Санкт-Петербургский государственный университет
информационных технологий, механики и оптики
Факультет информационных технологий и программирования
Кафедра компьютерных технологий

Ф.В. Подтелкин, А.А. Шалыто

Моделирование многоэтажной автоматической стоянки на основе автоматного программирования

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

Санкт-Петербург
2007

Оглавление

ВВЕДЕНИЕ **3**

1. ОПИСАНИЕ ПРОЕКТА **3**

1.1. Автоматические многоэтажные стоянки 3
1.2. Постановка задачи 3

2. ПРОЕКТИРОВАНИЕ **6**

2.1. Схема связей 6
2.2. Поставщики событий 7
2.2.1. Класс `ParkingEventProvider` *7*
2.2.2. Класс `SystemEventProvider` *7*
2.3. Объекты управления 7
2.3.1. Объект управления приложением *7*
2.3.2. Объект управления парковкой *8*
2.3.3. Объект управления подъемником *8*
2.4. Автомат 9
2.3. Описание состояний автомата 9

3. РЕАЛИЗАЦИЯ **11**

3.1. Структура программы 11
3.2. Интерпретационный подход 11
3.3. Компилятивный подход 12
3.4. Запуск программы 12
3.5. Пользовательский интерфейс 13

ЗАКЛЮЧЕНИЕ **15**

ЛИТЕРАТУРА **15**

ПРИЛОЖЕНИЕ 1. ПРИМЕР ПРОТОКОЛА РАБОТЫ ПРОГРАММЫ **16**

ПРИЛОЖЕНИЕ 2. СГЕНЕРИРОВАННОЕ XML-ОПИСАНИЕ **21**

ПРИЛОЖЕНИЕ 3. СГЕНЕРИРОВАННЫЙ ПО XML-ОПИСАНИЮ КОД НА JAVA **23**

Введение

Данный проект демонстрирует возможности автоматного программирования в сфере моделирования таких сложных устройств управления, как многоэтажная автоматическая стоянка. В качестве основного инструментального средства был использован *UniMod*, который представляет собой надстройку над открытой средой разработки *Eclipse.* При этом реализуются такие концепции как «Исполняемый *UML*» и «Разработка на базе моделей».

1. Описание проекта

1.1. Автоматические многоэтажные стоянки

В связи с резким увеличением количества автомобилей в мире в конце XX века, в крупных городах резко стала проблема нехватки места. Как правило, крупные компании занимают многоэтажные бизнес-центры в городе, в которых работают десятки тысяч человек. Обеспечить всех сотрудников парковочными местами для их автомобилей, используя классические стоянки, не представляется возможным, в первую очередь из того, что такие стоянки в центре мегаполиса экономически не выгодны из-за большой цены на землю. Тогда были предприняты первые попытки исправить это положение – стали строить подземные и многоэтажные винтовые парковки. Однако в некоторых городах они перестали решать проблемы с парковочными местами. Время парковки на них занимает много времени, ведь для того, чтобы подняться на десятый этаж необходимо потратить лишних 10 минут. Кроме того, из-за большого количества поворотов нередко возникали аварии, и весь поток машин останавливался.

Особенно актуально эта проблема стала в Японии (площадь Японии почти в 50 раз меньше площади России, а жителей там чуть меньше, чем в России). Японские компании предложили современное решение данной проблемы – строительство автоматических многоэтажных стоянок. Плюсы таких стоянок сразу же оценили. Теперь парковка автомобиля занимала не больше минуты, а вместимость таких стоянок увеличилась в два – три раза.

1.2. Постановка задачи

Задачей, решаемой в настоящей работе, является проектирование программы, которая управляет работой автоматической стоянки. Заметим, что существует несколько схем таких стоянок, например, такая в которой автомобили расположены по кругу, а в центре находится автоматический подъемник, распределяющий автомобили (рис.1).

Рис.1. Пример автоматической стоянки

В настоящей работе проектируется программа для одного подъемника стоянки с несколькими подъемниками. Преимуществом такой стоянки является то, что несколько автомобилей могут парковаться параллельно (это в работе не моделируется)

Схема такой стоянки представляется следующим образом: вся стоянка разделена на некоторое количество секций. Каждый подъемник будет отвечать за одну секцию. Въезды и выезды на стоянку будут располагаться напротив друг друга (рис.2).

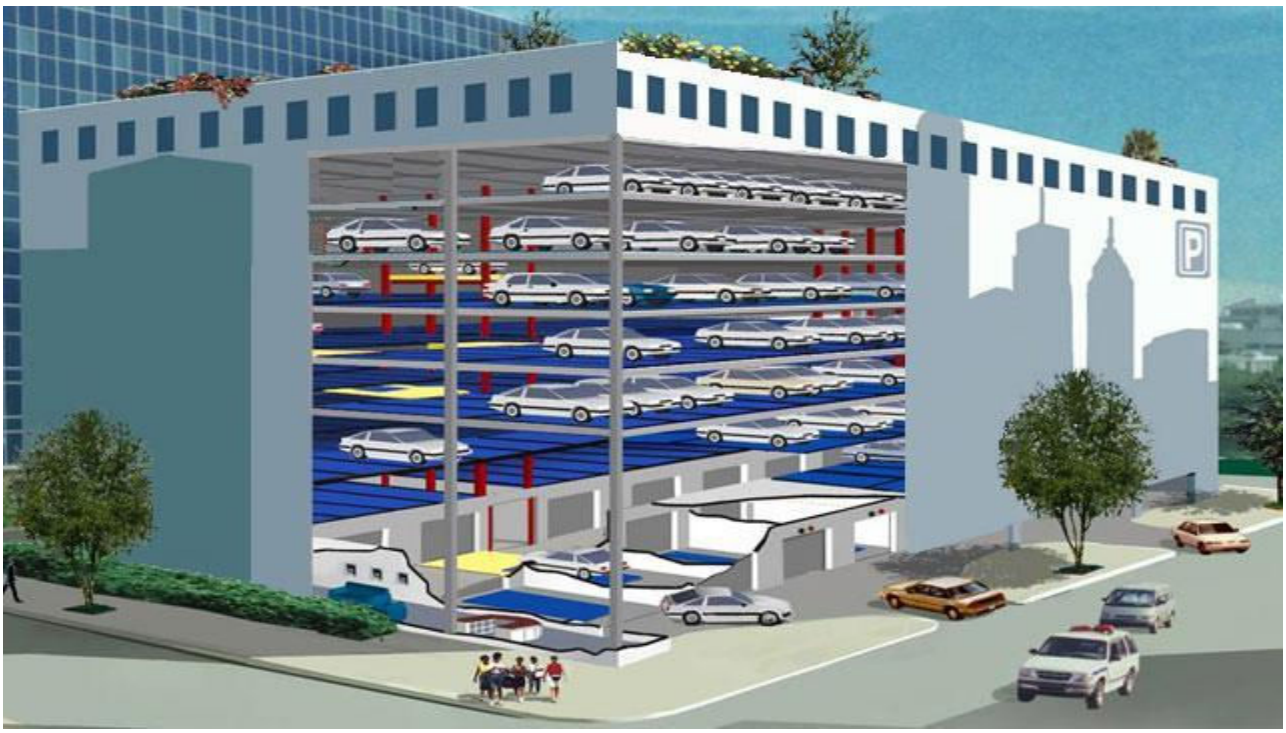

Рис.2. Автоматическая многоэтажная стоянка

Клиенты, которые хотят припарковаться, должны нажать кнопку и при въезде на стоянку. Через некоторое время, когда подъемник освободится, должен открыться въезд на стоянку с предложением поставить свой автомобиль на помост подъемника.

