Трансляция описаний конечных автоматов, представленных в формате Microsoft Visio, в исходный код на языке C

Леонид Столяров[*]
8 класс, лицей «Вторая школа», Москва

Научный руководитель: И. Р. Дединский, лицей «Вторая школа», Москва
Консультант: д.т.н., проф. А. А. Шалыто, ИТМО, Санкт-Петербург

Целью работы являлось создание инструментального средства для трансляции описаний переходных графов конечных автоматов, представленных в формате MS Visio в исходный программный код на языке C. Главной задачей данного проекта является устранение недостатков других подобных средств.

В качестве языка программирования был выбран язык Microsoft C#, предназначенный для платформы .NET, потому что он обеспечивал хорошую поддержку COM технологии, которая использовалась при взаимодействии с MS Visio. Интерфейсы COM объекта MS Visio позволяют открывать любые файлы, сохранённые в формате редактора Visio и получить список всех элементов схемы в открытом файле.

Программа транслятора может быть запущена в двух режимах, выбираемых ключом командной строки. В одном из режимов трансляция производится интерактивно, и вся информация считывается из командной строки. Этот режим удобен при использовании в средствах автоматической сборки (make, менеджер проектов MS Visual Studio). В другом режиме интерфейсом программы служит диалоговое окно.

В результате работы было создано инструментальное средство для трансляции графов переходов автоматов, представленных в формате Microsoft Visio в исходный программный код на языке C.

Планируется совершенствовать гибкость инструментального средства, в частности, добавить трансляцию в различные языки программирования.

Существует подход к созданию программ для систем логического управления с использованием конечных автоматов [1]. В последнее время этот подход сильно развивается, о чём свидетельствуют его многочисленные применения [3]. С его помощью удобно реализуется различные системы логического и событийного управления. Автоматное программирование, иначе называемое «программирование от состояний» основывается на использовании конечных автоматов для описания логики работы программ. Конечный автомат – это конечное множество состояний, в которых он может находиться и переходов между ними.

Подход основан на создании графов переходов автоматов в виде автоматных схем и, затем трансляции этих схем в исходный программный код (так называемая switch-технология [1]). Важное достоинство автоматного программирования заключается в возможности разделения этапов проектирования и реализации программы, в том числе и для того, чтобы эти этапы могли выполняться различными людьми. Автоматная схема является графическим отражением алгоритма, поэтому при разработке программ на основе автоматов она документирует этот алгоритм в неком общепринятом формате, что тоже не маловажно. Важная особенность автоматного подхода состоит также и в том, что он – один из немногих, который может быть формально верифицирован, в отличие от разработок, основанных на императивном или объектно-ориентированном подходе.

[*] E-mail для связи: lnd1212@rambler.ru.

Визуализация логики программы позволяет ускорить ее разработку и отладку, а также отделить алгоритмическую (логическую, управляющую) часть от остального кода. В автоматном программировании, как и в теории управления, используется три понятия: *входное воздействие, состояние* и *выходное воздействие.* При этом состояния выделяются на этапе проектирования явным образом.

Для создания графов переходов часто используется распространённые средства графического моделирования, в частности, среда Microsoft Visio. Этот редактор позволяет строить различные схемы, использующие множество элементов, как встроенных, так и загружаемых из внешних файлов шаблонов.

При создании проектов с использованием конечных автоматов часто встаёт проблема трансляции графов переходов в исходный код программы на конкретном языке программирования. Существуют методы как ручной трансляции (кодирование автоматной функции по диаграмме состояний), так и средства автоматической генерации кода. Ручные способы, будучи связаны с человеческим фактором, не могут быть вполне надёжны. Автоматические способы предпочтительнее. Однако в них достаточно важна организация взаимодействия средства трансляции с автором программы (настройка режимов трансляции, способ запуска, сообщения об ошибках трансляции).

Сейчас известно несколько инструментальных средств для поддержки автоматного программирования. Среди них трансляторы Visio2Switch для языка C, MetaAuto для множества языков, задаваемых шаблонами, UniMod для Java.

В этих средствах есть как свои достоинства, так и недостатки. В частности, недостаточна диагностика ошибок в графе переходов и низкая надёжность и качество сгенерированного кода (который иногда не компилируется) у MetaAuto, и невозможность или крайняя сложность интеграции в автоматизированные процессы сборки приложений у Visio2Switch.

Цель и задачи работы

Целью работы являлось создание инструментального средства для трансляции описаний переходных графов автоматов, представленных в формате MS Visio в исходный программный код и другие формы представления данных, например язык XML.

В основные требования, предъявляемые к средствам для работы с автоматами, были включены:

1. Автоматическое документирование логики работы программы.
2. Повышение эффективности отладки и тестирования программ.
3. Повышение эффективности ручного и автоматического кодирования программ.

