Санкт-Петербургский государственный университет информационных технологий, механики и оптики

Кафедра «Компьютерные технологии»

А.Х.Киракозов, А.А.Шалыто, Б.Р.Яминов

Система управления автомобильной сигнализацией

Программирование с явным выделением состояний

Проектная документация

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

Санкт-Петербург

2006

Оглавление

1. Введение

В данной работе приведен пример применения автоматного подхода для проектирования автомобильной сигнализации с использованием среды разработки *Unimod*.

Для алгоритмизации и программирования задач логического управления техническими объектами удобен автоматный подход, поскольку граф переходов построенного автомата наглядно представляет работу объекта. Это приводит к тому, что можно легко увидеть возможные ошибки в проектировании, такие как отсутствие некоторого перехода, недоступность состояния и т.д.

Использование среды разработки *Unimod* позволяет эффективно использовать автоматный подход. При его применении сначала в общем случае строится система взаимосвязанных автоматов, а затем на языке *Java* программируются функции входных и выходных воздействий. Таким образом, программа разделяется на независимые блоки, что позволяет облегчить ее написание и уменьшает вероятность возникновения ошибок.

2. Постановка задачи

Задача состоит в том, чтобы спроектировать и реализовать работу простой автомобильной сигнализации. Ее пример взят из руководства к сигнализации [1].

Пульт управления сигнализацией (брелок) содержит три кнопки «1», «2» и «3». Первая кнопка применяется для постановки автомобиля на сигнализацию. Вторая – для снятия с сигнализации. Третья кнопка позволяет использовать дополнительные функции. Также сигнализация обладает двухуровневым датчиком удара и светодиодом, расположенным в салоне автомобиля. По его миганию можно определить текущее состояние сигнализации.

При постановке на сигнализацию фары автомобиля мигают один раз, а сирена выдает короткий гудок. При снятии сигнализации – два мигания и два гудка. Если сигнализация включена, и датчик удара зарегистрировал слабый удар, то выдаются три коротких мигания и три гудка, которые символизируют, что на машину получила слабое воздействие, которое может иметь безобидный характер (например, рядом проезжающий трамвай). Если же срабатываает датчик сильного удара включается полная тревога: сирена гудит, и фары мигают в течение 15 с.

Если после слабого удара в течение пяти секунд последовал еще один удар, то сигнализация также переходит в состояние полной тревоги.

Третья кнопка используется для «тихой» постановки на сигнализацию (тихий режим) и снятия с нее. Это означает, что при последовательном нажатии (в течение трех секунд) кнопки «3» и кнопки «1», автомобиль будет поставлен на сигнализацию без звукового сопровождения, подтвердив команду лишь миганием фар. Для бесшумного снятия сигнализации необходимо выполнить те же действия, только используя кнопку «2» вместо кнопки «1».

Светодиод в салоне машины мигает, если сигнализация включена, и выключен в противном случае.

На рис.1 изображен пользьовательский интерфейс программы.

