МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра компьютерных технологий

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Образовательная программа: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РАЗРАБОТКА WYSIWYG РЕДАКТОРА ДЛЯ МОДУЛЯ ДИНАМИЧЕСКОЙ ВЕРИФИКАЦИИ ПРОЦЕСС-ОРИЕНТИРОВАННЫХ АЛГОРИТМОВ УПРАВЛЕНИЯ

утверждена распоряжением проректора по учебной работе №0290 от «30» сентября 2019 г.

Витченко Владислав Алексеевич, группа 16206 _____

«К защите допущена»

Зав. кафедрой КТ ФИТ НГУ,

д. т. н., доцент,

Зюбин В.Е. /……..

«……»……20…г.

Руководитель ВКР

Зав. кафедрой КТ ФИТ НГУ,

д. т. н., доцент,

Зюбин В.Е. /……...

«……»……20…г.

Соруководитель ВКР

б/с, ассистент кафедры ОИ ФИТ НГУ

Лях Т.В. /……

«……»……2020 г.

Дата защиты: «……»……2020 г.

Новосибирск, 2020 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра компьютерных технологий
Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УТВЕРЖДАЮ
Зав. кафедрой Зюбин В.Е.
……..
(подпись)
«30» сентября 2019 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студенту Витченко Владиславу Алексеевичу, группы 16206

Тема: Разработка WYSIWYG редактора для модуля динамической верификации процесс-ориентированных алгоритмов управления

утверждена распоряжением проректора по учебной работе №0290 от «30» сентября 2019 г.

Срок сдачи студентом готовой работы «31» мая 2020 г.

Исходные данные (или цель работы): разработка WYSIWYG редактора макета графического представления объекта управления для динамической верификации алгоритмов управления, создаваемых на языке Reflex

Структурные части работы: исследовательская (текст ВКР), программный модуль

Руководитель ВКР
зав. Каф КТ ФИТ НГУ,
Д.т.н.,
Зюбин В.Е. /……..
«30» сентября 2019 г.

Задание принял к исполнению

Витченко В.А. /……
«30» сентября 2019 г.

ОГЛАВЛЕНИЕ

СПИСОК СОКРАЩЕНИЙ И ОПРЕДЕЛЕНИЙ

WYSIWYG — What You See Is What You Get;

АУ — Алгоритм Управления;

КФС — Кибер-физические системы;

LabVIEW — Laboratory Virtual Instrumentation Engineering Workbench;

ОУ — Объект Управления;

XML — eXtensible Markup Language;

IDE — Integrated Development Environment;

ПО — Программное Обеспечение;

GUI — Graphical User Interface;

JSON-RPC — JavaScript Object Notation Remote Procedure Call;

API — Application Programming Interface;

DSM — Domain Specific Modules;

AST — Abstract Syntax Tree;

CSS — Cascading Style Sheets;

MVC — Model-View-Controller;

JSON — JavaScript Object Notation;

ВВЕДЕНИЕ

Сложность промышленных процесс-ориентированных алгоритмов управления (АУ) кибер-физическими системами растёт. Также растет тяжесть последствий из-за ошибок в АУ.

В связи с этим исследователи всё больше интересуются разработкой специализированных языков программирования АУ и методами верификации создаваемых АУ.

В ИАИЭ был разработан язык Reflex для эффективного описания АУ КФС и комплекс динамической верификации, который через тестирование проверяет АУ, написанный на языке Reflex, на соблюдение заданных свойств.

Комплекс динамической верификации реализован в виде LabVIEW приложение, интерфейс которого включает в себя окно с графическим представлением объекта управления. Графическое представление включает мнемосхему объекта управления (ОУ) с индикаторами состояния входных и выходных сигналов (датчиков и управляющих механизмов).

Входные данные этого модуля представляют собой XML документ, который задается вручную. В связи с данным подходом создание и модификация графических представлений ОУ является трудоемким процессом.

Стандартный подход для сокращения трудоемкости таких задач — это использование специализированных визуальных редакторов.

Поэтому целью работы стала разработка WYSIWYG редактора макета графического представления объекта управления для динамической верификации алгоритмов управления, создаваемых на языке Reflex. Который позволит значительно сократить трудоемкость процесса задания графического представления ОУ.

Поставлены следующие задачи:

1. Проанализировать специфику комплекса динамической верификации алгоритмов управления, создаваемых на языке Reflex, и Reflex IDE.

2. Сформулировать требования к разрабатываемому программному обеспечению.
3. Разработать интерфейс WYSIWYG редактора.
4. Спроектировать архитектуру приложения.
5. Определить формат описания графического представления ОУ.
6. Реализовать приложение.

Работа разбита 3 на главы: “Анализ”, “Проектирование”, “Реализация”. В главе “Анализ” приводятся результаты исследования предметной области и представлены технические требования к проекту. В главе “Проектирование” описана архитектура проекта и алгоритмы его работы. В главе “Реализация” описаны детали реализации проекта.

АНАЛИЗ

1. Особенности разработки АУ КФС и принцип виртуальных объектов управления

Киберфизические системы, в том числе индустриальные промышленные системы, объединяют в себе физические и информационные компоненты. Информационные компоненты — это компьютерный алгоритм управления системой. Физические компоненты — это объекты управления, например, производственный конвейер, комплекс по выращиванию монокристаллического кремния или ядерный реактор.

Как и любое ПО, программные комплексы, управляющие кибер-физической системой, требует отладки. Отлаживать алгоритм на реальном объекте управления не является целесообразным из-за высокой стоимости ошибок и риска возникновения катастрофы при неправильной работе алгоритма управления.

Для решения этой проблемы используются программные имитаторы объектов управления [1].

По сравнению с физическими моделями виртуальные объекты управления имеют ряд очевидных преимуществ, дающих существенное сокращение материальных и временных затрат при отладке алгоритмов управления.

Большинство разрабатываемых алгоритмов управления представляет собой сложные системы взаимодействия десятков процессов, что заставляет прибегнуть к использованию итерационного метода разработки алгоритмов.

2. Динамическая верификация

В ИАИЭ разработан программный комплекс автоматической динамической верификации алгоритмов управления, реализованный в виде

LabVIEW-приложения. В комплексе присутствует модуль графического представления объекта управления.

Комплекс отображает состояние объекта управления во время работы алгоритма управления в виде мнемосхемы с индикаторами (см. рис. 1).

Рисунок 1. окно модуля графического представления для ОУ “Перекрёсток”.

В качестве входных данных модуль принимает конфигурационный XML файл и схематичное изображение объекта управления. Учитывая сложность алгоритмов управления, нередко изменение конфигурационного файла в соответствии с изменением кода является неоправданно трудозатратной задачей.

Для уменьшения трудозатрат при задании подобного рода схем следует использовать визуальные редакторы, реализующие принцип WYSIWYG. WYSIWYG редакторы пользуется популярностью для задач создания графического пользовательского интерфейса в виде так называемых

GUI-конструкторов. Так как задачи создаваемого приложения схожи с задачами GUI-редакторов, принцип WYSIWYG является наиболее подходящим для реализации.

3. Язык Reflex и Reflex IDE

Процесс-ориентированный язык Reflex, также известный под именем "Си с процессами", предназначен для создания алгоритмов управления в области промышленной автоматизации: систем, предполагающих активное взаимодействие с внешней средой, технологическим оборудованием, физическими процессами через датчики и механизмы управления [2].

Для эффективной разработки алгоритмов управления на языке Reflex в ИАиЭ разрабатывается интегрированная среда разработки Reflex IDE.

За основу Reflex IDE взята платформа Eclipse Theia.

Eclipse Theia — это мощная и гибкая платформа для разработки облачных и настольных интегрированных сред разработки (IDE), написанная на языке TypeScript и исполняющаяся на платформе Node.js.