Среди задач, недостаточно хорошо решённых в других подобных инструментальных средствах можно отметить:

1. Качество и надёжность генерируемого кода.
2. Качественная диагностика ошибок в исходных данных.
3. Интеграция в автоматические процессы сборки приложений.
4. Удобство использования.

Для данного проекта главной задачей является устранение этих недостатков у других подобных средств и расширения возможностей.

В качестве языка программирования был выбран язык Microsoft C#, предназначенный для платформы .NET. Этот язык предоставляет хорошие возможности для работы с COM технологией, необходимой для процесса взаимодействия модулей транслятора с MS Visio. Её поддержка на платформе .NET сильно облегчает этот процесс.

Автоматная схема в редакторе Visio состоит из описания автомата, описаний элементов, собственно состояний автомата и дуг переходов между ними. Этих элементов нет в стандартной библиотеке Visio, поэтому одной из целей являлась разработка файла с шаблонами, необходимыми для построения графа переходов автомата.

Нотация графа перехода автомата

Граф переходов автомата [4] состоит из нескольких основных элементов: состояния, групповые состояния, описание входных и выходных воздействий, переходы (рис. 1).

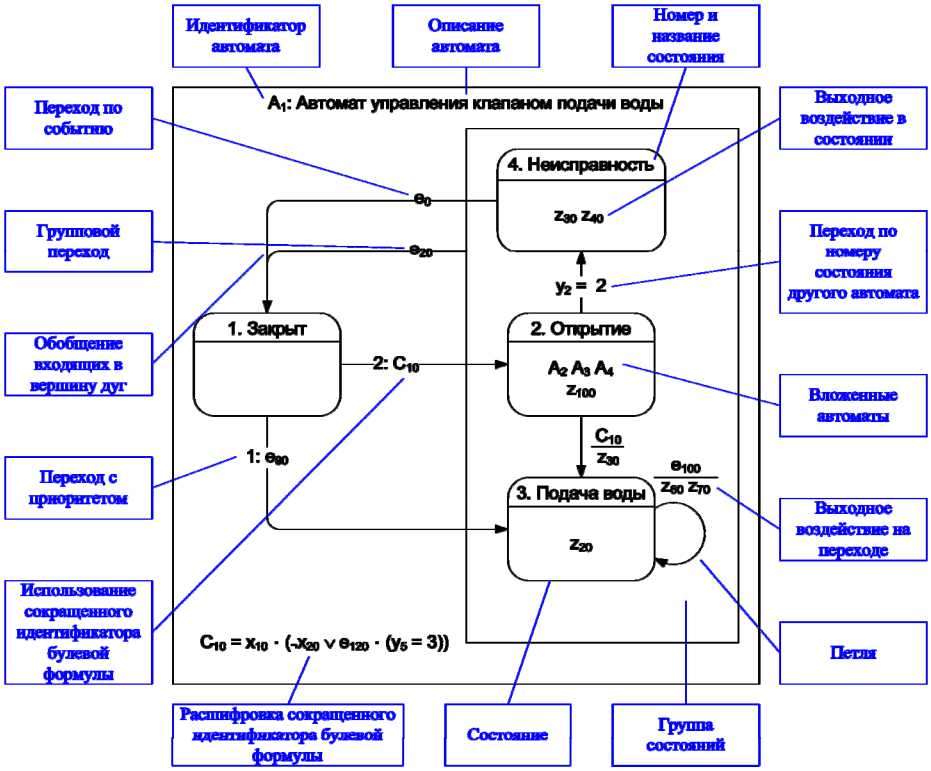

Рис.1. Нотация графов переходов [4].

Рассмотрим основные элементы, используемые для построения графа переходов автомата.

Состояние имеет имя, список других автоматов для запуска с тем же событием и список выходных воздействий для запуска в этом состоянии.

Событие – один из видов входных воздействий, которые могут происходить при вызове автомата, оно чаще всего используется в условиях перехода.

Групповые состояния предназначены для создания групповых переходов и могут содержать в себе состояния или другие группы состояний.

Групповой переход – такой вид перехода, который соединяет группу состояний с состоянием или другой группой состояний.

Дуга перехода соединяет два состояния или группы состояний. У перехода есть условие, представленное в виде произвольного логического выражения с использованием входных данных и воздействий, а также список действий (выходных воздействий), которые должны быть вызваны при переходе.

Описание входных или выходных воздействий – это по сути дела, элемент, комментирующий какое либо входное или выходное воздействие. Имена функций для входных или выходных воздействий чаще всего не являются интуитивно понятными, человек, разбирающийся в описании графа переходов автомата или в сгенерированном исходном программном коде, не может быстро и легко понять и разобраться в том, что делают те или иные функции воздействий. Это затрудняет процесс отладки и совместной разработки программы разными людьми. Поэтому данный элемент является важным для понимания графа переходов автомата. В таблице описаний есть два поля: имя входного или выходного воздействия и словесное описание, соответствующее воздействию. Это словесное описание будет использоваться как при разборе и понимании разработчиком графа переходов автомата, так и для составления комментариев в сгенерированном программном коде.

