Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Курсовая работа**

Дисциплина: Технологии разработки программного обеспечения

Тема: Модель движения поезда метро

Выполнили студенты

гр. 5140901/32001:

Белых Б.А.

Божко А.В.

Преподаватель:

Павлов Е.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Санкт-Петербург   
 2023

СОДЕРЖАНИЕ

[1. Описание используемого инструмента, NuSMV 3](#_Toc153278447)

[2. Описание разработанной системы 5](#_Toc153278448)

[3. Результаты проверки разработанной системы 9](#_Toc153278449)

[4. Выводы 10](#_Toc153278450)

[Список источников 10](#_Toc153278451)

[Приложение. Содержание разработанного SMV файла. 10](#_Toc153278452)

# Описание используемого инструмента, NuSMV

NuSMV – это инструмент для тестирования моделей, который широко используется для проверки корректности конечных автоматических систем. Он применяется в основном для верификации аппаратного обеспечения, программного обеспечения и протоколов связи. Является расширением SMV (Symbolic Model Verifier), основанном на принципе BDD (Behavior-driven development, дословно «разработка через поведение»).

Язык SMV обеспечивает модульное иерархическое описание и определение повторно используемых компонентов. Поскольку NuSMV предназначен для описания конечных автоматов, единственными типами данных в языке являются конечные, то есть логические, скалярные, битовые векторы и фиксированные массивы базовых типов данных. NuSMV позволяет моделировать асинхронные системы, конечные автоматы и протоколы, а затем выполняет анализ модели, чтобы найти возможные ошибки, например, зацикливание (deadlock). Он генерирует отчеты, указывающие на найденные ошибки и предоставляющие контрпримеры (counter examples), которые демонстрируют, как система достигла неверного состояния.

Являющийся модификацией языка модели проверки SMV, NuSMV предоставляет некоторые дополнительные функции. В дополнение к обычному пакетному режиму SMV, NuSMV предоставляет оболочку текстового взаимодействия. Через оболочку пользователь может активировать различные этапы вычисления NuSMV в виде системных команд с различными параметрами. Эти этапы вычислений могут быть вызваны отдельно и, возможно, отменены. Специализированные процедуры NuSMV позволяют проверять инварианты, то есть формулы, которые должны быть единообразными на модели, на лету во время анализа достижимости. Также, модель может быть разделена конъюнктивно и дизъюнктивно. Разделы могут быть проверены и (для конъюнктивного случая) упорядочены в соответствии с эвристикой. Ещё одна дополнительная функция, доступная в NuSMV, – это проверка модели LTL, которая выполняется путем сокращения до проверки модели CTL в соответствии с алгоритмом. Спецификация LTL автоматически преобразуется в таблицу, которая затем используется для расширения модели в синхронном продукте. Результат обеспечивается проверкой истинности формулы CTL в расширенной модели.

NuSMV имеет открытый исходный код, благодаря чему разрабатывается и поддерживается активным сообществом. Он предоставляет гибкую и мощную платформу для формального анализа систем, а также интегрируется с другими инструментами и методологиями верификации.

В данной работе используется NuSMV версии 2.6.0 для Windows.

# Описание разработанной системы

Представленная в файле «metro\_sim.SMV» система описывает движение поезда метро со станции на станцию в двух режимах – с посадкой и без посадки (разгрузкой). У поезда автоматизировано открытие и закрытие дверей, пассажиры на станции могут садиться или сходить с поезда, сам поезд чередует своё положение на станции и в туннеле.

По приезде на станцию в режиме с посадкой, поезд открывает ранее закрытые двери. При открытых дверях, с него сначала сходят пассажиры (если поезд изначально не пуст), после чего пассажиры в него заходят. Как только движение пассажиров заканчивается, двери поезда закрываются. Закончивший пребывание на станции в установленном режиме поезд уезжает в туннель.