IDE на базе Eclipse Theia представляет собой web-приложение, состоящее из frontend и backend модулей, сообщающихся посредством протокола JSON-RPC. Разработчикам IDE предоставляется возможность расширять функциональность Theia (поддержка различных языков, инструментов), добавляя расширения или плагины.

Плагины для Theia представляют собой приложения, запускающиеся в отдельном процессе и взаимодействующие со средой разработки через ограниченный API. Плагины могут быть добавлены во время исполнения Theia и предназначены для создания легковесных инструментов разработки [3].

Расширения — это составные компоненты Theia. Любая функциональность, которая не может быть оформлена в виде плагина (в силу ограничения API) должна быть реализована в виде расширения. Модули,

реализующие базовую функциональность Theia, также являются расширениями [4].

API взаимодействия плагинов для Theia не позволяет создавать окна графического интерфейса, управляемые плагином. Поэтому реализация плагина не является подходящим решением поставленных задач. Расширения напротив позволяют гибко настраивать и управлять взаимодействием пользователя со средой разработки.

Reflex IDE расширяет базовую функциональность Theia поддержкой языка Reflex и принципом организации встраиваемых модулей — domain specific modules (DSM). У каждого DSM есть возможность получить текущее абстрактное синтаксическое дерево (AST) Reflex программы в формате XML документа. AST предоставляется специальным Theia расширением (AST Theia Extension) в backend части Reflex IDE. DSM могут быть реализованы в виде Theia расширения [5].

Таким образом, WYSIWYG редактор может быть интегрирован в Reflex IDE. Для этого требуется реализовать редактор в виде Theia расширения, а для доступа к данным о Reflex программе в редакторе должно быть реализовано отслеживание состояние AST Reflex программы.

4. Формулировка требований

На основании проведенного анализа предметной области и сформулированных требований к функциональности были сформулированы требования к реализации:

1. Web-приложение в виде DSM для Reflex IDE.
2. Обеспечение возможности редактирования и создания макетов графического представления ОУ.
3. Взаимодействие с пользователем по принципу WYSIWYG.
4. Входные данные AST, генерируемый ядром Reflex IDE.
5. Выходной формат XML.

6. Соответствие отображения LabVIEW-индикаторам.

ВЫБОР ТЕХНОЛОГИЙ РАЗРАБОТКИ

Исходя из требования реализации приложения как DSM для Reflex IDE, было принято решение при выборе технологий в первую очередь ориентироваться на легкую интеграцию с Reflex IDE и Eclipse Theia.

Theia написана на TypeScript и предоставляет возможность встраивать расширения, оформленные в виде модуля Node.js, реализованные на языках JavaScript и TypeScript.

Дополнительно Theia предоставляет базовые классы для встраивания компонентов пользовательского интерфейса, реализованных с помощью фреймворка React.

Более того фреймворк React на сегодняшний день является наиболее популярным веб-фреймворком с большой и современной базой знаний, что делает альтернативные варианты (Vue.js, Angular) менее привлекательными для использования [6].

TypeScript позволяет реализовывать логику взаимодействия клиента и сервера, логику изменения состояния приложения, но для полноценной работы web-приложения необходимо задавать внешний вид компонентов пользовательского интерфейса и его структуру. В web-приложениях для решения этих задач чаще всего используют HTML (задание разметки) и CSS (задание стилей). Фреймворк React предлагает расширение для языка TypeScript — TSX. TSX позволяет описывать разметку внутри TypeScript кода, что добавляет наглядности при работе с заданием разметки компонентов пользовательского интерфейса.

Таким образом, было принято решение использовать в качестве инструментов разработки:

1. Язык программирования TypeScript.
2. Фреймворк React.
3. Язык TSX для описания React компонентов.

4. Язык CSS для задания стилей.

ПРОЕКТИРОВАНИЕ

1. Общие принципы

В связи с тем, что редактор разрабатывается как web-приложение, система была разделена на клиентскую и серверную части.

Основной функцией серверной части является извлечение и анализ текущего состояния AST Reflex программы, а также отправка этих данных на клиентскую часть.

Клиентская часть отвечает за взаимодействие с пользователем — предоставление возможности задать графическое представление объекта управления и выгрузка выходного файла.

2. Архитектура Reflex IDE

Так как приложение встроено в Reflex IDE, стоит рассказать о том, что представляет из себя клиент-серверное взаимодействие и организация архитектурных компонентов в Reflex IDE (см. рис. 2).

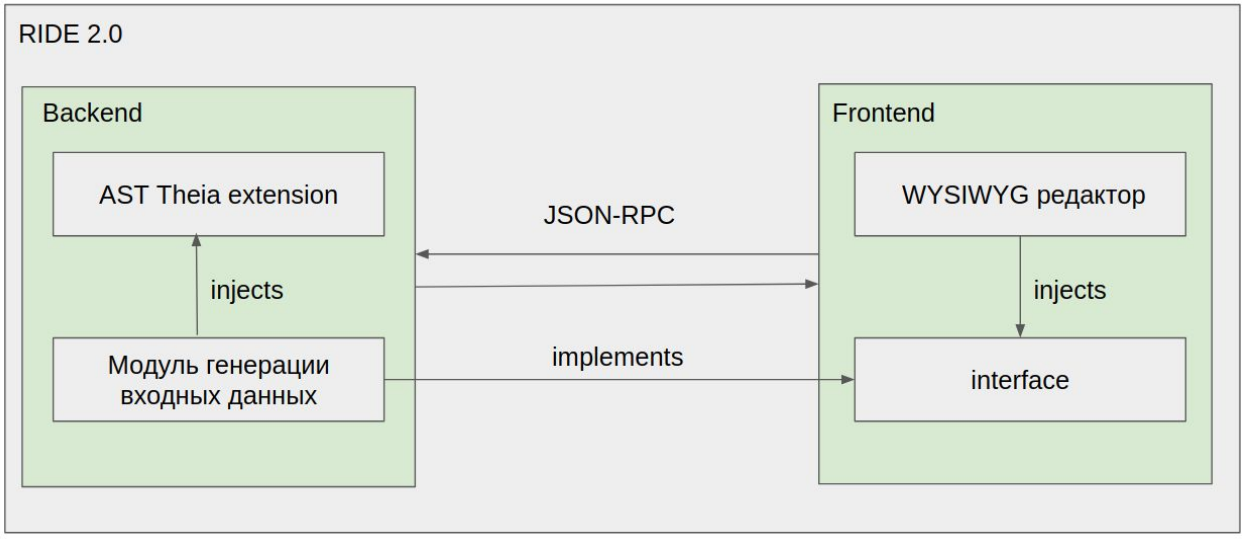

Рисунок 2. Архитектура Reflex IDE.

Reflex IDE как и любое современное web-приложение состоит из двух основных частей серверной и клиентской (Backend и Frontend). Взаимодействие этих частей происходит по протоколу JSON-RPC. JSON-RPC

это протокол, использующий HTTP в качестве инструмента отправки запросов. Передаваемые по протоколу данные — это записи, сериализованные в JSON и содержащие информацию о вызове какой-либо процедуры на сервере, реализующем API.

В серверной части IDE находится модуль, содержащий текущее AST Reflex программы. Другие серверные модули, в том числе написанные сторонними разработчиками, имеют возможность получить AST, вызвав соответствующий метод AST модуля. Также серверные модули должны реализовывать специальные интерфейсы, если предполагается, что обращение к модулю должно происходить с клиентской части приложения.

Клиентская часть состоит из модулей работы с пользователем, например, редактор кода, WYSIWYG редактор. Для взаимодействия с серверной частью модули вызывают методы соответствующих интерфейсов. Затем вызов преобразуется в формат, определённый протоколом JSON-RPC, и отправляется на серверную часть.

