Санкт-Петербургский государственный университет

Программная инженерия

Кузьмина Елизавета Владимировна

Веб-платформа предметно-ориентированного моделирования на базе REAL.NET

Выпускная квалификационная работа магистра

Научный руководитель:
доцент кафедры системного программирования, к.т.н.
Ю.В. Литвинов

Рецензент:
инженер-программист АО «ПФ «СКБ Контур»
А.О. Перешеина

Санкт-Петербург
2020

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering

Elizaveta Kuzmina

Web domain-specific modeling platform based on REAL.NET

Graduation Thesis

Scientific supervisor:
Assistant Prof., C.Sc. Yurii Litvinov

Reviewer:
developer at CJSC Production Company "SKB Kontur"
Anna Peresheina

Saint-Petersburg
2020

Оглавление

Введение

Применение в различных областях человеческой деятельности компьютеров и «умных устройств» заставляет многих осваивать программирование для убыстрения и упрощения различных процессов. Часто конечным пользователям приходится решать однотипные задачи, для которых освоение всех тонкостей какого-либо языка программирования общего назначения избыточно. Создание предметно-ориентированных языков для конкретной области применения позволяет реализовать для пользователя инструмент, помогающий решать ограниченный круг задач, но оттого более простой для изучения. В этом случае для того, чтобы запрограммировать необходимое поведение какой-либо системы, пользователю, знакомому с ее особенностями, достаточно усвоить основы адаптированного и упрощенного для конкретной области языка.

Еще одним упрощением программирования для конечного пользователя является переход от текстовых языков к программированию в терминах наглядных визуальных примитивов и диаграмм. Существующие универсальные визуальные языки, такие как UML, позволяют не только проектировать и сопровождать программное обеспечение, но и генерировать код по моделям. Тем не менее, такие языки довольно громоздки и использование их для любых возникающих задач не рационально. Применяя описанный выше предметно-ориентированный подход можно создавать визуальные языки для каждой конкретной области.

Использование узкоспециализированных визуальных языков требует наличия соответствующих инструментов. Кроме самого спроектированного языка, нужен редактор для его использования, генератор текстовой программы, средства проверки ограничений. Если весь набор необходимых инструментов реализовывать каждый раз заново, то создание подобных систем будет трудозатратным, а следовательно дорогим. Сами задачи в узких областях применения зачастую актуальны для небольшого числа специалистов, в связи с чем нецелесообразно тратить средства на создание для заранее ограниченного числа пользо-

вателей целой технологии с нуля. В таких случаях применяются DSM[1]-платформы — метатехнологии, позволяющие быстро создавать необходимые инструменты для работы с предметно-ориентированными визуальными языками [13].

Созданием таких технологий в рамках исследований визуальных языков занимаются на кафедре системного программирования в Санкт-Петербургском государственном университете. В разрабатываемых на кафедре DSM-платформах QReal [20] и REAL.NET [16] [12] для создания визуальных языков применяется метамоделирование [23]. По аналогии с тем, как грамматики задают текстовые языки, метамодели являются средствами задания визуальных языков, описывая все их корректные конструкции.

В платформе REAL.NET, в отличие от ее предшественника — платформы QReal, применен подход многоуровневого метамоделирования, позволяющий считать элементы метамодели как классами для объектов уровнями ниже, так и экземплярами классов верхних уровней. Таким образом создается иерархия метамоделей, от элементов которых разрешено инстанцироваться на более низких уровнях, если число метамоделей между двумя этими уровнями совпадает со специально заданным для метамодели числом-потенциалом. При этом сами собой разрешаются проблемы, с которыми сталкиваются разработчики, использующие двухуровневый подход, выделяющий только два уровня — модель на визуальном языке и метамодель языка. Такой проблемой является, например, невозможность создания без определения дополнительных ограничений класса и его экземпляра в одной модели [2].

Изначально использование платформы предполагало установку на компьютер необходимых программных модулей. Теперь же для удобства работы пользователя было решено перенести решение в веб, создав на базе предыдущей новую платформу REAL.NET Web [17]. Основная часть платформы, включающая репозиторий, переносится в виде микросервисов в серверную часть решения, а новые веб-редакторы являются клиентской стороной. В рамках системы разрабатывается специфич-

[1]Domain Specific Modeling

ный редактор для «умного дома», являющегося реализацией концепции «интернета вещей». Эта область в последнее время активно развивается в связи с распространением «умных устройств», которыми может управлять конечный пользователь в повседневной жизни для создания благоприятной обстановки в жилых помещениях. Кроме этого системе требуется универсальный редактор, конфигурируемый с помощью метамодели и представляющий модель в виде графа. Его можно будет использовать для различных областей применения, в том числе и для «умного дома». Хотя в этом случае редактор не настолько адаптирован под предметную область, как специально для нее созданный, зато должен позволить быстрее создавать различные готовые решения.

1. Постановка задачи

Целью работы является создание на базе платформы REAL.NET веб-платформы предметно-ориентированного моделирования и проведение апробации путем создания готового решения в рамках области «умных домов».

Для достижения этой цели были сформулированы следующие задачи.

- Выполнить проектирование архитектуры новой веб-платформы.
- Выполнить проектирование универсального веб-редактора и его коммуникаций с репозиторием платформы.
- Разработать универсальный веб-редактор, конфигурируемый с помощью метамодели языка.
- Провести апробацию решения путем проектирования визуального языка и реализации генерации по модели программы управления «умным домом».

Проект является групповым и прочие задачи решались другими участниками команды.

2. Обзор

2.1. Базовая платформа REAL.NET и многоуровневое метамоделирование

REAL.NET — это среда для предметно-ориентированного визуального моделирования, на основе которой создается новая веб-система REAL.NET Web. Платформа REAL.NET, в отличие от большинства DSM-решений, реализована на платформе .NET, что может в последующем упростить ее интеграцию с .NET приложениями. Для определения визуального языка в системе применяется метамоделирование. Метамодель задает синтаксически-верные конструкции языка, которые могут использоваться при конструировании пользователем модели.

Существует два подхода к метамоделированию. Одним из них является двухуровневый подход, при котором существует всего две модели. Одна из них — метамодель, задающая визуальный язык, вторая — модель пользователя на визуальном языке. При этом, если пользователь в своей модели создал класс, а затем ему понадобилось создать его экземпляр, то с точки зрения языка новый экземпляр в модели будет являться экземпляром элемента из метамодели и никаким образом не будет связан с заданным в модели классом. Для того, чтобы связать эти два элемента модели, разработчикам приходится добавлять правила валидации.

Другим подходом является многоуровневое метамоделирование. В этом случае сущности в разных метамоделях представляют собой одновременно и типы, и экземпляры. Таким образом, вышеупомянутую проблему с классами и экземплярами в модели можно решить, рассмотрев класс как элемент промежуточного уровня между уровнем метамодели языка и уровнем модели, в которой помещается экземпляр класса. Экземпляр таким образом будет находиться двумя уровнями ниже, чем метамодель, и инстанцироваться от определенного пользователем класса. Это помогает сразу передать все свойства класса объекту.

Рассмотрим такой пример: визуальный язык создан для управления

роботами и в языке есть элемент Speaker, который имеет атрибуты для определения громкости и типа звука. Если пользователь инстанцировал объект в модели от этого элемента и определил значения у атрибута громкости, то применение подхода многоуровневого метамоделирования позволит ему рассматривать новый объект как новый класс и создавать его экземпляры, в которых остается задать только атрибут типа звука. При этом у всех инстанцированных таким образом объектов можно одним изменением в их общем классе поменять значение громкости. Иерархия таких классов-объектов представлена на рисунке 1.

