Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

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

А. Ю. Законов, А. А. Клебанов

Моделирование устройства для обмена валюты

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

Санкт-Петербург

2007

Оглавление

Введение

Цель работы — изучение автоматного подхода к программированию с использованием инструментального средства *UniMod*. В качестве примера рассматривается программная реализация устройства для обмена валюты. Для систем подобного типа требуется обеспечить надежность взаимодействия ядра, в котором инкапсулирована логика функционирования устройства, с пользовательским интерфейсом. При автоматном подходе основной акцент делается на проектирование поведения программы — создание системы конечных автоматов, соответствующей предметной области и спецификации задачи, «поведение которой формализуется с помощью системы взаимосвязанных графов переходов» [1]. Это позволяет уже на ранних этапах разработки устранить неоднозначность в функционировании и получить представление о работе приложения при различных конфигурациях входных параметров. Поясним эту идею, основываясь на работе [2].

Вне зависимости от используемой парадигмы программирования, приложение в любой момент времени находится в том или ином состоянии, определяемом значениями всех данных. Описанную модель можно упростить, если рассматривать только бинарные переменные (флаги), участвующие в процессах передачи управления, а, следовательно, определяющие переходы между состояниями. Тем не менее, n флагам соответствует 2^n состояний программы — это число растет экспоненциально при увеличении n. Таким образом, некоторые из них могут оказаться непредусмотренными. Для решения описанной проблемы в работе [2] предложена следующая идея: «Для устранения самой возможности возникновения в программе непредусмотренных состояний следует еще на этапе проектирования явно определять все требуемые состояния и применять для их различения только одну многозначную управляющую переменную. После этого необходимо явно определить все возможные переходы между состояниями и построить программу так, чтобы она не могла сойти с проложенных «рельс».

1. Постановка задачи

Устройство для обмена валюты позволяет автоматизировать работу с банком. При этом необходимо реализовать пользовательский интерфейс, модель внутренних механизмов, отвечающую за функционирование обменного устройства, и серверную часть, которая будет эмулировать банк: следить за корректностью производимых операций и предоставлять информацию о текущих курсах валют.

На (рис. 1) изображена лицевая панель устройства в состоянии готовности к использованию, а на (рис. 2) — при запуске.

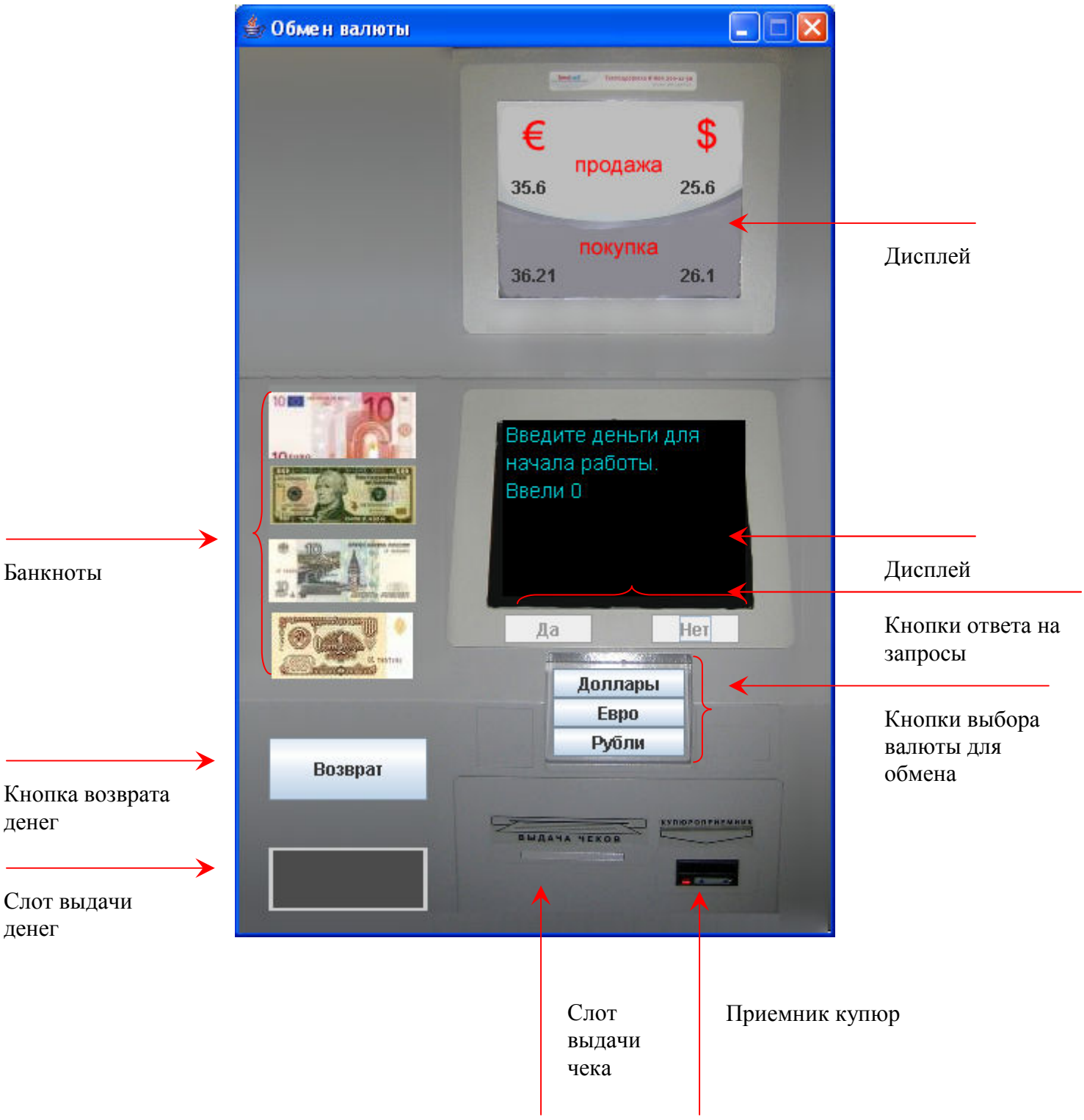

Рис. 1. Внешний вид приложения при готовности к использованию

Перечислим основные составляющие лицевой панели и опишем их функциональные особенности:

- два дисплея применяются для отображения текущих курсов и вывода сообщений;
- кнопка «Да» и кнопка «Нет» — для ответа пользователем на запросы автомата;
- кнопки «Доллары», «Евро», «Рубли» используются для выбора соответствующей валюты, подлежащей обмену;
- набор банкнот, приемник купюр — для эмуляции ввода денег клиентом;
- кнопка «Возврат» позволяет забрать введенную сумму.

Рис. 2. Внешний вид приложения при запуске

Теперь приведем неформальное описание работы приложения в целом.

В начале работы обменное устройство пытается авторизироваться на сервере банка и получить текущие курсы валют. Если обнаружена ошибка, то работа прекращается. Для получения прибыли за каждую операцию взимается комиссия, уведомление о которой выдается на дисплей.

Клиент может, как согласиться, так и отказаться от ее оплаты. В первом случае процедура обмена будет продолжена, а во втором — завершена. Выбор исходной валюты происходит автоматически при получении первой банкноты, которая не является фальшивой. Далее сумма увеличивается лишь в том случае, если устройство установило допустимость очередной купюры. Это возможно в двух случаях. Во-первых, банкнота не является фальшивой, а во-вторых, совпадает по «типу» с предыдущей (например, если до этого вводились доллары, то автомат допустит доллары, а евро или рубли вернет).

В зависимости от введенной суммы устройство выбирает стандартный или льготный курс. При условии наличия достаточного количества банкнот, производится обмен и выводится запрос о печати чека. В противном случае, банку посылается сообщение об

окончании определенного вида валюты. При этом клиент может либо забрать деньги, либо выбрать другую доступную валюту.

2. Проектирование

Выделим основные составляющие проекта. Согласно спецификации для поставленной задачи ими являются:

- пользовательский интерфейс;
- устройство обмена;
- банк (сервер);
- система управления.

Интерфейс обрабатывает воздействия пользователя и генерирует события, которые передаются системе управления. Она, в свою очередь, изменяет состояние устройства обмена. Заметим, что выполнение некоторых операций может потребовать контроля со стороны банка, а другие методы могут инициировать, например, вывод сообщений или печать чека. Таким образом, связь между интерфейсом, устройством обмена и сервером банка с одной стороны и системой управления с другой, оказывается двусторонней (рис. 3).

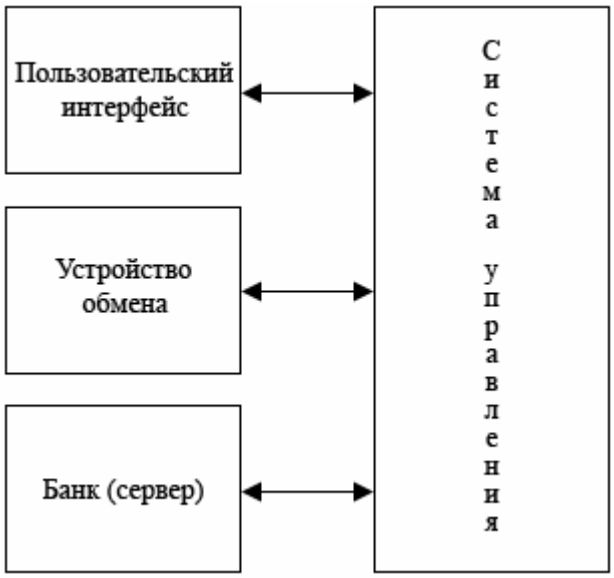

Рис. 3. Схема взаимодействия составляющих проекта

Построим более подробную схему взаимодействия (рис. 4).

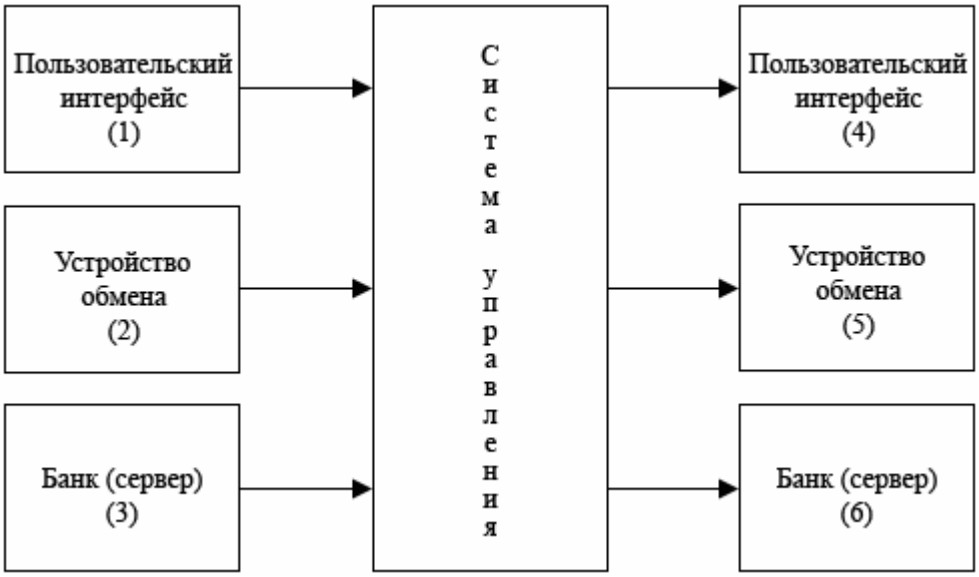

Рис. 4. Подробная схема взаимодействия

Перечислим функции, соответсвующие номерам на этой схеме.

1. Обработка нажатий на кнопки.
2. Посылка запросов серверу банка и информирование о результатах проведения операций.
3. Посылка ответов на запросы, полученные от устройства.
4. Отображение информационных сообщений на дисплеях.
5. Управление внутренними системами после обработки ответов сервера банка и воздействий пользователя.
6. Распознавание запросов, полученных от устройства.

Поведение системы управления реализовано с помощью конечных автоматов.

2.1. Схема связей

В работе [1] схема связей определяется следующим образом: «связи каждого автомата с его «окружением» формализуются схемой связей автомата, предназначенной для полного описания интерфейса автоматов».

Схема связей, объединенная со схемой взаимодействия автоматов (рис. 5), состоит из объектов трех видов — поставщиков событий, автоматов и объектов управления. Для их обозначения используется следующая нотация (здесь и далее вместо символа «*#*» стоит число):

- *p#* — поставщик событий;
- *e#* — событие;
- *o#* — объект управления;
- *A#* — автомат;
- *z#* — выходные воздействия;
- *x#* — входные переменные.

Шаблон взаимодействия элементов схемы связи выглядит следующим образом. Поставщик событий генерирует событие, которое обрабатывается одним из автоматов. При этом проверяются различные логические условия (если они имеются). После этого выбирается переход в новое состояние. С каждым переходом может быть ассоциирован некий набор выходных воздействий на объекты управления, выполняемый при выборе перехода. Заметим, что в общем случае действия могут выполняться не только на переходах, но и в вершинах.

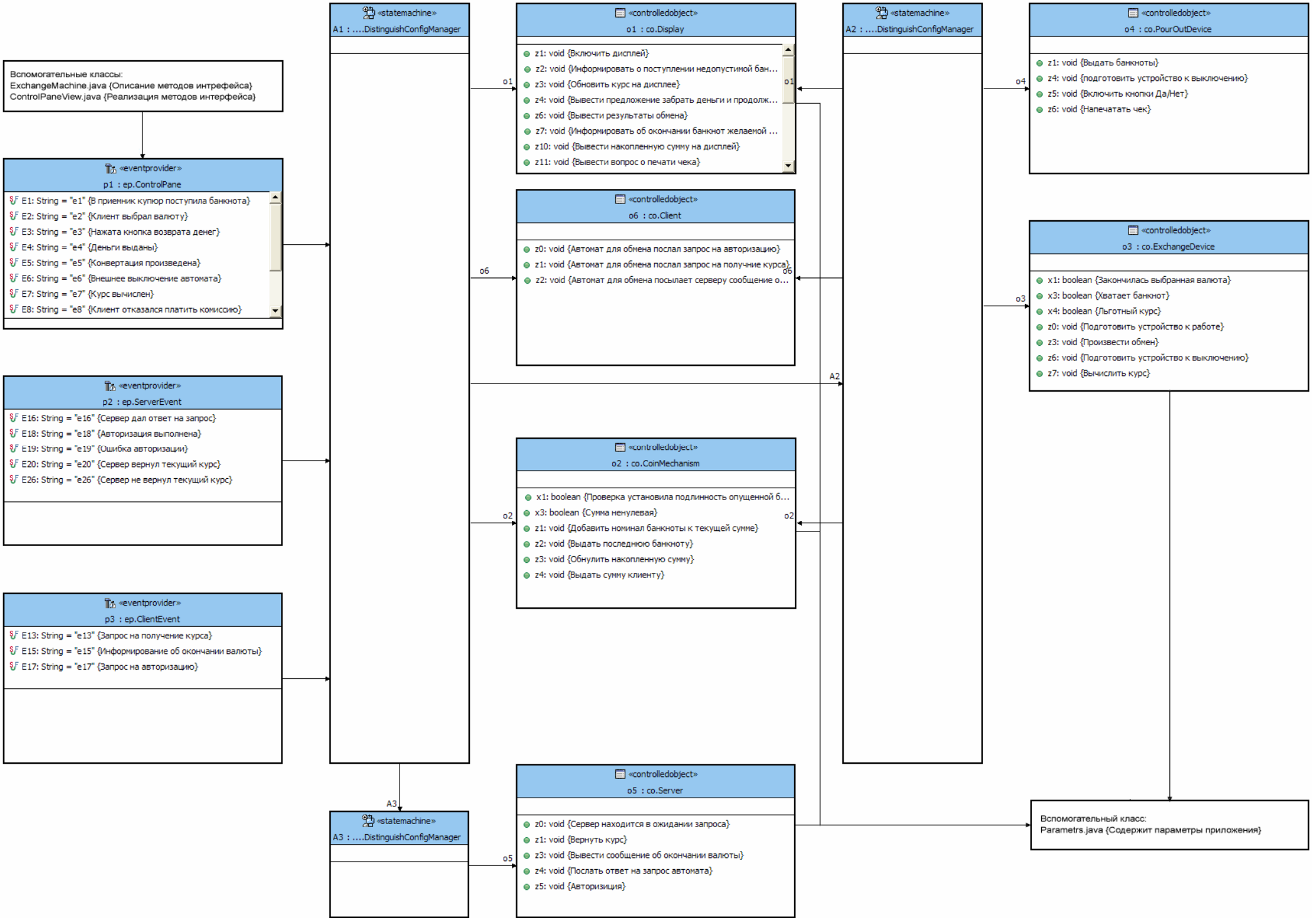

Рис. 5. Схема связей

Рассмотрим элементы схемы связей.