Как только клиент поставит свой автомобиль на подъемник, необходимо подождать, пока он не выйдет из автомобиля и не закроет его. После этого клиент должен будет нажать кнопку "Выйти из машины".

После этого подъемник должен припарковать автомобиль на любое свободное место. Также возможна ситуация, при которой на парковке нет свободных мест. При этом на въезде должна гореть большая красная надпись, информирующая об этом, и все запросы клиентов на парковку должны игнорироваться.

2. Проектирование

Проектирование приложения было выполнено при помощи инструментального средства *UniMod*. Применение этого средства позволило визуально построить схему связей в виде диаграммы классов и автоматы, формирующие логику поведения программы.

2.1. Схема связей

Схема связей в виде диаграммы классов данного приложения приведена на рис. 2. На ней представлены поставщик событий, главный автомат и три объекта управления.

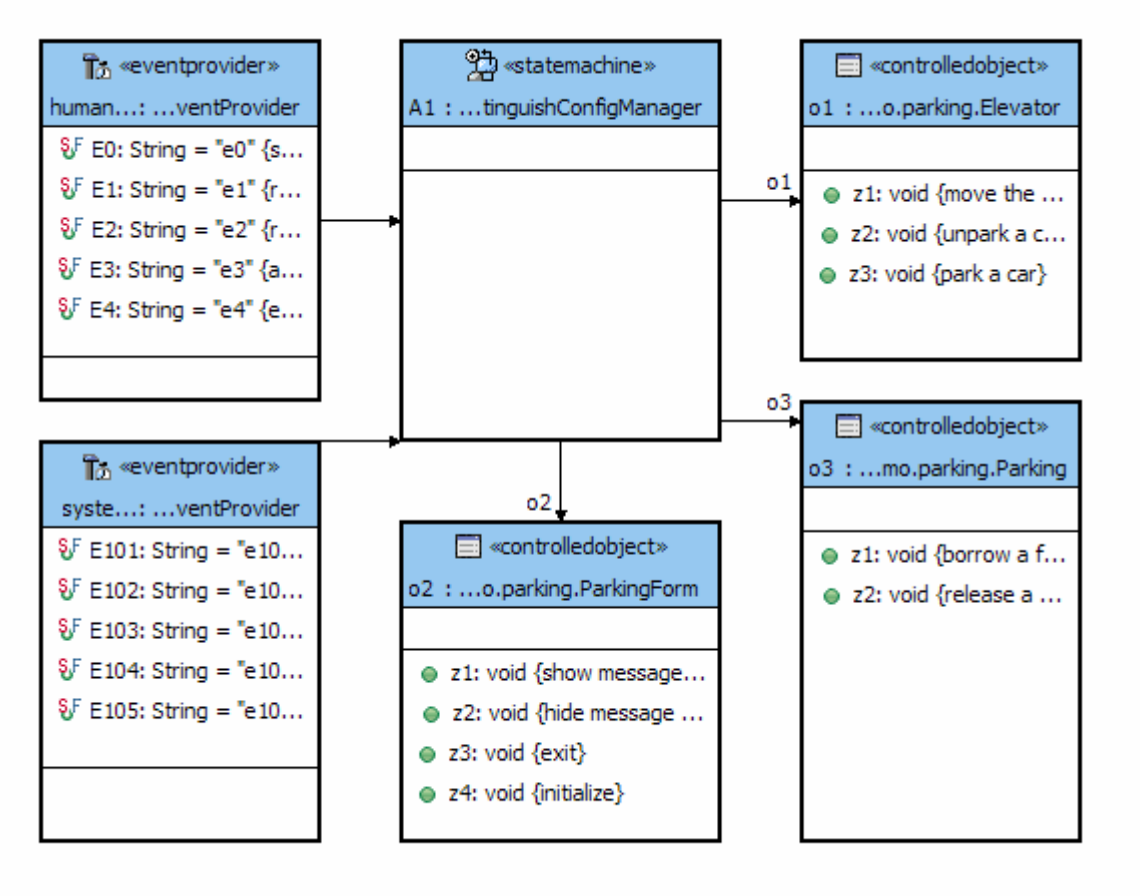

Рис. 3. Схема связей

2.2. Поставщики событий

2.2.1. Класс `ParkingEventProvider`

В системе автоматической парковки могут возникнуть следующие события, вызванные действиями пользователя (табл.1).

Таблица 1

Событие	**Описание**
E0	Запуск приложения, инициализация
E1	Поступил запрос на парковку автомобиля
E2	Поступил запрос на отпарковку автомобиля
E3	Автомобиль готов для парковки, клиент вышел из автомобиля
E4	Выход из приложения

2.2.2. Класс `SystemEventProvider`

В системе автоматической парковки могут возникнуть следующие события, вызванные системой (табл. 2).

Таблица 2

Событие	**Описание**
E101	Подъемник готов для парковки следующего автомобиля
E102	На парковке нет мест
E103	Парковочное место занято
E104	Автомобиль отпаркован
E105	Парковочное место освобождено

2.3. Объекты управления

2.3.1. Объект управления приложением

Этот объект управления предоставляет выходные воздействия (табл. 3), которые относятся к управлению визуализацией.

Таблица 3

Action	**Description**
z1	Показать сообщение, что парковка заполнена
z2	Убрать сообщение, что парковка заполнена
z3	Закрыть текущее приложение
z4	Инициализация

2.3.2. Объект управления парковкой

Этот объект управления предоставляет выходные воздействия (табл. 4), которые относятся к управлению парковкой.

Таблица 4

Action	**Description**
z1	Забронировать свободное место на стоянке. Новое место записывается в `application context`, чтобы подъемник знал, куда ему парковать следующий автомобиль.
z2	Освободить место на стоянке. Место, которое требуется освободить, передается также через `application context`.

2.3.3. Объект управления подъемником

Этот объект управления предоставляет выходные воздействия (табл. 5), которые относятся к управлению подъемником.

Таблица 5

Action	**Description**
z1	Передвинуть подъемник к входу
z2	Припарковать автомобиль на зарезервированное ранее место, которое он получает от парковки через `application context`
z3	Отпарковать автомобиль с указанным номером, который передается через `application context`.

2.4. Автомат

На рис. 4 приведен граф переходов автомата, который управляет моделированием стоянки.

