**УДК 004.436.4+004.415.2+004.434:004.94**

# КОНФИГУРИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ НА ОСНОВЕ ДИАГРАММ СОСТОЯНИЙ UML

**И. А. Габидуллин[1], А. А. Марченко [2]**

[1,2] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]ibragim0795@yandex.ru, [2]marchenko@it.kfu.ru

***Аннотация***

Описан способ использования UML-диаграмм для конфигурирования поведения веб-приложений: при помощи конфигурации определяются поведение системы, переходы между экранами, а также логика обработки данных. Изучены и сравнены разного рода UML-диаграммы на предмет их возможностей. Разработан веб-фреймворк для ASP.NET Core, который использует UML-диаграмму для формирования конфигурации в формате файлов XML или JSON, на основе которых выстраивается поведение веб-приложения. Рассмотрены дальнейшие шаги в использовании и развитии полученного веб-фреймворка.

***Ключевые слова**: UML-диаграммы, веб-сайт, конфигурация, веб-фреймворк*

## ВВЕДЕНИЕ

Современное программное обеспечение в виде веб-сайтов или приложений для операционных систем становится всё сложнее. Большинство веб-приложений наполнено разнообразным функционалом, и его становится только больше как в корпоративных системах, так и в широко распространённых веб-сайтах для пользователей интернета. Чем шире функционал, тем в большее количество состояний может перейти программное обеспечение.

С другой стороны, на начальной стадии разработки программного обеспечения удобно использовать такой инструмент, как UML (Unified Modeling Language) [1]. Этим языком моделирования могут быть описаны разные характеристики (начиная от модели данных, заканчивая поведением системы) разрабатываемого программного обеспечения в виде диаграмм. На основании некото-

рых UML-диаграмм, например, структурных (то есть модели данных), можно при помощи специальных UML-редакторов создавать шаблоны исходного кода, которые в дальнейшем можно преобразовать в рабочий программный код. Эти возможности поддерживаются некоторыми редакторами UML [2, 3]. Сейчас ведутся работы в направлении того, чтобы на основе поведенческих UML-диаграмм генерировать уже готовый исходный код программного продукта [4, 5], но очевидно, что при такой генерации получается неоптимальный, возможно нечитабельный и неочевидный программный код. Работы в данном направлении до сих пор продолжаются.

В статье предложен другой подход: использовать поведенческую UML-диаграмму для конфигурирования приложения. Для этого разработан веб-фреймворк, который использует UML-диаграмму, описывающую поведение веб-приложения для конфигурирования разрабатываемого веб-приложения. Таким образом, диаграмма состояний UML будет преобразована в конфигурацию системы. Для разработки были выбраны язык программирования C# и ASP.NET Core, потому что, во-первых, C# занимает в индексе Tiobe высокую позицию [6], а, во-вторых, ASP.NET Core используется как в корпоративной, так и индивидуальной разработках.

## 1. ДИАГРАММА СОСТОЯНИЙ UML

Пример такой диаграммы можно увидеть на Рис. 1. Оригинальное название – UML State Machine или UML Statechart.

