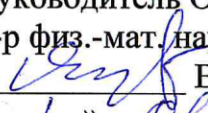


Министерство образования и науки Российской Федерации  
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (ТГУ)  
Радиофизический факультет  
Кафедра информационных технологий в исследовании дискретных структур  
(КИТИДиС)

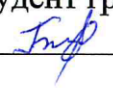
ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК  
Руководитель ООП  
д-р физ.-мат. наук, профессор  
 В.П.Якубов  
« 1 » августа 2018 г.

Батрацкий Сергей Васильевич

РАЗРАБОТКА И ТЕСТИРОВАНИЕ НА ОСНОВЕ ФОРМАЛЬНЫХ МОДЕЛЕЙ  
ПАКЕТА ПРИКЛАДНЫХ ПРОГРАММ «FSMTest-2.0»

**МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ**  
на соискание степени магистра радиофизики  
по направлению подготовки  
03.04.03 - Радиофизика

Руководитель ВКР  
доцент, к.т.н.  
 С.А.Прокопенко  
« 1 » июня 2018г.

Автор работы  
студент группы № 721  
 С.В.Батрацкий

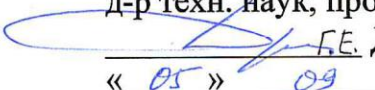
Томск-2018

Министерство образования и науки Российской Федерации  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (ТГУ)  
Радиофизический факультет (РФФ)  
Кафедра информационных технологий в исследовании дискретных структур  
(КИТИДиС)

УТВЕРЖДАЮ

Руководитель ООП

д-р техн. наук, профессор

 Г.Е. Дунаевский

« 05 » 09 2016 г.

ЗАДАНИЕ

по подготовке ВКР магистру студенту группы № 721 Батрацкому Сергею  
Васильевичу

1. Тема ВКР РАЗРАБОТКА И ТЕСТИРОВАНИЕ НА ОСНОВЕ  
ФОРМАЛЬНЫХ МОДЕЛЕЙ ПАКЕТА ПРИКЛАДНЫХ ПРОГРАММ «FSMTest-  
2.0»

2. Срок сдачи студентом выполненной ВКР:

а) на кафедре 31.05.18;

б) в ГАК 21-22.06.18.

3. Исходные данные к работе проверка и разбор XML-документа с  
переводом данных в формат, используемый ядром программы, тестирование  
программных реализаций, входящих в «FSMTest-2.0»

В качестве объекта исследования используется пакет прикладных программ  
«FSMTest-2.0». Исследования проводятся методами технической диагностики,  
теории автоматов и дискретной математики. Эффективность предложенного  
подхода к тестированию программных реализаций подтверждается  
экспериментальными результатами с программной реализацией ndfsm\_test.

4. Краткое содержание работы

– литературный обзор методов разбора и проверки XML-документов (ноябрь  
2016);

– создание модулей проверки и разбора XML-документов, а так же  
разработка алгоритма разбора дерева XML-документа, который преобразует  
данные в формат, используемый ядром пакета прикладных программ (февраль  
2017);

– выявление и устранение ошибок в пакете прикладных программ «FSMTest-  
2.0» (май 2017);

– извлечение автоматной модели из спецификации программы (ноябрь 2017);

– проверка правильности реализаций Java программ (март 2018);

– написание магистерской диссертации (май 2018).

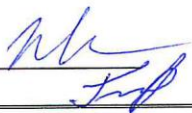
5. Работа выполняется в Национальном исследовательском Томском  
государственном университете

6. Дата выдачи задания «05» сентября 2016 г.

Руководитель ВКР

Доцент, к.т.н.

Задание принял к исполнению 05.09.16

 С.А. Прокопенко

## РЕФЕРАТ

Магистерская диссертация 91 с., 3 гл., 20 рис., 2 табл., 17 источников, 4 приложения.

КОНЕЧНЫЙ АВТОМАТ, РАСШИРЕННЫЙ АВТОМАТ, ОБОЛОЧКА, ТЕСТИРОВАНИЕ, FSMTest-2.0, ПАРСЕР, JUnit.

В данной диссертации рассматриваются задачи проверки и разбора XML-документов, а также тестирование программного обеспечения.

Разработаны и программно реализованы классы для проверки и разбора XML-документов в соответствии с требованиями к пакету прикладных программ «FSMTest-2.0». В данный пакет так же добавлены две реализации: нахождение синхронизирующих последовательностей в конечном автомате и поиск пересечения двух конечных, возможно недетерминированных, автоматов.

С помощью инструмента JUnit произведено тестирование программы на языке Java ndfsm\_test, входящей в «FSMTest-2.0» и позволяющей построить проверяющий тест для недетерминированного конечного автомата.

## СОДЕРЖАНИЕ

|                                                                                                                             |    |
|-----------------------------------------------------------------------------------------------------------------------------|----|
| ВВЕДЕНИЕ                                                                                                                    | 4  |
| 1 Основные понятия и определения                                                                                            | 8  |
| 2 Создание программы «FSMTest-2.0»                                                                                          | 13 |
| 2.1 Цель создания программы                                                                                                 | 13 |
| 2.2 Структура программы «FSMTest-2.0»                                                                                       | 14 |
| 2.3 Разработка алгоритма разбора дерева, позволяющего преобразовывать<br>данные в формат, используемый ядром «FSM Test-2.0» | 15 |
| 2.3.1 Существующие библиотеки реализации парсеров                                                                           | 15 |
| 2.3.2 Структура и свойства XML-документа                                                                                    | 16 |
| 2.3.3 Алгоритм проверки XML-документа                                                                                       | 21 |
| 2.3.4 Алгоритм разбора дерева парсинга XML-документа                                                                        | 21 |
| 2.3.5 Методы проверки XML-документа                                                                                         | 22 |
| 2.3.6 Методы разбора XML-документа                                                                                          | 23 |
| 2.3.7 Новые реализации                                                                                                      | 25 |
| 2.3.8 Ошибки, недоработки и их исправление                                                                                  | 27 |
| 2.3.9 Проверка работы программы «FSMTest-2.0»                                                                               | 32 |
| 2.3.10 Проверка работы новых реализаций                                                                                     | 37 |
| 3 Тестирование программных реализаций «FSMTest-2.0»                                                                         | 44 |
| 3.1 Схема и цель тестирования                                                                                               | 44 |
| 3.2 Представление программы                                                                                                 | 45 |
| 3.3 Тестирование программной реализации                                                                                     | 45 |
| ЗАКЛЮЧЕНИЕ                                                                                                                  | 48 |
| СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ                                                                                              | 49 |
| Приложение А. Код парсера                                                                                                   | 51 |
| Приложение Б. Код шаблонного XML-документа                                                                                  | 72 |
| Приложение В. XML-документы, описывающие окна интерфейса                                                                    | 74 |
| Приложение Г. Код теста                                                                                                     | 84 |

## ВВЕДЕНИЕ

**Актуальность проблемы.** Сотрудниками кафедры информационных технологий в исследовании дискретных структур создан пакет программ «FSMTest-1.0», предназначенный для анализа и синтеза конечных автоматов. Однако у данного пакета есть два недостатка: он не является кроссплатформенным, и в него трудно добавлять новые реализации программ. В связи с этим было принято решение создать новую оболочку, которая будет лишена данных проблем.

Задача создания оболочки, генерирующей интерфейс, является актуальной, так как имеется множество программных реализаций, которые нужно соединить воедино, при этом программа должна быть кроссплатформенной.

Под оболочкой понимается программа, созданная для упрощения работы со сложными программными системами, такими, например, как DOS. Они преобразуют неудобный командный пользовательский интерфейс, в дружелюбный графический интерфейс, или интерфейс типа «меню».

Реализуемая оболочка должна создавать графический интерфейс пользователя по описанию, хранящемуся в XML-документах. В связи с этим возникает задача разбора XML-документов и организация потока данных ядру оболочки.

Кроме того, требуется проверять программы, входящие в новый пакет «FSMTest-2.0». Для того, чтобы иметь возможность строить тесты с гарантированной полнотой, необходимо иметь математическую модель программы.

Существуют различные математические модели программ [1-3], такие как типизированное  $\lambda$ -исчисление, на базе которого могут быть построены среды автоматизации математических доказательств, язык формальной спецификации которых в то же время является статистически типизированным функциональным языком программирования; *PVS (Prototype Verification System)* – инструмент для выявления ошибок и потенциальных уязвимостей в исходном коде программ, написанных на языках C, C++ и C#; модель денотационной формальной семантики подмножества языка C (стандарта ANSI C), основная структура которой опирается на описание вычислений с использованием специальных конструкций; конечный автомат; расширенный автомат, логические схемы.

В работе [2] используется подход, основанный на построении расширенного автомата по программе. В данной работе предлагается выделять расширенный автомат из спецификации программы. И проверяющий тест для программы строится по конечному автомату, эквивалентному построенному расширенному автомату.

**Целью** данной работы является создание модуля проверки XML-документа, который позволяет разбирать дерево XML-документа и создает поток данных, поступающих на вход ядра программы «FSMTest-2.0», а так же проверка правильности реализаций Java программ.

**Методы исследования.** Используются методы дискретной математики, теории автоматов и технической диагностики.

**Научная новизна.** В работе предлагается на основе дерева, полученного в процессе разбора XML-документа, сформировать набор данных, которые пригодны для отрисовки вкладок программных реализаций ядром программы «FSMTest-2.0». Кроме того, проводится тестирование программной реализации на языке Java, входящей в «FSMTest-2.0», с использованием инструмента FSMTestJUnit на основе формальных моделей.

**Основные положения, выносимые на защиту.**

1. Алгоритм разбора дерева, построенного на основе парсинга XML-документа, позволяющий преобразовывать данные в формат, используемый ядром программы «FSMTest-2.0».
2. Использование модели расширенного автомата программы, позволяющей построить полный проверяющий тест программы относительно ошибок переходов и выходов.

**Достоверность результатов.** Эффективность предложенного подхода к тестированию программных реализаций подтверждается экспериментальными результатами с программной реализацией `ndfsm_test`.

**Практическая ценность.** Результаты, полученные в работе, могут быть использованы для добавления новых программных реализаций в «FSMTest-2.0», а так же для построения тестов с гарантированной полнотой для программ на языке Java.

**Апробация работы.** Основные положения и результаты, представленные в работе, обсуждались на трёх конференциях и опубликованы четыре работы:

1. Батрацкий, С.В. Создание кроссплатформенной версии приложение «FSM Test-1.0» / С.В. Батрацкий, В.С. Белых, А.С. Твардовский // Сборник научных трудов в 9 ч. / под ред. проф. Б.Ю. Лемешко, проф. А.А. Попова, проф. М.Э. Рояка, доц. В.С. Тимофеева. – Новосибирск: Изд-во НГТУ, 2016. – Часть 2. С. 163-165.
2. Батрацкий, С.В. К программной реализации пакета прикладных программ «FSMTest-2.0». / С.В. Батрацкий, В.С. Белых. // Всероссийская конференция «Студенческий научно-исследовательский инкубатор» ТГУ, 2017. – С. 6.
3. Шаляпина, Н.А. Тесты на константные неисправности как веб-сервис / Н.А. Шаляпина, А.А. Зайцев, С.В. Батрацкий, М.Л. Громов // Труды Института системного программирования РАН. Том 30, выпуск 1, 2018. С. 41-54.
4. Батрацкий, С.В. Тестирование Java программ с использованием инструмента FSMTEST2JUNIT / С.В. Батрацкий, С.А. Прокопенко, М.Л. Громов, Н.В. Шабалдина // Новые информационные технологии в исследовании сложных структур: материалы двенадцатой российской конференции с международным участием. Томск : Издательский Дом Томского государственного университета, 2018. – С. 67.

#### **Краткое содержание работы.**

**Во введении** приводится актуальность работы, сформулирована цель и задачи работы, представлены положения, выносимые на защиту и апробация работы.

**В первой главе** диссертации вводятся необходимые понятия и определения, используемые в работе.

**Во второй главе** приведено описание программы «FSMTest-1.0», которую требуется улучшить, структура программы «FSMTest-2.0», структура и свойства XML-документов, а также обзор библиотек для работы с XML-документами, алгоритмы разбора и проверки XML-документов, добавление новых реализаций и описание и проверка разработанной программы «FSMTest-2.0».

**В третьей главе** представлены схема тестирования программных компонент, входящих в «FSMTest-2.0», алгоритм тестирования, представление программы в виде расширенного автомата и результат тестирования.

**В заключении** кратко излагаются основные результаты, полученные в процессе выполнения работы.



## 1 Основные понятия и определения

В данной главе рассматриваются основные понятия и определения, используемые в работе.

Под конечным автоматом [4] понимается пятёрка  $A = (S, I, O, h_S, s_0)$ , где  $S$  – непустое конечное множество состояний с выделенным начальным состоянием  $s_0$ ,  $I$  и  $O$  – входной и выходной алфавиты соответственно,  $h_S \subseteq S \times I \times O \times S$  – отношение переходов. Мы рассматриваем конечный и детерминированный автомат, т.е. автомат, в котором для каждой пары  $(s, i) \in S \times I$  существует только один переход  $(s, i, o, s') \in h_S$ . Конечные автоматные модели широко используются при описании поведения систем, которые производят выходные последовательности в ответ на входные последовательности.

Пример конечного автомата: Дано вращение колеса, приводимого в движение двигателем, определяется двухпозиционным ключом; правое положение ключа соответствует вращению в направлении по часовой стрелке, левое – против часовой стрелки. В момент изменения направления вращения вспыхивает индикаторная лампа.

$I = \{\text{справа, слева}\},$

$O = \{\text{лампа включена, лампа выключена}\},$

$S = \{\text{по часовой стрелке, против часовой стрелки}\}.$

При состоянии в настоящий момент «по часовой стрелке» и входе «справа» или при состоянии в настоящий момент «против часовой стрелки» и в ходе «слева» состояние остаётся неизменным и имеется выход «лампа выключена». При состоянии в настоящий момент «по часовой стрелке» и входе «слева» или при состоянии в настоящий момент «против часовой стрелки» и входе «справа» состояние изменяется и появляется выход «лампа включена». На рисунке 1 данный автомат представлен в графическом виде.

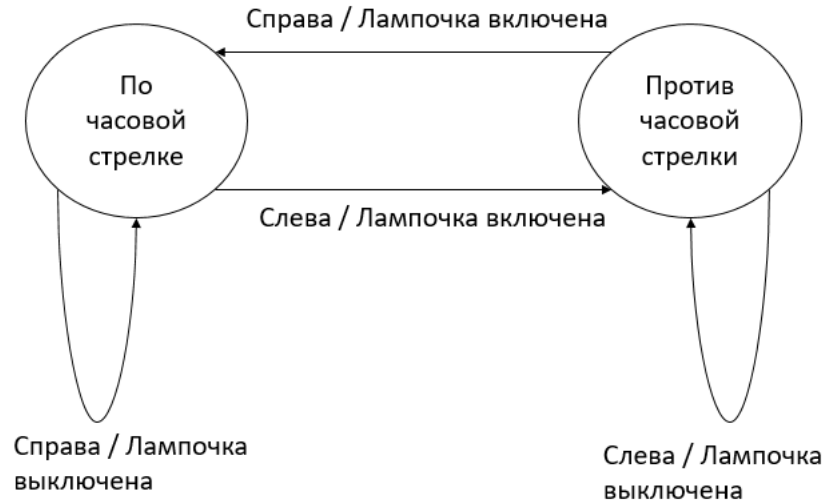


Рисунок 1 – Пример автомата в графическом виде

Расширенный автомат [5] представляет собой пятёрку  $M = (S, I, O, V, T)$ , где  $S$  – непустое конечное множество состояний автомата,  $I$  – непустое множество входных символов, называемое входным алфавитом,  $O$  – непустое множество выходных символов, называемое выходным алфавитом,  $V$  – конечное, возможно пустое множество контекстных переменных,  $T$  – отношение переходов. Каждый переход  $t$  из  $T$  – это семёрка  $(s, i, P, op, o, up, s')$ , где  $s, s' \in S$  являются начальным и конечным состояниями перехода;  $i \in I$  есть входной символ и  $D_{inp-i}$  обозначает множество векторов, компонентами которых являются значения параметров, соответствующих входному символу  $i$  (далее входные параметры);  $o \in O$  – выходной символ и  $D_{out-o}$  обозначает множество векторов, компонентами которых являются значения параметров, соответствующих выходному символу  $o$  (далее выходные параметры);  $P$ ,  $op$  и  $up$  – функции, определенные над входными параметрами и контекстными переменными из  $V$ :

$P: D_{inp-i} \times D_V \rightarrow \{\text{истина, ложь}\}$  – предикат, где  $D_V$  – множество контекстных векторов;

$op: D_{inp-i} \times D_V \rightarrow D_{out-o}$  – функции для определения значений выходных параметров;

$up: D_{inp-i} \times D_V \rightarrow D_V$  – функции для определения значений контекстных переменных.

Параметризованным входным (выходным) символом называется пара «входной символ  $i$ , вектор  $a$  из  $D_{inp-i}$ », т.е. пара  $(i, a)$  («выходной символ  $o$ , вектор  $b$  из  $D_{out-o}$ »).

Конфигурацией расширенного автомата  $M$  называется пара «состояние  $s$ , контекстный вектор  $v$ », то есть  $(s, v)$ .

Расширенный автомат называется полностью определенным, если в каждом состоянии  $s$  существует хотя бы один переход по каждому входному символу. Расширенный автомат  $M$  называется непротиворечивым, если из каждого состояния  $s$  для любого параметризованного входного символа и каждого значения контекстного вектора  $v$  из области определения существует не более одного перехода, предикат которого принимает значение «истина» при данном значении контекстного вектора. Расширенный автомат  $M$  называется детерминированным, если в каждом состоянии  $s$  существует не более одного выполнимого перехода по любому параметризованному входному символу.

*Graphical user interface (GUI)* – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т.п.), представленные пользователю на дисплее, исполнены в виде графических изображений. На рисунке 2 представлен пример GUI программы Guitar Pro 7, очевидно, что, если бы пользователю приходилось использовать командную строку, с программой было бы крайне сложно работать, если вообще возможно для обычного человека.

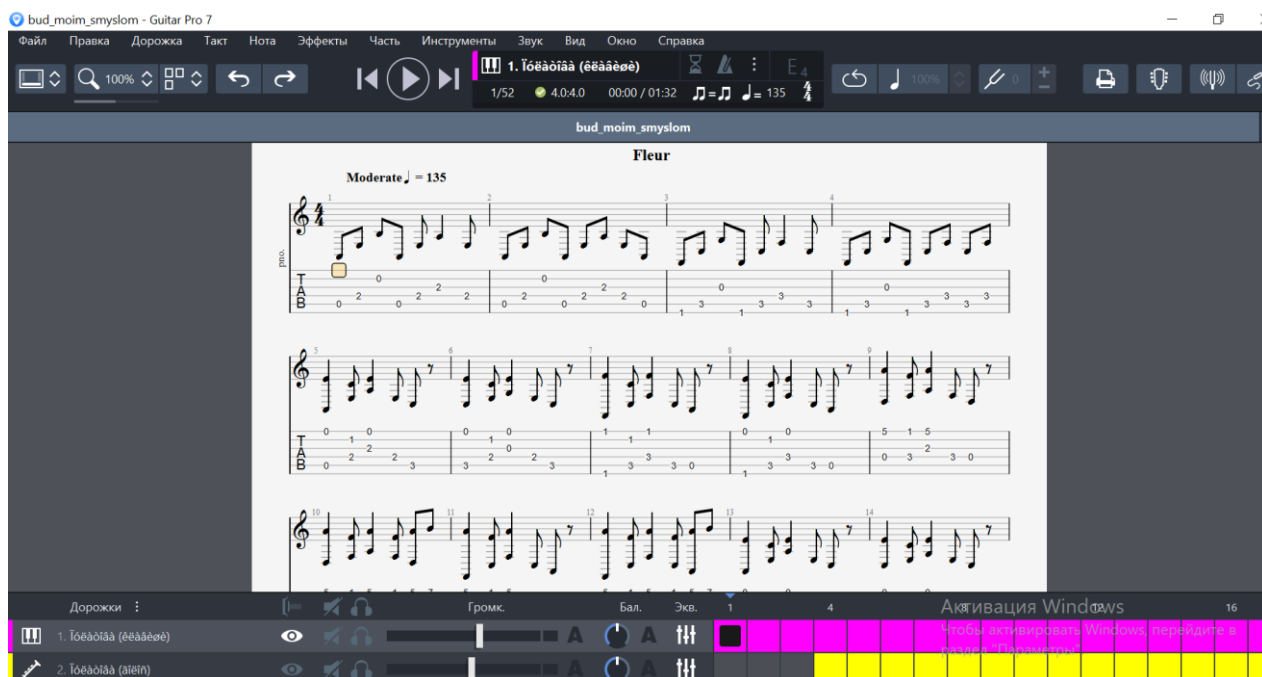


Рисунок 2 – Пример GUI

*Оболочка* [6] – программа, созданная для упрощения работы со сложными программными системами, такими, например, как DOS. Они преобразуют неудобный командный пользовательский интерфейс, в дружелюбный графический интерфейс, или интерфейс типа «меню».

