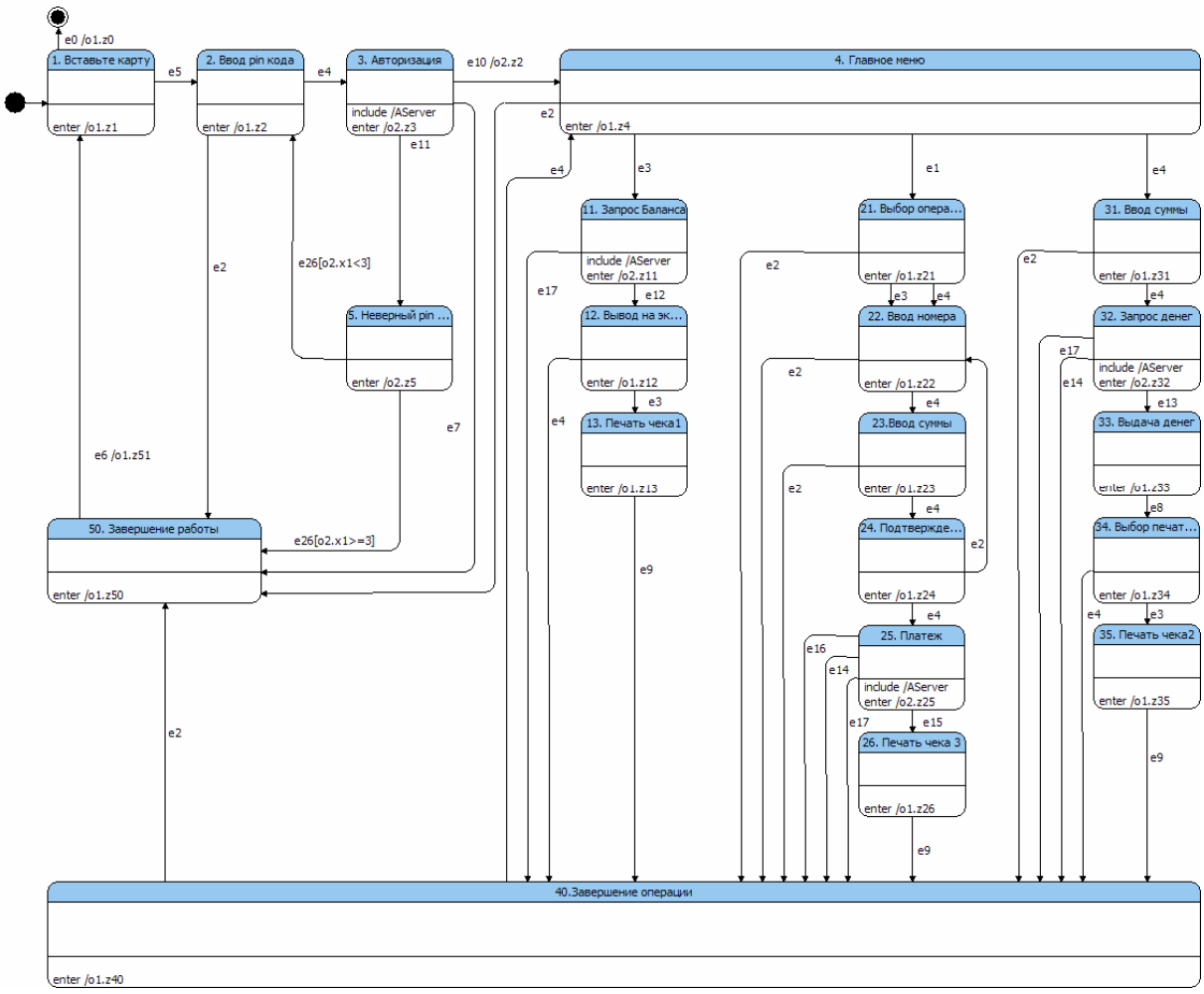

Рис. 2. Автомат, отвечающий за работу банкомата

3.2.3. Автомат, отвечающий за работу сервера

На рис. 3 приведен автомат, отвечающий за работу серверной системы банка.