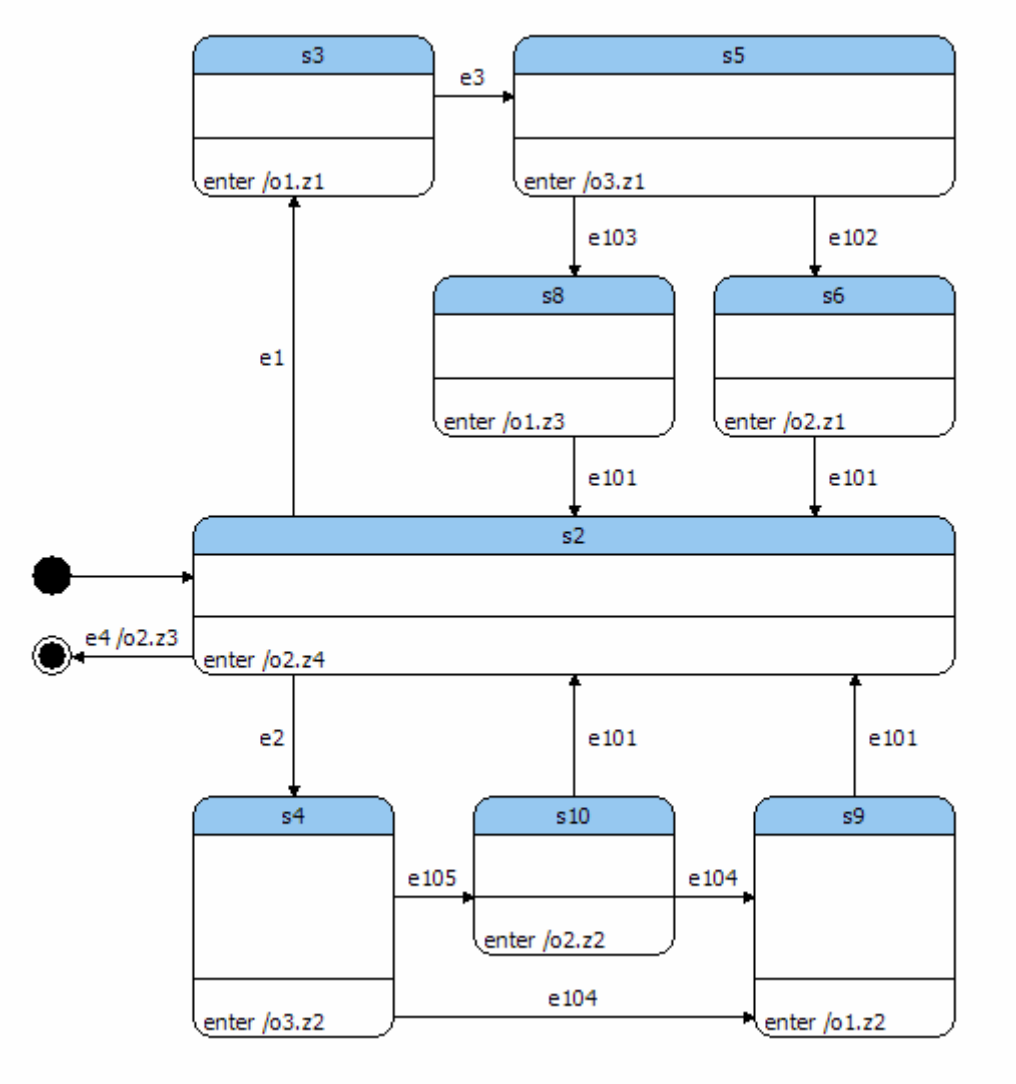

Рис. 4. Автомат

2.3. Описание состояний автомата

В табл. 6 приведено описание состояний автомата.

Таблица 6

State	**Description**
s1	Начальное состояние автомата
s2	Парковка находится в свободном состоянии, ожидая дальнейших заявок на парковку автомобилей
s3	Состояние, в которое переходит автомат, когда получает новый запрос на парковку автомобиля. В этом состоянии подъемник передвигается к въезду.
s4	Состояние, в которое переходит автомат, когда получает новый запрос на получение автомобиля
s5	Происходит определение места, куда будет припаркован автомобиль
s6	Показывается надпись, что на парковке больше нет свободного места
s8	Происходит парковка автомобиля
s9	Происходит отпарковка автомобиля
s10	Отключается надпись, говорящая о том, что парковка запрещена, т.к. появилось свободное место.
s11	Конечное состояние автомата

3. Реализация

3.1. Структура программы

Программа состоит из двух частей: из *XML*-описания, которое автоматически генерируется по схеме связей и диаграммам состояний, и кода, написанного вручную. Этот код написан в объектно-ориентированном стиле на языке *Java* в среде разработки *Eclipse* с использованием инструментального средства *UniMod*.

Существует два подхода к созданию к созданию исполняемого приложения: *интерпретационный* и *компилятивный*.

Вручную написанный код состоит из следующих классов:

`ru.ifmo.parking.Config` – представляет собой контейнер для хранения текущей конфигурации стоянки;

`ru.ifmo.parking.Elevator` – предоставляет методы входных и выходных воздействий. Эти действия относятся к управлению движением подъемника;

`ru.ifmo.parking.Logging` – предоставляет методы для логирования работы приложения;

`ru.ifmo.parking.Parking` – предоставляет методы входных и выходных воздействий. Эти действия относятся к управлению парковочных мест на стоянке;

`ru.ifmo.parking.ParkingEventProvider` – определяет события, вызванные действиями клиентов;

`ru.ifmo.parking.ParkingForm` – представляет собой визуализатор работы стоянки;

`ru.ifmo.parking.Place` – представляет собой внутренний объект парковочного места;

`ru.ifmo.parking.SystemEventProvider` – определяет события, вызванные оборудованием стоянки.

3.2. Интерпретационный подход

Данный подход требует использования большого числа библиотек *Unimod* для работы приложения.

Построенный автомат сохраняется в формате *XML*, затем компилируются классы, реализующие работу приложения. Теперь можно запускать приложение, использую набор библиотек *Unimod*, *XML*-файл и откомпилированные классы. При работе в командную консоль будут выводиться комментарии *Unimod* о происходящих событиях и совершаемых переходах в автомате. Данный подход невыгоден тем, что требует подключения большого количества библиотек, которые для работы данного конкретного приложения не являются необходимыми.

3.3. Компилятивный подход

При использовании данного подхода с помощью средств *Unimod* из *XML*-файла, описывающего автомат, генерируется *Java*-файл. В нем явно описываются все объекты системы, события, автоматы, их состояния и переходы. Для его работы также необходимы некоторые классы библиотек *Unimod*, однако гораздо меньше их количество, чем при использовании интерпретационного подхода.

Сгенерированный файл находится в пакете *ru.ifmo.parking* и называется `Model1EventProcessor.java`.