*Application programming interface* [7] (API) – набор готовых классов, процедур, функций, структур и констант, представляемых приложением.

*Extensible Markup Language* (XML) [8] даёт возможность создавать текст и размечать его при помощи обрамляющих тегов, превращая каждое слово, предложение или фрагмент в идентифицируемую, сортируемую информацию.

Первой строчкой XML-документа может быть *декларация*, это необязательная часть, но она может помочь определить документ, как XML.

Обязательным элементом XML-документа является корневой элемент, он должен быть в единственном экземпляре. *Корневым*, называется элемент, начальный и конечный теги которого, обрамляют весь документ, не считая декларации.

К элементам можно добавить атрибуты. *Атрибутом* называется пара имя-значение, где значение берётся в двойные кавычки (“value”). Атрибуты позволяют добавить элементам дополнительные параметры.

Под *корректным* XML-документом, понимается такой документ, который удовлетворяет синтаксису и требуемому формату XML-документа.

В работе [2] в качестве математической модели программы используется расширенный автомат. В объектно-ориентированном программировании экземпляр класса можно рассматривать как автомат. В данном случае, состоянию автомата ставится в соответствие состояние экземпляра класса; входному символу – операция в экземпляре класса с некоторым набором значений входных параметров; выходному символу – набор значений выходных параметров.

В данном случае тестирование можно осуществлять путем обхода графа переходов соответствующего автомата. В [2] приводится алгоритм для детерминированного автомата. Кроме того, обход графа переходов позволяет обнаруживать только выходные ошибки, но вопрос о полноте такого теста относительно ошибок переходов остается открытым.

В данной работе предлагается строить по спецификации программы расширенный автомат подобным образом. Состояниям расширенного автомата ставятся в соответствие экземпляры класса, входным символам – методы класса, а выходным символам – реакции экземпляра класса на входные воздействия. Поскольку методы построения тестов для расширенных автоматов недостаточно изучены, то расширенный автомат преобразуется к эквивалентному конечному автомату и тест строится для последнего одним из известных методов, например [9, 10].

## 2 Создание программы «FSMTest-2.0»

В данной главе рассматриваются цель создания программы, структура программы, а также тестирование программы и добавление новых реализаций.

### 2.1 Цель создания программы

Сотрудниками кафедры информационных технологий в исследовании дискретных структур создан пакет программ «FSMTest-1.0» [11], предназначенный для тестирования конечных автоматов (рисунок 3). У данного пакета есть два недостатка: он не является кроссплатформенным, и в него трудно добавлять новые программные реализации. В связи с этим было принято решение создать новую оболочку «FSMTest-2.0» [12-14], которая будет лишена данных проблем.

Реализуемая оболочка должна создавать графический интерфейс пользователя по описанию, хранящемуся в XML-документах. В связи с этим возникает задача разбора XML-документов и организация потока данных ядру оболочки.

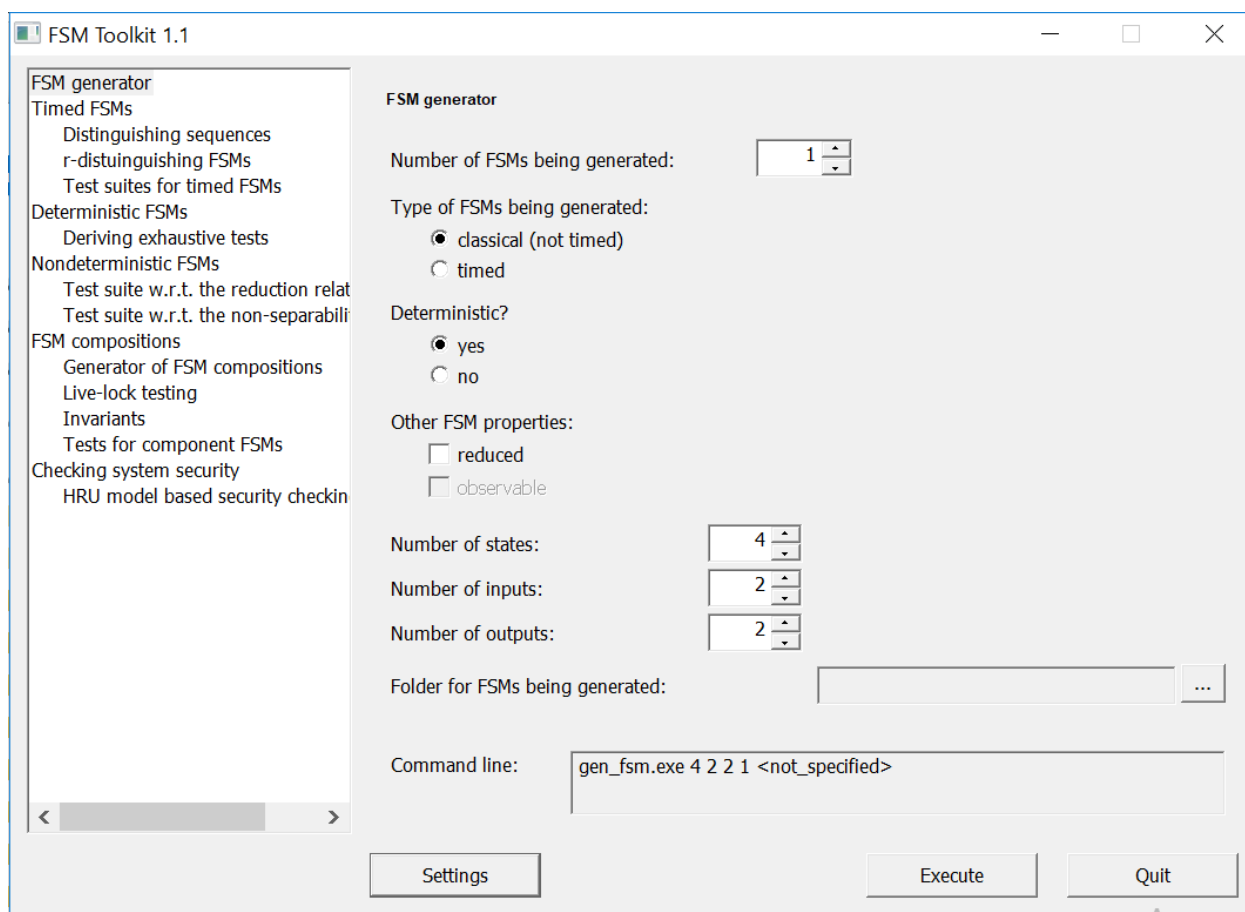


Рисунок 3 – Программа «FSMTest-1.0»

## 2.2 Структура программы «FSMTest-2.0»

В данном разделе мы рассмотрим части программы, а также их назначение.

На рисунке 4 изображена схема взаимодействия элементов программы. Данные из XML-файла поступают на парсер, где происходит проверка XML-документов. Если документ написан без ошибок, то он добавляется в список корректных документов, иначе игнорируется, и ошибки, допущенные при написании, выводятся в лог-файл.

Далее в ядре создаётся дерево вкладок, основанное на расположении XML-документов в выбранной директории. При формировании дерева, ядро посылает в парсер запрос о корректности XML-документа, и если в ответ получает «true», то вкладка добавляется в дерево, иначе XML-документ игнорируется. Когда дерево сформировано, происходит разбор соответствующих XML-документов и на основе данных, предоставленных в них, создаётся графический интерфейс для данной вкладки.



Рисунок 4 – Схема взаимосвязи элементов программы

В созданном окне выставляются необходимые параметры, после чего формируется поток данных, который передаётся обратно в ядро. Из ядра данные передаются в .exe файл, который служит API программы.

2.3 Разработка алгоритма разбора дерева, позволяющего преобразовывать данные в формат, используемый ядром «FSMTest-2.0»

### 2.3.1 Существующие библиотеки реализации парсеров

Для написания парсера необходимо сначала произвести разбор XML-документа. Разработчикам программного обеспечения очень часто приходится работать с XML-документами, для этого в Java существует несколько библиотек для работы с ними.

Объектная модель документа (DOM) [15] является официальной рекомендацией World Wide Web Consortium (W3C). Когда XML-парсер поддерживает DOM, это означает, что он реализует интерфейсы, определённые в данном стандарте. При разборе XML-документа получается структура дерева, которая представляет содержимое XML-документа.

Достоинства библиотеки DOM:

- а) сохранение в удобную древовидную структуру;
- б) лёгкий способ загрузки (создаётся экземпляр класса «Document», с которым впоследствии идёт работа);
- в) при внесении изменений в документ не надо переписывать парсер.

Недостатки библиотеки DOM:

- а) работает медленнее в сравнении с SAX;
- б) значения атрибутов могут быть записаны только в виде строки.

Simple API for XML (SAX) [15] является альтернативой для работы с XML-документами. Он разработан с меньшими требованиями к объёму памяти, но он перекладывает больше работы на программиста.

Достоинства библиотеки SAX:

- а) в базу сохраняются только те данные, которые нужны для работы;
- б) работает быстрее в сравнении с другими методами;
- в) удобный метод для проверки синтаксических ошибок в документе.

Недостатки библиотеки SAX:

- а) реализацию загрузки документа нужно писать программисту;
- б) отсутствие стандартных методов для добавления новых элементов.

JDOM [16] является инструментом для работы с XML-документами; создан для обеспечения быстрой разработки XML-приложений. JDOM, так же как и DOM,



строит дерево объектов, представляющее XML-документ, но предоставляет более удобные методы работы с данными.

Достоинства библиотеки JDOM:

- а) сохранение в удобную древовидную структуру;
- б) лёгкий способ загрузки (создаётся экземпляр класса «Document», с которым впоследствии идёт работа);
- в) при внесении изменений в документ не надо переписывать парсер;
- г) удобные методы для работы с элементами дерева.

Недостатки библиотеки JDOM:

- а) работает медленнее в сравнении с SAX, т.к. сохраняется вся структура;
- б) значения атрибутов могут быть записаны только в виде строки.

В связи с этим для проверки синтаксиса XML-документа решено использовать библиотеку SAX, а для разбора XML-документа – библиотеку JDOM.

### 2.3.2 Структура и свойства XML-документа

Свойства XML:

1. Язык с простым синтаксисом.
2. Удобен для обработки, т.к. существуют методы разбора.
3. Возможно создание нужного формата, в соответствии с требованиями задачи.
4. Широкое использование в разнообразных сферах.
5. Низкий порог вхождения. Это означает, что для начала работы с данным языком программирования не требуется его долгое изучение.

В таблице 1 приведён список узлов, соответствующих элементам интерфейса. Каждому элементу интерфейса соответствует узел, с определённым именем. У каждого узла существуют атрибуты и потомки, приведённые в таблице.

Таблица 1 – Описание элементов интерфейса в формате XML

| Элемент интерфейса    | Имя узла    | Имя атрибута  | Значение атрибута                       | Потомки      |
|-----------------------|-------------|---------------|-----------------------------------------|--------------|
| Check button          | flag        | Name          | Имя элемента                            | Нет потомков |
|                       |             | Var           | Переменная, в которой хранится значение |              |
|                       |             | Checked       | Значение, если элемент выбран           |              |
|                       |             | not_checked   | Значение, если элемент не выбран        |              |
| Radio button          | select      | Name          | Имя элемента                            | variant      |
|                       |             | Var           | Переменная, в которой хранится значение |              |
| Select directory/file | dirselector | Name          | Имя элемента                            | Нет потомков |
|                       |             | Var           | Переменная, в которой хранится значение |              |
|                       |             | Type          | Директория или файл                     |              |
| Spinbox               | range       | Name          | Имя элемента                            | min          |
|                       |             | Var           | Переменная, в которой хранится значение | max          |
| Const value           | const       | Name          | Имя элемента                            | Нет потомков |
|                       |             | Var           | Переменная, в которой хранится значение |              |
|                       |             | Val           | Значение константы                      |              |
| Command lines         | commands    | Нет атрибутов | Нет значений                            | commandline  |

Окончание таблицы 1

|                   |           |               |                            |              |
|-------------------|-----------|---------------|----------------------------|--------------|
|                   |           |               |                            | Command      |
|                   |           |               |                            | param        |
| Condition         | condition | var1          | Значение первой переменной | Нет потомков |
|                   |           | Symbol        | Знак сравнения             |              |
|                   |           | var2          | Значение второго элемента  |              |
| Blocking elements | if        | Нет атрибутов | Нет значений               | conditions   |
|                   |           |               |                            | condition    |
|                   |           |               |                            | then         |
|                   |           |               |                            | else         |
|                   |           |               |                            | enabled      |
|                   |           |               |                            | not_enabled  |
|                   |           |               |                            | param        |

В таблице 2 приведено описание потомков, которые могут иметь узлы, указанные в первой таблице.

Таблица 2 – Описание дочерних узлов главных элементов

| Имя узла | Имя атрибута | Значение атрибута                      | Свойства                                                                       |
|----------|--------------|----------------------------------------|--------------------------------------------------------------------------------|
| variant  | name         | Имя варианта                           | При выборе одного из вариантов Radio button принимает соответствующее значение |
|          | val          | Значение, которое принимает переменная |                                                                                |
| min      | val          | Ограничивающее значение                | Минимальное значение, которое может принять Spinbox                            |

Окончание таблицы 2

| max         | val    | Ограничивающее значение                       | Максимальное значение, которое может принять Spinbox (не обязательно)          |
|-------------|--------|-----------------------------------------------|--------------------------------------------------------------------------------|
| Conditions  | Нет    | Нет                                           | Хранит узлы с условиями                                                        |
| condition   | var1   | Первое значение                               | Условие, воздействующее на сопротивление                                       |
|             | symbol | Символ сравнения                              |                                                                                |
|             | var2   | Второе значение                               |                                                                                |
| then        | Нет    | Нет                                           | Действия при выполнении условий                                                |
| else        | Нет    | Нет                                           | Действия при невыполнении условий                                              |
| enabled     | Нет    | Нет                                           | Незатемненные элементы                                                         |
| not_enabled | Нет    | Нет                                           | Затемнённые элементы                                                           |
| param       | val    | Переменная                                    | Выделяет элементы, на которые будет оказано воздействие                        |
| commandline | Нет    | Нет                                           | Содержит узлы, которые определяют значения, входящие в данную командную строку |
| command     | val    | Имя .exe файла, в который передаются значения | Определяет .exe файл, который будет запущен                                    |