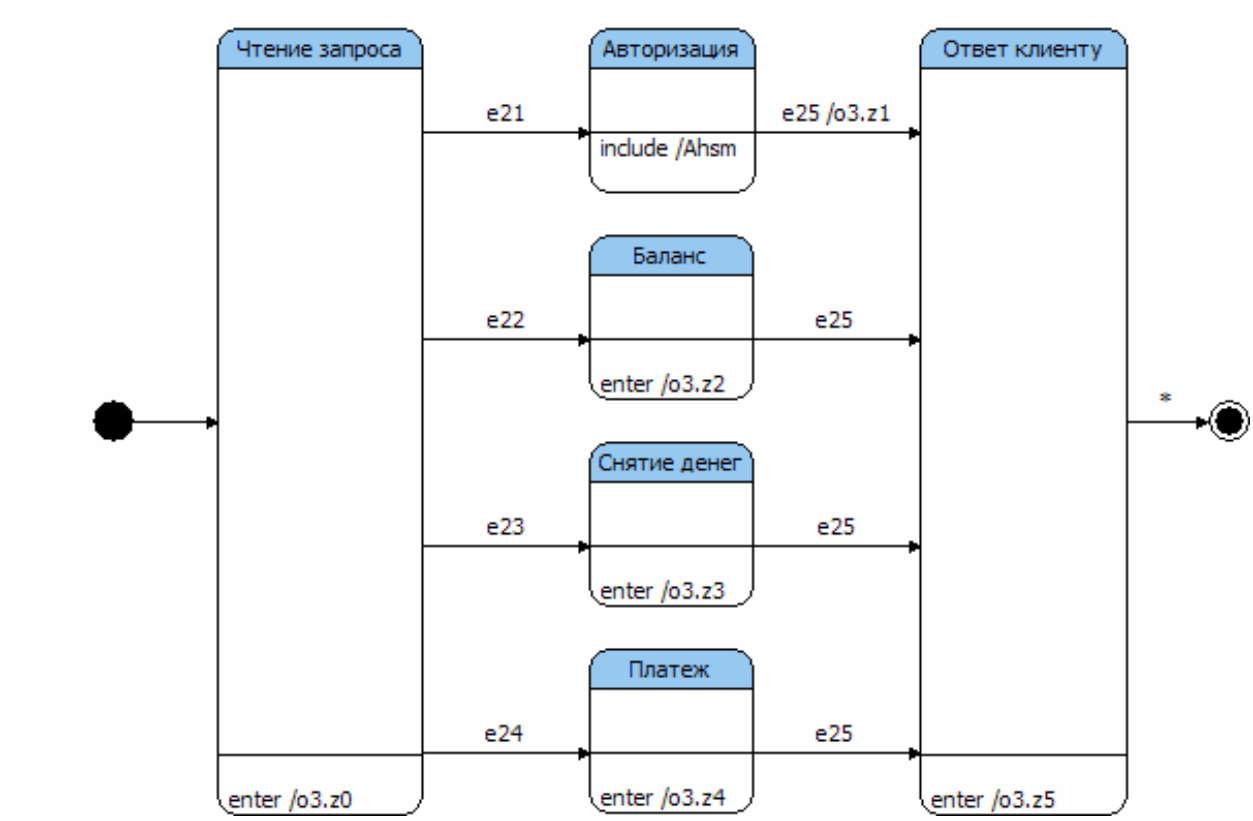

Рис. 3. Автомат, отвечающий за работу сервера

3.2.4. Автомат подсистемы авторизации

На рис. 4 изображен автомат, отвечающий за работу серверной подсистемы банка, отвечающей за авторизацию пользователя.

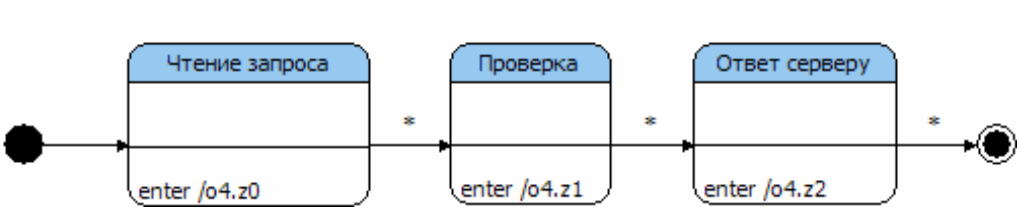

Рис. 4. Автомат подсистемы авторизации

3.3. Примеры внешнего вида модели банкомата

На рис. 5–7 приведены примеры внешнего вида модели банкомата в некоторых режимах.

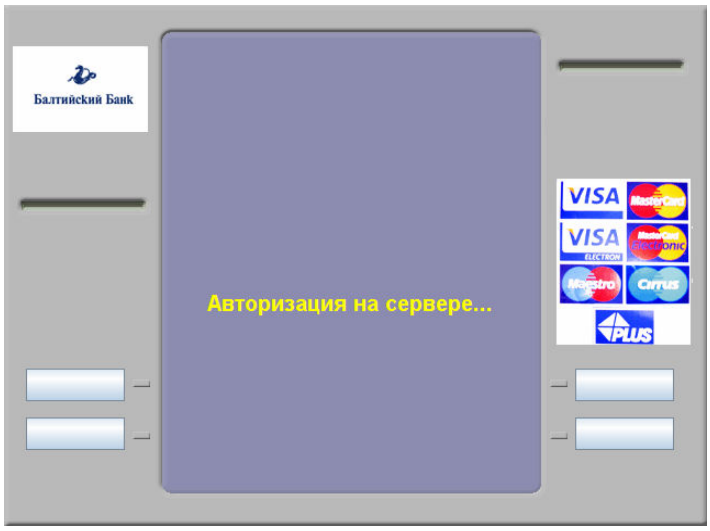

Рис. 5. Авторизация на сервере

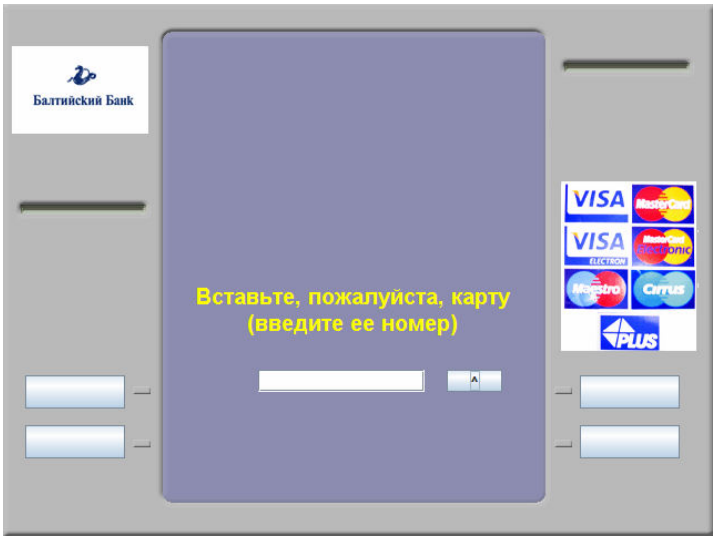

Рис. 6. Диалог с пользователем

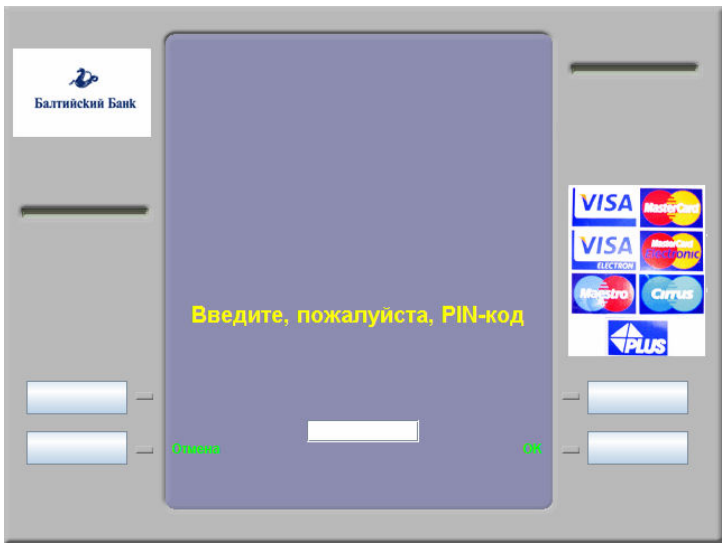

Рис. 7. Ввод *PIN*-кода

4. Исполнение программы

Использованное в ходе разработки программы инструментальное средство *Unimod* поддерживает два подхода к исполнению программ: интерпретационный и компилятивный.

4.1. Интерпретационный подход

При интерпретационном подходе автоматы запускаются, как исходный код – используется *XML*-описание модели (Приложение 1), которое интерпретируется *java*-классом. Для работы интерпретатору необходим *jar*-файл (архив) с проектом и библиотекой *Unimod.*

На рис. 8 приведена схема запуска приложения при интерпретационном подходе.