В целях логического разделения функциональных возможностей системы создано три автомата:

- *A1* — управляет подготовкой устройства, которая включает авторизацию устройства на сервере банка, получение текущего курса и прием денег от клиента;
- *A2* — контролирует внутренние системы во время операции обмена, обеспечивает диалог с пользователем при выборе валюты и предложении напечатать чек;
- *A3*— эмулирует описанную работу сервера.

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

Таблица 1. Описание классов

Название класса	**Связь с *UniMod***	**Описание класса**
`ControlPane.java`	Поставщик событий *p1*	Описывает все события, которые генерирует устройство
`ServerEvent.java`	Поставщик событий *p2*	Описывает события, которые генерирует сервер банка при ответе на запросы
`ClientEvent.java`	Поставщик событий *p3*	Описывает события, генерируемые при посылке клиентом запроса серверу банка
`Display.java`	Объект управления *o1*	Отвечает за вывод сообщений на дисплей и включение-выключение индикаторов выбора валюты для обмена
`CoinMechanism.java`	Объект управления *o2*	Хранит информацию о накопленной сумме, а также выполняет проверку допустимости банкнот
`ExchangeDevice.java`	Объект управления *o3*	Выбирает курс, проверяет наличие банкнот, а затем производит обмен
`PourOutDevice.java`	Объект управления *o4*	Выдает деньги пользователю и печатает чек
`Server.java`	Объект управления *o5*	Отвечает на запросы автомата: производит авторизацию и возвращает курс валют
`Client.java`	Объект управления *o6*	Предназначен для отправки запросов и сообщений серверу банка
`ExchangeMachine.java`	Вспомогательный класс	Описывает методы устройства для обмена валюты
`ControlPaneView.java`	Вспомогательный класс	Реализует интерфейс пользователя
`Parameters.java`	Вспомогательный класс	Содержит параметры приложения

Вспомогательные классы выделены для удобства, так как, в принципе, их можно не выделять отдельно.

Поставщиками событий в данном случае являются: пользователь и внутренние составляющие устройства обмена (табл. 2), устройство обмена, представляющее запросы серверу банка (табл. 3), и сервер (банк) (табл. 4).

Таблица 2. События, формируемые пользователем и внутренними составляющими устройства обмена

Событие	**Описание**
e1	Клиент опустил купюру в приемник банкнот
e2	Клиент выбрал валюту для обмена
e3	Клиент нажал кнопку возврата денег
e4	Деньги выданы клиенту
e5	Конвертация произведена
e6	Внешнее выключение автомата — пользователь выключил программу, моделирующую работу устройства
e7	Курс выбран — в зависимости от суммы выбран либо стандартный, либо льготный курс
e8	Клиент отказался платить комиссию
e9	Клиент согласился платить комиссию
e10	Клиент согласился напечатать чек
e11	Клиент отказался от печати чека
e21	Работа приложения завершена

Таблица 3. События, формируемые устройством обмена

Событие	**Описание**
e13	Запрос на получение курса
e15	Информирование об окончании валюты
e17	Запрос на авторизацию

Таблица 4. События, формируемые банком

Событие	**Описание**
e16	Сервер банка дал ответ на запрос
e18	Авторизация выполнена
e19	Ошибка авторизации
e20	Сервер банка вернул текущий курс
e26	Сервер банка не вернул текущий курс

2.2. Графы переходов автоматов

Начальное состояние автомата обозначено черным кругом, а конечное — двойным кругом с закрашенной серединой. Каждый автомат имеет одно начальное и произвольное количество 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 выполняется следующее:

1. Выходных воздействия, ассоциированные с переходом.
2. Выходные воздействия, предусмотренные для исполнения в момент входа в состояние.
3. Управление передается вложенному автомату *A#*.

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

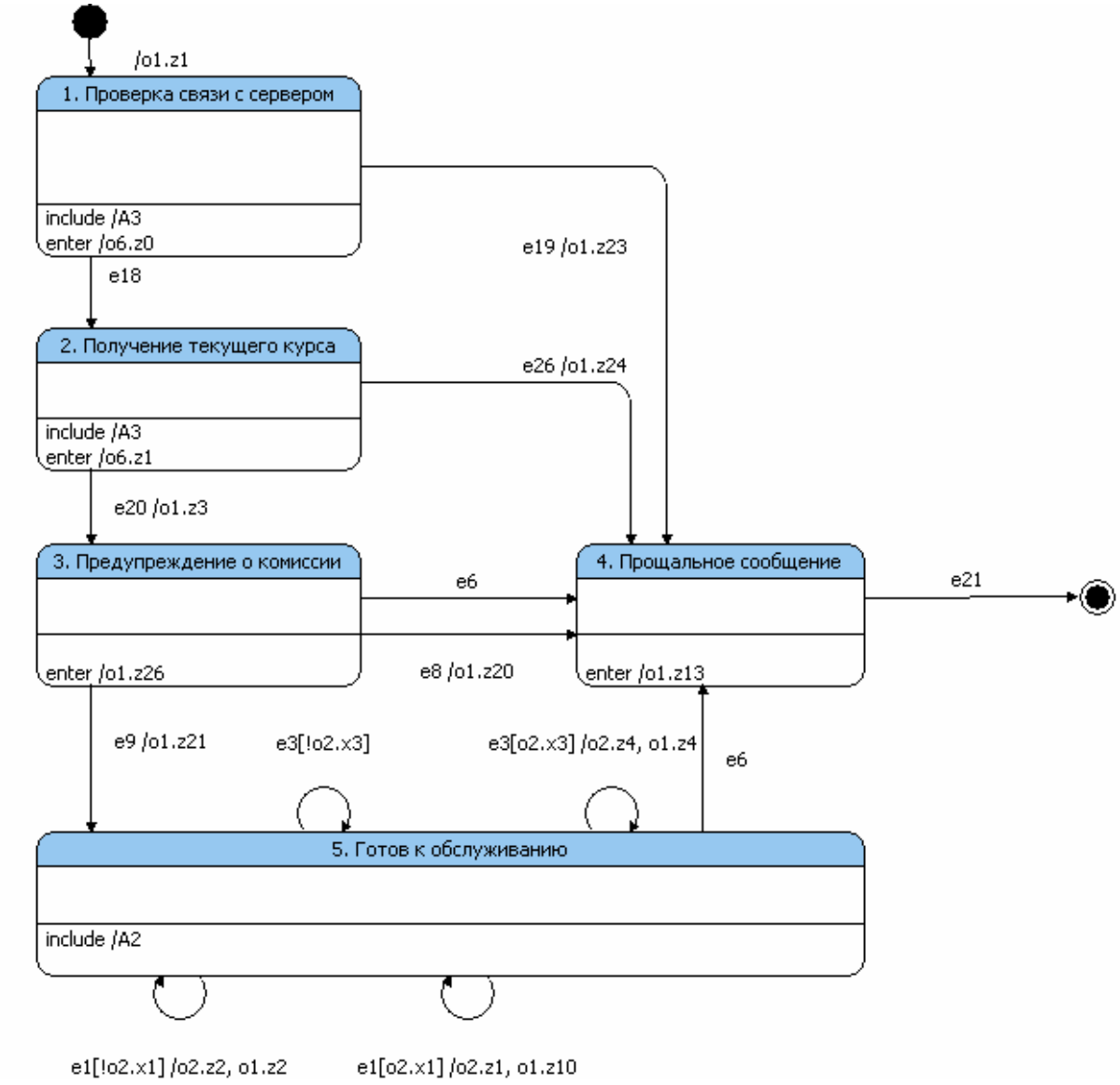

Рис. 6. Автомат *A1*

Автомат *A1* может находиться в одном из пяти состояний.

1. *Проверка связи с сервером* — устройство для обмена предпринимает попытку авторизироваться на сервере банка. При этом используется вложенный автомат *A3*, который распознает запрос и выдает ответ. В случае успешной авторизации автомат *A1* переходит в состояние 2, а в случае неудачи — в состояние 4.
2. *Получение текущего курса* — устройство получает текущие курсы всех валют. Поведение в этом состоянии аналогично поведению в состоянии 1. Разница состоит лишь в выдаваемых сообщениях при переходе в состояние 4.
3. *Предупреждение о комиссии* — клиент уведомляется о том, что для продолжения работы ему необходимо согласиться заплатить комиссию, равную одному проценту от результата операции. В случае отказа, автомат переходит в состояние 4. Поскольку в данном состоянии требуется ответ клиента, то во избежание «зависания» программы следует предусмотреть ситуацию, когда он захочет закрыть приложение. Для этого создается еще один переход в состояние 4, но уже по событию *e6*. Далее переход из какого-либо состояния в состояние 4 по событию *e6* будет иметь описанный смысл.
4. *Прощальное сообщение* — на дисплей выдается прощальное сообщение. После этого приложение заканчивает работу.
5. *Готов к обслуживанию* — завершение выполнения всех действий на петлях (на которых осуществляется проверка допустимости опущенной купюры и обработка нажатия кнопки возврата денег) означает окончание ввода денег клиентом и передачу управления автомату *A2*.

На диаграмме (рис. 7) приведен граф переходов автомата *A2*.

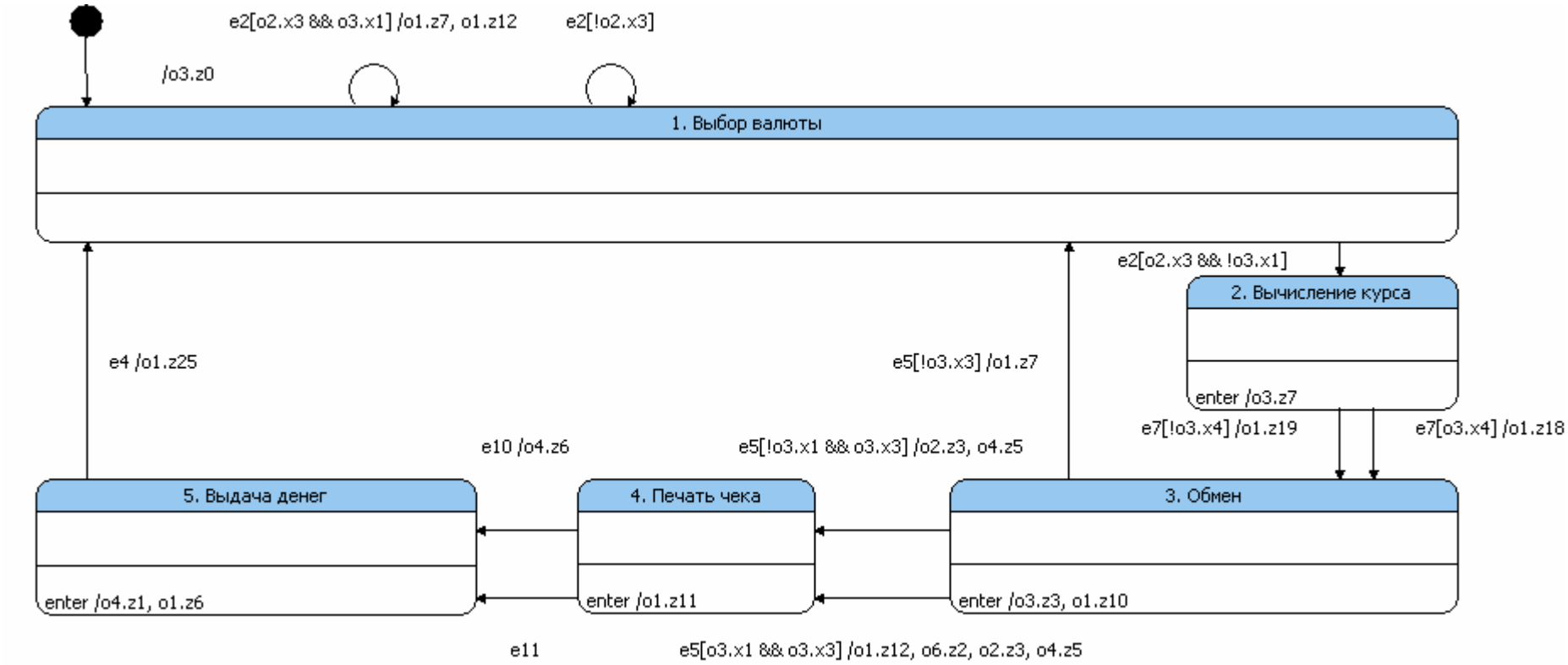

Рис. 7. Автомат *A2*

Автомат *A2* может находиться в одном из пяти состояний.

1. *Выбор валюты* — на петлях устройство проверяет все необходимые для начала операции условия: клиент выбрал валюту, сумма для обмена не равна нулю и выбранная валюта не закончилась. После этого автомат переходит в состояние 2.
2. *Вычисление курса* — в зависимости от суммы устройство выбирает либо стандартный, либо льготный курс.
3. *Обмен* — в этом состоянии производится обмен. Если достаточно желаемой валюты, то накопленная сумма обнуляется, а ее номинал добавляется к количеству валюты этого «типа» в обменном устройстве. При этом клиенту выдается запрос о печати чека (переход в состояние 4). В противном случае клиент информируется о недостаточном количестве валюты, и автомат возвращается в состояние 1. Для достижения корректности функционирования устройства в случае, когда после конвертации заканчивается валюта, на сервер банка посылается сообщение об окончании купюр. Соответствующая кнопка выбора валюты отключается до тех пор, пока в устройство не поступят купюры данного типа.
4. *Печать чека* — устройство предлагает клиенту напечатать чек. Вне зависимости от решения клиента осуществляется переход в состояние 5.
5. *Выдача денег* — клиент может забрать свои деньги из соответствующего слота. Автомат переходит в состояние 1.

Приведем граф переходов (рис. 8) автомата *A3*.