3.4. Запуск программы

Как запустить программу? Необходимо скачать и распаковать архив с исполняемой программой. После этого запустить файл *run.bat*. Как открыть проект в *Eclipse*? Необходимо в среде разработки *Eclipse* выбрать пункт меню "*File/Import:*" ("*Файл/Импорт:*"), а в открывшемся диалоге мастера - пункт "*Existing Project into Workspace*" ("*Существующий проект в рабочую область*") и нажать кнопку "*Next >*" ("*Далее >*"). В окне выбора папки с файлами проекта необходимо указать ту папку, в которую распакован архив с проектом.

На рис.5 изображен примера работы программы для случая, когда на стоянку паркуется новый автомобиль.

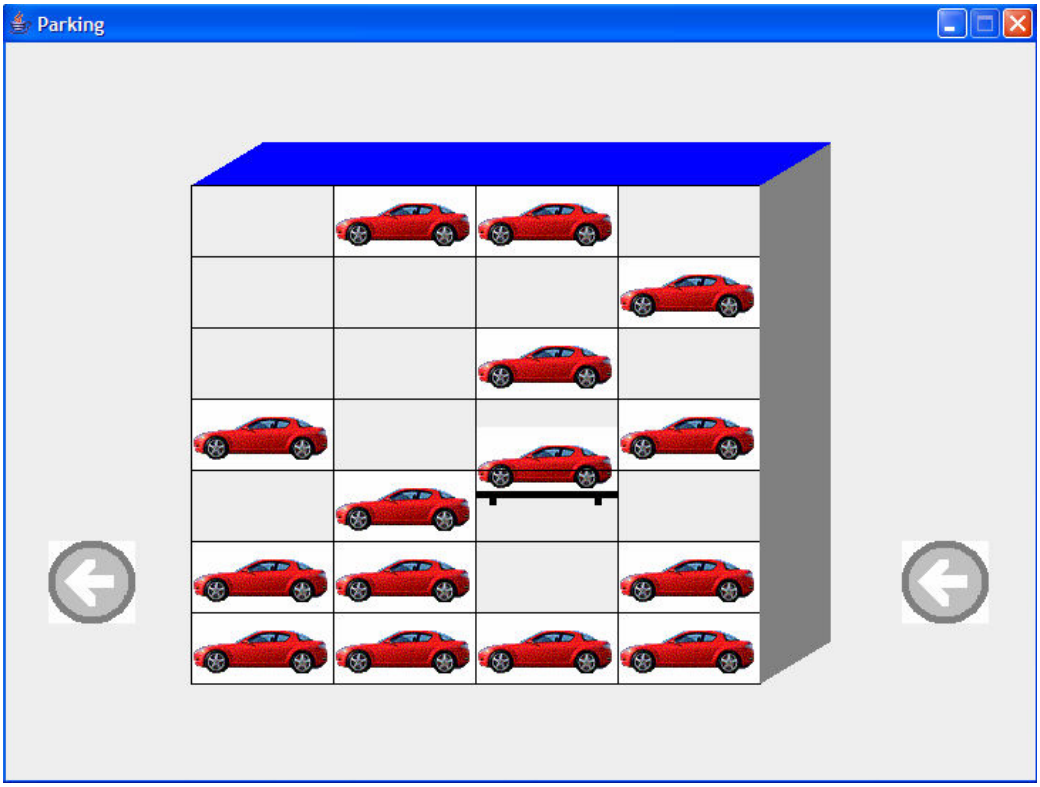

Рис. 5. Парковка автомобиля

На рис. 6 изображен пример работы программы для случая, когда со стоянки забирают автомобиль.

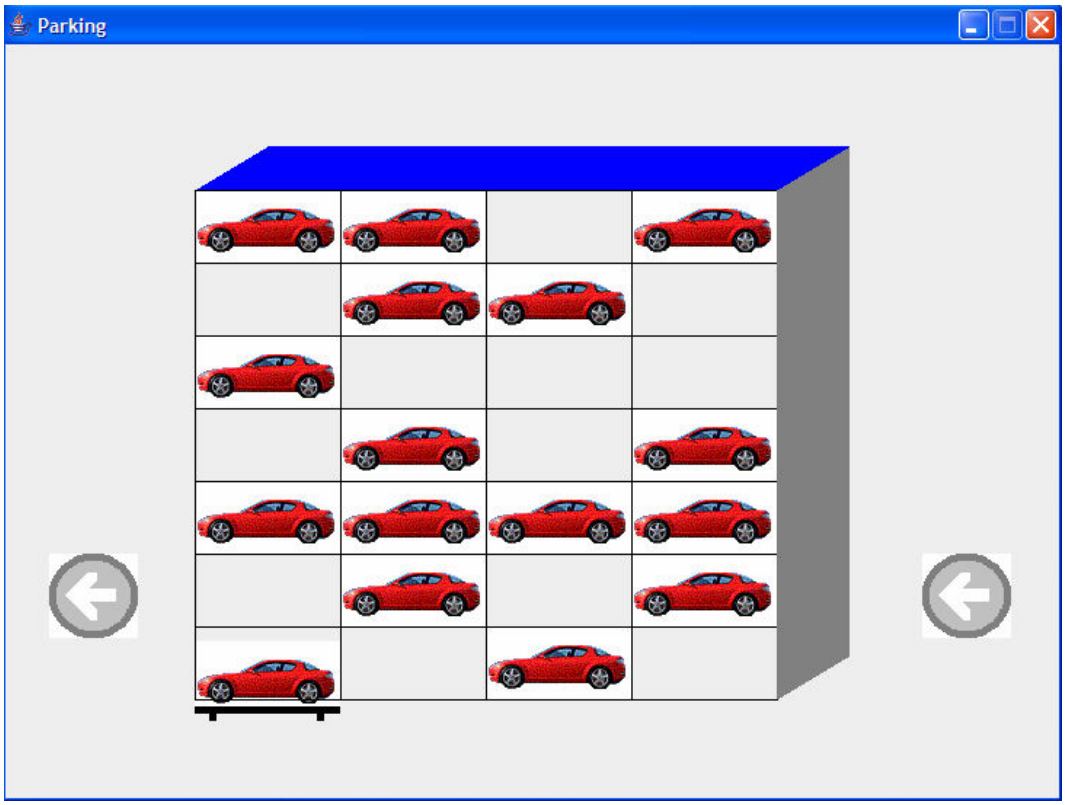

Рис.6. Отпарковка автомобиля

3.5. Пользовательский интерфейс

Программа, реализующая данный проект, является графическим приложением, которое состоит из одного окна. В этом окне схематически изображена одна секция многоэтажной парковки с двумя кнопками (key), управляющими парковкой и отпарковкой автомобиля.

Кнопка запроса на парковку автомобиля может находиться в одном из четырех состояний, описанных в табл. 7.