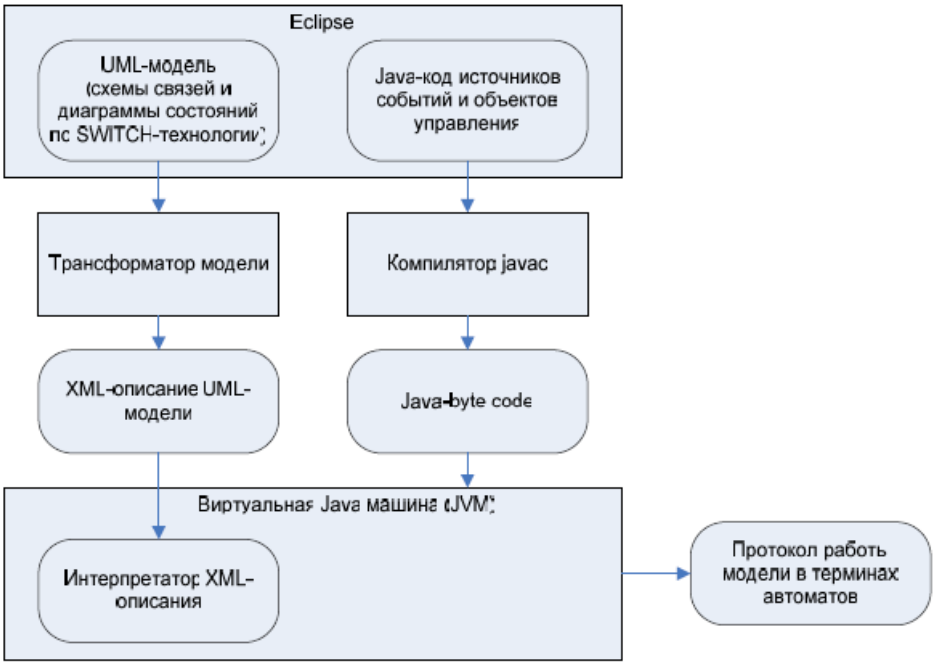

Рис. 8. Интерпретационный подход

4.2. Компилятивный подход

При компилятивном подходе *XML*-описание транслируется в *Java*-класс (Приложение 2). Этот класс и классы, реализующие источники событий и объекты управления, компилируются в единое *Java*-приложение, не зависящее от средства *Unimod*. Для исполнения скомпилированной программы потребуется библиотека, поставляемая вместе с инструментальным средством *Unimod*.

На рис. 9 приведена схема запуска приложения при компилятивном подходе.

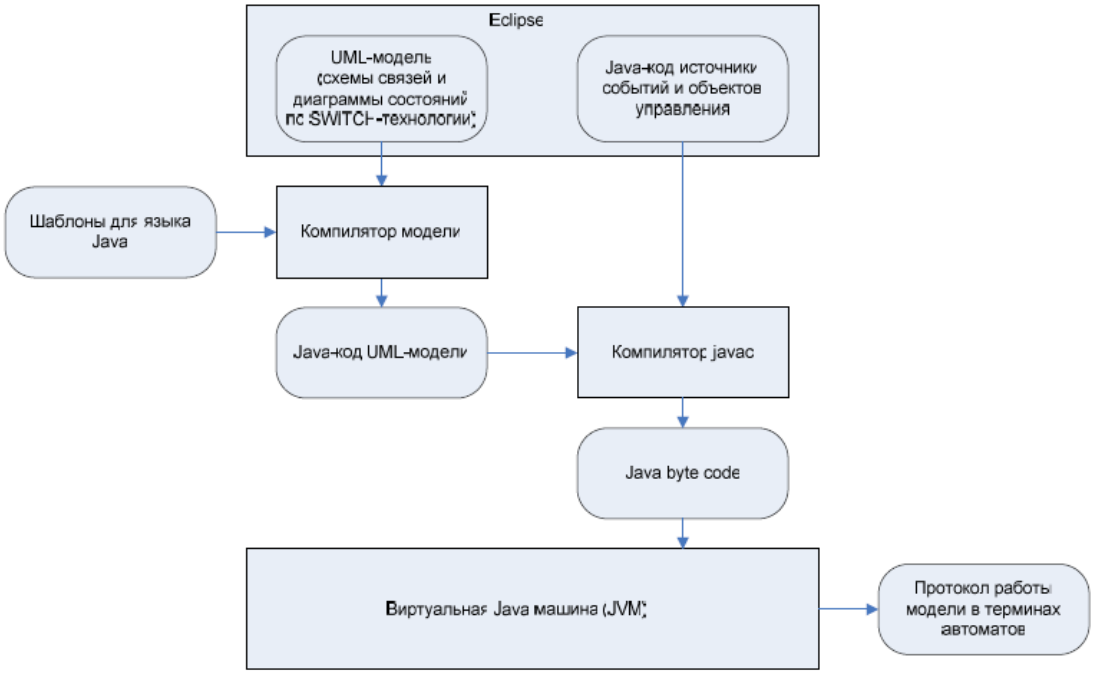

Рис. 9. Компилятивный подход

Выводы

Как следует из данной работы, использование автоматного подхода при разработке программ позволяет еще на этапе проектирования избежать существенных ошибок, давая возможность четко и наглядно представить логику программы. Отметим также и значительное улучшение документации по сравнению с традиционным подходом, сокращающее затраты на освоение того или иного программного продукта сторонними разработчиками. Было также продемонстрировано использование инструментального средства *Unimod* для построения программ на основе автоматного подхода.

5. Источники

1. Сайт по автоматному программированию и мотивации к творчеству. http://is.ifmo.ru/
2. Сайт проекта Unimod на SourceForge http://sourceforge.net/projects/unimod/
3. Automata-based programming (programming technology) http://en.wikipedia.org/wiki/Automata-based_programming_(programming_technology)
4. Моделирование работы банкомата. http://is.ifmo.ru/unimod-projects/bankomat/
5. Автоматизированная система оплаты мобильного телефона http://is.ifmo.ru/unimod-projects/teleplay/
6. Моделирование устройства для обмена валюты http://is.ifmo.ru/unimod-projects/exchange/

Приложения

Приложение 1. XML-описание модели (интерпретационный подход)

AClient.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><!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="client.FormPainter" name="o1"/>
  <controlledObject class="client.ServerQuery" name="o2"/>
  <controlledObject class="callserver.ServerReply" name="o3"/>
  <controlledObject class="hsm.HsmReply" name="o4"/>
  <eventProvider class="client.HardwareEventProvider" name="p1">
    <association clientRole="p1" targetRef="AClient"/>
  </eventProvider>
  <eventProvider class="client.HumanEventProvider" name="p2">