Важным является элемент *«Описание автомата»*. Он содержит имя автомата и его словесное описание. Имя автомата – это строка, состоящая из английских символов и цифр, которая будет использоваться для идентификации автомата после трансляции. Описание автомата – это любая строка, поясняющая назначение и суть данного автомата.

В целях устранения путаницы и лучшей понятности транслятор считает, что на каждой странице Visio определён один единственный автомат. Для удобства страница Visio должна называться так же, как и автомат, определённый на ней. Поскольку автомат не имеет смысла без имени и хотя бы краткого описания, этот элемент должен обязательно присутствовать на странице Visio, иначе транслятор не будет работать с этим документом, выдав ошибку.

Взаимодействие с MS Visio

Для анализа исходного документа Visio и получение из него данных, можно применить подход, основанный на взаимодействии с самим редактором MS Visio.

Такое взаимодействие можно осуществить при помощи COM-технологии. При установке на компьютер Microsoft Office Visio, устанавливается и соответствующий COM-объект. Интерфейсы COM объекта MS Visio позволяют открывать любые файлы, сохранённые в формате редактора Visio и получить список всех элементов схемы в открытом файле. Поскольку в качестве языка программирования был выбран язык Microsoft C#, предназначенный для платформы .NET, то взаимодействие с COM-объектом Visio происходит через .NET сборку Microsoft.Office.Interop.Visio, которая тоже устанавливается на компьютер вместе с этим COM-объектом.

Взаимодействие с редактором Visio в программе происходит по следующей схеме (см. рис. 2):

1. Создаётся экземпляр программы Visio.
2. В этом экземпляре открывается исходный документ.
3. Анализируется содержание документа.
4. Запускается транслятор, который в процессе своей работы использует данные из исходного документа.
5. После окончания трансляции данные из исходного документа больше не нужны, поэтому работа Visio завершается.

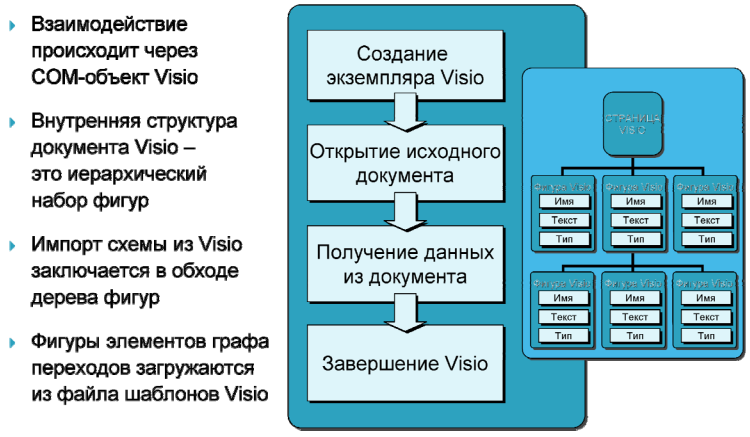

Рис. 2. Взаимодействие с MS Visio.

Внутренняя структура исходного документа Visio – это иерархический набор фигур. У каждой фигуры есть такие параметры, как имя, тип, мастер – имя, фигура-родитель, текст на фигуре.

Согласно структуре исходного документа Visio все элементы графа переходов автомата являются фигурами. Эти фигуры загружаются из файла шаблонов.

Определение типа элемента графа переходов автомата, которым является фигура Visio основано на том, что имя фигуры начинается с префикса, стандартного для конкретного типа шаблонных элементов.

Процесс трансляции

После получения данных из исходного документа Visio происходит их трансляция в исходные коды и различные формы представления данных (язык XML).

Этот процесс происходит в три этапа (см. рис. 4):

1. Преобразование исходных данных Visio в модель, основанную на состояниях и переходах.
2. Генерация файлов формата XML, содержащих описание модели.
3. Генерация файлов с исходным кодом на основе ранее полученной модели.

Как было сказано выше, исходные данные Visio преобразуются в модель, основанную на переходах и состояниях (см. рис. 3). Для представления элементов модели переходов и состояний созданы специальные классы для описания состояний, переходов, автомата и описаний входных или выходных воздействий.

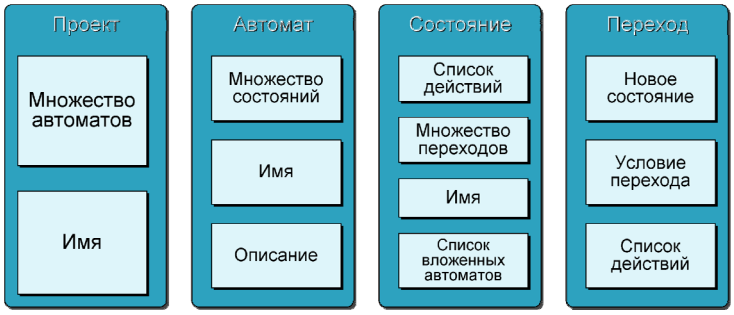

Рис. 3. Внутреннее представление данных. Внутренне исходные данные Visio представляются в виде модели, основанной на переходах и состояниях.