Таблица 7

	Key	Description
1		Парковка готова принять следующий автомобиль на парковку
2		Парковка невозможна – занята операцией парковки или отпарковки автомобиля
3		Подъемник готов к парковке текущего автомобиля. Для парковки требуется нажать эту кнопку. Она должна быть нажата после того, как автомобиль будет установлен на подъемник и водитель покинет свой автомобиль
4		Парковка полностью заполнена и не может больше принимать запросы на парковку, пока не освободиться место

Для того чтобы припарковать свой автомобиль, требуется дождаться, пока кнопка на въезде на парковку будет в первом состоянии, нажать на нее. После этого необходимо дождаться, пока подъемник будет перемещен к въезду. Кнопка должна перейти в третье состояние. После этого, требуется поставить автомобиль на подъемник, покинуть автомобиль и нажать кнопку.

Кнопка запроса на отпарковку автомобиля может находиться в одном из первых двух состояний, отображенных в табл. 7.

Для того чтобы забрать свой автомобиль со стоянки, необходимо дождаться пока кнопка на выезде с парковки будет в первом состоянии, выбрать свой автомобиль, кликнув на нем левой кнопкой мыши, и нажать на кнопку.

Заключение

Выполненный проект показал удобство и целесообразность использования применения автоматного подхода при проектировании и программировании устройств управления, рассмотренного класса. Такой подход позволяет расширить функциональность существующих программ за счет добавления новых состояний и переходов по событиям. Наглядность и доступность этих изменений заметно повышаются при использовании средств визуального построения схемы связи и автоматов.

Литература

1. *Шалыто А.А.* Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. http://is.ifmo.ru/books/switch/1
2. *Шалыто А.А.* Новая инициатива в программировании. Движение за открытую проектную документацию // Мир ПК. 2003. №9, с.52–56. http://is.ifmo.ru/works/open_do
3. *Гуров В.С., Мазин М.А., Нарвский А.С., Шалыто А.А.* UML. SWITCH-технология. Eclipse. // Информационно-управляющие системы. 2004. №6, с.12–17. http://is.ifmo.ru/works/uml-switch-eclipse

Приложение 1. Пример протокола работы программы

```
01:05:59,062 INFO  [Run] Start event [e0] processing. In state [/A1:Top]
01:05:59,062 INFO  [Run] Transition to go found [s1#s2##true]
01:05:59,062 INFO  [Run] Start on-enter action [o2.z4] execution
01:05:59,062 INFO  [Run] Finish on-enter action [o2.z4] execution
01:05:59,062 INFO  [Run] Finish event [e0] processing. In state [/A1:s2]
01:05:59,906 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:05:59,906 DEBUG [Run] Try transition [s2#s3#e1#true]
01:05:59,906 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:05:59,906 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:00,906 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:00,906 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:01,437 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:01,437 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:01,437 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:01,437 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:01,453 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:01,453 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:01,453 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:01,453 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:01,453 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:01,453 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:02,953 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:02,968 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:02,968 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:02,968 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:02,968 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:02,968 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:02,968 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:02,968 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:04,796 INFO  [Run] Start event [e2] processing. In state [/A1:s2]
01:06:04,796 DEBUG [Run] Try transition [s2#s4#e2#true]
01:06:04,796 INFO  [Run] Transition to go found [s2#s4#e2#true]
01:06:04,796 INFO  [Run] Start on-enter action [o3.z2] execution
01:06:04,796 INFO  [Run] Finish on-enter action [o3.z2] execution
01:06:04,796 INFO  [Run] Finish event [e2] processing. In state [/A1:s4]
01:06:04,796 INFO  [Run] Start event [e104] processing. In state [/A1:s4]
01:06:04,796 DEBUG [Run] Try transition [s4#s9#e104#true]
01:06:04,796 INFO  [Run] Transition to go found [s4#s9#e104#true]
01:06:04,796 INFO  [Run] Start on-enter action [o1.z2] execution
01:06:09,812 INFO  [Run] Finish on-enter action [o1.z2] execution
01:06:09,812 INFO  [Run] Finish event [e104] processing. In state [/A1:s9]
01:06:09,812 INFO  [Run] Start event [e101] processing. In state [/A1:s9]
01:06:09,812 DEBUG [Run] Try transition [s9#s2#e101#true]
01:06:09,812 INFO  [Run] Transition to go found [s9#s2#e101#true]
01:06:09,828 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:09,828 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:09,828 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:11,390 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:11,390 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:11,390 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:11,390 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:13,906 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:13,906 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:14,218 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:14,218 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:14,218 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:14,218 INFO  [Run] Start on-enter action [o3.z1] execution
```

```
01:06:14,218 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:14,218 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:14,218 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:14,218 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:14,218 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:14,218 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:16,234 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:16,234 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:16,234 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:16,234 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:16,234 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:16,234 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:16,234 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:16,234 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:18,218 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:18,218 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:18,218 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:18,218 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:20,218 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:20,218 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:20,765 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:20,765 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:20,765 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:20,765 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:20,765 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:20,765 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:20,765 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:20,765 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:20,765 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:20,765 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:23,281 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:23,281 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:23,281 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:23,281 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:23,281 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:23,281 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:23,281 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:23,281 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:23,640 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:23,640 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:23,640 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:23,640 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:26,156 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:26,156 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:26,531 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:26,531 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:26,531 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:26,531 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:26,531 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:26,531 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:26,531 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:26,531 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:26,531 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:26,531 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:29,046 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:29,046 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:29,046 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:29,046 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:29,046 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:29,046 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:29,046 INFO  [Run] Finish on-enter action [o2.z4] execution
```

```
01:06:29,046 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:29,250 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:29,250 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:29,250 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:29,250 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:31,765 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:31,765 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:31,812 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:31,812 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:31,812 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:31,812 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:31,812 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:31,812 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:31,812 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:31,812 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:31,812 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:31,812 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:34,328 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:34,328 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:34,328 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:34,328 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:34,328 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:34,328 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:34,328 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:34,328 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:34,593 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:34,593 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:34,593 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:34,593 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:37,109 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:37,109 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:37,406 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:37,406 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:37,406 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:37,406 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:37,406 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:37,406 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:37,406 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:37,406 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:37,406 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:37,406 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:40,437 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:40,437 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:40,437 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:40,437 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:40,437 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:40,437 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:40,437 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:40,437 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:40,687 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:40,687 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:40,687 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:40,687 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:43,718 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:43,718 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:43,921 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:43,921 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:43,921 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:43,921 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:43,921 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:43,921 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
```

```
01:06:43,921 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:43,921 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:43,921 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:43,921 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:46,937 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:46,937 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:46,937 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:46,937 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:46,937 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:46,937 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:46,937 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:46,937 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:47,250 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:47,250 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:47,250 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:47,250 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:50,265 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:50,265 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:50,796 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:50,796 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:50,796 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:50,796 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:50,796 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:50,796 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:50,796 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:50,796 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:50,796 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:50,796 INFO  [Run] Start on-enter action [o1.z3] execution
01:06:54,312 INFO  [Run] Finish on-enter action [o1.z3] execution
01:06:54,312 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:06:54,312 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:06:54,312 DEBUG [Run] Try transition [s8#s2#e101#true]
01:06:54,312 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:06:54,312 INFO  [Run] Start on-enter action [o2.z4] execution
01:06:54,312 INFO  [Run] Finish on-enter action [o2.z4] execution
01:06:54,312 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:06:54,625 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:06:54,625 DEBUG [Run] Try transition [s2#s3#e1#true]
01:06:54,625 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:06:54,625 INFO  [Run] Start on-enter action [o1.z1] execution
01:06:58,140 INFO  [Run] Finish on-enter action [o1.z1] execution
01:06:58,140 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:06:58,468 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:06:58,468 DEBUG [Run] Try transition [s3#s5#e3#true]
01:06:58,468 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:06:58,468 INFO  [Run] Start on-enter action [o3.z1] execution
01:06:58,468 INFO  [Run] Finish on-enter action [o3.z1] execution
01:06:58,468 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:06:58,468 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:06:58,468 DEBUG [Run] Try transition [s5#s8#e103#true]
01:06:58,468 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:06:58,468 INFO  [Run] Start on-enter action [o1.z3] execution
01:07:02,500 INFO  [Run] Finish on-enter action [o1.z3] execution
01:07:02,500 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:07:02,500 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:07:02,500 DEBUG [Run] Try transition [s8#s2#e101#true]
01:07:02,500 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:07:02,500 INFO  [Run] Start on-enter action [o2.z4] execution
01:07:02,500 INFO  [Run] Finish on-enter action [o2.z4] execution
01:07:02,500 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:07:02,796 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
```

```
01:07:02,796 DEBUG [Run] Try transition [s2#s3#e1#true]
01:07:02,796 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:07:02,796 INFO  [Run] Start on-enter action [o1.z1] execution
01:07:06,828 INFO  [Run] Finish on-enter action [o1.z1] execution
01:07:06,828 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:07:07,078 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:07:07,078 DEBUG [Run] Try transition [s3#s5#e3#true]
01:07:07,078 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:07:07,078 INFO  [Run] Start on-enter action [o3.z1] execution
01:07:07,078 INFO  [Run] Finish on-enter action [o3.z1] execution
01:07:07,078 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:07:07,078 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:07:07,078 DEBUG [Run] Try transition [s5#s8#e103#true]
01:07:07,078 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:07:07,078 INFO  [Run] Start on-enter action [o1.z3] execution
01:07:11,609 INFO  [Run] Finish on-enter action [o1.z3] execution
01:07:11,609 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:07:11,609 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:07:11,609 DEBUG [Run] Try transition [s8#s2#e101#true]
01:07:11,609 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:07:11,609 INFO  [Run] Start on-enter action [o2.z4] execution
01:07:11,609 INFO  [Run] Finish on-enter action [o2.z4] execution
01:07:11,609 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
01:07:11,921 INFO  [Run] Start event [e1] processing. In state [/A1:s2]
01:07:11,921 DEBUG [Run] Try transition [s2#s3#e1#true]
01:07:11,921 INFO  [Run] Transition to go found [s2#s3#e1#true]
01:07:11,921 INFO  [Run] Start on-enter action [o1.z1] execution
01:07:16,437 INFO  [Run] Finish on-enter action [o1.z1] execution
01:07:16,437 INFO  [Run] Finish event [e1] processing. In state [/A1:s3]
01:07:16,703 INFO  [Run] Start event [e3] processing. In state [/A1:s3]
01:07:16,703 DEBUG [Run] Try transition [s3#s5#e3#true]
01:07:16,703 INFO  [Run] Transition to go found [s3#s5#e3#true]
01:07:16,703 INFO  [Run] Start on-enter action [o3.z1] execution
01:07:16,703 INFO  [Run] Finish on-enter action [o3.z1] execution
01:07:16,703 INFO  [Run] Finish event [e3] processing. In state [/A1:s5]
01:07:16,703 INFO  [Run] Start event [e103] processing. In state [/A1:s5]
01:07:16,703 DEBUG [Run] Try transition [s5#s8#e103#true]
01:07:16,703 INFO  [Run] Transition to go found [s5#s8#e103#true]
01:07:16,703 INFO  [Run] Start on-enter action [o1.z3] execution
01:07:21,218 INFO  [Run] Finish on-enter action [o1.z3] execution
01:07:21,218 INFO  [Run] Finish event [e103] processing. In state [/A1:s8]
01:07:21,218 INFO  [Run] Start event [e101] processing. In state [/A1:s8]
01:07:21,218 DEBUG [Run] Try transition [s8#s2#e101#true]
01:07:21,218 INFO  [Run] Transition to go found [s8#s2#e101#true]
01:07:21,218 INFO  [Run] Start on-enter action [o2.z4] execution
01:07:21,218 INFO  [Run] Finish on-enter action [o2.z4] execution
01:07:21,218 INFO  [Run] Finish event [e101] processing. In state [/A1:s2]
```

