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

В.Г. Лашманов, А.А. Шалыто

Моделирование работы автоматического путеукладчика на основе автоматного программирования

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

Санкт-Петербург
2006

Оглавление

Введение

В проекте использовано инструментальное средство *UniMod*, основанное на объектно-ориентированном языке *Java,* среде разработки *Eclipse 3.1* и автоматном программировании.

1. Описание проекта

1.1. Описание работы автоматического путеукладчика

В данном проекте рассматривается автоматический путеукладчик, задачей которого является восстановление рельсового пути. Путь произвольной длины может состоять из хороших, испорченных и отсутствующих рельс. Путеукладчик может снимать испорченные и укладывать хорошие рельсы. Он может быть загружен ограниченным числом рельс и перевозить их по пути, используя для замены испорченных и установки вместо отсутствующих. После того, как загруженные рельсы заканчиваются, путеукладчик должен автоматически возвращаться в депо для пополнения запаса рельс. После окончания ремонта пути, путеукладчик также возвращается в депо.

Работа путеукладчика происходит следующим образом:

1. Движение вперед, пока впереди хорошие рельсы.
2. Если впереди испорченные рельсы, то снятие их с пути.
3. Укладка новых рельс.
4. Если в запасе не осталось рельс, то возвращение в депо и пополнение запаса.
5. После достижения конца рельсового пути, возвращение в депо и останов.

В начале работы путеукладчик загружен полностью (пять пар рельс) и находится в депо.

1.2. Постановка задачи

Целью работы является создание программы, моделирующей действия автоматического путеукладчика, с использованием инструментального средства *UniMod*.

На рис. 1 приведен внешний вид программы.

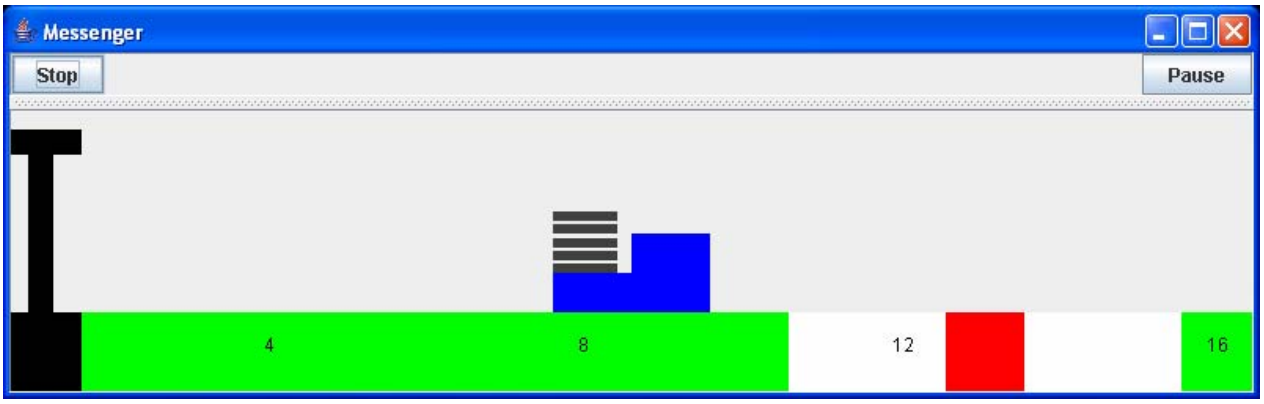

Рис. 1. Внешний вид программы

На этом рисунке синий объект – путеукладчик, а небольшие серые прямоугольники на нем – пары рельс, которые он перевозит. Слева изображено депо, в котором, в частности, происходит загрузка рельс. Путь изображается при помощи зеленых, красных и белых фрагментов. Зеленые фрагменты – исправные рельсы, красные – неисправные, а белые – отсутствующие.

При этом на пути каждый четвертый фрагмент помечен цифрой, указывающей расстояние до него от депо.

Путеукладчик движется слева направо.

Для управления моделированием используются две кнопки. Кнопка, расположенная слева, выполняет две функции: «Start» и «Stop», а кнопка справа – функции «Pause» и «Continue».

2. Проектирование

Проектирование программы выполнено с использованием инструментального средства *UniMod*, которое позволяет строить схему связей (диаграмму классов приложения) и автоматы, описывающие поведение приложения.

На рис. 2 приведена схема связей.

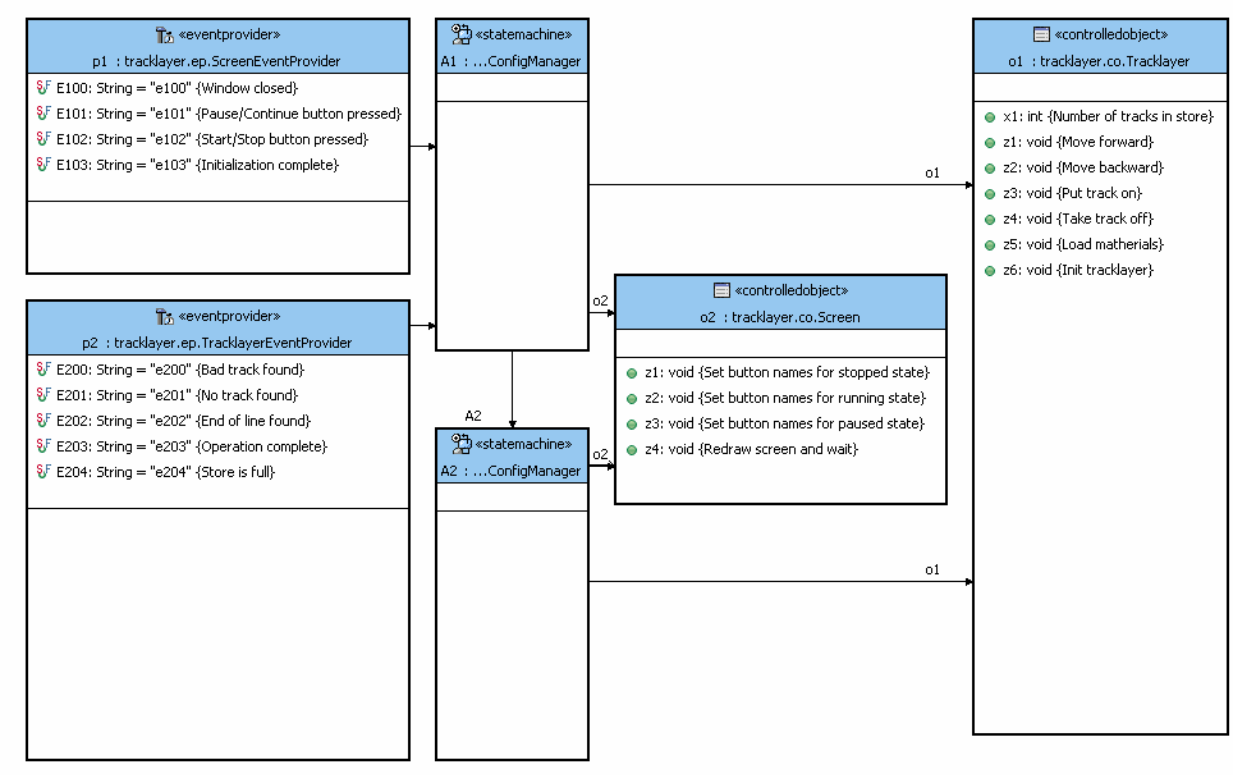

Рис. 2 Схема связей

На этой схеме слева направо изображены: поставщики событий, автоматы и объекты управления.

Опишем составляющие этой схемы:

- p1 – поставщик событий (*EventProvider*) *GUI*:
  - e100 – закрытие окна программы;

    - e101 – нажатие кнопки «Pause/Continue»;
    - e102 – нажатие кнопки «Start/Stop»;
    - e103 – закончена настройка параметров моделирование.
- p2 – поставщик событий (*EventProvider*) путеукладчика. Он создает события:
    - e200 – впереди обнаружена испорченная секция;
    - e201 – впереди обнаружена полностью разрушенная секция;
    - e202 – впереди обнаружен конец рельсового пути;
    - e203 – операция завершена;
    - e204 – материалы загружены.
- A1 – автомат, обеспечивающий работу с *GUI* (запуск, паузу и рестарт).
- A2 – вложенный автомат, обеспечивающий моделирование работы путеукладчика.
- o1 – объект управления (*ControlledObject*) – *GUI*. Он предоставляет методы (выходные воздействия), вызываемые из автомата *A1*:
    - z1 – настроить кнопки для режима останова;
    - z2 – настроить кнопки для режима моделирования;
    - z3 – настроить кнопки для режима паузы;
    - z4 – перерисовать экран.
- o2 – объект управления (*ControlledObject*) – путеукладчик, предоставляет методы, вызываемые из автоматов *A1* и *A2*. Эти методы делятся на входные переменные:
    - x1 – количество секций в запасе;

  и выходные воздействия:
    - z1 – двигаться вперед;
    - z2 – двигаться назад;
    - z3 – уложить секцию;
    - z4 – снять секцию;
    - z5 – пополнить запас;
    - z6 – инициализировать путеукладчик и путь.

2.1. Автомат A1

2.1.1. Описание

Этот автомат обеспечивает работу приложения с *GUI*.

2.1.2. Принцип работы

При запуске приложение инициализируется. После этого при нажатии кнопки «Start/Stop» начинается моделирование. При повторном нажатии этой кнопки работа останавливается, и автомат возвращается в состояние инициализации. При нажатии кнопки «Pause/Continue» работа приостанавливается и возобновляется при повторном нажатии.

2.1.3. Состояния

Автомат *A1* имеет следующие состояния:

1. *Initialization* – в этом состоянии происходит конфигурация параметров моделирования:
    - после вхождения в это состояние изменяются названия кнопок *GUI* и запускается конфигурация параметров;

- по нажатию кнопки «Start/Stop» переходит в состояние *Work*.

2. *Pause* – в этом состоянии автомат находится во время приостановки моделирования:
   - после вхождения в это состояние изменяются названия кнопок и перерисовывается экран;
   - по нажатию кнопки «Start/Stop» автомат переходит в состояние *Initialization*;
   - по нажатию кнопки «Pause/Continue» автомат переходит в состояние *Work*.
3. *Work* – рабочее состояние, в котором выполняется моделирование:
   - после вхождения в это состояние изменяются названия кнопок *GUI*; перерисовывается экран и запускается вложенный автомат *A2*, осуществляющий моделирование;
   - по нажатию кнопки «Start/Stop» автомат переходит в состояние *Initialization*;
   - по нажатию кнопки «Pause/Continue» автомат переходит в состояние *Pause*.

При закрытии окна автомат из любого состояния переходит в конечное.

2.1.4. Граф переходов

Граф переходов автомата *A1* представлен на рис. 3.

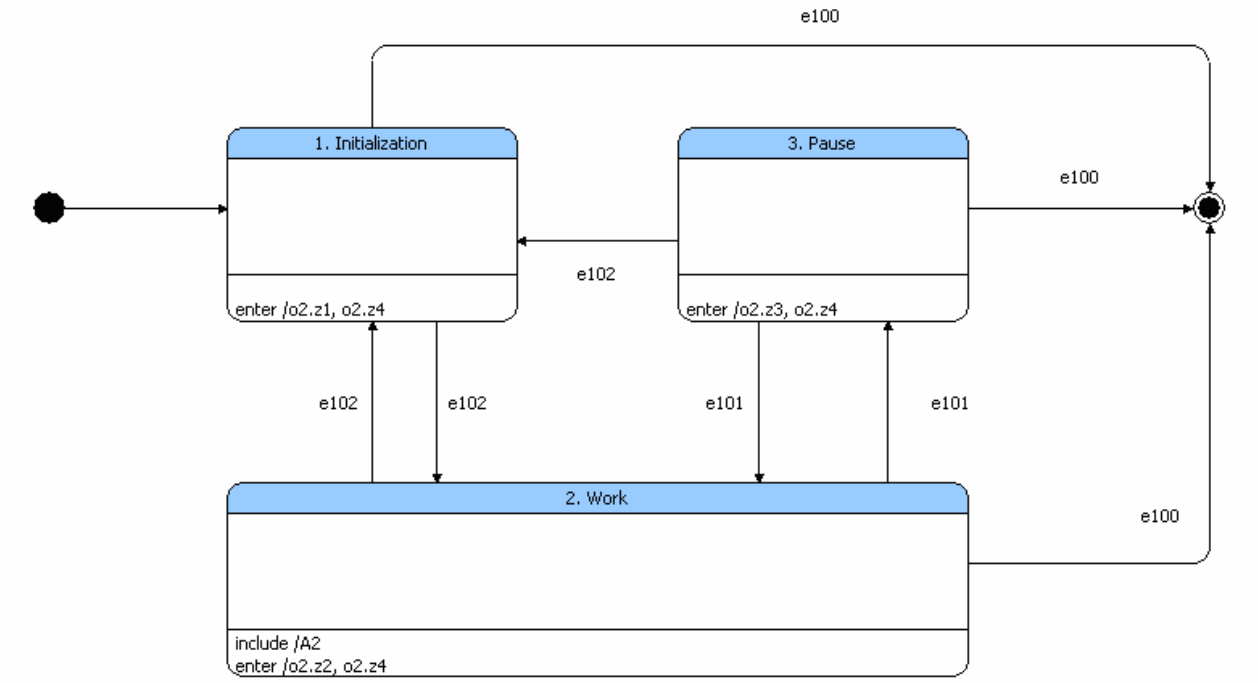

Рис. 3. Граф переходов автомата *A1*

2.2. Автомат A2

2.2.1. Описание

Автомат *A2* обеспечивает собственно моделирование и реализует логику путеукладчика.