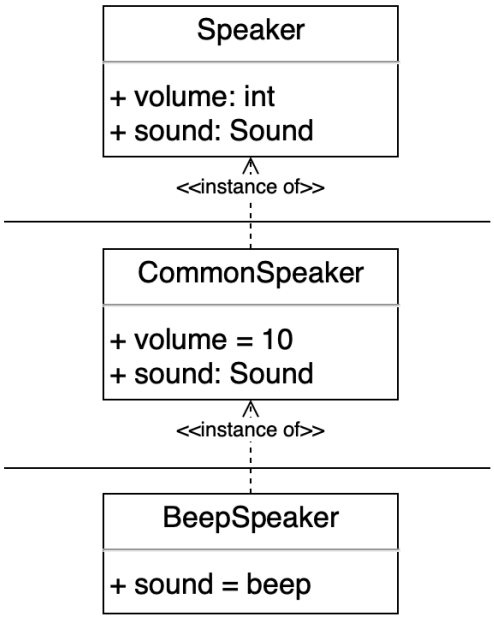

Рис. 1: Пример иерархии классов-объектов

На основе подхода многоуровневого метамоделирования реализован главный компонент платформы REAL.NET — репозиторий. Определенные в нем методы позволяют создавать языки и модели. Метаметаязык, используемый для конструирования метамоделей языков, состоит из нескольких уровней. На каждом из них располагается метамодель в терминах многоуровневого метамоделирования: она строится на основе верхних метамоделей и является метамоделью для нижних уровней. Составление такой иерархии помогает разделить между уровнями

функциональность. Например, в самом верхнем из них реализована базовая функциональность описания языков и возможность сохранения моделей, а на другом уровне, находящемся ниже, представлена инфраструктурная метамодель, определяющая возможности редактора.

В состав платформы REAL.NET также входят универсальные редакторы моделей, на основе которых создаются редакторы для конкретных визуальных языков, пример редактора платформы можно увидеть на рисунке 2. Один из редакторов системы реализован с помощью библиотеки Windows Forms, что позволяет ему быть запущенным на операционных системах, для которых существует реализация Mono Framework. Таким образом, решение REAL.NET может применяться на всех основных операционных системах. Тем не менее, для его использования нужна предварительная установка на компьютер конечного пользователя программных компонентов, которые из-за локальной установки сложно обновлять. Переход на веб-разработку позволит упростить этот этап работы с платформой и даст пользователю возможность получать доступ к системе с разных устройств.

2.2. Существующие аналоги

В обзор существующих аналогов были включены как DSM-платформы, имеющие веб-реализацию, так и визуальные редакторы графов.

Среда метамоделирования WebDPF

В основе веб-платформы WebDPF [4] лежит система DPF[2], написанная на языке Java. В ней используется подход многоуровневого метамоделирования, что задает формализм, позволяющий определять иерархию метамоделей. Платформа WebDPF дает пользователю возможность работать с любой моделью из иерархии в визуализированном виде. Сцена редактора разделена на четыре части. На основном из них показывается редактируемая модель. Остальные три используются для того, чтобы отобразить соответствующую рассматриваемой модели метамо-

[2]Diagram Predicate Framework

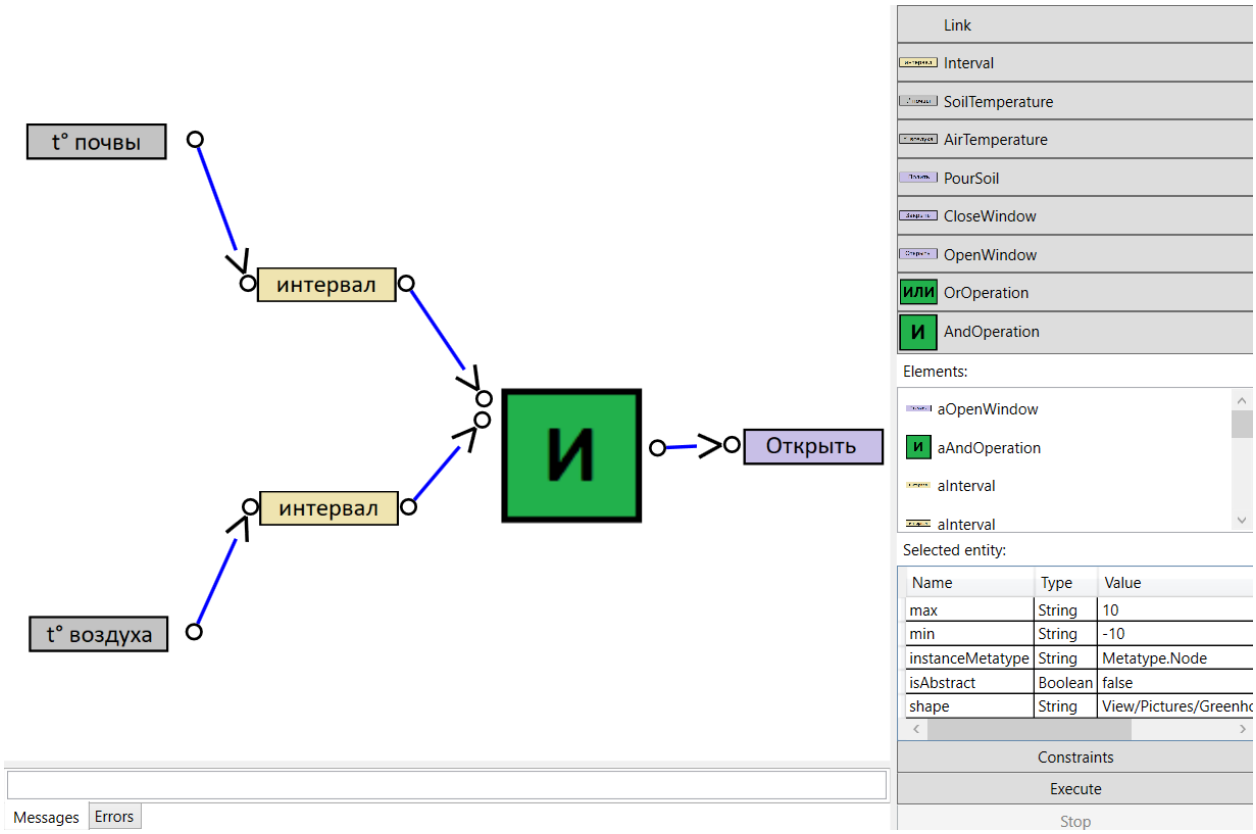

Рис. 2: Один из редакторов платформы REAL.NET

дель и задать необходимые ограничения. Для добавления в модель элементов из метамодели, а также для установки ограничений используется механизм drag-and-drop. При этом элемент из метамодели перетаскивается непосредственно из графа, представляющего метамодель, в граф модели. В редактор добавлены дополнительные возможности работы с большими моделями, занимающими много места на экране. Чтобы пользователю было удобнее ориентироваться, модель можно сворачивать, выполнять по ней поиск и фильтрацию. Также можно узнать, существует ли путь между определенными узлами графа. Всеми этими и другими возможностями пользователь может управлять на боковой панели инструмента, там же, где он выбирает модель для редактирования.

Платформа WebDPF разрабатывалась с использованием языка разметки HTML5 и динамического языка программирования JavaScript, что позволяет ей быть использованной в любых браузерах, поддержи-

вающих эти технологии. Система разрабатывалась в целях исследования подхода глубокого метамоделирования, ее авторы не преследовали цель создания универсального инструмента. С помощью платформы можно удобно редактировать иерархию метамоделей, но она довольно сложна для конечного пользователя.

Веб-платформа Web modeling project (WMP)

Платформа предметно-ориентированного моделирования WMP [18] является веб-инструментом. Ее отличие от новой разрабатываемой системы — это подход к созданию визуальных языков. В платформе WMP используется подход двухуровневого метамоделирования, поэтому разработчики столкнулись с описанными выше проблемами.

Веб-платформа WMP — клиент-серверное приложение. Основным языком разработки является Java. В начале система проектировалась как монолитные редактор и сервер, но со временем накопились сложные зависимости между входящими в монолитное решение частями. Для упрощения процесса разработки в последующем серверная часть была разделена на независимые микросервисы, а клиентская сторона — на плагины. Главным компонентом для клиента является редактор, написанный на языке TypeScript. На стороне клиента кроме модуля редактора также расположены модуль аутентификации и регистрации, модуль визуализации программы, заданной моделью. Разделение серверной части на микросервисы предполагает, что клиентская сторона обращается не к монолитному серверу, а только к отвечающему за конкретную функциональность модулю. Такими модулями являются, например, микросервис операций с пользовательскими данными или микросервис, представляющий репозиторий платформы.