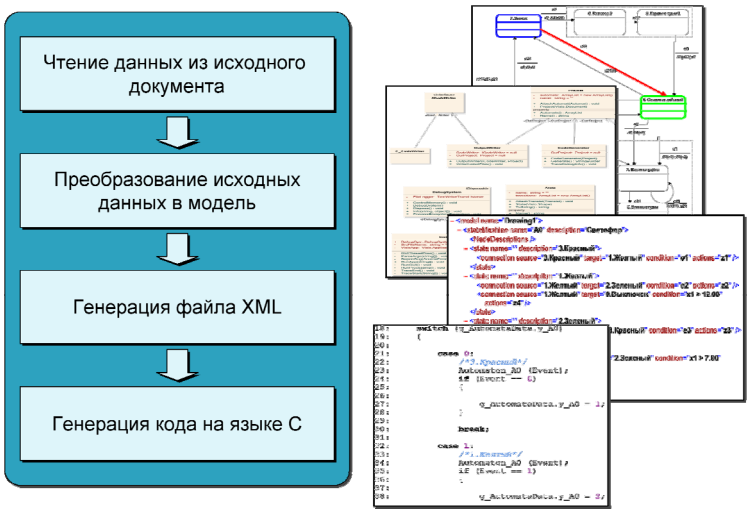

Рис. 4. Алгоритм трансляции.

В классе представления автомата содержится набор состояний, которые есть в этом автомате и описаний входных или выходных воздействий.

В классе представления состояния содержится массив ссылок на переходы, соединяющие данное состояние с другим состоянием того же автомата. Поскольку каждый элемент графа переходов автомата с точки зрения Visio есть фигура, взятая из шаблонного файла, в каждом

классе, представляющем элемент модели, предусмотрен специальный конструктор, создающий этот элемент из соответствующей шаблонной фигуры Visio.

Преобразование данных Visio в модель данных

Рассмотрим подробнее процесс преобразования исходных данных Visio в модель состояний и переходов (см. рис. 4, 5).

Всё начинается с создания объекта автомата. Считается, что если на странице исходного документа есть описание автомата, то он на ней присутствует, и следует рассматривать содержание страницы, как определение графа переходов автомата. Создаётся объект автомата, в который записываются его имя и описание.

Далее список фигур Visio на данной странице анализируется в цикле, где перебираются по очереди все фигуры.

Для каждой фигуры определяется ее тип. Он может быть определён как состояние, соединитель, расширенный соединитель, группа состояний или описание входных или выходных воздействий.

После определения типа фигуры, если фигура – состояние, то из фигуры сгенерируется объект состояния с соответствующими именем, описанием и другими данными. Затем состояние добавляется в список состояний текущего автомата.

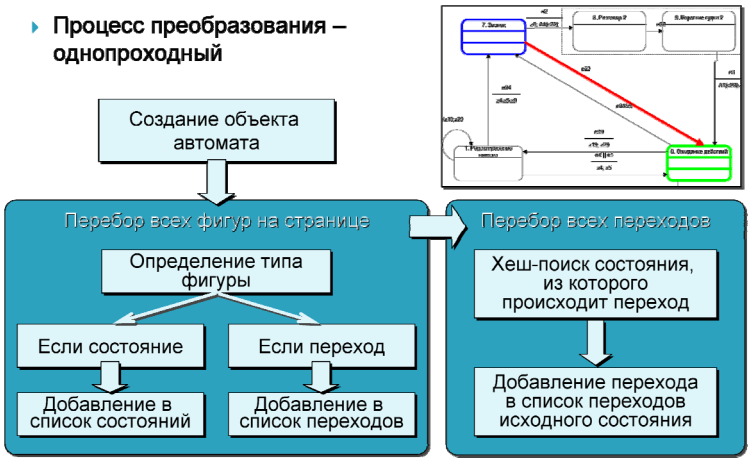

Рис. 5. Генерация модели данных.

Если фигура – описание входного или выходного воздействия, то из неё создаётся объект описания, которое затем добавляется в список описаний воздействий текущего автомата. Оно будет использовано в дальнейшем при генерации кода для автоматического комментирования.

Если фигура – соединение или расширенное соединение, то создаётся объект перехода.

Из исходных данных Visio можно узнать исходную фигуру соединителя, и к какой фигуре он присоединяется. После определения этих двух фигур из них создаются объекты состояний, которые потом указываются в объекте перехода.

После того, как все данные переходов и состояний хранятся в текущем объекте автомата, начинается обработка этих данных. При обработке в список переходов каждого состояния данного автомата добавляется те переходы, которые исходят из этого состояния.