2.2.2. Принцип работы

Путеукладчик инициализируется. Потом он начинает двигаться вперед до тех пор, пока не кончатся хорошие рельсы. Если впереди испорченные рельсы, то путеукладчик снимет их и положит новые рельсы. Если впереди рельсы отсутствуют, то путеукладчик положит новые рельсы.

После укладки новых рельс, если запас еще существует, путеукладчик продолжит движение вперед. Если же рельсы кончились, он вернется в депо и загрузит новые в запас. После этого путеукладчик снова начнет двигаться вперед. Если при движении вперед достигается конец пути, путеукладчик возвращается в депо и останавливается.

2.2.3. Состояния

Автомат *A2* имеет следующие состояния:

1. *Initialization* – в этом состоянии происходит инициализация путеукладчика:
   - после вхождения в это состояние изменяются названия кнопок, и запускается инициализация путеукладчика, по завершении которой генерируется событие *e203* («Операция завершена»);
   - после завершения инициализации, если в запасе есть рельсы, путеукладчик переходит в состояние *Move forward;*
   - после завершения инициализации, если в запасе нет рельс, путеукладчик переходит в состояние *Move backward.*
2. *Moving forward* – в этом состоянии путеукладчик двигается вперед:
   - после вхождения в это состояние, если впереди хорошие рельсы, путеукладчик движется вперед на один шаг. После завершения шага генерируется событие *e203* («Операция завершена»). В противном случае путеукладчик остается на месте и генерирует в зависимости от условий одно из событий *e200* («Впереди испорченные рельсы»), *e201* («Впереди отсутствуют рельсы») или *e202* («Достигнут конец пути»);
   - по событию *e203* («Операция завершена») автомат сохраняет состояние;
   - по событию *e200* («Впереди испорченная секция») автомат переходит в состояние *Taking off old rails*;
   - по событию *e201* («Впереди отсутствуют рельсы») автомат переходит в состояние *Putting new rails*;
   - по событию *e202* («Достигнут конец пути») автомат переходит в состояние *Shutting down*.
3. *Taking off old rails*:
   - после вхождения в это состояние, укладчик снимает следующую секцию рельс и генерирует событие *e203* («Операция завершена»);
   - по событию *e203* («Операция завершена») автомат переходит в состояние *Putting new rails*.
4. *Putting new rails*:
   - после вхождения в это состояние, укладчик выбирает секцию из запаса и укладывает ее перед собой. После этого генерируется событие *e203* («Операция завершена»);
   - по событию *e203* («Операция завершена»), если в запасе есть рельсы, автомат переходит в состояние *Move forward*;

- по событию *e203* («Операция завершена»), если в запасе нет рельс, автомат переходит в состояние *Move backward*.

5. *Moving backward*:
   - после вхождения в это состояние, если путеукладчик не в депо, он двигается на один шаг назад. После выполнения шага генерируется событие *e203* («Операция завершена»). В противном случае, генерируется событие *e202* («Достигнут конец пути»);
   - по событию *e203* («Операция завершена») автомат сохраняет состояние;
   - по событию *e202* («Достигнут конец рельсового пути») автомат переходит в состояние *Loading*.
6. *Loading*:
   - после вхождения в это состояние, если путеукладчик содержит не максимальное количество рельс, он загружает одну пару рельс в запас. После этого генерируется событие *e203* («Операция завершена»). В противном случае, генерируется событие *e204* («Материалы загружены»);
   - по событию *e203* («Операция завершена») автомат сохраняет состояние;
   - по событию *e204* («Материалы загружены») автомат переходит в состояние *Moving forward*.
7. *Shutting down*:
   - после вхождения в это состояние, если путеукладчик не в депо, он движется назад. После этого генерируется событие *e203* («Операция завершена»);
   - по событию *e203* («Операция завершена») автомат сохраняет состояние.

В заключение раздела отметим, что по событию *e101* («Pause/Continue») автомат сохраняет состояние, а по событию *e102* («Start/Stop») автомат из любого состояния переходит в состояние *Initialization*.

2.2.4. Граф переходов

Граф переходов автомата *A2* представлен на рис. 4.

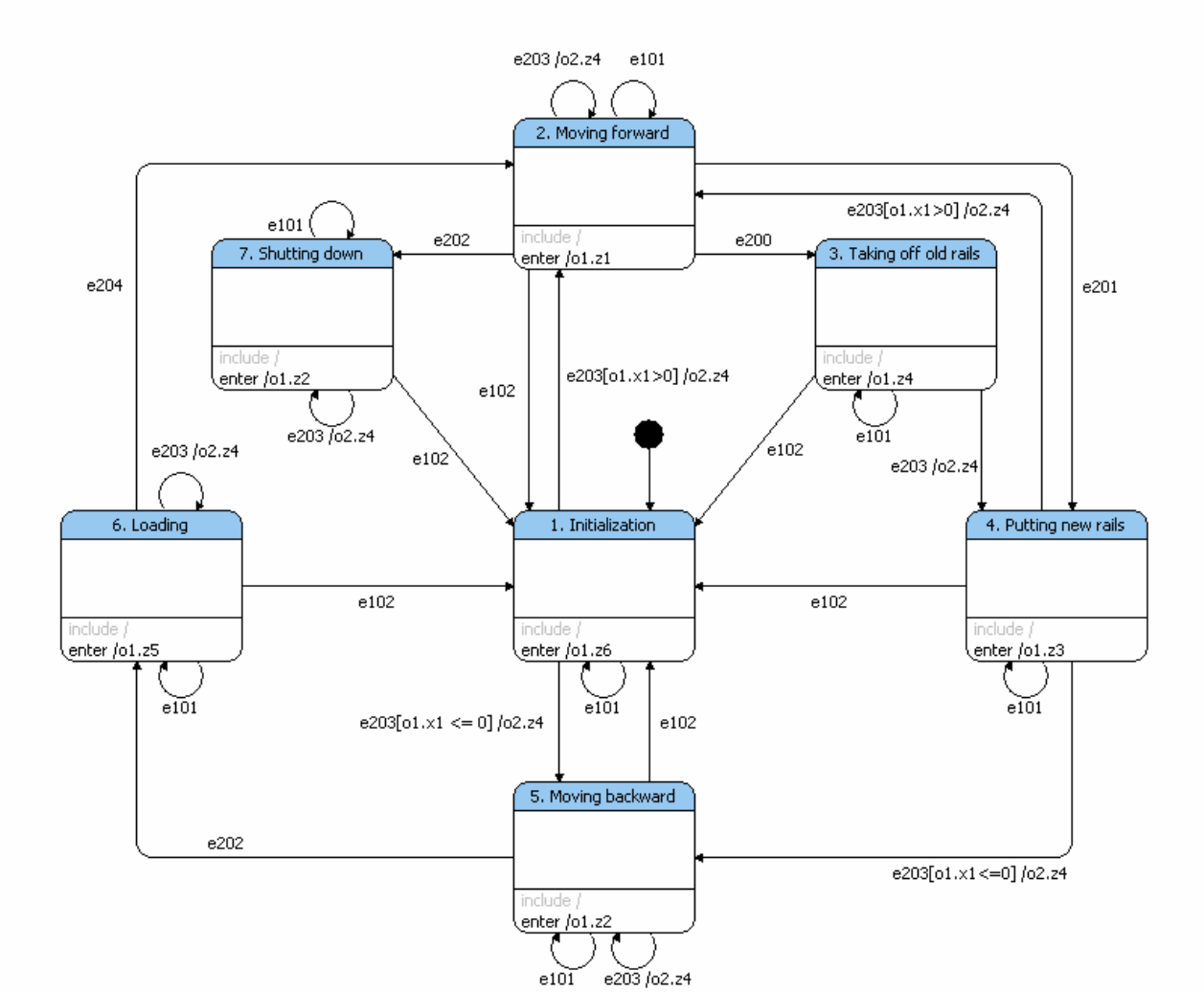

Рис. 4. Граф переходов автомата *A2*

4. Реализация

Проект реализован на языке *Java* и содержит пакеты:

- `ru.ifmo.tracklayer.co` – объекты управления:
  - `Screen.java` – объект управления GUI;
  - `Tracklayer.java` – объект управления «Путеукладчик»;
- `ru.ifmo.tracklayer.ep` – поставщики событий:
  - `ScreenEventProvider.java` – поставщик событий GUI;
  - `TracklayerEventProvider.java` – поставщик событий путеукладчика;
- `ru.ifmo.tracklayer.gui` – реализация GUI:
  - `GUIFrame.java` – код, создающий и поддерживающий главное окно программы;
- `ru.ifmo.tracklayer.runner` – классы, необходимые для запуска программы при компилятивном подходе:
  - `TracklayerRunner.java` – код, создающий и запускающий программу;
  - `Model1EventProcessor.java` – автоматически сгенерированный по диаграммам переходов *java*-код.

4.1. Интерпретационный подход

При интерпретационном подходе программа напрямую использует файл с `xml`-описанием автоматов, интерпретируя его средствами *UniMod*, совместно с кодом входных и выходных воздействий, написанным вручную. При этом программе требуется достаточно много *UniMod*-библиотек, что препятствует распространению программы с использованием этого подхода (суммарно около 2 Мб). Чтобы уменьшить число библиотек, от которых зависит программа, применяется компилятивный подход.

4.2. Компилятивный подход

При компилятивном подходе `xml`-файл, описывающий автоматы, преобразуется в *Java*-код, что позволяет (совместно с кодом входных и выходных воздействий, написанным вручную) запускать программу без использования интерпретатора *UniMod*. Для запуска программы по-прежнему потребуются некоторые библиотеки *UniMod*, но их будет намного меньше (суммарно около 200 Кб).

Заключение

Автоматный подход очень удобно использовать для создания программного обеспечения, реальных систем и их моделей. Явное выделение состояний делает логику более удобной для восприятия и, при необходимости, последующей доработки. Автоматный подход также значительно упрощает отладку программы и дает возможность создать простое и информативное протоколирование действий системы.

Источники

1. *Шалыто А.А.* Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998. http://is.ifmo.ru/books/switch/1
2. *Сайт* по автоматному программированию. http://is.ifmo.ru/
3. *UniMod* project. http://unimod.sourceforge.net/
4. *Среда разработки Eclipse*. http://www.eclipse.org/
5. *Хопкрафт Д.*, *Мотвани Р.*, *Ульман Д.* Введение в теорию автоматов, языков и вычислений. М.: Вильямс, 2002.

Приложение 1. Пример протокола работы программы

23:01:23,775 INFO [Run] Start event [e102] processing. In state [/A1:Top]
23:01:23,802 INFO [Run] Transition to go found [s1#1. Initialization##true]
23:01:23,804 INFO [Run] Start on-enter action [o2.z1] execution
23:01:23,810 INFO [Run] Finish on-enter action [o2.z1] execution
23:01:23,810 INFO [Run] Start on-enter action [o2.z4] execution
23:01:24,517 INFO [Run] Finish on-enter action [o2.z4] execution
23:01:24,518 DEBUG [Run] Try transition [1. Initialization#2. Work#e102#true]
23:01:24,523 INFO [Run] Transition to go found [1. Initialization#2. Work#e102#true]
23:01:24,530 INFO [Run] Start on-enter action [o2.z2] execution
23:01:24,536 INFO [Run] Finish on-enter action [o2.z2] execution
23:01:24,536 INFO [Run] Start on-enter action [o2.z4] execution
23:01:25,245 INFO [Run] Finish on-enter action [o2.z4] execution
23:01:25,250 INFO [Run] Start event [e102] processing. In state [/A1:2. Work/A2:Top]
23:01:25,254 INFO [Run] Transition to go found [s1#1. Initialization##true]
23:01:25,256 INFO [Run] Start on-enter action [o1.z6] execution
23:01:25,258 INFO [Run] Finish on-enter action [o1.z6] execution
23:01:25,258 DEBUG [Run] Try transition [1. Initialization#1. Initialization#e102#true]
23:01:25,260 INFO [Run] Transition to go found [1. Initialization#1. Initialization#e102#true]
23:01:25,260 INFO [Run] Start output action [o1.z6] execution
23:01:25,264 INFO [Run] Finish output action [o1.z6] execution
23:01:25,266 INFO [Run] Start on-enter action [o1.z6] execution
23:01:25,268 INFO [Run] Finish on-enter action [o1.z6] execution
23:01:25,268 INFO [Run] Finish event [e102] processing. In state [/A1:2. Work/A2:1. Initialization]
23:01:25,268 INFO [Run] Finish event [e102] processing. In state [/A1:2. Work]
23:01:25,269 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:25,272 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:1. Initialization]
23:01:25,276 DEBUG [Run] Try transition [1. Initialization#2. Moving forward#e203#o1.x1>0]
23:01:25,277 INFO [Run] Start input action [o1.x1] calculation
23:01:25,279 INFO [Run] Finish input action [o1.x1] calculation. Its value is [5]
23:01:25,280 INFO [Run] Transition to go found [1. Initialization#2. Moving forward#e203#o1.x1>0]
23:01:25,280 INFO [Run] Start output action [o2.z4] execution
23:01:25,985 INFO [Run] Finish output action [o2.z4] execution
23:01:25,986 INFO [Run] Start on-enter action [o1.z1] execution
23:01:25,991 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:25,991 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:25,992 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:25,992 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]