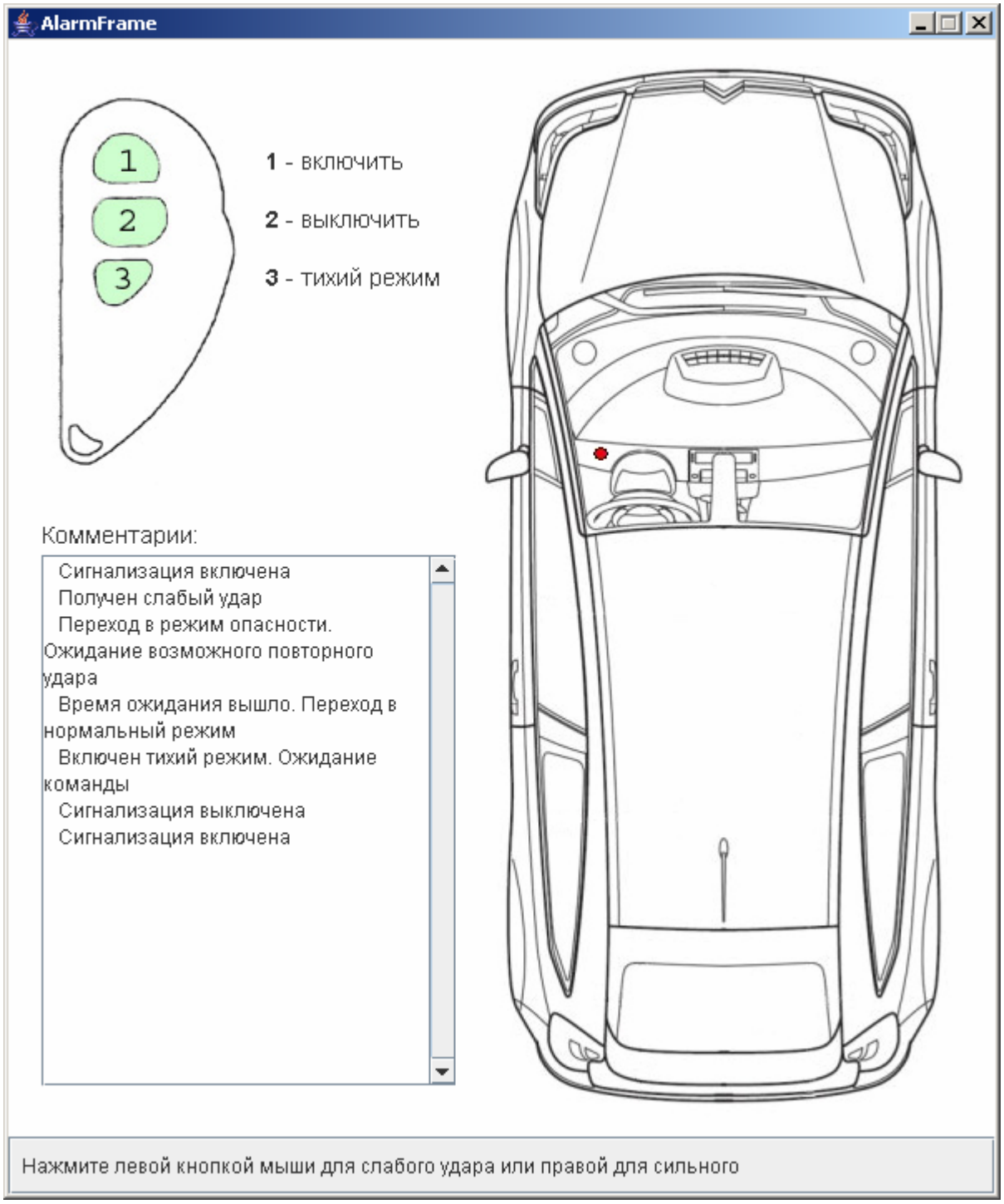

Рис. 1. Пользовательский интерфейс программы

В левом верхнем углу окна находится пульт управления с тремя кнопками, управляющими сигнализацией. Воздействия на машину производятся с помощью мыши: при нажатии на изображение автомобиля левой кнопкой, машина получает слабый удар, нажатие правой кнопкой моделирует сильный удар.

В нижней части окна выводится подсказка для выполнения действий над автомобилем или одной из трех кнопок, соответствующих положению указателя мыши. На рис. 1 указатель мыши находится над машиной.

Под лобовым стеклом на панели виден красный светодиод, он мигает при включенной сигнализации.

В область «Комментарии» выводятся комментарии к действиям, производимым системой. В данном примере был выполнен следующий набор

действий: сигнализацию включили нажатием кнопки «1», затем произвели слабый удар (нажатием левой кнопки мыши над машиной), спустя некоторое время нажали подряд кнопки «3» и «2». Затем опять нажали кнопку «1».

3. Формальный текст сценария

Жизненный цикл работы сигнализации можно разбить на несколько состояний, в которых сигнализация ждет различные события. Событиями могут быть нажатия кнопок на брелке сигнализации, получение автомобилем ударов и т.д.

Опишем сценарий формально. Состояния будем обозначать числами `i`, а переходы между состояниями парами чисел (`i`, `j`), где `i` – номер состояния, из которого осуществляется переход, а `j` – номер состояния, в который выполняется переход.

1. Сигнализация отключена

Исходно сигнализация машины отключена.

(1, 2) Нажатие кнопки «1», которое соответствует постановке машины на сигнализацию (событие `e11`). Фары автомобиля мигают один раз (входное воздействие `o1.z1`), красный светодиод сигнализации, расположенный на панели автомобиля, начинает мигать (`o4.z1`), выключаются все запущенные таймеры (`o2.z4`), разрешается подача звуковых сигналов (`o3.z6`). Если звук был включен, то подается звуковой сигнал (выходное воздействие `o3.z1`), в противном случае прерываются все другие звуковые сигналы (`o3.z5`).

2. Сигнализация включена

(2, 3) Получен слабый удар, сигнализация переходит в состояние опасности (`e21`). Фары автомобиля мигают три раза (`o1.z3`), подается звуковой сигнал (`o3.z3`). Запускается таймер «2» на 5 с (`o2.z2`). Если в течение 5 с автомобиль получит еще один удар, то сигнализация перейдет в режим опасности.

(2, 4) Получен сильный удар – сигнализация переходит в состояние тревоги (`e22`). Фары автомобиля начинают мигать (`o1.z4`), сирена начинает подавать тревожный сигнал (`o3.z4`), таймер «1» запускается на 15 с (`o2.z1`). В течение этого времени сигнализация будет подавать тревожные сигналы.

3. Опасность

(3, 4) Получен слабый удар (`e21`) или сильный удар (`e22`). Сигнализация переходит в состояние тревоги. Фары автомобиля начинают мигать (`o1.z4`), сирена подает тревожный сигнал (`o3.z4`), таймер «1» запускается на 15 с (`o2.z1`). В течение этого времени сигнализация подает тревожные сигналы.

(3, 2) Таймер «2» для отсчета времени в состоянии опасности закончил свою работу, сигнализация переходит в обычный режим (`e32`).

4. Тревога

(4, 2) Таймер «1» для отсчета времени в состоянии тревоги закончил свою работу, сигнализация переходит в обычный режим (`e33`). Фары автомобиля прекращают мигать (`o1.z5`), сигнализация перестает подавать сигналы (`o3.z5`)

4. Автоматная реализация

Описанную систему логично разделить на объекты следующим образом. Пульт управления (брелок) и датчик удара представить как генераторы событий, а фары, сирену и светодиод – как объекты управления.

Схема связей системы представлена на рис. 2.