На рисунке 5 приведён фрагмент правильно оформленного XML-документа. На рисунке 6 представлено дерево преемников, построенное по этому документу. Элемент «root» соответствует корню дерева. Элементы «range» и «select» соответствуют узлам первого яруса дерева и образуют поддеревья, в узлах которых хранится описание характеристик этих элементов. Например, в поддереве с корнем «range» узлы первого яруса «name» и «var» соответствуют атрибутам данного элемента и описывают его имя и переменную, в которой хранятся значения, а у узлов «min» и «max» есть свои потомки (атрибуты), которые описывают минимальное и максимальное значение счётчика.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <range name="Spinbox name" var="var 1">
    <min val="val 1"/>
    <max val="val 2"/>
  </range>
  <select name="Radio button" var="var 2">
    <variant name="true" val="1"/>
    <variant name="false" val="0"/>
  </select>
</root>
```

Рисунок 5 – Фрагмент XML-документа

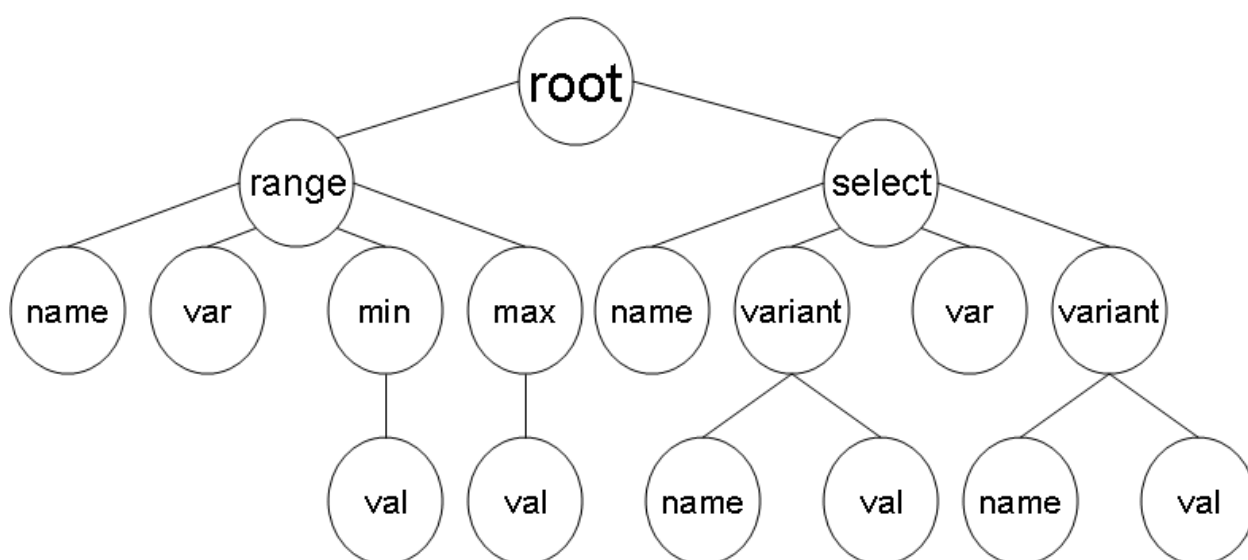


Рисунок 6 – Дерево преемников, соответствующее документу на рисунке 2

### 2.3.3 Алгоритм проверки XML-документа

На вход алгоритма поступает путь к директории, в которой находятся XML-файлы, и документ, используемый для проверки документов.

На выходе алгоритма получаем список корректных файлов, и список ошибок.

Шаг 1. Производим обход содержимого директории, и если элемент, путь к которому мы получаем, не является папкой, то шаг 2, иначе выбираем следующий элемент.

Шаг 2. При помощи библиотеки SAX проверяем документ на наличие синтаксических ошибок. Если ошибок не обнаружено – переходим на шаг 3, иначе добавляем ошибку в массив строк, который хранит ошибки, и переходим на шаг 1.

Шаг 3. По документу строится дерево его объектов и происходит обход узлов первого яруса. Если имя узла не соответствует списку возможных документов, то ошибка записывается в список и шаг 4. Если все узлы пройдены, и ошибок не обнаружено, то путь к файлу записывается в список корректных файлов и шаг 1.

Шаг 4. На вход приходит узел первого яруса, который является корнем поддерева, описывающего элемент. Проверяются все потомки данного поддерева, если каких-то узлов не хватает, или существуют лишние, то ошибка добавляется в список. Если все узлы пройдены, то переходим на шаг 3.

Формат вывода ошибок следующий: путь к файлу, имя узла, в котором допущена ошибка, а также тип ошибки.

### 2.3.4 Алгоритм разбора дерева парсинга XML-документа

На вход алгоритма поступает путь к документу.

На выходе алгоритма получаем поток данных, по которому строится окно реализации.

Шаг 1. По принятому документу строится дерево объектов.

Шаг 2. Происходит обход узлов первого яруса, каждый из которых является корнем поддерева, описывающего определённый элемент интерфейса. Для каждого элемента в зависимости от его свойств разработан свой метод разбора. Определив,

какой элемент создаётся, переходим на шаг 3. После того, как все узлы будут пройдены – программа закончит работу.

Шаг 3. Разбираем поддерево и по данным, хранящимся в его узлах, формируем поток данных. Переходим на шаг 2.

### 2.3.5 Методы проверки XML-документа

В данном разделе приводится краткое описание методов, написанных для проверки документа.

`public void processFilesFromFolder(File folder).` На вход данного метода поступает путь к директории, в которой хранятся документы. В данном методе происходит обход содержимого директории. Метод ничего не возвращает.

`boolean checkNames(Element root_element, File file).` На вход данного метода поступает корень дерева и путь к рассматриваемому документу. В данном методе происходит выбор узлов, являющихся корнями поддеревьев, и отправка данных узлов в следующий метод. Метод возвращает «true», если при написании документа не было допущено ошибок, иначе «false».

`boolean checkCurrentNode(Node main_node, File file).` На вход подаётся корень поддерева, описывающего элемент интерфейса и путь к файлу. В данном методе происходит проверка узлов первого яруса, и если имя узла соответствует формату, то происходит переход к методу, проверяющему узлы данного поддерева, иначе записывается ошибка. Метод возвращает «true», если при написании документа не было допущено ошибок, иначе «false».

`boolean checkPropertiesNode(Node main_node, Node checking_node, File file).` На вход данного метода поступает корень поддерева, его потомок, который будет проверяться, а так же путь к файлу. В данном методе проверяются узлы поддерева на соответствие формату. Если существуют узлы, которых быть не должно, или наоборот, каких-то узлов не хватает, то список ошибок пополняется. По окончании обхода, если были допущены ошибки, метод возвращает «false», иначе, если всё корректно, метод возвращает «true». Метод вызывается рекурсивно, пока не обойдёт всё поддерево.

`public File getFile(int i).` На вход поступает номер элемента. Метод возвращает файл с данным номером.

`public ArrayList < String > getErrorList()`. На вход ничего не подаётся. Метод возвращает список ошибок.

`public ArrayList < File > getFiles()`. На вход ничего не подаётся. Метод возвращает список корректных файлов.

`public boolean hasFile(File file)`. На вход поступает файл. Если файл корректный, то метод возвращает «true», иначе «false».

В приложении А приведён код парсера.

В приложении Б приведён код шаблонного XML-документа, по которому осуществляется семантическая проверка XML-документов.

### 2.3.6 Методы разбора XML-документа

В данном разделе приводится краткое описание методов, написанных для разбора документа.

`void ParseElements(File file)`. На вход поступает путь к файлу. В данном методе документ разбирается и приводится к древовидной структуре, а так же рассчитывается количество выделяемой памяти. Метод ничего не возвращает.

`void parsingMainNodes(NodeList list_of_main_childs)`. На вход поступает список корней поддеревьев, описывающих элементы интерфейса. В данном методе осуществляется проход по этому списку, и, в зависимости от имени корня, определяются, каким методом будет идти разбор. Метод ничего не возвращает.

`void parsingImageNodes(Node current_node)`. На вход принимается корень текущего поддерева. В данном методе происходит разбор элемента «Картинка». Метод ничего не возвращает.

`void parsingConstNodes(Node current_node)`. На вход принимается корень текущего поддерева. В данном методе происходит разбор элемента «Константа». Метод ничего не возвращает.

`void parsingConditionNodes(NodeList list_of_current_childs, NodeList list_of_main_childs)`. На вход принимается список потомков корня текущего поддерева, список всех корней поддеревьев. В данном методе происходит разбор элемента «Условие». Метод ничего не возвращает.

`void parsingIfNodes(NodeList list_childs_of_if, NodeList list_of_main_childs)`. На вход принимается список узлов поддерева, представляющего элемент



«Затемнение», список корней поддеревьев. В данном методе происходит разбор элемента «Затемнение». Метод ничего не возвращает.

`void conditionChildsOfIf(NodeList list_of_current_childs, NodeList list_of_main_childs).` На вход подаётся список узлов, описывающих условие для затемнения и список корней поддеревьев. Метод составляет условие для затемнения. Метод ничего не возвращает.

`void otherChildOfIf(NodeList list_of_current_childs, NodeList list_of_main_childs).` На вход подаётся список потомков поддерева, которое описывает затемнение, список всех корней поддеревьев. Метод записывает затемняемые элементы. Метод ничего не возвращает.

`void parsingRangeNode(Node current_node, NodeList list_of_current_childs).` На вход принимается корень текущего поддерева, описывающего элемент, список его потомков. Метод записывает поток данных для элемента «Spinbox». Метод ничего не возвращает.

`void parsingSelecteNode(Node current_node, NodeList list_of_current_childs).` На вход подаётся корень текущего поддерева, список его потомков. Метод записывает поток данных для элемента «Radion button». Метод ничего не возвращает.

`void parsingFlagNode(Node current_node).` На вход поступает корень текущего поддерева. Метод записывает поток данных для элемента «Check button». Метод ничего не возвращает.

`void parsingDirselectorNode(Node current_node).` На вход поступает корень текущего поддерева. Метод записывает поток данных для элемента «Check button». Метод ничего не возвращает.

`void parsingCommandsNode(NodeList list_of_current_nodes, NodeList list_of_main_nodes)` – на вход подаётся список потомков корня текущего поддерева, список всех корней поддеревьев. Данный метод заполняется поток данных для командных строк. Метод ничего не возвращает.

`void addElement().` На вход ничего не подаётся. Данный метод добавляет элемент, описанный данными текущего поддерева. Метод ничего не возвращает.

### 2.3.7 Новые реализации

В пакет «FSMTest-2.0» были добавлены реализации, не входящие в пакет «FSMTest-1.0».

Приложение «Synch\_Seq» предназначено для синтеза синхронизирующих последовательностей для конечных автоматов в формате .fsm. На рисунке 7 представлена вкладка, созданная для приложения «Synch\_Seq» по XML-документу, код которого представлен ниже:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<tool name = "InterFSM">
  <dirselector name = "Path to the file with FSM" var = "dir1" type = "file"/>
  <const name = "three" var = "const1" val = "3"/>
  <commands>
    <commandline>
      <command val = "\Synch_Seq.exe"/>
      <param val = "const1"/>
      <param val = "dir1"/>
    </commandline>
  </commands>
</tool>
```

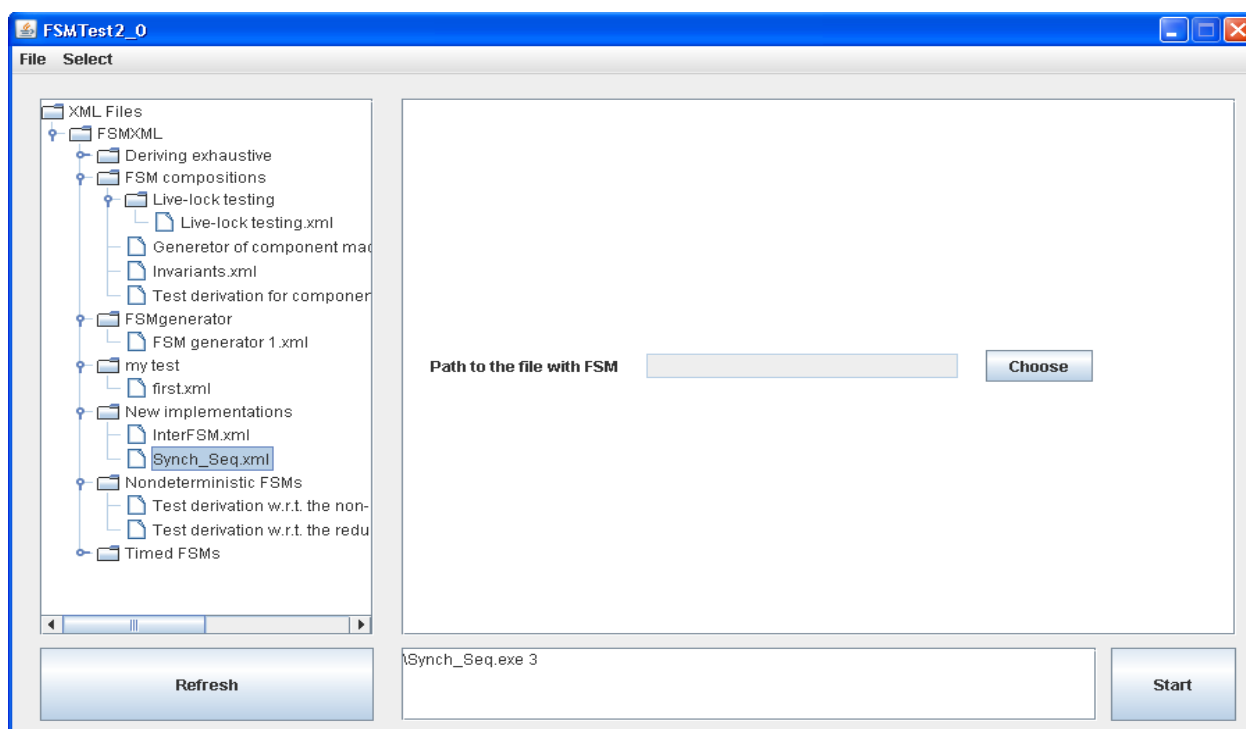


Рисунок 7 – Реализация для приложения «Synch\_Seq»

Входные параметры: путь к файлу в формате .fsm.

На выходе получаем .txt файл, в который записаны все безызыточные синхронизирующие последовательности и все найденные для данного автомата синхронизирующие последовательности.

Программа «InterFSM» строит пересечение пары полностью определённых, возможно недетерминированных, конечных автоматов, находящихся в произвольных состояниях. На рисунке 8 представлена вкладка, созданная для программы «InterFSM» по XML- описанию, код которого представлен ниже:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<tool name = "InterFSM">
  <dirselector name = "Path to the file with first FSM (not timed)" var = "dir1" type
= "file"/>
  <dirselector name = "Path to the file with second FSM (not timed)" var = "dir2"
type = "file"/>
    <range name = "Initial state of the first FSM" var = "first_state">
      <min val = "0"/>
      <max val = "100"/>
    </range>
    <range name = "Initial state of the second FSM" var = "two_state">
      <min val = "0"/>
      <max val = "100"/>
    </range>
  <dirselector name = "Folder for result" var = "dir3" type = "dir"/>
  <commands>
    <commandline>
      <command val = "\InterFSM.exe"/>
      <param val = "dir1"/>
      <param val = "dir2"/>
      <param val = "first_state"/>
      <param val = "two_state"/>
      <param val = "dir3"/>
    </commandline>
  </commands>
</tool>
```

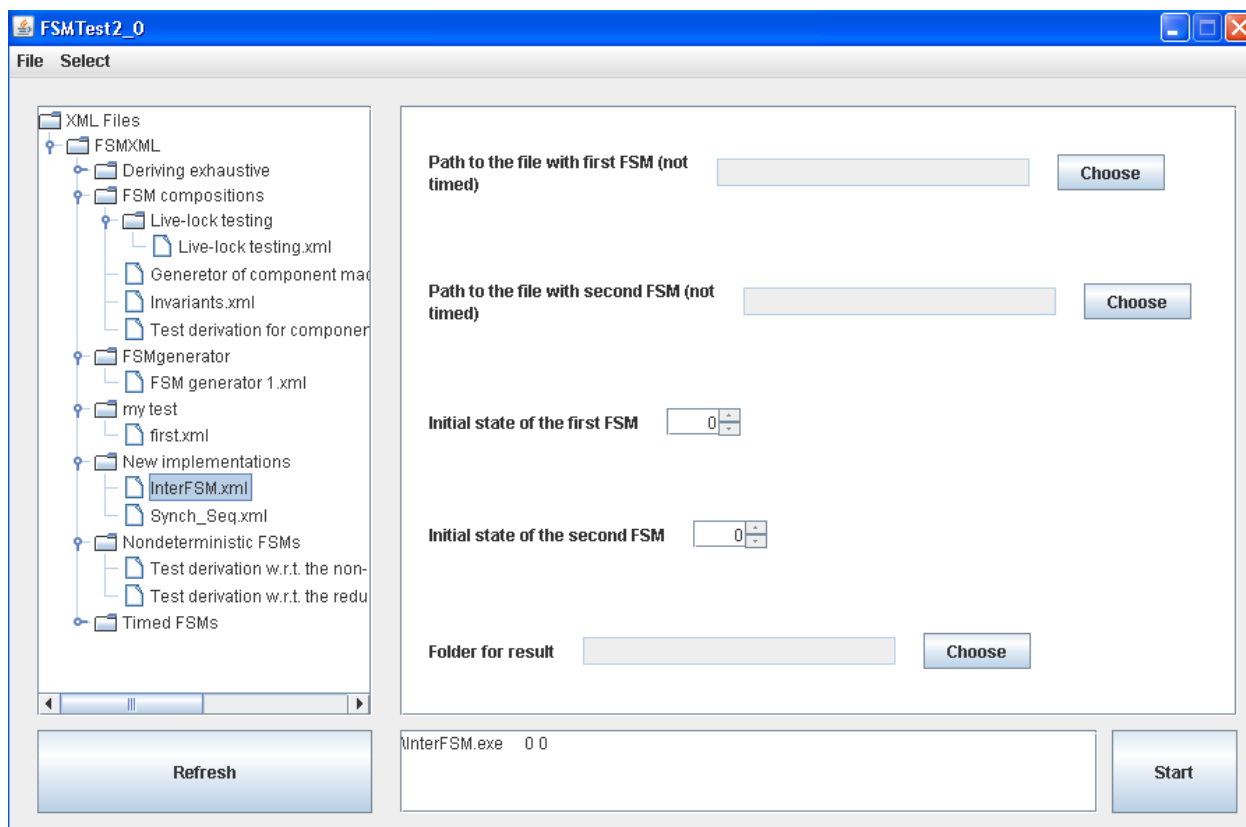


Рисунок 8 – Вкладка для программы «InterFSM»

Входные данные:

1. Путь к файлу в формате .fsm, содержащему первый конечный автомат.
2. Путь к файлу в формате .fsm, содержащему второй конечный автомат.
3. Целое число, отражающее состояние первого автомата, которое будет принято за начальное.
4. Целое число, отражающее состояние второго автомата, которое будет принято за начальное.
5. Путь к директории, в которую будет помещён результат.

Возможные выходные реакции:

- а) true – пересечение было построено;
- б) false – недостаточно аргументов.

### 2.3.8 Ошибки, недоработки и их исправление

В данном разделе мы рассмотрим ошибки, найденные в ядре и XML-документах, и способы их устранения.

При построении дерева вкладок, изображённого на рисунке 9, отсутствовала семантическая проверка, поскольку вместо парсера, написанного для отбора XML-документов, пригодных для программы «FSMTest-2.0», использовалась обычная проверка синтаксиса, предоставленная стандартной библиотекой SAX [15]. Вследствие этого программа обращалась к некорректным XML-документам, и происходило нарушение работы программы или создавались пустые/некорректные вкладки.

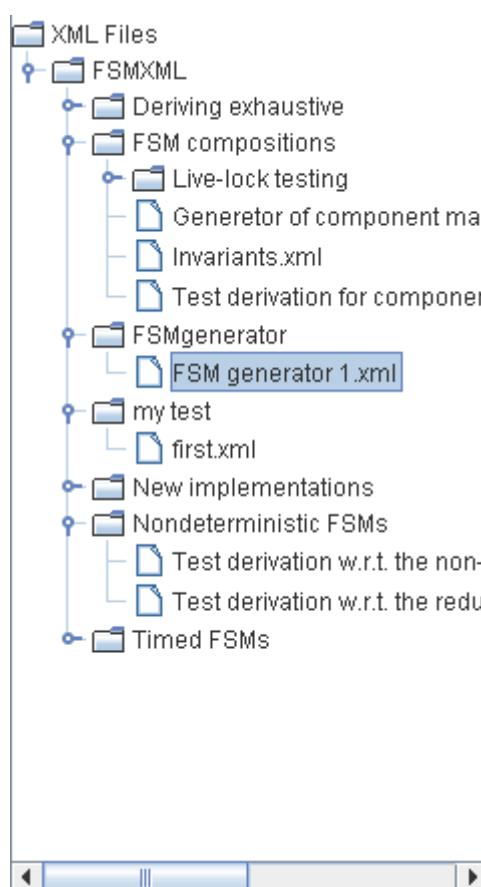


Рисунок 9 – Дерево вкладок

Ошибка была исправлена путём переписывания класса ядра, который строит дерево вкладок, а именно, добавление проверки, осуществляющейся при помощи класса «XMLParserFSM».

В программе отсутствовал вывод ошибок, допущенных в XML-файлах. Проблема была исправлена следующим образом:

1. При нажатии кнопки «Refresh», изображённой на рисунке 10, создаётся экземпляр класса «XMLParserFSM».

2. Вызывается метод «processFilesFromFolder(File folder)».
3. Создаётся экземпляр класса «PrintWriter». *PrintWriter* – класс для вывода данных в файл.
4. Вызывается метод «getErrorList», который возвращает строковый список, который содержит все ошибки, содержащиеся в XML-документах.



Рисунок 10 – Кнопка обновления вкладок реализаций

Программа не отработывала в тех случаях, когда в реализации присутствует несколько командных строк, как, например, на рисунке 11.

|                                                                       |                                                                   |                                                                      |
|-----------------------------------------------------------------------|-------------------------------------------------------------------|----------------------------------------------------------------------|
| Component FSM specification                                           | <input type="text"/>                                              | <input type="button" value="Choose"/>                                |
| Component FSM specification                                           | <input type="text"/>                                              | <input type="button" value="Choose"/>                                |
| Faulty component (1 of 2)                                             | <input checked="" type="radio"/> one<br><input type="radio"/> two |                                                                      |
| % of transitions for fault injecting                                  | <input type="text" value="4"/>                                    | <input type="button" value="↑"/><br><input type="button" value="↓"/> |
| Nondeterminism range of faulty transitions in the mutation FSM(lower) | <input type="text" value="2"/>                                    | <input type="button" value="↑"/><br><input type="button" value="↓"/> |
| Nondeterminism range of faulty transitions in the mutation FSM(upper) | <input type="text" value="3"/>                                    | <input type="button" value="↑"/><br><input type="button" value="↓"/> |
| Specification                                                         | <input type="text"/>                                              | <input type="button" value="Choose"/>                                |

|                                                                                                                                                                                    |                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|
| <pre> \Compose_Spec_MM.exe tmp.tmp tmp1.tmp \Partial_from_Complete.exe tmp2.tmp 1 4 \DeriveObservableMM.exe tmp2.tmp tmp3.tmp 1 2 3 \Compose_Spec_MM.exe tmp2.tmp           </pre> | <input type="button" value="Start"/> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|

Рисунок 11 – Реализация, в которой присутствует несколько командных строк

Изначально полагалось, что ошибка вызвана приложением «FSMTest-2.0», но как выяснилось позже, проблема была в некорректно написанных XML-документах. Код некорректного XML-документа, отображённого на рисунке 11, представлен ниже, в скобках указаны допущенные ошибки.