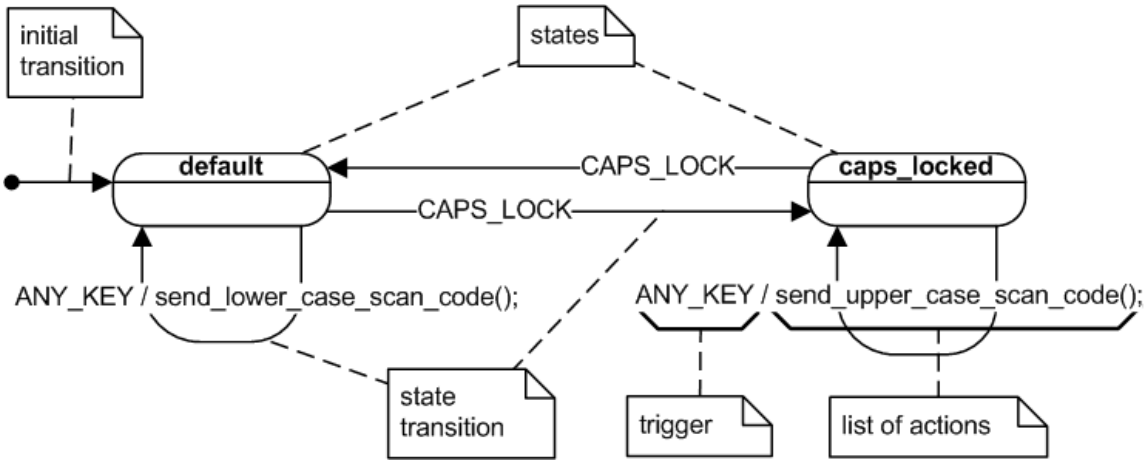


Рис. 1. Пример работы клавиатуры, описанный при помощи UML State Chart

Диаграмма состояний UML является развитием идей диаграммы состояний из теории автоматов. Диаграмма состояний – это направленный граф, вершины которого представляют состояние системы, а рёбра – это переходы между состояниями. Ключевой особенностью является то, что здесь применяется событийно-ориентированный подход. Изменение состояния системы происходит под действием каких-либо событий. Если рассматривать систему как веб-сайт, то это, например, клик мыши на какую-нибудь кнопку. Помимо этого, достоинством диаграммы состояний является то, что каждое состояние определяет специфическое поведение системы. Состоянием описывается бизнес-логика приложения. Таким образом, с помощью данной диаграммы можно спроектировать сложную систему, в которой будут строго организованные переходы и строго организованная обработка данных. Использование такой структуры позволяет добиться чёткости и явного поведения системы, когда на каждое событие происходит какое-либо действие.

UML-редакторы для хранения диаграмм используют специальный язык разметки XML (eXtensible Markup Language) [7]. Для UML существует специальный подкласс XML – XMI (XML Metadata Interchange) [8]. Данный язык является очень сложным и покрывает все возможные UML-диаграммы. Большинство UML редакторов поддерживает XMI [2, 3]. Но XMI не подходит под наши задачи именно из-за его сложности; XMI, который генерируется редакторами UML, хранит в себе помимо того, что важно для разработчика, ещё и то, что необходимо редакторам для отображения UML. В итоге получается очень большой файл, который содержит много ненужной информации.

С другой стороны, существует SCXML. Данный язык создан для поддержки как раз UML State Chart. Он может быть сгенерирован некоторыми UML-редакторами [2]. Язык отличается хорошей читаемостью. В то время как XMI разработан для программ, SCXML разработан скорее для передачи информации между людьми. Его может читать и понимать практически не подготовленный специалист. SCXML-файлы генерируются на основе UML State Chart.

Существует конвертер, который генерирует SCXML-файл на основе XMI [9]. Так же, как и XMI, SCXML-конвертер создан на основе XSLT (XSL Transformations) – язык преобразований XML-документов. Значит, можно путем преобразований

получить из XMI-файла SCXML-файл. В итоге для дальнейшей работы нами был выбран SCXML-язык. В разработке языка конфигурации системы мы отталкивались именно от SCXML.

## 2. ЯЗЫК КОНФИГУРАЦИИ СИСТЕМЫ

SCXML, несмотря на все достоинства, имеет громоздкую структуру с большим количество вложенности. Поэтому был разработан язык конфигурации. Также причиной новой разработки является оптимизация: исключены все лишние элементы, что освобождает память при работе с файлом конфигурации.

Были разработаны два вида языка конфигураций: на основе JSON и на основе XML. Соответственно разработчик может описать систему в том виде, в котором ему удобнее. Достоинством JSON является то, что он компактнее, чем XML. XML же, с одной стороны, более понятен для чтения, а с другой, является довольно громоздким языком разметки. В мире существуют разработчики, которые предпочитают, как XML-нотацию, так и JSON-нотацию, поэтому было принято решение поддерживать оба формата. Рассмотрим пример языка на основе JSON.

```
[
  {
    "Id": "1",
    "PreviousState": "1",
    "CurrentState": "FirstStep",
    "NextState": "2",
    "Conditions": {
      "cond1": 2,
      "cond2": 3
    }
  },
  {
    "Id": "2",
    "PreviousState": "1",
    "CurrentState": "SecondStep",
    "NextState": "3"
  },
```

Рис. 2. Пример языка конфигурации на основе JSON

На Рис. 2 мы видим часть поведения системы. Рассмотрим её подробнее.