3. Архитектура клиентской части WYSIWYG редактора

Так как бóльшая часть написанного кода и трудозатрат при разработке WYSIWYG редактора приходится на клиентскую часть приложения, следует подробно рассмотреть выбор внутренней архитектуры клиентской части приложения.

Существует несколько устоявшихся принципов организации архитектуры для приложений с пользовательским интерфейсом. Рассмотрим некоторые из них.

3.1. Model-View-Controller

Model-View-Controller — это крайне популярный архитектурный паттерн, так как множество фреймворков использует его для организации структуры приложений [7].

Приложение, написанное с использованием данного паттерна, разделяется на три компонента (см. рис. 3).

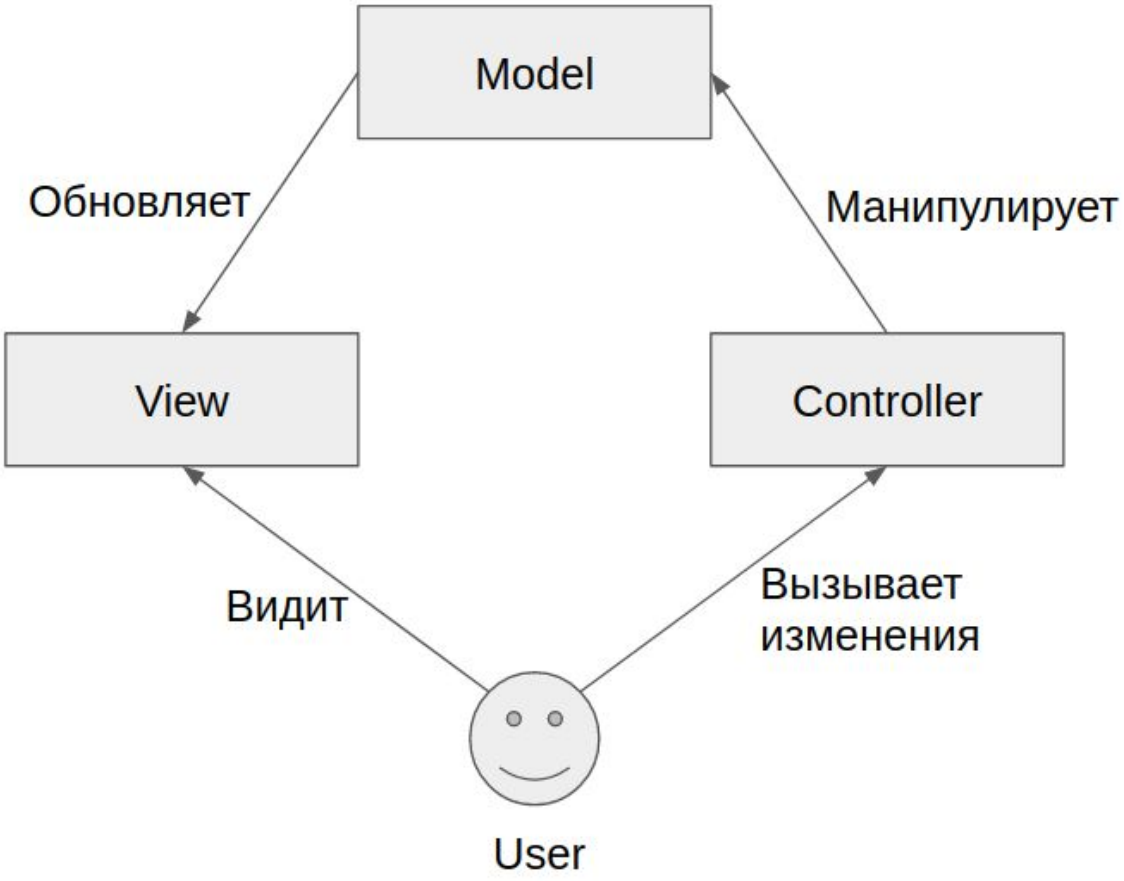

Рисунок 3. Схема архитектуры MVC

Рассмотрим каждый компонент:

Model — содержит логику работы с базами данных.

Controller — содержит бизнес логику.

View — содержит логику визуализации данных.

Данный принцип отлично работает для организации архитектуры всего web-приложения как совокупности клиента и сервера. Но при попытке адаптации данного паттерна для клиентской части web-приложения возникают проблемы [8].

Большинство JavaScript фреймворков заставляют разработчиков использовать механизм data-binding, который перекладывает ответственность

обработки пользовательских событий на View, при этом View начинает воздействовать на состояние Model напрямую. А для того, чтобы разделить ответственность между разными компонентами View, Model также разделяют на различные компоненты (см. рис. 4).

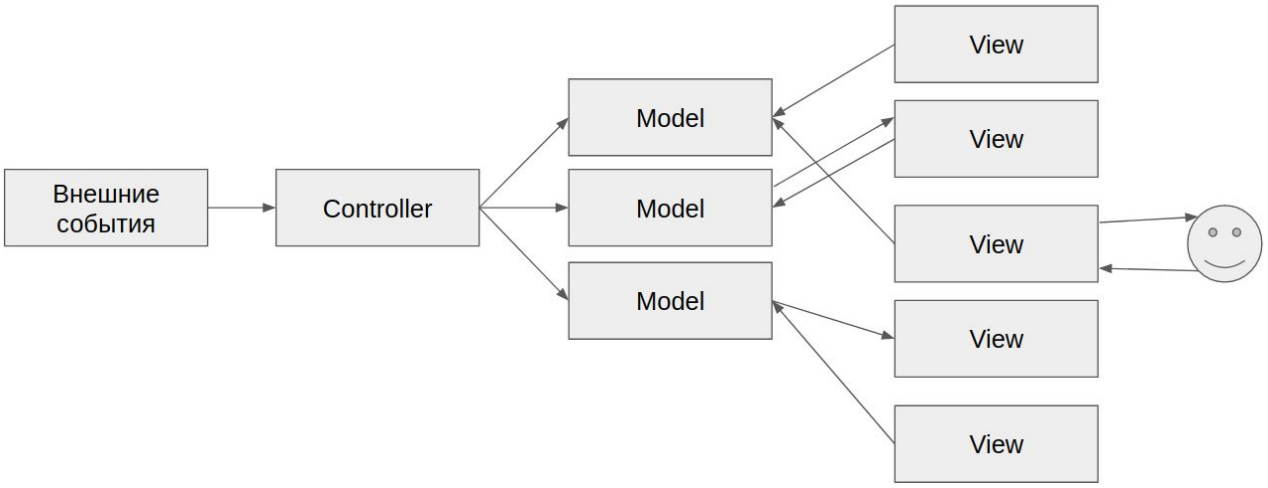

Рисунок 4. MVC в клиенте web-приложения

При такой схеме возникает проблема побочных эффектов и возрастает сложность отладки при разрастании приложения, так как одни и те же компоненты Model могут быть изменены по-разному из разных компонентов View.

3.2. Flux

Компания Facebook для создания клиентской части web-приложений предложила новый архитектурный подход — Flux.