23:01:25,995 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:25,997 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:25,998 INFO [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:25,998 INFO [Run] Start output action [o2.z4] execution
23:01:26,705 INFO [Run] Finish output action [o2.z4] execution
23:01:26,706 INFO [Run] Start on-enter action [o1.z1] execution
23:01:26,707 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:26,707 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:26,707 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:26,707 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:26,708 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:26,709 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:26,710 INFO [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:26,710 INFO [Run] Start output action [o2.z4] execution
23:01:27,417 INFO [Run] Finish output action [o2.z4] execution
23:01:27,418 INFO [Run] Start on-enter action [o1.z1] execution
23:01:27,419 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:27,419 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:27,419 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:27,419 INFO [Run] Start event [e201] processing. In state [/A1:2. Work]
23:01:27,421 INFO [Run] Start event [e201] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:27,421 DEBUG [Run] Try transition [2. Moving forward#4. Putting new rails#e201#true]
23:01:27,422 INFO [Run] Transition to go found [2. Moving forward#4. Putting new rails#e201#true]
23:01:27,423 INFO [Run] Start on-enter action [o1.z3] execution
23:01:27,424 INFO [Run] Finish on-enter action [o1.z3] execution
23:01:27,424 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:27,424 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work]
23:01:27,424 INFO [Run] Start event [e201] processing. In state [/A1:2. Work]
23:01:27,426 INFO [Run] Start event [e201] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:27,429 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:27,429 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work]
23:01:27,429 INFO [Run] Start event [e201] processing. In state [/A1:2. Work]
23:01:27,431 INFO [Run] Start event [e201] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:27,431 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:27,432 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work]
23:01:27,432 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:27,433 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:27,436 DEBUG [Run] Try transition [4. Putting new rails#2. Moving forward#e203#o1.x1>0]

23:01:27,437 INFO [Run] Start input action [o1.x1] calculation
23:01:27,438 INFO [Run] Finish input action [o1.x1] calculation. Its value is [4]
23:01:27,438 INFO [Run] Transition to go found [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:27,438 INFO [Run] Start output action [o2.z4] execution
23:01:28,145 INFO [Run] Finish output action [o2.z4] execution
23:01:28,146 INFO [Run] Start on-enter action [o1.z1] execution
23:01:28,147 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:28,147 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:28,147 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:28,147 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:28,148 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:28,149 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:28,149 INFO [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:28,149 INFO [Run] Start output action [o2.z4] execution
23:01:28,853 INFO [Run] Finish output action [o2.z4] execution
23:01:28,855 INFO [Run] Start on-enter action [o1.z1] execution
23:01:28,857 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:28,857 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:28,858 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:28,858 INFO [Run] Start event [e201] processing. In state [/A1:2. Work]
23:01:28,860 INFO [Run] Start event [e201] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:28,860 DEBUG [Run] Try transition [2. Moving forward#4. Putting new rails#e201#true]
23:01:28,861 INFO [Run] Transition to go found [2. Moving forward#4. Putting new rails#e201#true]
23:01:28,862 INFO [Run] Start on-enter action [o1.z3] execution
23:01:28,864 INFO [Run] Finish on-enter action [o1.z3] execution
23:01:28,864 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:28,864 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work]
23:01:28,864 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:28,865 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:28,866 DEBUG [Run] Try transition [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:28,867 INFO [Run] Start input action [o1.x1] calculation
23:01:28,867 INFO [Run] Finish input action [o1.x1] calculation. Its value is [3]
23:01:28,867 INFO [Run] Transition to go found [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:28,868 INFO [Run] Start output action [o2.z4] execution
23:01:29,573 INFO [Run] Finish output action [o2.z4] execution
23:01:29,574 INFO [Run] Start on-enter action [o1.z1] execution
23:01:29,575 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:29,575 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]

23:01:29,575 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:29,575 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:29,577 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:29,577 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:29,581 INFO [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:29,581 INFO [Run] Start output action [o2.z4] execution
23:01:30,285 INFO [Run] Finish output action [o2.z4] execution
23:01:30,287 INFO [Run] Start on-enter action [o1.z1] execution
23:01:30,287 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:30,288 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:30,288 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:30,288 INFO [Run] Start event [e201] processing. In state [/A1:2. Work]
23:01:30,290 INFO [Run] Start event [e201] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:30,291 DEBUG [Run] Try transition [2. Moving forward#4. Putting new rails#e201#true]
23:01:30,291 INFO [Run] Transition to go found [2. Moving forward#4. Putting new rails#e201#true]
23:01:30,292 INFO [Run] Start on-enter action [o1.z3] execution
23:01:30,293 INFO [Run] Finish on-enter action [o1.z3] execution
23:01:30,293 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:30,293 INFO [Run] Finish event [e201] processing. In state [/A1:2. Work]
23:01:30,293 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:30,295 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:30,296 DEBUG [Run] Try transition [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:30,297 INFO [Run] Start input action [o1.x1] calculation
23:01:30,297 INFO [Run] Finish input action [o1.x1] calculation. Its value is [2]
23:01:30,298 INFO [Run] Transition to go found [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:30,298 INFO [Run] Start output action [o2.z4] execution
23:01:31,002 INFO [Run] Finish output action [o2.z4] execution
23:01:31,003 INFO [Run] Start on-enter action [o1.z1] execution
23:01:31,004 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:31,004 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:31,005 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:31,005 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:31,006 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:31,007 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:31,007 INFO [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:31,007 INFO [Run] Start output action [o2.z4] execution
23:01:31,714 INFO [Run] Finish output action [o2.z4] execution
23:01:31,714 INFO [Run] Start on-enter action [o1.z1] execution

23:01:31,715 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:31,715 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:31,715 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:31,716 INFO [Run] Start event [e200] processing. In state [/A1:2. Work]
23:01:31,716 INFO [Run] Start event [e200] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:31,718 DEBUG [Run] Try transition [2. Moving forward#3. Taking off old rails#e200#true]
23:01:31,718 INFO [Run] Transition to go found [2. Moving forward#3. Taking off old rails#e200#true]
23:01:31,719 INFO [Run] Start on-enter action [o1.z4] execution
23:01:31,720 INFO [Run] Finish on-enter action [o1.z4] execution
23:01:31,720 INFO [Run] Finish event [e200] processing. In state [/A1:2. Work/A2:3. Taking off old rails]
23:01:31,720 INFO [Run] Finish event [e200] processing. In state [/A1:2. Work]
23:01:31,721 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:31,722 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:3. Taking off old rails]
23:01:31,722 DEBUG [Run] Try transition [3. Taking off old rails#4. Putting new rails#e203#true]
23:01:31,724 INFO [Run] Transition to go found [3. Taking off old rails#4. Putting new rails#e203#true]
23:01:31,724 INFO [Run] Start output action [o2.z4] execution
23:01:32,430 INFO [Run] Finish output action [o2.z4] execution
23:01:32,431 INFO [Run] Start on-enter action [o1.z3] execution
23:01:32,432 INFO [Run] Finish on-enter action [o1.z3] execution
23:01:32,432 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:32,432 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:32,432 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:32,433 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:32,434 DEBUG [Run] Try transition [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:32,434 INFO [Run] Start input action [o1.x1] calculation
23:01:32,435 INFO [Run] Finish input action [o1.x1] calculation. Its value is [1]
23:01:32,435 INFO [Run] Transition to go found [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:32,435 INFO [Run] Start output action [o2.z4] execution
23:01:33,142 INFO [Run] Finish output action [o2.z4] execution
23:01:33,142 INFO [Run] Start on-enter action [o1.z1] execution
23:01:33,143 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:33,143 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:33,143 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:33,144 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:33,145 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:33,146 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:33,147 INFO [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:33,147 INFO [Run] Start output action [o2.z4] execution

23:01:33,854 INFO [Run] Finish output action [o2.z4] execution
23:01:33,855 INFO [Run] Start on-enter action [o1.z1] execution
23:01:33,856 INFO [Run] Finish on-enter action [o1.z1] execution
23:01:33,856 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:33,856 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:33,856 INFO [Run] Start event [e200] processing. In state [/A1:2. Work]
23:01:33,857 INFO [Run] Start event [e200] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:33,858 DEBUG [Run] Try transition [2. Moving forward#3. Taking off old rails#e200#true]
23:01:33,859 INFO [Run] Transition to go found [2. Moving forward#3. Taking off old rails#e200#true]
23:01:33,862 INFO [Run] Start on-enter action [o1.z4] execution
23:01:33,863 INFO [Run] Finish on-enter action [o1.z4] execution
23:01:33,863 INFO [Run] Finish event [e200] processing. In state [/A1:2. Work/A2:3. Taking off old rails]
23:01:33,863 INFO [Run] Finish event [e200] processing. In state [/A1:2. Work]
23:01:33,863 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:33,864 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:3. Taking off old rails]
23:01:33,865 DEBUG [Run] Try transition [3. Taking off old rails#4. Putting new rails#e203#true]
23:01:33,866 INFO [Run] Transition to go found [3. Taking off old rails#4. Putting new rails#e203#true]
23:01:33,866 INFO [Run] Start output action [o2.z4] execution
23:01:34,574 INFO [Run] Finish output action [o2.z4] execution
23:01:34,574 INFO [Run] Start on-enter action [o1.z3] execution
23:01:34,575 INFO [Run] Finish on-enter action [o1.z3] execution
23:01:34,575 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:34,576 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:34,576 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:34,577 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:4. Putting new rails]
23:01:34,577 DEBUG [Run] Try transition [4. Putting new rails#2. Moving forward#e203#o1.x1>0]
23:01:34,578 INFO [Run] Start input action [o1.x1] calculation
23:01:34,578 INFO [Run] Finish input action [o1.x1] calculation. Its value is [0]
23:01:34,579 DEBUG [Run] Try transition [4. Putting new rails#5. Moving backward#e203#o1.x1<=0]
23:01:34,580 INFO [Run] Transition to go found [4. Putting new rails#5. Moving backward#e203#o1.x1<=0]
23:01:34,580 INFO [Run] Start output action [o2.z4] execution
23:01:35,290 INFO [Run] Finish output action [o2.z4] execution
23:01:35,292 INFO [Run] Start on-enter action [o1.z2] execution
23:01:35,293 INFO [Run] Finish on-enter action [o1.z2] execution
23:01:35,293 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:35,293 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:35,293 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:35,295 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]

23:01:35,296 DEBUG [Run] Try transition [5. Moving backward#5. Moving backward#e203#true]
23:01:35,298 INFO [Run] Transition to go found [5. Moving backward#5. Moving backward#e203#true]
23:01:35,298 INFO [Run] Start output action [o2.z4] execution
23:01:36,002 INFO [Run] Finish output action [o2.z4] execution
23:01:36,003 INFO [Run] Start on-enter action [o1.z2] execution
23:01:36,004 INFO [Run] Finish on-enter action [o1.z2] execution
23:01:36,004 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:36,004 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:36,004 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:36,007 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:36,008 DEBUG [Run] Try transition [5. Moving backward#5. Moving backward#e203#true]
23:01:36,009 INFO [Run] Transition to go found [5. Moving backward#5. Moving backward#e203#true]
23:01:36,009 INFO [Run] Start output action [o2.z4] execution
23:01:36,714 INFO [Run] Finish output action [o2.z4] execution
23:01:36,719 INFO [Run] Start on-enter action [o1.z2] execution
23:01:36,720 INFO [Run] Finish on-enter action [o1.z2] execution
23:01:36,720 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:36,720 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:36,720 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:36,725 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:36,726 DEBUG [Run] Try transition [5. Moving backward#5. Moving backward#e203#true]
23:01:36,727 INFO [Run] Transition to go found [5. Moving backward#5. Moving backward#e203#true]
23:01:36,727 INFO [Run] Start output action [o2.z4] execution
23:01:37,434 INFO [Run] Finish output action [o2.z4] execution
23:01:37,435 INFO [Run] Start on-enter action [o1.z2] execution
23:01:37,435 INFO [Run] Finish on-enter action [o1.z2] execution
23:01:37,436 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:37,436 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:37,436 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:37,437 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:37,438 DEBUG [Run] Try transition [5. Moving backward#5. Moving backward#e203#true]
23:01:37,439 INFO [Run] Transition to go found [5. Moving backward#5. Moving backward#e203#true]
23:01:37,439 INFO [Run] Start output action [o2.z4] execution
23:01:38,146 INFO [Run] Finish output action [o2.z4] execution
23:01:38,147 INFO [Run] Start on-enter action [o1.z2] execution
23:01:38,149 INFO [Run] Finish on-enter action [o1.z2] execution
23:01:38,149 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:38,149 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]