    <association clientRole="p2" targetRef="AClient"/>
  </eventProvider>
  <eventProvider class="client.ServerEventProvider" name="p3">
    <association clientRole="p3" targetRef="AClient"/>
  </eventProvider>
  <eventProvider class="callserver.ClientEventProvider" name="p4">
    <association clientRole="p4" targetRef="AClient"/>
  </eventProvider>
  <rootStateMachine>

    <stateMachineRef name="AClient"/>
  </rootStateMachine>
  <stateMachine name="AClient">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="AClient" supplierRole="o1" targetRef="o1"/>
    <association clientRole="AClient" supplierRole="o2" targetRef="o2"/>
    <association clientRole="AClient" supplierRole="AServer"
targetRef="AServer"/>
    <state name="Top" type="NORMAL">
      <state name="s2" type="FINAL"/>

      <state name="1. Вставьте карту" type="NORMAL">
        <outputAction ident="o1.z1"/>
      </state>
      <state name="2. Ввод pin кода" type="NORMAL">
        <outputAction ident="o1.z2"/>
      </state>
      <state name="3. Авторизация" type="NORMAL">
        <stateMachineRef name="AServer"/>
        <outputAction ident="o2.z3"/>

      </state>
      <state name="4. Главное меню" type="NORMAL">
        <outputAction ident="o1.z4"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="11. Запрос Баланса" type="NORMAL">
        <stateMachineRef name="AServer"/>
        <outputAction ident="o2.z11"/>
      </state>
```

```
      <state name="21. Выбор оператора" type="NORMAL">
        <outputAction ident="o1.z21"/>
      </state>
      <state name="31. Ввод суммы" type="NORMAL">
        <outputAction ident="o1.z31"/>
      </state>
      <state name="5. Неверный pin код" type="NORMAL">
        <outputAction ident="o2.z5"/>
      </state>

      <state name="12. Вывод на экран" type="NORMAL">
        <outputAction ident="o1.z12"/>
      </state>
      <state name="22. Ввод номера" type="NORMAL">
        <outputAction ident="o1.z22"/>
      </state>
      <state name="32. Запрос денег" type="NORMAL">
        <stateMachineRef name="AServer"/>
        <outputAction ident="o2.z32"/>

      </state>
      <state name="33. Выдача денег" type="NORMAL">
        <outputAction ident="o1.z33"/>
      </state>
      <state name="13. Печать чека1" type="NORMAL">
        <outputAction ident="o1.z13"/>
      </state>
      <state name="23.Ввод суммы" type="NORMAL">
        <outputAction ident="o1.z23"/>

      </state>
      <state name="34. Выбор печати чека" type="NORMAL">
        <outputAction ident="o1.z34"/>
      </state>
      <state name="50. Завершение работы" type="NORMAL">
        <outputAction ident="o1.z50"/>
      </state>
      <state name="24. Подтверждение" type="NORMAL">
        <outputAction ident="o1.z24"/>

      </state>
      <state name="35. Печать чека2" type="NORMAL">
        <outputAction ident="o1.z35"/>
      </state>
      <state name="25. Платеж" type="NORMAL">
        <stateMachineRef name="AServer"/>
        <outputAction ident="o2.z25"/>
      </state>
      <state name="26. Печать чека 3" type="NORMAL">

        <outputAction ident="o1.z26"/>
      </state>
      <state name="40.Завершение операции" type="NORMAL">
        <outputAction ident="o1.z40"/>
      </state>
    </state>
    <transition event="e0" sourceRef="1. Вставьте карту" targetRef="s2">
      <outputAction ident="o1.z0"/>
    </transition>

    <transition event="e5" sourceRef="1. Вставьте карту" targetRef="2. Ввод
pin кода"/>
```

```
    <transition event="e4" sourceRef="2. Ввод pin кода" targetRef="3.
Авторизация"/>
    <transition event="e2" sourceRef="2. Ввод pin кода" targetRef="50.
Завершение работы"/>
    <transition event="e10" sourceRef="3. Авторизация" targetRef="4. Главное
меню">
      <outputAction ident="o2.z2"/>
    </transition>
    <transition event="e11" sourceRef="3. Авторизация" targetRef="5. Неверный
pin код"/>
    <transition event="e7" sourceRef="3. Авторизация" targetRef="50.
Завершение работы"/>
    <transition event="e3" sourceRef="4. Главное меню" targetRef="11. Запрос
Баланса"/>

    <transition event="e1" sourceRef="4. Главное меню" targetRef="21. Выбор
оператора"/>
    <transition event="e4" sourceRef="4. Главное меню" targetRef="31. Ввод
суммы"/>
    <transition event="e2" sourceRef="4. Главное меню" targetRef="50.
Завершение работы"/>
    <transition sourceRef="s1" targetRef="1. Вставьте карту"/>
    <transition event="e12" sourceRef="11. Запрос Баланса" targetRef="12.
Вывод на экран"/>
    <transition event="e17" sourceRef="11. Запрос Баланса"
targetRef="40.Завершение операции"/>
    <transition event="e3" sourceRef="21. Выбор оператора" targetRef="22.
Ввод номера"/>
    <transition event="e4" sourceRef="21. Выбор оператора" targetRef="22.
Ввод номера"/>
    <transition event="e2" sourceRef="21. Выбор оператора"
targetRef="40.Завершение операции"/>

    <transition event="e4" sourceRef="31. Ввод суммы" targetRef="32. Запрос
денег"/>
    <transition event="e2" sourceRef="31. Ввод суммы"
targetRef="40.Завершение операции"/>
    <transition event="e26" guard="o2.x1&amp;lt;3" sourceRef="5. Неверный pin
код" targetRef="2. Ввод pin кода"/>
    <transition event="e26" guard="o2.x1&amp;gt;=3" sourceRef="5. Неверный pin
код" targetRef="50. Завершение работы"/>
    <transition event="e3" sourceRef="12. Вывод на экран" targetRef="13.
Печать чека1"/>
    <transition event="e4" sourceRef="12. Вывод на экран"
targetRef="40.Завершение операции"/>
    <transition event="e4" sourceRef="22. Ввод номера" targetRef="23.Ввод
суммы"/>
    <transition event="e2" sourceRef="22. Ввод номера"
targetRef="40.Завершение операции"/>
    <transition event="e13" sourceRef="32. Запрос денег" targetRef="33.
Выдача денег"/>

