УДК 004.5

М.А. Братуха, Н.В. Семенистый, А.Н. Мирошкин
Донецкий национальный технический университет, г. Донецк
кафедра компьютерной инженерии

ПОСТРОЕНИЕ ГРАФ-СХЕМ КОНЕЧНЫХ АВТОМАТОВ НА ОСНОВЕ ФОРМАТА KISS2 ПРИ ПОМОЩИ JAVASCRIPT

Аннотация

Братуха М.А., Семенистый Н.В., Мирошкин А.Н. Построение граф-схем конечных автоматов на основе формата kiss2 при помощи Javascript*. Выполнен анализ разработки приложения на языке Javascript с применением нескольких паттернов проектирования. Создано приложение для построения граф-схемы конечного автомата на основе формата KISS2.*

Ключевые слова**:** *конечный автомат, KISS2, Javascript, KnockoutJs, jCanvas, Mediator.js.*

Постановка проблемы: формат KISS2 применяется для описания конечных автоматов, которые являются эталонными и применяются для оценки спроектированного устройства. Для удобства анализа автоматов, описанных в данном формате, необходима граф-схема.

Цель доклада – проанализировать современные подходы проектирования приложения на языке Javascript с применением паттернов отслеживания состояния модели и слабого связывания.

Постановка задачи. Необходимо проанализировать формат KISS2, извлечь полезную информацию и на ее базе построить граф-схему с применением технологии HTML5 Canvas и Javascript.

Актуальность темы. При синтезе схем устройств управления часто используется модель микропрограммного автомата (МПА) Мура. Одним из современных базисов, используемых для реализации схем МПА, являются программируемые логические интегральные схемы (ПЛИС).[4] А так как управляющий управления базируются на конечных автоматах, то актуальность рассматриваемой темы высока.

Анализ формата KISS2. Конечный автомат (КА) является наиболее общей моделью синхронных последовательностных логических схем и позволяет проводить операции их синтеза, анализа, композиции, декомпозиции, минимизации, оптимизации на абстрактном и структурном уровнях. [1]

Одним из форматов описания конечного автомата является формат KISS2. Он является текстовым и состоит из 2 частей: заголовков и списка переходов. Структура файла формата KISS2 отображена на рисунке 1.

В заголовке декларируются количественные характеристики конечного автомата: .i - количество входов, .o - количество выходов, .p - количество переходов, .s - количество состояний, .r - состояние сброса.

Следующие строки характеризуют переходы из одного состояния в другой, одна строка соответствует одному переходу. В ней описываются входные и выходные значения, а также исходное и результирующее состояние.

Последовательности входных и выходных значений могут быть представлены символом из алфавита {0,1,-}, где “1” выражает прямое значение, “0” – обратное, а “–” – отсутствие влияния значения в данном состоянии. Входное и результирующее состояние представляет собой текстовое описание и может содержать любое количество знаков.

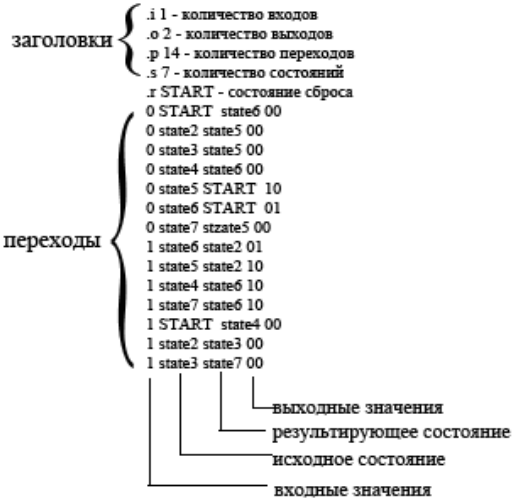

Рисунок 1 – Структура файла формата KISS2

Анализ использованных технологий. Web-технологии с каждым годом завоевывают все большую долю рынка программного обеспечения. В особенности это касается проектов, требующих отображения какой-то информации пользователю, так как язык разметки HTML становится де-факто языком пользовательских интерфейсов после выхода стандарта HTML5. Основным же языком разработки является Javascript стандарта ECMAScript 5.

Первым шагом построения граф-схемы является анализ формата KISS2 с целью выявления полезной для отображения информации.

Для связи модели данных с отображением был использован подход MVVM (Model – View – ViewModel). Он подразумевает двустороннюю связь, то есть данные, полученные с отображения, автоматически отображаются в модели и наоборот. Структура паттерна MVVM отображена на рисунке 2.

MVVM базируется на MVC (Model-View-Controller) и MVP (Model-View-Presenter), пытаясь более прозрачно разделить разработку UI от бизнес логики

и поведения приложения. Многие реализации данного шаблона используют декларативную привязку данных.