23:01:38,149 INFO [Run] Start event [e202] processing. In state [/A1:2. Work]
23:01:38,150 INFO [Run] Start event [e202] processing. In state [/A1:2. Work/A2:5. Moving backward]
23:01:38,151 DEBUG [Run] Try transition [5. Moving backward#6. Loading#e202#true]
23:01:38,151 INFO [Run] Transition to go found [5. Moving backward#6. Loading#e202#true]
23:01:38,152 INFO [Run] Start on-enter action [o1.z5] execution
23:01:38,152 INFO [Run] Finish on-enter action [o1.z5] execution
23:01:38,153 INFO [Run] Finish event [e202] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:38,153 INFO [Run] Finish event [e202] processing. In state [/A1:2. Work]
23:01:38,153 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:38,154 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:38,155 DEBUG [Run] Try transition [6. Loading#6. Loading#e203#true]
23:01:38,155 INFO [Run] Transition to go found [6. Loading#6. Loading#e203#true]
23:01:38,155 INFO [Run] Start output action [o2.z4] execution
23:01:38,862 INFO [Run] Finish output action [o2.z4] execution
23:01:38,863 INFO [Run] Start on-enter action [o1.z5] execution
23:01:38,864 INFO [Run] Finish on-enter action [o1.z5] execution
23:01:38,864 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:38,865 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:38,865 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:38,866 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:38,867 DEBUG [Run] Try transition [6. Loading#6. Loading#e203#true]
23:01:38,870 INFO [Run] Transition to go found [6. Loading#6. Loading#e203#true]
23:01:38,870 INFO [Run] Start output action [o2.z4] execution
23:01:39,574 INFO [Run] Finish output action [o2.z4] execution
23:01:39,575 INFO [Run] Start on-enter action [o1.z5] execution
23:01:39,575 INFO [Run] Finish on-enter action [o1.z5] execution
23:01:39,576 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:39,576 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:39,576 INFO [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:39,577 INFO [Run] Start event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:39,578 DEBUG [Run] Try transition [6. Loading#6. Loading#e203#true]
23:01:39,579 INFO [Run] Transition to go found [6. Loading#6. Loading#e203#true]
23:01:39,579 INFO [Run] Start output action [o2.z4] execution
23:01:40,286 INFO [Run] Finish output action [o2.z4] execution
23:01:40,287 INFO [Run] Start on-enter action [o1.z5] execution
23:01:40,288 INFO [Run] Finish on-enter action [o1.z5] execution
23:01:40,288 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:40,288 INFO [Run] Finish event [e203] processing. In state [/A1:2. Work]

```
23:01:40,288 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:40,289 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:40,292 DEBUG [Run] Try transition [6. Loading#6. Loading#e203#true]
23:01:40,293 INFO  [Run] Transition to go found [6. Loading#6. Loading#e203#true]
23:01:40,293 INFO  [Run] Start output action [o2.z4] execution
23:01:40,998 INFO  [Run] Finish output action [o2.z4] execution
23:01:40,999 INFO  [Run] Start on-enter action [o1.z5] execution
23:01:40,999 INFO  [Run] Finish on-enter action [o1.z5] execution
23:01:41,000 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:41,000 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:41,000 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:41,002 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:41,003 DEBUG [Run] Try transition [6. Loading#6. Loading#e203#true]
23:01:41,004 INFO  [Run] Transition to go found [6. Loading#6. Loading#e203#true]
23:01:41,004 INFO  [Run] Start output action [o2.z4] execution
23:01:41,710 INFO  [Run] Finish output action [o2.z4] execution
23:01:41,711 INFO  [Run] Start on-enter action [o1.z5] execution
23:01:41,714 INFO  [Run] Finish on-enter action [o1.z5] execution
23:01:41,714 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:41,714 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:41,715 INFO  [Run] Start event [e204] processing. In state [/A1:2. Work]
23:01:41,716 INFO  [Run] Start event [e204] processing. In state [/A1:2. Work/A2:6. Loading]
23:01:41,717 DEBUG [Run] Try transition [6. Loading#2. Moving forward#e204#true]
23:01:41,717 INFO  [Run] Transition to go found [6. Loading#2. Moving forward#e204#true]
23:01:41,724 INFO  [Run] Start on-enter action [o1.z1] execution
23:01:41,725 INFO  [Run] Finish on-enter action [o1.z1] execution
23:01:41,726 INFO  [Run] Finish event [e204] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:41,726 INFO  [Run] Finish event [e204] processing. In state [/A1:2. Work]
23:01:41,726 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:41,727 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:41,728 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:41,728 INFO  [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:41,728 INFO  [Run] Start output action [o2.z4] execution
23:01:42,434 INFO  [Run] Finish output action [o2.z4] execution
23:01:42,435 INFO  [Run] Start on-enter action [o1.z1] execution
23:01:42,436 INFO  [Run] Finish on-enter action [o1.z1] execution
23:01:42,436 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:42,436 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work]
```

```
23:01:42,436 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work]
23:01:42,438 INFO  [Run] Start event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:42,440 DEBUG [Run] Try transition [2. Moving forward#2. Moving forward#e203#true]
23:01:42,440 INFO  [Run] Transition to go found [2. Moving forward#2. Moving forward#e203#true]
23:01:42,440 INFO  [Run] Start output action [o2.z4] execution
23:01:43,146 INFO  [Run] Finish output action [o2.z4] execution
23:01:43,148 INFO  [Run] Start on-enter action [o1.z1] execution
23:01:43,149 INFO  [Run] Finish on-enter action [o1.z1] execution
23:01:43,149 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work/A2:2. Moving forward]
23:01:43,150 INFO  [Run] Finish event [e203] processing. In state [/A1:2. Work]
23:01:43,150 INFO  [Run] Start event [e102] processing. In state [/A1:2. Work]
23:01:43,150 DEBUG [Run] Try transition [2. Work#1. Initialization#e102#true]
23:01:43,152 INFO  [Run] Transition to go found [2. Work#1. Initialization#e102#true]
23:01:43,153 INFO  [Run] Start on-enter action [o2.z1] execution
23:01:43,154 INFO  [Run] Finish on-enter action [o2.z1] execution
23:01:43,155 INFO  [Run] Start on-enter action [o2.z4] execution
23:01:43,858 INFO  [Run] Finish on-enter action [o2.z4] execution
23:01:43,858 INFO  [Run] Finish event [e102] processing. In state [/A1:1. Initialization]
23:01:43,859 INFO  [Run] Start event [e203] processing. In state [/A1:1. Initialization]
23:01:43,859 INFO  [Run] Finish event [e203] processing. In state [/A1:1. Initialization]
```

Приложение 2. Исходные коды программы

2.1. Поставщики событий

2.1.1 `ScreenEventProvider.java`

```
package ru.ifmo.tracklayer.ep;

import javax.swing.SwingUtilities;

import ru.ifmo.tracklayer.gui.GUIFrame;

import com.evelopers.unimod.core.stateworks.Event;
import com.evelopers.unimod.runtime.EventProvider;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.Parameter;
import com.evelopers.unimod.runtime.context.StateMachineContextImpl;

public class ScreenEventProvider implements EventProvider {

        /**
         * @unimod.event.descr Window closed
         */
    public static final String E100 = "e100";

    /**
         * @unimod.event.descr Pause/Continue button pressed
         */
    public static final String E101 = "e101";

    /**
         * @unimod.event.descr Start/Stop button pressed
         */
    public static final String E102 = "e102";

    /**
         * @unimod.event.descr Initialization complete
         */
    public static final String E103 = "e103";

    private static ModelEngine engine;

    public void init(ModelEngine engine) {
        ScreenEventProvider.engine = engine;

        try {
                // invoke in Swing event thread
            SwingUtilities.invokeAndWait(new Runnable() {
                public void run() {
                        GUIFrame.getFrame().setVisible(true);
                }
            });
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    /* (non-Javadoc)
     * @see com.evelopers.unimod.runtime.EventProvider#dispose()
     */
    public void dispose() {
        try {
                // invoke in Swing event thread
```

```
            SwingUtilities.invokeAndWait(new Runnable() {
                public void run() {
                        GUIFrame.getFrame().setVisible(false);
                        GUIFrame.getFrame().dispose();
                }
            });
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void fireEvent(String eventName, Parameter parameter) {
            if (engine != null) {
                if (parameter == null) {
                        engine.getEventManager().handle(new Event(eventName),
                                                StateMachineContextImpl.create());
                } else {
                        engine.getEventManager().handle(new Event(eventName,
                                                new Parameter[]{parameter}),
                                                StateMachineContextImpl.create());
                }
            }
    }

        public static void close() {
                fireEvent(E100, null);
        }

        public static void pauseContinue() {
                fireEvent(E101, null);
        }

        public static void startStop() {
                fireEvent(E102, null);
        }

        public static void inited() {
                fireEvent(E103, null);
        }
}
```

2.1.2. `TraclayerEventProvider.java`

```
package ru.ifmo.tracklayer.ep;

import com.evelopers.common.exception.CommonException;
import com.evelopers.unimod.core.stateworks.Event;
import com.evelopers.unimod.runtime.EventProvider;
import com.evelopers.unimod.runtime.ModelEngine;
import com.evelopers.unimod.runtime.context.Parameter;
import com.evelopers.unimod.runtime.context.StateMachineContextImpl;

public class TracklayerEventProvider implements EventProvider {

        /**
         * @unimod.event.descr Bad track found
         */
    public static final String E200 = "e200";

    /**
         * @unimod.event.descr No track found
         */
    public static final String E201 = "e201";

    /**
```

```
	 * @unimod.event.descr End of line found
	 */
	public static final String E202 = "e202";

	/**
	 * @unimod.event.descr Operation complete
	 */
	public static final String E203 = "e203";

	/**
	 * @unimod.event.descr Store is full
	 */
	public static final String E204 = "e204";

	    private static ModelEngine engine;

	    public void init(ModelEngine engine) throws CommonException {
	            TracklayerEventProvider.engine = engine;
	    }

	    private static void fireEvent(String eventName, Parameter parameter) {
	        if (engine != null) {
	            if (parameter == null) {
	                    engine.getEventManager().handle(new Event(eventName),
	                            StateMachineContextImpl.create());
	            } else {
	                    engine.getEventManager().handle(new Event(eventName,
	                            new Parameter[]{parameter}),
	                            StateMachineContextImpl.create());
	            }
	        }
	}

	    public void dispose() {
	            // TODO Auto-generated method stub

	    }

	    public static void badFound() {
	            fireEvent(E200, null);
	    }

	    public static void noneFound() {
	            fireEvent(E201, null);
	    }

	    public static void endFound() {
	            fireEvent(E202, null);
	    }

	    public static void operationComplete() {
	            fireEvent(E203, null);
	    }

	    public static void storeIsFull() {
	            fireEvent(E204, null);
	    }
}
```

2.2. Объекты управления

2.2.1. `Screen.java`

```
package ru.ifmo.tracklayer.co;

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

import ru.ifmo.tracklayer.gui.GUIFrame;

public class Screen implements ControlledObject {

        /**
        *@unimod.action.descr Set button names for stopped state
        */
        public void z1 (StateMachineContext context)
        {
                GUIFrame.setButtonsStop();
        }

        /**
        *@unimod.action.descr Set button names for running state
        */
        public void z2 (StateMachineContext context)
        {
                GUIFrame.setButtonsRun();
        }

        /**
        *@unimod.action.descr Set button names for paused state
        */
        public void z3 (StateMachineContext context)
        {
                GUIFrame.setButtonsPause();
        }

        /**
         * @unimod.action.descr Redraw screen and wait
         */
        public void z4(StateMachineContext context) {
                GUIFrame.getFrame().repaint();
                try {
                        Thread.sleep(700);
                } catch (InterruptedException e) {

                }
        }
}
```

2.2.2. `Tracklayer.java`

```
package ru.ifmo.tracklayer.co;

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

import ru.ifmo.tracklayer.ep.TracklayerEventProvider;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
```

```
public class Tracklayer implements ControlledObject {
        private final static int NO_TRACK     = 0;
        private final static int GOOD_TRACK   = 1;
        private final static int BAD_TRACK    = 2;
        private final static int END_OF_TRACK = 3;
        private final static int DEPOT        = 4;
        private static int[] railroad;
        private static int position;
        private int storeCapacity;
        private static int tracksInStore;
        private static boolean inited = false;
        private static final int trackSize = 50;

        public Tracklayer() {
                Math.rint(System.currentTimeMillis());
                railroad = new int[30];
                Init();
                inited = true;
        }

        public void Init(){
                for(int i = 0; i < railroad.length; ++i) {
                        railroad[i] = (int) (Math.random() * 3);
                }
                railroad[0] = DEPOT;
                railroad[1] = DEPOT;
                railroad[railroad.length - 1] = END_OF_TRACK;
                position = 1;
                tracksInStore = 5;
                storeCapacity = 5;
        }

        /**
        *@unimod.action.descr Number of tracks in store
        */
        public int x1 (StateMachineContext context)
        {
                return tracksInStore;
        }

        /**
        *@unimod.action.descr Move forward
        */
        public void z1 (StateMachineContext context) {
                if (railroad[position + 1] == GOOD_TRACK) {
                        ++position;
                        TracklayerEventProvider.operationComplete();
                } else {
                        switch(railroad[position + 1]) {
                        case GOOD_TRACK:
                                break;
                        case BAD_TRACK:
                                TracklayerEventProvider.badFound();
                                break;
                        case NO_TRACK:
                                TracklayerEventProvider.noneFound();
                                break;
                        case END_OF_TRACK:
                                TracklayerEventProvider.endFound();
                                break;
                        }
                }
        }

        /**
        *@unimod.action.descr Move backward
```

```
*/
public void z2 (StateMachineContext context) {
        if (position > 1) {
                --position;
                TracklayerEventProvider.operationComplete();
        } else {
                TracklayerEventProvider.endFound();
        }
}

/**
*@unimod.action.descr Put track on
*/
public void z3 (StateMachineContext context)
{
        if (railroad[position + 1] != GOOD_TRACK) {;
                --tracksInStore;
                railroad[position + 1] = GOOD_TRACK;
        }
        TracklayerEventProvider.operationComplete();
}

/**
*@unimod.action.descr Take track off
*/
public void z4 (StateMachineContext context)
{
        railroad[position + 1] = NO_TRACK;
        TracklayerEventProvider.operationComplete();
}

/**
*@unimod.action.descr Load matherials
*/
public void z5 (StateMachineContext context)
{
        if (tracksInStore >= storeCapacity) {
                TracklayerEventProvider.storeIsFull();
        } else {
                ++tracksInStore;
                TracklayerEventProvider.operationComplete();
        };
}

/**
*@unimod.action.descr Init tracklayer
*/
public void z6 (StateMachineContext context)
{
        Init();
        TracklayerEventProvider.operationComplete();
}

private static void drawTrack(int trackType, int x, int y, Graphics g) {
        switch(trackType) {
        case GOOD_TRACK:
                g.setColor(Color.GREEN);
                break;
        case BAD_TRACK:
                g.setColor(Color.RED);
                break;
        case NO_TRACK:
                g.setColor(Color.WHITE);
                break;
        case END_OF_TRACK:
        case DEPOT:
```

```
                g.setColor(Color.BLACK);
                break;
        }
        g.fillRect(x,y,trackSize,trackSize);
        if (trackType == END_OF_TRACK) {
                g.fillRect(x,y - trackSize,trackSize / 3,trackSize);
        }

        if (trackType == DEPOT) {
                g.fillRect(x + trackSize / 3,y - trackSize * 2,
                                trackSize / 3,trackSize * 2);
                g.fillRect(x, y - trackSize * 2 - trackSize / 3,
                                trackSize, trackSize / 3);
        }
}

private static void drawTracklayer(int x, int y, Graphics g) {
        g.setColor(Color.BLUE);
        g.fillRect(x,y,trackSize,trackSize);
        g.fillRect(x - trackSize, y + trackSize / 2,
                        trackSize, trackSize / 2);
        g.setColor(Color.DARK_GRAY);
        for (int i = 0; i < tracksInStore; ++i) {
                g.fillRect(x - trackSize,
                                y + (trackSize / 2) - (trackSize / 8)
                                   - (i * trackSize) / 6,
                                trackSize * 5 / 6,
                                trackSize / 8);
        }
}

public static void drawScreen(Graphics g) {
        if (!inited) return;
        Rectangle bound = g.getClipBounds();
        int shift;
        int startTrack;
        int endTrack;
        if (position * trackSize < bound.width / 2) {
                shift = 0;
                startTrack = 0;
                endTrack = bound.width / trackSize + 1;
        } else {
                shift = -((position * trackSize) - (bound.width / 2));
                startTrack = -((bound.width / 2) / trackSize) + position - 1;
                endTrack = ((bound.width / 2) / trackSize) + position + 1;
        }
        startTrack = Math.max(startTrack, 0);
        endTrack =   Math.min(endTrack, railroad.length - 1);
        drawTracklayer(shift + position * trackSize,
                        bound.height - 2*trackSize, g);
        for (int i = startTrack; i <= endTrack; ++i) {
                drawTrack(railroad[i], shift + i * trackSize,
                                bound.height - trackSize,g);
                if (i % 4 == 0) {
                        g.setColor(Color.BLACK);
                        g.drawString(new Integer(i).toString(),
                         hift + i * trackSize + trackSize / 3,
                                        bound.height - trackSize / 2);
                }
        }
        boolean allCorrect = true;
        for (int i = 0; i < railroad.length; ++i) {
                if (railroad[i] == BAD_TRACK || railroad[i] == NO_TRACK) {
                        allCorrect = false;
                        break;
                }
```

```
                }
                if (allCorrect && position <= 1) {
                        g.drawString("Complete", trackSize * 3, trackSize);
                }
        }
}
```