    <transition event="e14" sourceRef="32. Запрос денег"
targetRef="40.Завершение операции"/>
    <transition event="e17" sourceRef="32. Запрос денег"
targetRef="40.Завершение операции"/>
    <transition event="e8" sourceRef="33. Выдача денег" targetRef="34. Выбор
печати чека"/>
    <transition event="e9" sourceRef="13. Печать чека1"
targetRef="40.Завершение операции"/>
    <transition event="e4" sourceRef="23.Ввод суммы" targetRef="24.
Подтверждение"/>
    <transition event="e2" sourceRef="23.Ввод суммы" targetRef="40.Завершение
операции"/>
```

```
    <transition event="e3" sourceRef="34. Выбор печати чека" targetRef="35.
Печать чека2"/>
    <transition event="e4" sourceRef="34. Выбор печати чека"
targetRef="40.Завершение операции"/>
    <transition event="e6" sourceRef="50. Завершение работы" targetRef="1.
Вставьте карту">

      <outputAction ident="o1.z51"/>
    </transition>
    <transition event="e2" sourceRef="24. Подтверждение" targetRef="22. Ввод
номера"/>
    <transition event="e4" sourceRef="24. Подтверждение" targetRef="25.
Платеж"/>
    <transition event="e9" sourceRef="35. Печать чека2"
targetRef="40.Завершение операции"/>
    <transition event="e15" sourceRef="25. Платеж" targetRef="26. Печать чека
3"/>
    <transition event="e17" sourceRef="25. Платеж" targetRef="40.Завершение
операции"/>
    <transition event="e14" sourceRef="25. Платеж" targetRef="40.Завершение
операции"/>
    <transition event="e16" sourceRef="25. Платеж" targetRef="40.Завершение
операции"/>

    <transition event="e9" sourceRef="26. Печать чека 3"
targetRef="40.Завершение операции"/>
    <transition event="e4" sourceRef="40.Завершение операции" targetRef="4.
Главное меню"/>
    <transition event="e2" sourceRef="40.Завершение операции" targetRef="50.
Завершение работы"/>
  </stateMachine>
  <stateMachine name="AServer">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="AServer" supplierRole="o3" targetRef="o3"/>
    <association clientRole="AServer" supplierRole="Ahsm" targetRef="Ahsm"/>
    <state name="Top" type="NORMAL">

      <state name="Чтение запроса" type="NORMAL">
        <outputAction ident="o3.z0"/>
      </state>
      <state name="Авторизация" type="NORMAL">
        <stateMachineRef name="Ahsm"/>
      </state>
      <state name="Ответ клиенту" type="NORMAL">
        <outputAction ident="o3.z5"/>
      </state>

      <state name="Баланс" type="NORMAL">
        <outputAction ident="o3.z2"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="s2" type="FINAL"/>
      <state name="Снятие денег" type="NORMAL">
        <outputAction ident="o3.z3"/>
      </state>
      <state name="Платеж" type="NORMAL">

        <outputAction ident="o3.z4"/>
      </state>
    </state>
    <transition event="e21" sourceRef="Чтение запроса"
targetRef="Авторизация"/>
    <transition event="e22" sourceRef="Чтение запроса" targetRef="Баланс"/>
```

```
    <transition event="e23" sourceRef="Чтение запроса" targetRef="Снятие
денег"/>
    <transition event="e24" sourceRef="Чтение запроса" targetRef="Платеж"/>
    <transition event="e25" sourceRef="Авторизация" targetRef="Ответ
клиенту">
      <outputAction ident="o3.z1"/>

    </transition>
    <transition event="*" sourceRef="Ответ клиенту" targetRef="s2"/>
    <transition event="e25" sourceRef="Баланс" targetRef="Ответ клиенту"/>
    <transition sourceRef="s1" targetRef="Чтение запроса"/>
    <transition event="e25" sourceRef="Снятие денег" targetRef="Ответ
клиенту"/>
    <transition event="e25" sourceRef="Платеж" targetRef="Ответ клиенту"/>
  </stateMachine>
  <stateMachine name="Ahsm">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>

    <association clientRole="Ahsm" supplierRole="o4" targetRef="o4"/>
    <state name="Top" type="NORMAL">
      <state name="Чтение запроса" type="NORMAL">
        <outputAction ident="o4.z0"/>
      </state>
      <state name="Проверка" type="NORMAL">
        <outputAction ident="o4.z1"/>
      </state>
      <state name="Ответ серверу" type="NORMAL">

        <outputAction ident="o4.z2"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="s5" type="FINAL"/>
    </state>
    <transition event="*" sourceRef="Чтение запроса" targetRef="Проверка"/>
    <transition event="*" sourceRef="Проверка" targetRef="Ответ серверу"/>
    <transition event="*" sourceRef="Ответ серверу" targetRef="s5"/>
    <transition sourceRef="s1" targetRef="Чтение запроса"/>

  </stateMachine>
</model>
```

AServer.xml

```
<?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="client.FormPainter" name="o1"/>
  <controlledObject class="client.ServerQuery" name="o2"/>
  <controlledObject class="server.ServerReply" name="o3"/>
  <eventProvider class="client.HardwareEventProvider" name="p1">
    <association clientRole="p1" targetRef="AClient"/>
  </eventProvider>
  <eventProvider class="client.HumanEventProvider" name="p2">
    <association clientRole="p2" targetRef="AClient"/>
  </eventProvider>
  <eventProvider class="client.ServerEventProvider" name="p3">
    <association clientRole="p3" targetRef="AClient"/>
  </eventProvider>
  <eventProvider class="server.ClientEventProvider" name="p4">
    <association clientRole="p4" targetRef="AClient"/>
  </eventProvider>
  <rootStateMachine>
```