На этом преобразование одного графа переходов автомата в формате Visio в модель, основанную на переходах и состояниях, заканчивается. Аналогичный процесс повторяется для всех других автоматов в исходном документе. Множество преобразованных в модель автоматов хранится в специальном объекте проекта, который также имеет имя. Таким образом, после окончательного преобразования всех автоматов в модель мы имеем объект проекта с массивом всех его автоматов и именем. После этого момента исходные данные из документа Visio больше не понадобятся в процессе трансляции.

В данной реализации процесс импорта данных из Visio, занимающий наибольшее время, является однопроходным, что ускоряет время преобразования, в отличие, например, от MetaAuto, где он является двухпроходным.

Генерация файлов с исходным кодом

После преобразования исходных данных в модель запускается генерация выходных данных. Они записываются в файл формата XML, содержащего в себе автоматную модель, и набор файлов со сгенерированным программным кодом на языке C (см. рис. 6).

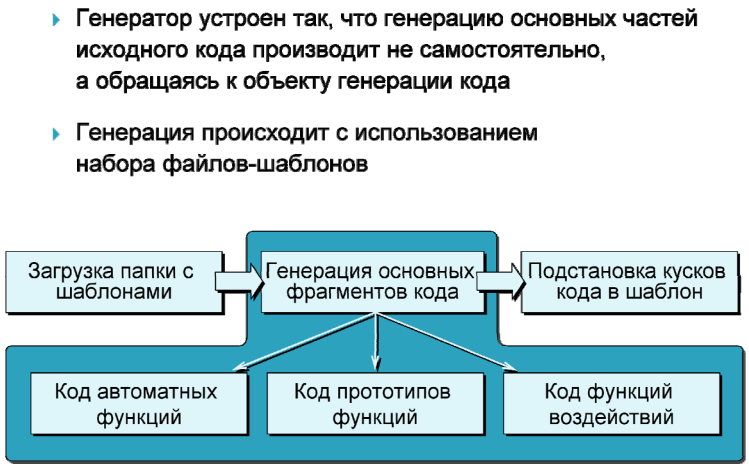

Рис. 6. Генерация файлов с кодом.

Сначала происходит генерация файла XML. В этом файле описываются все автоматы данного документа. В каждом автомате описываются их состояния и список состояний, с которыми это состояние соединено, с описанием условия перехода и списка выходных воздействий. Такие файлы могут использоваться в качестве исходных данных для работы сторонних инструментальных средств.

После сохранения файла формата XML запускается генерация файлов с исходным программным кодом. Для удобства пользователя эти файлы лежат в папке, которая называется так же, как и исходный документ Visio. Генерация файлов с исходным программным кодом происходит на основе ранее созданной модели состояний и переходов.

Важно отметить, что кодогенерация происходит на основе специфичной для каждого языка шаблонной директории. В этой директории содержится произвольный набор шаблонных файлов с произвольным содержанием, в которых расставлены специальные метки (например, $automata_functions). В процессе трансляции эти метки будут заменены на соответствующие участки сгенерированного кода. Такой подход позволяет любому пользователю настроить стиль генерируемых файлов, так как ему больше нравится, не прикладывая особых усилий.

Кодогенератор устроен так, что генерацию основных частей исходного кода производит не самостоятельно, а обращаясь объекту генерации кода, который тоже передаётся генератору в качестве входных данных. Этот объект содержит функции для генерации частей кода для конкретного языка программирования. В качестве такого объекта может выступать любой объект, поддерживающий интерфейс генерации кода. Такой подход позволяет генерировать исходный программный код на нескольких языках программирования, имея объект генерации кода для каждого из этих языков, что предполагается в будущем.

Интерфейс программы

Программа транслятора может быть запущена в двух режимах, выбираемых ключом командной строки. В одном из режимов трансляция производится неинтерактивно, и вся информация считывается из командной строки (рис. 7). Этот режим удобен при использовании

в средствах автоматической сборки (make, менеджер проектов MS Visual Studio). Другой режим задается ключом командной строки –gui. В нем интерфейсом программы служит диалоговое окно, где пользователь может задавать параметры, пользуясь элементами графического интерфейса .NET (рис. 8). Если же программа запущенна без параметров командной строки, то она открывается в режиме графического интерфейса.