```
<?xml version="1.0" encoding="UTF-8"?>
<tool name="Test derivation for component FSMs">
  <image url="\image.gif"/>
  <dirselector name="Component FSM specification" var="destdir" type="file"/>
  <dirselector name="Component FSM specification" var="destdir1" type="file"/>
  <select name="Faulty component (1 of 2)" var="type">
    <variant name="one" val="1"/>
    <variant name="two" val="2"/>
  </select>
  <range name="% of transitions for fault injecting" var="transitions">
    <min val="1"/>
    <max val="100"/>
  </range>
  <range name="Nondeterminism range of faulty transitions in the mutation
FSM(lower)" var="number">
    <min val="1"/>
    <max val="100"/>
  </range>
  <range name="Nondeterminism range of faulty transitions in the mutation
FSM(upper)" var="number1">
    <min val="1"/>
    <max val="100"/>
  </range>
  <const name="tmp" var="tmp" val="tmp.tmp"/>
  <const name="tmp1" var="tmp1" val="tmp1.tmp"/>
  <const name="tmp2" var="tmp2" val="tmp2.tmp"/>
  <const name="tmp3" var="tmp3" val="tmp3.tmp"/>
  <const name="tmp4" var="tmp4" val="tmp4.tmp"/>
  <const name="tmp5" var="tmp5" val="tmp5.tmp"/>
  <conditions>
    <condition var1="number" symbol="#x3C" var2="number1"/>
  </conditions>
  <dirselector name="Specification" var="destdir2" type="dir"/>
  <commands>
    <commandline>
```

```

        <command val="\COMpose_Spec_MM.exe"/>
        <param val="destdir"/>
        <param val="destdir1"/>
        <param val="tmp"/>
        <param val="tmp1"/>
    </commandline>
    <commandline>
        <command val="\Pertial_from_Complete.exe"/> (ошибка в имени
.exe файла, правильное имя «Partial_from_Complete.exe»)
        <param val="destdir"/>
        <param val="tmp2"/>
        <param val="type"/>
        <param val="transitions"/>
    </commandline>
</commandline>
    <commandline>
        <command val="\DeriveObservableMM.exe"/>
        <param val="tmp2"/>
        <param val="tmp3"/>
        <param val="type"/>
        <param val="number"/>
        <param val="number1"/>
    </commandline>
</commandline>
    <commandline>
        <command val="\COMpose_Spec_MM.exe"/>
        <param val="destdir"/>
        <param val="tmp3"/>
    </commandline>
</commandline>
    <commandline>
        <command val="\test_fsm.exe"/>
        <param val="tmp"/>
        <param val="tmp4"/>
    </commandline>
    <commandline>
        <command val="\TestConverter.exe"/>
        <param val="tm4"/> (опечатка в значении параметра)
        <param val="tm1"/> (опечатка в значении параметра)
        <param val="destdir2"/>
    </commandline>
</commands>
</tool>

```



### 2.3.9 Проверка работы программы «FSMTest-2.0»

В данной главе показана работа программы на примере разработанного пакета. Описание XML-документов, по которым строятся окна интерфейса, находятся в приложении В.

На рисунке 12 изображено главное окно программы со сгенерированными вкладками.

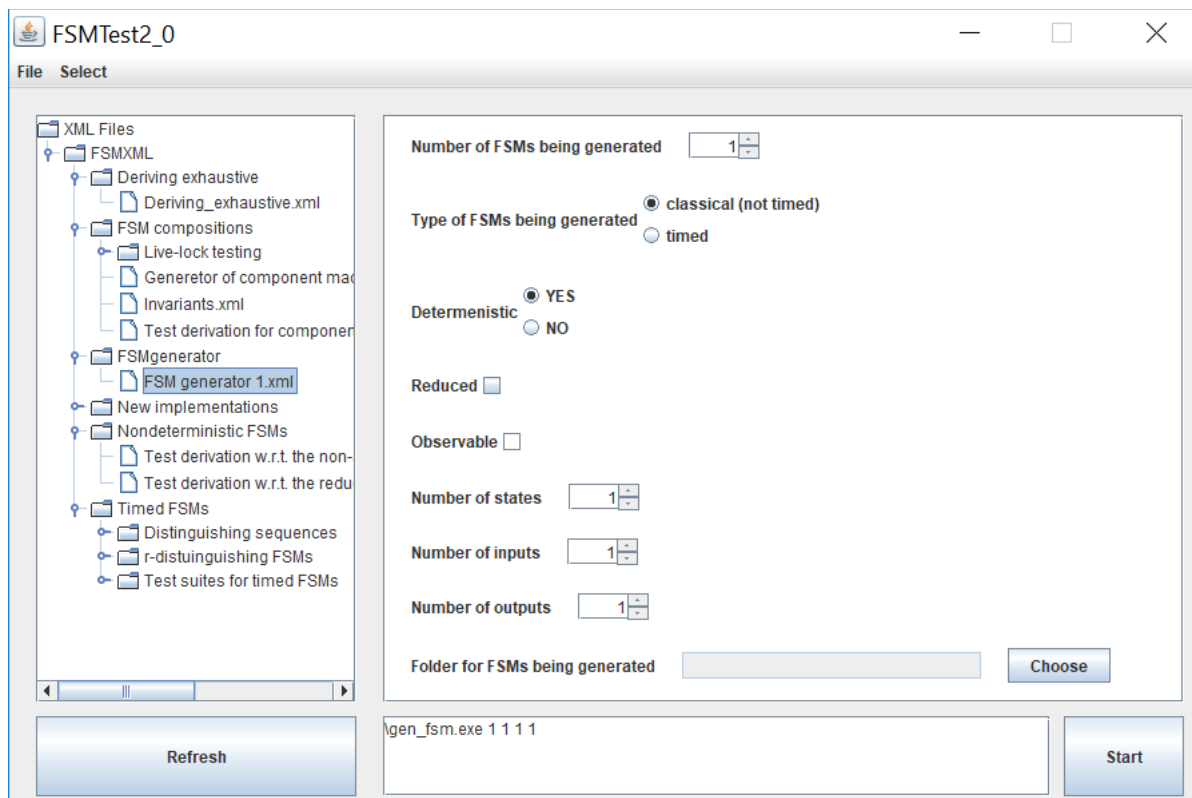


Рисунок 12 – Окно программы

На рисунке 13 изображена левая часть программы, сверху находится список вкладок с реализациями, снизу кнопка «Refresh», предназначенная для обновления вкладок.

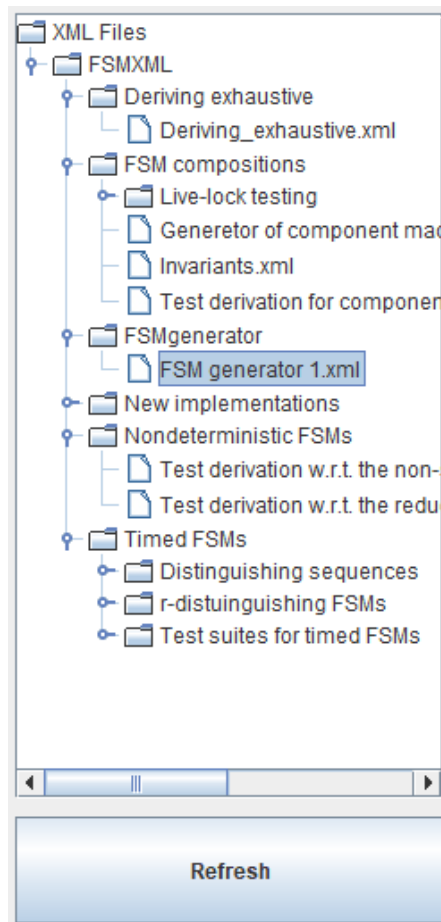


Рисунок 13 – Вкладки и кнопка обновления

На рисунке 14 изображена правая часть программы. Сверху находится сгенерированный интерфейс реализации. В данном окне присутствуют элементы: spinbox, check button, radio-button и выбор пути к директории. Снизу изображены командная строка, в которую выводится информация, которая будет передаваться в исполняемый файл, и кнопка «Start», которая запускает .exe файл с заданными значениями.

The screenshot shows a window for generating FSMs. It contains several configuration options:

- Number of FSMs being generated:** A spinner box set to 6.
- Type of FSMs being generated:** Radio buttons for "classical (not timed)" and "timed". The "timed" option is selected.
- Deterministic:** Radio buttons for "YES" and "NO". The "NO" option is selected.
- Reduced:** A checked checkbox.
- Observable:** An unchecked checkbox.
- Number of states:** A spinner box set to 4.
- Number of inputs:** A spinner box set to 3.
- Number of outputs:** A spinner box set to 7.
- Folder for FSMs being generated:** A text input field followed by a "Choose" button.

At the bottom, there is a command line area with the text: `ngen_fsm.exe -T -ND 4 3 7 6` and a "Start" button.

Рисунок 14 – Сгенерированное окно, командная строка и запуск

Ошибки выводятся в файл с названием «log errors», ниже представлен список полученных ошибок. Формат ошибок следующий. Если ошибка синтаксическая, выводится путь к XML-документу, номер строки и номер столбца, в которых находится ошибка, а так же тип ошибки. Если ошибка в имени узла или атрибута, то выводятся имя узла, в котором совершена ошибка, путь к XML-документу и тип ошибки.

Ниже приведены все найденные парсером ошибки:

- information about error : org.xml.sax.SAXParseException; systemId: file:/C:/Documents and Settings/User/workspace/FSMTest2\_0/FSMXML/Deriving exhaustive/Deriving\_exhaustive.xml; lineNumber: 52; columnNumber: 17; Open quote is expected for attribute "val" associated with an element type "command";
- information about error : org.xml.sax.SAXParseException; systemId: file:/C:/Documents and Settings/User/workspace/FSMTest2\_0/FSMXML/FSM

compositions/Generetor of component machines for the composition of two FSMs.xml;  
lineNumber: 105; columnNumber: 3; The element type "range" must be terminated by  
the matching end-tag "</range>";

- information about error : org.xml.sax.SAXParseException; systemId:  
file:/C:/Documents and Settings/User/workspace/FSMTest2\_0/FSMXML/FSM  
compositions/Invariants.xml; lineNumber: 21; columnNumber: 37; The value of attribute  
"symbol" associated with an element type "condition" must not contain the '<' character;

- information about error : org.xml.sax.SAXParseException; systemId:  
file:/C:/Documents and Settings/User/workspace/FSMTest2\_0/FSMXML/FSM  
compositions/Live-lock testing/Live-lock testing.xml; lineNumber: 10; columnNumber:  
5; The value of attribute "val" associated with an element type "command" must not  
contain the '<' character.

- information about error : org.xml.sax.SAXParseException; systemId:  
file:/C:/Documents and Settings/User/workspace/FSMTest2\_0/FSMXML/FSM  
compositions/Test derivation for component FSMs.xml; lineNumber: 30;  
columnNumber: 37; The value of attribute "symbol" associated with an element type  
"condition" must not contain the '<' character.

- error:

- node: dirselector

- path: FSMXML\Nondeterministic FSMs\Test derivation w.r.t. the non-separability  
relation.xml

- type: in node dirselector, not found attribute: type

- error:

- node: dirselector

- path: FSMXML\Nondeterministic FSMs\Test derivation w.r.t. the non-separability  
relation.xml

- type: in node dirselector, not found attribute: type

- error:

- node: dirselector

- path: FSMXML\Nondeterministic FSMs\Test derivation w.r.t. the reduction  
relation.xml

- type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Nondeterministic FSMs\Test derivation w.r.t. the reduction relation.xml

type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Timed FSMs\Deriving sequebces\Deriving sequebces.xml

type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Timed FSMs\Deriving sequebces\Deriving sequebces.xml

type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Timed FSMs\r-distuinguishing FSMs\r-distuinguishing FSMs.xml

type: in node dirselector, not found attribute: var

- error:

node: dirselector

path: FSMXML\Timed FSMs\r-distuinguishing FSMs\r-distuinguishing FSMs.xml

type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Timed FSMs\r-distuinguishing FSMs\r-distuinguishing FSMs.xml

type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Timed FSMs\r-distuinguishing FSMs\r-distuinguishing FSMs.xml

type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Timed FSMs\Test suites for timed FSMs\Test suites for timed FSMs.xml

type: in node dirselector, not found attribute: type

- error:

node: dirselector

path: FSMXML\Timed FSMs\Test suites for timed FSMs\Test suites for timed FSMs.xml

type: in node dirselector, not found attribute: type

### 2.3.10 Проверка работы новых реализаций

В данном разделе показана работа программы «FSMTest-2.0» на примере дополнительных реализаций, не вошедших в программу «FSMTest-1.0».

На рисунке 15 изображено окно программы «Synch\_Seq», с выставленными входными значениями. Данная программа позволяет строить все безызыточные синхронизирующие последовательности конечного автомата.

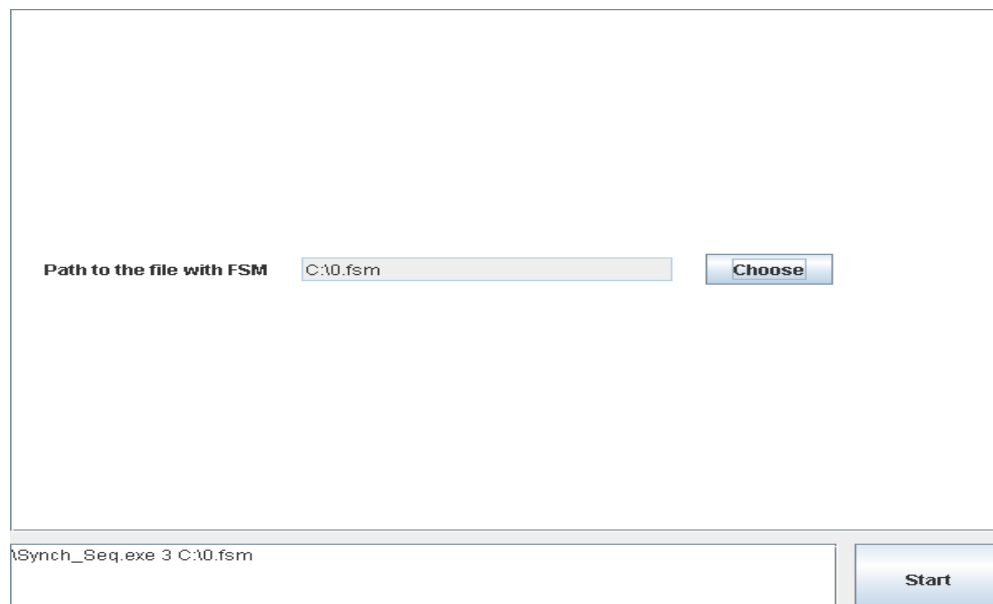


Рисунок 15 – Входные параметры для тестирования приложения «Synch\_Seq»

Содержимое файла «0.fsm»:

F 0

s 10  
i 5  
o 3  
n 0 0  
p 50  
0 0 2 1  
0 1 0 2  
0 2 3 0  
0 3 9 2  
0 4 2 1  
1 0 4 0  
1 1 3 2  
1 2 2 2  
1 3 9 1  
1 4 9 0  
2 0 1 0  
2 1 3 0  
2 2 0 0  
2 3 7 0  
2 4 5 2  
3 0 3 0  
3 1 6 2  
3 2 6 0  
3 3 9 0  
3 4 2 1  
4 0 5 0  
4 1 7 2  
4 2 1 1  
4 3 8 2  
4 4 4 2  
5 0 3 1  
5 1 3 0  
5 2 6 2  
5 3 8 2  
5 4 3 2  
6 0 9 0  
6 1 9 1  
6 2 2 1  
6 3 6 0  
6 4 0 1  
7 0 1 0  
7 1 0 2  
7 2 1 1  
7 3 5 0

7 4 0 2  
8 0 2 0  
8 1 8 2  
8 2 5 2  
8 3 2 2  
8 4 0 2  
9 0 1 1  
9 1 7 1  
9 2 5 0  
9 3 8 2  
9 4 1 2

Результаты работы программы, которыми являются все безызбыточные синхронизирующие последовательности и все найденные для данного автомата синхронизирующие последовательности, представлены ниже.

fin state: 3  
length: 5  
set of inputs: 0 0 0 0 0

fin state: 1  
length: 16  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0

fin state: 1  
length: 19  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 0

fin state: 3  
length: 21  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 1

fin state: 9  
length: 23  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 1 3

fin state: 2  
length: 23  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 1 4

fin state: 3  
length: 23  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 1



fin state: 2  
length: 25  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 1 0

fin state: 3  
length: 25  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 1

fin state: 5  
length: 26  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 3 2

fin state: 0  
length: 29  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 3 3 1 1 4

fin state: 1  
length: 28  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 3 3 3 0

fin state: 9  
length: 29  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 3 3 3 2 3

fin state: 7  
length: 31  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 3 3 3 3 2 0 1

fin state: 8  
length: 31  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 3 3 3 3 2 0 3

fin state: 2  
length: 30  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 2 3 3 3 3 2 2

fin state: 0  
length: 24  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 2 4 3 4

fin state: 8  
length: 22  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 0 4 3

fin state: 0  
length: 20  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 1 1 0 2 0 3 1 1

fin state: 8  
length: 13  
set of inputs: 0 0 0 0 1 0 0 0 1 0 0 0 3

fin state: 0  
length: 10  
set of inputs: 0 0 0 0 1 0 0 0 1 4

fin state: 9  
length: 8  
set of inputs: 0 0 0 0 1 0 0 3

fin state: 6  
length: 5  
set of inputs: 0 0 0 0 2

fin state: 3  
length: 33  
set of inputs: 0 0 0 1 0 0 2 0 0 4 0 4 2 1 0 2 0 2 2 1 1 3 1 1 1 3 3 1 2 1 1 2 1

fin state: 0  
length: 35  
set of inputs: 0 0 0 1 0 0 2 0 0 4 0 4 2 1 0 2 0 2 2 1 1 3 1 1 1 3 3 1 2 1 1 3 1 1 4

fin state: 8  
length: 37  
set of inputs: 0 0 0 1 0 0 2 0 0 4 0 4 2 1 0 2 0 2 2 1 1 3 1 1 1 3 3 1 2 1 1 3 1 4 0 4 3

fin state: 9  
length: 30  
set of inputs: 0 0 0 1 0 0 2 0 0 4 0 4 2 1 0 2 0 2 2 1 1 3 1 1 3 1 1 3 4 3

На рисунке 16 изображено окно программы «InterFSM», осуществляющей пересечение двух конечных автоматов, с выставленными входными данными.

|                                              |                                       |                                       |
|----------------------------------------------|---------------------------------------|---------------------------------------|
| Path to the file with first FSM (not timed)  | <input type="text" value="C:\0.fsm"/> | <input type="button" value="Choose"/> |
| Path to the file with second FSM (not timed) | <input type="text" value="C:\1.fsm"/> | <input type="button" value="Choose"/> |
| Initial state of the first FSM               | <input type="text" value="0"/>        | <input type="button" value="Choose"/> |
| Initial state of the second FSM              | <input type="text" value="0"/>        | <input type="button" value="Choose"/> |
| Folder for result                            | <input type="text" value="C:\"/>      | <input type="button" value="Choose"/> |
| InterFSM.exe C:\0.fsm C:\1.fsm 0 0 C:\       |                                       | <input type="button" value="Start"/>  |

Рисунок 16 – Входные параметры для тестирования программы «InterFSM»

Содержимое файла «0.fsm»:

```

F 0
s 6
i 2
o 2
n0 0
p 12
0 0 3 0
0 1 5 1
1 0 1 1
1 1 1 0
2 0 3 0
2 1 4 0
3 0 2 0
3 1 0 0
4 0 5 1
4 1 3 0

```

5 0 1 0

5 1 0 0

Содержимое файла «1.fsm»:

F 0

s 10

i 2

o 2

n0 0

p 20

0 0 6 1

0 1 3 0

1 0 9 0

1 1 7 1

2 0 5 1

2 1 4 1

3 0 8 1

3 1 2 0

4 0 1 1

4 1 3 1

5 0 8 0

5 1 6 1

6 0 1 0

6 1 2 1

7 0 6 1

7 1 3 0

8 0 7 0

8 1 2 1

9 0 2 1

9 1 8 0

Результат работы программы, которым является пересечение пары полностью определённых, возможно недетерминированных, конечных автоматов, представлен ниже.

F 0

s 2

i 2

o 2

n0 0

p 1

0 0 1 0

### 3 Тестирование программных реализаций «FSMTest-2.0»

В данной главе мы рассмотрим тестирование программных реализаций, входящих в пакет «FSMTest-2.0».

#### 3.1 Схема и цель тестирования

Поскольку программные продукты пишутся программистами, то они могут содержать ошибки, т.е. поведение программной реализации может отличаться от ожидаемого поведения. Поэтому возникает задача тестирования разработанных программных продуктов. На рисунке 17 представлена схема тестирования программных реализаций.

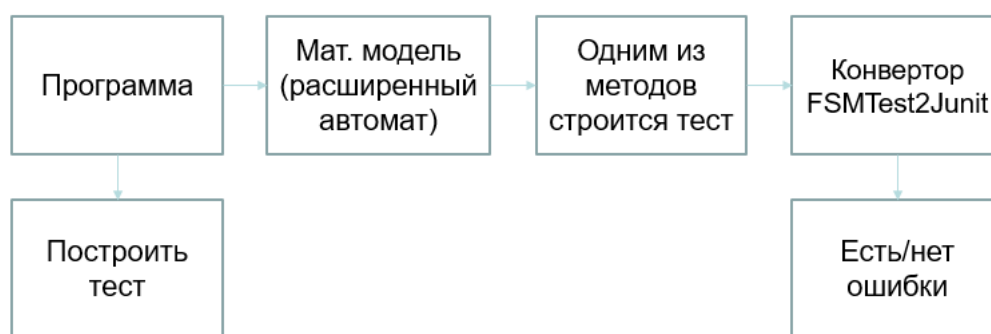


Рисунок 17 – Схема тестирования программных реализаций

Если мы хотим построить тест с гарантированной полнотой, т.е. тест, обнаруживающий ошибки определенного типа, то необходимо иметь адекватную математическую модель программы. Одной из таких моделей является расширенный автомат, в котором помимо состояний, параметризованных входных и выходных символов содержатся внутренние переменные, и переходы из состояния в состояние определяются соответствующими функциями.

Поскольку не существует методов построения тестов с гарантированной полнотой непосредственно для расширенных автоматов, то поведение расширенного автомата моделируется на входных последовательностях. В итоге получается конечный автомат, методы синтеза тестов для которого хорошо изучены и представлены в пакете «FSMTest-2.0».

Для автоматизации процесса тестирования используется инструмент JUnit [17], который подает построенный проверяющий тест на программную

реализацию, проверяет, совпадает ли ответ реализации с ожидаемой реакцией, и делает вывод о наличии ошибок в программной реализации.

### 3.2 Представление программы

Одной из адекватных математических моделей, описывающих поведение программных реализаций, является расширенный автомат. Состояниям автомата, как правило, соответствуют классы программ, а входо-выходным последовательностям, помечающим переходы, - методы класса и соответствующие им выходные реакции, как показано на рисунке 18.

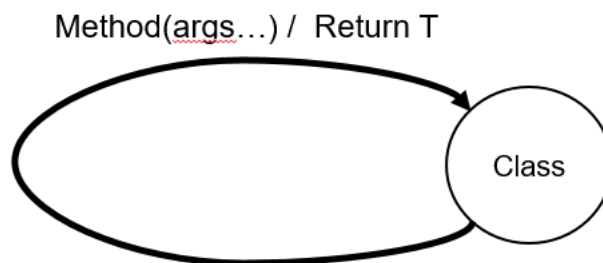


Рисунок 18 – Расширенный автомат программы

После построения расширенного автомата, мы переходим от него к конечному автомату, в котором состояниям соответствует состояние экземпляра класса в данный момент, т.е. то, какое значение принимают его член-данные, а в качестве входо-выходных последовательностей выступают методы с определёнными входными значениями и изменения в тестируемом классе.

Переход осуществляется следующим образом: единственное состояние расширенного автомата, которое соответствует экземпляру класса, разбивается на множество состояний, каждое из которых соответствует определённому набору значений член-данных класса. В качестве входных данных выступают методы с определёнными значениями аргументов, а в качестве выходных данных выступают ожидаемые изменения экземпляра класса.

### 3.3 Тестирование программной реализации

Для тестирования мы выбрали программную реализацию «ndfsm\_test», поскольку она реализована на Java и её можно тестировать при помощи JUnit.

Поведение данной реализации мы описали расширенным автоматом с одним состоянием, которому соответствует класс automata программы, и девятью переходами: generateInputSequences, setPartiple, removeFromCollector, initializeNewDevStates, setVD, setStates, setInitial, removeInitial, addState. Расширенный автомат, описывающий поведение программы, представлен на рисунке 19.

```
generateInputSequences(int maxSequenceLength, String[] inputs)
                                / ArrayList < String[] >
setPartiple(boolean isPartiple) / isPartiple
removeFromCollector(String collector, String input) / String collector
initializeNewDevStates(String[] addArray, String newElement)
                                /String[] newArray
setVD(Map < String, Sequence[] > vd) / VD(this)
setStates(ArrayList < state > states) / thisStates
setInitial(String stateName) / state.setInitial
removeInitial() / state.setInitial(false)
addState(State state) / states.add(state)
```

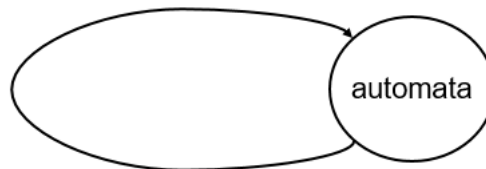


Рисунок 19 – Расширенный автомат, описывающий поведение программной реализации «ndfsm\_test»

Алгоритм тестирования данной программной реализации приведен ниже.

1. Строится расширенный автомат, описывающий программную реализацию.
2. По расширенному автомату строится эквивалентный конечный автомат.
3. По конечному автомату строится проверяющий тест одним из методов, реализованных в «FSMTest-2.0».
4. С помощью JUnit подать тест на программную реализацию.
5. JUnit проверяет, совпадает ли реакция реализации с ожидаемой реакцией, и делает заключение о наличии или отсутствии ошибки.

Часть построенного теста и входных данных изображены на рисунке 20.

|                                                                                                                                                                                                                                            |                                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Входные параметры</b><br><b>Object getInputs(); String[] inputs</b><br><b>= (String[]) ObjectResult; int size =</b><br><b>inputs.length</b><br><b>Object getOutputs(); String[]</b><br><b>outputs = (String[]) ObjectResult;</b><br>... | <b>Выходные реакции</b><br><b>true</b><br><b>Exception</b><br><b>size == sizeO</b>                                                                                 |
| <b>Входо-выходные</b><br><b>последовательности</b><br>0/2<br>1/2<br>2/0 0/2<br>0/2 2/0 0/2<br>2/0 1/2<br>1/2 2/0 1/2<br>2/0 3/0 0/2<br>0/2 2/0 3/0 0/2<br>2/0 3/0 1/2<br>1/2 2/0 3/0 1/2                                                   | Файл инициализации<br>import java.io.*;<br>import automata.Automata;<br>import automata.State;<br>import java.util.ArrayList;<br>#!<br><br>int sizeO;<br>#!<br>... |

Рисунок 20 – Часть построенного теста и входных данных

Реализация, построенная по данным, изображённым на рисунке 20, приведена в приложении Г.

Тестирование показало, что два тестируемых метода работают некорректно [14]. Если мы хотя бы раз вызвали метод возвращающий входы или выходы, а затем добавили новые состояния, то при следующем вызове метода он вернёт прошлые значения, а не обновлённые с учётом новых состояний. Без использования конечного автомата данная ошибка не обнаружилась, т.к. обычное тестирование позволяет обнаружить только все выходные ошибки, но не ошибки переходов. В случае тестирования при помощи формальных моделей мы можем обнаружить не только всех выходные ошибки, но и все ошибки переходов, для заданной модели.



## ЗАКЛЮЧЕНИЕ

В процессе работы изучены язык разметки XML и библиотеки, предоставляемые Java для работы с XML-документами.

Разработаны и программно реализованы алгоритмы проверки и разбора XML-документа. Для проверки XML-документов, реализован метод *processFilesFromFolder(File folder)*, принимающий на вход путь к директории с документами. Для разбора XML-документов, реализован метод *ParseElements(File file)*, принимающий на вход путь к документу, и формирующий поток данных, описывающий свойства элемента интерфейса.

В результате работы программы было выявлено, что созданные алгоритмы работают корректно, поскольку в результате получаются правильные окна реализации, а также список ошибок, допущенных при создании XML-документов.

Так же в новый пакет прикладных программ добавлены две программные реализации, отсутствующие в программе «FSMTest-1.0», и проверена правильность добавления.

Произведено тестирование программы *ndfsm\_test*. Для этого построен расширенный автомат для класса *automat* по его спецификации. Для автоматизации тестирования изучена библиотека JUnit. Подход к тестированию на основе автоматных моделей позволил обнаружить ошибки в программе *ndfsm\_test*, которые не обнаруживались обычным в тестировании программ подходом на основе обхода графа переходов.

## СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Васенин, В.А. Формальные модели программ и языков программирования. Часть 2. Современное состояние исследований / В.А. Васенин, М.А. Кривчиков // Программная инженерия, 2015. – №6. С. 24-33.
2. Бурдонов, И.Б. Использование конечных автоматов для тестирования программ / И.Б. Бурданов, А.С. Косачев, В.В. Кулямин // Программирование. – 2000. – №2. – С. 12-28.
3. Шаляпина, Н.А. Тесты на константные неисправности как веб-сервис / Н.А. Шаляпина, А.А. Зайцев, С.В. Батрацкий, М.Л. Громов // Труды Института системного программирования РАН. Том 30, выпуск 1, 2018. С. 41-54.
4. Villa, T. The unknown component problem: theory and applications / T. Villa, N. Yevtushenko, A. Mishenko, R.K. Brayton, A. Petrenko, A. Sangiovanni-Vincentelli // Berlin: Springer, 2012. – 311 p.
5. Petrenko, A. Confirming configurations in EFSM testing / A. Petrenko, S. Boroday, R. Groz // IEEE Transactions on Software Engineering – V.1 – 2004. – P. 29-42.
6. Что такое программная оболочка? [Электронный ресурс]. URL: [http://book.kbsu.ru/theory/chapter6/1\\_6\\_8.html](http://book.kbsu.ru/theory/chapter6/1_6_8.html) (Дата обращения: 21.09.16)
7. API [Электронный ресурс]: свободная энциклопедия Википедия. URL: <https://ru.wikipedia.org/wiki/API> (Дата обращения: 12.10.16)
8. Основы XML для начинающих [Электронный ресурс]. URL: <https://www.ibm.com/developerworks/ru/library/x-newxml/> (Дата обращения 10.11.16)
9. Василевский, М.П. О распознавании неисправностей автоматов. / М.П. Василевский. // Кибернетика, 9(4) – 1973. – С. 93-108.
10. Dorofeeva, R. Experimental evaluation of FSM-based testing methods // R. Dorofeeva, K. El-Fakih, S. Maag, A.R. Cavalli, N. Yevtushenko // In Proc. Of the IEEE International Conference on Software Engineering and Formal Methods (SEFM05) – 2005. – P. 23-32.
11. Shabaldina, N. FSMTest-1.0: a manual for researches / N. Shabaldina, M. Gromov // Proceedings of IEEE East-West Design & Test Symposium EWDTS'2015, 2015. – P. 216-219.

12. Батрацкий, С.В. Создание кроссплатформенной версии приложения «FSM Test-1.0» / С.В. Батрацкий, В.С. Белых, А.С. Твардовский // Сборник научных трудов в 9 ч. / под ред. проф. Б.Ю. Лемешко, проф. А.А. Попова, проф. М.Э. Рояка, доц. В.С. Тимофеева. – Новосибирск: Изд-во НГТУ, 2016. – Часть 2. С. 163-165.
13. Батрацкий, С.В. К программной реализации пакета прикладных программ «FSMTest-2.0». / С.В. Батрацкий, В.С. Белых. // Всероссийская конференция «Студенческий научно-исследовательский инкубатор» ТГУ, 2017. – С. 6.
14. Батрацкий, С.В. Тестирование Java программ с использованием инструмента FSMTEST2JUNIT / С.В. Батрацкий, С.А. Прокопенко, М.Л. Громов, Н.В. Шабалдина // Новые информационные технологии в исследовании сложных структур: материалы двенадцатой российской конференции с международным участием. Томск : Издательский Дом Томского государственного университета, 2018. – С. 67.
15. XML-программирование в технологии Java [Электронный ресурс]. URL: <http://khpi-iip.mipk.kharkiv.edu/library/extent/prog/iipXML/xmljava.html> (Дата обращения 6.12.16)
16. Упрощение XML-программирования при помощи JDOM [Электронный ресурс]. URL: <http://www.ibm.com/developerworks/ru/library/j-jdom/> (Дата обращения 7.12.16)
17. Тестирование в java. JUnit [Электронный ресурс] URL: <https://habrahabr.ru/post/120101/> (дата обращения 2.12.2017)

## ПРИЛОЖЕНИЕ А

### Код парсера

```
package FSMPackege;
import java.io.File;
import java.io.IOException;
import java.lang.reflect.Array;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
public class XMLParserFSM {
    ArrayList < File > valid_files = new ArrayList < File >();
    String invalid_node;
    ArrayList < String > list_errors = new ArrayList < String >();
    public void processFilesFromFolder(File folder) throws IOException,
    ParserConfigurationException, SAXException {
        File[] folderEntries = folder.listFiles();
        for (File file : folderEntries) {
            if (file.isDirectory()) {
                processFilesFromFolder(file);
                continue;
            }
            DocumentBuilder documentBuilder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            try {
                Document document = documentBuilder.parse(file);
                if(checkNames(document.getDocumentElement(), file)) {
                    valid_files.add(file);
                }
            }
            catch (SAXException e) {
                list_errors.add("information about error : " +
e.toString());
                list_errors.add("");
            }
        }
    }
    public File getFile(int i) {
        return valid_files.get(i);
    }
    public ArrayList < String > getErrorList() {
        return list_errors;
    }
    public ArrayList < File > getFiles() {
        return valid_files;
    }
    public boolean hasFile(File file) {
        for(File i : valid_files) {
            if(file.equals(i)) {
                return true;
            }
        }
        return false;
    }
}
```

```

    }
    boolean checkNames(Element root_element, File file) throws
ParserConfigurationException,
        SAXException, IOException {
        NodeList list_of_main_childs = root_element.getChildNodes();
        boolean valid = true;
        for(int i = 0; i < list_of_main_childs.getLength(); ++i) {
            Node current_node = list_of_main_childs.item(i);
            if(current_node.getNodeType() == Node.ELEMENT_NODE) {
                invalid_node = current_node.getNodeName();
                if(valid)
                    valid = checkCurrentNode(current_node, file);
                else
                    checkCurrentNode(current_node, file);
            }
        }
        return valid;
    }

    boolean checkCurrentNode(Node main_node, File file) throws
ParserConfigurationException,
        SAXException, IOException {
        boolean valid = true, entered = false;
        File containing_namespace = new File("namespace.xml");
        DocumentBuilder documentBuilder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
        Document document = documentBuilder.parse(containing_namespace);
        Element root_element_namespace = document.getDocumentElement();
        NodeList main_childs_of_namespace =
root_element_namespace.getChildNodes();
        for(int i = 0; i <
main_childs_of_namespace.getLength(); ++i) {
            Node current_node_namespace =
main_childs_of_namespace.item(i);
            if(current_node_namespace.getNodeType() ==
Node.ELEMENT_NODE) {
                if(main_node.getNodeName().equals(current_node_namespace.getNodeName())
) {
                    if(valid)
                        valid = checkPropertiesNode(main_node,
current_node_namespace, file);
                    else
                        checkPropertiesNode(main_node,
current_node_namespace, file);
                    entered = true;
                }
            }
        }
        if(!entered) {
            valid = false;
            list_errors.add("error:");
            list_errors.add("node: " + invalid_node);
            list_errors.add("path: " + file);
            list_errors.add("type: child with name: " +
main_node.getNodeName() + ", don't exist");
            list_errors.add("");
        }
        return valid;
    }

    boolean checkPropertiesNode(Node main_node, Node checking_node, File
file) throws IOException {

        boolean valid = true, entered = false;
        NodeList list_childs_of_main_node = main_node.getChildNodes();

```

```

        NodeList list_childs_of_checking_node =
checking_node.getChildNodes();
        Element main_element = (Element) main_node;
        for(int i = 0; i < list_childs_of_checking_node.getLength(); ++i)
        {
            Node current_node_attribute =
list_childs_of_checking_node.item(i);
            if(current_node_attribute.getNodeType() ==
Node.ELEMENT_NODE) {
                if(current_node_attribute.getNodeName() ==
"attribute") {
                    Element current_element_checking = (Element)
current_node_attribute;
                    if(!main_element.hasAttribute(current_element_checking.getAttribute("na
me"))) {
                        valid = false;
                        list_errors.add("error:");
                        list_errors.add("node: " + invalid_node);
                        list_errors.add("path: " + file);
                        list_errors.add("type: in node " +
main_node.getNodeName() + ", not found attribute: " +
current_element_checking.getAttribute("name"));
                        list_errors.add("");
                    }
                }
            }
        }
        for(int i = 0; i < list_childs_of_main_node.getLength(); ++i) {
            Node current_main_node = list_childs_of_main_node.item(i);
            if(current_main_node.getNodeType() == Node.ELEMENT_NODE) {
                for(int j = 0; j <
list_childs_of_checking_node.getLength(); ++j) {
                    Node current_checking_node =
list_childs_of_checking_node.item(j);
                    if(current_checking_node.getNodeType() ==
Node.ELEMENT_NODE) {
                        Element checking_element = (Element)
current_checking_node;
                        if(current_main_node.getNodeName().equals(checking_element.getAttribute
("name"))) {
                            if(valid)
                                valid =
checkPropertiesNode(current_main_node, current_checking_node, file);
                            else
                                checkPropertiesNode(current_main_node, current_checking_node, file);
                                entered = true;
                        }
                    }
                }
            }
            if(!entered) {
                valid = false;
                list_errors.add("error:");
                list_errors.add("node: " + invalid_node);
                list_errors.add("path: " + file);
                list_errors.add("type: child with name: " +
main_node.getNodeName() + ", don't exist");
                list_errors.add("");
            }
            entered = false;
        }
    }
    return valid;
}

```

```

    }
}

package FSMPackege;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.util.ArrayList;

import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

public class FSMExecutor {

    int num_of_elements = 0;
    int number_of_element = 0;
    String current_name = null;
    String type_object = null;
    ArrayList < String > tmpsc = new ArrayList < String >();
    ArrayList < String > info_i = new ArrayList<>();
    ArrayList < String > tmpcl = new ArrayList < String >();
    boolean one_command;

    FSMInput[] inputs; //массив входов

    ArrayList<String> Conditions;// тройки, аргумент-знак(< или <=) -
    аргумент(в трёх разных элементах)
    String image; //рисунок, нужен ли список?
    ArrayList<ArrayList<String>> CommandLine; //командные строки
    ArrayList<ArrayList<String>> SwitchConditions; //условия для блокировки
    аргументов

    //списки элементов интерфейса
    ArrayList<FSMSpinner> Spinners;// выбор значения
    ArrayList<FSMCheckBox> CheckBoxes;// флаг
    ArrayList<FSMGroupRB> RadioButtons;// список с выбором
    ArrayList<FSMFileChooser> FileChoosers;// файл
    ArrayList<String> Constants;

```

```

JTextArea Comline;

//int[] NumOfOutputs; // отчёты о результате работы !!!НУЖНЫ ЛИ?
//String[] outputs;

FSMExecuter(JTextArea FSMStatus)
{
    inputs = null;
    Spinners = new ArrayList<FSMSpinner>();
    CheckBoxes = new ArrayList<FSMCheckBox>();
    RadioButtons = new ArrayList<FSMGroupRB>();
    FileChoosers = new ArrayList<FSMFileChooser>();
    Constants = new ArrayList<String>();
    CommandLine = null;
    Conditions = new ArrayList<String>();

    Comline = FSMStatus;
}

void ParseElements(File file) throws ParserConfigurationException,
SAXException, IOException {