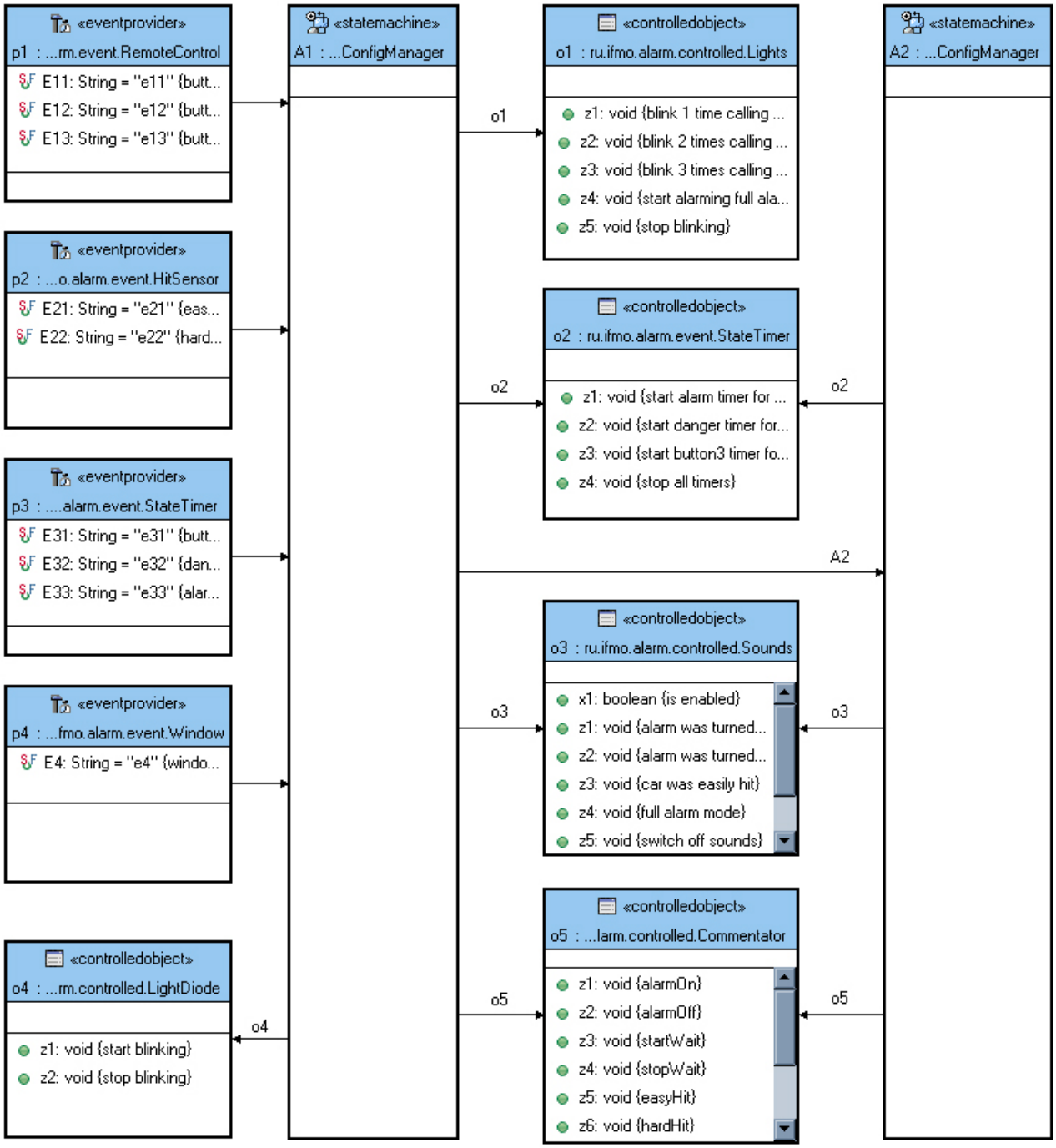

Рис. 2. Схема связей

Кроме перечисленных объектов в системе имеется генератор событий, являющийся таймером, и объект управления, который может его запускать на указанное время. Также имеется объект управления, который выводит комментарии.

4.1. Автоматы

Как следует из рассмотрения схемы связей, в системе имеется два автомата: *A1* и *A2*. Первый из них является главным, а второй играет вспомогательную роль. Он вложен в определенные состояния первого.

Использование вложенных автоматов оправдано тем, что это сильно упрощает диаграммы автоматов и уменьшает количество переходов.

На рис. 3 изображен автомат *A1*.

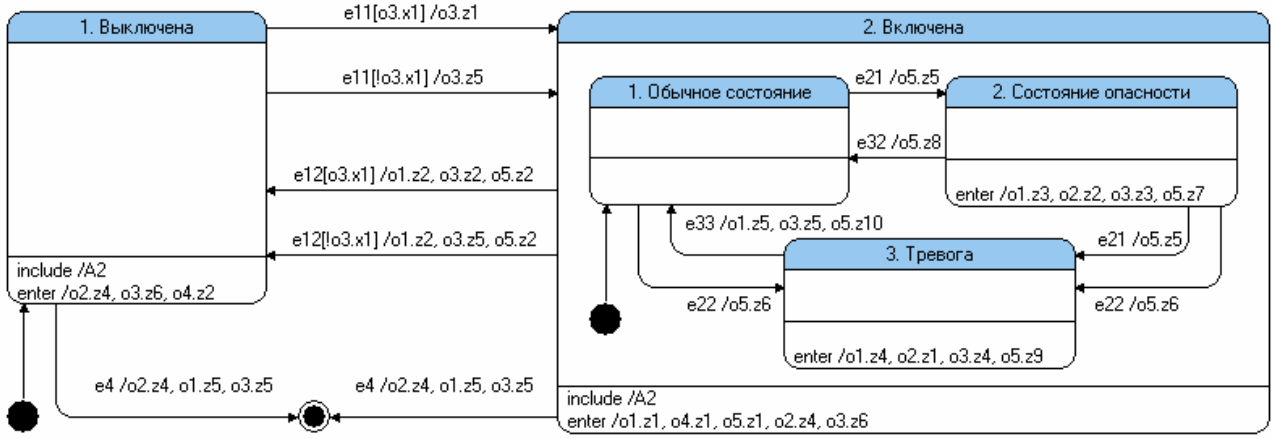

Рис. 3. Автомат *A1*

Основными состояниями автомата являются состояния «*1. Выключена*» и «*2. Включена*». Первое является начальным. Переходы между этими состояниями производятся по событиям, соответствующим нажатию кнопок «1» и «2». При этом если звук включен, то переход выполняется со звуковым подтверждением, а иначе без него. Включением и выключением звука управляет автомат *A2*, вложенный в оба основных состояния.

Второе состояние является сложным. Оно реализует систему, управляющую сигнализацией во включенном режиме. При получении слабого удара осуществляется переход из состояния «*1. Обычное состояние*» в состояние «*2. Состояние опасности*». Если в течение последующих пяти секунд не будет никакого удара, то система вернется обратно в первое из вложенных состояний. В противном случае, будет выполнен переход в состояние «*3. Тревога*», и на 15 с будут включены сирена и фары.

Использование сложного состояния обеспечивает возможность выключения сигнализации в каком бы состоянии не находился автомат.

Конечное состояние достигается при закрытии приложения.

Автомат *A2* управляет включением и выключением звука. Включенное состояние звука означает, что если понадобится воспроизвести некоторый звук, это будет сделано. В выключенном состоянии никакие звуки воспроизводиться не могут. Схема автомата изображена на рис. 4.