2.3. Классы, обеспечивающие взаимодействие с пользователем

2.3.1. `GUIFrame.java`

```
package ru.ifmo.tracklayer.gui;

import java.awt.AWTEvent;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;

import javax.swing.*;

import ru.ifmo.tracklayer.co.Tracklayer;
import ru.ifmo.tracklayer.ep.ScreenEventProvider;


public class GUIFrame extends JFrame {

        /**
         *
         */
        private static final long serialVersionUID = 1L;

        private class ViewPanel extends JPanel {
                /**
                 *
                 */
                private static final long serialVersionUID = 1L;

                public void paintComponent(Graphics g) {
                        Tracklayer.drawScreen(g);
                }
        }

        private ViewPanel viewPanel = new ViewPanel();
        private JButton startStop = new JButton();
        private JButton pauseContinue = new JButton();


        private GUIFrame() {
            enableEvents(AWTEvent.WINDOW_EVENT_MASK);
            init();
         }

        private void init() {
            setSize(new Dimension(800, 250));
            setTitle("TrackLayer");

            startStop.setText("Start");
            startStop.addActionListener(
                new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
```

```
                        ScreenEventProvider.startStop();
                    }
                }
                );

            pauseContinue.setText("Pause");
            pauseContinue.addActionListener(
                new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        ScreenEventProvider.pauseContinue();
                    }
                }
                );
            pauseContinue.setEnabled(false);

            JPanel buttonsPanel = new JPanel(new BorderLayout());
            buttonsPanel.add(startStop, BorderLayout.WEST);
            buttonsPanel.add(pauseContinue, BorderLayout.EAST);

            JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
            splitPane.add(buttonsPanel,JSplitPane.TOP);
            splitPane.add(viewPanel,JSplitPane.BOTTOM);
            this.getContentPane().add(splitPane);
        }

        protected void processWindowEvent(WindowEvent e) {
            //super.processWindowEvent(e);
            if (e.getID() == WindowEvent.WINDOW_CLOSING) {
                ScreenEventProvider.close();
            }
        }

        private static GUIFrame frame = new GUIFrame();

        public static GUIFrame getFrame() {
                    return frame;
                  }

        public static void setButtonsStop() {
                getFrame().startStop.setText("Start");
                getFrame().pauseContinue.setEnabled(false);
        }

        public static void setButtonsRun() {
                getFrame().startStop.setText("Stop");
                getFrame().pauseContinue.setEnabled(true);
                getFrame().pauseContinue.setText("Pause");
        }

        public static void setButtonsPause() {
                getFrame().startStop.setText("Stop");
                getFrame().pauseContinue.setEnabled(true);
                getFrame().pauseContinue.setText("Continue");
        }
}
```

2.4. Интерпретационный подход. XML-описание автоматов

2.4.1. A1.xml

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE model PUBLIC "-//eVelopers Corp.//DTD
State machine model V1.0//EN" "http://www.evelopers.com/dtd/unimod/statemachine.dtd">
<model name="Model1">
```

```
<controlledObject class="ru.ifmo.tracklayer.co.Tracklayer" name="o1"/>
<controlledObject class="ru.ifmo.tracklayer.co.Screen" name="o2"/>
<eventProvider class="ru.ifmo.tracklayer.ep.ScreenEventProvider" name="p1">
  <association clientRole="p1" targetRef="A1"/>
</eventProvider>
<eventProvider class="ru.ifmo.tracklayer.ep.TracklayerEventProvider" name="p2">
  <association clientRole="p2" 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="o2" targetRef="o2"/>
  <association clientRole="A1" supplierRole="A2" targetRef="A2"/>
  <state name="Top" type="NORMAL">
    <state name="1. Initialization" type="NORMAL">
      <outputAction ident="o2.z1"/>
      <outputAction ident="o2.z4"/>
    </state>
    <state name="3. Pause" type="NORMAL">
      <outputAction ident="o2.z3"/>
      <outputAction ident="o2.z4"/>
    </state>
    <state name="s1" type="INITIAL"/>
    <state name="s2" type="FINAL"/>
    <state name="2. Work" type="NORMAL">
      <stateMachineRef name="A2"/>
      <outputAction ident="o2.z2"/>
      <outputAction ident="o2.z4"/>
    </state>
  </state>
  <transition event="e100" sourceRef="1. Initialization" targetRef="s2"/>
  <transition event="e102" sourceRef="1. Initialization" targetRef="2. Work"/>
  <transition event="e102" sourceRef="3. Pause" targetRef="1. Initialization"/>
  <transition event="e100" sourceRef="3. Pause" targetRef="s2"/>
  <transition event="e101" sourceRef="3. Pause" targetRef="2. Work"/>
  <transition sourceRef="s1" targetRef="1. Initialization"/>
  <transition event="e102" sourceRef="2. Work" targetRef="1. Initialization"/>
  <transition event="e101" sourceRef="2. Work" targetRef="3. Pause"/>
  <transition event="e100" sourceRef="2. Work" targetRef="s2"/>
</stateMachine>
<stateMachine name="A2">
  <configStore class="com.evelopers.unimod.runtime.config.DistinguishConfigManager"/>
  <association clientRole="A2" supplierRole="o1" targetRef="o1"/>
  <association clientRole="A2" supplierRole="o2" targetRef="o2"/>
  <state name="Top" type="NORMAL">
    <state name="2. Moving forward" type="NORMAL">
      <outputAction ident="o1.z1"/>
    </state>
    <state name="7. Shutting down" type="NORMAL">
      <outputAction ident="o1.z2"/>
    </state>
    <state name="3. Taking off old rails" type="NORMAL">
      <outputAction ident="o1.z4"/>
    </state>
    <state name="s1" type="INITIAL"/>
    <state name="6. Loading" type="NORMAL">
      <outputAction ident="o1.z5"/>
    </state>
    <state name="1. Initialization" type="NORMAL">
      <outputAction ident="o1.z6"/>
    </state>
    <state name="4. Putting new rails" type="NORMAL">
      <outputAction ident="o1.z3"/>
```

```
      </state>
      <state name="5. Moving backward" type="NORMAL">
        <outputAction ident="o1.z2"/>
      </state>
    </state>
    <transition event="e203" sourceRef="2. Moving forward" targetRef="2. Moving forward">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e101" sourceRef="2. Moving forward" targetRef="2. Moving
forward"/>
    <transition event="e202" sourceRef="2. Moving forward" targetRef="7. Shutting down"/>
    <transition event="e200" sourceRef="2. Moving forward" targetRef="3. Taking off old
rails"/>
    <transition event="e102" sourceRef="2. Moving forward" targetRef="1.
Initialization"/>
    <transition event="e201" sourceRef="2. Moving forward" targetRef="4. Putting new
rails"/>
    <transition event="e203" sourceRef="7. Shutting down" targetRef="7. Shutting down">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e101" sourceRef="7. Shutting down" targetRef="7. Shutting down"/>
    <transition event="e102" sourceRef="7. Shutting down" targetRef="1. Initialization"/>
    <transition event="e101" sourceRef="3. Taking off old rails" targetRef="3. Taking off
old rails"/>
    <transition event="e102" sourceRef="3. Taking off old rails" targetRef="1.
Initialization"/>
    <transition event="e203" sourceRef="3. Taking off old rails" targetRef="4. Putting
new rails">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition sourceRef="s1" targetRef="1. Initialization"/>
    <transition event="e204" sourceRef="6. Loading" targetRef="2. Moving forward"/>
    <transition event="e203" sourceRef="6. Loading" targetRef="6. Loading">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e101" sourceRef="6. Loading" targetRef="6. Loading"/>
    <transition event="e102" sourceRef="6. Loading" targetRef="1. Initialization"/>
    <transition event="e203" guard="o1.x1&gt;0" sourceRef="1. Initialization"
targetRef="2. Moving forward">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e101" sourceRef="1. Initialization" targetRef="1.
Initialization"/>
    <transition event="e102" sourceRef="1. Initialization" targetRef="1. Initialization">
      <outputAction ident="o1.z6"/>
    </transition>
    <transition event="e203" guard="o1.x1 &lt;= 0" sourceRef="1. Initialization"
targetRef="5. Moving backward">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e203" guard="o1.x1&gt;0" sourceRef="4. Putting new rails"
targetRef="2. Moving forward">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e102" sourceRef="4. Putting new rails" targetRef="1.
Initialization"/>
    <transition event="e101" sourceRef="4. Putting new rails" targetRef="4. Putting new
rails"/>
    <transition event="e203" guard="o1.x1&lt;=0" sourceRef="4. Putting new rails"
targetRef="5. Moving backward">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e202" sourceRef="5. Moving backward" targetRef="6. Loading"/>
    <transition event="e102" sourceRef="5. Moving backward" targetRef="1.
Initialization"/>
```

```
    <transition event="e203" sourceRef="5. Moving backward" targetRef="5. Moving
backward">
      <outputAction ident="o2.z4"/>
    </transition>
    <transition event="e101" sourceRef="5. Moving backward" targetRef="5. Moving
backward"/>
  </stateMachine>
</model>
```

2.5. Компилятивный подход. Сгенерированный класс логики автоматов и основной класс программы

2.5.1. `TracklayerRunner.java`

```
package ru.ifmo.tracklayer.runner;

import com.evelopers.unimod.runtime.*;
import com.evelopers.unimod.runtime.logger.*;
import com.evelopers.unimod.runtime.context.*;
import org.apache.commons.logging.*;

import ru.ifmo.tracklayer.co.Screen;
import ru.ifmo.tracklayer.co.Tracklayer;
import ru.ifmo.tracklayer.ep.ScreenEventProvider;
import ru.ifmo.tracklayer.ep.TracklayerEventProvider;

import com.evelopers.common.exception.*;

public class TracklayerRunner {

    /**
     * @param args
     */
    public static void main(String[] args) {
    TracklayerRunner runner = new TracklayerRunner();
    runner.run();
    }

    private class ExitListener extends AbstractEventProcessorListener {
    private boolean done;
    public void stateMachineCameToFinalState(StateMachineContext context,
                                                    StateMachinePath path,
                                  StateMachineConfig config) {
        if (path.isRoot()) done = true;
    }
    }

    private void run() {
    ModelEngine me;
    try {
        ControlledObjectsManager b = new ControlledObjectsManager() {
                 Tracklayer o1 = new Tracklayer();
        Screen o2 = new Screen();

        public void init(ModelEngine engine) {
        }

        public void dispose() {
        }

        public ControlledObject getControlledObject(String coName) {
            if (coName.equals("o1")) return o1;
            if (coName.equals("o2")) return o2;
```

```
            return null;
            }
        };
        me = ModelEngine.createStandAlone(new QueuedHandler(),
           new ru.ifmo.tracklayer.runner.Model1EventProcessor(),
                        b,  new EventProvidersManager() {
            ScreenEventProvider p1 = new ScreenEventProvider();
            TracklayerEventProvider p2 = new TracklayerEventProvider();

            public void init(ModelEngine engine) throws CommonException {
                    p1.init(engine);
                    p2.init(engine);
                }

                public void dispose() {
                    p1.dispose();
                    p2.dispose();
                }

                public EventProvider getEventProvider(String epName) {
                    if (epName.equals("ep1")) return p1;
                    if (epName.equals("ep2")) return p2;
                    return null;
                }
            });
            me.getEventProcessor().addEventProcessorListener(
                    new SimpleLogger(LogFactory.getLog(SimpleLogger.class)));
            ExitListener l1 = new ExitListener();
            me.getEventProcessor().addEventProcessorListener(l1);
            me.start();