Онлайн редакторы Visio и draw.io

Online Visio — веб-версия популярного редактора Visio [8], поддерживаемого корпорацией Microsoft. Существует также и мобильная версия редактора. Инструмент создан для редактирования различных типов схем и диаграмм. Как и в большинстве редакторов диаграмм, в

левой части главной страницы веб-приложения расположена палитра со всеми возможными узлами и связями. Все остальное пространство занимает сцена редактора. За время существования редактора сформировалась большая библиотека шаблонов, которые служат для быстрого создания диаграмм, применяемых в самых различных областях. Реализована возможность доступа нескольких пользователей к одной схеме. Кроме этого существуют расширения, использующие систему как основу для создания специализированных редакторов [11]. Продукт является платным.

Бесплатным аналогом редактора Visio является веб-инструмент с открытым исходным кодом draw.io [9]. Редактор написан с использованием языка JavaScript. Для реализации работы с графом используется библиотека mxGraph. Элементы палитры редактора разделены на группы, соответствующие одной области применения. Внешний вид редактора draw.io представлен на рисунке 3.

В упомянутых редакторах палитру удобно настраивать, оставляя в ней только необходимые наборы графических объектов. Таким образом пользователь конфигурирует редактор под свои потребности, обусловленные областью применения составляемой диаграммы. Но кроме сокращения числа объектов в палитре до минимального набора необходимых элементов, редактор никак не изменяется при определении области применения диаграммы. Например, для набора с графическими примитивами для «умного дома» может быть полезным добавить редактор свойств объектов с пунктом о номере порта, к которому подключается физическое устройство, задающееся узлом в модели. Но в редакторе все свойства элементов являются стандартными свойствами стилей.

2.3. Используемые инструменты

Библиотека mxGraph

MxGraph [7] — это JavaScript-библиотека визуализации диаграмм на стороне клиента. Для отрисовки используется формат изображений

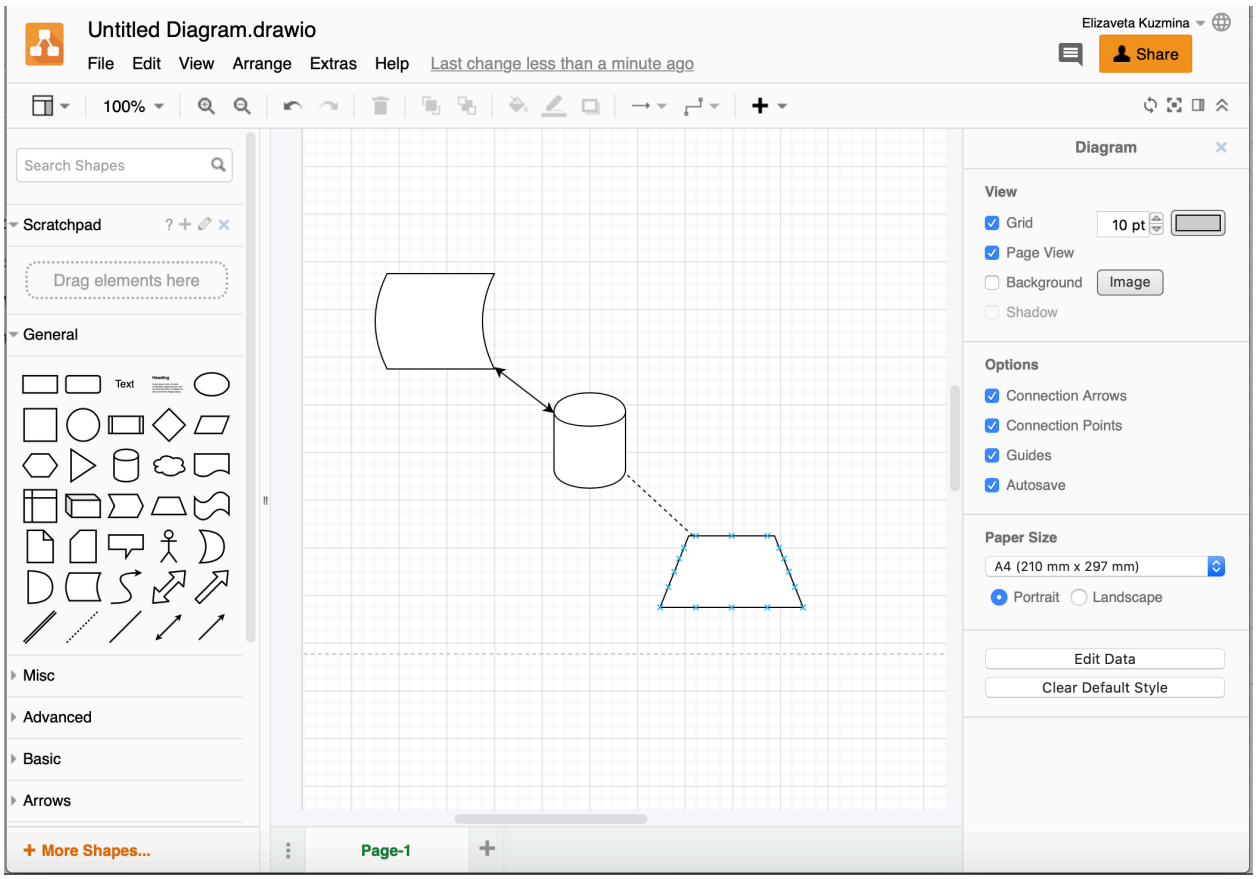

Рис. 3: Редактор draw.io

SVG и язык разметки HTML. Библиотека поддерживает работу с последними версиями большинства современных браузеров. Веб-редактор draw.io, рассмотренный выше и являющийся широко применяемым инструментом для визуализации графов, изначально создавался как апробация возможностей библиотеки mxGraph. Успешное применение данной библиотеки для создания клиентского приложения повлияло на выбор этого инструмента как основного при создании универсального веб-редактора новой платформы предметно-ориентированного моделирования.

В библиотеке определены методы для создания и изменения самого графа, методы для создания палитры перетаскиваемых элементов, изменения их стилей. В репозиторий, где лежит открытый исходный код библиотеки, авторами инструмента добавлены различные примеры использования для наглядной демонстрации его возможностей. Суще-

ствует документация, описывающая все методы, включенные в библиотеку [10].

Библиотека jQuery

Библиотека jQuery [6] является одним из самых распространенных инструментов, позволяющих упростить для разработчика взаимодействие между языком программирования JavaScript, языком разметки HTML и таблицами стилей CSS. Сразу несколько надежных CDN[3] серверов поддерживают данную библиотеку, что упрощает ее подключение.

Инструмент позволяет перехватывать события и обрабатывать их, выполнять асинхронные запросы для взаимодействия с сервером без перезагрузки веб-страницы. Также имеет большое количество подключаемых модулей для создания различных элементов пользовательского интерфейса. Таким образом, типовые задачи, с которыми сталкиваются веб-разработчики, требующие объемного кода без использования библиотеки, с ее помощью решаются в несколько строчек. Библиотека определяет удобный интерфейс для работы с DOM[4]-объектами. При этом программисту не нужно задумываться о том, на каком браузере его код будет исполняться, создатели инструмента позаботились о том, чтобы функции были совместимы со всеми основными браузерами.

Шаблонизатор Razor

Синтаксис разметки Razor [19] обычно применяется в ASP.NET MVC [1] веб-приложениях для внедрения в HTML-разметку представления результатов вычисления и значений элементов соответствующих им моделей. Но он также может быть использован как шаблонизатор для создания исходного кода программы, в которой некоторые участки кода задаются с помощью директив и преобразовываются в момент генерации. Но использование механизма при генерации обычного кода затруднено из-за недостатка специальных методов.