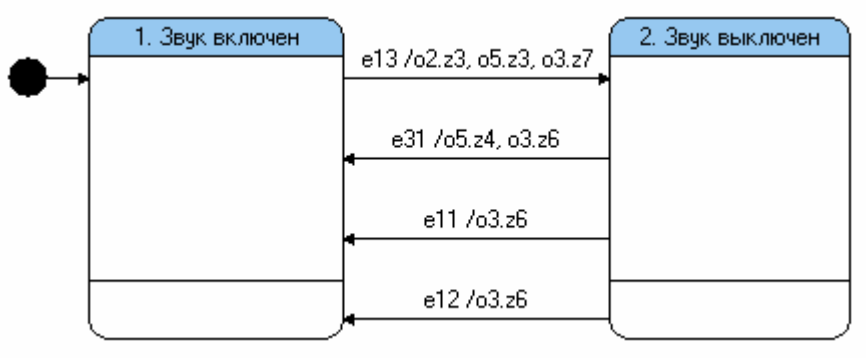

Рис. 4 Автомат *A2*

Звук выключается при нажатии кнопки «3». При этом запускается таймер на три секунды, по истечении этого времени автомат *A2* возвращается из состояния «*2. Звук выключен*» в состояние «*1. Звук включен*». Также звук включается после нажатия любой из кнопок «1» или «2».

4.2. Генераторы событий

Далее будет описан физический смысл и описание генераторов событий данной системы.

Генератор событий *p1*

Этот объект описывает события, производимые пультом управления сигнализацией. События:

e11 – нажатие кнопки «1»;
e12 – нажатие кнопки «2»;
e13 – нажатие кнопки «3».

Генератор событий *p2*

Этот объект соответствует датчику удара. Он может выдавать два события:

e21 – фиксирование слабого удара;
e22 – фиксирование сильного удара.

Генератор событий *p3*

Этот объект запускает один из трех таймеров. Когда отсчет завершается, он генерирует соответствующее событие. Таймер запускается по запросу объекта управления «*o2*» с указанием номера и требуемого времени.

e31 – таймер «1» завершил отсчет (таймер для управления выключением звука);

e32 – таймер «2» завершил отсчет (таймер для отсчета времени в состоянии «*2. Состояние опасности*» автомата *A2*);

e33 – таймер «3» завершил отсчет (таймер для отсчета времени в состоянии «*3. Тревога*» автомата *A2*).

Генератор событий *p4*

Этот объект генерирует событие, когда пользователь закрывает окно программы.

e4 – закрытие окна рограммы.

4.3. Объекты управления

Объект управления *o1*

Этот объект описывает действия, совершаемые фарами автомобиля.

z1 – мигнуть один раз;
z2 – мигнуть два раза;
z3 – мигнуть три раза;
z4 – начать подавать тревожный сигнал;
z5 – прервать любые действия.

Объект управления *o2*

Этот объект используется для запуска таймера «*p3*».

z1 – запустить таймер «1» на 15 с;
z2 – запустить таймер «2» на 5 с;
z3 – запустить таймер «3» на 3 с;
z4 – остановить все таймеры.

Объект управления *o3*

Данный объект управления отражает работу сирены. Его выходные воздействия практически совпадают с выходными воздействиями фар.

x1 – логическая переменная, показывающая включен звук (то есть может подавать сигналы) или нет;
z1 – генерация звука, соответствующего постановке автомобиля на сигнализацию;
z2 – генерация звука, соответствующего снятию автомобиля с сигнализации;
z3 – генерация звука, соответствующего реакции на слабый удар;
z4 – начать подавать тревожный сигнал;
z5 – прервать звук;
z6 – включить звук (разрешить подавать сигналы);
z7 – выключить звук (запретить подавать сигналы).

Объект управления *o4*

Этот объект управления описывает работу светодиода, расположенного в машине. Он показывает текущее состояние сигнализации.

z1 – начать мигание;
z2 – завершить мигание.

Объект управления *o5*

Данный объект управления используется для вывода комментариев к действиям, производимым над системой и ее реакции на них.

z1 – сигнализация включена;
z2 – сигнализация выключена;

z3 – включен тихий режим, ожидание команды;
z4 – время ожидания вышло, тихий режим выключен;
z5 – получен слабый удар;
z6 – получен сильный удар;
z7 – переход в режим опасности, ожидание возможного повторного удара;
z8 – время ожидания вышло, переход в нормальный режим;
z9 – включена тревога;
z10 – тревога выключена.

5. Реализация программы

Программа реализуется с помощью плагина *Unimod* к среде разработки *Java*-приложений *Eclipse*. Указанный плагин позволяет построить схему связей, а также автоматы, изображенные на рис.2 – 4. После этого вручную реализуются объекты управления, источники событий и интерфейс программы на языке *Java*. Поведение приложения описывается двумя автоматами.

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

5.1. Интерпретационный подход

Этот подход основан на использовании библиотек *Unimod* для работы приложения.