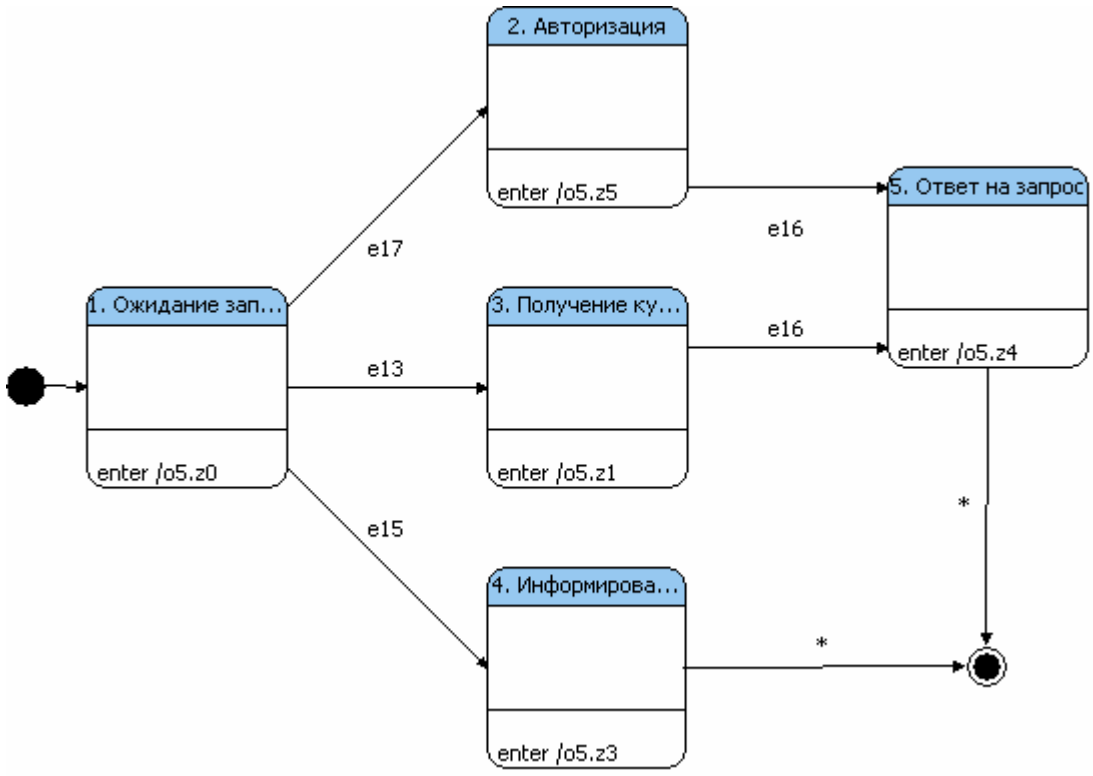

Рис. 8. Автомат *A3*

Автомат *A3* может находиться в одном из пяти состояний.

1. *Ожидание запроса* — сервер банка ожидает сообщение и распознает, какой запрос он получил от устройства для обмена.
2. *Авторизация* — устройство послало запрос на авторизацию. В этом случае сервер банка пытается найти идентификационный номер устройства в своей базе и, получив результат, переводит автомат в состояние 5.

3. *Получение курса* — устройство послало запрос на получение текущего курса валют. Сервер банка либо скачивает его из Интернета, либо считывает из файла. Получив курс, он переводит автомат в состояние 5.
4. *Информирование об окончании валюты* — сервер банка получил сообщение об окончании валюты. Автомат переходит в заключительное состояние.
5. *Ответ на запрос* — сервер банка обработал запрос и отправил ответ клиенту. После этого работа автомата завершается.

3. Реализация

Программа реализуется с помощью плагина *UniMod* к среде разработки *Java*-приложений *Eclipse*. Указанный плагин позволяет построить схему связей, а также графы переходов (рис. 4–6). После этого вручную реализуются объекты управления, источники событий и интерфейс программы на языке *Java*, а также выделенные для удобства реализации три вспомогательных класса.

Существует два подхода для создания приложения: *интерпретационный* и *компилятивный*.

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.

В Приложении 1 приведен пример протокола работы программы, в Приложении 2 — сгенерированное *XML*-описание, а в Приложении 3 — сгенерированный по *XML*-описанию код на языке *Java*.

3.2. Компиляционный подход

Альтернативой интерпретационному подходу является компиляционный подход. 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].

Структурные схемы интерпретационного и компиляционного подходов приведены в статье [4].

Заключение

Выбор автоматного подхода оказался полностью оправданным. Концентрируя основное внимание на архитектуре программы, а не на ее реализации, удалось достичь требуемой функциональности за короткий срок, так как средство *Unimod*, во-первых,

проверяет автоматы на корректность, а во-вторых, генерирует большую часть кода автоматически (рис. 9).

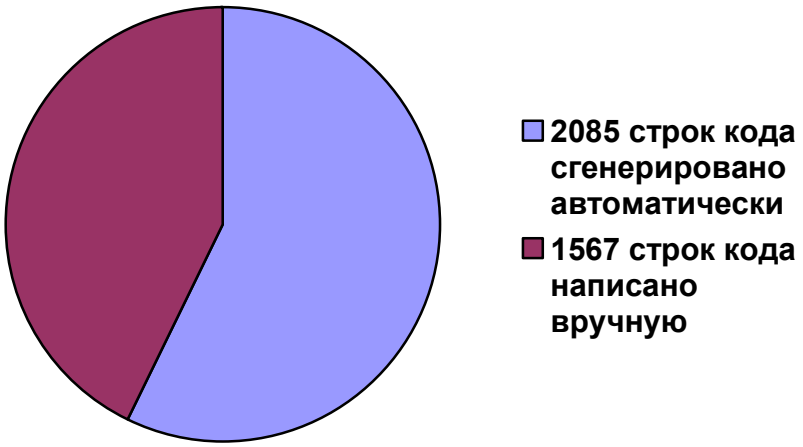

Рис. 9. Сравнение количества строк кода на языке *Java*, написанных автоматически и сгенерированных по *UML*-диаграммам

Возможным путем развития проекта авторы видят добавление функций для работы с Интернет-деньгами.

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

Приложение 1. Пример протокола работы программы

```
16:05:34,828 INFO  [Run] Start event [E12] processing. In state [/A1:Top]
16:05:34,828 INFO  [Run] Transition to go found [0. Начальное состояние#1.
Проверка связи с сервером##true]
16:05:34,828 INFO  [Run] Start output action [o1.z1] execution
Включение дисплея. Установка связи с сервером.
16:05:35,828 INFO  [Run] Finish output action [o1.z1] execution
16:05:35,828 INFO  [Run] Start on-enter action [o6.z0] execution
Автомат для обмена послал запрос на авторизацию.
16:05:35,828 INFO  [Run] Finish on-enter action [o6.z0] execution
16:05:35,828 INFO  [Run] Start event [E12] processing. In state [/A1:1.
Проверка связи с сервером/A3:Top]
16:05:35,828 INFO  [Run] Transition to go found [s1#1. Ожидание
запроса##true]
16:05:35,828 INFO  [Run] Start on-enter action [o5.z0] execution
Сервер находится в ожидании запроса.
16:05:35,828 INFO  [Run] Finish on-enter action [o5.z0] execution
16:05:35,828 INFO  [Run] Finish event [E12] processing. In state [/A1:1.
Проверка связи с сервером/A3:1. Ожидание запроса]
16:05:35,828 INFO  [Run] Finish event [E12] processing. In state [/A1:1.
Проверка связи с сервером]
16:05:35,843 INFO  [Run] Start event [e17] processing. In state [/A1:1.
Проверка связи с сервером]
16:05:35,843 INFO  [Run] Start event [e17] processing. In state [/A1:1.
Проверка связи с сервером/A3:1. Ожидание запроса]
16:05:35,843 DEBUG [Run] Try transition [1. Ожидание запроса#2.
Авторизация#e17#true]
16:05:35,843 INFO  [Run] Transition to go found [1. Ожидание запроса#2.
Авторизация#e17#true]
16:05:35,843 INFO  [Run] Start on-enter action [o5.z5] execution
Авторизация на сервере прошла успешно.
16:05:35,843 INFO  [Run] Finish on-enter action [o5.z5] execution
16:05:35,843 INFO  [Run] Finish event [e17] processing. In state [/A1:1.
Проверка связи с сервером/A3:2. Авторизация]
16:05:35,843 INFO  [Run] Finish event [e17] processing. In state [/A1:1.
Проверка связи с сервером]
16:05:35,843 INFO  [Run] Start event [e16] processing. In state [/A1:1.
Проверка связи с сервером]
16:05:35,843 INFO  [Run] Start event [e16] processing. In state [/A1:1.
Проверка связи с сервером/A3:2. Авторизация]
16:05:35,843 DEBUG [Run] Try transition [2. Авторизация#5. Ответ на
запрос#e16#true]
16:05:35,843 INFO  [Run] Transition to go found [2. Авторизация#5. Ответ на
запрос#e16#true]
16:05:35,843 INFO  [Run] Start on-enter action [o5.z4] execution
Сервер выслал ответ на запрос автомата.
16:05:35,843 INFO  [Run] Finish on-enter action [o5.z4] execution
16:05:35,843 INFO  [Run] Finish event [e16] processing. In state [/A1:1.
Проверка связи с сервером/A3:5. Ответ на запрос]
16:05:35,843 INFO  [Run] Finish event [e16] processing. In state [/A1:1.
Проверка связи с сервером]
16:05:35,843 INFO  [Run] Start event [e18] processing. In state [/A1:1.
Проверка связи с сервером]
16:05:35,843 DEBUG [Run] Try transition [1. Проверка связи с сервером#2.
Получение текущего курса#e18#true]
16:05:35,843 INFO  [Run] Transition to go found [1. Проверка связи с
сервером#2. Получение текущего курса#e18#true]
16:05:35,843 INFO  [Run] Start on-enter action [o6.z1] execution
Автомат для обмена послал запрос на получние курса.
16:05:35,843 INFO  [Run] Finish on-enter action [o6.z1] execution
```

```
16:05:35,843 INFO  [Run] Start event [e18] processing. In state [/A1:2.
Получение текущего курса/A3:Top]
16:05:35,843 INFO  [Run] Transition to go found [s1#1. Ожидание
запроса##true]
16:05:35,843 INFO  [Run] Start on-enter action [o5.z0] execution
Сервер находится в ожидании запроса.
16:05:35,843 INFO  [Run] Finish on-enter action [o5.z0] execution
16:05:35,843 INFO  [Run] Finish event [e18] processing. In state [/A1:2.
Получение текущего курса/A3:1. Ожидание запроса]
16:05:35,843 INFO  [Run] Finish event [e18] processing. In state [/A1:2.
Получение текущего курса]
16:05:35,843 INFO  [Run] Start event [e13] processing. In state [/A1:2.
Получение текущего курса]
16:05:35,843 INFO  [Run] Start event [e13] processing. In state [/A1:2.
Получение текущего курса/A3:1. Ожидание запроса]
16:05:35,843 DEBUG [Run] Try transition [1. Ожидание запроса#3. Получение
курса#e13#true]
16:05:35,843 INFO  [Run] Transition to go found [1. Ожидание запроса#3.
Получение курса#e13#true]
16:05:35,843 INFO  [Run] Start on-enter action [o5.z1] execution
Попытка получения курса.
Курс получен.
16:05:35,843 INFO  [Run] Finish on-enter action [o5.z1] execution
16:05:35,843 INFO  [Run] Finish event [e13] processing. In state [/A1:2.
Получение текущего курса/A3:3. Получение курса]
16:05:35,843 INFO  [Run] Finish event [e13] processing. In state [/A1:2.
Получение текущего курса]
16:05:35,843 INFO  [Run] Start event [e16] processing. In state [/A1:2.
Получение текущего курса]
16:05:35,843 INFO  [Run] Start event [e16] processing. In state [/A1:2.
Получение текущего курса/A3:3. Получение курса]
16:05:35,843 DEBUG [Run] Try transition [3. Получение курса#5. Ответ на
запрос#e16#true]
16:05:35,843 INFO  [Run] Transition to go found [3. Получение курса#5. Ответ
на запрос#e16#true]
16:05:35,843 INFO  [Run] Start on-enter action [o5.z4] execution
Сервер выслал ответ на запрос автомата.
16:05:35,843 INFO  [Run] Finish on-enter action [o5.z4] execution
16:05:35,843 INFO  [Run] Finish event [e16] processing. In state [/A1:2.
Получение текущего курса/A3:5. Ответ на запрос]
16:05:35,843 INFO  [Run] Finish event [e16] processing. In state [/A1:2.
Получение текущего курса]
16:05:35,843 INFO  [Run] Start event [e20] processing. In state [/A1:2.
Получение текущего курса]
16:05:35,843 DEBUG [Run] Try transition [2. Получение текущего курса#3.
Предупреждение о комиссии#e20#true]
16:05:35,843 INFO  [Run] Transition to go found [2. Получение текущего
курса#3. Предупреждение о комиссии#e20#true]
16:05:35,843 INFO  [Run] Start output action [o1.z3] execution
Обновлен курс на экране.
16:05:35,843 INFO  [Run] Finish output action [o1.z3] execution
16:05:35,843 INFO  [Run] Start on-enter action [o1.z26] execution
Задан вопрос о комиссии.
16:05:36,843 INFO  [Run] Finish on-enter action [o1.z26] execution
16:05:36,843 INFO  [Run] Finish event [e20] processing. In state [/A1:3.
Предупреждение о комиссии]
comission
16:05:37,234 INFO  [Run] Start event [e9] processing. In state [/A1:3.
Предупреждение о комиссии]
16:05:37,234 DEBUG [Run] Try transition [3. Предупреждение о комиссии#5.
Готов к обслуживанию#e9#true]
16:05:37,234 INFO  [Run] Transition to go found [3. Предупреждение о
комиссии#5. Готов к обслуживанию#e9#true]
16:05:37,234 INFO  [Run] Start output action [o1.z21] execution
```

```
Клиент согласился заплатить комиссию.
16:05:38,234 INFO  [Run] Finish output action [o1.z21] execution
16:05:38,234 INFO  [Run] Start event [e9] processing. In state [/A1:5. Готов
к обслуживанию/A2:Top]
16:05:38,234 INFO  [Run] Transition to go found [0. Начальное состояние#1.
Выбор валюты##true]
16:05:38,234 INFO  [Run] Start output action [o3.z0] execution
Включено устройство для обмена.
16:05:38,234 INFO  [Run] Finish output action [o3.z0] execution
16:05:38,234 INFO  [Run] Finish event [e9] processing. In state [/A1:5. Готов
к обслуживанию/A2:1. Выбор валюты]
16:05:38,234 INFO  [Run] Finish event [e9] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:41,828 INFO  [Run] Start event [e1] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:41,828 DEBUG [Run] Try transition [5. Готов к обслуживанию#5. Готов к
обслуживанию#e1#!o2.x1]
16:05:41,828 INFO  [Run] Start input action [o2.x1] calculation
Сумма нулевая
Банкнота принята
16:05:41,828 INFO  [Run] Finish input action [o2.x1] calculation. Its value
is [true]
16:05:41,828 DEBUG [Run] Try transition [5. Готов к обслуживанию#5. Готов к
обслуживанию#e1#o2.x1]
16:05:41,828 INFO  [Run] Transition to go found [5. Готов к обслуживанию#5.
Готов к обслуживанию#e1#o2.x1]
16:05:41,828 INFO  [Run] Start output action [o2.z1] execution
Добавлен номинал банкноты к текущей сумме.
16:05:41,828 INFO  [Run] Finish output action [o2.z1] execution
16:05:41,828 INFO  [Run] Start output action [o1.z10] execution
Показали накопленную сумму.
16:05:42,828 INFO  [Run] Finish output action [o1.z10] execution
16:05:42,828 INFO  [Run] Start event [e1] processing. In state [/A1:5. Готов
к обслуживанию/A2:1. Выбор валюты]
16:05:42,828 INFO  [Run] Finish event [e1] processing. In state [/A1:5. Готов
к обслуживанию/A2:1. Выбор валюты]
16:05:42,828 INFO  [Run] Finish event [e1] processing. In state [/A1:5. Готов
к обслуживанию]
RUB
16:05:47,171 INFO  [Run] Start event [e2] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:47,171 INFO  [Run] Start event [e2] processing. In state [/A1:5. Готов
к обслуживанию/A2:1. Выбор валюты]
16:05:47,171 DEBUG [Run] Try transition [1. Выбор валюты#1. Выбор
валюты#e2#o2.x3 && o3.x1]
16:05:47,171 INFO  [Run] Start input action [o2.x3] calculation
Сумма ненулевая
16:05:47,171 INFO  [Run] Finish input action [o2.x3] calculation. Its value
is [true]
16:05:47,171 INFO  [Run] Start input action [o3.x1] calculation
Выбранная валюта не закончилась.
16:05:47,171 INFO  [Run] Finish input action [o3.x1] calculation. Its value
is [false]
16:05:47,171 DEBUG [Run] Try transition [1. Выбор валюты#1. Выбор
валюты#e2#!o2.x3]
16:05:47,171 DEBUG [Run] Try transition [1. Выбор валюты#2. Вычисление
курса#e2#o2.x3 && !o3.x1]
16:05:47,171 INFO  [Run] Transition to go found [1. Выбор валюты#2.
Вычисление курса#e2#o2.x3 && !o3.x1]
16:05:47,171 INFO  [Run] Start on-enter action [o3.z7] execution
Вычисление курса
16:05:47,171 INFO  [Run] Finish on-enter action [o3.z7] execution
16:05:47,171 INFO  [Run] Finish event [e2] processing. In state [/A1:5. Готов
к обслуживанию/A2:2. Вычисление курса]
```

```
16:05:47,171 INFO  [Run] Finish event [e2] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:48,171 INFO  [Run] Start event [e7] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:48,171 INFO  [Run] Start event [e7] processing. In state [/A1:5. Готов
к обслуживанию/A2:2. Вычисление курса]
16:05:48,171 DEBUG [Run] Try transition [2. Вычисление курса#3.
Обмен#e7#o3.x4]
16:05:48,171 INFO  [Run] Start input action [o3.x4] calculation
16:05:48,171 INFO  [Run] Finish input action [o3.x4] calculation. Its value
is [false]
16:05:48,171 DEBUG [Run] Try transition [2. Вычисление курса#3.
Обмен#e7#!o3.x4]
16:05:48,171 INFO  [Run] Transition to go found [2. Вычисление курса#3.
Обмен#e7#!o3.x4]
16:05:48,171 INFO  [Run] Start output action [o1.z19] execution
Обмен будет произведен по стандартному курсу.
16:05:49,171 INFO  [Run] Finish output action [o1.z19] execution
16:05:49,171 INFO  [Run] Start on-enter action [o3.z3] execution
Произвели обмен.
16:05:49,171 INFO  [Run] Finish on-enter action [o3.z3] execution
16:05:49,171 INFO  [Run] Start on-enter action [o1.z10] execution
Показали накопленную сумму.
16:05:50,171 INFO  [Run] Finish on-enter action [o1.z10] execution
16:05:50,171 INFO  [Run] Finish event [e7] processing. In state [/A1:5. Готов
к обслуживанию/A2:3. Обмен]
16:05:50,171 INFO  [Run] Finish event [e7] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:50,171 INFO  [Run] Start event [e5] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:50,171 INFO  [Run] Start event [e5] processing. In state [/A1:5. Готов
к обслуживанию/A2:3. Обмен]
16:05:50,171 DEBUG [Run] Try transition [3. Обмен#1. Выбор валюты#e5#!o3.x3]
16:05:50,171 INFO  [Run] Start input action [o3.x3] calculation
Пересчет количества имеющихся банкнот.
16:05:50,171 INFO  [Run] Finish input action [o3.x3] calculation. Its value
is [true]
16:05:50,171 DEBUG [Run] Try transition [3. Обмен#4. Печать чека#e5#!o3.x1 &&
o3.x3]
16:05:50,171 INFO  [Run] Start input action [o3.x1] calculation
Выбранная валюта не закончилась.
16:05:50,171 INFO  [Run] Finish input action [o3.x1] calculation. Its value
is [false]
16:05:50,171 INFO  [Run] Transition to go found [3. Обмен#4. Печать
чека#e5#!o3.x1 && o3.x3]
16:05:50,171 INFO  [Run] Start output action [o2.z3] execution
Накопленная сумма обнулена
16:05:50,171 INFO  [Run] Finish output action [o2.z3] execution
16:05:50,171 INFO  [Run] Start output action [o4.z5] execution
Включены кнопки Да/Нет.
16:05:50,187 INFO  [Run] Finish output action [o4.z5] execution
16:05:50,187 INFO  [Run] Start on-enter action [o1.z11] execution
Задан вопрос о печати чека.
16:05:51,187 INFO  [Run] Finish on-enter action [o1.z11] execution
16:05:51,187 INFO  [Run] Finish event [e5] processing. In state [/A1:5. Готов
к обслуживанию/A2:4. Печать чека]
16:05:51,187 INFO  [Run] Finish event [e5] processing. In state [/A1:5. Готов
к обслуживанию]
receipt
16:05:53,031 INFO  [Run] Start event [e10] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:53,031 INFO  [Run] Start event [e10] processing. In state [/A1:5. Готов
к обслуживанию/A2:4. Печать чека]
```

```
16:05:53,031 DEBUG [Run] Try transition [4. Печать чека#5. Выдача
денег#e10#true]
16:05:53,031 INFO  [Run] Transition to go found [4. Печать чека#5. Выдача
денег#e10#true]
16:05:53,031 INFO  [Run] Start output action [o4.z6] execution
Напечатан чек.
16:05:53,046 INFO  [Run] Finish output action [o4.z6] execution
16:05:53,046 INFO  [Run] Start on-enter action [o4.z1] execution
Банкноты выданы.
16:05:53,046 INFO  [Run] Finish on-enter action [o4.z1] execution
16:05:53,046 INFO  [Run] Start on-enter action [o1.z6] execution
Вывели результаты обмена
16:05:54,046 INFO  [Run] Finish on-enter action [o1.z6] execution
16:05:54,046 INFO  [Run] Finish event [e10] processing. In state [/A1:5.
Готов к обслуживанию/A2:5. Выдача денег]
16:05:54,046 INFO  [Run] Finish event [e10] processing. In state [/A1:5.
Готов к обслуживанию]
16:05:54,046 INFO  [Run] Start event [e4] processing. In state [/A1:5. Готов
к обслуживанию]
16:05:54,046 INFO  [Run] Start event [e4] processing. In state [/A1:5. Готов
к обслуживанию/A2:5. Выдача денег]
16:05:54,046 DEBUG [Run] Try transition [5. Выдача денег#1. Выбор
валюты#e4#true]
16:05:54,046 INFO  [Run] Transition to go found [5. Выдача денег#1. Выбор
валюты#e4#true]
16:05:54,046 INFO  [Run] Start output action [o1.z25] execution
Включен индикатор Евро
16:05:54,046 INFO  [Run] Finish output action [o1.z25] execution
16:05:54,046 INFO  [Run] Finish event [e4] processing. In state [/A1:5. Готов
к обслуживанию/A2:1. Выбор валюты]
16:05:54,046 INFO  [Run] Finish event [e4] processing. In state [/A1:5. Готов
к обслуживанию]
16:06:10,875 INFO  [Run] Start event [e6] processing. In state [/A1:5. Готов
к обслуживанию]
16:06:10,875 DEBUG [Run] Try transition [5. Готов к обслуживанию#4.
Прощальное сообщение#e6#true]
16:06:10,875 INFO  [Run] Transition to go found [5. Готов к обслуживанию#4.
Прощальное сообщение#e6#true]
16:06:10,875 INFO  [Run] Start on-enter action [o1.z13] execution
Выключение дисплея.
16:06:11,875 INFO  [Run] Finish on-enter action [o1.z13] execution
16:06:11,875 INFO  [Run] Finish event [e6] processing. In state [/A1:4.
Прощальное сообщение]
16:06:11,875 INFO  [Run] Start event [e21] processing. In state [/A1:4.
Прощальное сообщение]
16:06:11,875 DEBUG [Run] Try transition [4. Прощальное сообщение#2. Финальное
состояние#e21#true]
16:06:11,875 INFO  [Run] Transition to go found [4. Прощальное сообщение#2.
Финальное состояние#e21#true]
16:06:11,890 INFO  [Run] State machine came to final state [/A1:2. Финальное
состояние]
16:06:11,890 INFO  [Run] Finish event [e21] processing. In state [/A1:2.
Финальное состояние]
```