```
    <stateMachineRef name="AServer"/>
  </rootStateMachine>
  <stateMachine name="AClient">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="AClient" supplierRole="o1" targetRef="o1"/>
    <association clientRole="AClient" supplierRole="o2" targetRef="o2"/>
    <state name="Top" type="NORMAL">
      <state name="s2" type="FINAL"/>
      <state name="9. Запрос денег" type="NORMAL">
        <stateMachineRef name="AServer"/>
        <outputAction ident="o2.z9"/>
      </state>
      <state name="12. Печать чека 2" type="NORMAL">
        <outputAction ident="o1.z12"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="10. Выдача денег" type="NORMAL">
        <outputAction ident="o1.z10"/>
      </state>
      <state name="11. Печатать чек" type="NORMAL">
        <outputAction ident="o1.z11"/>
      </state>
      <state name="6. Вывод на экран" type="NORMAL">
        <outputAction ident="o1.z6"/>
      </state>
      <state name="7. Печать чека 1" type="NORMAL">
        <outputAction ident="o1.z7"/>
      </state>
      <state name="13. Возврат карты" type="NORMAL">
        <outputAction ident="o1.z13"/>
      </state>
      <state name="4. Главное меню" type="NORMAL">
        <outputAction ident="o1.z4"/>
      </state>
      <state name="8. Ввод суммы" type="NORMAL">
        <outputAction ident="o1.z8"/>
      </state>
      <state name="5. Запрос Баланса" type="NORMAL">
        <stateMachineRef name="AServer"/>
        <outputAction ident="o2.z5"/>
      </state>
      <state name="3. Авторизация" type="NORMAL">
        <stateMachineRef name="AServer"/>
        <outputAction ident="o2.z3"/>
      </state>
      <state name="2. Ввод pin кода" type="NORMAL">
        <outputAction ident="o1.z2"/>
      </state>
      <state name="1. Вставьте карту" type="NORMAL">
        <outputAction ident="o1.z1"/>
      </state>
    </state>
    <transition event="e13" sourceRef="9. Запрос денег" targetRef="10. Выдача
денег"/>
    <transition event="e14" sourceRef="9. Запрос денег" targetRef="13.
Возврат карты"/>
    <transition event="e8" sourceRef="12. Печать чека 2" targetRef="13.
Возврат карты"/>
    <transition sourceRef="s1" targetRef="1. Вставьте карту"/>
    <transition event="e9" sourceRef="10. Выдача денег" targetRef="11.
Печатать чек"/>
    <transition event="e3" sourceRef="11. Печатать чек" targetRef="12. Печать
чека 2"/>
```

```
    <transition event="e2" sourceRef="11. Печатать чек" targetRef="13.
Возврат карты"/>
    <transition event="e3" sourceRef="6. Вывод на экран" targetRef="7. Печать
чека 1"/>
    <transition event="e4" sourceRef="6. Вывод на экран" targetRef="13.
Возврат карты"/>
    <transition event="e8" sourceRef="7. Печать чека 1" targetRef="13.
Возврат карты"/>
    <transition event="e7" sourceRef="13. Возврат карты" targetRef="1.
Вставьте карту"/>
    <transition event="e2" sourceRef="4. Главное меню" targetRef="13. Возврат
карты"/>
    <transition event="e3" sourceRef="4. Главное меню" targetRef="5. Запрос
Баланса"/>
    <transition event="e4" sourceRef="4. Главное меню" targetRef="8. Ввод
суммы"/>
    <transition event="e4" sourceRef="8. Ввод суммы" targetRef="9. Запрос
денег"/>
    <transition event="e2" sourceRef="8. Ввод суммы" targetRef="13. Возврат
карты"/>
    <transition event="e12" sourceRef="5. Запрос Баланса" targetRef="6. Вывод
на экран"/>
    <transition event="e15" sourceRef="5. Запрос Баланса" targetRef="13.
Возврат карты"/>
    <transition event="e10" sourceRef="3. Авторизация" targetRef="4. Главное
меню"/>
    <transition event="e11" sourceRef="3. Авторизация" targetRef="13. Возврат
карты"/>
    <transition event="e4" sourceRef="2. Ввод pin кода" targetRef="3.
Авторизация"/>
    <transition event="e2" sourceRef="2. Ввод pin кода" targetRef="13.
Возврат карты"/>
    <transition event="e0" sourceRef="1. Вставьте карту" targetRef="s2">
      <outputAction ident="o1.z0"/>
    </transition>
    <transition event="e6" sourceRef="1. Вставьте карту" targetRef="2. Ввод
pin кода"/>
  </stateMachine>
  <stateMachine name="AServer">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="AServer" supplierRole="o3" targetRef="o3"/>
    <state name="Top" type="NORMAL">
      <state name="Ответ клиенту" type="NORMAL">
        <outputAction ident="o3.z4"/>
      </state>
      <state name="Снятие денег" type="NORMAL">
        <outputAction ident="o3.z3"/>
      </state>
      <state name="Чтение запроса" type="NORMAL">
        <outputAction ident="o3.z0"/>
      </state>
      <state name="Авторизация" type="NORMAL">
        <outputAction ident="o3.z1"/>
      </state>
      <state name="s2" type="FINAL"/>
      <state name="s1" type="INITIAL"/>
      <state name="Баланс" type="NORMAL">
        <outputAction ident="o3.z2"/>
      </state>
    </state>
    <transition event="*" sourceRef="Ответ клиенту" targetRef="s2"/>
    <transition event="e24" sourceRef="Снятие денег" targetRef="Ответ
клиенту"/>
```

```
    <transition event="e23" sourceRef="Чтение запроса" targetRef="Снятие
денег"/>
    <transition event="e21" sourceRef="Чтение запроса"
targetRef="Авторизация"/>
    <transition event="e22" sourceRef="Чтение запроса" targetRef="Баланс"/>
    <transition event="e24" sourceRef="Авторизация" targetRef="Ответ
клиенту"/>
    <transition sourceRef="s1" targetRef="Чтение запроса"/>
    <transition event="e24" sourceRef="Баланс" targetRef="Ответ клиенту"/>
  </stateMachine>
</model>
```

Приложение 2. Компилятивный подход. Исходные коды

ServerQuery.java

```
package client;

import callserver.ClientEventProvider;

import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.context.StateMachineContext;

public class ServerQuery implements ControlledObject {
    public static class ServerInfo{
            public static int balance;
            public static char command;
            public static int cardN;
            public static int cardPIN;
            public static int dispenseAmount;
            public static int phoneNumber;
            public static int PINTrials;
            public static void make(char com){
                    command = com;
                    try{
                            cardN =
Integer.parseInt(FormPainter.form.panelMain.edit_CardNumber.getText());
                    }catch(Exception e){
                            cardN = 0;
                    }
                    try{
                            cardPIN =
Integer.parseInt(FormPainter.form.panelMain.edit_Pin.getText());
                    }catch(Exception e){
                            cardPIN = 0;
                    }
                    try{
                            dispenseAmount =
Integer.parseInt(FormPainter.form.panelMain.edit_Sum.getText());
                    }catch(Exception e){
                            dispenseAmount = 0;
                    }
                    try{
                            phoneNumber =
Integer.parseInt(FormPainter.form.panelMain.edit_PhoneNumber.getText());
                    }catch(Exception e){
                            phoneNumber = 0;
                    }
            }
    }

    /**
     * @unimod.action.descr Количество попыток авторизации
     */
    public int x1(StateMachineContext context) {
            return ServerInfo.PINTrials;
    }

