МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ВИЗУАЛЬНОГО РЕДАКТОРА МНЕМОСХЕМ И СРЕДЫ ИСПОЛНЕНИЯ WEB-HMI ДЛЯ МОНИТОРИНГА ТЕХНОЛОГИЧЕСКОГО ОБОРУДОВАНИЯ В РЕАЛЬНОМ ВРЕМЕНИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Козицкого Вениамина Ильича

Научный руководитель
доцент, к. ф.-м. н. _____ Ю. Н. Кондратова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. Современные технологические производства, энергетические объекты и инженерные системы оснащаются большим количеством контрольно-измерительных устройств, которые непрерывно формируют значительные объёмы данных о состоянии и работе оборудования. Оперативное наблюдение за этими данными, диагностика отклонений и принятие управляющих решений возлагаются на диспетчерский персонал, рабочее место которого организуется средствами систем человеко-машинного интерфейса (HMI). Ключевым элементом такого интерфейса является мнемосхема, представляющая оператору состояние технологического объекта в визуальной форме.

HMI-системы чаще всего разрабатывались как настольные приложения, привязанные к конкретному оборудованию и операционной системе. Сейчас отрасль постепенно переходит к веб-решениям, в которых браузер становится основной средой работы оператора. Преимущества веб-ориентированного подхода и связанные с ним инженерные ограничения подробно рассматриваются в первой главе работы.

Отдельной составляющей задачи разработки Web-HMI является создание инструментов конфигурирования. Конечный пользователь системы не должен писать сложный программный код для построения каждой новой схемы, вместо этого ему должен предоставляться визуальный редактор, в котором мнемосхема собирается из графических компонентов, а связи отдельных элементов с источниками данных задаются декларативно.

Актуальность выбранной темы определяется потребностью в создании единой браузерной среды, в которой инженер может разрабатывать мнемосхемы, настраивать графические элементы и передавать готовые решения в исполнение. Разрабатываемое программное обеспечение ориентировано на мониторинг технологического оборудования в реальном времени и связывает визуальное представление объекта с поступающими значениями параметров.

Целью работы является разработка Web-HMI, включающего визуальный редактор мнемосхем с библиотекой переиспользуемых компонентов и среду исполнения, обеспечивающую отображение значений параметров технологического оборудования в режиме реального времени.

Для достижения поставленной цели необходимо решить следующие задачи:

— проанализировать назначение HMI/SCADA-систем и роль мнемосхем в мониторинге технологических процессов;
— спроектировать архитектуру веб-приложения, включающего библиотеку элементов, редактор графических элементов, редактор мнемосхем и среду исполнения;
— разработать модель данных для хранения библиотеки элементов, мнемосхем и привязок к внешним параметрам;
— реализовать визуальный редактор, позволяющий создавать и настраивать графические элементы и мнемосхемы;
— реализовать механизм загрузки, сохранения и исполнения мнемосхемы;
— реализовать обмен данными в реальном времени между клиентской частью и сервером исполнения;
— проверить работоспособность разработанного приложения на тестовой мнемосхеме.

Методологические основы разработки человеко-машинных интерфейсов и систем диспетчерского управления технологическими процессами представлены в работах В. В. Кангина, Д. В. Коннова, А. Г. Полетыкина, С. И. Макаренко, А. П. Веревкина, А. И. Боровкова, а также зарубежных исследователей Д. Бейли, К. Л. С. Шармы, У. Манке, К. Х. Джона и К. Уэйра.

Практическая значимость работы заключается в создании единой браузерной среды разработки и исполнения мнемосхем, ориентированной на мониторинг технологического оборудования в реальном времени. Разработанный программный комплекс принят к внедрению на производственном предприятии-разработчике, что подтверждается соответствующим актом, и пригоден к дальнейшему применению как в составе внутренних диспетчерских систем предприятия, так и в качестве самостоятельного программного продукта.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и трёх приложений. Общий объём работы — 62 страницы, из них около 40 страниц — основное содержание, включающее 6 рисунков; цифровой носитель с исходным кодом приведён в качестве приложения. Список использованных источников информации содержит 21 наименование.