Учитывая то, что данное приложение не имеет серверной части, оно реализует только часть данного паттерна: View – ViewModel. В случае же необходимости сохранять данные на сервер или же наоборот импортировать их с сервера на клиентскую сторону, это может быть реализовано безболезненно, расширив структуру элементом Model.

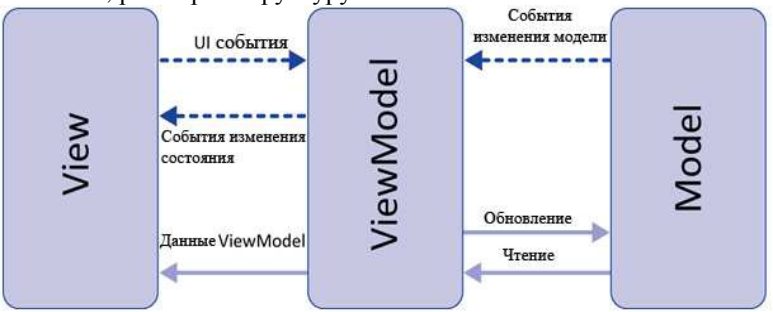

Рисунок 2 – Структура паттерна MVVM

Конкретной реализацией данного подхода является библиотека KnockoutJs. Она предоставляет несколько функций: observable и observableArray для отслеживания состояния переменной или элементов массива соответственно.[2] Эти функции реализуют паттерн Observer, таким образом, при изменении свойств объекта, он публикует эти изменения, а другие объекты могут подписаться на эти изменения.

Функция applyBindings выполняет связывание объекта и DOM элемента, то есть этот элемент становится подписчиком обновлений данного объекта и наоборот.

Вернемся к конкретной реализации построения граф-схемы. Итак, после анализа входного текста, определяются параметры схемы из заголовка KISS2. Ими являются: количество входных и выходных сигналов, количество состояний и количество переходов. Затем выполняется цикл прохода по строкам, отображающим состояния, и формируются объекты, каждый из которых характеризует один переход. Эту обработку выполняет экземпляр класса KissFormat. Структура классов приложения отображена на рисунке 3.

Графическим же отображением занимается экземпляр класса KissGraphics. Для связи между этими классами используется паттерн Mediator и его конкретная реализация для JavaScript Mediator.js. Оба объекта имеют зависимость от медиатора, который дает возможность разорвать связь между этими объектами. Он реализует подход publish/subscribe, один из основных при разработке JavaScript-based приложений.

Медиатор – это поведенческий шаблон проектирования, который позволяет создать унифицированный интерфейс для разных частей системы, которые могут общаться между собой. Он становится центральной точкой системы вместо прямых отношений между модулями (коллегами).[3]

В библиотеке медиатор есть несколько возможностей управления подписками. Так, можно задать количество принятых сообщений, после которого подписка будет удалена, в случае одного сообщения подписка осуществляется через специальный метод once.

Другой возможностью является задание предиката, то есть функции, которая позволяет обработать сообщение только в том случае, если ее возвращаемым значением является true.

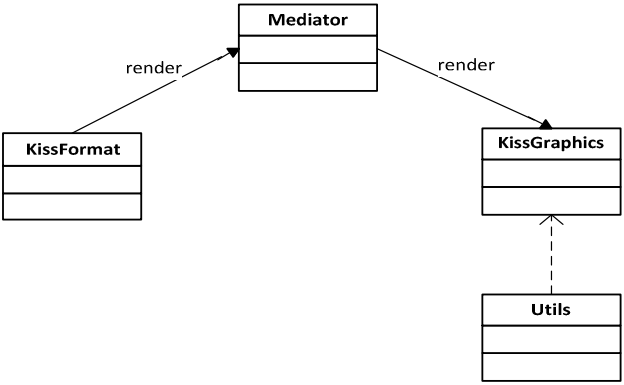

Рисунок 3 – Структура классов приложения

После изменения свойств объекта KissFormat, отображающих состояние входных параметров для рендеринга, этот объект публикует сообщение, на которое подписан объект KissGraphics. Он же, в свою очередь, обрабатывает информацию, полученную через медиатор, и далее производит вызов собственных методов, которые формируют граф-схему на экране.

Процесс визуализации (рендеринга) входных данных, разбит на три этапа:

1) Расчёт размеров поля canvas и координат вершин графа на этом поле
2) Прорисовка вершин с их названием
3) Прорисовка переходов (линий связи) между вершинами графа с соответствующими условиями перехода

Размер поля canvas зависит от ширины окна браузера и от количества состояний автомата, т.е. чем больше состояний у автомата, тем больше высота поля canvas. Это сделано для того, чтобы вершины не вышли за границы поля рисования.