“Flux — это архитектура приложений, которую Facebook использует для создания клиентских web-приложений. Она дополняет компоненты составного представления React, используя однонаправленный поток данных.”[9]

Приложение, разработанное с использованием архитектуры Flux (см. рис. 5), содержит следующие группы компонентов:

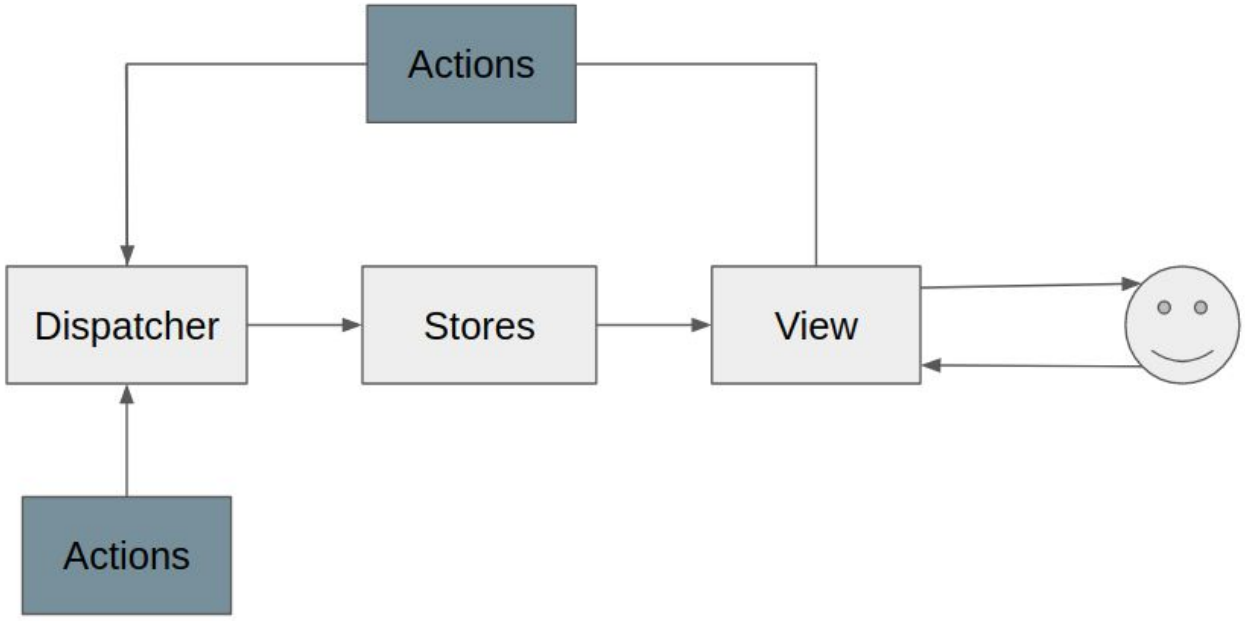

Рисунок 5. Flux архитектура.

Actions — простые объекты, содержащие поле `type`, а также поля, содержащие информацию о конкретном Action.

Stores — компоненты, содержащие данные о состоянии приложения и логику работы с ними. Каждый Store это отдельный объект, который ассоциирован с чистой функцией, которая на основе текущего состояния Store и произошедшего Action вычисляет новое состояние.

Dispatcher — это центральная шина, куда посылаются все сгенерированные Action. Dispatcher вызывает чистые функции из Stores, передавая им в качестве параметра текущий Store и полученный Action. В Dispatcher не содержится бизнес-логики, как в случае с Controller из MVC.

View — это компоненты отображения, которые мало чем отличаются от аналогичных в MVC.

Таким образом, Flux представляет собой модификацию MVC паттерна, адаптированную под особенности web-разработки.

3.3. Redux

Для фреймворка React существует несколько библиотек, позволяющих использовать готовые реализации Dispatcher. Но есть и библиотеки, которые не просто реализуют концепции Flux, но и модифицируют их. Наиболее популярная из таких библиотек — Redux [10].

Redux предлагает вместо группы Stores использовать один Store, который является “единственным источником правды” приложения. Кроме того, создатели Redux предлагают вынести из Store логику изменения, оставив только функцию хранения состояния.

Вынесенная логика организуется в новый компонент Reducers. Reducers не изменяют объекты текущего состояния, а возвращают его модифицированную копию, что является следствием иммутабельности Store.

Так как Store теперь всего один компонент, потребность в Dispatcher отпадает, и его больше не выделяют в отдельный модуль. Вместо обращения к Dispatcher, сущности вызывают глобальную функцию dispatch, которая запускает все Reducers, передавая в них сгенерированный Action.

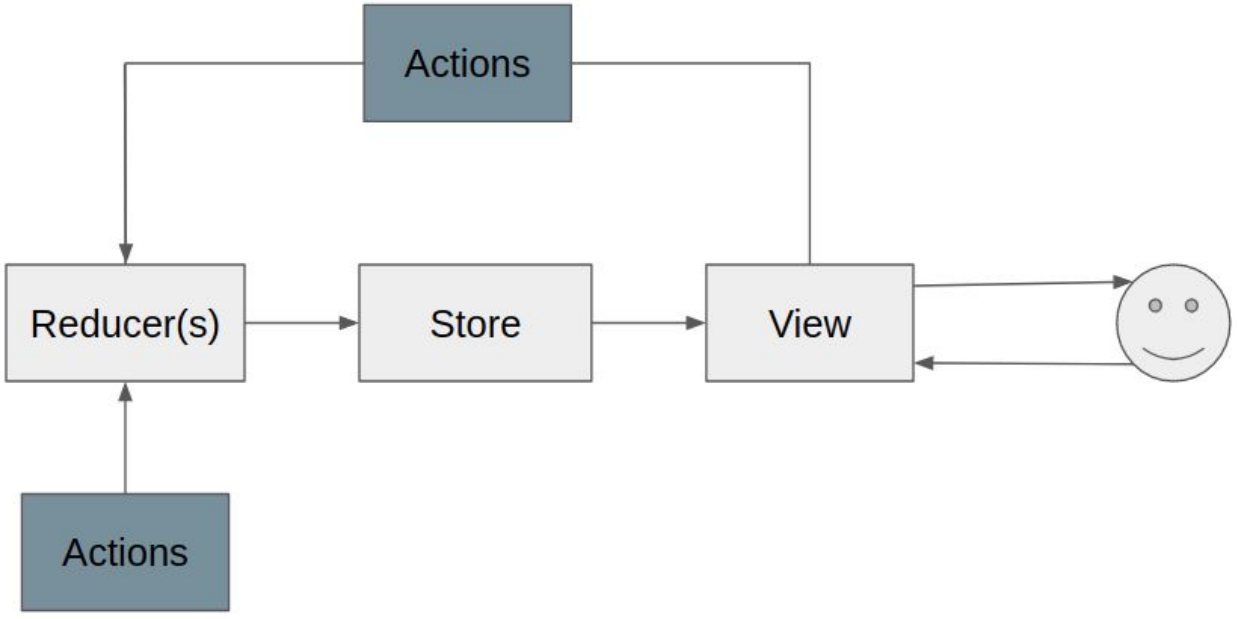

Рисунок 6. Redux архитектура.

Таким образом, было решено для реализации приложения выбрать архитектурный подход, предложенный библиотекой Redux. Поскольку он обеспечивает расширяемость приложения, учитывая особенности web-разработки. А также имеет реализацию для фреймворка React, обепечивающую сокращение количества кода не связанного с бизнес логикой.

РЕАЛИЗАЦИЯ

1. Элементы пользовательского интерфейса

Полное изображение пользовательского интерфейса присутствует в “приложении 1”.

В системе присутствует список всех доступных переменных, извлеченных из AST Reflex программы (см. рис. 7).

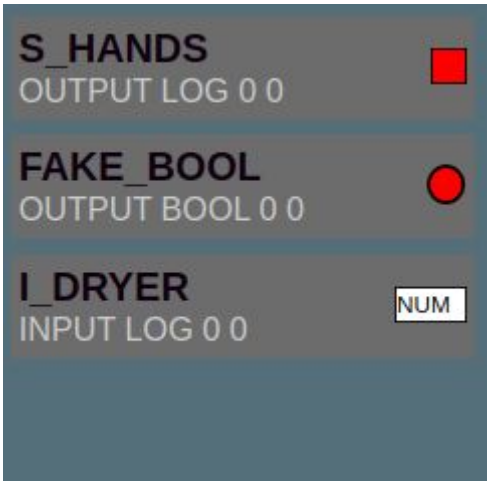

Рисунок 7. список переменных.