Мы видим, что состояние «1» имеет предыдущее состояние «0», то есть самое начальное состояние. В свою очередь из состояния «1» по умолчанию происходит переход в состояние «2». Но также есть условные переходы. При определенном условии система перейдет в состояние «2», а при определенных – в состояние «3». Таким образом, описание системы покрывает как последовательные переходы, так и условные переходы, что важно в условиях комплексного программного обеспечения.

Как будет выглядеть данная конфигурация, написанная с использованием XML, можно увидеть на Рис. 3.

```
<?xml version="1.0"?>
<ArrayOfState>
  <State>
    <Id>1</Id>
    <PreviousState>1</PreviousState>
    <CurrentState>FirstStep</CurrentState>
    <NextState>2</NextState>
  </State>
  <State>
    <Id>2</Id>
    <PreviousState>1</PreviousState>
    <CurrentState>SecondStep</CurrentState>
    <NextState>3</NextState>
  </State>
```

Рис. 3. Пример языка конфигурации на основе XML

На данный момент язык конфигурации поддерживает основные возможности для разработки программного обеспечения.

## 3. ПРОГРАММНОЕ РЕШЕНИЕ ДЛЯ ВЗАИМОДЕЙСТВИЯ С КОНФИГУРАЦИЕЙ

Рассмотрим программное решение. Веб-фреймворк взаимодействует с базовым подходом .NET Core – MVC, то есть реализована схема разделения принципа работы на основании – «Модель – Представление – Контроллер». Разработанное программное обеспечение используется на уровне «Контроллер».

Программное решение представляет собой несколько классов, таких, как:

1. FlowService – является самой важной частью фреймворка, именно этот класс решает, в какое следующее состояние переходит система;
2. JsonWorker или XmlWorker – реализуют интерфейс IFlowWorker; эти классы обеспечивают работу с файлами конфигурации, написанными соответственно на языках JSON и XML;
3. FlowData – это модель данных; после считывания классы из второго пункта хранят данные в этом файле.

Класс FlowService регистрируется в конфигурации ASP.NET Core в виде сервиса. После этого в Контроллере веб-приложения вызывается метод данного класса, для перехода в следующее состояние. В свою очередь Контроллер по возвращаемому значению из метода определяет, какие действия будут выполняться в дальнейшем.

FlowService взаимодействуют как с файлом конфигурации, так и с базой данных веб-приложения. В ней хранятся история пользователя (на каком именно состоянии остановилось приложение), а также история переходов между состояниями (для возврата, при необходимости, в предыдущие состояния).

Одним из достоинств такой системы является то, что можно добавлять новый функционал прямо из системы, таким образом, мы можем создать приложение, которое может конфигурировать само себя. Данный функционал важен для корпоративных систем, в которых существует необходимость настроить приложение для каждой из ролей.

На текущий момент похожих разработок для ASP.NET Core обнаружено не было. Такого рода разработки есть для языка программирования Java Script [10, 11]. Их недостатком является то, что программа будет выполняться на стороне пользователя, значит, существует возможность для ее модификации пользователем, а это может повлиять на безопасность веб-приложения. Также существует похожий по идеям фреймворк Spring Web Flow для языка программирования Java [12], в нем также используется конфигурация в виде XML-файла. Недостатком является то, что этот фреймворк разработан в 2008 году и не поддерживает современные возможности веб-приложений. В свою очередь, так как мы используем современный веб-фреймворк ASP.NET Core, это обеспечивает нам поддержку современных возможностей веб-приложений.

## ЗАКЛЮЧЕНИЕ

На данный момент времени решение, которое существует, позволяет создавать веб-приложения, которые включают в себя большое количество переходов между состояниями системы. Данный веб-фреймворк подходит для таких приложений, в которых есть чёткая структура переходов между состояниями системы, например, какой-нибудь тест, экзамен или же такие действия, как покупка билетов или бронирование отеля, где на каждом отдельном шаге необходимо вводить какие-либо данные.

В дальнейшем планируется развитие как языка конфигурации, так и самого веб-фреймворка. Будет продолжено изучение разметки SCXML на предмет её возможностей, а также на предмет того, какой еще функционал можно будет привнести. Планируется развитие веб-фреймворка в сторону создания веб-приложений, которые могут конфигурировать сами себя. Помимо этого, планируется разработать веб-приложение для апробации фреймворка, представленного выше.