По приезде на станцию в режиме без посадки, поезд может не открывать двери. Однако, если поезд не пуст, ему необходимо выгрузить пассажиров. В таком случае двери обязательно открываются. Пассажиры начинают сходить с поезда, а значит он впоследствии оказывается пуст. Как только это происходит, двери поезда закрываются. Закончивший пребывание на станции в установленном режиме поезд уезжает в туннель.

Для наглядности работы оба этих режима чередуются на каждой следующей станции.

Таким образом, сформулируем задействованные модули:

|  |  |
| --- | --- |
| Модуль | Принимаемые значения |
| Пассажиры | Бездействуют  Входят  Выходят |
| Наполненность поезда | С пассажирами  Пустой |
| Положение поезда | Не на станции  На станции |
| Двери поезда | Закрыты  Открыты |
| Режим поезда | Для погрузки  Без погрузки  Конец погрузки |

Реализуем данные модули в формате SMV:

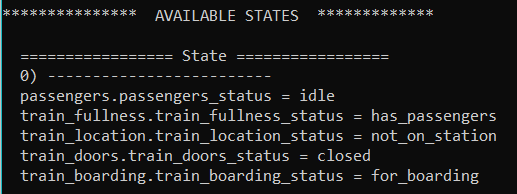
|  |
| --- |
| MODULE train\_location(train\_boarding\_status, train\_doors\_status, train\_fullness\_status)  VAR  train\_location\_status : {on\_station, not\_on\_station};  …  MODULE train\_boarding(train\_doors\_status, passengers\_status, train\_location\_status)  VAR  train\_boarding\_status : {for\_boarding, no\_boarding, finish\_boarding};  …  MODULE train\_doors(train\_boarding\_status, train\_location\_status, passengers\_status, train\_fullness\_status)  VAR  train\_doors\_status : {closed, open};  …  MODULE passengers(train\_doors\_status, train\_location\_status, train\_boarding\_status, train\_fullness\_status)  VAR  passengers\_status : {entering, exiting, idle};  …  MODULE train\_fullness(passengers\_status, train\_boarding\_status)  VAR  train\_fullness\_status : {empty, has\_passengers};  … |

Опишем условия, при которых будут приниматься значения переменных в приведённых модулях. Ниже приведены отрывки полученных условий, к которым приведены соответствующие комментарии.