В нём отражена основная информация о каждой переменной. Для первого элемента списка имеем:

S_HANDS — имя переменной;

OUTPUT — тип порта;

LOG — тип переменной;

0 — номер байта в массиве порта;

0 — номер бита элемента.

Также каждой переменной ассоциировано её текущее графическое представление — индикатор. Для переменной `S_HANDS` выбранный индикатор имеет тип `SquareLED`.

По нажатию левой кнопкой мыши на индикатор переменной, появляется выпадающий список для замены текущего индикатора на один из представленных (см. рис. 8).

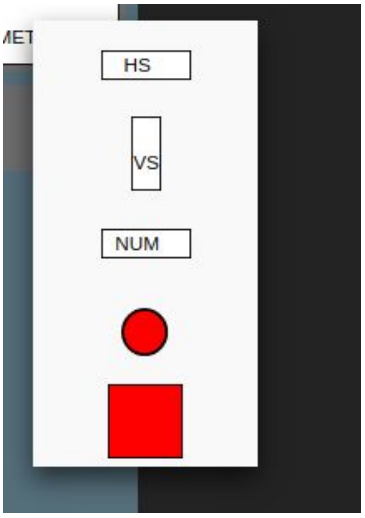

Рисунок 8. выпадающий список выбора индикатора.

Замена производится нажатием на соответствующий элемент списка. Доступны следующие типы индикаторов, соответствующие индикаторам LabVIEW: VerticalSlide, HorizontalSlide, Numeric, RoundLED, SquareLED.

В системе присутствуют кнопка выбора файла для загрузки изображения на макет объекта и кнопка, вызывающая генерацию выходного XML файла (см. рис. 9). По нажатию на кнопку “Select Image” браузер открывает стандартный для операционной системы пользователя механизм выбора изображения из файловой системы.

Рисунок 9. кнопки для работы с файлами.

В системе присутствует область макета панели объекта управления, где пользователь может комбинировать загруженное изображение и индикаторы. На рисунке 10 представлен макет панели объекта управления с загруженным изображением и установленными индикаторами (7), (8).

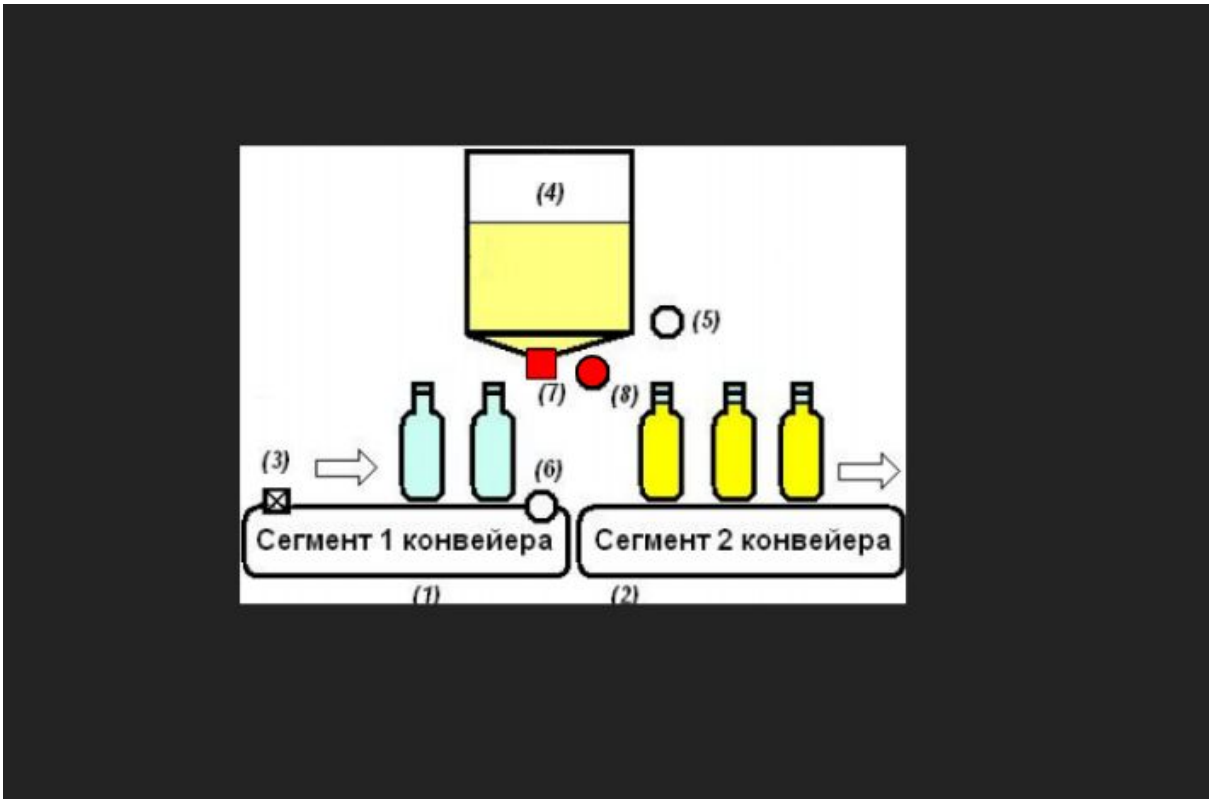

Рисунок 10. Макет в процессе компоновки.

В системе присутствует окно отображения и модификации свойств индикатора (см. рис. 11), на нём помимо свойств индикатора также присутствуют свойства привязанной к нему Reflex переменной.

Для модификации доступных числовых свойств необходимо нажать на иконку карандаша в соответствующей строке.

Для модификации цветовых свойств, необходимо нажать на соответствующую строку. Система откроет окно выбора цвета, соответствующее браузеру пользователя.

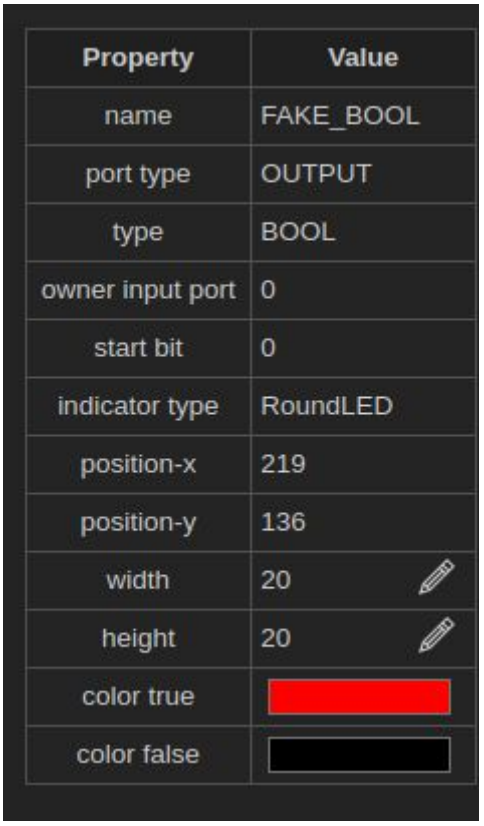

Рисунок 11. Окно свойств Reflex переменных.

2. Перемещение элементов

Механизм перемещения элементов (drag-and-drop). Для React существует несколько популярных библиотек для реализации drag-and-drop, но все из них поддерживают только перемещение элементов внутри или между списков. Для решение задачи требуется перемещать элементы между списком и областью, где элементы не имеют четкой структуры расположения в виде списка или сетки. Поэтому была предложена своя реализация механизма drag-and-drop.

Индикаторы отслеживают события зажимания на них правой кнопкой мыши, события движения с зажатой на элементе правой кнопки мыши, и события отпускания правой кнопки мыши с элемента. Каждое из перечисленных событий генерирует Action, который обрабатывается в соответствующем Reducer, обновляющим информацию о нахождении элемента.