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.parking.Elevator" name="o1"/>
  <controlledObject class="ru.ifmo.parking.Parking" name="o3"/>
  <controlledObject class="ru.ifmo.parking.ParkingForm" name="o2"/>
  <eventProvider class="ru.ifmo.parking.ParkingEventProvider"
name="humanEventProvider">
    <association clientRole="humanEventProvider" targetRef="A1"/>
  </eventProvider>
  <eventProvider class="ru.ifmo.parking.SystemEventProvider"
name="systemEventProvider">
    <association clientRole="systemEventProvider" 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="o3" targetRef="o3"/>
    <association clientRole="A1" supplierRole="o2" targetRef="o2"/>
    <state name="Top" type="NORMAL">
      <state name="s3" type="NORMAL">
        <outputAction ident="o1.z1"/>
      </state>
      <state name="s5" type="NORMAL">
        <outputAction ident="o3.z1"/>
      </state>
      <state name="s8" type="NORMAL">
        <outputAction ident="o1.z3"/>
      </state>
      <state name="s6" type="NORMAL">
        <outputAction ident="o2.z1"/>
      </state>
      <state name="s2" type="NORMAL">
        <outputAction ident="o2.z4"/>
      </state>
      <state name="s1" type="INITIAL"/>
      <state name="s7" type="FINAL"/>
      <state name="s4" type="NORMAL">
        <outputAction ident="o3.z2"/>
      </state>
      <state name="s10" type="NORMAL">
        <outputAction ident="o2.z2"/>
      </state>
      <state name="s9" type="NORMAL">
        <outputAction ident="o1.z2"/>
      </state>
    </state>
    <transition event="e3" sourceRef="s3" targetRef="s5"/>
    <transition event="e103" sourceRef="s5" targetRef="s8"/>
    <transition event="e102" sourceRef="s5" targetRef="s6"/>
    <transition event="e101" sourceRef="s8" targetRef="s2"/>
    <transition event="e101" sourceRef="s6" targetRef="s2"/>
```