    SwitchConditions = new ArrayList < ArrayList < String > >();
    Conditions = new ArrayList < String >();
    CommandLine = new ArrayList < ArrayList < String > >();
    //синтаксический анализатор, который производит деревья объекта
DOM XML-документа
    DocumentBuilder documentBuilder =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
    //строится дерево объектов и возвращается ссылка на него, в виде
объекта Document
    Document current_document = documentBuilder.parse(file);
    //Получаем корневой элемент
    Element root_element = current_document.getDocumentElement();
    //создаём список потомков корневого элемента
    NodeList list_of_main_childs = root_element.getChildNodes();
    for(int i = 0; i < list_of_main_childs.getLength(); ++i) {
        Node current_node = list_of_main_childs.item(i);
        if(current_node.getNodeType() == Node.ELEMENT_NODE) {
            NodeList list_of_current_childs =
current_node.getChildNodes();
            if(((Element)
list_of_current_childs).hasAttribute("name"))
                ++num_of_elements;
        }
    }
    inputs = new FSMInput[num_of_elements];
    parsingMainNodes(list_of_main_childs);
}

//parsingMainNodes - метод, разбирающий главные узлы
//list_of_main_childs - список главных узлов
void parsingMainNodes(NodeList list_of_main_childs) {
    for(int i = 0; i < list_of_main_childs.getLength(); ++i) {
        Node current_node = list_of_main_childs.item(i);
        info_i.clear();
        tmpsc.clear();
        if(current_node.getNodeType() == Node.ELEMENT_NODE) {
            NodeList childs_list_of_current_node =
current_node.getChildNodes();
            type_object = current_node.getNodeName();
            if(current_node.getNodeName() == "image")
                parsingImageNodes(current_node);
        }
    }
}

```



```

        if(current_node.getNodeName() == "const")
            parsingConstNodes(current_node);
        if(current_node.getNodeName() == "conditions")

            parsingConditionNodes(childs_list_of_current_node,
list_of_main_childs);
        if(current_node.getNodeName() == "if")
            parsingIfNodes(childs_list_of_current_node,
list_of_main_childs);
        if(current_node.getNodeName() == "range")
            parsingRangeNode(current_node,
childs_list_of_current_node);
        if(current_node.getNodeName() == "select")
            parsingSelecteNode(current_node,
childs_list_of_current_node);
        if(current_node.getNodeName() == "flag")
            parsingFlagNode(current_node);
        if(current_node.getNodeName() == "dirselector")
            parsingDirselectorNode(current_node);
        if(current_node.getNodeName() == "commands")

            parsingCommandsNode(childs_list_of_current_node, list_of_main_childs);
    }
}

void parsingImageNodes(Node current_node) {
    Element element = (Element) current_node;
    image = element.getAttribute("url");
}

void parsingConstNodes(Node current_node) {
    type_object = "const";
    Element element = (Element) current_node;
    current_name = element.getAttribute("name");
    info_i.add(element.getAttribute("val"));
    addElement();
}

void parsingConditionNodes(NodeList list_of_current_childs, NodeList
list_of_main_childs) {
    for(int j = 0; j < list_of_current_childs.getLength(); ++j) {
        Node condition_node = list_of_current_childs.item(j);
        if(condition_node.getNodeType() == Node.ELEMENT_NODE) {
            Element condition_element = (Element) condition_node;
            for(int i = 0; i < 2; ++i) {
                for(int k = 0; k <
list_of_main_childs.getLength(); ++k) {
                    Node tmp_node =
list_of_main_childs.item(k);
                    if(tmp_node.getNodeType() ==
Node.ELEMENT_NODE) {
                        Element tmp_element = (Element)
tmp_node;
                        if(i == 0) {

                            if(tmp_element.getAttribute("var").equals(condition_element.getAttribut
e("var1"))) {

                                Conditions.add(tmp_element.getAttribute("name"));

                                Conditions.add(tmp_element.getAttribute("symbol"));
                            }

```

```

        } else {

            if(tmp_element.getAttribute("var").equals(condition_element.getAttribut
e("var2")))

                Conditions.add(tmp_element.getAttribute("name"));
        }
    }
}

void parsingIfNodes(NodeList list_childs_of_if, NodeList
list_of_main_childs) {
    for(int j = 0; j < list_childs_of_if.getLength(); ++j) {
        Node child_of_if = list_childs_of_if.item(j);
        if(child_of_if.getNodeType() == Node.ELEMENT_NODE) {
            NodeList list_of_current_childs =
child_of_if.getChildNodes();
            if(child_of_if.getNodeName() == "conditions")
                conditionChildOfIf(list_of_current_childs,
list_of_main_childs);
            else
                otherChildOfIf(list_of_current_childs,
list_of_main_childs);
        }
    }
}

void conditionChildOfIf(NodeList list_of_current_childs, NodeList
list_of_main_childs) {
    for(int i = 0; i < list_of_current_childs.getLength(); ++i) {
        Node condition_node = list_of_current_childs.item(i);
        if(condition_node.getNodeType() == Node.ELEMENT_NODE) {
            Element condition_element = (Element) condition_node;
            for(int j = 0; j < 2; ++j) {
                for(int k = 0; k <
list_of_main_childs.getLength(); ++k) {
                    Node tmp_node =
list_of_main_childs.item(k);
                    if(tmp_node.getNodeType() ==
Node.ELEMENT_NODE) {
                        Element tmp_element = (Element)
tmp_node;
                        if(j == 0) {

                            if(tmp_element.getAttribute("var").equals(condition_element.getAttribut
e("var1"))) {

                                tmpsc.add(tmp_element.getAttribute("name"));

                                tmpsc.add(condition_element.getAttribute("symbol"));
                            }
                        } else {

                            if(tmp_element.getAttribute("var").equals(condition_element.getAttribut
e("var2"))) {

                                tmpsc.add(tmp_element.getAttribute("name"));
                                j = 2;
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    if(j == 1)

tmpsc.add(condition_element.getAttribute("var2"));
    }
}
SwitchConditions.add(tmpsc);
tmpsc = new ArrayList < String >();
}

void otherChildOfIf(NodeList list_of_current_childs, NodeList
list_of_main_childs) {
    for(int i = 0; i < list_of_current_childs.getLength(); ++i) {
        Node enabled_or_non = list_of_current_childs.item(i);
        NodeList list_of_parameters =
enabled_or_non.getChildNodes();
        if(enabled_or_non.getNodeType() == Node.ELEMENT_NODE) {
            for(int k = 0; k < list_of_parameters.getLength();
++k) {
                Node parameter = list_of_parameters.item(k);
                if(parameter.getNodeType() ==
Node.ELEMENT_NODE) {
                    Element element = (Element) parameter;
                    for(int j = 0; j <
list_of_main_childs.getLength(); ++j) {
                        Node tmp_node =
list_of_main_childs.item(j);
                        if(tmp_node.getNodeType() ==
Node.ELEMENT_NODE) {
                            Element tmp_element =
(Element) tmp_node;

                            if(tmp_element.getAttribute("var").equals(element.getAttribute("val")))

tmpsc.add(tmp_element.getAttribute("name"));
                                }
                            }
                        }
                    }
                SwitchConditions.add(tmpsc);
                tmpsc = new ArrayList < String >();
            }
        }
    }

void parsingRangeNode(Node current_node, NodeList
list_of_current_childs) {
    type_object = "range";
    Element element = (Element) current_node;
    current_name = element.getAttribute("name");
    for(int j = 0; j < list_of_current_childs.getLength(); ++j) {
        Node property_node = list_of_current_childs.item(j);
        if(property_node.getNodeType() == Node.ELEMENT_NODE) {
            Element tmp = (Element) property_node;
            if(property_node.getNodeName() == "min")
                info_i.add(tmp.getAttribute("val"));
        }
    }
    for(int j = 0; j < list_of_current_childs.getLength(); ++j) {
        Node property_node = list_of_current_childs.item(j);

```

```

        if(property_node.getNodeType() == Node.ELEMENT_NODE) {
            Element tmp = (Element) property_node;
            if(property_node.getNodeName() == "max")
                info_i.add(tmp.getAttribute("val"));
        }
    }
    addElement();
}

void parsingSelecteNode(Node current_node, NodeList
list_of_current_childs) {
    type_object = "select";
    Element element = (Element) current_node;
    current_name = element.getAttribute("name");
    for(int j = 0; j < list_of_current_childs.getLength(); ++j) {
        Node property_node = list_of_current_childs.item(j);
        if(property_node.getNodeType() == Node.ELEMENT_NODE) {
            Element tmp = (Element) property_node;
            if(property_node.getNodeName() == "variant") {
                info_i.add(tmp.getAttribute("name"));
                info_i.add(tmp.getAttribute("val"));
            }
        }
    }
    addElement();
}

void parsingFlagNode(Node current_node) {
    type_object = "flag";
    Element element = (Element) current_node;
    current_name = element.getAttribute("name");
    info_i.add(element.getAttribute("checked"));
    info_i.add(element.getAttribute("not_checked"));
    addElement();
}

void parsingDirselectorNode(Node current_node) {
    type_object = "dirselector";
    Element element = (Element) current_node;
    current_name = element.getAttribute("name");
    info_i.add(element.getAttribute("type"));
    addElement();
}

void parsingCommandsNode(NodeList list_of_current_nodes, NodeList
list_of_main_nodes) {
    for(int j = 0; j < list_of_current_nodes.getLength(); ++j) {
        Node tmp = list_of_current_nodes.item(j);
        if(tmp.getNodeType() == Node.ELEMENT_NODE) {
            tmpcl = new ArrayList < String >();
            NodeList command_line_nodes = tmp.getChildNodes();
            for(int k = 0; k < command_line_nodes.getLength();
++k) {
                Node command = command_line_nodes.item(k);
                if(command.getNodeType() == Node.ELEMENT_NODE)
                {
                    Element vars = (Element) command;
                    if(vars.getTagName() == "command") {

                        tmpcl.add(vars.getAttribute("val"));
                    } else {
                        for(int i = 0; i <
list_of_main_nodes.getLength(); ++i) {

```

```

list_of_main_nodes.item(i);
Node.ELEMENT_NODE)
tmp_node).getAttribute("var").equals(vars.getAttribute("val"))
        if(((Element)
tmp_node).getAttribute("name"));
    }
    }
    }
    CommandLine.add(tmpcl);
}
}

void addElement() {
    inputs[number_of_element] = new FSMInput(current_name,
type_object, info_i, true);
    ++number_of_element;
}

void TMPElements()
{
    // сюда пишутся условия для затемнения элементов
    SwitchConditions = new ArrayList<ArrayList<String>>();
    ArrayList<String> tmpsc = new ArrayList<String>();//содержит
условия из конъюнкции
    tmpsc.add("Any List1"); //название аргумента 1
    tmpsc.add("==");
    tmpsc.add("First type 1"); //значение аргумента 1
    //tmpsc.add("Number of anythinks1"); //название аргумента 2
    //tmpsc.add("<");
    //tmpsc.add("2");// константа для аргумента 2
    SwitchConditions.add(tmpsc);
    // далее два списка имён элементов из then, т.е. если конъюнкция
выполняется
    tmpsc = new ArrayList<String>();//содержит аргументы из enable,
то что не затемняется
    tmpsc.add("Any File1");
    tmpsc.add("Number of anythinks1");
    SwitchConditions.add(tmpsc);
    tmpsc = new ArrayList<String>();//содержит аргументы из not
enable, то что затемняется
    tmpsc.add("Any Flag1");
    SwitchConditions.add(tmpsc);
    // далее два списка имён элементов из else, т.е. если конъюнкция
не выполняется
    tmpsc = new ArrayList<String>();//содержит аргументы из enable,
    tmpsc.add("Any File1");
    tmpsc.add("Any Flag1");
    SwitchConditions.add(tmpsc);
    tmpsc = new ArrayList<String>();//содержит аргументы из not
enable,
    tmpsc.add("Number of anythinks1");
    SwitchConditions.add(tmpsc);
    //затем всё повторяется т.е. в следующем списке лежат другие
условия для затемнения элементов, как и в 0-ом элементе списка
    // т.е. один if есть 5 строковых списков, или 5 элементов
    SwitchConditions

    inputs = new FSMInput[4];

```

```

        ArrayList<String> info_i = new ArrayList<>();
        //для range
        info_i.add("1");//минимальное значение
        info_i.add("1000");//максимальное значение, если его нет, то
        ничего не добавляй, пусть будет один элемент в списке
        //в Input передаётся название, тип, info, boolean(true, если
        аргумент может т влиять на затемнение false)
        inputs[0] = new FSMInput("Number of anythinks1", "range", info_i,
        true);

        info_i.clear();

        //для select
        info_i.add("First type 1"); //первый вариант(то что отобразиться
        в интерфейсе)
        info_i.add("First type 1 arg"); //первый вариант(то что
        передаётся в ехе файл)
        info_i.add("Second type 1"); //первый выбор(то что отобразиться в
        интерфейсе)
        info_i.add("First type 2 arg"); //первый выбор(то что передаётся
        в ехе файл)
        inputs[1] = new FSMInput("Any List1", "select", info_i, true);
        info_i.clear();

        //для flag
        info_i.add("First arg"); //аргумент, который передаётся в ехе
        файл в случае true
        info_i.add("Second arg"); //аргумент, который передаётся в ехе
        файл в случае false
        inputs[2] = new FSMInput("Any Flag1", "flag", info_i, true);
        info_i.clear();

        info_i.add("dir"); // dir, либо file
        inputs[3] = new FSMInput("Any File1", "dirselector", info_i,
        true);

        info_i.clear();

        //пример заполнения командных строк
        CommandLine = new ArrayList<ArrayList<String>>(); //память под
        список списков
        ArrayList<String> tmpcl = new ArrayList<String>(); // создаём
        новую командную строку
        tmpcl.add("E:\\FSMTestV"); // первый элемент - путь к файлу
        tmpcl.add("Any Flag1"); // дальше - аргументы
        tmpcl.add("Any File1");
        CommandLine.add(tmpcl);

        /*//пример заполнения условий для range
        Conditions = new ArrayList<String>();
        Conditions.add("Number of anythinks1"); //имя аргумента типа
        range
        Conditions.add("<"); // знак < или <=
        Conditions.add("Number of anythinks1"); //имя аргумента range
        Conditions.add("Number of anythinks1"); //далее второе условие и
        т д
        Conditions.add("<=");
        Conditions.add("Number of anythinks1");*/
    }

    void FSMCreateElements()
    {
        PropertyChangeListener VLis = new FSMValListener();
        int sp=0, ch=0, rb=0, fc=0, ff=0;

```

```

        for(int i=0;i<inputs.length;i++)
        {
            switch(inputs[i].type)
            {
                case "range":
                {
                    FSMSpinner tmps = new FSMSpinner(inputs[i].name,
inputs[i].info, VLis, inputs[i].inflag );
                    Spinners.add(tmps);
                    inputs[i].NumInList = sp;
                    sp++;
                    break;
                }
                case "flag":
                {
                    FSMCheckBox tmpc = new FSMCheckBox(inputs[i].name,
VLis, inputs[i].inflag);
                    CheckBoxes.add(tmpc);

                    inputs[i].NumInList = ch;
                    ch++;
                    break;
                }
                case "select":
                {
                    inputs[i].NumInList = rb;
                    FSMGroupRB trb = new FSMGroupRB(inputs[i].info,
inputs[i].name, VLis, inputs[i].inflag);
                    RadioButtons.add(trb);
                    rb++;
                    break;
                }
                case "dirselector":
                {
                    FSMFileChooser tmpf = new
FSMFileChooser(inputs[i].name, VLis, inputs[i].inflag,
inputs[i].info.get(0));
                    FileChoosers.add(tmpf);
                    inputs[i].NumInList = fc;
                    fc++;
                    break;
                }
                case "const":
                {
                    Constants.add(inputs[i].name);
                    inputs[i].NumInList = ff;
                    ff++;
                    break;
                }
            }
        }
        FSMVerifSwitchCon();
    }

    boolean FSMVerifCon()
    {
        Integer min = null;
        Integer max = null;
        for(int i=0;i<Conditions.size();i=i+3)
        {
            for(int j = 0;j < inputs.length;j++)
            {

```

```

        if(inputs[j].name.equals(Conditions.get(i))) min =
new Integer(Spinners.get(inputs[j].NumInList).val);
        if(inputs[j].name.equals(Conditions.get(i+2))) max =
new Integer(Spinners.get(inputs[j].NumInList).val);
    }
    switch(Conditions.get(i+1))
    {
    case "<":
    {
        if(min>=max) return false;
    }
    case "<=":
    {
        if(min>max) return false;
    }
    }
    }
    return true;
}

boolean FSMVerifArgs()
{
    for(int i=0;i<inputs.length;i++)
    {
        switch(inputs[i].type)
        {
        case "range":
        {
            if(Spinners.get(inputs[i].NumInList).EnableFlag)
            {
                try {

Integer.parseInt(Spinners.get(inputs[i].NumInList).val);
                } catch (NumberFormatException e) {
                    return false;
                }

if((Integer.parseInt(Spinners.get(inputs[i].NumInList).val)<Integer.parseInt(
inputs[i].info.get(0))) || (Integer.parseInt(Spinners.get(inputs[i].NumInList).
val)>Integer.parseInt(inputs[i].info.get(1)))) return false;
            }
            break;
        }
        case "dirselector":
        {
            if(FileChoosers.get(inputs[i].NumInList).EnableFlag)
            {
                if(inputs[i].info.get(0).equals("dir"))
                {
                }
                if(inputs[i].info.get(0).equals("file"))
                {
                    if (!(new
File(FileChoosers.get(inputs[i].NumInList).val)).exists())) {
                        return false;
                    }
                }
            }
            break;
        }
    }
}
}
}

```



```

        return true;
    }

    void FSMVerifSwitchCon()
    {
        String Var1t=null;
        String Var2t=null;
        int nk = 0;
        boolean flag = true;
        Integer v1 = null, v2 = null;

        for(int k=0;k<SwitchConditions.size();k=k+5)
        {
            flag = true;
            for(int c=0;c<SwitchConditions.get(k).size();c=c+3)
            {
                Var1t="";
                Var2t="";
                for(int i=0;i<inputs.length;i++)
                {
                    if(inputs[i].name.equals(SwitchConditions.get(k).get(c)))
                    {
                        switch(inputs[i].type)
                        {
                            case "range":
                            {
                                Var1t=Spinners.get(inputs[i].NumInList).val;
                                break;
                            }
                            case "flag":
                            {
                                Var1t=CheckBoxes.get(inputs[i].NumInList).val;
                                break;
                            }
                            case "select":
                            {
                                Var1t=RadioButtons.get(inputs[i].NumInList).val;
                                break;
                            }
                            case "dirselector":
                            {
                                Var1t=FileChoosers.get(inputs[i].NumInList).val;
                                break;
                            }
                        }
                    }
                }

                if(inputs[i].name.equals(SwitchConditions.get(k).get(c+2)))
                {
                    switch(inputs[i].type)
                    {
                        case "range":
                        {
                            Var2t=Spinners.get(inputs[i].NumInList).val;
                            break;
                        }
                    }
                }
            }
        }
    }

```

```

        case "flag":
        {
            Var2t=CheckBoxes.get(inputs[i].NumInList).val;
            break;
        }
        case "select":
        {
            Var2t=RadioButtons.get(inputs[i].NumInList).val;
            break;
        }
        case "dirselector":
        {
            Var2t=FileChoosers.get(inputs[i].NumInList).val;
            break;
        }
    }
    }
    if(Var1t.equals("")) Var1t =
SwitchConditions.get(k).get(c);
    if(Var2t.equals("")) Var2t =
SwitchConditions.get(k).get(c+2);
    switch(SwitchConditions.get(k).get(c+1))
    {
        case "==":
        {
            if(!(Var1t.equals(Var2t))) flag = false;
            break;
        }
        case "!=":
        {
            if(Var1t.equals(Var2t)) flag = false;
            break;
        }
        case "<":
        {
            v1 = new Integer(Var1t);
            v2 = new Integer(Var2t);
            if(v1>=v2) flag = false;
            break;
        }
        case ">":
        {
            v1 = new Integer(Var1t);
            v2 = new Integer(Var2t);
            if(v1<=v2) flag = false;
            break;
        }
        case "<=":
        {
            v1 = new Integer(Var1t);
            v2 = new Integer(Var2t);
            if(v1>v2) flag = false;
            break;
        }
        case ">=":
        {
            v1 = new Integer(Var1t);
            v2 = new Integer(Var2t);
            if(v1<v2) flag = false;

```