[3]Content Delivery Network

[4]Document Object Model

Для упрощения взаимодействия с механизмом Razor на его основе реализован движок шаблонов RazorEngine [15]. В нем определены методы для динамической генерации текста по шаблону с использованием синтаксиса Razor. При генерации статический класс Engine принимает шаблон и объект модели. В средах разработки, таких как VisualStudio и Rider, удобно подсвечиваются элементы синтаксиса Razor, что дает возможность быстро находить и исправлять ошибки и опечатки, в отличие, например, от использования шаблонов T4 [3], где и текст самой программы, и директивы для генерации являются однотонным кодом и не выделяются на фоне друг друга.

Библиотека Reactive Extentions

Библиотека Reactive Extentions [5] от компании Microsoft предоставляет возможность быстро создавать цепочки обработки данных. С ее помощью можно создать последовательность объектов, реагирующих на изменения друг друга и передающих друг другу данные. В библиотеки определены три основных интерфейса: IObserver, IObservable и ISubject. Первый из них может подписываться на изменения наследников интерфейса IObserveble. Интерфейс IObserver задает методы для обработки полученной информации, а также маркеров окончания последовательности, которые говорят о прекращении поступления в систему новых данных и о завершении работы цепочки. Интерфейс ISubject определяет элементы, являющиеся одновременно и генераторами новых событий и наблюдателями за обновлениями других объектов.

Библиотеку удобно использовать при генерации кода по модели-графу, так как в основном такие диаграммы задают процесс обработки потока данных. В таком случае элементы, имеющие исходящие ребра, удобно считать наследниками интерфейса IObservable, как элементы, передающие данные. А элементы, находящиеся на концах отношений, удобно рассматривать как элементы IObserver, принимающие информацию. Узлы, находящиеся в центре, сочетают в себе свойства этих двух интерфейсов, то есть считаются объектами ISubject в терминах библиотеки Reactive Extentions.

3. Архитектура веб-платформы REAL.NET Web

Новый веб-инструмент предметно-ориентированного визуального моделирования REAL.NET Web является клиент-серверным приложением. Так как базовая платформа REAL.NET писалась на платформе Microsoft.NET, то веб-решение реализуется на платформе разработки веб-приложений ASP.NET [21]. Описываемый в данной работе универсальный веб-редактор моделей на визуальных языках в совокупности со специфичными веб-редакторами для разных областей применения считается клиентской стороной. Редакторы общаются с серверной частью для авторизации пользователей, создания моделей в базе пользователя, получения информации из репозитория платформы.

Серверная часть реализует микросервисную архитектуру. При проектировании общего решения была проанализирована и разделена на независимые части необходимая функциональность сервера платформы. На основе такого разделения были выделены независимые микросервисы, каждый из которых позволяет решать строго ограниченный набор задач. Одним из плюсов такого подхода к проектированию системы является возможность размещения различных компонент сервера на разных машинах. Чтобы унифицировать общение с разными микросервисами, редактор системы отправляет запросы к компонентам сервера через специально добавленный шлюз. Общение осуществляется посредством REST-запросов. Микросервисный подход позволяет легко изменять существующие сервисы и добавлять к серверной части новые модули, например, специфичные для языков редакторы, не затрагивая функциональность других частей. В виде микросервисов представлены репозиторий платформы, хранилище моделей, модуль авторизации, генераторы кода. Архитектура новой веб-платформы представлена на рисунке 4.

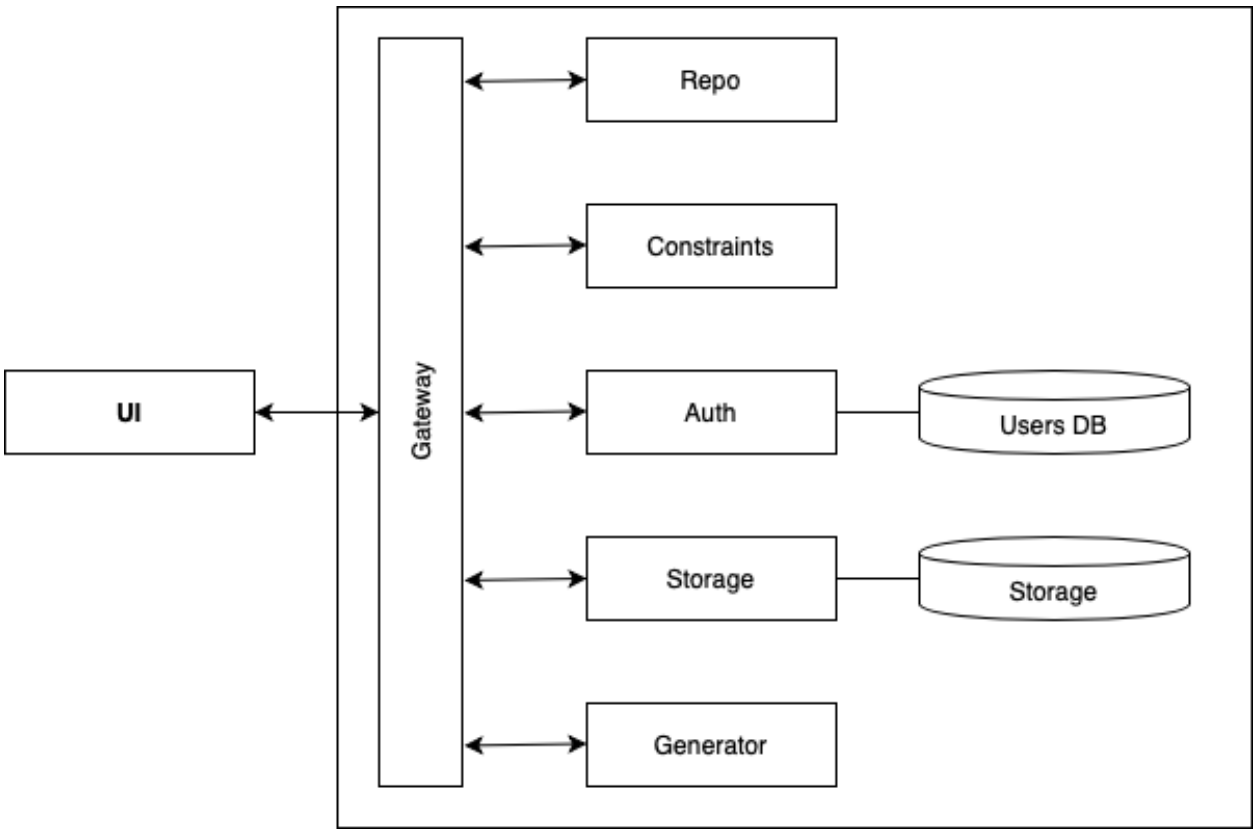

Рис. 4: Архитектура платформы REAL.NET Web

4. Архитектура универсального редактора

При проектировании редактора был использован паттерн Model-View-Controller [14]. Применение шаблона позволяет отделить представление на экране от бизнес-логики. Архитектура представлена на рисунке 5. Для сцены редактора и для палитры заданы свои контроллеры. При каждом действии пользователя они помогают синхронизировать измененную визуальную модель на сцене с ее представлением в системе.