Построенные автоматы сохраняются в формате *XML* (Приложение 2), затем компилируются классы, написанные вручную. После этого запускается приложение, используя библиотеки *Unimod*, *XML*-файл и откомпилированные классы. При работе в протокол выводятся комментарии *Unimod* о происходящих событиях и совершаемых переходах в автомате. Далее приведен пример такого протокола для следующих действий: запустили приложение, нажали кнопку «1» и закрыли окно.

```
18:18:29,890 INFO  [Run] Start event [e11] processing. In state [/A1:Top]
18:18:29,890 INFO  [Run] Transition to go found [s1#1. Выключена##true]
18:18:29,890 INFO  [Run] Start on-enter action [o2.z4] execution
18:18:29,890 INFO  [Run] Finish on-enter action [o2.z4] execution
18:18:29,890 INFO  [Run] Start on-enter action [o3.z6] execution
18:18:30,953 INFO  [Run] Finish on-enter action [o3.z6] execution
18:18:30,968 INFO  [Run] Start on-enter action [o4.z2] execution
18:18:30,984 INFO  [Run] Finish on-enter action [o4.z2] execution
18:18:30,984 DEBUG [Run] Try transition [1. Выключена#2. Включена#e11#!o3.x1]
18:18:31,000 INFO  [Run] Start input action [o3.x1] calculation
18:18:31,000 INFO  [Run] Finish input action [o3.x1] calculation. Its value is
[true]
18:18:31,000 DEBUG [Run] Try transition [1. Выключена#2. Включена#e11#o3.x1]
18:18:31,000 INFO  [Run] Transition to go found [1. Выключена#2.
Включена#e11#o3.x1]
18:18:31,000 INFO  [Run] Start output action [o3.z1] execution
18:18:31,000 INFO  [Run] Finish output action [o3.z1] execution
18:18:31,000 INFO  [Run] Start on-enter action [o1.z1] execution
18:18:31,000 INFO  [Run] Finish on-enter action [o1.z1] execution
18:18:31,000 INFO  [Run] Start on-enter action [o4.z1] execution
18:18:31,000 INFO  [Run] Finish on-enter action [o4.z1] execution
18:18:31,000 INFO  [Run] Start on-enter action [o5.z1] execution
18:18:31,031 INFO  [Run] Finish on-enter action [o5.z1] execution
18:18:31,031 INFO  [Run] Start on-enter action [o2.z4] execution
```

```
18:18:31,031 INFO  [Run] Finish on-enter action [o2.z4] execution
18:18:31,031 INFO  [Run] Start on-enter action [o3.z6] execution
18:18:31,031 INFO  [Run] Finish on-enter action [o3.z6] execution
18:18:31,031 INFO  [Run] Transition to go found [s2#1. Обычное состояние##true]
18:18:31,031 INFO  [Run] Start event [e11] processing. In state [/A1:2.
Включена/A2:Top]
18:18:31,031 INFO  [Run] Transition to go found [s1#1. Звук включен##true]
18:18:31,109 INFO  [Run] Finish event [e11] processing. In state [/A1:2.
Включена/A2:1. Звук включен]
18:18:31,109 INFO  [Run] Finish event [e11] processing. In state [/A1:1. Обычное
состояние]
18:18:33,015 INFO  [Run] Start event [e4] processing. In state [/A1:1. Обычное
состояние]
18:18:33,015 DEBUG [Run] Try transition [2. Включена#s3#e4#true]
18:18:33,015 INFO  [Run] Transition to go found [2. Включена#s3#e4#true]
18:18:33,015 INFO  [Run] Start output action [o2.z4] execution
18:18:33,015 INFO  [Run] Finish output action [o2.z4] execution
18:18:33,015 INFO  [Run] Start output action [o1.z5] execution
18:18:33,015 INFO  [Run] Finish output action [o1.z5] execution
18:18:33,015 INFO  [Run] Start output action [o3.z5] execution
18:18:33,015 INFO  [Run] Finish output action [o3.z5] execution
18:18:33,140 INFO  [Run] State machine came to final state [/A1:s3]
18:18:33,140 INFO  [Run] Finish event [e4] processing. In state [/A1:s3]
```

Интерпретационный подход обладает тем недостатком, что требует подключения большого количества библиотек, которые для работы конкретного приложения не являются необходимыми.

Специализированный протокол для рассматриваемой программы приведен в Приложении 1.

5.2. Компилятивный подход

В этом подходе с помощью средств *Unimod* из *XML*-файла (Приложение 2), описывающего структуру программы и автоматы, генерируется *Java*-файл. Для его работы также необходимы некоторые классы библиотек *Unimod*, однако гораздо меньшее их количество, чем при использовании интерпретационного подхода.

Автоматически сгенерированный файл находится в пакете `ru.ifmo.alarm` и называется `Model1EventProcessor.java`. Опишем его структуру.

```
/**
 * This file was generated from model [Model1] on [Sun Dec 11 22:26:56 MSK
2005].
 * Do not change content of this file.
 */

package ru.ifmo.alarm;

import java.util.*;

import com.evelopers.common.exception.*;
import com.evelopers.unimod.core.stateworks.*;
import com.evelopers.unimod.runtime.*;
import com.evelopers.unimod.runtime.context.*;

public class Model1EventProcessor extends AbstractEventProcessor {
```

В начале файла указано, какие пакеты требуются для работы класса. Затем объявляются некоторые поля, среди которых присутствуют следующие:

```
private A1EventProcessor _A1;
private A2EventProcessor _A2;
```

Эти поля являются экземплярами классов, описывающих автоматы, изображенные на рис. 3, 4. Перейдем к описанию этих классов.

```
private class A1EventProcessor {

    // states
    private static final int Top = 1;
    private static final int s3 = 2;
    private static final int _1__Выключена = 3;
    private static final int _2__Включена = 4;
    private static final int _1__Обычное_состояние = 5;
    private static final int _2__Состояние_опасности = 6;
    private static final int _3__Тревога = 7;
    private static final int s2 = 8;
    private static final int s1 = 9;


    private int decodeState(String state) {
        if ("Top".equals(state)) {
            return Top;
        } else
        if ("s3".equals(state)) {
            return s3;
        } else
        if ("1. Выключена".equals(state)) {
            return _1__Выключена;
        } else
        ...
    }
```

Здесь описаны состояния первого автомата. Состояние `Top` является некоторым внутренним состоянием, используемым инструментальным средством *Unimod* как «нулевое». Состояние `s1` – начальное состояние, `s3` – конечное состояние. Состояние `s2` – начальное состояние внутри состояния «*2. Включена*» (рис. 3). Метод `decodeState` преобразует название состояния в его номер.

Затем описываются события и поля объектов управления.

```
    // events
    private static final int e33 = 1;
    private static final int e4 = 2;
    private static final int e22 = 3;
    private static final int e11 = 4;
    private static final int e32 = 5;
    private static final int e21 = 6;
    private static final int e12 = 7;

    private int decodeEvent(String event) {
        ...
    }

    private ru.ifmo.alarm.controlled.Commentator o5;

    private ru.ifmo.alarm.controlled.LightDiode o4;

    private ru.ifmo.alarm.controlled.Sounds o3;

    private ru.ifmo.alarm.controlled.Lights o1;

    private ru.ifmo.alarm.controlled.StateTimerStarter o2;
```

Следующий метод обрабатывает поступившие события и осуществляет переходы:

```
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;
}
```

Этот метод получает на вход событие (`event`), текущую конфигурацию автомата (`config`) и некоторые дополнительные входные данные. Затем определяется допустимый переход. После этого автомат переходит в новое устойчивое состояние. Начальное состояние `s1` автомата *A1* не является устойчивым, и автомат не может оставаться в нем длительно. После этого, если в новое состояние вложен автомат, то он исполняется.