```

        break;
    }
}
if(flag) nk = k+1;
else nk = k+3;

for(int c=0;c<SwitchConditions.get(nk).size();c++)
{
    for(int i=0;i<inputs.length;i++)
    {
        if(inputs[i].name.equals(SwitchConditions.get(nk).get(c)))
        {
            switch(inputs[i].type)
            {
                case "range":
                {
                    Spinners.get(inputs[i].NumInList).FSMSwitch(true);
                    break;
                }
                case "flag":
                {
                    CheckBoxes.get(inputs[i].NumInList).FSMSwitch(true);
                    break;
                }
                case "select":
                {
                    RadioButtons.get(inputs[i].NumInList).FSMSwitch(true);
                    break;
                }
                case "dirselector":
                {
                    FileChoosers.get(inputs[i].NumInList).FSMSwitch(true);
                    break;
                }
                case "const":
                {
                    Constants.set(inputs[i].NumInList,
inputs[i].info.get(0));
                    break;
                }
            }
        }
    }
}
nk++;
for(int c=0;c<SwitchConditions.get(nk).size();c++)
{
    for(int i=0;i<inputs.length;i++)
    {
        if(inputs[i].name.equals(SwitchConditions.get(nk).get(c)))
        {
            switch(inputs[i].type)
            {
                case "range":
                {

```

```

Spinners.get(inputs[i].NumInList).FSMSwitch(false);
        break;
    }
    case "flag":
    {

CheckBoxes.get(inputs[i].NumInList).FSMSwitch(false);
        break;
    }
    case "select":
    {

RadioButtons.get(inputs[i].NumInList).FSMSwitch(false);
        break;
    }
    case "dirselector":
    {

FileChooser.get(inputs[i].NumInList).FSMSwitch(false);
        break;
    }
    case "const":
    {
        Constants.set(inputs[i].NumInList, "");
        break;
    }
    }
    }
}

}

}

}

void FSMRunEXEtmp()
{
    ArrayList<String> tmparg = new ArrayList<String>();
    tmparg.add("E:\\FSMTestV");
    tmparg.add("test2");
    ProcessBuilder pb = new ProcessBuilder(tmparg);
    try {
        Process p = pb.start();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

void FSMRunEXE(String dir) throws InterruptedException
{
    if((!FSMVerifCon()) || (!FSMVerifArgs()))
    {
        JOptionPane.showMessageDialog(null, "Error! Invalid value
of arguments.");
        return;
    }

    String line;
    ArrayList<String> StrArg = new ArrayList<String>();
    int n=0;
    int k=0;
    for(int i=0;i<CommandLine.size();i++)
    {
        StrArg.add(dir+CommandLine.get(i).get(0));
    }
}

```

```

        for(int j = 1;j<CommandLine.get(i).size();j++)
        {
            n=0;
            for(k = 0;k < inputs.length;k++)
            {

if(inputs[k].name.equals(CommandLine.get(i).get(j)))
                {
                    n=k;
                    k=inputs.length+1;
                }
            }
            switch(inputs[n].type)
            {
            case "const":
            {
                StrArg.add(inputs[n].info.get(0));
                break;
            }
            case "range":
            {

StrArg.add(Spinners.get(inputs[n].NumInList).val);
                break;
            }
            case "flag":
            {

if(CheckBoxes.get(inputs[n].NumInList).val.equals("checked"))
                {
                    StrArg.add(inputs[n].info.get(0));
                }
                else
                {
                    StrArg.add(inputs[n].info.get(1));
                }
                break;
            }
            case "select":
            {
                for(int l = 0;l <
RadioButtons.get(inputs[n].NumInList).RadioButtons.length*2;l=l+2)
                {

                    if(RadioButtons.get(inputs[n].NumInList).val.equals(inputs[n].info.get(
1)))
                    {

StrArg.add(inputs[n].info.get(l+1));
                    }
                }
                break;
            }
            case "dirselector":
            {

StrArg.add(FileChoosers.get(inputs[n].NumInList).val);
                break;
            }
            }
            if(StrArg.get(StrArg.size()-1).equals(""))
            {
                StrArg.remove(StrArg.size()-1);

```

```

        }
    }
    //ArrayList<String> StrArg2 = new ArrayList<String>();
    //StrArg2.add("cmd");
    for(int as = 0;as < StrArg.size();as++)
    {
        //StrArg2.add(StrArg.get(as));
        System.out.println(StrArg.get(as));
    }
    ProcessBuilder pb = new ProcessBuilder(StrArg);
    try {
        pb.redirectErrorStream(true);
        Process p = pb.start();
        p.waitFor();
    } catch (IOException e) {
        //e.printStackTrace();
        i = CommandLine.size();
        JOptionPane.showMessageDialog(null, "Error! Exe-file
can not be started");
    }
    StrArg.clear();
}

void FSMShowElements(JPanel p, GridBagLayout FSMGBL,
GridBagConstraints FSMGBLC)
{
    for(int i=0;i<inputs.length;i++)
    {
        if(inputs[i].type.equals("range"))
        {
            FSMGBL.setConstraints(Spinners.get(inputs[i].NumInList), FSMGBLC);
            p.add(Spinners.get(inputs[i].NumInList));
        }
        if(inputs[i].type.equals("flag"))
        {
            FSMGBL.setConstraints(CheckBoxes.get(inputs[i].NumInList), FSMGBLC);
            p.add(CheckBoxes.get(inputs[i].NumInList));
        }
        if(inputs[i].type.equals("select"))
        {
            FSMGBL.setConstraints(RadioButtons.get(inputs[i].NumInList), FSMGBLC);
            p.add(RadioButtons.get(inputs[i].NumInList));
        }
        if(inputs[i].type.equals("dirselector"))
        {
            FSMGBL.setConstraints(FileChoosers.get(inputs[i].NumInList), FSMGBLC);
            p.add(FileChoosers.get(inputs[i].NumInList));
        }
    }
    p.revalidate();
    RefreshCommandLine();
}

public class FSMInput {
    String name; // название параметра, например number of inputs
    String type; // тип входа (range, select, flag,dirselector)
    static для неизменяемых значений

```

```

        ArrayList<String> info;//список параметров зависящий от типа.
        //max/min значения для range в разных элементах
        // в случае с RadioButtons содержит пары(в двух элементах)
названия параметров и соответствующий аргумент для командной строки
        // для флага значения, передаваемые exe файлу в обоих случаях
        // для файла - строку file или dir, т е что должно передаваться
файл или директория ДОБАВИТЬ В XML???
        int NumInList;//номер в списке элеменов, например в Spinners
        boolean inflag;// 1, если изменение аргумента может повлиять на
исключение(затемнение) другого элемента, иначе 0
        //т.е., 0-ой элемент = значению аргумента, при котором аргумент с
названием из 1-го элемента затемняется и т.д. с 2-ым и 3-им.
        FSMInput(String name_i, String type_i, ArrayList<String> info_i,
boolean f)
    {
        name = name_i;
        type = type_i;
        info = new ArrayList<>(info_i);
        inflag = f;
    }
}

void RefreshCommandLine()
{
    ArrayList<String> StrArg = new ArrayList<String>();
    String CL = "";
    int n=0;
    int k=0;
    for(int i=0;i<CommandLine.size();i++)
    {
        StrArg.add(CommandLine.get(i).get(0));
        for(int j = 1;j<CommandLine.get(i).size();j++)
        {
            n=0;
            for(k = 0;k < inputs.length;k++)
            {
                if(inputs[k].name.equals(CommandLine.get(i).get(j)))
                {
                    n=k;
                    k=inputs.length+1;
                }
            }
            switch(inputs[n].type)
            {
            case "const":
            {
                StrArg.add(inputs[n].info.get(0));
                break;
            }
            case "range":
            {
                StrArg.add(Spinners.get(inputs[n].NumInList).val);
                break;
            }
            case "flag":
            {
                if(CheckBoxes.get(inputs[n].NumInList).val.equals("checked"))
                {
                    StrArg.add(inputs[n].info.get(0));
                }
            }
            }
        }
    }
}

```

```

        else
        {
            StrArg.add(inputs[n].info.get(1));
        }
        break;
    }
    case "select":
    {
        for(int l = 0;l <
RadioButtons.get(inputs[n].NumInList).RadioButtons.length*2;l=l+2)
        {

            if(RadioButtons.get(inputs[n].NumInList).val.equals(inputs[n].info.get(
1)))
            {

                StrArg.add(inputs[n].info.get(l+1));
            }
            break;
        }
        case "dirselector":
        {

            StrArg.add(FileChoosers.get(inputs[n].NumInList).val);
            break;

        }
        }
        if(StrArg.get(StrArg.size()-1).equals(""))
        {
            StrArg.remove(StrArg.size()-1);
        }
    }
    for(int as = 0;as < StrArg.size();as++)
    {
        CL += StrArg.get(as);
        CL += " ";
    }
    CL += "\n";
    StrArg.clear();
    Comline.setText(CL);
}

class FSMValListener implements PropertyChangeListener {

    public void propertyChange(PropertyChangeEvent event) {
        FSMVerifSwitchCon();
        RefreshCommandLine();
    }

}

```



## ПРИЛОЖЕНИЕ Б

### Код шаблонного XML-документа

```
<?xml version="1.0" encoding="UTF-8"?>
<namespace>
  <image>
    <atribute name="url"/>
  </image>
  <range>
    <atribute name="name"/>
    <atribute name="var"/>
    <node name="min">
      <atribute name="val"/>
    </node>
    <node name="max">
      <atribute name="val"/>
    </node>
  </range>
  <select>
    <atribute name="name"/>
    <atribute name="var"/>
    <node name="variant">
      <atribute name="name"/>
      <atribute name="val"/>
    </node>
  </select>
  <flag>
    <atribute name="name"/>
    <atribute name="var"/>
    <atribute name="checked"/>
    <atribute name="not_checked"/>
  </flag>
  <if>
    <node name="conditions">
      <node name="condition">
        <atribute name="var1"/>
        <atribute name="symbol"/>
        <atribute name="var2"/>
      </node>
    </node>
    <node name="then">
      <node name="enabled">
        <node name="param">
          <atribute name="val"/>
        </node>
      </node>
      <node name="not_enabled">
        <node name="param">
          <atribute name="val"/>
        </node>
      </node>
    </node>
    <node name="else">
      <node name="enabled">
        <node name="param">
          <atribute name="val"/>
        </node>
      </node>
      <node name="not_enabled">
        <node name="param">
          <atribute name="val"/>
        </node>
      </node>
    </node>
  </if>
</namespace>
```

```

        </node>
    </node>
</if>
<dirselector>
    <atribute name="name"/>
    <atribute name="var"/>
    <atribute name="type"/>
</dirselector>
<commands>
    <node name="commandline">
        <node name="command">
            <atribute name="val"/>
        </node>
        <node name="param">
            <atribute name="val"/>
        </node>
        <node name="static_param">
            <atribute name="val"/>
        </node>
    </node>
</commands>
<const>
    <atribute name="name"/>
    <atribute name="var"/>
    <atribute name="val"/>
</const>
<conditions>
    <node name="condition">
        <atribute name="var1"/>
        <atribute name="symbol"/>
        <atribute name="var2"/>
    </node>
</conditions>
</namespace>

```

## ПРИЛОЖЕНИЕ В

### XML-документы, описывающие окна интерфейса

```
<?xml version="1.0" encoding="UTF-8"?>
<tool name="Deriving exhaustive">
  <image url="\image.gif"/>
  <dirselector name="Deterministic" var="destdir" type="file"/>
  <select name="Test suite" var="type">
    <variant name="W" val="1"/>
    <variant name="Wp" val="4"/>
    <variant name="HSI" val="2"/>
    <variant name="H" val="3"/>
  </select>
  <select name="Upper bound(m) on the number of states of an
implementation" var="type1">
    <variant name="the same as in the specidcation FSM (m=n) "
val="0"/>
    <variant name="greater than in the specidcation FSM (m>n) "
val=""/>
  </select>
  <range name="m=" var="in">
    <min val="1"/>
    <max val="100"/>
  </range>
  <const name="zero" var="z1" val="0"/>
  <if>
    <conditions>
      <condition var1="type1" symbol="==" var2="the same as in
the specidcation FSM (m=n)"/>
    </conditions>
    <then>
      <enabled>
        <param val="destdir"/>
        <param val="destdir_1"/>
        <param val="type"/>
        <param val="type1"/>
      </enabled>
      <not_enabled>
        <param val="in"/>
      </not_enabled>
    </then>
    <else>
      <enabled>
        <param val="destdir"/>
        <param val="destdir_1"/>
        <param val="type"/>
        <param val="in"/>
      </enabled>
      <not_enabled>
      </not_enabled>
    </else>
  </if>
  <dirselector name="File for saving the test suite" var="destdir_1"
type="dir"/>
  <commands>
    <commandline>
      <command val="\test_fsm.exe"/>
      <param val="destdir"/>
      <param val="destdir_1"/>
      <param val="type"/>
      <param val="type1"/>
      <param val="in"/>
    </commandline>
  </commands>
</tool>
```

```

        </commandline>
    </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="Deriving exhaustive">
    <image url="\image.gif"/>
    <dirselector name="Specification FSM 1" var="destdir"
type="dir"/>
    <dirselector name="Specification FSM 2" var="destdir_1"
type="dir"/>
    <dirselector name="File for saving the test suite:"
var="destdir_2" type="dir" />
    <commands>
        <commandline>
            <command val="\Livelock_test.exe"/>
            <param val="destdir"/>
            <param val="destdir_1"/>
        </commandline>
    </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="Generetor of componet machines for the composition of two FSMs">
    <image url="\image.gif"/>

    <range name="Number of external I1" var="inputs_I1">
        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Number of external outputs O1" var="outputs_O1">
        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Number of states1" var="states1">
        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Number of internal outputs U " var="outputs_U">
        <min val="1"/>
        <max val="100"/>
    </range>

    <range name="Number of external I2" var="inputs_I2">
        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Number of external outputs O2" var="outputs_O2">
        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Number of states2" var="states2">
        <min val="2"/>
        <max val="100"/>
    </range>
    <range name="Number of internal outputs V " var="outputs_V">
        <min val="1"/>
        <max val="100"/>
    </range>
    <select name="% of the undefined transitions (on)" var="type1">
        <variant name="off" val="0"/>
        <variant name="on" val="1"/>
    </select>

```

```

        </select>
        <range name="% of the undefined transitions" var="transitions">
        <min val="1"/>
        <max val="100"/>
        </range>
        <const name="zero" var="z1" val="0"/>

        <dirselector name="File for saving the specification 1"
var="destdir" type="dir"/>
        <dirselector name="File for saving the specification 2"
var="destdir1" type="dir"/>
        <if>
        <conditions>
            <condition var1="type1" symbol="==" var2="off"/>
        </conditions>
        <then>
            <enabled>
                <param val="type1"/>
                <param val="inputs_I1"/>
                <param val="outputs_O1"/>
                <param val="states1"/>
                <param val="inputs_I2"/>
                <param val="outputs_O2"/>
                <param val="states2"/>
                <param val="outputs_U"/>
                <param val="outputs_V"/>
                <param val="transitions"/>
                <param val="destdir"/>
                <param val="destdir1"/>
            </enabled>
            <not_enabled>
                <param val="transitions"/>
            </not_enabled>
        </then>
        <else>
            <enabled>
                <param val="type1"/>
                <param val="inputs_I1"/>
                <param val="outputs_O1"/>
                <param val="states1"/>
                <param val="inputs_I2"/>
                <param val="outputs_O2"/>
                <param val="states2"/>
                <param val="outputs_U"/>
                <param val="outputs_V"/>
                <param val="transitions"/>
                <param val="destdir"/>
                <param val="destdir1"/>
            </enabled>
            <not_enabled>
        <param val="z1"/>
            </not_enabled>
        </else>
    </if>

    <commands>
        <commandline>
            <command val="\Generator_Specs.exe"/>
            <param val="type1"/>
            <param val="inputs_I1"/>
            <param val="outputs_O1"/>
            <param val="states1"/>
            <param val="inputs_I2"/>

```

```

        <param val="outputs_02"/>
        <param val="states2"/>
        <param val="outputs_U"/>
        <param val="outputs_V"/>
        <param val="transitions"/>
        <param val="destdir"/>
        <param val="destdir1"/>
    </commandline>
</commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="Invariants">
    <image url="\image.gif"/>
    <dirselector name="Specification of the" var="destdir" type="file"/>
    <select name="Component number" var="type">
        <variant name="1" val="1"/>
        <variant name="2" val="2"/>
    </select>
    <range name="Nondeterminism degree (lower bound on the transition
number" var="number">
        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Nondeterminism degree ( upper bound on the
transition number1" var="number1">
        <min val="1"/>
        <max val="100"/>
    </range>
    <conditions>
        <condition var1="number" symbol="&lt;" var2="number1"/>
    </conditions>
    <dirselector name="File for saving invariants:" var="destdir_1"
type="dir"/>
    <const name="tmp" var="tmp" val="tmp4A43.tmp"/>
    <commands>
    <commandline>
        <command val="\DeriveObservableMM.exe"/>
        <param val="destdir"/>
        <param val="tmp"/>
        <param val="type"/>
        <param val="number"/>
        <param val="number1"/>
    </commandline>
    <commandline>
        <command val="\invariants.exe"/>
        <param val="destdir_1"/>
        <param val="tmp"/>
        <param val="destdir_1"/>
    </commandline>
    </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="Test derivation for component FSMs">
    <image url="\image.gif"/>
    <dirselector name="Component FSM specification" var="destdir" type="file"/>
    <dirselector name="Component FSM specification" var="destdir1" type="file"/>
    <select name="Faulty component (1 of 2)" var="type">
        <variant name="one" val="1"/>
        <variant name="two" val="2"/>
    </select>
    <range name="% of transitions for fault injecting" var="transitions">

```

```

        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Nondeterminism range of faulty transitions in the mutation
FSM(lower)" var="number">
        <min val="1"/>
        <max val="100"/>
    </range>
    <range name="Nondeterminism range of faulty transitions in the
mutation FSM(upper)" var="number1">
        <min val="1"/>
        <max val="100"/>
    </range>
    <const name="tmp" var="tmp" val="tmp.tmp"/>
    <const name="tmp1" var="tmp1" val="tmp1.tmp"/>
    <const name="tmp2" var="tmp2" val="tmp2.tmp"/>
    <const name="tmp3" var="tmp3" val="tmp3.tmp"/>
    <const name="tmp4" var="tmp4" val="tmp4.tmp"/>
    <const name="tmp5" var="tmp5" val="tmp5.tmp"/>
    <conditions>
    <condition var1="number" symbol="#x3C" var2="number1"/>
    </conditions>
    <dirselector name="Specification" var="destdir2" type="dir"/>
    <commands>
        <commandline>
            <command val="\Compose_Spec_MM.exe"/>
            <param val="destdir"/>
            <param val="destdir1"/>
            <param val="tmp"/>
            <param val="tmp1"/>
        </commandline>
        <commandline>
            <command val="\Partial_from_Complete.exe"/>
            <param val="destdir"/>
            <param val="tmp2"/>
            <param val="type"/>
            <param val="transitions"/>
        </commandline>
        <commandline>
            <command val="\DeriveObservableMM.exe"/>
            <param val="tmp2"/>
            <param val="tmp3"/>
            <param val="type"/>
            <param val="number"/>
            <param val="number1"/>
        </commandline>
        <commandline>
            <command val="\Compose_Spec_MM.exe"/>
            <param val="destdir"/>
            <param val="tmp3"/>
        </commandline>
        <commandline>
            <command val="\test_fsm.exe"/>
            <param val="tmp"/>
            <param val="tmp4"/>
        </commandline>
        <commandline>
            <command val="\TestConverter.exe"/>
            <param val="tmp4"/>
            <param val="tmp1"/>
            <param val="destdir2"/>
        </commandline>
    </commands>

```