|  |
| --- |
| …  ASSIGN  init(train\_location\_status) := not\_on\_station;  **-- Дефолтно поезд едет в туннеле**  next(train\_location\_status) := case  train\_location\_status = not\_on\_station : on\_station;  **-- ЕСЛИ не на станции => на станции**  train\_location\_status = on\_station & ((train\_boarding\_status = no\_boarding & train\_fullness\_status = empty) | train\_boarding\_status = finish\_boarding) & train\_doors\_status = closed : not\_on\_station;  **-- ЕСЛИ на станции И ((без погрузки И пустой) ИЛИ конец погрузки) И закрыты => не на станции**  TRUE : train\_location\_status;  esac;  …  ASSIGN  init(train\_boarding\_status) := for\_boarding;  **-- Дефолтно на поезд будет совершаться посадка пассажиров**  next(train\_boarding\_status) := case  train\_boarding\_status = for\_boarding & train\_doors\_status = open & passengers\_status = idle: finish\_boarding;  **-- ЕСЛИ для погрузки И открыты И бездействуют => конец погрузки**  train\_boarding\_status = finish\_boarding & train\_location\_status = not\_on\_station : no\_boarding;  **-- ЕСЛИ конец погрузки И не на станции => без погрузки**  train\_boarding\_status = no\_boarding & train\_location\_status = not\_on\_station : for\_boarding;  **-- ЕСЛИ без погрузки И не на станции => для погрузки**  TRUE : train\_boarding\_status;  esac;  ASSIGN  init(train\_doors\_status) := closed;  **-- Дефолтно двери поезда закрыты**  next(train\_doors\_status) := case  train\_doors\_status = closed & train\_boarding\_status = for\_boarding & train\_location\_status = on\_station : open;  **-- ЕСЛИ закрыты И для погрузки И на станции => открыты**  train\_doors\_status = open & (train\_boarding\_status = finish\_boarding | (train\_boarding\_status = no\_boarding & train\_fullness\_status = empty)) & passengers\_status = idle : closed;  **-- ЕСЛИ открыты И (конец погрузки ИЛИ (без погрузки И пустой)) И бездействуют => закрыты**  train\_doors\_status = closed & train\_location\_status = on\_station & train\_boarding\_status = no\_boarding & train\_fullness\_status = has\_passengers : open;  **-- ЕСЛИ закрыты И на станции И без погрузки И с пассажирами => открыты**  TRUE : train\_doors\_status;  esac;  …  ASSIGN  init(passengers\_status) := idle;  **-- Пассажиры бездействуют, пока не будут иметь возможность погружаться или сходить с поезда**  next(passengers\_status) := case  passengers\_status != idle & (train\_doors\_status = closed | train\_location\_status = not\_on\_station) : idle;  **-- ЕСЛИ НЕ бездействуют и (закрыты ИЛИ не на станции) => бездействуют**  passengers\_status = idle & train\_location\_status = on\_station & train\_boarding\_status = for\_boarding & train\_fullness\_status = has\_passengers & train\_doors\_status = open : exiting;  **-- ЕСЛИ бездействуют И на станции И для погрузки И с пассажирами И открыты => выходят**  (passengers\_status = exiting | train\_fullness\_status = empty) & train\_location\_status = on\_station & train\_doors\_status = open & train\_boarding\_status != no\_boarding: entering;  **-- ЕСЛИ (выходят ИЛИ пустой) И на станции И НЕ без погрузки И открыты => входят**  passengers\_status = entering : idle;  **-- ЕСЛИ входят => бездействуют**  passengers\_status != exiting & train\_doors\_status = open & train\_boarding\_status = no\_boarding & train\_fullness\_status = has\_passengers : exiting;  **-- ЕСЛИ НЕ выходят И открыты И без погрузки И с пассажирами => выходят**  passengers\_status = exiting & train\_fullness\_status = empty : idle;  **-- ЕСЛИ выходят И пустой => бездействуют**  TRUE : passengers\_status;  esac;  …  ASSIGN  init(train\_fullness\_status) := has\_passengers;  **-- Дефолтно поезд не пустой**  next(train\_fullness\_status) := case  train\_fullness\_status = empty & passengers\_status = entering : has\_passengers;  **-- ЕСЛИ пустой И входят => с пассажирами**  train\_fullness\_status = has\_passengers & train\_boarding\_status = no\_boarding & passengers\_status = exiting : empty;  **-- ЕСЛИ без погрузки И с пассажирами И выходят => пустой**  TRUE : train\_fullness\_status;  esac;  … |

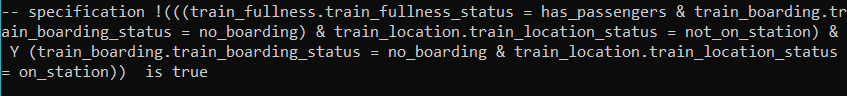
# Результаты проверки разработанной системы

Запустим NuSMV 2.6.0. Прочитаем файл с нашей моделью и соберём её. Начальное состояние системы:

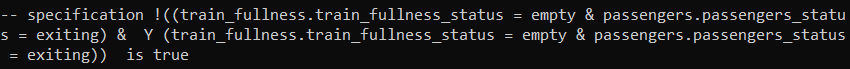


Проведём следующие проверки:

1. Поезд в режиме без посадки не должен покинуть станцию с пассажирами внутри.



1. Из пустого поезда не могут выходить пассажиры.



1. Поезд всегда может находиться в туннеле только на протяжении одного состояния. Нахождение в туннеле два состояния подряд и более не допускаются.



1. Пассажиры не могут входить или выходить из поезда, когда он находится в туннеле.



1. Пассажиры не могут входить или выходить из поезда, когда его двери закрыты.



1. Поезд не может быть в туннеле с открытыми дверями.