    /**
     * @unimod.action.descr Сброс количества попыток авторизации
     */
    public void z2(StateMachineContext context) {
            FormPainter.pinEnterMessage = "";
            ServerInfo.PINTrials = 0;
    }

    /**
     * @unimod.action.descr Авторизация на сервере
     */
    public void z3(StateMachineContext context) {
            FormPainter.form.panelMain.showState("", "auth", "");
            FormPainter.form.panelMain.markButton("empty", "empty", "empty", "empty");
            ServerInfo.make('A');
    }
```

```
	/**
	 * @unimod.action.descr Проверка количества попыток авторизации
	 */
	public void z5(StateMachineContext context) {
		ServerInfo.PINTrials++;
		FormPainter.goodbyeMessage = "goodbyePINLimit";
		FormPainter.pinEnterMessage = "wrongPIN";
		ClientEventProvider.notify(ClientEventProvider.E26);
	}

	/**
	 * @unimod.action.descr Запрос баланса
	 */
	public void z11(StateMachineContext context) {
		FormPainter.form.panelMain.showState("", "balanceInquiry", "");
		FormPainter.form.panelMain.markButton("empty", "empty", "empty", "empty");
		ServerInfo.make('B');
	}

	/**
	 * @unimod.action.descr Запрос проведения платежа
	 */
	public void z25(StateMachineContext context) {
		FormPainter.form.panelMain.showState("", "operating", "");
		FormPainter.form.panelMain.markButton("empty", "empty", "empty", "empty");
		ServerInfo.make('P');
	}

	/**
	 * @unimod.action.descr Запрос снятия денег
	 */
	public void z32(StateMachineContext context) {
		FormPainter.form.panelMain.showState("", "operating", "");
		FormPainter.form.panelMain.markButton("empty", "empty", "empty", "empty");
		ServerInfo.make('D');
	}
}
```

ServerReply.java

```
package callserver;

import hsm.HsmReply;

import java.net.MalformedURLException;
import java.rmi.*;


import client.FormPainter;
import client.ServerEventProvider;
import client.ServerQuery;
import server.Bank;

import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.context.StateMachineContext;

public class ServerReply implements ControlledObject {
	private char command;
	private int cardN;
	private int dispenseAmount;
	private int phoneNumber;
	private String reply;
	Bank bank;



	/**
	 * @unimod.action.descr Чтение запроса
	 */
	public void z0(StateMachineContext context) {
		command = ServerQuery.ServerInfo.command;
		cardN = ServerQuery.ServerInfo.cardN;
		dispenseAmount = ServerQuery.ServerInfo.dispenseAmount;
		phoneNumber = ServerQuery.ServerInfo.phoneNumber;
		switch(command){
		case 'A': ClientEventProvider.notify(ClientEventProvider.E21); break;
		case 'B': ClientEventProvider.notify(ClientEventProvider.E22); break;
		case 'D': ClientEventProvider.notify(ClientEventProvider.E23); break;
		case 'P': ClientEventProvider.notify(ClientEventProvider.E24); break;
		}
	}

	/**
	 * @unimod.action.descr Авторизация
	 */
	public void z1(StateMachineContext context) {

		try{
```

```
                bank = (Bank)Naming.lookup("rmi://localhost/bank");
        }catch(NotBoundException e){
                System.out.println("Not bound");
                e.printStackTrace();
        }catch(MalformedURLException e){
                System.out.println("MalformedURL");
                e.printStackTrace();
        }catch(RemoteException e){
                System.out.println("RemoteException");
                e.printStackTrace();
        };


        reply = HsmReply.reply;
    }

    /**
     * @unimod.action.descr Вернуть баланс
     */
    public void z2(StateMachineContext context) {
        try{
                ServerQuery.ServerInfo.balance = bank.getBalance(cardN);
                reply = ServerEventProvider.E12;
                FormPainter.endOperationMessage = "operEndOK";
        }catch(RemoteException e){
                reply = ServerEventProvider.E17;
                FormPainter.endOperationMessage = "operEndNoConnect";
        }
        ClientEventProvider.notify(ClientEventProvider.E25);
    }

    /**
     * @unimod.action.descr Снять деньги
     */
    public void z3(StateMachineContext context) {
        try{
                if(bank.dispenseMoney(cardN, dispenseAmount)){
                        reply = ServerEventProvider.E13;
                        FormPainter.endOperationMessage = "operEndOK";
                }
                else {
                        reply = ServerEventProvider.E14;
                        FormPainter.endOperationMessage = "operEndNoMoney";
                }
        }catch(RemoteException e){
                reply = ServerEventProvider.E17;
                FormPainter.endOperationMessage = "operEndNoConnect";
        }
        ClientEventProvider.notify(ClientEventProvider.E25);
    }

    /**
     * @unimod.action.descr Провести платеж
     */
    public void z4(StateMachineContext context) {
        try{
                int ans = bank.makePayment(cardN, dispenseAmount, phoneNumber);
                if (ans == -2){
                        reply = ServerEventProvider.E16;
                        FormPainter.endOperationMessage = "operEndNoPhoneNumber";

                }
                if (ans == -1){
                        reply = ServerEventProvider.E14;
                        FormPainter.endOperationMessage = "operEndNoMoney";
                }
                if (ans == 1){
                        reply = ServerEventProvider.E15;
                        FormPainter.endOperationMessage = "operEndOK";
                }
        }catch(RemoteException e){
                reply = ServerEventProvider.E17;
                FormPainter.endOperationMessage = "operEndNoConnect";

        }
        ClientEventProvider.notify(ClientEventProvider.E25);

    }
    /**
     * @unimod.action.descr Ответить клиенту
     */
    public void z5(StateMachineContext context) {
        ClientEventProvider.notify("temp");
        ClientEventProvider.notify(reply);
    }

}
```

HSMReply.java

```
package hsm;

import java.rmi.*;
import java.net.*;


import server.Bank;
import callserver.ClientEventProvider;
import client.FormPainter;
import client.HardwareEventProvider;
import client.ServerQuery;
import client.ServerEventProvider;


import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.context.StateMachineContext;

public class HsmReply implements ControlledObject {
    private int cardN;
    private int cardPIN;
    public static String reply;
    Bank bank;

    /**
     * @unimod.action.descr Разбор запроса
     */
    public void z0(StateMachineContext context) {
        cardN = ServerQuery.ServerInfo.cardN;
        cardPIN = ServerQuery.ServerInfo.cardPIN;
    }