```
    <transition event="e1" sourceRef="s2" targetRef="s3"/>
    <transition event="e4" sourceRef="s2" targetRef="s7">
      <outputAction ident="o2.z3"/>
    </transition>
    <transition event="e2" sourceRef="s2" targetRef="s4"/>
    <transition sourceRef="s1" targetRef="s2"/>
    <transition event="e105" sourceRef="s4" targetRef="s10"/>
    <transition event="e104" sourceRef="s4" targetRef="s9"/>
    <transition event="e101" sourceRef="s10" targetRef="s2"/>
    <transition event="e104" sourceRef="s10" targetRef="s9"/>
    <transition event="e101" sourceRef="s9" targetRef="s2"/>
  </stateMachine>
</model>
```

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

```
/**
 * This file was generated from model [Model1] on [Tue Jun 12 21:12:27 MSD
2007].
 * Do not change content of this file.
 */
package ru.ifmo.parking;

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 int decodeStateMachine(String sm) {

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

        return -1;
    }

    private A1EventProcessor _A1;

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

        _A1 = new A1EventProcessor();
    }

    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 ru.ifmo.parking.Elevator o1 = null;
        private ru.ifmo.parking.Parking o3 = null;
        private ru.ifmo.parking.ParkingForm o2 = null;
        private ru.ifmo.parking.ParkingEventProvider humanEventProvider =
null;
        private ru.ifmo.parking.SystemEventProvider systemEventProvider =
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 ru.ifmo.parking.Elevator();
                    }
                    return o1;
                }
                if (StringUtils.equals(coName, "o3")) {
                    if (o3 == null) {
                        o3 = new ru.ifmo.parking.Parking();
                    }
                    return o3;
                }
                if (StringUtils.equals(coName, "o2")) {
                    if (o2 == null) {
                        o2 = new ru.ifmo.parking.ParkingForm();
                    }
                    return o2;
                }
                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("humanEventProvider");
                ep.init(engine);
                ep = getEventProvider("systemEventProvider");
                ep.init(engine);
            }

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

            public EventProvider getEventProvider(String epName) {
                if (StringUtils.equals(epName, "humanEventProvider")) {
                    if (humanEventProvider == null) {
```

```
                        humanEventProvider = new
ru.ifmo.parking.ParkingEventProvider();
                    }
                    return humanEventProvider;
                }
                if (StringUtils.equals(epName, "systemEventProvider")) {
                    if (systemEventProvider == null) {
                        systemEventProvider = new
ru.ifmo.parking.SystemEventProvider();
                    }
                    return systemEventProvider;
                }
                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);
    }

    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);
                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);
                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());
        }

        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");
                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.s7:
```

```
                            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 s3 = 2;
        private static final int s5 = 3;
        private static final int s8 = 4;
        private static final int s6 = 5;
        private static final int s2 = 6;
        private static final int s1 = 7;
        private static final int s7 = 8;
        private static final int s4 = 9;
        private static final int s10 = 10;
        private static final int s9 = 11;

        private int decodeState(String state) {

            if ("Top".equals(state)) {
                return Top;
            } else if ("s3".equals(state)) {
                return s3;
            } else if ("s5".equals(state)) {
                return s5;
            } else if ("s8".equals(state)) {
                return s8;
            } else if ("s6".equals(state)) {
                return s6;
            } else if ("s2".equals(state)) {
                return s2;
            } else if ("s1".equals(state)) {
                return s1;
            } else if ("s7".equals(state)) {
                return s7;
            } else if ("s4".equals(state)) {
                return s4;
            } else if ("s10".equals(state)) {
                return s10;
            } else if ("s9".equals(state)) {
                return s9;
            }

            return -1;
        }

        // events
        private static final int e4 = 1;
        private static final int e2 = 2;
        private static final int e101 = 3;
        private static final int e103 = 4;
```

```
        private static final int e1 = 5;
        private static final int e105 = 6;
        private static final int e104 = 7;
        private static final int e102 = 8;
        private static final int e3 = 9;

        private int decodeEvent(String event) {

            if ("e4".equals(event)) {
                return e4;
            } else if ("e2".equals(event)) {
                return e2;
            } else if ("e101".equals(event)) {
                return e101;
            } else if ("e103".equals(event)) {
                return e103;
            } else if ("e1".equals(event)) {
                return e1;
            } else if ("e105".equals(event)) {
                return e105;
            } else if ("e104".equals(event)) {
                return e104;
            } else if ("e102".equals(event)) {
                return e102;
            } else if ("e3".equals(event)) {
                return e3;
            }

            return -1;
        }

        private ru.ifmo.parking.Elevator o1;
        private ru.ifmo.parking.Parking o3;
        private ru.ifmo.parking.ParkingForm o2;

        private void init(ControlledObjectsMap controlledObjectsMap) {
            o1 = (ru.ifmo.parking.Elevator)
controlledObjectsMap.getControlledObject("o1");
            o3 = (ru.ifmo.parking.Parking)
controlledObjectsMap.getControlledObject("o3");
            o2 = (ru.ifmo.parking.ParkingForm)
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 s3:

                        return;
                    case s5:

                        return;
                    case s8:

                        return;
                    case s6:

                        return;
                    case s2:

                        return;
                    case s1:

                        return;
                    case s7:

                        return;
                    case s4:

                        return;
                    case s10:

                        return;
                    case s9:

                        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, "s1");

                    // s1->s2 [true]/
                    event = Event.NO_EVENT;
                    fireTransitionFound(context, path, "s1", event,
"s1#s2##true");


                    fireComeToState(context, path, "s2");

                    // s2 [o2.z4]
                    fireBeforeOutputActionExecution(context, path,
"s1#s2##true", "o2.z4");
```

```
                    o2.z4(context);

                    fireAfterOutputActionExecution(context, path,
"s1#s2##true", "o2.z4");

                    return new StateMachineConfig("s2");
            }

            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 s3:


                        switch (e) {
                            case e3:

// s3->s5 e3[true]/

                                fireTransitionCandidate(context, path, "s3",
event, "s3#s5#e3#true");


                                fireTransitionFound(context, path, "s3",
event, "s3#s5#e3#true");


                                fireComeToState(context, path, "s5");

// s5 [o3.z1]
                                fireBeforeOutputActionExecution(context, path,
"s3#s5#e3#true", "o3.z1");

                                o3.z1(context);

                                fireAfterOutputActionExecution(context, path,
"s3#s5#e3#true", "o3.z1");
                                return new StateMachineConfig("s5");


                            default:


                                // transition not found
                                return config;
                        }

                    case s5:
```

```
                        switch (e) {
                            case e103:

// s5->s8 e103[true]/

                                fireTransitionCandidate(context, path, "s5",
event, "s5#s8#e103#true");

                                fireTransitionFound(context, path, "s5",
event, "s5#s8#e103#true");

                                fireComeToState(context, path, "s8");

// s8 [o1.z3]
                                fireBeforeOutputActionExecution(context, path,
"s5#s8#e103#true", "o1.z3");

                                o1.z3(context);

                                fireAfterOutputActionExecution(context, path,
"s5#s8#e103#true", "o1.z3");
                                return new StateMachineConfig("s8");

                            case e102:

// s5->s6 e102[true]/

                                fireTransitionCandidate(context, path, "s5",
event, "s5#s6#e102#true");