Вложенный автомат обрабатывает следующий метод:

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

    while (true) {
        switch (state) {
        case s3:

            return;
        case _1__Выключена:
            // 1. Выключена includes A2

            fireBeforeSubmachineExecution(context, event, path,
                    "1. Выключена", "A2");

            Model1EventProcessor.this.process(event, context,
                    new StateMachinePath(path, "1. Выключена","A2"));

            fireAfterSubmachineExecution(context, event, path,
                    "1. Выключена", "A2");

            return;
        case _2__Включена:
            // 2. Включена includes A2

            ...

            return;
        case _1__Обычное_состояние:

            state = _2__Включена;
            break;
        case _2__Состояние_опасности:

            state = _2__Включена;
            break;
        case _3__Тревога:
```

```
                ...
            }
        }
    }
```

Таким образом, если автомат *A1* находится в состоянии «1. Выключена» или «2. Включена», то события передаются во вложенный автомат *A2*. Если же текущее состояние – одно из вложенных в составное состояние «2. Включена», то цикл повторяется для состояния «2. Включена» и также находит автомат *A2*. После этого вложенный автомат *A2* исполняется.

Следующий метод осуществляет переход в устойчивое состояние.

```
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. Выключена [true]/
        event = Event.NO_EVENT;
        fireTransitionFound(context, path, "s1", event,
                "s1#1. Выключена##true");

        fireComeToState(context, path, "1. Выключена");

        // 1. Выключена []

        return new StateMachineConfig("1. Выключена");
    case _2__Включена:
        fireCompositeTargetState(context, path, "2. Включена");

        fireComeToState(context, path, "s2");

        // s2->1. Обычное состояние [true]/
        event = Event.NO_EVENT;
        fireTransitionFound(context, path, "s2", event,
                "s2#1. Обычное состояние##true");

        fireComeToState(context, path, "1. Обычное состояние");

        // 1. Обычное состояние []

        return new StateMachineConfig("1. Обычное состояние");
    }

    return config;
}
```

Если автомат находится в состоянии Top, следовательно, он только начал работу и требуется перейти из начального состояния `s1` в состояние «*1. Выключена*». Если автомат пришел в состояние «*2. Включена*», то требуется перейти среди вложенных в него состояний в устойчивое – в состояние «*1. Обычное состояние*». Это и выполняется в приведенном выше методе.

Следующий метод находит переходы между состояниями. Он является основной частью описания автомата.

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

    boolean

    o3_x1 = false;

    BitSet calculatedInputActions = new BitSet(1);

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

    while (true) {
        switch (s) {
        case _1__Выключена:

            switch (e) {
            case e11:

                // 1. Выключена->2. Включена
                // e11[!o3.x1]/o1.z1,o3.z5,o4.z1,o5.z1,o2.z4,o3.z6

                fireTransitionCandidate(context, path, "1. Выключена",
                        event, "1. Выключена#2. Включена#e11#!o3.x1");

                if (!isInputActionCalculated(calculatedInputActions,
                        _o3_x1)) {

                    fireBeforeInputActionExecution(context, path,
                            "1. Выключена#2. Включена#e11#!o3.x1",
                            "o3.x1");

                    o3_x1 = o3.x1(context);

                    fireAfterInputActionExecution(context, path,
                            "1. Выключена#2. Включена#e11#!o3.x1",
                            "o3.x1", new Boolean(o3_x1));
                }

                if (!o3_x1) {

                    fireTransitionFound(context, path, "1. Выключена",
                            event,
                            "1. Выключена#2. Включена#e11#!o3.x1");

                    fireBeforeOutputActionExecution(context, path,
                            "1. Выключена#2. Включена#e11#!o3.x1",
                            "o1.z1");

                    o1.z1(context);

                    fireAfterOutputActionExecution(context, path,
                            "1. Выключена#2. Включена#e11#!o3.x1",
                            "o1.z1");
                    fireBeforeOutputActionExecution(context, path,
                            "1. Выключена#2. Включена#e11#!o3.x1",
                            "o3.z5");

                    o3.z5(context);
```

```
                fireAfterOutputActionExecution(context, path,
                        "1. Выключена#2. Включена#e11#!o3.x1",
                        "o3.z5");

                ...


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

                // 2. Включена []
                return new StateMachineConfig("2. Включена");

            }
            // 1. Выключена->2. Включена
            // e11[o3.x1]/o1.z1,o3.z1,o4.z1,o5.z1,o2.z4,o3.z6

            fireTransitionCandidate(context, path, "1. Выключена",
                    event, "1. Выключена#2. Включена#e11#o3.x1");

            if (o3_x1) {

                fireTransitionFound(context, path, "1. Выключена",
                        event, "1. Выключена#2. Включена#e11#o3.x1");

                fireBeforeOutputActionExecution(context, path,
                        "1. Выключена#2. Включена#e11#o3.x1",
                        "o1.z1");

                o1.z1(context);

                fireAfterOutputActionExecution(context, path,
                        "1. Выключена#2. Включена#e11#o3.x1",
                        "o1.z1");
                ...

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

                // 2. Включена []
                return new StateMachineConfig("2. Включена");

            }

            // transition not found
            return config;
        case e4:

            // 1. Выключена->s3 e4[true]/o2.z4,o1.z5,o3.z5

        ...

        default:

            // transition not found
            return config;
        }

    case _2__Включена:

        fireTransitionsOfSuperstate(context, path, "2. Включена",
                event);

        switch (e) {
        case e4:
```

```
                // 2. Включена->s3 e4[true]/o2.z4,o1.z5,o3.z5

                ...

            case e12:

                // 2. Включена->1. Выключена
                // e12[o3.x1]/o1.z2,o3.z2,o4.z2,o5.z2,o2.z4,o3.z6
                ...
                if (o3_x1) {
                    ...
                }
                ...
                // 2. Включена->1. Выключена
                // e12[!o3.x1]/o1.z2,o3.z5,o4.z2,o5.z2,o2.z4,o3.z6
                ...
                if (!o3_x1) {
                    ...
                }
                ...
                // transition not found
                return config;
            }

        case _1__Обычное_состояние:

            ...

        case _2__Состояние_опасности:

            ...

        case _3__Тревога:

            ...

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

Этот метод во вложенных операторах `switch` в зависимости от состояния, события и значения соответствующего условия определяет необходимый переход и выполняет связанные с ним методы объектов управления. После этого возвращается новая конфигурация автомата.

Класс `A2EventProcessor` имеет аналогичную структуру.

На этом краткое описание структуры сгенерированного файла завершается.

Для работы приложения необходимо создать небольшой исполняемый класс, который будет запускать программу (**Ошибка! Источник ссылки не найден.**, **Ошибка! Источник ссылки не найден.Ошибка! Источник ссылки не найден.**). Он использует сгенерированный *Java*-файл и **написанные вручную реализации методов**, соответствующих поставщикам событий и объектам управления.

В заключение отметим, что объем программы для интерпретационного подхода в данном случае составляет около четырех мегабайт, а для компилятивного – меньше одного мегабайта.

Выводы

Выполненный проект подтверждает тот факт, что для задач рассматриваемого класса применение автоматного подхода целесообразно. Этот подход позволяет разделить систему на отдельные несвязанные объекты, что сильно упрощает ее программирование. Наглядные диаграммы состояний позволяют отслеживать поведение автоматов и возможные ошибки в этой части.

Набор инструментов *Unimod* [2] сильно упростил задачу реализации спроектированной системы. Это удобное средство написания автоматных программ.

Список источников

1. *Alarm System* MONGOOSE. Руководство по эксплуатации и инструкция по установке.

2. *Unimod.* http://unimod.sourceforge.net

Приложение 1. Пример протокола работы программы

Протокол приводится для следующей последовательности действий. Сигнализация включается, затем машина получает слабый удар, а после этого сильный удар. Через некоторое время сигнализация выключается в тихом режиме, и окно программы закрывается.