    /**
     * @unimod.action.descr Проверка PIN
     */
    public void z1(StateMachineContext context) {
        try{
            bank = (Bank)Naming.lookup("rmi://localhost/bank");
        }catch(NotBoundException e){
            System.out.println("Not bound");
            e.printStackTrace();
        }catch(MalformedURLException e){
            System.out.println("MalformedURL");
            e.printStackTrace();
        }catch(RemoteException e){
            System.out.println("RemoteException");
            e.printStackTrace();
        };

        try{
            int ans = bank.checkCard(cardN, cardPIN);
            if (ans == 1) {
                reply = ServerEventProvider.E10;
                FormPainter.goodbyeMessage = "goodbyeOK";
            }
            if (ans == -1){
                reply = ServerEventProvider.E11;
            }
            if (ans == -2){
                reply = HardwareEventProvider.E7;
                FormPainter.goodbyeMessage = "goodbyeWrongCard";
            }

        }catch(RemoteException e){
            reply = ServerEventProvider.E17;
        }
        ClientEventProvider.notify(ClientEventProvider.E20);
    }

    /**
     * @unimod.action.descr Отправка ответа
     */
    public void z2(StateMachineContext context) {
        ClientEventProvider.notify(ClientEventProvider.E20);
        ClientEventProvider.notify(ClientEventProvider.E25);
    }
}
```

FormPainter.java

```
package client;


import com.evelopers.unimod.runtime.ControlledObject;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.StateMachineContext;
```

```
public class FormPainter implements ControlledObject {
     public static TForm form;
     public static String endOperationMessage = "operEndCancel";
     public static String goodbyeMessage = "goodbyeOK";
     public static String pinEnterMessage = "";

     public static void init(ModelEngine engine){
            form = new TForm(engine);
     }

     /**
      * @unimod.action.descr Выключение
      */
     public void z0(StateMachineContext context) {
            form.dispose();
     }

     /**
      * @unimod.action.descr Приветствие
      */
     public void z1(StateMachineContext context) {
            form.panelMain.showState("welcome", "insertCard", "");
         form.panelMain.markButton("empty", "empty", "empty", "empty");
     }

     /**
      * @unimod.action.descr Ввод pin кода
      */
     public void z2(StateMachineContext context) {
            form.panelMain.edit_Pin.setText("");
            form.panelMain.showState("enter pin", "enterPin", pinEnterMessage);
            form.panelMain.markButton("empty", "btncancel", "empty", "btnok");
     }

     /**
      * @unimod.action.descr Главное меню
      */
     public void z4(StateMachineContext context) {
            form.panelMain.showState("main menu", "mainMenu", "");
            form.panelMain.markButton("btnpayment", "btnbackcard", "btnbalance", "btnmoney");
     }

     /**
      * @unimod.action.descr Вывод баланса на экран
      */
     public void z12(StateMachineContext context) {
            form.panelMain.showState("balance", "amountAvailable",
Integer.toString(ServerQuery.ServerInfo.balance));//todo
            form.panelMain.markButton("empty", "empty", "btnprint", "btnnoreceipt");
     }

     /**
      * @unimod.action.descr Печать баланса на чеке
      */
     public void z13(StateMachineContext context) {
            form.panelMain.showState("print", "takeReceipt", "");
            form.panelMain.markButton("empty", "empty", "empty", "empty");
     }

     /**
      * @unimod.action.descr Выбор оператора услуг связи
      */
     public void z21(StateMachineContext context) {
            form.panelMain.showState("back card", "selectOperator", "");
            form.panelMain.markButton("empty", "btncancel", "btnmegafon", "btnmts");
     }

     /**
      * @unimod.action.descr Экран ввода номера
      */
     public void z22(StateMachineContext context) {
            form.panelMain.showState("enter phoneNumber", "enterPhoneNumber", "");
            form.panelMain.markButton("empty", "btncancel", "empty", "btnok");
     }

     /**
      * @unimod.action.descr Ввод суммы
      */
     public void z23(StateMachineContext context) {
            form.panelMain.showState("enter sum", "enterSum", "");
            form.panelMain.markButton("empty", "btncancel", "empty", "btnok");
     }

     /**
      * @unimod.action.descr Проверка введенных данных
      */
     public void z24(StateMachineContext context) {
            form.panelMain.showState("check payment", "checkPhonePaymentInfo", "");
            /*TODO:  btnchange*/
            form.panelMain.markButton("empty", "btnchange", "empty", "btnok");
```

```
	}

	/**
	 * @unimod.action.descr Печать чека об оплате телефона
	 */
	public void z26(StateMachineContext context) {
		form.panelMain.showState("print", "takeReceipt", "");
		form.panelMain.markButton("empty", "empty", "empty", "empty");
	}

	/**
	 * @unimod.action.descr Ввод суммы
	 */
	public void z31(StateMachineContext context) {
		form.panelMain.showState("enter sum", "enterSum", "");
		form.panelMain.markButton("empty", "btncancel", "empty", "btnok");
	}

	/**
	 * @unimod.action.descr Выдача денег
	 */
	public void z33(StateMachineContext context) {
		form.panelMain.showState("money out", "takeMoney", "");
		form.panelMain.markButton("empty", "empty", "empty", "empty");
	}

	/**
	 * @unimod.action.descr Печатать ли чек
	 */
	public void z34(StateMachineContext context) {
		form.panelMain.showState("", "askReceipt", "");
		form.panelMain.markButton("empty", "empty", "btnprint", "btnnoreceipt");
	}

	/**
	 * @unimod.action.descr Печать чека по операции
	 */
	public void z35(StateMachineContext context) {
		form.panelMain.showState("print", "takeReceipt", "");
		form.panelMain.markButton("empty", "empty", "empty", "empty");
	}

	/**
	 * @unimod.action.descr Завершение операции
	 */
	public void z40(StateMachineContext context) {
		form.panelMain.showState("", endOperationMessage, "");
		form.panelMain.markButton("empty", "btnbackcard", "empty", "btnok");
	}

	/**
	 * @unimod.action.descr Возврат карты
	 */
	public void z50(StateMachineContext context) {
		form.panelMain.showState("card return", "takeCard", goodbyeMessage);
		form.panelMain.markButton("empty", "empty", "empty", "empty");
	}

	/**
	 * @unimod.action.descr Очистка
	 */
	public void z51(StateMachineContext context) {
		FormPainter.form.panelMain.edit_CardNumber.setText("");
		FormPainter.form.panelMain.edit_Pin.setText("");
		FormPainter.form.panelMain.edit_PhoneNumber.setText("");
		FormPainter.form.panelMain.edit_Sum.setText("");
	}

}
```