                                fireTransitionFound(context, path, "s5",
event, "s5#s6#e102#true");

                                fireComeToState(context, path, "s6");

// s6 [o2.z1]
                                fireBeforeOutputActionExecution(context, path,
"s5#s6#e102#true", "o2.z1");

                                o2.z1(context);

                                fireAfterOutputActionExecution(context, path,
"s5#s6#e102#true", "o2.z1");
                                return new StateMachineConfig("s6");

                            default:

                                // transition not found
                                return config;
                        }

                    case s8:
```

```
                        switch (e) {
                            case e101:

// s8->s2 e101[true]/

                                fireTransitionCandidate(context, path, "s8",
event, "s8#s2#e101#true");


                                fireTransitionFound(context, path, "s8",
event, "s8#s2#e101#true");


                                fireComeToState(context, path, "s2");

// s2 [o2.z4]
                                fireBeforeOutputActionExecution(context, path,
"s8#s2#e101#true", "o2.z4");

                                o2.z4(context);

                                fireAfterOutputActionExecution(context, path,
"s8#s2#e101#true", "o2.z4");
                                return new StateMachineConfig("s2");


                            default:


                                // transition not found
                                return config;
                        }

                    case s6:


                        switch (e) {
                            case e101:

// s6->s2 e101[true]/

                                fireTransitionCandidate(context, path, "s6",
event, "s6#s2#e101#true");


                                fireTransitionFound(context, path, "s6",
event, "s6#s2#e101#true");


                                fireComeToState(context, path, "s2");

// s2 [o2.z4]
                                fireBeforeOutputActionExecution(context, path,
"s6#s2#e101#true", "o2.z4");

                                o2.z4(context);

                                fireAfterOutputActionExecution(context, path,
"s6#s2#e101#true", "o2.z4");
```

```
                                return new StateMachineConfig("s2");

                            default:

                                // transition not found
                                return config;
                        }

                    case s2:

                        switch (e) {
                            case e4:

// s2->s7 e4[true]/o2.z3

                                fireTransitionCandidate(context, path, "s2",
event, "s2#s7#e4#true");

                                fireTransitionFound(context, path, "s2",
event, "s2#s7#e4#true");

                                fireBeforeOutputActionExecution(context, path,
"s2#s7#e4#true", "o2.z3");

                                o2.z3(context);

                                fireAfterOutputActionExecution(context, path,
"s2#s7#e4#true", "o2.z3");

                                fireComeToState(context, path, "s7");

// s7 []
                                return new StateMachineConfig("s7");

                            case e2:

// s2->s4 e2[true]/

                                fireTransitionCandidate(context, path, "s2",
event, "s2#s4#e2#true");

                                fireTransitionFound(context, path, "s2",
event, "s2#s4#e2#true");

                                fireComeToState(context, path, "s4");

// s4 [o3.z2]
                                fireBeforeOutputActionExecution(context, path,
"s2#s4#e2#true", "o3.z2");

                                o3.z2(context);

                                fireAfterOutputActionExecution(context, path,
"s2#s4#e2#true", "o3.z2");
```

```
                                return new StateMachineConfig("s4");

                            case e1:
// s2->s3 e1[true]/
                                fireTransitionCandidate(context, path, "s2",
event, "s2#s3#e1#true");

                                fireTransitionFound(context, path, "s2",
event, "s2#s3#e1#true");

                                fireComeToState(context, path, "s3");
// s3 [o1.z1]
                                fireBeforeOutputActionExecution(context, path,
"s2#s3#e1#true", "o1.z1");
                                o1.z1(context);
                                fireAfterOutputActionExecution(context, path,
"s2#s3#e1#true", "o1.z1");
                                return new StateMachineConfig("s3");

                            default:

                                // transition not found
                                return config;
                        }

                    case s4:

                        switch (e) {
                            case e105:
// s4->s10 e105[true]/
                                fireTransitionCandidate(context, path, "s4",
event, "s4#s10#e105#true");

                                fireTransitionFound(context, path, "s4",
event, "s4#s10#e105#true");

                                fireComeToState(context, path, "s10");
// s10 [o2.z2]
                                fireBeforeOutputActionExecution(context, path,
"s4#s10#e105#true", "o2.z2");
                                o2.z2(context);
                                fireAfterOutputActionExecution(context, path,
"s4#s10#e105#true", "o2.z2");
```

```
                                return new StateMachineConfig("s10");

                            case e104:
// s4->s9 e104[true]/
                                fireTransitionCandidate(context, path, "s4",
event, "s4#s9#e104#true");

                                fireTransitionFound(context, path, "s4",
event, "s4#s9#e104#true");

                                fireComeToState(context, path, "s9");
// s9 [o1.z2]
                                fireBeforeOutputActionExecution(context, path,
"s4#s9#e104#true", "o1.z2");
                                o1.z2(context);
                                fireAfterOutputActionExecution(context, path,
"s4#s9#e104#true", "o1.z2");
                                return new StateMachineConfig("s9");

                            default:

                                // transition not found
                                return config;
                        }

                    case s10:

                        switch (e) {
                            case e101:
// s10->s2 e101[true]/
                                fireTransitionCandidate(context, path, "s10",
event, "s10#s2#e101#true");

                                fireTransitionFound(context, path, "s10",
event, "s10#s2#e101#true");

                                fireComeToState(context, path, "s2");
// s2 [o2.z4]
                                fireBeforeOutputActionExecution(context, path,
"s10#s2#e101#true", "o2.z4");
                                o2.z4(context);
                                fireAfterOutputActionExecution(context, path,
"s10#s2#e101#true", "o2.z4");
```

```
                                return new StateMachineConfig("s2");

                            case e104:
// s10->s9 e104[true]/
                                fireTransitionCandidate(context, path, "s10",
event, "s10#s9#e104#true");

                                fireTransitionFound(context, path, "s10",
event, "s10#s9#e104#true");

                                fireComeToState(context, path, "s9");
// s9 [o1.z2]
                                fireBeforeOutputActionExecution(context, path,
"s10#s9#e104#true", "o1.z2");
                                o1.z2(context);
                                fireAfterOutputActionExecution(context, path,
"s10#s9#e104#true", "o1.z2");
                                return new StateMachineConfig("s9");

                            default:

                                // transition not found
                                return config;
                        }

                    case s9:

                        switch (e) {
                            case e101:
// s9->s2 e101[true]/
                                fireTransitionCandidate(context, path, "s9",
event, "s9#s2#e101#true");

                                fireTransitionFound(context, path, "s9",
event, "s9#s2#e101#true");

                                fireComeToState(context, path, "s2");
// s2 [o2.z4]
                                fireBeforeOutputActionExecution(context, path,
"s9#s2#e101#true", "o2.z4");
                                o2.z4(context);
                                fireAfterOutputActionExecution(context, path,
"s9#s2#e101#true", "o2.z4");
```

```
                                return new StateMachineConfig("s2");


                            default:


                                // transition not found
                                return config;
                        }

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


    }

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

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

        return b;
    }

}
```