            // wait for final state
            while (!l1.done) {
                Thread.sleep(10);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

2.5.2. `Model1EventProcessor.java` (генерируется автоматически)

```
package ru.ifmo.tracklayer.runner;

/**
 * This file was generated from model [Model1] on [Fri Oct 13 21:08:24 GMT 2006].
 * Do not change content of this file.
 */

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

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

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

```
public class Model1EventProcessor extends AbstractEventProcessor {

    private ModelStructure modelStructure;

    private static final int A1 = 1;

    private static final int A2 = 2;

    private int decodeStateMachine(String sm) {

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

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

        return -1;
    }

    private A1EventProcessor _A1;

    private A2EventProcessor _A2;

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

        _A1 = new A1EventProcessor();
        _A2 = new A2EventProcessor();
    }

    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.tracklayer.co.Tracklayer o1 = null;

        private ru.ifmo.tracklayer.co.Screen o2 = null;

        private ru.ifmo.tracklayer.ep.ScreenEventProvider p1 = null;

        private ru.ifmo.tracklayer.ep.TracklayerEventProvider p2 = null;

        private ControlledObjectsManager controlledObjectsManager = new
ControlledObjectsManagerImpl();

        private EventProvidersManager eventProvidersManager = new EventProvidersManagerImpl();

        public ControlledObjectsManager getControlledObjectsManager() {
            return controlledObjectsManager;
        }

        public EventProvidersManager getEventProvidersManager() {
            return eventProvidersManager;
        }

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

            public void dispose() {
            }

            public ControlledObject getControlledObject(String coName) {
                if (StringUtils.equals(coName, "o1")) {
                    if (o1 == null) {
                        o1 = new ru.ifmo.tracklayer.co.Tracklayer();
                    }
                    return o1;
                }
                if (StringUtils.equals(coName, "o2")) {
                    if (o2 == null) {
                        o2 = new ru.ifmo.tracklayer.co.Screen();
                    }
                    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("p1");
                ep.init(engine);
                ep = getEventProvider("p2");
                ep.init(engine);
            }

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

```
                ep.dispose();
            }
        }

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

public ModelStructure getModelStructure() {
    return modelStructure;
}

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

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

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

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

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

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

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

    try {
        switch (sm) {
        case A1:
            return _A1.transiteToStableState(context, path, config);
        case A2:
            return _A2.transiteToStableState(context, path, config);
```

```
            default:
                throw new EventProcessorException("Unknown state machine ["
                        + path.getStateMachine() + "]");
            }
        } catch (Exception e) {
            if (e instanceof SystemException) {
                throw (SystemException) e;
            } else {
                throw new SystemException(e);
            }
        }
    }

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

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

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

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

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

            switch (sm) {
            case A1:
                return new StateMachineConfig("Top");
            case A2:
                return new StateMachineConfig("Top");
            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.s2:
                    return true;
                default:
                    return false;
                }
            case A2:
                state = _A2.decodeState(config.getActiveState());
                switch (state) {
                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 _1__Initialization = 2;

    private static final int _3__Pause = 3;

    private static final int s1 = 4;

    private static final int s2 = 5;

    private static final int _2__Work = 6;

    private int decodeState(String state) {

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

        if ("1. Initialization".equals(state)) {
            return _1__Initialization;
        } else

        if ("3. Pause".equals(state)) {
            return _3__Pause;
        } else

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

        if ("s2".equals(state)) {
            return s2;
        } else

        if ("2. Work".equals(state)) {
            return _2__Work;
        }

        return -1;
    }

    // events
    private static final int e101 = 1;

    private static final int e100 = 2;

    private static final int e102 = 3;

    private int decodeEvent(String event) {

        if ("e101".equals(event)) {
            return e101;
        } else

        if ("e100".equals(event)) {
            return e100;
        } else

        if ("e102".equals(event)) {
            return e102;
        }

        return -1;
    }

    private ru.ifmo.tracklayer.co.Tracklayer o1;
```

```
private ru.ifmo.tracklayer.co.Screen o2;

private void init(ControlledObjectsMap controlledObjectsMap) {
    o1 = (ru.ifmo.tracklayer.co.Tracklayer) controlledObjectsMap
            .getControlledObject("o1");
    o2 = (ru.ifmo.tracklayer.co.Screen) 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 _1__Initialization:

            return;
        case _3__Pause:

            return;
        case s1:

            return;
        case s2:

            return;
        case _2__Work:
            // 2. Work includes A2

            fireBeforeSubmachineExecution(context, event, path,
                    "2. Work", "A2");

            Model1EventProcessor.this.process(event, context,
                    new StateMachinePath(path, "2. Work", "A2"));

            fireAfterSubmachineExecution(context, event, path,
                    "2. Work", "A2");

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

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

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

    switch (s) {
    case Top:

        fireComeToState(context, path, "s1");
```

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

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

            // 1. Initialization [o2.z1, o2.z4]
            fireBeforeOutputActionExecution(context, path,
                    "s1#1. Initialization##true", "o2.z1");

            o2.z1(context);

            fireAfterOutputActionExecution(context, path,
                    "s1#1. Initialization##true", "o2.z1");
            fireBeforeOutputActionExecution(context, path,
                    "s1#1. Initialization##true", "o2.z4");

            o2.z4(context);

            fireAfterOutputActionExecution(context, path,
                    "s1#1. Initialization##true", "o2.z4");

            return new StateMachineConfig("1. Initialization");
        }

        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 _1__Initialization:

                switch (e) {
                case e100:

                    // 1. Initialization->s2 e100[true]/

                    fireTransitionCandidate(context, path,
                            "1. Initialization", event,
                            "1. Initialization#s2#e100#true");

                    fireTransitionFound(context, path, "1. Initialization",
                            event, "1. Initialization#s2#e100#true");

                    fireComeToState(context, path, "s2");

                    // s2 []
                    return new StateMachineConfig("s2");

                case e102:

                    // 1. Initialization->2. Work e102[true]/

                    fireTransitionCandidate(context, path,
                            "1. Initialization", event,
                            "1. Initialization#2. Work#e102#true");

                    fireTransitionFound(context, path, "1. Initialization",
                            event, "1. Initialization#2. Work#e102#true");

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

                    // 2. Work [o2.z2, o2.z4]
                    fireBeforeOutputActionExecution(context, path,
```

```
                "1. Initialization#2. Work#e102#true", "o2.z2");

            o2.z2(context);

            fireAfterOutputActionExecution(context, path,
                    "1. Initialization#2. Work#e102#true", "o2.z2");
            fireBeforeOutputActionExecution(context, path,
                    "1. Initialization#2. Work#e102#true", "o2.z4");

            o2.z4(context);

            fireAfterOutputActionExecution(context, path,
                    "1. Initialization#2. Work#e102#true", "o2.z4");
            return new StateMachineConfig("2. Work");

        default:

            // transition not found
            return config;
        }

    case _3__Pause:

        switch (e) {
        case e101:

            // 3. Pause->2. Work e101[true]/

            fireTransitionCandidate(context, path, "3. Pause",
                    event, "3. Pause#2. Work#e101#true");

            fireTransitionFound(context, path, "3. Pause", event,
                    "3. Pause#2. Work#e101#true");

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

            // 2. Work [o2.z2, o2.z4]
            fireBeforeOutputActionExecution(context, path,
                    "3. Pause#2. Work#e101#true", "o2.z2");

            o2.z2(context);

            fireAfterOutputActionExecution(context, path,
                    "3. Pause#2. Work#e101#true", "o2.z2");
            fireBeforeOutputActionExecution(context, path,
                    "3. Pause#2. Work#e101#true", "o2.z4");

            o2.z4(context);

            fireAfterOutputActionExecution(context, path,
                    "3. Pause#2. Work#e101#true", "o2.z4");
            return new StateMachineConfig("2. Work");

        case e100:

            // 3. Pause->s2 e100[true]/

            fireTransitionCandidate(context, path, "3. Pause",
                    event, "3. Pause#s2#e100#true");

            fireTransitionFound(context, path, "3. Pause", event,
                    "3. Pause#s2#e100#true");

            fireComeToState(context, path, "s2");

            // s2 []
            return new StateMachineConfig("s2");

        case e102:

            // 3. Pause->1. Initialization e102[true]/

            fireTransitionCandidate(context, path, "3. Pause",
                    event, "3. Pause#1. Initialization#e102#true");
```

```
            fireTransitionFound(context, path, "3. Pause", event,
                    "3. Pause#1. Initialization#e102#true");

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

            // 1. Initialization [o2.z1, o2.z4]
            fireBeforeOutputActionExecution(context, path,
                    "3. Pause#1. Initialization#e102#true", "o2.z1");

            o2.z1(context);

            fireAfterOutputActionExecution(context, path,
                    "3. Pause#1. Initialization#e102#true", "o2.z1");
            fireBeforeOutputActionExecution(context, path,
                    "3. Pause#1. Initialization#e102#true", "o2.z4");

            o2.z4(context);

            fireAfterOutputActionExecution(context, path,
                    "3. Pause#1. Initialization#e102#true", "o2.z4");
            return new StateMachineConfig("1. Initialization");

        default:

            // transition not found
            return config;
        }

    case _2__Work:

        switch (e) {
        case e101:

            // 2. Work->3. Pause e101[true]/

            fireTransitionCandidate(context, path, "2. Work",
                    event, "2. Work#3. Pause#e101#true");

            fireTransitionFound(context, path, "2. Work", event,
                    "2. Work#3. Pause#e101#true");

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

            // 3. Pause [o2.z3, o2.z4]
            fireBeforeOutputActionExecution(context, path,
                    "2. Work#3. Pause#e101#true", "o2.z3");

            o2.z3(context);

            fireAfterOutputActionExecution(context, path,
                    "2. Work#3. Pause#e101#true", "o2.z3");
            fireBeforeOutputActionExecution(context, path,
                    "2. Work#3. Pause#e101#true", "o2.z4");

            o2.z4(context);

            fireAfterOutputActionExecution(context, path,
                    "2. Work#3. Pause#e101#true", "o2.z4");
            return new StateMachineConfig("3. Pause");

        case e100:

            // 2. Work->s2 e100[true]/

            fireTransitionCandidate(context, path, "2. Work",
                    event, "2. Work#s2#e100#true");

            fireTransitionFound(context, path, "2. Work", event,
                    "2. Work#s2#e100#true");

            fireComeToState(context, path, "s2");

            // s2 []
```

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

                case e102:

                    // 2. Work->1. Initialization e102[true]/

                    fireTransitionCandidate(context, path, "2. Work",
                            event, "2. Work#1. Initialization#e102#true");

                    fireTransitionFound(context, path, "2. Work", event,
                            "2. Work#1. Initialization#e102#true");

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

                    // 1. Initialization [o2.z1, o2.z4]
                    fireBeforeOutputActionExecution(context, path,
                            "2. Work#1. Initialization#e102#true", "o2.z1");

                    o2.z1(context);

                    fireAfterOutputActionExecution(context, path,
                            "2. Work#1. Initialization#e102#true", "o2.z1");
                    fireBeforeOutputActionExecution(context, path,
                            "2. Work#1. Initialization#e102#true", "o2.z4");

                    o2.z4(context);

                    fireAfterOutputActionExecution(context, path,
                            "2. Work#1. Initialization#e102#true", "o2.z4");
                    return new StateMachineConfig("1. Initialization");

                default:

                    // transition not found
                    return config;
                }

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

}

private class A2EventProcessor {

    // states
    private static final int Top = 1;

    private static final int _2__Moving_forward = 2;

    private static final int _7__Shutting_down = 3;

    private static final int _3__Taking_off_old_rails = 4;

    private static final int s1 = 5;

    private static final int _6__Loading = 6;

    private static final int _1__Initialization = 7;

    private static final int _4__Putting_new_rails = 8;

    private static final int _5__Moving_backward = 9;

    private int decodeState(String state) {

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

```
    if ("2. Moving forward".equals(state)) {
        return _2__Moving_forward;
    } else

    if ("7. Shutting down".equals(state)) {
        return _7__Shutting_down;
    } else

    if ("3. Taking off old rails".equals(state)) {
        return _3__Taking_off_old_rails;
    } else

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

    if ("6. Loading".equals(state)) {
        return _6__Loading;
    } else

    if ("1. Initialization".equals(state)) {
        return _1__Initialization;
    } else

    if ("4. Putting new rails".equals(state)) {
        return _4__Putting_new_rails;
    } else

    if ("5. Moving backward".equals(state)) {
        return _5__Moving_backward;
    }

    return -1;
}

// events
private static final int e203 = 1;

private static final int e201 = 2;

private static final int e101 = 3;

private static final int e204 = 4;

private static final int e202 = 5;

private static final int e200 = 6;

private static final int e102 = 7;

private int decodeEvent(String event) {

    if ("e203".equals(event)) {
        return e203;
    } else

    if ("e201".equals(event)) {
        return e201;
    } else

    if ("e101".equals(event)) {
        return e101;
    } else

    if ("e204".equals(event)) {
        return e204;
    } else

    if ("e202".equals(event)) {
        return e202;
    } else

    if ("e200".equals(event)) {
```

```
            return e200;
        } else

        if ("e102".equals(event)) {
            return e102;
        }

        return -1;
    }

    private ru.ifmo.tracklayer.co.Tracklayer o1;

    private ru.ifmo.tracklayer.co.Screen o2;

    private void init(ControlledObjectsMap controlledObjectsMap) {
        o1 = (ru.ifmo.tracklayer.co.Tracklayer) controlledObjectsMap
                .getControlledObject("o1");
        o2 = (ru.ifmo.tracklayer.co.Screen) 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 _2__Moving_forward:

                return;
            case _7__Shutting_down:

                return;
            case _3__Taking_off_old_rails:

                return;
            case s1:

                return;
            case _6__Loading:

                return;
            case _1__Initialization:

                return;
            case _4__Putting_new_rails:

                return;
            case _5__Moving_backward:

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

    private StateMachineConfig transiteToStableState(
```

```
            StateMachineContext context, StateMachinePath path,
            StateMachineConfig config) throws Exception {

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

        switch (s) {
        case Top:

            fireComeToState(context, path, "s1");

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

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

            // 1. Initialization [o1.z6]
            fireBeforeOutputActionExecution(context, path,
                    "s1#1. Initialization##true", "o1.z6");

            o1.z6(context);

            fireAfterOutputActionExecution(context, path,
                    "s1#1. Initialization##true", "o1.z6");

            return new StateMachineConfig("1. Initialization");
        }

        return config;
    }

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

        int

        o1_x1 = 0;

        BitSet calculatedInputActions = new BitSet(1);

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

        while (true) {
            switch (s) {
            case _2__Moving_forward:

                switch (e) {
                case e203:

                    // 2. Moving forward->2. Moving forward e203[true]/o2.z4

                    fireTransitionCandidate(context, path,
                            "2. Moving forward", event,
                            "2. Moving forward#2. Moving forward#e203#true");

                    fireTransitionFound(context, path, "2. Moving forward",
                            event,
                            "2. Moving forward#2. Moving forward#e203#true");

                    fireBeforeOutputActionExecution(
                            context,
                            path,
                            "2. Moving forward#2. Moving forward#e203#true",
                            "o2.z4");

                    o2.z4(context);

                    fireAfterOutputActionExecution(
                            context,
                            path,
```

```
                "2. Moving forward#2. Moving forward#e203#true",
                "o2.z4");

        fireComeToState(context, path, "2. Moving forward");

        // 2. Moving forward [o1.z1]
        fireBeforeOutputActionExecution(
                context,
                path,
                "2. Moving forward#2. Moving forward#e203#true",
                "o1.z1");

        o1.z1(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "2. Moving forward#2. Moving forward#e203#true",
                "o1.z1");
        return new StateMachineConfig("2. Moving forward");

    case e201:

        // 2. Moving forward->4. Putting new rails e201[true]/

        fireTransitionCandidate(context, path,
                "2. Moving forward", event,
                "2. Moving forward#4. Putting new rails#e201#true");

        fireTransitionFound(context, path, "2. Moving forward",
                event,
                "2. Moving forward#4. Putting new rails#e201#true");

        fireComeToState(context, path, "4. Putting new rails");

        // 4. Putting new rails [o1.z3]
        fireBeforeOutputActionExecution(
                context,
                path,
                "2. Moving forward#4. Putting new rails#e201#true",
                "o1.z3");

        o1.z3(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "2. Moving forward#4. Putting new rails#e201#true",
                "o1.z3");
        return new StateMachineConfig("4. Putting new rails");

    case e101:

        // 2. Moving forward->2. Moving forward e101[true]/

        fireTransitionCandidate(context, path,
                "2. Moving forward", event,
                "2. Moving forward#2. Moving forward#e101#true");

        fireTransitionFound(context, path, "2. Moving forward",
                event,
                "2. Moving forward#2. Moving forward#e101#true");

        fireComeToState(context, path, "2. Moving forward");

        // 2. Moving forward [o1.z1]
        fireBeforeOutputActionExecution(
                context,
                path,
                "2. Moving forward#2. Moving forward#e101#true",
                "o1.z1");

        o1.z1(context);
```

```
        fireAfterOutputActionExecution(
                context,
                path,
                "2. Moving forward#2. Moving forward#e101#true",
                "o1.z1");
        return new StateMachineConfig("2. Moving forward");

    case e202:

        // 2. Moving forward->7. Shutting down e202[true]/

        fireTransitionCandidate(context, path,
                "2. Moving forward", event,
                "2. Moving forward#7. Shutting down#e202#true");

        fireTransitionFound(context, path, "2. Moving forward",
                event,
                "2. Moving forward#7. Shutting down#e202#true");

        fireComeToState(context, path, "7. Shutting down");

        // 7. Shutting down [o1.z2]
        fireBeforeOutputActionExecution(context, path,
                "2. Moving forward#7. Shutting down#e202#true",
                "o1.z2");

        o1.z2(context);

        fireAfterOutputActionExecution(context, path,
                "2. Moving forward#7. Shutting down#e202#true",
                "o1.z2");
        return new StateMachineConfig("7. Shutting down");

    case e200:

        // 2. Moving forward->3. Taking off old rails e200[true]/

        fireTransitionCandidate(context, path,
                "2. Moving forward", event,
                "2. Moving forward#3. Taking off old rails#e200#true");

        fireTransitionFound(context, path, "2. Moving forward",
                event,
                "2. Moving forward#3. Taking off old rails#e200#true");

        fireComeToState(context, path,
                "3. Taking off old rails");

        // 3. Taking off old rails [o1.z4]
        fireBeforeOutputActionExecution(
                context,
                path,
                "2. Moving forward#3. Taking off old rails#e200#true",
                "o1.z4");

        o1.z4(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "2. Moving forward#3. Taking off old rails#e200#true",
                "o1.z4");
        return new StateMachineConfig("3. Taking off old rails");

    case e102:

        // 2. Moving forward->1. Initialization e102[true]/

        fireTransitionCandidate(context, path,
                "2. Moving forward", event,
                "2. Moving forward#1. Initialization#e102#true");

        fireTransitionFound(context, path, "2. Moving forward",
                event,
```

```
                "2. Moving forward#1. Initialization#e102#true");

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

        // 1. Initialization [o1.z6]
        fireBeforeOutputActionExecution(
                context,
                path,
                "2. Moving forward#1. Initialization#e102#true",
                "o1.z6");

        o1.z6(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "2. Moving forward#1. Initialization#e102#true",
                "o1.z6");
        return new StateMachineConfig("1. Initialization");

    default:

        // transition not found
        return config;
    }

case _7__Shutting_down:

    switch (e) {
    case e203:

        // 7. Shutting down->7. Shutting down e203[true]/o2.z4

        fireTransitionCandidate(context, path,
                "7. Shutting down", event,
                "7. Shutting down#7. Shutting down#e203#true");

        fireTransitionFound(context, path, "7. Shutting down",
                event,
                "7. Shutting down#7. Shutting down#e203#true");

        fireBeforeOutputActionExecution(context, path,
                "7. Shutting down#7. Shutting down#e203#true",
                "o2.z4");

        o2.z4(context);

        fireAfterOutputActionExecution(context, path,
                "7. Shutting down#7. Shutting down#e203#true",
                "o2.z4");

        fireComeToState(context, path, "7. Shutting down");

        // 7. Shutting down [o1.z2]
        fireBeforeOutputActionExecution(context, path,
                "7. Shutting down#7. Shutting down#e203#true",
                "o1.z2");

        o1.z2(context);

        fireAfterOutputActionExecution(context, path,
                "7. Shutting down#7. Shutting down#e203#true",
                "o1.z2");
        return new StateMachineConfig("7. Shutting down");

    case e101:

        // 7. Shutting down->7. Shutting down e101[true]/

        fireTransitionCandidate(context, path,
                "7. Shutting down", event,
                "7. Shutting down#7. Shutting down#e101#true");

        fireTransitionFound(context, path, "7. Shutting down",
```

```
                event,
                "7. Shutting down#7. Shutting down#e101#true");

        fireComeToState(context, path, "7. Shutting down");

        // 7. Shutting down [o1.z2]
        fireBeforeOutputActionExecution(context, path,
                "7. Shutting down#7. Shutting down#e101#true",
                "o1.z2");

        o1.z2(context);

        fireAfterOutputActionExecution(context, path,
                "7. Shutting down#7. Shutting down#e101#true",
                "o1.z2");
        return new StateMachineConfig("7. Shutting down");

    case e102:

        // 7. Shutting down->1. Initialization e102[true]/

        fireTransitionCandidate(context, path,
                "7. Shutting down", event,
                "7. Shutting down#1. Initialization#e102#true");

        fireTransitionFound(context, path, "7. Shutting down",
                event,
                "7. Shutting down#1. Initialization#e102#true");

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

        // 1. Initialization [o1.z6]
        fireBeforeOutputActionExecution(context, path,
                "7. Shutting down#1. Initialization#e102#true",
                "o1.z6");

        o1.z6(context);

        fireAfterOutputActionExecution(context, path,
                "7. Shutting down#1. Initialization#e102#true",
                "o1.z6");
        return new StateMachineConfig("1. Initialization");

    default:

        // transition not found
        return config;
    }

case _3__Taking_off_old_rails:

    switch (e) {
    case e203:

        // 3. Taking off old rails->4. Putting new rails e203[true]/o2.z4

        fireTransitionCandidate(context, path,
                "3. Taking off old rails", event,
                "3. Taking off old rails#4. Putting new rails#e203#true");

        fireTransitionFound(context, path,
                "3. Taking off old rails", event,
                "3. Taking off old rails#4. Putting new rails#e203#true");

        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Taking off old rails#4. Putting new rails#e203#true",
                "o2.z4");

        o2.z4(context);

        fireAfterOutputActionExecution(
                context,
```

```
                path,
                "3. Taking off old rails#4. Putting new rails#e203#true",
                "o2.z4");

        fireComeToState(context, path, "4. Putting new rails");

        // 4. Putting new rails [o1.z3]
        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Taking off old rails#4. Putting new rails#e203#true",
                "o1.z3");

        o1.z3(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "3. Taking off old rails#4. Putting new rails#e203#true",
                "o1.z3");
        return new StateMachineConfig("4. Putting new rails");

    case e101:

        // 3. Taking off old rails->3. Taking off old rails e101[true]/

        fireTransitionCandidate(context, path,
                "3. Taking off old rails", event,
                "3. Taking off old rails#3. Taking off old rails#e101#true");

        fireTransitionFound(context, path,
                "3. Taking off old rails", event,
                "3. Taking off old rails#3. Taking off old rails#e101#true");

        fireComeToState(context, path,
                "3. Taking off old rails");

        // 3. Taking off old rails [o1.z4]
        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Taking off old rails#3. Taking off old rails#e101#true",
                "o1.z4");

        o1.z4(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "3. Taking off old rails#3. Taking off old rails#e101#true",
                "o1.z4");
        return new StateMachineConfig("3. Taking off old rails");

    case e102:

        // 3. Taking off old rails->1. Initialization e102[true]/

        fireTransitionCandidate(context, path,
                "3. Taking off old rails", event,
                "3. Taking off old rails#1. Initialization#e102#true");

        fireTransitionFound(context, path,
                "3. Taking off old rails", event,
                "3. Taking off old rails#1. Initialization#e102#true");

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

        // 1. Initialization [o1.z6]
        fireBeforeOutputActionExecution(
                context,
                path,
                "3. Taking off old rails#1. Initialization#e102#true",
                "o1.z6");
```

```
            o1.z6(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "3. Taking off old rails#1. Initialization#e102#true",
                    "o1.z6");
            return new StateMachineConfig("1. Initialization");

        default:

            // transition not found
            return config;
        }

    case _6__Loading:

        switch (e) {
        case e203:

            // 6. Loading->6. Loading e203[true]/o2.z4

            fireTransitionCandidate(context, path, "6. Loading",
                    event, "6. Loading#6. Loading#e203#true");

            fireTransitionFound(context, path, "6. Loading", event,
                    "6. Loading#6. Loading#e203#true");

            fireBeforeOutputActionExecution(context, path,
                    "6. Loading#6. Loading#e203#true", "o2.z4");

            o2.z4(context);

            fireAfterOutputActionExecution(context, path,
                    "6. Loading#6. Loading#e203#true", "o2.z4");

            fireComeToState(context, path, "6. Loading");

            // 6. Loading [o1.z5]
            fireBeforeOutputActionExecution(context, path,
                    "6. Loading#6. Loading#e203#true", "o1.z5");

            o1.z5(context);

            fireAfterOutputActionExecution(context, path,
                    "6. Loading#6. Loading#e203#true", "o1.z5");
            return new StateMachineConfig("6. Loading");

        case e101:

            // 6. Loading->6. Loading e101[true]/

            fireTransitionCandidate(context, path, "6. Loading",
                    event, "6. Loading#6. Loading#e101#true");

            fireTransitionFound(context, path, "6. Loading", event,
                    "6. Loading#6. Loading#e101#true");

            fireComeToState(context, path, "6. Loading");

            // 6. Loading [o1.z5]
            fireBeforeOutputActionExecution(context, path,
                    "6. Loading#6. Loading#e101#true", "o1.z5");

            o1.z5(context);

            fireAfterOutputActionExecution(context, path,
                    "6. Loading#6. Loading#e101#true", "o1.z5");
            return new StateMachineConfig("6. Loading");

        case e204:

            // 6. Loading->2. Moving forward e204[true]/
```

```
            fireTransitionCandidate(context, path, "6. Loading",
                    event, "6. Loading#2. Moving forward#e204#true");

            fireTransitionFound(context, path, "6. Loading", event,
                    "6. Loading#2. Moving forward#e204#true");

            fireComeToState(context, path, "2. Moving forward");

            // 2. Moving forward [o1.z1]
            fireBeforeOutputActionExecution(context, path,
                    "6. Loading#2. Moving forward#e204#true",
                    "o1.z1");

            o1.z1(context);

            fireAfterOutputActionExecution(context, path,
                    "6. Loading#2. Moving forward#e204#true",
                    "o1.z1");
            return new StateMachineConfig("2. Moving forward");

        case e102:

            // 6. Loading->1. Initialization e102[true]/

            fireTransitionCandidate(context, path, "6. Loading",
                    event, "6. Loading#1. Initialization#e102#true");

            fireTransitionFound(context, path, "6. Loading", event,
                    "6. Loading#1. Initialization#e102#true");