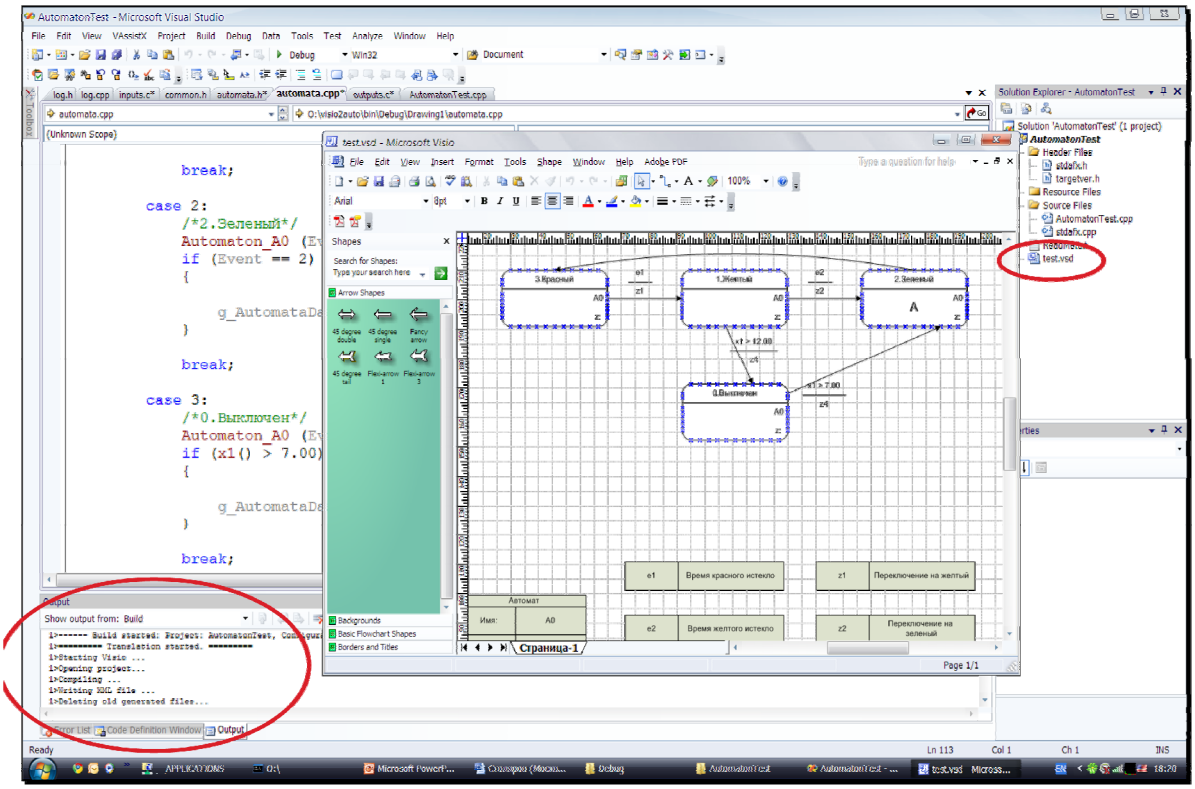

Рис. 7. Пример работы транслятора в пакетном режиме интеграции с MS Visual Studio.

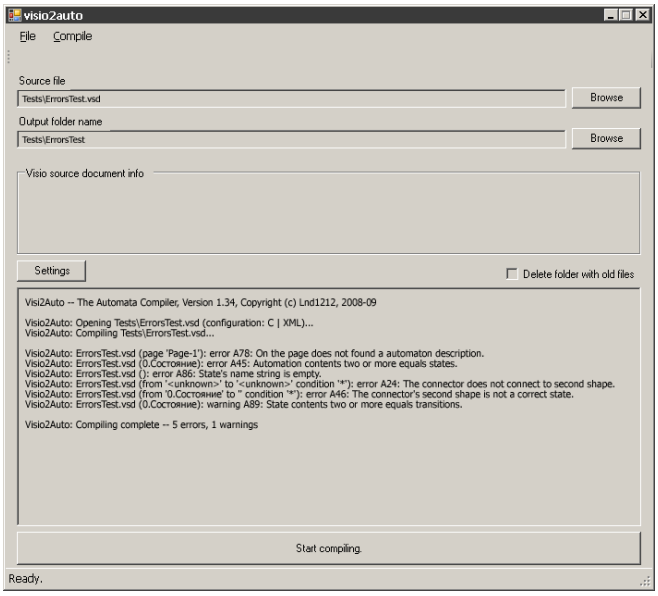

Рис. 8. Пример работы транслятора в интерактивном режиме.

Архитектура проекта

Основным элементом архитектуры проекта является класс. На схеме ниже (рис. 9) показана иерархия классов в проекте. Рассмотрим назначение основных элементов архитектуры.