Для области макет объекта управления и списка переменных в Store существуют соответствующие массивы, содержащие данные о расположенных на областях индикаторах.

Когда на перемещаемом элементе отпускают зажатую кнопку мыши, Reducer вычисляет для элемента область, в которой был закончен drag-and-drop, и перемещает данные соответствующего элемента в массив для вычисленной области. После чего View компоненты отображают новое состояние приложения.

3. Интеграция в Reflex IDE

Для интеграции с Reflex IDE приложение было оформлено в виде расширения для Theia.

Theia и расширения для Theia представляют собой Node.js модули. Каждый Node.js модуль представляет собой директорию, содержащую директории с исходными файлами, и файл `package.json`. В данном файле содержится справочная информация о модуле, а также список зависимостей проекта от других Node.js модулей.

Для интеграции в Reflex IDE в первую очередь необходимо добавить зависимость в `package.json` Reflex IDE (см. рис. 12).

```
{
    ...
    "dependencies": {
         ...
         wysiwyg-editor: "~1.0.0"
        ...
    },
    ...
}
```

Рисунок 12. Добавление зависимости в package.json Reflex IDE.

Theia реализует принцип внедрения зависимостей (DI [11]) для настройки объектов, используя библиотеку Inversify. На старте приложения

создается контейнер зависимостей, в который помещаются объекты всех классов, помеченных аннотацией `@injectable`.

В классы, содержащиеся в контейнере зависимостей, существует возможность “внедрить” другие классы из контейнера. Для этого необходимо создать в классе поле, имеющее тип внедряемого класса и помеченный аннотацией `@inject` (см. рис. 13).

```
@injectable()
export class WysiwygWidget extends ReactWidget {

    ...

    @inject(MessageService)
    protected readonly messageService!: MessageService;

    ...
}
```

Рисунок 13. Пример класса из контейнера зависимостей.

Данный подход был применен для создания виджета, отвечающего за взаимодействие с пользователем, и для backend-сервиса, собирающего данные о AST Reflex программы.

Theia расширение представляет собой набор классов, объекты которых добавлены в контейнер зависимостей. Объекты классов расширений имеют возможность внедрять и использовать любые объекты, содержащиеся в контейнере зависимостей.

Для реализации возможности отображения виджета редактора был зарегистрирован новый frontend модуль, путём добавления класса-наследника `AbstractViewContribution`, параметризованного классом-виджетом (см. рис. 14). Также в добавленном классе происходит регистрация команды и элемента меню управления среды разработки, открывающих виджет.

```
@injectable()
export class WysiwygContribution extends
AbstractViewContribution<WysiwygWidget> {

//конструктор, задающий основные
параметры для регистрации модуля
    constructor() {
        super({
            widgetId: WysiwygWidget.ID,
            widgetName: WysiwygWidget.LABEL,
            defaultWidgetOptions: { area: 'left' },
            toggleCommandId: WysiwygCommand.id
        });
    }

// регистрация команды в меню быстрого
доступа
    registerCommands(commands: CommandRegistry): void {
        commands.registerCommand(WysiwygCommand, {
            execute: () => super.openView({ activate: false,
                                            reveal: true })
        });
    }

// регистрация команды в меню (тулбаре)
    registerMenus(menus: MenuModelRegistry): void {
        super.registerMenus(menus);
    }
}
```

Рисунок 14. Регистрация виджета редактора.

Для связывания зарегистрированных команд с функциями, выполняющими открытие виджета, был создан экземпляр `ContainerModule` с переданной в него функцией, определяющей назначения созданных классов и связи (см. рис. 15).

```
export default new ContainerModule(bind => {
    bindViewContribution(bind, WysiwygContribution);
bind(FrontendApplicationContribution)
.toService(WysiwygContribution);
    bind(WysiwygWidget).toSelf();
    bind(WidgetFactory).toDynamicValue(ctx => ({
        id: WysiwygWidget.ID,
        createWidget: () =>
ctx.container.get<WysiwygWidget>(WysiwygWidget)
    })).inSingletonScope();
});
```

Рисунок 15. Связывание команд и обработчиков.

Помимо этого для регистрации модуля был была добавлена соответствующая конфигурация в package.json (см. рис. 16).

```
{
    ...
    "theiaExtensions": [
        {
            "frontend": "lib/browser/wysiwyg-frontend-module"
        }
    ]
    ...
}
```

Рисунок 16. Добавление конфигурации.

4. Работа с AST Reflex программы

Для получения актуального AST Reflex программы в Reflex IDE существует класс `AstService`. Объект данного класса добавлен в контейнер зависимостей, поэтому объекты классов расширения могут получить к нему доступ.

`AstService` содержит публичный метод `getAST`, возвращающий объект класса `string`, содержащий AST в формате XML.

Для работы редактора backend модуль выделяет из AST информацию о переменных объекта управления, содержащуюся в узлах `variables` внутри узла `processes` (см. рис. 17).

```
...
<processes name="P1">
      ...
      <variables xsi:type="reflex:PhysicalVariable"
                 type="int8"
                 name="hands_under_dryer">
            <mapping>
                  <port name="inp" addr1="0x01" addr2="0x02" size="8"/>
            </mapping>
      </variables>
      <variables xsi:type="reflex:PhysicalVariable"
                 type="bool"
                 name="dryer_control">
            <mapping bit="1">
                  <port type="output"
                        name="out"
                        addr1="0x03"
                        addr2="0x04"
                        size="16"/>
            </mapping>
      </variables>
      ...
</processes>
...
```

Рисунок 17. Фрагмент AST дерева, используемый редактором.

Каждый раз, когда виджет редактора получает фокус, виджет запрашивает список переменных, отправляя запрос на backend модуль. Backend модуль запрашивает AST из `AstService`, после чего преобразует его в список переменных в формате JSON (см. рис. 18) и возвращает его. Получив JSON, виджет обновляет свое состояние на основе нового списка переменных.

```
[
      ...
      {
            "purpose": "input";
            "type": "int8";
            "name": "hands_under_dryer";
            "ownerInputPort": "1";
            "startBit": "2";
      },
      {
            "purpose": "output";
            "type": "bool";
            "name": "dryer_control";
            "ownerInputPort": "3";
            "startBit": "4";
      }
      ...
]
```

Рисунок 18. Пример JSON документа — списка переменных.

ЗАКЛЮЧЕНИЕ

В процессе работы были выполнены следующие задачи:

1. Проанализирована специфика комплекса динамической верификации алгоритмов управления, создаваемых на языке Reflex, и Reflex IDE.
2. Сформулированы требования к разрабатываемому программному обеспечению.
3. Разработан пользовательский интерфейс WYSIWYG редактора.
4. Спроектирована архитектура приложения.
5. Определен формат описания графического представления ОУ.
6. Было реализовано приложение.
7. Приложение интегрировано в Reflex IDE.