Для реализации механизма Undo/Redo был применен паттерн проектирования Command [22] и в архитектуру изначально был добавлен менеджер команд. Все возможные действия пользователя с моделью в редакторе заданы с помощью специальных классов, наследующихся от общего интерфейса Command. При совершении пользователем действия создается экземпляр соответствующей команды, который не

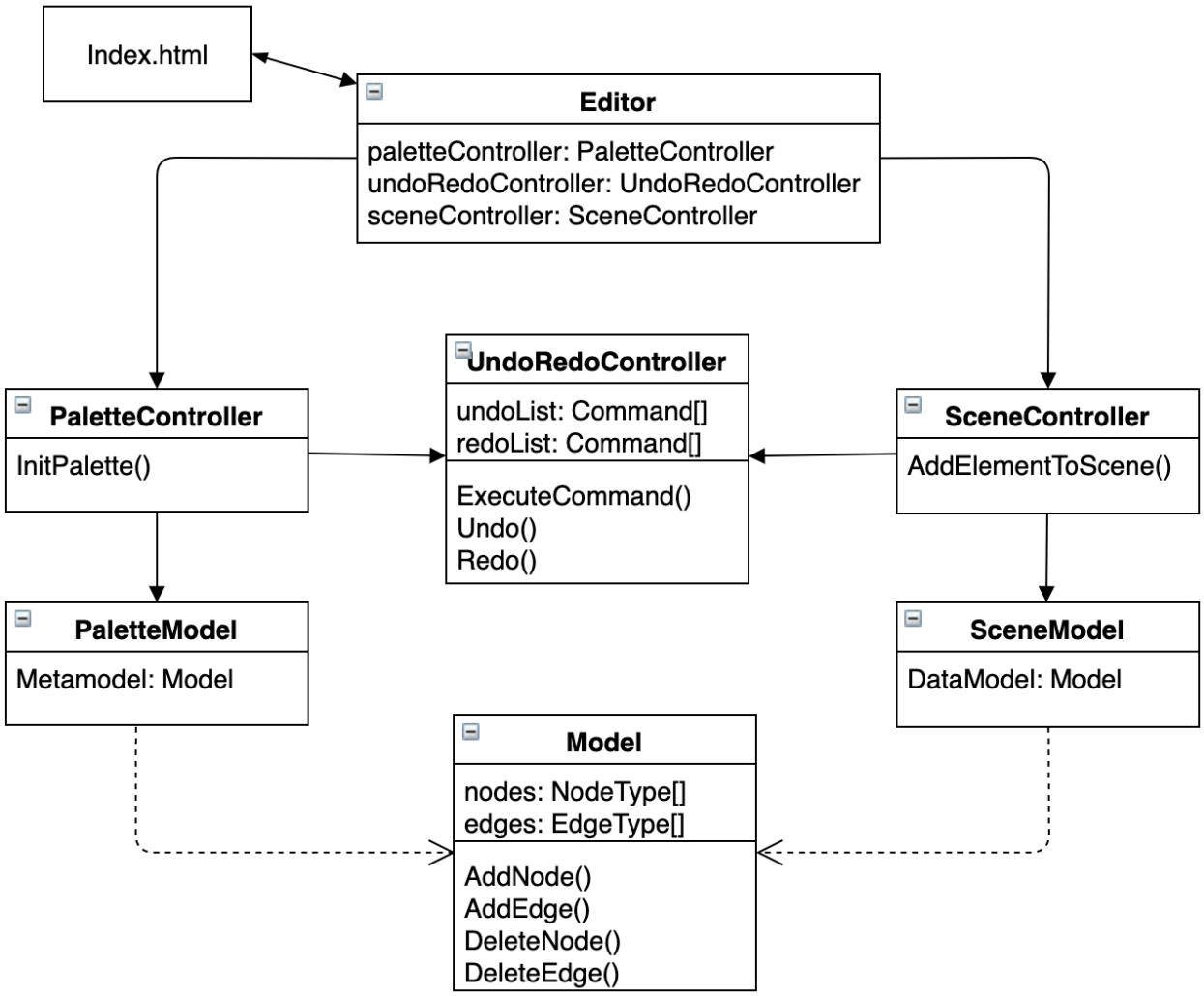

Рис. 5: Архитектура универсального редактора

только исполняется, но и помещается в список Undo-действий, то есть действий, которые можно отменить. До тех пор, пока пользователь не решил воспользоваться отменой, этот список пополняется. Как только пользователь переходит от отмены действий к выполнению новых команд, список Redo-действий, то есть команд, которые можно после отмены выполнить заново, очищается и в него при каждом последующей отмене помещаются команды с конца Undo-списка. Аналогично при переходе к исполнению Redo-команд соответствующие элементы из конца данного списка переносятся в конец списка Undo-действий. Как только пользователем создается новая команда, Redo-список очищается. Архитектура контроллера Undo/Redo действий представлена

на рисунке 6.

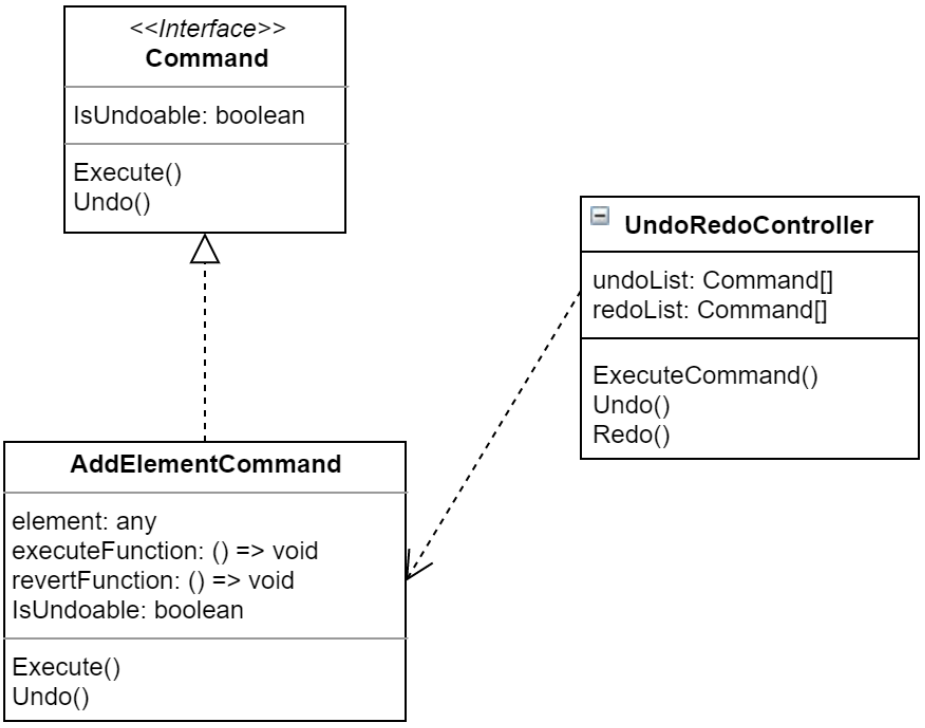

Рис. 6: Undo/Redo менеджер

При исполнении команд, содержащих информацию о действиях с моделью, эта информация передается в репозиторий системы. Взаимодействие редактора с микросервисом реализовано с помощью программного интерфейса, сформированного из REST-запросов, предназначенных для использования методов репозитория. Так, в набор доступных команд, позволяющих работать с моделью, входят запросы для создания и получения по названию самой модели, запросы для получения по идентификатору входящего в модель элемента, добавления узлов и связей, редактирования их свойств. С помощью методов программного интерфейса при запуске приложения из репозитория достается информация о метамодели языка. Для этого отправляется запрос на получение модели с указанием имени метамодели.

5. Реализация графового веб-редактора

Модель в репозитории платформы представляется набором элементов и связей. Универсальный редактор системы позволяет создавать и редактировать модель в виде графа. Внешний вид графового редактора представлен на рисунке 7. В левой части редактора располагается палитра с элементами метамодели языка. Визуальное представление каждого такого элемента определяется с помощью значения атрибута shape, имеющегося у любого объекта языка, и обычно задающегося при проектировании визуального языка как путь к файлу с подходящим по смыслу изображением. При создании экземпляра элемента в модели добавляются и соответствующие атрибуты. Поэтому при перетаскивании из палитры элемента метамодели на сцене — центральной части редактора, появляется вновь созданный элемент модели с идентичными визуальными свойствами.

Для редактирования визуального стиля объекта в правом верхнем углу страницы располагается редактор стилей. На рисунке 7 можно видеть, что у выделенного объекта было изменено значение свойства стиля shape со значения image на значение ellipse, а также определен цвет графического примитива. В модели это все тот же экземпляр элемента из метамодели, сохраняющий все остальные свойства.