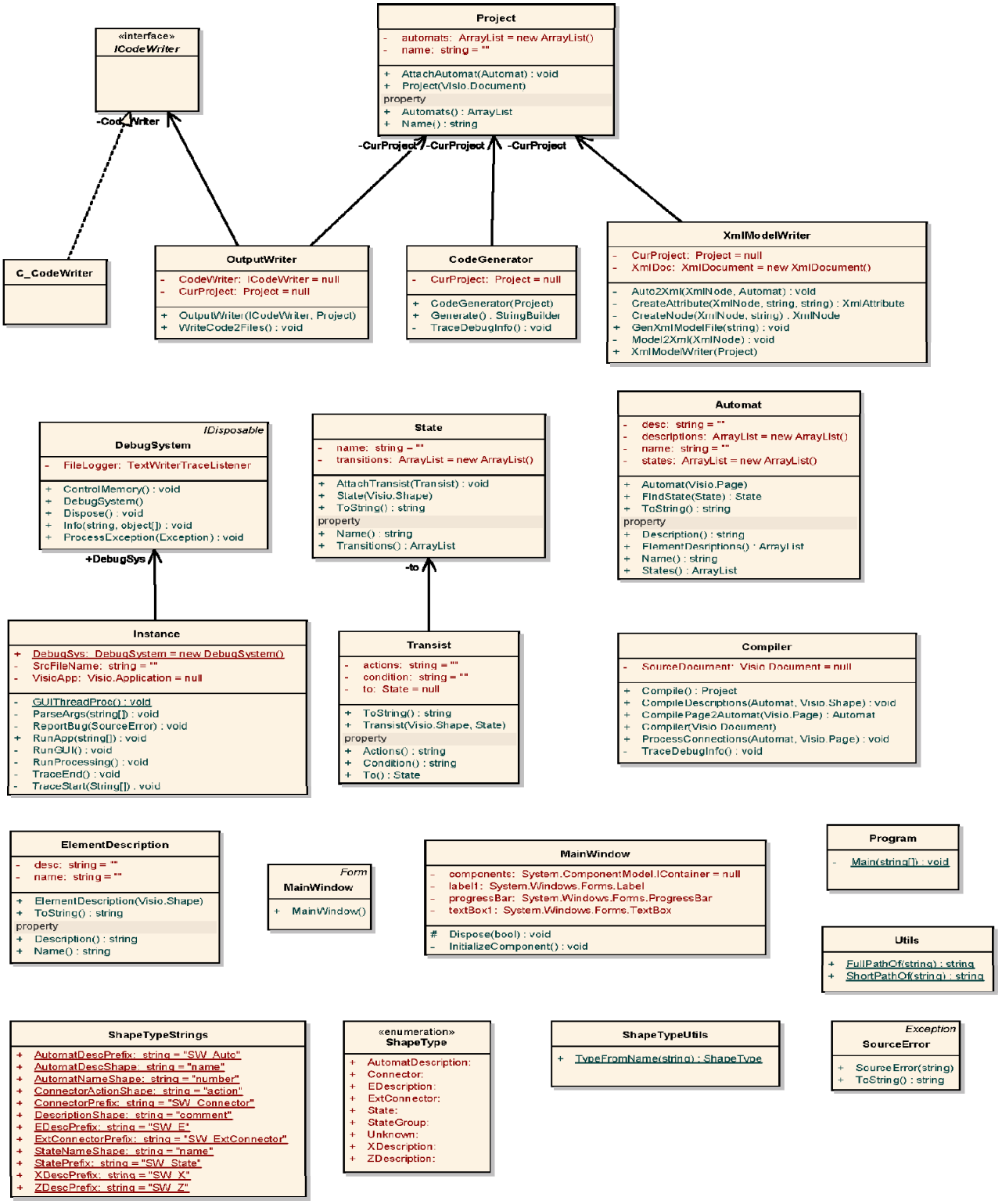

Рис. 9. Архитектура проекта.

Класс Project

Этот класс предназначен для представления исходного документа Visio в модели переходов и состояний. В нём содержатся все автоматы документа.

Класс Sate

Класс для представления состояния. Содержит все данные о конкретном состоянии.

Класс Automaton

Класс для представления автомата. Содержит данные об автомате и набор его состояний.

Класс ElementDescription

Класс для описания входного или выходного воздействия.

Класс Transition

Класс для представления перехода между состояниями. Содержит исходное и конечное состояние, а также условия перехода и список выходных воздействий.

Класс Compiler

Класс занимается преобразованием исходного документа Visio в модель, основанную на переходах и состояниях. После преобразования возвращает объект класса Project

Класс CodeGenerator

Класс является вспомогательным для процесса генерации исходного кода. Он генерирует конкретные части исходного кода на конкретном языке, которые впоследствии будут подставлены в сгенерированный код, согласно шаблонным файлам.

Класс OutputCodeWriter

Класс предназначен для генерации файлов с исходным программным кодом на основе директории с шаблонными файлами.

Класс C_CodeWriter

Класс содержит функции генерации частей кода на языке программирования C. Объект этого класса передаётся в CodeGenerator.

Класс XmlModelWriter

Класс используется для генерации файла формата XML на основе объекта класса Project.

Интерфейс ICodeWriter

Этот интерфейс обязательно должны реализовывать те классы, объекты которых будут использоваться при генерации исходного кода на конкретном языке программирования (например, C_CodeWriter).

Перечисление ShapeType

В этом перечислении содержатся типы элементов графа переходов автомата (состояние, переход, описание и т.д.).

Класс ShapeTypeUtils

Утилитарный класс используется для получения типа фигуры по её имени.