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

            // 1. Initialization [o1.z6]
            fireBeforeOutputActionExecution(context, path,
                    "6. Loading#1. Initialization#e102#true",
                    "o1.z6");

            o1.z6(context);

            fireAfterOutputActionExecution(context, path,
                    "6. Loading#1. Initialization#e102#true",
                    "o1.z6");
            return new StateMachineConfig("1. Initialization");

        default:

            // transition not found
            return config;
        }

    case _1__Initialization:

        switch (e) {
        case e203:

            // 1. Initialization->2. Moving forward e203[o1.x1>0]/o2.z4

            fireTransitionCandidate(context, path,
                    "1. Initialization", event,
                    "1. Initialization#2. Moving forward#e203#o1.x1>0");

            if (!isInputActionCalculated(calculatedInputActions,
                    _o1_x1)) {

                fireBeforeInputActionExecution(
                        context,
                        path,
                        "1. Initialization#2. Moving forward#e203#o1.x1>0",
                        "o1.x1");

                o1_x1 = o1.x1(context);

                fireAfterInputActionExecution(
                        context,
```

```
                path,
                "1. Initialization#2. Moving forward#e203#o1.x1>0",
                "o1.x1", new Integer(o1_x1));
    }

    if (o1_x1 > 0) {

        fireTransitionFound(context, path,
                "1. Initialization", event,
                "1. Initialization#2. Moving forward#e203#o1.x1>0");

        fireBeforeOutputActionExecution(
                context,
                path,
                "1. Initialization#2. Moving forward#e203#o1.x1>0",
                "o2.z4");

        o2.z4(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "1. Initialization#2. Moving forward#e203#o1.x1>0",
                "o2.z4");

        fireComeToState(context, path, "2. Moving forward");

        // 2. Moving forward [o1.z1]
        fireBeforeOutputActionExecution(
                context,
                path,
                "1. Initialization#2. Moving forward#e203#o1.x1>0",
                "o1.z1");

        o1.z1(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "1. Initialization#2. Moving forward#e203#o1.x1>0",
                "o1.z1");
        return new StateMachineConfig("2. Moving forward");

    }
    // 1. Initialization->5. Moving backward e203[o1.x1 <= 0]/o2.z4

    fireTransitionCandidate(context, path,
            "1. Initialization", event,
            "1. Initialization#5. Moving backward#e203#o1.x1 <= 0");

    if (o1_x1 <= 0) {

        fireTransitionFound(context, path,
                "1. Initialization", event,
                "1. Initialization#5. Moving backward#e203#o1.x1 <= 0");

        fireBeforeOutputActionExecution(
                context,
                path,
                "1. Initialization#5. Moving backward#e203#o1.x1 <= 0",
                "o2.z4");

        o2.z4(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "1. Initialization#5. Moving backward#e203#o1.x1 <= 0",
                "o2.z4");

        fireComeToState(context, path, "5. Moving backward");

        // 5. Moving backward [o1.z2]
        fireBeforeOutputActionExecution(
```

```
                context,
                path,
                "1. Initialization#5. Moving backward#e203#o1.x1 <= 0",
                "o1.z2");

        o1.z2(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "1. Initialization#5. Moving backward#e203#o1.x1 <= 0",
                "o1.z2");
        return new StateMachineConfig("5. Moving backward");

    }

    // transition not found
    return config;
case e101:

    // 1. Initialization->1. Initialization e101[true]/

    fireTransitionCandidate(context, path,
            "1. Initialization", event,
            "1. Initialization#1. Initialization#e101#true");

    fireTransitionFound(context, path, "1. Initialization",
            event,
            "1. Initialization#1. Initialization#e101#true");

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

    // 1. Initialization [o1.z6]
    fireBeforeOutputActionExecution(
            context,
            path,
            "1. Initialization#1. Initialization#e101#true",
            "o1.z6");

    o1.z6(context);

    fireAfterOutputActionExecution(
            context,
            path,
            "1. Initialization#1. Initialization#e101#true",
            "o1.z6");
    return new StateMachineConfig("1. Initialization");

case e102:

    // 1. Initialization->1. Initialization e102[true]/o1.z6

    fireTransitionCandidate(context, path,
            "1. Initialization", event,
            "1. Initialization#1. Initialization#e102#true");

    fireTransitionFound(context, path, "1. Initialization",
            event,
            "1. Initialization#1. Initialization#e102#true");

    fireBeforeOutputActionExecution(
            context,
            path,
            "1. Initialization#1. Initialization#e102#true",
            "o1.z6");

    o1.z6(context);

    fireAfterOutputActionExecution(
            context,
            path,
            "1. Initialization#1. Initialization#e102#true",
            "o1.z6");
```

```
            fireComeToState(context, path, "1. Initialization");

            // 1. Initialization [o1.z6]
            fireBeforeOutputActionExecution(
                    context,
                    path,
                    "1. Initialization#1. Initialization#e102#true",
                    "o1.z6");

            o1.z6(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "1. Initialization#1. Initialization#e102#true",
                    "o1.z6");
            return new StateMachineConfig("1. Initialization");

        default:

            // transition not found
            return config;
        }

    case _4__Putting_new_rails:

        switch (e) {
        case e203:

            // 4. Putting new rails->2. Moving forward e203[o1.x1>0]/o2.z4

            fireTransitionCandidate(context, path,
                    "4. Putting new rails", event,
                    "4. Putting new rails#2. Moving forward#e203#o1.x1>0");

            if (!isInputActionCalculated(calculatedInputActions,
                    _o1_x1)) {

                fireBeforeInputActionExecution(
                        context,
                        path,
                        "4. Putting new rails#2. Moving forward#e203#o1.x1>0",
                        "o1.x1");

                o1_x1 = o1.x1(context);

                fireAfterInputActionExecution(
                        context,
                        path,
                        "4. Putting new rails#2. Moving forward#e203#o1.x1>0",
                        "o1.x1", new Integer(o1_x1));
            }

            if (o1_x1 > 0) {

                fireTransitionFound(context, path,
                        "4. Putting new rails", event,
                        "4. Putting new rails#2. Moving forward#e203#o1.x1>0");

                fireBeforeOutputActionExecution(
                        context,
                        path,
                        "4. Putting new rails#2. Moving forward#e203#o1.x1>0",
                        "o2.z4");

                o2.z4(context);

                fireAfterOutputActionExecution(
                        context,
                        path,
                        "4. Putting new rails#2. Moving forward#e203#o1.x1>0",
                        "o2.z4");

                fireComeToState(context, path, "2. Moving forward");
```

```
            // 2. Moving forward [o1.z1]
            fireBeforeOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#2. Moving forward#e203#o1.x1>0",
                    "o1.z1");

            o1.z1(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#2. Moving forward#e203#o1.x1>0",
                    "o1.z1");
            return new StateMachineConfig("2. Moving forward");

        }
        // 4. Putting new rails->5. Moving backward e203[o1.x1<=0]/o2.z4

        fireTransitionCandidate(context, path,
                "4. Putting new rails", event,
                "4. Putting new rails#5. Moving backward#e203#o1.x1<=0");

        if (o1_x1 <= 0) {

            fireTransitionFound(context, path,
                    "4. Putting new rails", event,
                    "4. Putting new rails#5. Moving backward#e203#o1.x1<=0");

            fireBeforeOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#5. Moving backward#e203#o1.x1<=0",
                    "o2.z4");

            o2.z4(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#5. Moving backward#e203#o1.x1<=0",
                    "o2.z4");

            fireComeToState(context, path, "5. Moving backward");

            // 5. Moving backward [o1.z2]
            fireBeforeOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#5. Moving backward#e203#o1.x1<=0",
                    "o1.z2");

            o1.z2(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#5. Moving backward#e203#o1.x1<=0",
                    "o1.z2");
            return new StateMachineConfig("5. Moving backward");

        }

        // transition not found
        return config;
    case e101:

        // 4. Putting new rails->4. Putting new rails e101[true]/

        fireTransitionCandidate(context, path,
                "4. Putting new rails", event,
                "4. Putting new rails#4. Putting new rails#e101#true");
```

```
            fireTransitionFound(context, path,
                    "4. Putting new rails", event,
                    "4. Putting new rails#4. Putting new rails#e101#true");

            fireComeToState(context, path, "4. Putting new rails");

            // 4. Putting new rails [o1.z3]
            fireBeforeOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#4. Putting new rails#e101#true",
                    "o1.z3");

            o1.z3(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#4. Putting new rails#e101#true",
                    "o1.z3");
            return new StateMachineConfig("4. Putting new rails");

        case e102:

            // 4. Putting new rails->1. Initialization e102[true]/

            fireTransitionCandidate(context, path,
                    "4. Putting new rails", event,
                    "4. Putting new rails#1. Initialization#e102#true");

            fireTransitionFound(context, path,
                    "4. Putting new rails", event,
                    "4. Putting new rails#1. Initialization#e102#true");

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

            // 1. Initialization [o1.z6]
            fireBeforeOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#1. Initialization#e102#true",
                    "o1.z6");

            o1.z6(context);

            fireAfterOutputActionExecution(
                    context,
                    path,
                    "4. Putting new rails#1. Initialization#e102#true",
                    "o1.z6");
            return new StateMachineConfig("1. Initialization");

        default:

            // transition not found
            return config;
        }

    case _5__Moving_backward:

        switch (e) {
        case e203:

            // 5. Moving backward->5. Moving backward e203[true]/o2.z4

            fireTransitionCandidate(context, path,
                    "5. Moving backward", event,
                    "5. Moving backward#5. Moving backward#e203#true");

            fireTransitionFound(context, path,
                    "5. Moving backward", event,
                    "5. Moving backward#5. Moving backward#e203#true");

            fireBeforeOutputActionExecution(
```

```
                context,
                path,
                "5. Moving backward#5. Moving backward#e203#true",
                "o2.z4");

        o2.z4(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "5. Moving backward#5. Moving backward#e203#true",
                "o2.z4");

        fireComeToState(context, path, "5. Moving backward");

        // 5. Moving backward [o1.z2]
        fireBeforeOutputActionExecution(
                context,
                path,
                "5. Moving backward#5. Moving backward#e203#true",
                "o1.z2");

        o1.z2(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "5. Moving backward#5. Moving backward#e203#true",
                "o1.z2");
        return new StateMachineConfig("5. Moving backward");

    case e101:

        // 5. Moving backward->5. Moving backward e101[true]/

        fireTransitionCandidate(context, path,
                "5. Moving backward", event,
                "5. Moving backward#5. Moving backward#e101#true");

        fireTransitionFound(context, path,
                "5. Moving backward", event,
                "5. Moving backward#5. Moving backward#e101#true");

        fireComeToState(context, path, "5. Moving backward");

        // 5. Moving backward [o1.z2]
        fireBeforeOutputActionExecution(
                context,
                path,
                "5. Moving backward#5. Moving backward#e101#true",
                "o1.z2");

        o1.z2(context);

        fireAfterOutputActionExecution(
                context,
                path,
                "5. Moving backward#5. Moving backward#e101#true",
                "o1.z2");
        return new StateMachineConfig("5. Moving backward");

    case e202:

        // 5. Moving backward->6. Loading e202[true]/

        fireTransitionCandidate(context, path,
                "5. Moving backward", event,
                "5. Moving backward#6. Loading#e202#true");

        fireTransitionFound(context, path,
                "5. Moving backward", event,
                "5. Moving backward#6. Loading#e202#true");

        fireComeToState(context, path, "6. Loading");
```

```
                        // 6. Loading [o1.z5]
                        fireBeforeOutputActionExecution(context, path,
                                "5. Moving backward#6. Loading#e202#true",
                                "o1.z5");

                        o1.z5(context);

                        fireAfterOutputActionExecution(context, path,
                                "5. Moving backward#6. Loading#e202#true",
                                "o1.z5");
                        return new StateMachineConfig("6. Loading");

                    case e102:

                        // 5. Moving backward->1. Initialization e102[true]/

                        fireTransitionCandidate(context, path,
                                "5. Moving backward", event,
                                "5. Moving backward#1. Initialization#e102#true");

                        fireTransitionFound(context, path,
                                "5. Moving backward", event,
                                "5. Moving backward#1. Initialization#e102#true");

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

                        // 1. Initialization [o1.z6]
                        fireBeforeOutputActionExecution(
                                context,
                                path,
                                "5. Moving backward#1. Initialization#e102#true",
                                "o1.z6");

                        o1.z6(context);

                        fireAfterOutputActionExecution(
                                context,
                                path,
                                "5. Moving backward#1. Initialization#e102#true",
                                "o1.z6");
                        return new StateMachineConfig("1. Initialization");

                    default:

                        // transition not found
                        return config;
                    }

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

        //o1.x1
        private static final int _o1_x1 = 0;

    }

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

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

        return b;
    }
}
```

Приложение 3. Сборка и запуск программы

3.1. Требования

Для сборки требуется:

- *Java SDK 1.5.0*
- *Apache Ant 1.6.5*

Перед сборкой необходимо убедиться, что переменные окружения `JAVA_HOME` и `ANT_HOME` сконфигурированы корректно.

3.2. Интерпретационный подход

3.2.1. Сборка

Для сборки версии, использующей интерпретационный подход, необходимо запустить командный файл `rebuild_interp.cmd`.

3.2.2. Запуск

Для запуска версии, использующей интерпретационный подход, необходимо запустить командный файл `run_interp.cmd.`

3.3. Компилятивный подход

3.3.1. Сборка

Для сборки версии, использующей компилятивный подход, необходимо запустить командный файл `rebuild_compiled.cmd`.

3.3.2. Запуск

Для запуска версии, использующей компилятивный подход, необходимо запустить командный файл `run_compiled.cmd.`