Не визуальные свойства элементов модели, которые как и картинка изначально задаются значениями по умолчанию из метамодели языка, можно менять с помощью редактора свойств в правом нижнем углу веб-страницы. Такими атрибутами элемента может быть, например, его имя или номер реального физического устройства, символизируемого графическим объектом.

Универсальный редактор написан на языке TypeScript. Применение библиотеки mxGraph позволяет визуализировать граф модели. Средствами библиотеки реализована как палитра, так и перемещение элементов способом drag-and-drop по сцене редактора. При изменениях стилей объектов элементы сразу перерисовываются. Для достижения такого эффекта в библиотеке не достаточно поменять модель графа

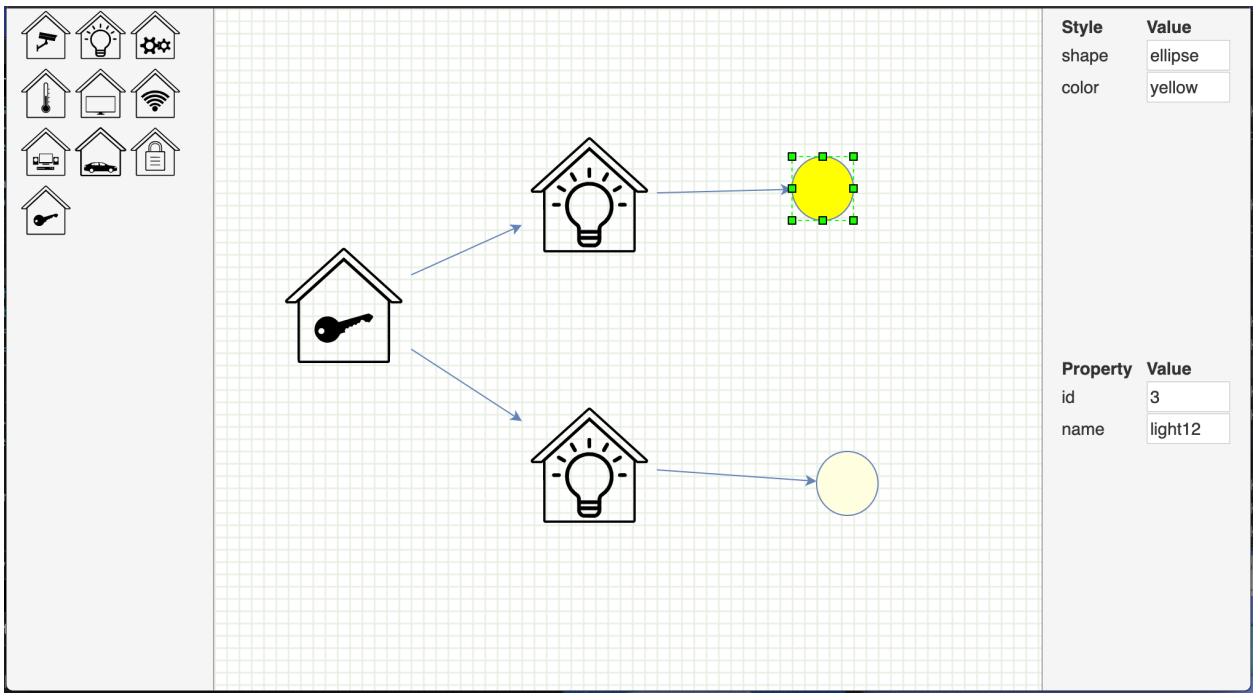

Рис. 7: Графовый редактор DSM-платформы REAL.NET Web

в терминах mxGraph и обновить представление, необходимо вызвать специальный метод для обновления графа. Библиотека имеет документацию, в которой перечислены все предоставляемые методы работы с диаграммой, но описания часто очень короткие и подобные особенности реализации в ней редко затрагиваются.

При изменении модели посылаются запросы к серверной части. Такими запросами являются получение метамодели как модели, добавленный к программному интерфейсу репозитория запрос на получение идентификаторов всех не абстрактных элементов (для добавления только таковых в палитру), создание модели, добавление в модель новых элементов, редактирование их свойств.

Разделение сцены на участки реализовано с помощью библиотеки Bootstrap, а с помощью библиотеки JQuery становится возможным удобно управлять DOM-объектами из кода программы. Для сборки решения используется инструмент webpack, который позволяет компилировать JavaScript-модули приложения в один файл и добавлять к сборке зависимости. Кроме этого, инструмент позволяет удобно отлаживать программу в браузере, показывая исходные файлы по отдельности для

лучшей навигации по коду в процессе выполнения.

6. Апробация

Для проведения апробации системы необходимо было разработать на ее основе решение для конкретной предметной области. В качестве такой области была выбрана сфера «умных домов». Кроме проектирования метамодели нового языка требовалось реализовать генератор для превращения визуальной модели в текстовую программу.

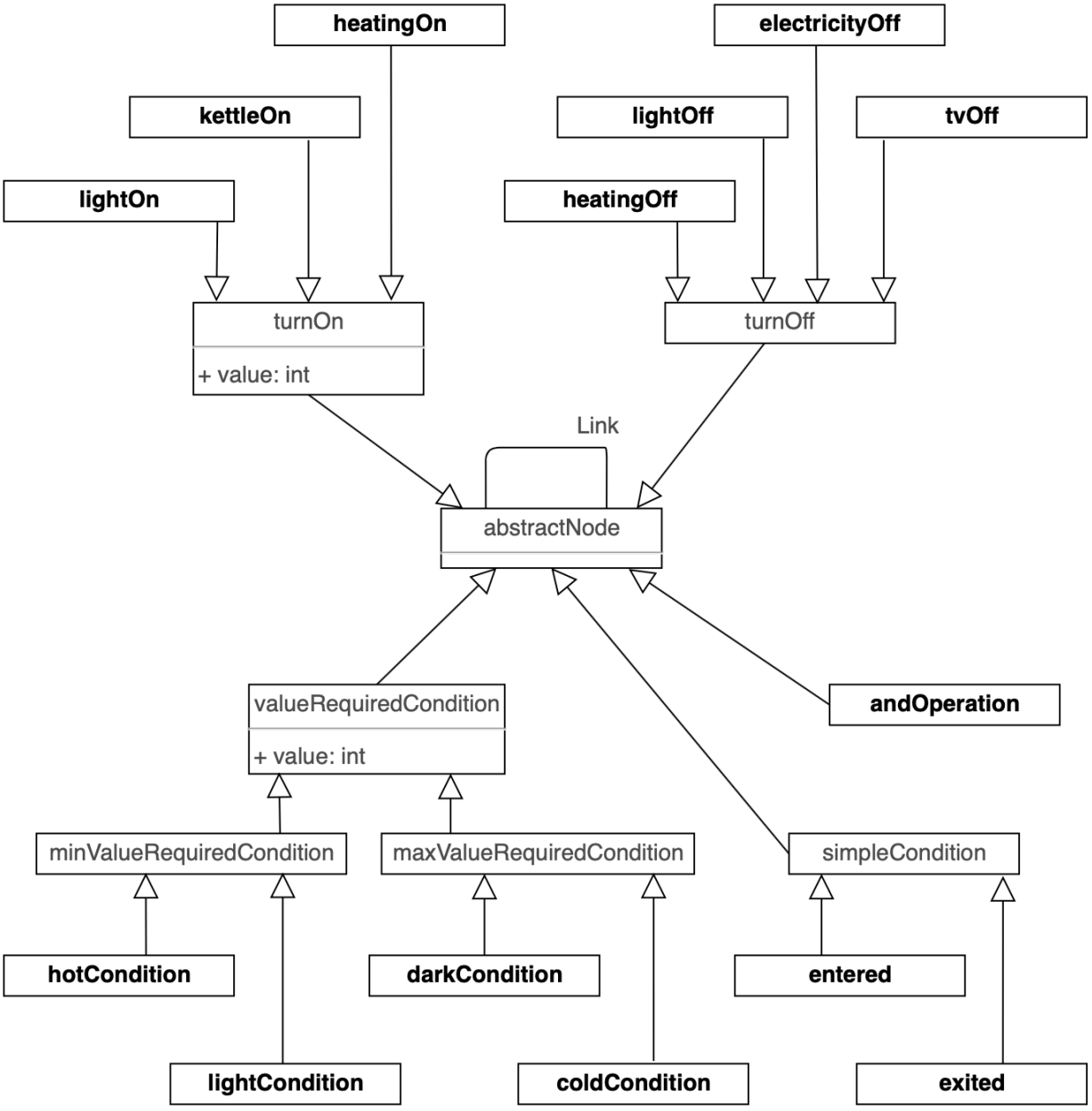

Рис. 8: Метамодель визуального языка для «умного дома»

6.1. Визуальный язык для «умного дома»

Выбранная для апробации область «умных домов» определила специфику созданного для апробации тестового визуального языка, метамодель которого представлена на рисунке 8. Главным элементом метамодели является абстрактный узел — экземпляр элемента из инфраструктурной метамодели иерархии. От него наследуются другие узлы языка. Элементы можно разделить по предназначению на три части: факторы окружающей среды, влияющие на состояние системы «умного дома», логическая операция И для комбинирования этих факторов и предполагаемые действия в случае срабатывания условий, определенных факторами.