Класс MainWindow

Класс описывает главное окно приложения (оно запускается в режиме GUI).

Класс ShapeTypeStrings

Класс хранит в себе строки, используемые в программе для определения типа фигур. Вынесение этих строк в отдельный класс позволит с большей независимостью совершенствовать файл шаблонов Visio.

Система отладки

Система отладки имеет большое значение для данного проекта, так как во время использования транслятора неизбежно будут проявляться различные ошибки и недоработки его автора.

Хорошо реализованная система отладки позволит пользователю сообщить об ошибке разработчику, а разработчику – быстрее устранить недоработки, используя отладочную информацию, сгенерированную системой отладки в случае ошибки.

В этом случае системой отладки генерируется лог-файл с подробной отладочной информацией об ошибке, который потом может быть передан автору для анализа. Так же в отладочном режиме программа выводит отладочную информацию в output debug string.

Система отладки построена на использовании исключений языка C#. Она обрабатывает как стандартные исключения, так и исключения пользовательских типов.

Система обработки ошибок в автоматной схеме

При разработке проекта особое внимание было уделено системе обработки ошибок в графе переходов автомата. Эта система оперирует двумя понятиями: ошибка и предупреждение. Ошибка – это некий дефект схемы, без исправления которого нельзя продолжить трансляцию. Предупреждение – это некий дефект, без устранения которого можно продолжить трансляцию, но он скорее всего является следствием высокоуровневой логической ошибки в исходной схеме. После трансляции все ошибки и предупреждения выводятся в консоль транслятора. В режиме интеграции со средствами сборки приложений при возникновении ошибок транслятор возвращает 1, что позволяет вызвавшему средству сборки приложений понять, что произошла ошибка трансляции и остановить процесс сборки. Формат вывода ошибки или предупреждения идентичен формату компилятора Visual Studio. В нём указывается имя исходного документа, имя автомата, ошибочный элемент, номер (уникальный идентификатор) ошибки и строка с сообщением об ошибке.

Рассмотрим основные типы определяемых ошибок (рис. 10).

1. Переход соединяет не состояния, а другие фигуры Visio.
2. Переход не присоединён к фигуре.
3. В автомате найдены одинаковые элементы.
4. У состояния нет имени.
5. У перехода нет условия.

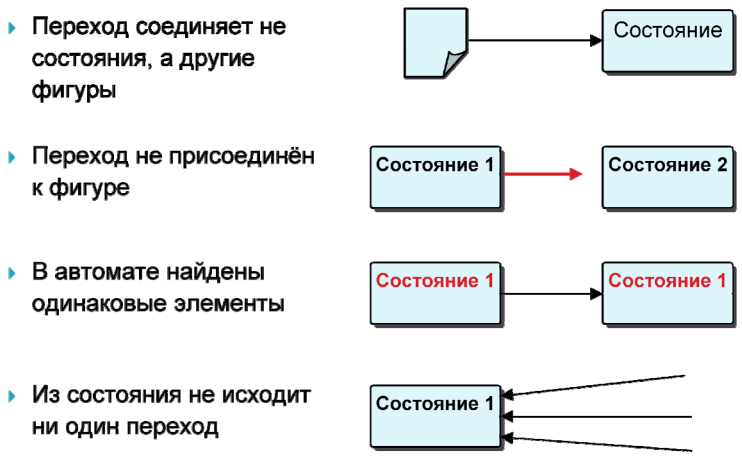

Рис. 10. Типы определяемых ошибок.

Результаты работы и планы развития

В результате работы было создано инструментальное средство для трансляции графов переходов автоматов, представленных в формате Microsoft Visio в исходный программный код на языке C.

Разработаны сохранение исходных данных в файл XML, средства диагностики ошибок в исходных данных, средства интеграции в автоматические процессы сборки приложений (make-файлы, MS Visual Studio build system).

Планируется совершенствовать гибкость инструментального средства, в частности, добавить трансляцию в различные языки программирования. Это планируется реализовать с помощью подключаемых модулей (plug in), что позволит любому желающему, знающему программирование на небольшом уровне, на любом известном ему языке программирования написать модуль и таким образом реализовать трансляцию в нужный ему язык не имея исходного кода транслятора.

Список литературы

1. Шалыто А.А. Switch технология. Алгоритмизация и программирование задач логического управления. СПб, Наука, 1998.
2. Шалыто А.А., Наумов Л.А. Реализация автоматов в объектно – ориентированных программах. Искусственный интеллект, №4, 2004.
3. Шалыто А.А., Туккель Н.И. Реализация автоматов при программировании событийных систем. Программист, №2, 2004.
4. Канжелев С. Ю., Шалыто А. А. Преобразование графов переходов, представленных в формате MS Visio в исходные коды программ для различных языков программирования (инструментальное средство MetaAuto). Проектная документация. http://is.ifmo.ru/projects/metaauto.