# Выводы

В ходе выполнения курсовой работы был освоен и задействован инструмент NuSMV, при помощи которого была разработана и протестирована модель движения поезда метро. Тестирование подтвердило, что выдвинутые к системе требования и ограничения были реализованы корректно, а работа модели полностью соответствует ожиданиям.

# Список источников

Сайт NuSMV: <https://nusmv.fbk.eu/>

NuSMV 2.6 User Manual: <https://nusmv.fbk.eu/userman/v26/nusmv.pdf>

Обзор средства NuSMV: <https://studizba.com/files/matematicheskie-metody-verifikacii-shem-i/lectures/237461-4.-nusmv-obzor-sredstva..html>

# Приложение. Содержание разработанного SMV файла.

|  |
| --- |
| MODULE train\_location(train\_boarding\_status, train\_doors\_status, train\_fullness\_status)  VAR  train\_location\_status : {on\_station, not\_on\_station};  ASSIGN  init(train\_location\_status) := not\_on\_station;  -- Дефолтно поезд едет в туннеле  next(train\_location\_status) := case  train\_location\_status = not\_on\_station : on\_station;  -- ЕСЛИ не на станции => на станции  train\_location\_status = on\_station & ((train\_boarding\_status = no\_boarding & train\_fullness\_status = empty) | train\_boarding\_status = finish\_boarding) & train\_doors\_status = closed : not\_on\_station;  -- ЕСЛИ на станции И ((без погрузки И пустой) ИЛИ конец погрузки) И закрыты => не на станции  TRUE : train\_location\_status;  esac;  MODULE train\_boarding(train\_doors\_status, passengers\_status, train\_location\_status)  VAR  train\_boarding\_status : {for\_boarding, no\_boarding, finish\_boarding};  ASSIGN  init(train\_boarding\_status) := for\_boarding;  -- Дефолтно на поезд будет совершаться посадка пассажиров  next(train\_boarding\_status) := case  train\_boarding\_status = for\_boarding & train\_doors\_status = open & passengers\_status = idle: finish\_boarding;  -- ЕСЛИ для погрузки И открыты И бездействуют => конец погрузки  train\_boarding\_status = finish\_boarding & train\_location\_status = not\_on\_station : no\_boarding;  -- ЕСЛИ конец погрузки И не на станции => без погрузки  train\_boarding\_status = no\_boarding & train\_location\_status = not\_on\_station : for\_boarding;  -- ЕСЛИ без погрузки И не на станции => для погрузки  TRUE : train\_boarding\_status;  esac;    MODULE train\_doors(train\_boarding\_status, train\_location\_status, passengers\_status, train\_fullness\_status)  VAR  train\_doors\_status : {closed, open};  ASSIGN  init(train\_doors\_status) := closed;  -- Дефолтно двери поезда закрыты  next(train\_doors\_status) := case  train\_doors\_status = closed & train\_boarding\_status = for\_boarding & train\_location\_status = on\_station : open;  -- ЕСЛИ закрыты И для погрузки И на станции => открыты  train\_doors\_status = open & (train\_boarding\_status = finish\_boarding | (train\_boarding\_status = no\_boarding & train\_fullness\_status = empty)) & passengers\_status = idle : closed;  -- ЕСЛИ открыты И (конец погрузки ИЛИ (без погрузки И пустой)) И бездействуют => закрыты  train\_doors\_status = closed & train\_location\_status = on\_station & train\_boarding\_status = no\_boarding & train\_fullness\_status = has\_passengers : open;  -- ЕСЛИ закрыты И на станции И без погрузки И с пассажирами => открыты  TRUE : train\_doors\_status;  esac;    MODULE passengers(train\_doors\_status, train\_location\_status, train\_boarding\_status, train\_fullness\_status)  VAR  passengers\_status : {entering, exiting, idle};  ASSIGN  init(passengers\_status) := idle;  -- Пассажиры бездействуют, пока не будут иметь возможность погружаться или сходить с поезда  next(passengers\_status) := case  passengers\_status != idle & (train\_doors\_status = closed | train\_location\_status = not\_on\_station) : idle;  -- ЕСЛИ закрыты ИЛИ не на станции => бездействуют  passengers\_status = idle & train\_location\_status = on\_station & train\_boarding\_status = for\_boarding & train\_fullness\_status = has\_passengers & train\_doors\_status = open : exiting;  -- ЕСЛИ бездействуют И на станции И для погрузки И с пассажирами И открыты => выходят  (passengers\_status = exiting | train\_fullness\_status = empty) & train\_location\_status = on\_station & train\_doors\_status = open & train\_boarding\_status != no\_boarding: entering;  -- ЕСЛИ (выходят ИЛИ пустой) И на станции И для погрузки И открыты И НЕ без погрузки=> входят  passengers\_status = entering : idle;  -- ЕСЛИ входят И на станции И для погрузки И открыты => бездействуют  passengers\_status != exiting & train\_doors\_status = open & train\_boarding\_status = no\_boarding & train\_fullness\_status = has\_passengers : exiting;  -- ЕСЛИ НЕ выходят И открыты И без погрузки И с пассажирами => выходят  passengers\_status = exiting & train\_fullness\_status = empty : idle;  -- ЕСЛИ выходят И пустой => бездействуют  TRUE : passengers\_status;  esac;  MODULE train\_fullness(passengers\_status, train\_boarding\_status)  VAR  train\_fullness\_status : {empty, has\_passengers};  ASSIGN  init(train\_fullness\_status) := has\_passengers;  -- Дефолтно поезд не пустой  next(train\_fullness\_status) := case  train\_fullness\_status = empty & passengers\_status = entering : has\_passengers;  -- ЕСЛИ входят => с пассажирами  train\_fullness\_status = has\_passengers & train\_boarding\_status = no\_boarding & passengers\_status = exiting : empty;  -- ЕСЛИ без погрузки И с пассажирами И выходят => пустой  TRUE : train\_fullness\_status;  esac;    MODULE main()  VAR  passengers : passengers(train\_doors.train\_doors\_status, train\_location.train\_location\_status, train\_boarding.train\_boarding\_status, train\_fullness.train\_fullness\_status);  train\_fullness : train\_fullness(passengers.passengers\_status, train\_boarding.train\_boarding\_status);  train\_location : train\_location(train\_boarding.train\_boarding\_status, train\_doors.train\_doors\_status, train\_fullness.train\_fullness\_status);  train\_doors : train\_doors(train\_boarding.train\_boarding\_status, train\_location.train\_location\_status, passengers.passengers\_status, train\_fullness.train\_fullness\_status);  train\_boarding : train\_boarding(train\_doors.train\_doors\_status, passengers.passengers\_status, train\_location.train\_location\_status);      -- LTL  -- без посадки не может уехать с пассажирами – не может быть не на станции и с пассажирами  LTLSPEC !(train\_fullness.train\_fullness\_status = has\_passengers & train\_boarding.train\_boarding\_status = no\_boarding & train\_location.train\_location\_status = not\_on\_station & Y (train\_boarding.train\_boarding\_status = no\_boarding & train\_location.train\_location\_status = on\_station))  -- из пустого поезда не могут выходить пассажиры  LTLSPEC !(train\_fullness.train\_fullness\_status = empty & passengers.passengers\_status = exiting & Y (train\_fullness.train\_fullness\_status = empty & passengers.passengers\_status = exiting))  -- CTL  -- Поезд всегда находится не на станции только меньше, чем два состояния подряд  SPEC AG (train\_location.train\_location\_status = not\_on\_station -> AF train\_location.train\_location\_status = on\_station)  -- пассажиры не могут входить или выходить при поезде не на станции  SPEC AG !(passengers.passengers\_status != idle & train\_location.train\_location\_status = not\_on\_station)  -- пассажиры не могут входить или выходить при закрытых дверях  SPEC AG !(passengers.passengers\_status != idle & train\_doors.train\_doors\_status = closed)  -- поезд не может быть не на станции с открытыми дверями  SPEC AG !(train\_location.train\_location\_status = not\_on\_station & train\_doors.train\_doors\_status = open) |