Для удобства использования языка условия среды разделены на простые, имеющие всего два состояния, например, возвращение человека домой, и сложные, требующие определения значения, при котором условие считается действительным, например, уровень освещенности. При этом сложные условия разделены на еще две группы. Одна из них задает факторы окружающей среды, определяемые с помощью верхней границы значений, например, то, что на улице темно, определяется исходя из того, не выходит ли значение освещенности за заданную максимальную границу. Вторая же группа задает условия, определяемые с помощью минимального значения, например, обозначающего то, что на улице светло, если достигнут минимальный показатель освещенности. Таким образом, например, при написании сценариев «светло –> открыть шторы», «темно –> занавесить окна» пользователь будет использовать разные говорящие картинки, а не одну, отвечающую за освещенность. Такие узлы предполагают наличие свойства «значение», поэтому они наследуются от общего абстрактного элемента valueRequiredCondition и затем уже делятся на требующие определения максимальной или минимальной границы.

Отвечающие за совершаемые действия элементы делятся на элементы-включатели и элементы-выключатели. Во втором случае нет необходимости в задании какого-либо атрибута-значения, а в первом в наследу-

емом абстрактном узле определен атрибут, отвечающий за мощность включения. Такое разделение позволяет использовать простые картинки для демонстрации противоположных состояний. Например, если бы включение и выключение отопления задавалось бы одним элементом с указанием значения включить/выключить, то чтобы пользователь случайно не перепутал состояния, пришлось бы добавлять эффекты для их визуального различия.

В метамодели языка задано одно отношение ассоциации, являющееся экземпляром ассоциации инфраструктурного слоя. Для него задается начальная и конечная вершина, наследуемые от абстрактного узла метамодели.

6.2. Генератор текста

Генератор для нового языка помещается в серверной части решения в виде микросервиса. При POST-запросе к серверу в качестве параметров задаются имя обрабатываемой модели и словарь с ключами и значениями типа string. Словарь используется в случае необходимости передачи генератору специфических параметров. Например, если в дальнейшем будет реализовано несколько шаблонов генерируемого кода на разных языках программирования для разного оборудования, то выбранный пользователем в редакторе язык может быть указан в запросе. Перед началом генерации из репозитория достается модель с заданным именем. В генераторе реализована возможность представления загруженной модели в удобном виде для составления модели, передаваемой в используемый шаблон.

Для генерации кода по модели был выбран механизм Razor и реализованный на его основе инструмент RazorEngine, позволяющий динамически генерировать текст по шаблону. В cshtml-шаблоне для составления сценария работы «умного дома» описываются построения из элементов модели цепочек подписанных друг на друга объектов в терминах библиотеки Reactive Extentions. Это позволяет удобно подписывать элементы на события друг друга, передавая таким образом

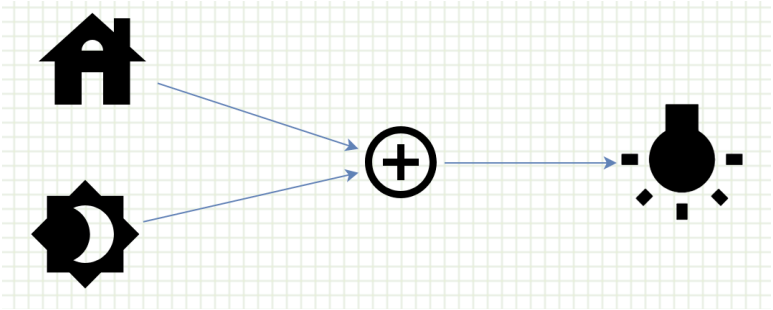

Рис. 9: Модель на визуальном языке для «умного дома»

данные от элементов, расположенных в начале отношения ассоциации, элементам на ее конце. Для проверки правильности сгенерированного сценария для всех узлов-условий создается Observable элемент в терминах Reactive Extentions и соответствующий ему объект-симулятор для генерации со случайными интервалами значений изменения окружающей среды. При срабатывании событий, кроме передачи данных, в консоль записываются соответствующие изменения. Часть сгенерированного кода по модели на рисунке 9 представлена на листинге кода 1. Элемент element0 относится к узлу включения света (лампочка), element1 отвечает за работу центрального элемента — логической операции И, element2 относится к условию захода в дом и element3 — к узлу условия освещенности.

6.3. Конфигурирование редактора для языка «умного дома»

Для конфигурирования редактора потребовалось только поменять в коде имя загружаемой метамодели на название метамодели языка «умного дома». Она определила возможные конструкции на сцене редактора, элементы метамодели были отображены на палитре. Таким образом, полученный редактор уже можно считать настроенным на работу с визуальным языком «умного дома». Редактор можно видеть на рисунке 10.

В результате апробации было показано, что на основе новой веб-платформы можно быстро строить решение для конкретной предмет-

```
Actuator element0;
AndOperation element1;
Condition element2;
Condition element3;

element0 = new Actuator(0, 100);
IObservable<int> observable0 =
    System.Reactive.Linq.Observable.FromEventPattern<int>(
        h => element0.Event += h, h => element0.Event -= h)
            .Select(e => e.EventArgs).Synchronize().DistinctUntilChanged();
IObserver<int> observer0 = Observer.Create<int>(x => element0.Action(x));
ISubject<int> reactElement0 = Subject.Create<int>(observer0, observable0);

element1 = new AndOperation(1);
IObservable<int> observable1 =
    System.Reactive.Linq.Observable.FromEventPattern<int>(
        h => element1.Event += h, h => element1.Event -= h)
            .Select(e => e.EventArgs).Synchronize().DistinctUntilChanged();
IObserver<int> observer1 = Observer.Create<int>(x => element1.Action(x));
ISubject<int> reactElement1 = Subject.Create<int>(observer1, observable1);

element2 = new Condition(2);
IObservable<int> observable2 =
    System.Reactive.Linq.Observable.FromEventPattern<int>(
        h => element2.Event += h, h => element2.Event -= h)
            .Select(e => e.EventArgs).Synchronize().DistinctUntilChanged();
IObserver<int> observer2 = Observer.Create<int>(x => element2.Action(x));
ISubject<int> reactElement2 = Subject.Create<int>(observer2, observable2);

element3 = new Condition(3, 5, false);
IObservable<int> observable3 =
    System.Reactive.Linq.Observable.FromEventPattern<int>(
        h => element3.Event += h, h => element3.Event -= h)
            .Select(e => e.EventArgs).Synchronize().DistinctUntilChanged();
IObserver<int> observer3 = Observer.Create<int>(x => element3.Action(x));
ISubject<int> reactElement3 = Subject.Create<int>(observer3, observable3);

element1.IncomingValues.Add(2, null);
element1.IncomingValues.Add(3, null);

var sub0 = reactElement1.Subscribe(reactElement0);
var sub1 = reactElement2.Subscribe(reactElement1);
var sub2 = reactElement3.Subscribe(reactElement1);
```