Приложение 2. 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="co.Display" name="o1"/>
  <controlledObject class="co.PourOutDevice" name="o4"/>
  <controlledObject class="co.Client" name="o6"/>
  <controlledObject class="co.ExchangeDevice" name="o3"/>
  <controlledObject class="co.CoinMechanism" name="o2"/>
  <controlledObject class="co.Server" name="o5"/>
  <eventProvider class="ep.ControlPane" name="p1">
    <association clientRole="p1" targetRef="A1"/>
  </eventProvider>
  <eventProvider class="ep.ServerEvent" name="p2">
    <association clientRole="p2" targetRef="A1"/>
  </eventProvider>
  <eventProvider class="ep.ClientEvent" name="p3">
    <association clientRole="p3" 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="o6" targetRef="o6"/>
    <association clientRole="A1" supplierRole="o2" targetRef="o2"/>
    <association clientRole="A1" supplierRole="A3" targetRef="A3"/>
    <state name="Top" type="NORMAL">
      <state name="0. Начальное состояние" type="INITIAL"/>
      <state name="1. Проверка связи с сервером" type="NORMAL">
        <stateMachineRef name="A3"/>
        <outputAction ident="o6.z0"/>
      </state>
      <state name="2. Получение текущего курса" type="NORMAL">
        <stateMachineRef name="A3"/>
        <outputAction ident="o6.z1"/>
      </state>
      <state name="3. Предупреждение о комиссии" type="NORMAL">
        <outputAction ident="o1.z26"/>
      </state>
      <state name="4. Прощальное сообщение" type="NORMAL">
        <outputAction ident="o1.z13"/>
      </state>
      <state name="2. Финальное состояние" type="FINAL"/>
      <state name="5. Готов к обслуживанию" type="NORMAL">
        <stateMachineRef name="A2"/>
      </state>
    </state>
    <transition sourceRef="0. Начальное состояние" targetRef="1. Проверка
связи с сервером">
      <outputAction ident="o1.z1"/>
    </transition>
    <transition event="e18" sourceRef="1. Проверка связи с сервером"
targetRef="2. Получение текущего курса"/>
    <transition event="e19" sourceRef="1. Проверка связи с сервером"
targetRef="4. Прощальное сообщение">
      <outputAction ident="o1.z23"/>
    </transition>
```

```
    <transition event="e20" sourceRef="2. Получение текущего курса"
targetRef="3. Предупреждение о комиссии">
      <outputAction ident="o1.z3"/>
    </transition>
    <transition event="e26" sourceRef="2. Получение текущего курса"
targetRef="4. Прощальное сообщение">
      <outputAction ident="o1.z24"/>
    </transition>
    <transition event="e8" sourceRef="3. Предупреждение о комиссии"
targetRef="4. Прощальное сообщение">
      <outputAction ident="o1.z20"/>
    </transition>
    <transition event="e6" sourceRef="3. Предупреждение о комиссии"
targetRef="4. Прощальное сообщение"/>
    <transition event="e9" sourceRef="3. Предупреждение о комиссии"
targetRef="5. Готов к обслуживанию">
      <outputAction ident="o1.z21"/>
    </transition>
    <transition event="e21" sourceRef="4. Прощальное сообщение" targetRef="2.
Финальное состояние"/>
    <transition event="e6" sourceRef="5. Готов к обслуживанию" targetRef="4.
Прощальное сообщение"/>
    <transition event="e1" guard="!o2.x1" sourceRef="5. Готов к обслуживанию"
targetRef="5. Готов к обслуживанию">
      <outputAction ident="o2.z2"/>
      <outputAction ident="o1.z2"/>
    </transition>
    <transition event="e1" guard="o2.x1" sourceRef="5. Готов к обслуживанию"
targetRef="5. Готов к обслуживанию">
      <outputAction ident="o2.z1"/>
      <outputAction ident="o1.z10"/>
    </transition>
    <transition event="e3" guard="!o2.x3" sourceRef="5. Готов к обслуживанию"
targetRef="5. Готов к обслуживанию"/>
    <transition event="e3" guard="o2.x3" sourceRef="5. Готов к обслуживанию"
targetRef="5. Готов к обслуживанию">
      <outputAction ident="o2.z4"/>
      <outputAction ident="o1.z4"/>
    </transition>
  </stateMachine>
  <stateMachine name="A2">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A2" supplierRole="o1" targetRef="o1"/>
    <association clientRole="A2" supplierRole="o4" targetRef="o4"/>
    <association clientRole="A2" supplierRole="o6" targetRef="o6"/>
    <association clientRole="A2" supplierRole="o3" targetRef="o3"/>
    <association clientRole="A2" supplierRole="o2" targetRef="o2"/>
    <state name="Top" type="NORMAL">
      <state name="0. Начальное состояние" type="INITIAL"/>
      <state name="1. Выбор валюты" type="NORMAL"/>
      <state name="2. Вычисление курса" type="NORMAL">
        <outputAction ident="o3.z7"/>
      </state>
      <state name="5. Выдача денег" type="NORMAL">
        <outputAction ident="o4.z1"/>
        <outputAction ident="o1.z6"/>
      </state>
      <state name="4. Печать чека" type="NORMAL">
        <outputAction ident="o1.z11"/>
      </state>
      <state name="3. Обмен" type="NORMAL">
        <outputAction ident="o3.z3"/>
        <outputAction ident="o1.z10"/>
```

```
      </state>
    </state>
    <transition sourceRef="0. Начальное состояние" targetRef="1. Выбор
валюты">
      <outputAction ident="o3.z0"/>
    </transition>
    <transition event="e2" guard="o2.x3 &amp;amp;&amp;amp; o3.x1" sourceRef="1. Выбор
валюты" targetRef="1. Выбор валюты">
      <outputAction ident="o1.z7"/>
      <outputAction ident="o1.z12"/>
    </transition>
    <transition event="e2" guard="!o2.x3" sourceRef="1. Выбор валюты"
targetRef="1. Выбор валюты"/>
    <transition event="e2" guard="o2.x3 &amp;amp;&amp;amp; !o3.x1" sourceRef="1.
Выбор валюты" targetRef="2. Вычисление курса"/>
    <transition event="e7" guard="o3.x4" sourceRef="2. Вычисление курса"
targetRef="3. Обмен">
      <outputAction ident="o1.z18"/>
    </transition>
    <transition event="e7" guard="!o3.x4" sourceRef="2. Вычисление курса"
targetRef="3. Обмен">
      <outputAction ident="o1.z19"/>
    </transition>
    <transition event="e4" sourceRef="5. Выдача денег" targetRef="1. Выбор
валюты">
      <outputAction ident="o1.z25"/>
    </transition>
    <transition event="e10" sourceRef="4. Печать чека" targetRef="5. Выдача
денег">
      <outputAction ident="o4.z6"/>
    </transition>
    <transition event="e11" sourceRef="4. Печать чека" targetRef="5. Выдача
денег"/>
    <transition event="e5" guard="!o3.x3" sourceRef="3. Обмен" targetRef="1.
Выбор валюты">
      <outputAction ident="o1.z7"/>
    </transition>
    <transition event="e5" guard="!o3.x1 &amp;amp;&amp;amp; o3.x3" sourceRef="3.
Обмен" targetRef="4. Печать чека">
      <outputAction ident="o2.z3"/>
      <outputAction ident="o4.z5"/>
    </transition>
    <transition event="e5" guard="o3.x1 &amp;amp;&amp;amp; o3.x3" sourceRef="3.
Обмен" targetRef="4. Печать чека">
      <outputAction ident="o1.z12"/>
      <outputAction ident="o2.z3"/>
      <outputAction ident="o4.z5"/>
    </transition>
  </stateMachine>
  <stateMachine name="A3">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A3" supplierRole="o5" targetRef="o5"/>
    <state name="Top" type="NORMAL">
      <state name="2. Авторизация" type="NORMAL">
        <outputAction ident="o5.z5"/>
      </state>
      <state name="1. Ожидание запроса" type="NORMAL">
        <outputAction ident="o5.z0"/>
      </state>
      <state name="3. Получение курса" type="NORMAL">
        <outputAction ident="o5.z1"/>
      </state>
      <state name="5. Ответ на запрос" type="NORMAL">
```