Алгоритм расчёта позиций вершин графа таков: если текущая анализируемая вершина может перейти в более чем одну вершину, то те

вершины, в которые она может перейти, расставляются в ряд (по горизонтали) под текущей вершиной. После установления координат вершинам, производится анализ других вершин, которые не участвовали в процессе рассмотрения текущей вершины.

Расчёт радиуса каждой вершины производится автоматически, в зависимости от доступной ширины экрана браузера: чем больше ширина, тем больший будет радиус. Размер шрифта внутри вершины также автоматически подстраивается под размеры вершины.

Весь процесс рендеринга осуществляется с помощью библиотеки jCanvas, которая работает с HTML5 элементов canvas. В ней представлено множество методов для рисования графических примитивов, вывода текста, наложения градиента, всяческие трансформации над выводимыми объектами и прочее. Результат работы приложения изображен на рисунке 4.

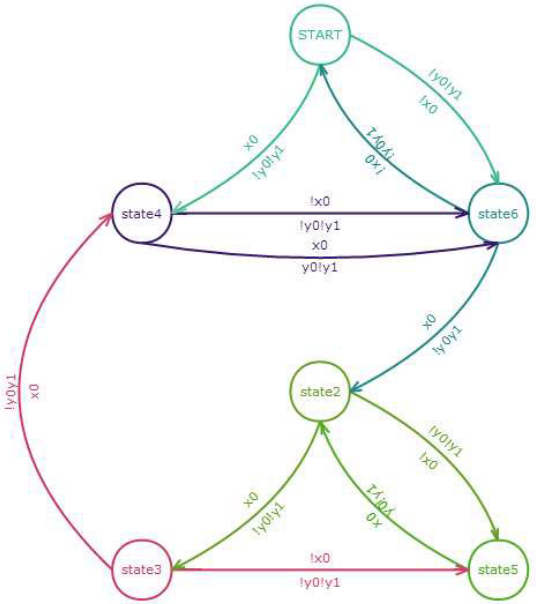

Рисунок 4 – Результат работы приложения

Процесс прорисовки линий переходов автомата является самым сложным этапом, поскольку необходимо сделать так, чтобы линии не налагались друга на друга и не пересекались с вершинами. Чтобы максимально избежать таких случаев, каждая линия связи учитывает позиции вершин относительно друг друга. Например, если текущая вершина находится выше и левее следующей, то линию связи нужно отклонить вправо и вверх, что хорошо видно из графа

на рисунке 4. В нём вершина START имеет две связи с вершинами state3 и state 6. Линия связи от START до state6 отклонена вправо и вверх.

Также у линий связи присутствует зависимость от расстояния между вершинами – чем дальше вершины находятся друг от друга, тем сильнее нужно отклонять линию связи в какую-то из сторон. Данная методика позволяет избежать наложения линий друг на друга.

В том случае, если линия связи уже была проведена к какой-то вершине, и текущая рассматриваемая линия связи связана с той же вершиной, но только по другому условию, то линия связи будет располагаться с противоположной стороны вершины.

Каждая линия связи рисуется по трём точкам: начальная, контрольная (точка изгиба) и конечная. Условие перехода по каждой из линий связи отображается в центре линии изгиба и автоматически поворачивается так, чтобы текст оказался «лежащим» на линии.

Для удобства восприятия каждая вершина раскрашивается в свой цвет (случайно сгенерированный). Соответственно, каждая линия связи, выходящая из текущей вершины, так же окрашивается в такой же цвет.

Выводы. Проведен анализ формата описания конечных автоматов KISS2 и на его основе разработано Web-приложение для построения граф-схем, которое позволяет наглядно увидеть связность состояний конечного автомата.

Дальнейшие разработки приложения будут направлены на улучшение отображения граф-схемы, также будет добавлено отображение в виде блок-схемы для разных типов автоматов (Мили, Мура), что позволит избежать недостатков применяемого метода, и позволит отобразить работу автомата ещё более наглядным образом. Так же будет пересмотрен текущий алгоритм формирования позиций вершин относительно друг друга.

Список литературы

1. Татолов Е.Р., Солдатов К.А., Зеленёва И.Я. Методика получения КА Мура из KISS2-спецификаций и ее применение к исследованиям в базисе FPGA // «Информатика и компьютерные технологии-2011» - 2011 г., Донецк, ДонНТУ. С. 64-68.

2. Knockout : Observables. Электронный ресурс. – Режим доступа: http://knockoutjs.com/documentation/observables.html

3. Osmani A. Learning Javascript Design Patterns – O'Reilly, 2012. - с. 49.

4. Оптимизация схемы автомата Мура на однородных ПЛИС / А.А. Баркалов, А.В. Матвиенко, С.А. Цололо // Комп'ютерні засоби, мережі та системи. — 2009. — № 8. — С. 45-51.