Listing 1: Пример сгенерированного кода

ной области. При использовании универсального графового редактора настройка редактора для нового языка требует минимальных уси-

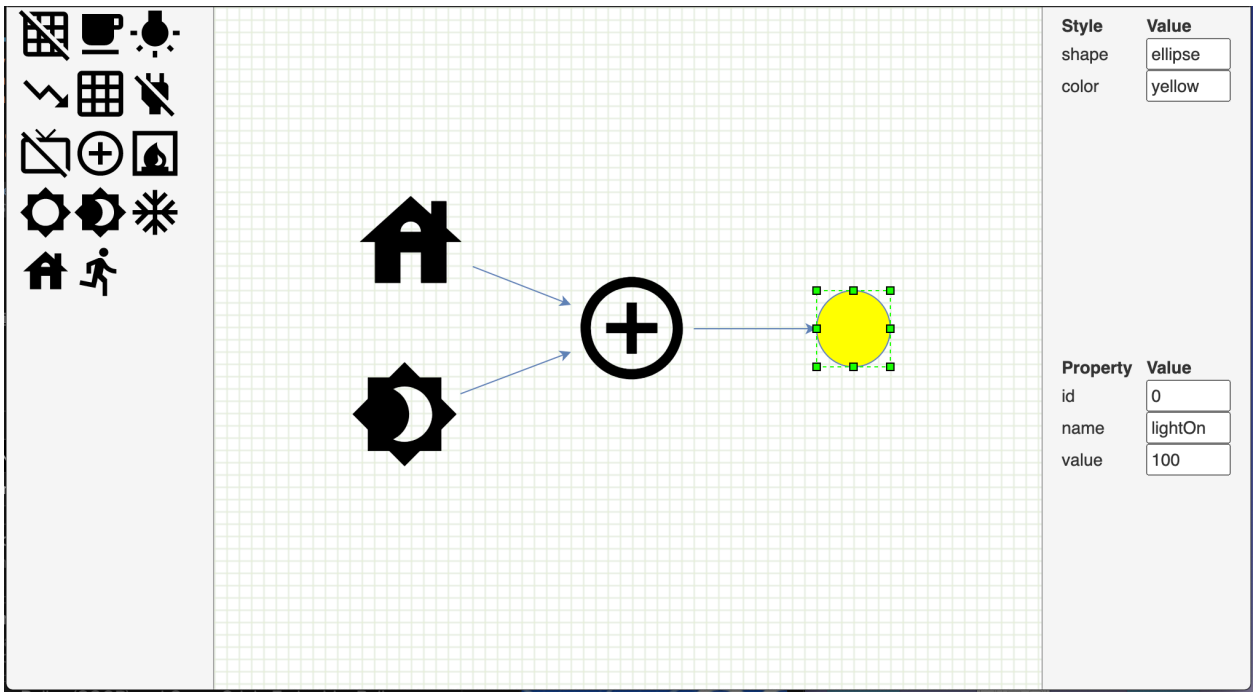

Рис. 10: Сконфигурированный редактор для «умного дома»

лий. Если системе нужен будет специфичный визуальный инструмент для составления моделей, то на создание редактора потребуется больше времени, но при этом уже реализованные части системы, позволяющие работать с языками, гарантируют, что решение с добавлением нового редактора, нового генератора и нового языка, будет разрабатываться быстрее, чем вся система с нуля.

Заключение

В ходе данной работы были получены следующие результаты.

- Спроектирована архитектура новой веб-платформы предметно-ориентированного моделирования.
- Спроектирована архитектура универсального веб-редактора графов для визуальных языков, конфигурируемого с помощью метамодели, и определены механизмы коммуникации между репозиторием платформы и редактором для синхронизации моделей на сцене и в памяти системы.
- Реализован на языке TypeScript универсальный графовый веб-редактор для визуальных языков.
- Проведена апробация системы путем создания решения для «умного дома»: создан новый визуальный язык, с помощью шаблонизатора Razor реализован генератор кода для него, на основе универсального редактора сконфигурирован веб-редактор для нового языка.

  Результаты данной работы были представлены вместе с описанием веб-платформы REAL.NET Web на конференции по программной инженерии и организации информации SEIM [22].

Список литературы

[1] ASP.NET MVC. — URL: https://dotnet.microsoft.com/apps/aspnet/mvc (дата обращения: 26.05.2020).

[2] Atkinson Colin, Kühne Thomas. The essence of multilevel metamodeling. — Springer, Berlin, Heidelberg, 2001. — P. 19–33.

[3] Text Template Transformation Toolkit. — URL: https://msdn.microsoft.com/ru-ru/library/ee844259.asp (дата обращения: 26.05.2020).

[4] WebDPF: A web-based metamodelling and model transformation environment / F Rabbi, Y Lamo, IC Yu, LM Kristensen. — 2016. — P. 87–98.

[5] Библиотека Reactive Extentions. — URL: http://reactivex.io (дата обращения: 26.05.2020).

[6] Библиотека jQuery. — URL: https://jquery.com (дата обращения: 14.05.2020).

[7] Библиотека mxGraph. — URL: https://github.com/jgraph/mxgraph (дата обращения: 14.05.2020).

[8] Веб-редактор Visio. — URL: https://office.live.com/start/visio.aspx (дата обращения: 14.05.2020).

[9] Веб-редактор draw.io. — URL: https://app.diagrams.net (дата обращения: 14.05.2020).

[10] Документация библиотеки mxGraph. — URL: https://jgraph.github.io/mxgraph/ (дата обращения: 14.05.2020).

[11] Кознов Д. В. Иванов А. Н. Поддержка концептуального моделирования при разработке визуальных языков с использованием Microsoft DSL Tools. — Издательство Санкт-Петербургского университета, 2009. — P. 105–127.

[12] Литвинов Ю.В. Кузьмина Е.В. Небогатиков И.Ю. Алымова Д.А. Среда предметно-ориентированного визуального моделирования REAL.NET // Всероссийская научная конференция по проблемам информатики СПИСОК-2017. — 2017. — URL: http://spisok.math.spbu.ru/2017/txt/SPISOK-2017.pdf (дата обращения: 14.04.2018).

[13] Павлинов А.А., Кознов Д. В., Перегудов А.Ф. и др. О средствах разработки предметно-ориентированных визуальных языков. — URL: https://www.math.spbu.ru/user/dkoznov/papers/SurveyDSMplatforms.pdf (дата обращения: 14.04.2018).

[14] Паттерн Model-View-Controller. — URL: http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/mvc.html (дата обращения: 26.05.2020).

[15] Проект RazorEngine. — URL: https://github.com/Antaris/RazorEngine (дата обращения: 26.05.2020).

[16] Репозитории проекта REAL.NET. — URL: https://github.com/yurii-litvinov/REAL.NET (дата обращения: 14.05.2020).

[17] Репозитории проекта REAL.NET Web. — URL: https://github.com/REAL-NET (дата обращения: 26.05.2020).

[18] Репозиторий проекта WMP. — URL: https://github.com/qreal/wmp (дата обращения: 14.05.2020).

[19] Синтаксис Razor. — URL: https://docs.microsoft.com/ru-ru/aspnet/core/mvc/views/razor?view=aspnetcore-3.1 (дата обращения: 25.05.2020).

[20] Терехов А.Н. Брыксин Т.А. Литвинов Ю.В. и др. Архитектура среды визуального моделирования QReal. — Системное программирование. Вып. 4. – СПб.: Изд-во СПбГУ, 2009. — С. 171–196.

[21] Фреймворк ASP.NET. — URL: https://dotnet.microsoft.com/apps/aspnet (дата обращения: 26.05.2020).

[22] Э. Гамма Р. Хелм Р. Джонсон Дж. Влиссидес. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. — ИД Питер, 2001. — P. 367.

[23] Ю.В. Литвинов. Методы и средства разработки графических предметно-ориентированных языков. — URL: https://disser.spbu.ru/files/disser2/727/disser/dissertation_Litvinov_4-12-2015.pdf (дата обращения: 14.05.2020).