```
        <outputAction ident="o5.z4"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="s3" type="FINAL"/>
      <state name="4. Информирование об окончании валюты" type="NORMAL">
        <outputAction ident="o5.z3"/>
      </state>
    </state>
    <transition event="e16" sourceRef="2. Авторизация" targetRef="5. Ответ на
запрос"/>
    <transition event="e17" sourceRef="1. Ожидание запроса" targetRef="2.
Авторизация"/>
    <transition event="e13" sourceRef="1. Ожидание запроса" targetRef="3.
Получение курса"/>
    <transition event="e15" sourceRef="1. Ожидание запроса" targetRef="4.
Информирование об окончании валюты"/>
    <transition event="e16" sourceRef="3. Получение курса" targetRef="5.
Ответ на запрос"/>
    <transition event="*" sourceRef="5. Ответ на запрос" targetRef="s3"/>
    <transition sourceRef="s1" targetRef="1. Ожидание запроса"/>
    <transition event="e16" sourceRef="4. Информирование об окончании валюты"
targetRef="5. Ответ на запрос"/>
  </stateMachine>
</model>
```

Приложение 3. Сгенерированный по *XML*-описанию код на *Java*

```
/**
 * This file was generated from model [Model1] on [Mon May 14 02:46:18 MSD 2007].
 * Do not change content of this file.
 */

import java.io.IOException;
import java.util.*;

import org.apache.commons.lang.BooleanUtils;
import org.apache.commons.lang.math.NumberUtils;
import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.evelopers.common.exception.*;
import com.evelopers.unimod.core.stateworks.*;
import com.evelopers.unimod.debug.app.AppDebugger;
import com.evelopers.unimod.debug.protocol.JavaSpecificMessageCoder;
import com.evelopers.unimod.runtime.*;
import com.evelopers.unimod.runtime.context.*;
import com.evelopers.unimod.runtime.logger.SimpleLogger;

public class Model1EventProcessor extends AbstractEventProcessor {

 private ModelStructure modelStructure;

 private static final int A1 = 1;

 private static final int A2 = 2;

 private static final int A3 = 3;

 private int decodeStateMachine(String sm) {

     if ("A1".equals(sm)) {
         return A1;
     } else

     if ("A2".equals(sm)) {
             return A2;
     } else

     if ("A3".equals(sm)) {
         return A3;
     }

     return -1;
 }

 private A1EventProcessor _A1;

 private A2EventProcessor _A2;

 private A3EventProcessor _A3;

 public Model1EventProcessor() {
     modelStructure = new Model1ModelStructure();
```

```
    _A1 = new A1EventProcessor();
    _A2 = new A2EventProcessor();
    _A3 = new A3EventProcessor();
}

public static void run(int debuggerPort, boolean debuggerSuspend)
        throws InterruptedException, EventProcessorException,
        CommonException, IOException {

    /* Create runtime engine */
    ModelEngine engine = createModelEngine(true);

    /* Setup logger */
    final Log log = LogFactory.getLog(Model1EventProcessor.class);
    engine.getEventProcessor().addEventProcessorListener(
            new SimpleLogger(log));

    /* Setup exception handler */
    engine.getEventProcessor().addExceptionHandler(new ExceptionHandler() {
        public void handleException(StateMachineContext context,
                SystemException e) {
            log.fatal(e.getChainedMessage(), e.getRootException());
        }
    });

    if (debuggerPort > 0) {
        AppDebugger d = new AppDebugger(debuggerPort, debuggerSuspend,
                new JavaSpecificMessageCoder(), engine);
        d.start();
    }
    engine.start();
}

public static void main(String[] args) throws Exception {
    int debuggerPort = NumberUtils.stringToInt(System
            .getProperty("debugger.port"), -1);
    boolean debuggerSuspend = BooleanUtils.toBoolean(System
            .getProperty("debugger.suspend"));
    Model1EventProcessor.run(debuggerPort, debuggerSuspend);
}

public static ModelEngine createModelEngine(boolean useEventQueue)
        throws CommonException {
    ObjectsManager objectsManager = new ObjectsManager();
    return ModelEngine.createStandAlone(
            useEventQueue ? (EventManager) new QueuedHandler()
                    : (EventManager) new StrictHandler(),
            new Model1EventProcessor(), objectsManager
                    .getControlledObjectsManager(), objectsManager
                    .getEventProvidersManager());
}

public static class ObjectsManager {
    private co.Display o1 = null;

    private co.PourOutDevice o4 = null;

    private co.Client o6 = null;

    private co.ExchangeDevice o3 = null;

    private co.CoinMechanism o2 = null;

    private co.Server o5 = null;
```

```
    private ep.ControlPane p1 = null;

    private ep.ClientEvent p3 = null;

    private ep.ServerEvent p2 = null;

    private ControlledObjectsManager controlledObjectsManager = new
ControlledObjectsManagerImpl();

    private EventProvidersManager eventProvidersManager = new
EventProvidersManagerImpl();

    public ControlledObjectsManager getControlledObjectsManager() {
        return controlledObjectsManager;
    }

    public EventProvidersManager getEventProvidersManager() {
        return eventProvidersManager;
    }

    private class ControlledObjectsManagerImpl implements
            ControlledObjectsManager {
        public void init(ModelEngine engine) throws CommonException {
        }

        public void dispose() {
        }

        public ControlledObject getControlledObject(String coName) {
            if (StringUtils.equals(coName, "o1")) {
                if (o1 == null) {
                    o1 = new co.Display();
                }
                return o1;
            }
            if (StringUtils.equals(coName, "o4")) {
                if (o4 == null) {
                    o4 = new co.PourOutDevice();
                }
                return o4;
            }
            if (StringUtils.equals(coName, "o6")) {
                if (o6 == null) {
                    o6 = new co.Client();
                }
                return o6;
            }
            if (StringUtils.equals(coName, "o3")) {
                if (o3 == null) {
                    o3 = new co.ExchangeDevice();
                }
                return o3;
            }
            if (StringUtils.equals(coName, "o2")) {
                if (o2 == null) {
                    o2 = new co.CoinMechanism();
                }
                return o2;
            }
            if (StringUtils.equals(coName, "o5")) {
                if (o5 == null) {
                    o5 = new co.Server();
                }
```

```
                return o5;
            }
            throw new IllegalArgumentException(
                    "Controlled object with name [" + coName
                            + "] wasn't found");
        }
    }

    private class EventProvidersManagerImpl implements
            EventProvidersManager {
        private List nonameEventProviders = new ArrayList();

        public void init(ModelEngine engine) throws CommonException {
            EventProvider ep;
            ep = getEventProvider("p1");
            ep.init(engine);
            ep = getEventProvider("p3");
            ep.init(engine);
            ep = getEventProvider("p2");
            ep.init(engine);
        }

        public void dispose() {
            EventProvider ep;
            ep = getEventProvider("p1");
            ep.dispose();
            ep = getEventProvider("p3");
            ep.dispose();
            ep = getEventProvider("p2");
            ep.dispose();
            for (Iterator i = nonameEventProviders.iterator(); i.hasNext();) {
                ep = (EventProvider) i.next();
                ep.dispose();
            }
        }

        public EventProvider getEventProvider(String epName) {
            if (StringUtils.equals(epName, "p1")) {
                if (p1 == null) {
                    p1 = new ep.ControlPane();
                }
                return p1;
            }
            if (StringUtils.equals(epName, "p3")) {
                if (p3 == null) {
                    p3 = new ep.ClientEvent();
                }
                return p3;
            }
            if (StringUtils.equals(epName, "p2")) {
                if (p2 == null) {
                    p2 = new ep.ServerEvent();
                }
                return p2;
            }
            throw new IllegalArgumentException("Event provider with name ["
                    + epName + "] wasn't found");
        }
    }
}

public ModelStructure getModelStructure() {
    return modelStructure;
}
```

```
public void setControlledObjectsMap(
        ControlledObjectsMap controlledObjectsMap) {
    super.setControlledObjectsMap(controlledObjectsMap);

    _A1.init(controlledObjectsMap);
    _A2.init(controlledObjectsMap);
    _A3.init(controlledObjectsMap);
}

protected StateMachineConfig process(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws SystemException {

    // get state machine from path
    int sm = decodeStateMachine(path.getStateMachine());

    try {
        switch (sm) {
        case A1:
            return _A1.process(event, context, path, config);
        case A2:
            return _A2.process(event, context, path, config);
        case A3:
            return _A3.process(event, context, path, config);
        default:
            throw new EventProcessorException("Unknown state machine ["
                    + path.getStateMachine() + "]");
        }
    } catch (Exception e) {
        if (e instanceof SystemException) {
            throw (SystemException) e;
        } else {
            throw new SystemException(e);
        }
    }
}

protected StateMachineConfig transiteToStableState(
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws SystemException {

    // get state machine from path
    int sm = decodeStateMachine(path.getStateMachine());

    try {
        switch (sm) {
        case A1:
            return _A1.transiteToStableState(context, path, config);
        case A2:
            return _A2.transiteToStableState(context, path, config);
        case A3:
            return _A3.transiteToStableState(context, path, config);
        default:
            throw new EventProcessorException("Unknown state machine ["
                    + path.getStateMachine() + "]");
        }
    } catch (Exception e) {
        if (e instanceof SystemException) {
            throw (SystemException) e;
        } else {
            throw new SystemException(e);
        }
    }
```

```
}

private class Model1ModelStructure implements ModelStructure {
    private Map configManagers = new HashMap();

    private Model1ModelStructure() {
        configManagers
                .put(
                        "A1",
                        new
com.evelopers.unimod.runtime.config.DistinguishConfigManager());
        configManagers
                .put(
                        "A2",
                        new
com.evelopers.unimod.runtime.config.DistinguishConfigManager());
        configManagers
                .put(
                        "A3",
                        new
com.evelopers.unimod.runtime.config.DistinguishConfigManager());
    }

    public StateMachinePath getRootPath() throws EventProcessorException {
        return new StateMachinePath("A1");
    }

    public StateMachineConfigManager getConfigManager(String stateMachine)
            throws EventProcessorException {
        return (StateMachineConfigManager) configManagers.get(stateMachine);
    }

    public StateMachineConfig getTopConfig(String stateMachine)
            throws EventProcessorException {
        int sm = decodeStateMachine(stateMachine);

        switch (sm) {
        case A1:
            return new StateMachineConfig("Top");
        case A2:
            return new StateMachineConfig("Top");
        case A3:
            return new StateMachineConfig("Top");
        default:
            throw new EventProcessorException("Unknown state machine ["
                    + stateMachine + "]");
        }
    }

    public boolean isFinal(String stateMachine, StateMachineConfig config)
            throws EventProcessorException {
        /* Get state machine from path */
        int sm = decodeStateMachine(stateMachine);
        int state;
        switch (sm) {
        case A1:
            state = _A1.decodeState(config.getActiveState());
            switch (state) {
            case A1EventProcessor._2__Финальное_состояние:
                return true;
            default:
                return false;
            }
        case A2:
```

```
            state = _A2.decodeState(config.getActiveState());
            switch (state) {
            default:
                return false;
            }
        case A3:
            state = _A3.decodeState(config.getActiveState());
            switch (state) {
            case A3EventProcessor.s3:
                return true;
            default:
                return false;
            }
        default:
            throw new EventProcessorException("Unknown state machine ["
                    + stateMachine + "]");
        }
    }
}

private class A1EventProcessor {

    // states
    private static final int Top = 1;

    private static final int _0__Начальное_состояние = 2;

    private static final int _1__Проверка_связи_с_сервером = 3;

    private static final int _2__Получение_текущего_курса = 4;

    private static final int _3__Предупреждение_о_комиссии = 5;

    private static final int _4__Прощальное_сообщение = 6;

    private static final int _2__Финальное_состояние = 7;

    private static final int _5__Готов_к_обслуживанию = 8;

    private int decodeState(String state) {

        if ("Top".equals(state)) {
            return Top;
        } else

        if ("0. Начальное состояние".equals(state)) {
            return _0__Начальное_состояние;
        } else

        if ("1. Проверка связи с сервером".equals(state)) {
            return _1__Проверка_связи_с_сервером;
        } else

        if ("2. Получение текущего курса".equals(state)) {
            return _2__Получение_текущего_курса;
        } else

        if ("3. Предупреждение о комиссии".equals(state)) {
            return _3__Предупреждение_о_комиссии;
        } else

        if ("4. Прощальное сообщение".equals(state)) {
            return _4__Прощальное_сообщение;
        } else
```

```
    if ("2. Финальное состояние".equals(state)) {
        return _2__Финальное_состояние;
    } else

    if ("5. Готов к обслуживанию".equals(state)) {
        return _5__Готов_к_обслуживанию;
    }

    return -1;
}

// events
private static final int e19 = 1;

private static final int e20 = 2;

private static final int e26 = 3;

private static final int e9 = 4;

private static final int e1 = 5;

private static final int e21 = 6;

private static final int e3 = 7;

private static final int e6 = 8;

private static final int e18 = 9;

private static final int e8 = 10;

private int decodeEvent(String event) {

    if ("e19".equals(event)) {
        return e19;
    } else

    if ("e20".equals(event)) {
        return e20;
    } else

    if ("e26".equals(event)) {
        return e26;
    } else

    if ("e9".equals(event)) {
        return e9;
    } else

    if ("e1".equals(event)) {
        return e1;
    } else

    if ("e21".equals(event)) {
        return e21;
    } else

    if ("e3".equals(event)) {
        return e3;
    } else

    if ("e6".equals(event)) {
```

```
        return e6;
    } else

    if ("e18".equals(event)) {
        return e18;
    } else

    if ("e8".equals(event)) {
        return e8;
    }

    return -1;
}

private co.Display o1;

private co.Client o6;

private co.CoinMechanism o2;

private void init(ControlledObjectsMap controlledObjectsMap) {
    o1 = (co.Display) controlledObjectsMap.getControlledObject("o1");
    o6 = (co.Client) controlledObjectsMap.getControlledObject("o6");
    o2 = (co.CoinMechanism) controlledObjectsMap
            .getControlledObject("o2");
}

private StateMachineConfig process(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {
    config = lookForTransition(event, context, path, config);

    config = transiteToStableState(context, path, config);

    // execute included state machines
    executeSubmachines(event, context, path, config);

    return config;
}

private void executeSubmachines(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {
    int state = decodeState(config.getActiveState());

    while (true) {
        switch (state) {
        case _0__Начальное_состояние:

            return;
        case _1__Проверка_связи_с_сервером:
            // 1. Проверка связи с сервером includes A3

            fireBeforeSubmachineExecution(context, event, path,
                    "1. Проверка связи с сервером", "A3");