Помимо этого работа была опубликована в рамках научной конференции МНСК-2020.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

Витченко В.А. _____

Подпись студента

«31» мая 2020 г

СПИСОК ЛИТЕРАТУРЫ

1. Использование виртуальных объектов для обучения программированию информационно-управляющих систем. В.Е. Зюбин к.т.н. 2009 г. https://www.iae.nsk.su/images/stories/6_DepPages/0_Labs/TG16-1/pub/09-zyubin-IT-REFLEXonFN.pdf
2. Язык Рефлекс. 2006-2011 г. http://reflex-language.narod.ru/
3. Authoring Theia Plug-ins. 2018 г. https://theia-ide.org/docs/authoring_plugins/
4. Authoring Theia Extensions. 2018 г. https://theia-ide.org/docs/authoring_extensions/
5. Использование Eclipse Theia для создания интегрированной среды разработки программ на процесс-ориентированном языке Reflex. К. В. Марченко. 2020 г.
6. Front-end frameworks popularity (React, Vue and Angular). Tanguy Krotoff. 2019 г. https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190
7. The DCI Architecture: A New Vision of Object-Oriented Programming. Trygve Reenskaug, James O. Coplien. 2009 г. https://web.archive.org/web/20090323032904/https://www.artima.com/articles/dci_vision.html
8. Why I No Longer Use MVC Frameworks. Jean-Jacques Dubray. 2016 г. https://www.infoq.com/articles/no-more-mvc-frameworks/
9. In-Depth Overview. Yangshun Tay. 2019 г. https://facebook.github.io/flux/docs/in-depth-overview/
10. Why Use React Redux. 2015 г. https://react-redux.js.org/introduction/why-use-react-redux#why-use-react-redux

11. Inversion of Control Containers and the Dependency Injection pattern. Martin Fowler. 2004 г. https://martinfowler.com/articles/injection.html

ПРИЛОЖЕНИЕ 1

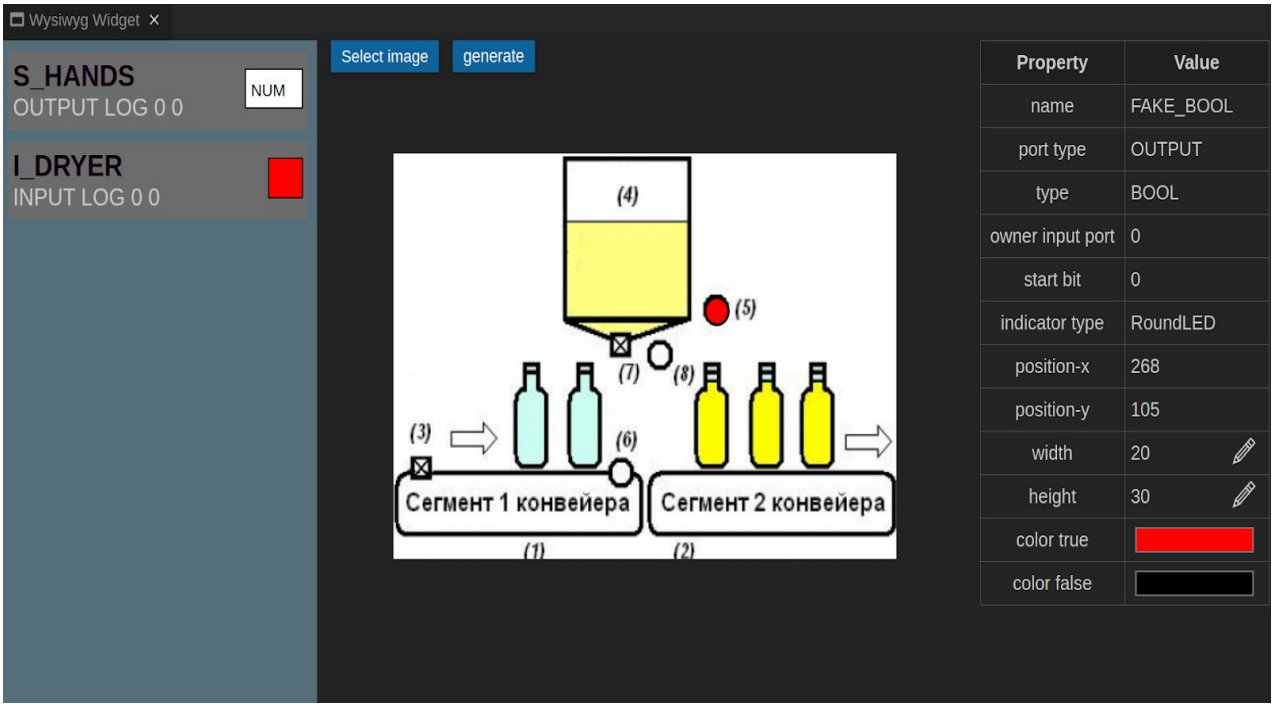

Рисунок 19. Интерфейс WYSIWYG редактора.

ПРИЛОЖЕНИЕ 2

WYSIWYG РЕДАКТОР ДЛЯ МОДУЛЯ ДИНАМИЧЕСКОЙ ВЕРИФИКАЦИИ ПРОЦЕСС-ОРИЕНТИРОВАННЫХ АЛГОРИТМОВ УПРАВЛЕНИЯ

РУКОВОДСТВО ОПЕРАТОРА

Новосибирск 2020

ОГЛАВЛЕНИЕ

АННОТАЦИЯ

В данном программном документе приведено руководство оператора по применению и эксплуатации WYSIWYG редактора для модуля динамической верификации процесс-ориентированных алгоритмов управления.

В разделе «Назначение программы» указаны сведения о назначении программы и ее функциональности.

Раздел «Условия выполнения программы» содержит условия, необходимые для выполнения программы.

Раздел «Выполнение программы» содержит последовательность действий оператора, которые необходимы для загрузки, запуска, выполнения и завершения программы.

НАЗНАЧЕНИЕ ПРОГРАММЫ

1. Функциональное назначение программы

Программное обеспечение предназначено для оптимизации процесса создания макета объекта управления модуля графического представления в комплексе динамической верификации процесс-ориентированных алгоритмов на языке Reflex.

2. Эксплуатационное назначение программы

Приложение предназначено для использования целях разработки программного обеспечения. Предполагается, что конечными пользователями программного обеспечения будут являться физические лица и коммерческие организации, занимающиеся разработкой алгоритмов промышленной автоматизации.

3. Состав функций

3.1. Вывод списка переменных объекта управления, извлеченных из кода Reflex программы.

3.2. Загрузка изображения объекта управления с компьютера пользователя.

3.3. Отображение загруженного пользователем изображения.

3.4. Отображение списка свойств Reflex переменных.

3.5. Задание типа индикаторов для Reflex переменных.

3.6. Изменение параметров индикаторов (размеры, цвета, граничные значения).

3.7. Размещение индикаторов на области посредством drag-and-drop.

3.8. Изменение положения индикаторов.

3.9. Выгрузка выходного XML файла.

УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

1. Минимальный состав аппаратных средств

Для использования программного обеспечения необходимо наличие персонального компьютера, включающего в себя:

1. Процессор на архитектуре x86 с частотой 2 ГГц или выше;
2. Оперативную память объемом 4 Гб или выше;
3. Графический адаптер;
4. Монитор, клавиатура, мышь;
5. Доступ в интернет.

2. Минимальный состав программных средств.

Для использования программного обеспечения необходима предустановка следующих программных средств:

1. Операционная система, поддерживающая графический пользовательский интерфейс, выход в интернет посредством web-браузера;
2. Web-браузер, поддерживающий отображение динамических html-страниц стандарта HTML5.

3. Требования к оператору

Для конечного оператора (пользователя) программного обеспечения предъявляются следующие требования:

1. Умение использовать графический пользовательский интерфейс операционной системы.
2. Умение использовать web-браузер.
3. Наличие зарегистрированной учётной записи Reflex IDE.

ВЫПОЛНЕНИЕ ПРОГРАММЫ

1. Загрузка и запуск программы

Для запуска программы необходимо запустить браузер и перейти на на страницу Reflex IDE. Актуальный адрес страницы может быть найден на сайте ИАиЭ.

После загрузки страницы необходимо пройти аутентификацию. Для этого необходимо ввести данные учётной записи Reflex IDE.

Далее необходимо вызвать контекстное меню View, после чего выбрать пункт Wysiwyg Widget.

2. Выполнение программы

2.1. Загрузка изображения

Необходимо нажать кнопку select image. После чего следовать инструкциям операционной системы.