```

</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="FSM generator">
  <image url="\image.gif"/>
  <range name="Number of FSMs being generated" var="fsmnumber">
    <min val="1"/>
    <max val="100"/>
  </range>
  <select name="Type of FSMs being generated" var="type">
    <variant name="classical (not timed)" val=""/>
    <variant name="timed" val="-T"/>
  </select>
  <select name="Deterministic" var="type1">
    <variant name="YES" val=""/>
    <variant name="NO" val="-ND"/>
  </select>
  <flag name="Reduced" var="red1" checked="-R" not_checked=""/>
  <flag name="Observable" var="red2" checked="-O" not_checked=""/>
  <if>
    <conditions>
      <condition var1="type1" symbol="==" var2="YES"/>
    </conditions>
    <then>
      <enabled>
        <param val="type"/>
        <param val="type1"/>
        <param val="red1"/>
        <param val="statesnum"/>
        <param val="inputnum"/>
        <param val="outputnum"/>
        <param val="fsmnumber"/>
        <param val="destdir"/>
      </enabled>
      <not_enabled>
        <param val="red2"/>
      </not_enabled>
    </then>
    <else>
      <enabled>
        <param val="type"/>
        <param val="type1"/>
        <param val="red2"/>
        <param val="statesnum"/>
        <param val="inputnum"/>
        <param val="outputnum"/>
        <param val="fsmnumber"/>
        <param val="destdir"/>
      </enabled>
      <not_enabled>
        <param val="red1"/>
      </not_enabled>
    </else>
  </if>
  <range name="Number of states" var="statesnum">
    <max val="100"/>
    <min val="1"/>
  </range>
  <range name="Number of inputs" var="inputnum">
    <max val="100"/>
    <min val="1"/>
  </range>
  <range name="Number of outputs" var="outputnum">

```



```

        <max val="100"/>
        <min val="1"/>
    </range>
    <dirselector name="Folder for FSMs being generated" var="destdir"
type="dir"/>
    <commands>
        <commandline>
            <command val="\gen_fsm.exe"/>
            <param val="type"/>
            <param val="type1"/>
            <param val="red1"/>
            <param val="red2"/>
            <param val="statesnum"/>
            <param val="inputnum"/>
            <param val="outputnum"/>
            <param val="fsmnumber"/>
            <param val="destdir"/>
        </commandline>
    </commands>
</tool>

<?xml version = "1.0" encoding = "UTF-8"?>
<tool name = "InterFSM">
    <dirselector name = "Path to the file with first FSM (not timed)" var =
"dir1" type = "file"/>
    <dirselector name = "Path to the file with second FSM (not timed)" var
= "dir2" type = "file"/>
    <range name = "Initial state of the first FSM" var = "first_state">
        <min val = "0"/>
        <max val = "100"/>
    </range>
    <range name = "Initial state of the second FSM" var = "two_state">
        <min val = "0"/>
        <max val = "100"/>
    </range>
    <dirselector name = "Folder for result" var = "dir3" type = "dir"/>
    <commands>
        <commandline>
            <command val = "\InterFSM.exe"/>
            <param val = "dir1"/>
            <param val = "dir2"/>
            <param val = "first_state"/>
            <param val = "two_state"/>
            <param val = "dir3"/>
        </commandline>
    </commands>
</tool>

<?xml version = "1.0" encoding = "UTF-8"?>
<tool name = "InterFSM">
    <dirselector name = "Path to the file with FSM" var = "dir1" type =
"file"/>
    <const name = "three" var = "const1" val = "3"/>
    <commands>
        <commandline>
            <command val = "\Synch_Seq.exe"/>
            <param val = "const1"/>
            <param val = "dir1"/>
        </commandline>
    </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>

```

```

<tool name="Test derivation w.r.t. the non-separability relation">
  <image url=".\\image.gif"/>
  <dirselector name="Specification FSM:" var="destdir"
type="file"/>
  <dirselector name="File for saving the test suite:"
var="destdir_1" type="dir"/>

  <commands>
    <commandline>
      <command val="\testimprove.exe"/>
      <param val="destdir"/>
      <param val="destdir_1"/>
    </commandline>
  </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="Test derivation w.r.t. the reduction relation">
  <image url=".\\image.gif"/>
  <dirselector name="Observable FSM specification:" var="destdir"
type="file"/>
  <dirselector name="File for saving the test:"
var="destdir_1" type="dir"/>

  <commands>
    <commandline>
      <command val="\-jar ndfsm_test.jar"/>
      <param val="destdir"/>
      <param val="destdir_1"/>
    </commandline>
  </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="Distinguishing sequences">
  <image url=".\\image.gif"/>
  <dirselector name="Specification of a time FSM" var="destdir" type="file"/>
  <dirselector name="Folder of with FSM implementetions" var="destdir1"
type="dir"/>
  <select name="Result" var="type1">
    <variant name="display on screen" val=""/>
    <variant name="seve of file" val=""/>
  </select>
  <dirselector name="Save" var="destdir2" type="dir"/>
  <if>
    <conditions>
      <condition var1="type1" symbol="==" var2="display on
screen"/>
    </conditions>
    <then>
      <enabled>
        <param val="destdir"/>
        <param val="destdir1"/>
        <param val="save"/>
        <param val="type1"/>
      </enabled>
      <not_enabled>
        <param val="destdir2"/>
      </not_enabled>
    </then>
    <else>
      <enabled>

```

```

        <param val="destdir"/>
        <param val="destdir1"/>
        <param val="save"/>
        <param val="destdir2"/>
    </enabled>
    <not_enabled>

    </not_enabled>
</else>
</if>
    <commands>
        <commandline>
            <command val="\cmp_dtfsm.exe"/>
            <param val="destdir"/>
            <param val="destdir1"/>
            <param val="save"/>
            <param val="destdir2"/>
        </commandline>
    </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="r-distuinguishing FSMs">
    <image url="\image.gif"/>
    <dirselector name="Specification of a timed FSM" var="destdir"
type="file"/>
    <dirselector name="Folder for timed FSM implementetions"
var="destdir_1" type="dir"/>
    <dirselector name="Folder for r-distinguishing FSMs" var="destdir_2"
type="dir"/>
    <commands>
        <commandline>
            <command val="\r_ndtfsm.exe"/>
            <param val="destdir"/>
            <param val="destdir_1"/>
            <param val="destdir_2"/>
        </commandline>
    </commands>
</tool>

<?xml version="1.0" encoding="UTF-8"?>
<tool name="Test suites for timed FSMs">
    <image url="\image.gif"/>

    <dirselector name="Timed FSM specification" var="destdir" type="file"/>
    <select name="Test suite" var="type">
        <variant name="W" val="1"/>
        <variant name="HSI" val="2"/>
        <variant name="H" val="3"/>
    </select>
    <const name="zero" var="z1" val="0"/>
    <const name="tmp1" var="tmp1" val="tmp1.tmp"/>
    <const name="tmp2" var="tmp2" val="tmp2.tmp"/>
    <const name="tmp3" var="tmp3" val="tmp3.tmp"/>
    <dirselector name="File for saving the test" var="destdir_1"
type="dir"/>
    <commands>
        <commandline>
            <command val="\tfsm-fsm.exe"/>
            <param val="destdir"/>
            <param val="tmp1"/>
        </commandline>
        <commandline>

```

```
        <command val="\test_fsm.exe "/>
        <param val="tmp2"/>
        <param val="type"/>
    <param val="z1"/>
    </commandline>
    <commandline>
    <command val="\test_tfsm.exe"/>
    <param val="tmp3"/>
    <param val="destdir_1"/>
    </commandline>
</commands>
</tool>
```

## ПРИЛОЖЕНИЕ Г

### Код теста

```
package automata;

import static org.junit.Assert.*;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
/* Imports, provided by the User */
import java.io.*;
import automata.Automata;
import automata.State;
import java.util.ArrayList;

public class test2 {
    Automata SUT = null;
    byte byteResult;
    char charResult;
    double doubleResult;
    float floatResult;
    int intResult;
    long longResult;
    short shortResult;
    boolean booleanResult;
    String StringResult;
    Object ObjectResult;

    /* Extra definitions provided by the user */
    int sizeIO;

    @Before
    public void setUp() throws Exception {
        /* Initicialization is provided by the user. */
        //TODO Make sure, that SUT is defined and callable.

        try {
            SUT = new Automata();
            State state = new State();
            state.setName("state0");
            Switch[] switches = new Switch[1];
            switches[0] = new Switch("1", "1");
            state.addSwitch("1", switches);
            SUT.addState(state);
            sizeIO = 1;
        } catch (Exception e) {
            fail("An exception during initialization");
            e.printStackTrace();
        }
    }

    @After
    public void tearDown() throws Exception {
        SUT = null;
    }

    @Test
    public void testCaseNumber0() {
        /* Input-Output pair number 0 */
        try {
            ObjectResult = SUT.getInputs(); String[] inputs =
            (String[]) ObjectResult; int size = inputs.length;
```

```

        assertTrue("Test case number 0, io pair number 0", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 0, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

}

@Test
public void testCaseNumber1() {
    /* Input-Output pair number 0 */
    try {
        ObjectResult = SUT.getOutputs(); String[] outputs =
(String[]) ObjectResult; int size = outputs.length;
        assertTrue("Test case number 1, io pair number 0", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 1, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

}

@Test
public void testCaseNumber2() {
    /* Input-Output pair number 0 */
    try {
        SUT.isPartible(); State state = new State();
state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
++sizeIO;
        assertTrue("Test case number 2, io pair number 0", (true));
    }
    catch(Exception e) {
        assertTrue("For test case number 2, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 1 */
    try {
        ObjectResult = SUT.getInputs(); String[] inputs =
(String[]) ObjectResult; int size = inputs.length;
        assertTrue("Test case number 2, io pair number 1", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 2, io pair number 1 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

}

@Test
public void testCaseNumber3() {
    /* Input-Output pair number 0 */
    try {

```

```

        ObjectResult = SUT.getInputs(); String[] inputs =
(String[]) ObjectResult; int size = inputs.length;
        assertTrue("Test case number 3, io pair number 0", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 3, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 1 */
    try {
        SUT.isPartible(); State state = new State();
state.setName("statel"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
++sizeIO;

        assertTrue("Test case number 3, io pair number 1", (true));
    }
    catch(Exception e) {
        assertTrue("For test case number 3, io pair number 1 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 2 */
    try {
        ObjectResult = SUT.getInputs(); String[] inputs =
(String[]) ObjectResult; int size = inputs.length;
        assertTrue("Test case number 3, io pair number 2", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 3, io pair number 2 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

}

@Test
public void testCaseNumber4() {
    /* Input-Output pair number 0 */
    try {
        SUT.isPartible(); State state = new State();
state.setName("statel"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
++sizeIO;

        assertTrue("Test case number 4, io pair number 0", (true));
    }
    catch(Exception e) {
        assertTrue("For test case number 4, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 1 */
    try {
        ObjectResult = SUT.getOutputs(); String[] outputs =
(String[]) ObjectResult; int size = outputs.length;
        assertTrue("Test case number 4, io pair number 1", (size ==
sizeIO));
    }

```

```

        catch(Exception e) {
            assertTrue("For test case number 4, io pair number 1 an
exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }

    }

    @Test
    public void testCaseNumber5() {
        /* Input-Output pair number 0 */
        try {
            ObjectResult = SUT.getOutputs(); String[] outputs =
(String[]) ObjectResult; int size = outputs.length;
            assertTrue("Test case number 5, io pair number 0", (size ==
sizeIO));
        }
        catch(Exception e) {
            assertTrue("For test case number 5, io pair number 0 an
exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }

        /* Input-Output pair number 1 */
        try {
            SUT.isPartible(); State state = new State();
state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
++sizeIO;

            assertTrue("Test case number 5, io pair number 1", (true));
        }
        catch(Exception e) {
            assertTrue("For test case number 5, io pair number 1 an
exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }

        /* Input-Output pair number 2 */
        try {
            ObjectResult = SUT.getOutputs(); String[] outputs =
(String[]) ObjectResult; int size = outputs.length;
            assertTrue("Test case number 5, io pair number 2", (size ==
sizeIO));
        }
        catch(Exception e) {
            assertTrue("For test case number 5, io pair number 2 an
exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }

    }

    @Test
    public void testCaseNumber6() {
        /* Input-Output pair number 0 */
        try {
            SUT.isPartible(); State state = new State();
state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
++sizeIO;

            assertTrue("Test case number 6, io pair number 0", (true));
        }
        catch(Exception e) {

```



```

        assertTrue("For test case number 6, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 1 */
    try {
        SUT.isPartible(); State state = new State();
state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("3", "3"); state.addSwitch("3", switches); SUT.addState(state);
++sizeIO;

        assertTrue("Test case number 6, io pair number 1", (true));
    }
    catch(Exception e) {
        assertTrue("For test case number 6, io pair number 1 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 2 */
    try {
        ObjectResult = SUT.getInputs(); String[] inputs =
(String[]) ObjectResult; int size = inputs.length;
        assertTrue("Test case number 6, io pair number 2", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 6, io pair number 2 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

}

@Test
public void testCaseNumber7() {
    /* Input-Output pair number 0 */
    try {
        ObjectResult = SUT.getInputs(); String[] inputs =
(String[]) ObjectResult; int size = inputs.length;
        assertTrue("Test case number 7, io pair number 0", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 7, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 1 */
    try {
        SUT.isPartible(); State state = new State();
state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
++sizeIO;

        assertTrue("Test case number 7, io pair number 1", (true));
    }
    catch(Exception e) {
        assertTrue("For test case number 7, io pair number 1 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

}

```

```

        /* Input-Output pair number 2 */
        try {
            SUT.isPartiple(); State state = new State();
            state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
            Switch("3", "3"); state.addSwitch("3", switches); SUT.addState(state);
            ++sizeIO;

            assertTrue("Test case number 7, io pair number 2", (true));
        }
        catch(Exception e) {
            assertTrue("For test case number 7, io pair number 2 an
            exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }

        /* Input-Output pair number 3 */
        try {
            ObjectResult = SUT.getInputs(); String[] inputs =
            (String[]) ObjectResult; int size = inputs.length;
            assertTrue("Test case number 7, io pair number 3", (size ==
            sizeIO));
        }
        catch(Exception e) {
            assertTrue("For test case number 7, io pair number 3 an
            exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }
    }

    @Test
    public void testCaseNumber8() {
        /* Input-Output pair number 0 */
        try {
            SUT.isPartiple(); State state = new State();
            state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
            Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
            ++sizeIO;

            assertTrue("Test case number 8, io pair number 0", (true));
        }
        catch(Exception e) {
            assertTrue("For test case number 8, io pair number 0 an
            exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }

        /* Input-Output pair number 1 */
        try {
            SUT.isPartiple(); State state = new State();
            state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
            Switch("3", "3"); state.addSwitch("3", switches); SUT.addState(state);
            ++sizeIO;

            assertTrue("Test case number 8, io pair number 1", (true));
        }
        catch(Exception e) {
            assertTrue("For test case number 8, io pair number 1 an
            exception has occured, but should not be" + e.getMessage(), false);
            e.printStackTrace();
        }

        /* Input-Output pair number 2 */
        try {
            ObjectResult = SUT.getOutputs(); String[] outputs =
            (String[]) ObjectResult; int size = outputs.length;

```

```

        assertTrue("Test case number 8, io pair number 2", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 8, io pair number 2 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

}

@Test
public void testCaseNumber9() {
    /* Input-Output pair number 0 */
    try {
        ObjectResult = SUT.getOutputs(); String[] outputs =
(String[]) ObjectResult; int size = outputs.length;
        assertTrue("Test case number 9, io pair number 0", (size ==
sizeIO));
    }
    catch(Exception e) {
        assertTrue("For test case number 9, io pair number 0 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 1 */
    try {
        SUT.isPartible(); State state = new State();
state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("2", "2"); state.addSwitch("2", switches); SUT.addState(state);
++sizeIO;

        assertTrue("Test case number 9, io pair number 1", (true));
    }
    catch(Exception e) {
        assertTrue("For test case number 9, io pair number 1 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 2 */
    try {
        SUT.isPartible(); State state = new State();
state.setName("state1"); Switch[] switches = new Switch[1]; switches[0] = new
Switch("3", "3"); state.addSwitch("3", switches); SUT.addState(state);
++sizeIO;

        assertTrue("Test case number 9, io pair number 2", (true));
    }
    catch(Exception e) {
        assertTrue("For test case number 9, io pair number 2 an
exception has occurred, but should not be" + e.getMessage(), false);
        e.printStackTrace();
    }

    /* Input-Output pair number 3 */
    try {
        ObjectResult = SUT.getOutputs(); String[] outputs =
(String[]) ObjectResult; int size = outputs.length;
        assertTrue("Test case number 9, io pair number 3", (size ==
sizeIO));
    }
    catch(Exception e) {

```

```
        assertTrue("For test case number 9, io pair number 3 an  
exception has occurred, but should not be" + e.getMessage(), false);  
        e.printStackTrace();  
    }  
}  
}
```

# Отчет о проверке на заимствования №1

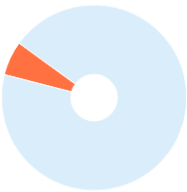
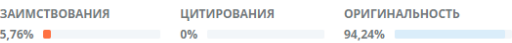
Автор: [pride080993@gmail.com](mailto:pride080993@gmail.com) / ID: 3487488  
Проверяющий: [pride080993@gmail.com](mailto:pride080993@gmail.com) / ID: 3487488  
Отчет предоставлен сервисом «Антиплагиат»- <http://www.antiplagiat.ru>

## ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 5  
Начало загрузки: 15.06.2018 08:39:28  
Длительность загрузки: 00:00:00  
Имя исходного файла:  
\_дисер\_Батрацкий\_14\_06\_конечный  
Размер текста: 125 кБ  
Символов в тексте: 114106  
Слов в тексте: 14613  
Число предложений: 1265

## ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)  
Начало проверки: 15.06.2018 08:39:29  
Длительность проверки: 00:00:02  
Комментарии: не указано  
Модули поиска:



Заимствования — доля всех найденных текстовых пересечений, за исключением тех, которые система отнесла к цитированиям, по отношению к общему объему документа.  
Цитирования — доля текстовых пересечений, которые не являются авторскими, но система посчитала их использование корректным, по отношению к общему объему документа. Сюда относятся оформленные по ГОСТу цитаты; общеупотребительные выражения; фрагменты текста, найденные в источниках из коллекций нормативно-правовой документации.  
Текстовое пересечение — фрагмент текста проверяемого документа, совпадающий или почти совпадающий с фрагментом текста источника.  
Источник — документ, проиндексированный в системе и содержащийся в модуле поиска, по которому проводится проверка.  
Оригинальность — доля фрагментов текста проверяемого документа, не обнаруженных ни в одном источнике, по которым шла проверка, по отношению к общему объему документа.  
Заимствования, цитирования и оригинальность являются отдельными показателями и в сумме дают 100%, что соответствует всему тексту проверяемого документа.  
Обращаем Ваше внимание, что система находит текстовые пересечения проверяемого документа с проиндексированными в системе текстовыми источниками. При этом система является вспомогательным инструментом, определение корректности и правомерности заимствований или цитирований, а также авторства текстовых фрагментов проверяемого документа остается в компетенции проверяющего.

| №    | Доля в отчете | Доля в тексте | Источник                                             | Ссылка                                                      | Актуален на | Модуль поиска          | Блоков в отчете | Блоков в тексте |
|------|---------------|---------------|------------------------------------------------------|-------------------------------------------------------------|-------------|------------------------|-----------------|-----------------|
| [01] | 0,77%         | 1,83%         | Проверка переходов в расширенном автомате на осно... | <a href="http://sun.tsu.ru">http://sun.tsu.ru</a>           | 16 Ноя 2012 | Модуль поиска Интернет | 4               | 12              |
| [02] | 1%            | 1,79%         | Проверка переходов в расширенном автомате наоснов... | <a href="http://journals.tsu.ru">http://journals.tsu.ru</a> | 08 Окт 2016 | Модуль поиска Интернет | 7               | 13              |
| [03] | 0,55%         | 1,04%         | Download full volume                                 | <a href="http://ispras.ru">http://ispras.ru</a>             | 26 Окт 2017 | Модуль поиска Интернет | 10              | 15              |

Еще источников: 11  
Еще заимствований: 3,46%