            Model1EventProcessor.this.process(event, context,
                    new StateMachinePath(path,
                            "1. Проверка связи с сервером", "A3"));

            fireAfterSubmachineExecution(context, event, path,
                    "1. Проверка связи с сервером", "A3");

            return;
```

```
        case _2__Получение_текущего_курса:
            // 2. Получение текущего курса includes A3

            fireBeforeSubmachineExecution(context, event, path,
                    "2. Получение текущего курса", "A3");

            Model1EventProcessor.this.process(event, context,
                    new StateMachinePath(path,
                            "2. Получение текущего курса", "A3"));

            fireAfterSubmachineExecution(context, event, path,
                    "2. Получение текущего курса", "A3");

            return;
        case _3__Предупреждение_о_комиссии:

            return;
        case _4__Прощальное_сообщение:

            return;
        case _2__Финальное_состояние:

            return;
        case _5__Готов_к_обслуживанию:
            // 5. Готов к обслуживанию includes A2

            fireBeforeSubmachineExecution(context, event, path,
                    "5. Готов к обслуживанию", "A2");

            Model1EventProcessor.this.process(event, context,
                    new StateMachinePath(path,
                            "5. Готов к обслуживанию", "A2"));

            fireAfterSubmachineExecution(context, event, path,
                    "5. Готов к обслуживанию", "A2");

            return;
        default:
            throw new EventProcessorException("State with name ["
                    + config.getActiveState()
                    + "] is unknown for state machine [A1]");
        }
    }
}

private StateMachineConfig transiteToStableState(
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {

    int s = decodeState(config.getActiveState());
    Event event;

    switch (s) {
    case Top:

        fireComeToState(context, path, "0. Начальное состояние");

        // 0. Начальное состояние->1. Проверка связи с сервером
        // [true]/o1.z1
        event = Event.NO_EVENT;
        fireTransitionFound(context, path, "0. Начальное состояние",
                event,
                "0. Начальное состояние#1. Проверка связи с сервером##true");
```

```
        fireBeforeOutputActionExecution(
                context,
                path,
                "0. Начальное состояние#1. Проверка связи с сервером##true",
                "o1.z1");

        o1.z1(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "0. Начальное состояние#1. Проверка связи с сервером##true",
                "o1.z1");

        fireComeToState(context, path, "1. Проверка связи с сервером");

        // 1. Проверка связи с сервером [o6.z0]
        fireBeforeOutputActionExecution(
                context,
                path,
                "0. Начальное состояние#1. Проверка связи с сервером##true",
                "o6.z0");

        o6.z0(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "0. Начальное состояние#1. Проверка связи с сервером##true",
                "o6.z0");

        return new StateMachineConfig("1. Проверка связи с сервером");
    }

    return config;
}

private StateMachineConfig lookForTransition(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {

    boolean

    o2_x3 = false,

    o2_x1 = false;

    BitSet calculatedInputActions = new BitSet(2);

    int s = decodeState(config.getActiveState());
    int e = decodeEvent(event.getName());

    while (true) {
        switch (s) {
        case _1__Проверка_связи_с_сервером:

            switch (e) {
            case e19:

                // 1. Проверка связи с сервером->4. Прощальное сообщение
                // e19[true]/o1.z23

                fireTransitionCandidate(context, path,
                        "1. Проверка связи с сервером", event,
```

```
"1. Проверка связи с сервером#4. Прощальное сообщение#e19#true");

                    fireTransitionFound(context, path,
                            "1. Проверка связи с сервером", event,
"1. Проверка связи с сервером#4. Прощальное сообщение#e19#true");

                    fireBeforeOutputActionExecution(
                            context,
                            path,
"1. Проверка связи с сервером#4. Прощальное сообщение#e19#true",
                            "o1.z23");

                    o1.z23(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
"1. Проверка связи с сервером#4. Прощальное сообщение#e19#true",
                            "o1.z23");

                    fireComeToState(context, path,
                            "4. Прощальное сообщение");

                    // 4. Прощальное сообщение [o1.z13]
                    fireBeforeOutputActionExecution(
                            context,
                            path,
"1. Проверка связи с сервером#4. Прощальное сообщение#e19#true",
                            "o1.z13");

                    o1.z13(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
"1. Проверка связи с сервером#4. Прощальное сообщение#e19#true",
                            "o1.z13");
                    return new StateMachineConfig("4. Прощальное сообщение");

                case e18:

                    // 1. Проверка связи с сервером->2. Получение текущего
                    // курса e18[true]/

                    fireTransitionCandidate(context, path,
                            "1. Проверка связи с сервером", event,
"1. Проверка связи с сервером#2. Получение текущего курса#e18#true");

                    fireTransitionFound(context, path,
                            "1. Проверка связи с сервером", event,
"1. Проверка связи с сервером#2. Получение текущего курса#e18#true");

                    fireComeToState(context, path,
                            "2. Получение текущего курса");

                    // 2. Получение текущего курса [o6.z1]
                    fireBeforeOutputActionExecution(
                            context,
                            path,
"1. Проверка связи с сервером#2. Получение текущего курса#e18#true",
                            "o6.z1");

                    o6.z1(context);
```

```
                    fireAfterOutputActionExecution(
                            context,
                            path,
"1. Проверка связи с сервером#2. Получение текущего курса#e18#true",
                            "o6.z1");
                    return new StateMachineConfig(
                            "2. Получение текущего курса");

                default:

                    // transition not found
                    return config;
                }

            case _2__Получение_текущего_курса:

                switch (e) {
                case e20:

                    // 2. Получение текущего курса->3. Предупреждение о
                    // комиссии e20[true]/o1.z3

                    fireTransitionCandidate(context, path,
                            "2. Получение текущего курса", event,
"2. Получение текущего курса#3. Предупреждение о комиссии#e20#true");

                    fireTransitionFound(context, path,
                            "2. Получение текущего курса", event,
"2. Получение текущего курса#3. Предупреждение о комиссии#e20#true");

                    fireBeforeOutputActionExecution(
                            context,
                            path,
 "2. Получение текущего курса#3. Предупреждение о комиссии#e20#true",
                            "o1.z3");

                    o1.z3(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
"2. Получение текущего курса#3. Предупреждение о комиссии#e20#true",
                            "o1.z3");

                    fireComeToState(context, path,
                            "3. Предупреждение о комиссии");

                    // 3. Предупреждение о комиссии [o1.z26]
                    fireBeforeOutputActionExecution(
                            context,
                            path,
     "2. Получение текущего курса#3. Предупреждение о комиссии#e20#true",
                            "o1.z26");

                    o1.z26(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
     "2. Получение текущего курса#3. Предупреждение о комиссии#e20#true",
                            "o1.z26");
                    return new StateMachineConfig(
                            "3. Предупреждение о комиссии");
```

```
            case e26:

                // 2. Получение текущего курса->4. Прощальное сообщение
                // e26[true]/o1.z24

                fireTransitionCandidate(context, path,
                        "2. Получение текущего курса", event,
                "2. Получение текущего курса#4. Прощальное сообщение#e26#true");

                fireTransitionFound(context, path,
                        "2. Получение текущего курса", event,
                "2. Получение текущего курса#4. Прощальное сообщение#e26#true");

                fireBeforeOutputActionExecution(
                        context,
                        path,
"2. Получение текущего курса#4. Прощальное сообщение#e26#true",
                        "o1.z24");

                o1.z24(context);

                fireAfterOutputActionExecution(
                        context,
                        path,
                "2. Получение текущего курса#4. Прощальное сообщение#e26#true",
                        "o1.z24");

                fireComeToState(context, path,
                        "4. Прощальное сообщение");

                // 4. Прощальное сообщение [o1.z13]
                fireBeforeOutputActionExecution(
                        context,
                        path,
                "2. Получение текущего курса#4. Прощальное сообщение#e26#true",
                        "o1.z13");

                o1.z13(context);

                fireAfterOutputActionExecution(
                        context,
                        path,
"2. Получение текущего курса#4. Прощальное сообщение#e26#true",
                        "o1.z13");
                return new StateMachineConfig("4. Прощальное сообщение");

            default:

                // transition not found
                return config;
            }

        case _3__Предупреждение_о_комиссии:

            switch (e) {
            case e9:

                // 3. Предупреждение о комиссии->5. Готов к обслуживанию
                // e9[true]/o1.z21

                fireTransitionCandidate(context, path,
                        "3. Предупреждение о комиссии", event,
                "3. Предупреждение о комиссии#5. Готов к обслуживанию#e9#true");
```

```
        fireTransitionFound(context, path,
                "3. Предупреждение о комиссии", event,
        "3. Предупреждение о комиссии#5. Готов к обслуживанию#e9#true");

        fireBeforeOutputActionExecution(
                context,
                path,
        "3. Предупреждение о комиссии#5. Готов к обслуживанию#e9#true",
                "o1.z21");

        o1.z21(context);

        fireAfterOutputActionExecution(
                context,
                path,
        "3. Предупреждение о комиссии#5. Готов к обслуживанию#e9#true",
                "o1.z21");

        fireComeToState(context, path,
                "5. Готов к обслуживанию");

        // 5. Готов к обслуживанию []
        return new StateMachineConfig("5. Готов к обслуживанию");

    case e6:

        // 3. Предупреждение о комиссии->4. Прощальное сообщение
        // e6[true]/

        fireTransitionCandidate(context, path,
                "3. Предупреждение о комиссии", event,
        "3. Предупреждение о комиссии#4. Прощальное сообщение#e6#true");

        fireTransitionFound(context, path,
                "3. Предупреждение о комиссии", event,
        "3. Предупреждение о комиссии#4. Прощальное сообщение#e6#true");

        fireComeToState(context, path,
                "4. Прощальное сообщение");

        // 4. Прощальное сообщение [o1.z13]
        fireBeforeOutputActionExecution(
                context,
                path,
        "3. Предупреждение о комиссии#4. Прощальное сообщение#e6#true",
                "o1.z13");

        o1.z13(context);

        fireAfterOutputActionExecution(
                context,
                path,
        "3. Предупреждение о комиссии#4. Прощальное сообщение#e6#true",
                "o1.z13");
        return new StateMachineConfig("4. Прощальное сообщение");

    case e8:

        // 3. Предупреждение о комиссии->4. Прощальное сообщение
        // e8[true]/o1.z20

        fireTransitionCandidate(context, path,
                "3. Предупреждение о комиссии", event,
        "3. Предупреждение о комиссии#4. Прощальное сообщение#e8#true");
```

```
            fireTransitionFound(context, path,
                    "3. Предупреждение о комиссии", event,
            "3. Предупреждение о комиссии#4. Прощальное сообщение#e8#true");

            fireBeforeOutputActionExecution(
                    context,
                    path,
            "3. Предупреждение о комиссии#4. Прощальное сообщение#e8#true",
                    "o1.z20");

            o1.z20(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
            "3. Предупреждение о комиссии#4. Прощальное сообщение#e8#true",
                    "o1.z20");

            fireComeToState(context, path,
                    "4. Прощальное сообщение");

            // 4. Прощальное сообщение [o1.z13]
            fireBeforeOutputActionExecution(
                    context,
                    path,
            "3. Предупреждение о комиссии#4. Прощальное сообщение#e8#true",
                    "o1.z13");

            o1.z13(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
            "3. Предупреждение о комиссии#4. Прощальное сообщение#e8#true",
                    "o1.z13");
            return new StateMachineConfig("4. Прощальное сообщение");

        default:

            // transition not found
            return config;
        }

    case _4__Прощальное_сообщение:

        switch (e) {
        case e21:

            // 4. Прощальное сообщение->2. Финальное состояние
            // e21[true]/

            fireTransitionCandidate(context, path,
                    "4. Прощальное сообщение", event,
                    "4. Прощальное сообщение#2. Финальное
                        состояние#e21#true");

            fireTransitionFound(context, path,
                    "4. Прощальное сообщение", event,
                    "4. Прощальное сообщение#2. Финальное
                        состояние#e21#true");

            fireComeToState(context, path, "2. Финальное состояние");
```

```
            // 2. Финальное состояние []
            return new StateMachineConfig("2. Финальное состояние");

        default:

            // transition not found
            return config;
        }

    case _5__Готов_к_обслуживанию:

        switch (e) {
        case e1:

            // 5. Готов к обслуживанию->5. Готов к обслуживанию
            // e1[!o2.x1]/o2.z2,o1.z2

            fireTransitionCandidate(context, path,
                    "5. Готов к обслуживанию", event,
                    "5. Готов к обслуживанию#5. Готов к
                        обслуживанию#e1#!o2.x1");

            if (!isInputActionCalculated(calculatedInputActions,
                    _o2_x1)) {

                fireBeforeInputActionExecution(
                        context,
                        path,
                    "5. Готов к обслуживанию#5. Готов к обслуживанию#e1#!o2.x1",
                        "o2.x1");

                o2_x1 = o2.x1(context);

                fireAfterInputActionExecution(
                        context,
                        path,
                    "5. Готов к обслуживанию#5. Готов к обслуживанию#e1#!o2.x1",
                        "o2.x1", new Boolean(o2_x1));
            }

            if (!o2_x1) {

                fireTransitionFound(context, path,
                        "5. Готов к обслуживанию", event,
                    "5. Готов к обслуживанию#5. Готов к обслуживанию#e1#!o2.x1");

                fireBeforeOutputActionExecution(
                        context,
                        path,
                    "5. Готов к обслуживанию#5. Готов к обслуживанию#e1#!o2.x1",
                        "o2.z2");

                o2.z2(context);

                fireAfterOutputActionExecution(
                        context,
                        path,
                    "5. Готов к обслуживанию#5. Готов к обслуживанию#e1#!o2.x1",
                        "o2.z2");
                fireBeforeOutputActionExecution(
                        context,
                        path,
                    "5. Готов к обслуживанию#5. Готов к обслуживанию#e1#!o2.x1",
                        "o1.z2");
```

```
                o1.z2(context);

                fireAfterOutputActionExecution(
                        context,
                        path,
                "5. Готов к обслуживанию#5. Готов к обслуживанию#e1#!o2.x1",
                        "o1.z2");

                fireComeToState(context, path,
                        "5. Готов к обслуживанию");

                // 5. Готов к обслуживанию []
                return new StateMachineConfig(
                        "5. Готов к обслуживанию");

            }
            // 5. Готов к обслуживанию->5. Готов к обслуживанию
            // e1[o2.x1]/o2.z1,o1.z10

            fireTransitionCandidate(context, path,
                    "5. Готов к обслуживанию", event,
                    "5. Готов к обслуживанию#5. Готов к
                        обслуживанию#e1#o2.x1");

            if (o2_x1) {

                fireTransitionFound(context, path,
                        "5. Готов к обслуживанию", event,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e1#o2.x1");

                fireBeforeOutputActionExecution(
                        context,
                        path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e1#o2.x1",
                        "o2.z1");

                o2.z1(context);

                fireAfterOutputActionExecution(
                        context,
                        path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e1#o2.x1",
                        "o2.z1");
                fireBeforeOutputActionExecution(
                        context,
                        path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e1#o2.x1",
                        "o1.z10");

                o1.z10(context);

                fireAfterOutputActionExecution(
                        context,
                        path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e1#o2.x1",
                        "o1.z10");

                fireComeToState(context, path,
                        "5. Готов к обслуживанию");

                // 5. Готов к обслуживанию []
                return new StateMachineConfig(
                        "5. Готов к обслуживанию");
```

```
                }

                // transition not found
                return config;
            case e3:

                // 5. Готов к обслуживанию->5. Готов к обслуживанию
                // e3[!o2.x3]/

                fireTransitionCandidate(context, path,
                        "5. Готов к обслуживанию", event,
                        "5. Готов к обслуживанию#5. Готов к
                            обслуживанию#e3#!o2.x3");

                if (!isInputActionCalculated(calculatedInputActions,
                        _o2_x3)) {

                    fireBeforeInputActionExecution(
                            context,
                            path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#!o2.x3",
                            "o2.x3");

                    o2_x3 = o2.x3(context);

                    fireAfterInputActionExecution(
                            context,
                            path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#!o2.x3",
                            "o2.x3", new Boolean(o2_x3));
                }

                if (!o2_x3) {

                    fireTransitionFound(context, path,
                            "5. Готов к обслуживанию", event,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#!o2.x3");

                    fireComeToState(context, path,
                            "5. Готов к обслуживанию");

                    // 5. Готов к обслуживанию []
                    return new StateMachineConfig(
                            "5. Готов к обслуживанию");

                }
                // 5. Готов к обслуживанию->5. Готов к обслуживанию
                // e3[o2.x3]/o2.z4,o1.z4

                fireTransitionCandidate(context, path,
                        "5. Готов к обслуживанию", event,
                        "5. Готов к обслуживанию#5. Готов к
                            обслуживанию#e3#o2.x3");

                if (o2_x3) {

                    fireTransitionFound(context, path,
                            "5. Готов к обслуживанию", event,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#o2.x3");

                    fireBeforeOutputActionExecution(
                            context,
                            path,
```

```
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#o2.x3",
                        "o2.z4");

                    o2.z4(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#o2.x3",
                            "o2.z4");
                    fireBeforeOutputActionExecution(
                            context,
                            path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#o2.x3",
                            "o1.z4");

                    o1.z4(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
"5. Готов к обслуживанию#5. Готов к обслуживанию#e3#o2.x3",
                            "o1.z4");

                    fireComeToState(context, path,
                            "5. Готов к обслуживанию");

                    // 5. Готов к обслуживанию []
                    return new StateMachineConfig(
                            "5. Готов к обслуживанию");

                }

                // transition not found
                return config;
            case e6:

                // 5. Готов к обслуживанию->4. Прощальное сообщение
                // e6[true]/

                fireTransitionCandidate(context, path,
                        "5. Готов к обслуживанию", event,
                        "5. Готов к обслуживанию#4. Прощальное
                            сообщение#e6#true");

                fireTransitionFound(context, path,
                        "5. Готов к обслуживанию", event,
                        "5. Готов к обслуживанию#4. Прощальное
                            сообщение#e6#true");

                fireComeToState(context, path,
                        "4. Прощальное сообщение");

                // 4. Прощальное сообщение [o1.z13]
                fireBeforeOutputActionExecution(
                        context,
                        path,
                        "5. Готов к обслуживанию#4. Прощальное сообщение#e6#true",
                        "o1.z13");

                o1.z13(context);

                fireAfterOutputActionExecution(
                        context,
```

```
                            path,
                            "5. Готов к обслуживанию#4. Прощальное сообщение#e6#true",
                            "o1.z13");
                    return new StateMachineConfig("4. Прощальное сообщение");

                default:

                    // transition not found
                    return config;
                }

            default:
                throw new EventProcessorException(
                        "Incorrect stable state ["
                                + config.getActiveState()
                                + "] in state machine [A1]");
            }
        }
    }

    // o2.x3
    private static final int _o2_x3 = 0;

    // o2.x1
    private static final int _o2_x1 = 1;

}

private class A2EventProcessor {

    // states
    private static final int Top = 1;

    private static final int _0__Начальное_состояние = 2;

    private static final int _1__Выбор_валюты = 3;

    private static final int _2__Вычисление_курса = 4;

    private static final int _5__Выдача_денег = 5;

    private static final int _4__Печать_чека = 6;

    private static final int _3__Обмен = 7;

    private int decodeState(String state) {

        if ("Top".equals(state)) {
            return Top;
        } else

        if ("0. Начальное состояние".equals(state)) {
            return _0__Начальное_состояние;
        } else

        if ("1. Выбор валюты".equals(state)) {
            return _1__Выбор_валюты;
        } else

        if ("2. Вычисление курса".equals(state)) {
            return _2__Вычисление_курса;
        } else

        if ("5. Выдача денег".equals(state)) {
```

```
        return _5__Выдача_денег;
    } else

    if ("4. Печать чека".equals(state)) {
        return _4__Печать_чека;
    } else

    if ("3. Обмен".equals(state)) {
        return _3__Обмен;
    }

    return -1;
}

// events
private static final int e4 = 1;

private static final int e11 = 2;

private static final int e2 = 3;

private static final int e7 = 4;

private static final int e5 = 5;

private static final int e10 = 6;

private int decodeEvent(String event) {

    if ("e4".equals(event)) {
        return e4;
    } else

    if ("e11".equals(event)) {
        return e11;
    } else

    if ("e2".equals(event)) {
        return e2;
    } else

    if ("e7".equals(event)) {
        return e7;
    } else

    if ("e5".equals(event)) {
        return e5;
    } else

    if ("e10".equals(event)) {
        return e10;
    }

    return -1;
}

private co.Display o1;

private co.PourOutDevice o4;

private co.Client o6;

private co.ExchangeDevice o3;
```

```
private co.CoinMechanism o2;

private void init(ControlledObjectsMap controlledObjectsMap) {
    o1 = (co.Display) controlledObjectsMap.getControlledObject("o1");
    o4 = (co.PourOutDevice) controlledObjectsMap
            .getControlledObject("o4");
    o6 = (co.Client) controlledObjectsMap.getControlledObject("o6");
    o3 = (co.ExchangeDevice) controlledObjectsMap
            .getControlledObject("o3");
    o2 = (co.CoinMechanism) controlledObjectsMap
            .getControlledObject("o2");
}

private StateMachineConfig process(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {
    config = lookForTransition(event, context, path, config);

    config = transiteToStableState(context, path, config);

    // execute included state machines
    executeSubmachines(event, context, path, config);

    return config;
}

private void executeSubmachines(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {
    int state = decodeState(config.getActiveState());

    while (true) {
        switch (state) {
        case _0__Начальное_состояние:

            return;
        case _1__Выбор_валюты:

            return;
        case _2__Вычисление_курса:

            return;
        case _5__Выдача_денег:

            return;
        case _4__Печать_чека:

            return;
        case _3__Обмен:

            return;
        default:
            throw new EventProcessorException("State with name ["
                    + config.getActiveState()
                    + "] is unknown for state machine [A2]");
        }
    }
}

private StateMachineConfig transiteToStableState(
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {

    int s = decodeState(config.getActiveState());
```

```
    Event event;

    switch (s) {
    case Top:

        fireComeToState(context, path, "0. Начальное состояние");

        // 0. Начальное состояние->1. Выбор валюты [true]/o3.z0
        event = Event.NO_EVENT;
        fireTransitionFound(context, path, "0. Начальное состояние",
                event, "0. Начальное состояние#1. Выбор валюты##true");

        fireBeforeOutputActionExecution(context, path,
                "0. Начальное состояние#1. Выбор валюты##true", "o3.z0");

        o3.z0(context);

        fireAfterOutputActionExecution(context, path,
                "0. Начальное состояние#1. Выбор валюты##true", "o3.z0");

        fireComeToState(context, path, "1. Выбор валюты");

        // 1. Выбор валюты []

        return new StateMachineConfig("1. Выбор валюты");
    }

    return config;
}

private StateMachineConfig lookForTransition(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {

    boolean

    o2_x3 = false,

    o3_x1 = false,

    o3_x3 = false,

    o3_x4 = false;

    BitSet calculatedInputActions = new BitSet(4);

    int s = decodeState(config.getActiveState());
    int e = decodeEvent(event.getName());

    while (true) {
        switch (s) {
        case _1__Выбор_валюты:

            switch (e) {
            case e2:

                // 1. Выбор валюты->1. Выбор валюты e2[o2.x3 &&
                // o3.x1]/o1.z7,o1.z12

                fireTransitionCandidate(context, path,
                        "1. Выбор валюты", event,
                        "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1");

                if (!isInputActionCalculated(calculatedInputActions,
```

```
        _o2_x3)) {

    fireBeforeInputActionExecution(
            context,
            path,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
            "o2.x3");

    o2_x3 = o2.x3(context);

    fireAfterInputActionExecution(
            context,
            path,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
            "o2.x3", new Boolean(o2_x3));
}
if (!isInputActionCalculated(calculatedInputActions,
        _o3_x1)) {

    fireBeforeInputActionExecution(
            context,
            path,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
            "o3.x1");

    o3_x1 = o3.x1(context);

    fireAfterInputActionExecution(
            context,
            path,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
            "o3.x1", new Boolean(o3_x1));
}

if (o2_x3 && o3_x1) {

    fireTransitionFound(context, path,
            "1. Выбор валюты", event,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1");

    fireBeforeOutputActionExecution(
            context,
            path,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
            "o1.z7");

    o1.z7(context);

    fireAfterOutputActionExecution(
            context,
            path,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
            "o1.z7");
    fireBeforeOutputActionExecution(
            context,
            path,
            "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
            "o1.z12");

    o1.z12(context);

    fireAfterOutputActionExecution(
            context,
            path,
```

```
                "1. Выбор валюты#1. Выбор валюты#e2#o2.x3 && o3.x1",
                "o1.z12");

        fireComeToState(context, path, "1. Выбор валюты");

        // 1. Выбор валюты []
        return new StateMachineConfig("1. Выбор валюты");

    }
    // 1. Выбор валюты->1. Выбор валюты e2[!o2.x3]/

    fireTransitionCandidate(context, path,
            "1. Выбор валюты", event,
            "1. Выбор валюты#1. Выбор валюты#e2#!o2.x3");

    if (!o2_x3) {

        fireTransitionFound(context, path,
                "1. Выбор валюты", event,
                "1. Выбор валюты#1. Выбор валюты#e2#!o2.x3");

        fireComeToState(context, path, "1. Выбор валюты");

        // 1. Выбор валюты []
        return new StateMachineConfig("1. Выбор валюты");

    }
    // 1. Выбор валюты->2. Вычисление курса e2[o2.x3 &&
    // !o3.x1]/

    fireTransitionCandidate(context, path,
            "1. Выбор валюты", event,
            "1. Выбор валюты#2. Вычисление курса#e2#o2.x3 && !o3.x1");

    if (o2_x3 && !o3_x1) {

        fireTransitionFound(context, path,
                "1. Выбор валюты", event,
                "1. Выбор валюты#2. Вычисление курса#e2#o2.x3 &&
                        !o3.x1");

        fireComeToState(context, path,
                "2. Вычисление курса");

        // 2. Вычисление курса [o3.z7]
        fireBeforeOutputActionExecution(
                context,
                path,
                "1. Выбор валюты#2. Вычисление курса#e2#o2.x3 &&
                !o3.x1",
                "o3.z7");

        o3.z7(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "1. Выбор валюты#2. Вычисление курса#e2#o2.x3 &&
                !o3.x1",
                "o3.z7");
        return new StateMachineConfig("2. Вычисление курса");

    }
```

```
            // transition not found
            return config;
        default:

            // transition not found
            return config;
        }

    case _2__Вычисление_курса:

        switch (e) {
        case e7:

            // 2. Вычисление курса->3. Обмен e7[o3.x4]/o1.z18

            fireTransitionCandidate(context, path,
                    "2. Вычисление курса", event,
                    "2. Вычисление курса#3. Обмен#e7#o3.x4");

            if (!isInputActionCalculated(calculatedInputActions,
                    _o3_x4)) {

                fireBeforeInputActionExecution(context, path,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4",
                        "o3.x4");

                o3_x4 = o3.x4(context);

                fireAfterInputActionExecution(context, path,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4",
                        "o3.x4", new Boolean(o3_x4));
            }

            if (o3_x4) {

                fireTransitionFound(context, path,
                        "2. Вычисление курса", event,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4");

                fireBeforeOutputActionExecution(context, path,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4",
                        "o1.z18");

                o1.z18(context);

                fireAfterOutputActionExecution(context, path,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4",
                        "o1.z18");

                fireComeToState(context, path, "3. Обмен");

                // 3. Обмен [o3.z3, o1.z10]
                fireBeforeOutputActionExecution(context, path,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4",
                        "o3.z3");

                o3.z3(context);

                fireAfterOutputActionExecution(context, path,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4",
                        "o3.z3");
                fireBeforeOutputActionExecution(context, path,
                        "2. Вычисление курса#3. Обмен#e7#o3.x4",
                        "o1.z10");
```

```
            o1.z10(context);

            fireAfterOutputActionExecution(context, path,
                    "2. Вычисление курса#3. Обмен#e7#o3.x4",
                    "o1.z10");
            return new StateMachineConfig("3. Обмен");

        }
        // 2. Вычисление курса->3. Обмен e7[!o3.x4]/o1.z19

        fireTransitionCandidate(context, path,
                "2. Вычисление курса", event,
                "2. Вычисление курса#3. Обмен#e7#!o3.x4");

        if (!o3_x4) {

            fireTransitionFound(context, path,
                    "2. Вычисление курса", event,
                    "2. Вычисление курса#3. Обмен#e7#!o3.x4");

            fireBeforeOutputActionExecution(context, path,
                    "2. Вычисление курса#3. Обмен#e7#!o3.x4",
                    "o1.z19");

            o1.z19(context);

            fireAfterOutputActionExecution(context, path,
                    "2. Вычисление курса#3. Обмен#e7#!o3.x4",
                    "o1.z19");

            fireComeToState(context, path, "3. Обмен");

            // 3. Обмен [o3.z3, o1.z10]
            fireBeforeOutputActionExecution(context, path,
                    "2. Вычисление курса#3. Обмен#e7#!o3.x4",
                    "o3.z3");

            o3.z3(context);

            fireAfterOutputActionExecution(context, path,
                    "2. Вычисление курса#3. Обмен#e7#!o3.x4",
                    "o3.z3");
            fireBeforeOutputActionExecution(context, path,
                    "2. Вычисление курса#3. Обмен#e7#!o3.x4",
                    "o1.z10");

            o1.z10(context);

            fireAfterOutputActionExecution(context, path,
                    "2. Вычисление курса#3. Обмен#e7#!o3.x4",
                    "o1.z10");
            return new StateMachineConfig("3. Обмен");

        }

        // transition not found
        return config;
    default:

        // transition not found
        return config;
    }
```

```
case _5__Выдача_денег:

    switch (e) {
    case e4:

        // 5. Выдача денег->1. Выбор валюты e4[true]/o1.z25

        fireTransitionCandidate(context, path,
                "5. Выдача денег", event,
                "5. Выдача денег#1. Выбор валюты#e4#true");

        fireTransitionFound(context, path, "5. Выдача денег",
                event,
                "5. Выдача денег#1. Выбор валюты#e4#true");

        fireBeforeOutputActionExecution(context, path,
                "5. Выдача денег#1. Выбор валюты#e4#true",
                "o1.z25");

        o1.z25(context);

        fireAfterOutputActionExecution(context, path,
                "5. Выдача денег#1. Выбор валюты#e4#true",
                "o1.z25");

        fireComeToState(context, path, "1. Выбор валюты");

        // 1. Выбор валюты []
        return new StateMachineConfig("1. Выбор валюты");

    default:

        // transition not found
        return config;
    }

case _4__Печать_чека:

    switch (e) {
    case e11:

        // 4. Печать чека->5. Выдача денег e11[true]/

        fireTransitionCandidate(context, path,
                "4. Печать чека", event,
                "4. Печать чека#5. Выдача денег#e11#true");

        fireTransitionFound(context, path, "4. Печать чека",
                event,
                "4. Печать чека#5. Выдача денег#e11#true");

        fireComeToState(context, path, "5. Выдача денег");