2.2. Выбор индикатора для Reflex переменной

Необходимо нажать нажать правой кнопкой мыши на текущий индикатор переменной, после чего система отобразит возможные варианты в контекстном меню. Нажать на элемент контекстного меню, с выбранным индикатором.

2.3. Перемещение индикатора на область макета

Необходимо зажать левую кнопку мыши на индикаторе, после чего переместить курсор на область макета и отпустить левую клавишу мыши.

2.4. Просмотр свойств Reflex переменной

Нажать левую кнопку мыши на индикатор переменной, свойства будут отображены в виде списка в левой части основного окна.

2.5. Выгрузка выходного XML файла.

Необходимо нажать кнопку generate. После этого выходной файл будет добавлен в папку для загрузки файлов текущего браузера.

3. Завершение программы

Для закрытия программы оператору необходимо закрыть вкладку браузера, в котором открыто приложение.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа	Подпись	Дата
Номер изм.	измененных	замененных	новых	аннулированных					

Таблица 1. Лист регистрации изменений.

ПРИЛОЖЕНИЕ 3

WYSIWYG РЕДАКТОР ДЛЯ МОДУЛЯ ДИНАМИЧЕСКОЙ ВЕРИФИКАЦИИ ПРОЦЕСС-ОРИЕНТИРОВАННЫХ АЛГОРИТМОВ УПРАВЛЕНИЯ

ОПИСАНИЕ ПРОГРАММЫ

Новосибирск 2020

ОГЛАВЛЕНИЕ

АННОТАЦИЯ

В данном программном документе приведено руководство оператора по применению и эксплуатации WYSIWYG редактора для модуля динамической верификации процесс-ориентированных алгоритмов управления.

В разделе «Общие сведения» указаны сведения о программном обеспечении, необходимом для функционирования приложения и языке программирования.

В разделе «Назначение программы» указаны сведения о назначении программы и ее функциональности.

В разделе «Описание логической структуры» указаны сведения об алгоритме программы, используемых методах и структуре программы.

Раздел «Условия выполнения программы» содержит условия, необходимые для выполнения программы.

Раздел «Выполнение программы» содержит последовательность действий оператора, которые необходимы для загрузки, запуска, выполнения и завершения программы.

В разделах «Входные данные» и «Выходные данные» указаны форматы и особенности организации входных и выходных данных.

ОБЩИЕ СВЕДЕНИЯ

1. Обозначение и наименование программы

Наименование программы: WYSIWYG редактор для модуля динамической верификации процесс-ориентированных алгоритмов управления. Программа интегрирована в среду разработки Reflex IDE.

2. Программное обеспечение, необходимое для функционирования программы

- Программная платформа Node.js версии 10.8.
- Операционная система, поддерживающая упомянутую версию Node.js.
- Среда разработки Reflex IDE.

3. Языки программирования

4. я Исходным языком программного обеспечения редактора является TypeScript. Основным средством является интегрированная среда разработки IntelliJ Idea от компании Jetbrains. Для отладки использовались средства браузера — Google Chrome dev tools.

НАЗНАЧЕНИЕ ПРОГРАММЫ

4. Функциональное назначение программы

Программное обеспечение предназначено для оптимизации процесса создания макета объекта управления модуля графического представления в комплексе динамической верификации процесс-ориентированных алгоритмов на языке Reflex.

5. Эксплуатационное назначение программы

Приложение предназначено для использования целях разработки программного обеспечения. Предполагается, что конечными пользователями программного обеспечения будут являться физические лица и коммерческие организации, занимающиеся разработкой алгоритмов промышленной автоматизации.

6. Состав функций

6.1. Вывод списка переменных объекта управления, извлеченных из кода Reflex программы.

6.2. Загрузка изображения объекта управления с компьютера пользователя.

6.3. Отображение загруженного пользователем изображения.

6.4. Отображение списка свойств Reflex переменных.

6.5. Задание типа индикаторов для Reflex переменных.

6.6. Изменение параметров индикаторов (размеры, цвета, граничные значения).

6.7. Размещение индикаторов на области посредством drag-and-drop.

6.8. Изменение положения индикаторов.

6.9. Выгрузка выходного XML файла.

7. Минимальный состав аппаратных средств

Для использования программного обеспечения необходимо наличие персонального компьютера, включающего в себя:

6. Процессор на архитектуре x86 с частотой 2 ГГц или выше;
7. Оперативную память объемом 4 Гб или выше;
8. Графический адаптер;
9. Монитор, клавиатура, мышь;
10. Доступ в интернет.

ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

1. Алгоритм программы

В WYSIWYG редакторе происходит создание конфигурационного файла макета объекта управления для использования в модуле графического представления комплекса динамической верификации процесс-ориентированных объектов управления на языке Reflex.

На запуске программы и при каждом получении окном программы фокуса специальный модуль запрашивает данные о Reflex программы, после чего анализирует его, выделяя список переменных объекта управления. На основе данного списка формируется список доступных для размещения переменных-индикаторов, отображенных в пользовательском интерфейсе.

Также пользователь имеет возможность загрузить изображение на область макета для комбинирования его с индикаторами из списка.

Приложение написано с использованием архитектуры Redux, в связи с чем взаимодействие компонентов пользовательского интерфейса, модели программы и модуля получения списка переменных происходит следующим образом:

1. Пользователь взаимодействует с интерфейсом посредством нажатием и перемещением мышью, или окно пользовательского интерфейса получает фокус.
2. Компоненты, получившие внешний сигнал генерируют Action, содержащий информацию о произошедшем событии (нажата клавиша, получен фокус, элемент перемещен). И отравляют запрос о диспетчеризации сгенерированного Action.
3. Специальные чистые функции рассчитывают новое состояние приложения на основе сгенерированного Action и текущего состояния приложения. Текущее состояние приложения заменяется новым — сгенерированным.

4. Далее компоненты отображения оповещаются о изменении состояния и заново рендерят пользовательский интерфейс.

2. Структура программы

Код программы можно разделить на два компонента — backend часть и frontend часть.

Backend часть представляет собой компонент взаимодействия с ядром Reflex IDE. А именно с модулем AstService, который предоставляет AST Reflex программы в формате XML документа.

Frontend часть представляет логику взаимодействия пользователя с приложением и логику формирования выходных данных программы.

Взаимодействие частей происходит посредством протокола JSON-RPC. Логика взаимодействия инкапсулирована в Reflex IDE.

ВХОДНЫЕ ДАННЫЕ

Входными данными для WYSIWYG редактора является XML документ, содержащий данные о состоянии AST Reflex программы.

Для работы редактор извлекает из документа данные об узлах variables.

ВЫХОДНЫЕ ДАННЫЕ

Выходные данные для редактора представляют собой XML файл, содержащий информацию о переменных Reflex программы, а также их графических представлений (см. рис. 20).

```
<indicators>
    <indicator>
        <type>RoundLED</type>
        <portType>OUTPUT</portType>
        <portNum>0</portNum>
        <bitNum>0</bitNum>
        <positionX>268</positionX>
        <positionY>140</positionY>
        <width>50</width>
        <height>50</height>
        <colorTrue>FF0000</colorTrue>
        <colorFalse>000000</colorFalse>
    </indicator>
    <indicator>
        <type>Meter</type>
        <portType>OUTPUT</portType>
        <portNum>0</portNum>
        <bitNum>0</bitNum>
        <positionX>265</positionX>
        <positionY>56</positionY>
        <width>100</width>
        <height>50</height>
        <minRange>0</minRange>
        <difRange>100</difRange>
    </indicator>
    ...
</indicators>
```

Рисунок 20. Фрагмент выходного XML документа.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа	Подпись	Дата
Номер изм.	измененных	замененных	новых	аннулированных					

Таблица 2. Лист регистрации изменений.