```
Обработка события [e11 Нажатие кнопки 1] в состоянии [A1.start]
  [A1.start] ==> [A1.1 Выключена]
  [A1.1 Выключена] ==> [A1.2 Включена]
    [o1.z11 Мигнуть один раз]
    [o3.z31 Выдать короткий гудок]
    [o4.z41 Включить мигание светодиода]
      Обработка события [e11] в состоянии [A2.start]
        [A2.start] ==> [A2.1 Обычное состояние]
      Обработка события [e11] завершена в состоянии [A2.1 Обычное состояние]
Обработка события [e11] завершена в состоянии [A1.2 Включена]

Обработка события [e21 Слабый удар] в состоянии [A1.2 Включена]
      Обработка события [e21] в состоянии [A2.1 Обычное состояние]
        Проверка условия перехода [o1.x1 Подается сигнал тревоги]
          [o1.x1] = false
        [A2.1 Обычное состояние] ==> [A2.2 Состояние опасности]
          [o1.z13 Мигнуть три раза]
          [o3.z33 Выдать три коротких гудка]
          [o2.z25 Запустить таймер на пять секунд]
      Обработка события [e21] завершена в состоянии [A2.2 Состояние опасности]
Обработка события [e21] завершена в состоянии [A1.2 Включена]

Обработка события [e22 Сильный удар] в состоянии [A1.2. Включена]
      Обработка события [e22] в состоянии [A2.2. Состояние опасности]
        [A2.2 Состояние опасности] ==> [A2.1 Обычное состояние]
          [o1.z14 Запустить тревожный сигнал фар на 10 секунд]
          [o3.z34 Запустить тревожный сигнал сирены на 10 секунд]
      Обработка события [e22] завершена в состоянии [A2.1 Обычное состояние]
Обработка события [e22] завершена в состоянии [A1.2. Включена]

Обработка события [e31 Завершение таймера] в состоянии [A1.2. Включена]
      Обработка события [e31] в состоянии [A2.1 Обычное состояние]
      Обработка события [e31] завершена в состоянии [A2.1 Обычное состояние]
Обработка события [e31] завершена в состоянии [A1.2. Включена]

Обработка события [e13 Нажатие кнопки 3] в состоянии [A1.2. Включена]
  [A1.2. Включена] ==> [A1.4. Ожидание тихого выключения]
    [o2.z23 Запустить таймер на три секунды]
      Обработка события [e13] в состоянии [A2.start]
        [A2.start] ==> [A2.1 Обычное состояние]
      Обработка события [e13] завершена в состоянии [A2.1 Обычное состояние]
Обработка события [e13] завершена в состоянии [A1.4. Ожидание тихого выключения]

Обработка события [e12 Нажатие кнопки 2] в состоянии [A1.4. Ожидание тихого
выключения]
  [A1.4. Ожидание тихого выключения] ==> [A1.1. Выключена]
    [o1.z12 Мигнуть два раза]
    [o3.z35 Завершить звук]
    [o4.z42 Выключить светодиод]
Обработка события [e12] завершена в состоянии [A1.1. Выключена]

Обработка события [e31 Завершение таймера] в состоянии [A1.1. Выключена]
Обработка события [e31] завершена в состоянии [A1.1. Выключена]

Обработка события [e32 Закрытие окна программы] в состоянии [A1.1. Выключена]
```