## СПИСОК ЛИТЕРАТУРЫ

1. Unified Modeling Language. URL: http://www.uml.org/
2. Enterprise Architect. URL: http://www.sparxsystems.com/products/ea/
3. Visual Paradigm. URL: http://www.visual-paradigm.com/
4. *Yusufu M., Zhang H.J., Yusufu G., Liu Z.D., Cheng P., Dilisati D.* Modeling and Analysis of Complex System with UML: A Case Study // Applied Mechanics and Materials, January 2014. V. 513–517. P. 1346–1351.
5. *Pham V.C., Radermacher A., Gerard S., Li S.* Complete code generation from UML state machine // 5th Int. Conf. on Model-Driven Engineering and Software Development, Porto, Portugal, 19–21 Feb. 2017. P. 208–219.
6. TIOBE Index. URL: http://tiobe.com/tiobe-index/
7. Extensible Markup Language (XML). URL: https://www.w3.org/XML/
8. XML Metadata Interchange (XMI). URL: https://www.omg.org/spec/XMI/
9. XMI to SCXML Converter. URL: http://github.com/apache/commons-scxml/blob/master/extras/xmi2scxml.xsl/
10. Next-gen state management based on Harel Statechart and SCXML. URL: http://github.com/aksonov/statem/

11. State Machine Cat. URL: http://github.com/sverweij/state-machine-cat/

12. *Vervaet E.* The Definitive Guide to Spring Web Flow. Berkeley: Apress, 2008. 380 p.

## CONFIGURING WEB APPLICATIONS BASED ON UML STATE MACHINE DIAGRAM

**I. A. Gabidullin[1], A. A. Marchenko[2]**

[1,2] *Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University*

[1]ibragim0795@yandex.ru, [2]marchenko@it.kfu.ru

***Abstract***

In this paper, we describe a way to use UML diagrams for configuring web-application's behavior. That is, using the configuration will determine the behavior of the system, the transitions between the screens, as well as the business logic of application. We examine and compare various UML diagrams for their possible features. A web framework for ASP.NET Core is developed, which uses the UML diagram to form a configuration in the XML or JSON file formats. Configuration will determine the behavior of the system. In addition, we describe further steps in using and developing the web-framework.

***Keywords****: UML diagram, web site, configuration, web-framework*

**REFERENCES**

1. Unified Modeling Language. URL: http://www.uml.org/
2. Enterprise Architect. URL: http://www.sparxsystems.com/products/ea/
3. Visual Paradigm. URL: http://www.visual-paradigm.com/
4. *Yusufu M., Zhang H.J., Yusufu G., Liu Z.D., Cheng P., Dilisati D.* Modeling and Analysis of Complex System with UML: A Case Study // Applied Mechanics and Materials, January 2014. V. 513–517. P. 1346–1351.

5. *Pham V.C., Radermacher A., Gerard S., Li S.* Complete code generation from UML state machine // 5th Int. Conf. on Model-Driven Engineering and Software Development, Porto, Portugal, 19–21 Feb. 2017. P. 208–219.

6. TIOBE Index. URL: http://tiobe.com/tiobe-index/

7. Extensible Markup Language (XML). URL: https://www.w3.org/XML/

8. XML Metadata Interchange (XMI). URL: https://www.omg.org/spec/XMI/

9. XMI to SCXML Converter. URL: http://github.com/apache/commons-scxml/blob/master/extras/xmi2scxml.xsl/

10. Next-gen state management based on Harel Statechart and SCXML. URL: http://github.com/aksonov/statem/

11. State Machine Cat. URL: http://github.com/sverweij/state-machine-cat/

12. *Vervaet E.* The Definitive Guide to Spring Web Flow. Berkeley: Apress, 2008. 380 p.

**СВЕДЕНИЯ ОБ АВТОРАХ**

***ГАБИДУЛЛИН Ибрагим Анварович*** – магистрант Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

***Ibragim Anvarovich GABIDULLIN*** – has master's degree of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University.

email: ibragim0795@yandex.ru

***МАРЧЕНКО Антон Александрович*** – ассистент кафедры Программной инженерии Высшей школы информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета.

***Anton Aleksandrovich MARCHENKO*** – assistant of the Department of Program Engineering of the Higher School of Information Technologies and Intelligent Systems at Kazan (Volga region) Federal University.

email: marchenko@it.kfu.ru

*Материал поступил в редакцию 28 мая 2018 года*