        // 5. Выдача денег [o4.z1, o1.z6]
        fireBeforeOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e11#true",
                "o4.z1");

        o4.z1(context);

        fireAfterOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e11#true",
                "o4.z1");
        fireBeforeOutputActionExecution(context, path,
```

```
                "4. Печать чека#5. Выдача денег#e11#true",
                "o1.z6");

        o1.z6(context);

        fireAfterOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e11#true",
                "o1.z6");
        return new StateMachineConfig("5. Выдача денег");

    case e10:

        // 4. Печать чека->5. Выдача денег e10[true]/o4.z6

        fireTransitionCandidate(context, path,
                "4. Печать чека", event,
                "4. Печать чека#5. Выдача денег#e10#true");

        fireTransitionFound(context, path, "4. Печать чека",
                event,
                "4. Печать чека#5. Выдача денег#e10#true");

        fireBeforeOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e10#true",
                "o4.z6");

        o4.z6(context);

        fireAfterOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e10#true",
                "o4.z6");

        fireComeToState(context, path, "5. Выдача денег");

        // 5. Выдача денег [o4.z1, o1.z6]
        fireBeforeOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e10#true",
                "o4.z1");

        o4.z1(context);

        fireAfterOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e10#true",
                "o4.z1");
        fireBeforeOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e10#true",
                "o1.z6");

        o1.z6(context);

        fireAfterOutputActionExecution(context, path,
                "4. Печать чека#5. Выдача денег#e10#true",
                "o1.z6");
        return new StateMachineConfig("5. Выдача денег");

    default:

        // transition not found
        return config;
    }

case _3__Обмен:

    switch (e) {
```

```
case e5:

    // 3. Обмен->1. Выбор валюты e5[!o3.x3]/o1.z7

    fireTransitionCandidate(context, path, "3. Обмен",
            event, "3. Обмен#1. Выбор валюты#e5#!o3.x3");

    if (!isInputActionCalculated(calculatedInputActions,
            _o3_x3)) {

        fireBeforeInputActionExecution(context, path,
                "3. Обмен#1. Выбор валюты#e5#!o3.x3",
                "o3.x3");

        o3_x3 = o3.x3(context);

        fireAfterInputActionExecution(context, path,
                "3. Обмен#1. Выбор валюты#e5#!o3.x3",
                "o3.x3", new Boolean(o3_x3));
    }

    if (!o3_x3) {

        fireTransitionFound(context, path, "3. Обмен",
                event, "3. Обмен#1. Выбор валюты#e5#!o3.x3");

        fireBeforeOutputActionExecution(context, path,
                "3. Обмен#1. Выбор валюты#e5#!o3.x3",
                "o1.z7");

        o1.z7(context);

        fireAfterOutputActionExecution(context, path,
                "3. Обмен#1. Выбор валюты#e5#!o3.x3",
                "o1.z7");

        fireComeToState(context, path, "1. Выбор валюты");

        // 1. Выбор валюты []
        return new StateMachineConfig("1. Выбор валюты");

    }
    // 3. Обмен->4. Печать чека e5[!o3.x1 &&
    // o3.x3]/o2.z3,o4.z5

    fireTransitionCandidate(context, path, "3. Обмен",
            event,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3");

    if (!isInputActionCalculated(calculatedInputActions,
            _o3_x1)) {

        fireBeforeInputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
                "o3.x1");

        o3_x1 = o3.x1(context);

        fireAfterInputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
```

```
            "o3.x1", new Boolean(o3_x1));
}

if (!o3_x1 && o3_x3) {

    fireTransitionFound(context, path, "3. Обмен",
            event,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3");

    fireBeforeOutputActionExecution(
            context,
            path,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
            "o2.z3");

    o2.z3(context);

    fireAfterOutputActionExecution(
            context,
            path,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
            "o2.z3");
    fireBeforeOutputActionExecution(
            context,
            path,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
            "o4.z5");

    o4.z5(context);

    fireAfterOutputActionExecution(
            context,
            path,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
            "o4.z5");

    fireComeToState(context, path, "4. Печать чека");

    // 4. Печать чека [o1.z11]
    fireBeforeOutputActionExecution(
            context,
            path,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
            "o1.z11");

    o1.z11(context);

    fireAfterOutputActionExecution(
            context,
            path,
            "3. Обмен#4. Печать чека#e5#!o3.x1 && o3.x3",
            "o1.z11");
    return new StateMachineConfig("4. Печать чека");

}
// 3. Обмен->4. Печать чека e5[o3.x1 &&
// o3.x3]/o1.z12,o2.z3,o4.z5

fireTransitionCandidate(context, path, "3. Обмен",
        event,
        "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3");

if (o3_x1 && o3_x3) {
```

```
        fireTransitionFound(context, path, "3. Обмен",
                event,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3");

        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o1.z12");

        o1.z12(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o1.z12");
        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o2.z3");

        o2.z3(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o2.z3");
        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o4.z5");

        o4.z5(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o4.z5");

        fireComeToState(context, path, "4. Печать чека");

        // 4. Печать чека [o1.z11]
        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o1.z11");

        o1.z11(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "3. Обмен#4. Печать чека#e5#o3.x1 && o3.x3",
                "o1.z11");
        return new StateMachineConfig("4. Печать чека");

    }
```

```
                    // transition not found
                    return config;
                default:

                    // transition not found
                    return config;
                }

            default:
                throw new EventProcessorException(
                        "Incorrect stable state ["
                                + config.getActiveState()
                                + "] in state machine [A2]");
            }
        }
    }

    // o2.x3
    private static final int _o2_x3 = 0;

    // o3.x1
    private static final int _o3_x1 = 1;

    // o3.x3
    private static final int _o3_x3 = 2;

    // o3.x4
    private static final int _o3_x4 = 3;

}

private class A3EventProcessor {

    // states
    private static final int Top = 1;

    private static final int _2__Авторизация = 2;

    private static final int _1__Ожидание_запроса = 3;

    private static final int _3__Получение_курса = 4;

    private static final int _5__Ответ_на_запрос = 5;

    private static final int s1 = 6;

    private static final int _4__Информирование_об_окончании_валюты = 7;

    private static final int s3 = 8;

    private int decodeState(String state) {

        if ("Top".equals(state)) {
            return Top;
        } else

        if ("2. Авторизация".equals(state)) {
            return _2__Авторизация;
        } else

        if ("1. Ожидание запроса".equals(state)) {
            return _1__Ожидание_запроса;
        } else
```

```
    if ("3. Получение курса".equals(state)) {
        return _3__Получение_курса;
    } else

    if ("5. Ответ на запрос".equals(state)) {
        return _5__Ответ_на_запрос;
    } else

    if ("s1".equals(state)) {
        return s1;
    } else

    if ("4. Информирование об окончании валюты".equals(state)) {
        return _4__Информирование_об_окончании_валюты;
    } else

    if ("s3".equals(state)) {
        return s3;
    }

    return -1;
}

// events
private static final int e13 = 1;

private static final int e15 = 2;

private static final int e17 = 3;

private static final int e16 = 4;

private int decodeEvent(String event) {

    if ("e13".equals(event)) {
        return e13;
    } else

    if ("e15".equals(event)) {
        return e15;
    } else

    if ("e17".equals(event)) {
        return e17;
    } else

    if ("e16".equals(event)) {
        return e16;
    }

    return -1;
}

private co.Server o5;

private void init(ControlledObjectsMap controlledObjectsMap) {
    o5 = (co.Server) controlledObjectsMap.getControlledObject("o5");
}

private StateMachineConfig process(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {
    config = lookForTransition(event, context, path, config);
```

```
    config = transiteToStableState(context, path, config);

    // execute included state machines
    executeSubmachines(event, context, path, config);

    return config;
}

private void executeSubmachines(Event event,
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {
    int state = decodeState(config.getActiveState());

    while (true) {
        switch (state) {
        case _2__Авторизация:

            return;
        case _1__Ожидание_запроса:

            return;
        case _3__Получение_курса:

            return;
        case _5__Ответ_на_запрос:

            return;
        case s1:

            return;
        case _4__Информирование_об_окончании_валюты:

            return;
        case s3:

            return;
        default:
            throw new EventProcessorException("State with name ["
                    + config.getActiveState()
                    + "] is unknown for state machine [A3]");
        }
    }
}

private StateMachineConfig transiteToStableState(
        StateMachineContext context, StateMachinePath path,
        StateMachineConfig config) throws Exception {

    int s = decodeState(config.getActiveState());
    Event event;

    switch (s) {
    case Top:

        fireComeToState(context, path, "s1");

        // s1->1. Ожидание запроса [true]/
        event = Event.NO_EVENT;
        fireTransitionFound(context, path, "s1", event,
                "s1#1. Ожидание запроса##true");

        fireComeToState(context, path, "1. Ожидание запроса");

        // 1. Ожидание запроса [o5.z0]
```

```
            fireBeforeOutputActionExecution(context, path,
                    "s1#1. Ожидание запроса##true", "o5.z0");

            o5.z0(context);

            fireAfterOutputActionExecution(context, path,
                    "s1#1. Ожидание запроса##true", "o5.z0");

            return new StateMachineConfig("1. Ожидание запроса");
        }

        return config;
    }

    private StateMachineConfig lookForTransition(Event event,
            StateMachineContext context, StateMachinePath path,
            StateMachineConfig config) throws Exception {

        BitSet calculatedInputActions = new BitSet(0);

        int s = decodeState(config.getActiveState());
        int e = decodeEvent(event.getName());

        while (true) {
            switch (s) {
            case _2__Авторизация:

                switch (e) {
                case e16:

                    // 2. Авторизация->5. Ответ на запрос e16[true]/

                    fireTransitionCandidate(context, path,
                            "2. Авторизация", event,
                            "2. Авторизация#5. Ответ на запрос#e16#true");

                    fireTransitionFound(context, path, "2. Авторизация",
                            event,
                            "2. Авторизация#5. Ответ на запрос#e16#true");

                    fireComeToState(context, path, "5. Ответ на запрос");

                    // 5. Ответ на запрос [o5.z4]
                    fireBeforeOutputActionExecution(context, path,
                            "2. Авторизация#5. Ответ на запрос#e16#true",
                            "o5.z4");

                    o5.z4(context);

                    fireAfterOutputActionExecution(context, path,
                            "2. Авторизация#5. Ответ на запрос#e16#true",
                            "o5.z4");
                    return new StateMachineConfig("5. Ответ на запрос");

                default:

                    // transition not found
                    return config;
                }

            case _1__Ожидание_запроса:

                switch (e) {
                case e13:
```

```
                    // 1. Ожидание запроса->3. Получение курса e13[true]/

                    fireTransitionCandidate(context, path,
                            "1. Ожидание запроса", event,
                            "1. Ожидание запроса#3. Получение курса#e13#true");

                    fireTransitionFound(context, path,
                            "1. Ожидание запроса", event,
                            "1. Ожидание запроса#3. Получение курса#e13#true");

                    fireComeToState(context, path, "3. Получение курса");

                    // 3. Получение курса [o5.z1]
                    fireBeforeOutputActionExecution(
                            context,
                            path,
                            "1. Ожидание запроса#3. Получение курса#e13#true",
                            "o5.z1");

                    o5.z1(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
                            "1. Ожидание запроса#3. Получение курса#e13#true",
                            "o5.z1");
                    return new StateMachineConfig("3. Получение курса");

                case e15:

                    // 1. Ожидание запроса->4. Информирование об окончании
                    // валюты e15[true]/

                    fireTransitionCandidate(context, path,
                            "1. Ожидание запроса", event,
  "1. Ожидание запроса#4. Информирование об окончании валюты#e15#true");

                    fireTransitionFound(context, path,
                            "1. Ожидание запроса", event,
  "1. Ожидание запроса#4. Информирование об окончании валюты#e15#true");

                    fireComeToState(context, path,
                            "4. Информирование об окончании валюты");

                    // 4. Информирование об окончании валюты [o5.z3]
                    fireBeforeOutputActionExecution(
                            context,
                            path,
"1. Ожидание запроса#4. Информирование об окончании валюты#e15#true",
                            "o5.z3");

                    o5.z3(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
"1. Ожидание запроса#4. Информирование об окончании валюты#e15#true",
                            "o5.z3");
                    return new StateMachineConfig(
                            "4. Информирование об окончании валюты");

                case e17:
```

```
            // 1. Ожидание запроса->2. Авторизация e17[true]/

            fireTransitionCandidate(context, path,
                    "1. Ожидание запроса", event,
                    "1. Ожидание запроса#2. Авторизация#e17#true");

            fireTransitionFound(context, path,
                    "1. Ожидание запроса", event,
                    "1. Ожидание запроса#2. Авторизация#e17#true");

            fireComeToState(context, path, "2. Авторизация");

            // 2. Авторизация [o5.z5]
            fireBeforeOutputActionExecution(context, path,
                    "1. Ожидание запроса#2. Авторизация#e17#true",
                    "o5.z5");

            o5.z5(context);

            fireAfterOutputActionExecution(context, path,
                    "1. Ожидание запроса#2. Авторизация#e17#true",
                    "o5.z5");
            return new StateMachineConfig("2. Авторизация");

        default:

            // transition not found
            return config;
        }

    case _3__Получение_курса:

        switch (e) {
        case e16:

            // 3. Получение курса->5. Ответ на запрос e16[true]/

            fireTransitionCandidate(context, path,
                    "3. Получение курса", event,
                    "3. Получение курса#5. Ответ на запрос#e16#true");

            fireTransitionFound(context, path,
                    "3. Получение курса", event,
                    "3. Получение курса#5. Ответ на запрос#e16#true");

            fireComeToState(context, path, "5. Ответ на запрос");

            // 5. Ответ на запрос [o5.z4]
            fireBeforeOutputActionExecution(
                    context,
                    path,
                    "3. Получение курса#5. Ответ на запрос#e16#true",
                    "o5.z4");

            o5.z4(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "3. Получение курса#5. Ответ на запрос#e16#true",
                    "o5.z4");
            return new StateMachineConfig("5. Ответ на запрос");

        default:
```

```
                // transition not found
                return config;
            }

        case _5__Ответ_на_запрос:

            switch (e) {
            default:

                // 5. Ответ на запрос->s3 *[true]/

                fireTransitionCandidate(context, path,
                        "5. Ответ на запрос", event,
                        "5. Ответ на запрос#s3#*#true");

                fireTransitionFound(context, path,
                        "5. Ответ на запрос", event,
                        "5. Ответ на запрос#s3#*#true");

                fireComeToState(context, path, "s3");

                // s3 []
                return new StateMachineConfig("s3");

            }

        case _4__Информирование_об_окончании_валюты:

            switch (e) {
            case e16:

                // 4. Информирование об окончании валюты->s3 e16[true]/

                fireTransitionCandidate(context, path,
                        "4. Информирование об окончании валюты", event,
                        "4. Информирование об окончании валюты#s3#e16#true");

                fireTransitionFound(context, path,
                        "4. Информирование об окончании валюты", event,
                        "4. Информирование об окончании валюты#s3#e16#true");

                fireComeToState(context, path, "s3");

                // s3 []
                return new StateMachineConfig("s3");

            default:

                // transition not found
                return config;
            }

        default:
            throw new EventProcessorException(
                    "Incorrect stable state ["
                            + config.getActiveState()
                            + "] in state machine [A3]");
        }
    }
}

}
```

```
 private static boolean isInputActionCalculated(
         BitSet calculatedInputActions, int k) {
     boolean b = calculatedInputActions.get(k);

     if (!b) {
         calculatedInputActions.set(k);
     }

     return b;
 }

}
```