```
  [A1.1. Выключена] ==> [A1.final]
    [o3.z35 Завершить звук]
  Автомат достиг конечного состояния
```
Обработка события [e32] завершена в состоянии [A1.final]

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="ru.ifmo.alarm.controlled.Lights" name="o1"/>
  <controlledObject class="ru.ifmo.alarm.event.StateTimer" name="o2"/>
  <controlledObject class="ru.ifmo.alarm.controlled.Sounds" name="o3"/>
  <controlledObject class="ru.ifmo.alarm.controlled.LightDiode" name="o4"/>
  <controlledObject class="ru.ifmo.alarm.controlled.Commentator" name="o5"/>
  <eventProvider class="ru.ifmo.alarm.event.RemoteControl" name="p1">
    <association clientRole="p1" targetRef="A1"/>
  </eventProvider>
  <eventProvider class="ru.ifmo.alarm.event.HitSensor" name="p2">
    <association clientRole="p2" targetRef="A1"/>
  </eventProvider>
  <eventProvider class="ru.ifmo.alarm.event.StateTimer" name="p3">
    <association clientRole="p3" targetRef="A1"/>
  </eventProvider>
  <eventProvider class="ru.ifmo.alarm.event.Window" name="p4">
    <association targetRef="A1"/>
  </eventProvider>
  <rootStateMachine>
    <stateMachineRef name="A1"/>
  </rootStateMachine>
  <stateMachine name="A1">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A1" supplierRole="o2" targetRef="o2"/>
    <association clientRole="A1" supplierRole="A2" targetRef="A2"/>
    <association clientRole="A1" supplierRole="o4" targetRef="o4"/>
    <association clientRole="A1" supplierRole="o1" targetRef="o1"/>
    <association clientRole="A1" supplierRole="o3" targetRef="o3"/>
    <association clientRole="A1" supplierRole="o5" targetRef="o5"/>
    <state name="Top" type="NORMAL">
      <state name="s3" type="FINAL"/>
      <state name="1. Выключена" type="NORMAL">
        <stateMachineRef name="A2"/>
        <outputAction ident="o2.z4"/>
        <outputAction ident="o3.z6"/>
        <outputAction ident="o4.z2"/>
      </state>
      <state name="2. Включена" type="NORMAL">
        <state name="1. Обычное состояние" type="NORMAL"/>
        <state name="2. Состояние опасности" type="NORMAL">
          <outputAction ident="o1.z3"/>
          <outputAction ident="o2.z2"/>
          <outputAction ident="o3.z3"/>
          <outputAction ident="o5.z7"/>
        </state>
        <state name="3. Тревога" type="NORMAL">
          <outputAction ident="o1.z4"/>
          <outputAction ident="o2.z1"/>
          <outputAction ident="o3.z4"/>
          <outputAction ident="o5.z9"/>
        </state>
```

```
        <state name="s2" type="INITIAL"/>
        <stateMachineRef name="A2"/>
        <outputAction ident="o1.z1"/>
        <outputAction ident="o4.z1"/>
        <outputAction ident="o5.z1"/>
        <outputAction ident="o2.z4"/>
        <outputAction ident="o3.z6"/>
      </state>
      <state name="s1" type="INITIAL"/>
    </state>
    <transition event="e11" guard="!o3.x1" sourceRef="1. Выключена" targetRef="2.
Включена">
      <outputAction ident="o3.z5"/>
    </transition>
    <transition event="e11" guard="o3.x1" sourceRef="1. Выключена" targetRef="2.
Включена">
      <outputAction ident="o3.z1"/>
    </transition>
    <transition event="e4" sourceRef="1. Выключена" targetRef="s3">
      <outputAction ident="o2.z4"/>
      <outputAction ident="o1.z5"/>
      <outputAction ident="o3.z5"/>
    </transition>
    <transition event="e12" guard="o3.x1" sourceRef="2. Включена" targetRef="1.
Выключена">
      <outputAction ident="o1.z2"/>
      <outputAction ident="o3.z2"/>
      <outputAction ident="o5.z2"/>
    </transition>
    <transition event="e12" guard="!o3.x1" sourceRef="2. Включена" targetRef="1.
Выключена">
      <outputAction ident="o1.z2"/>
      <outputAction ident="o3.z5"/>
      <outputAction ident="o5.z2"/>
    </transition>
    <transition event="e4" sourceRef="2. Включена" targetRef="s3">
      <outputAction ident="o2.z4"/>
      <outputAction ident="o1.z5"/>
      <outputAction ident="o3.z5"/>
    </transition>
    <transition event="e21" sourceRef="1. Обычное состояние" targetRef="2.
Состояние опасности">
      <outputAction ident="o5.z5"/>
    </transition>
    <transition event="e22" sourceRef="1. Обычное состояние" targetRef="3.
Тревога">
      <outputAction ident="o5.z6"/>
    </transition>
    <transition event="e32" sourceRef="2. Состояние опасности" targetRef="1.
Обычное состояние">
      <outputAction ident="o5.z8"/>
    </transition>
    <transition event="e22" sourceRef="2. Состояние опасности" targetRef="3.
Тревога">
      <outputAction ident="o5.z6"/>
    </transition>
    <transition event="e21" sourceRef="2. Состояние опасности" targetRef="3.
Тревога">
      <outputAction ident="o5.z5"/>
    </transition>
    <transition event="e33" sourceRef="3. Тревога" targetRef="1. Обычное
состояние">
      <outputAction ident="o1.z5"/>
      <outputAction ident="o3.z5"/>
```

```
      <outputAction ident="o5.z10"/>
    </transition>
    <transition sourceRef="s2" targetRef="1. Обычное состояние"/>
    <transition sourceRef="s1" targetRef="1. Выключена"/>
  </stateMachine>
  <stateMachine name="A2">
    <configStore
class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
    <association clientRole="A2" supplierRole="o2" targetRef="o2"/>
    <association clientRole="A2" supplierRole="o3" targetRef="o3"/>
    <association clientRole="A2" supplierRole="o5" targetRef="o5"/>
    <state name="Top" type="NORMAL">
      <state name="s1" type="INITIAL"/>
      <state name="2. Звук выключен" type="NORMAL"/>
      <state name="1. Звук включен" type="NORMAL"/>
    </state>
    <transition sourceRef="s1" targetRef="1. Звук включен"/>
    <transition event="e11" sourceRef="2. Звук выключен" targetRef="1. Звук
включен">
      <outputAction ident="o3.z6"/>
    </transition>
    <transition event="e31" sourceRef="2. Звук выключен" targetRef="1. Звук
включен">
      <outputAction ident="o5.z4"/>
      <outputAction ident="o3.z6"/>
    </transition>
    <transition event="e12" sourceRef="2. Звук выключен" targetRef="1. Звук
включен">
      <outputAction ident="o3.z6"/>
    </transition>
    <transition event="e13" sourceRef="1. Звук включен" targetRef="2. Звук
выключен">
      <outputAction ident="o2.z3"/>
      <outputAction ident="o5.z3"/>
      <outputAction ident="o3.z7"/>
    </transition>
  </stateMachine>
   </model>
```