Краткое содержание работы

Первый раздел «Анализ предметной области и обоснование технических решений» посвящён рассмотрению назначения и классификации HMI-систем, роли мнемосхемы как центрального элемента диспетчерского интерфейса, а также особенностям веб-ориентированной реализации HMI. Под HMI понимается совокупность программных и аппаратных средств, посредством которых оператор технологической установки наблюдает за её текущим состоянием и оказывает на неё управляющие воздействия. Такое определение помещает HMI в иерархию автоматизированной системы управления технологическим процессом на промежуточный уровень между контроллерным и диспетчерским. Назначение HMI определяется характером решаемых оператором задач: непрерывный мониторинг параметров процесса, своевременное выявление отклонений и аварийных ситуаций, выдача управляющих команд и регистрация действий оператора.

По способу размещения вычислительных ресурсов выделяются встроенные (панельные) и программные HMI, а среди программных — настольные, мобильные и веб-ориентированные решения. Перенос интерфейса в браузер снимает зависимость от платформы, упрощает развёртывание и обновление и обеспечивает удалённый доступ к мнемосхеме, однако накладывает ряд технических ограничений: изолированность браузерной среды, необходимость сетевого взаимодействия с источниками данных, требования к защищённому транспорту и отказоустойчивости соединения. Отдельное внимание уделено эргономическим аспектам проектирования мнемосхемы и нормативной базе — ГОСТ Р МЭК 62682 и ГОСТ Р МЭК 60073, — закрепляющей требования минимизации когнитивной нагрузки на оператора.

В разделе выполнен сравнительный анализ существующих решений. Коммерческие платформы — Siemens WinCC Unified, Ignition Perspective, AVEVA InTouch, Rockwell FactoryTalk Optix — закрывают типовые задачи диспетчеризации, но требуют значительных лицензионных отчислений, привязывают пользователя к одному производителю и хранят проектную информацию в закрытых форматах. Открытые решения — Rapid SCADA, OpenSCADA, ScadaLTS — снимают ценовое ограничение, но уступают коммерческим в качестве визуального редактора и полноте веб-клиента. Рассмотрены три технологии графического представления мнемосхем в браузере. Декларативная модель SVG удобна, но

влечёт высокую стоимость операций над деревом DOM при большом числе изменяющихся элементов. Императивный контекст Canvas 2D без затруднений работает с тысячами объектов, не обращаясь к DOM. Аппаратно-ускоренный WebGL ориентирован на трёхмерную визуализацию и для двумерных диспетчерских интерфейсов избыточен. Обоснован выбор Canvas 2D как разумного компромисса между производительностью и сложностью реализации, а в качестве библиотек — Konva для рендеринга и обработки событий и Paper.js для булевых операций над контурами. По итогам анализа установлены характерные пробелы существующих решений и определена функциональная ниша разрабатываемого комплекса.

Второй раздел «Архитектура и программная реализация» посвящён реализации программного комплекса и составляет основное содержание работы. В нём описаны самостоятельно спроектированная архитектура, доменная модель данных, декларативный язык описания примитивов, подсистема отрисовки с кэшированием, механизм программного формирования формы, протокол обмена с сервером и реализация редакторов.

Программный комплекс реализован как одностраничное браузерное приложение с серверной частью. В архитектуре выделены две пользовательские подсистемы: проектирования, объединяющая три специализированных редактора — библиотеки переиспользуемых компонентов, отдельного библиотечного элемента и мнемосхемы, — и исполнения, обеспечивающая загрузку проектных артефактов, поддержание соединения с сервером значений и применение поступающих значений к свойствам экземпляров мнемосхемы. Серверная сторона предоставляет два логических интерфейса: HTTP API для загрузки и сохранения проектных артефактов и WebSocket-канал для передачи значений параметров в реальном времени; в рамках работы серверная часть реализована как программный эмулятор источника данных.

В основе комплекса лежит самостоятельно спроектированная трёхуровневая доменная модель «библиотека — мнемосхема — рантайм». Первый уровень описывает переиспользуемые графические компоненты и их типизированные свойства; особое место занимает тип BINDVARIABLE, обеспечивающий подстановку текущих значений свойств непосредственно перед отрисовкой. Второй уровень описывает конкретную диспетчерскую сцену — экземпляры элементов, их геометрические преобразования, статические переопределения и динамиче-

ские привязки свойств, а также источники данных и внешние параметры. Третий уровень — среда исполнения — существует исключительно в памяти и сопоставляет адресам внешних параметров их текущие значения. Принципиальной особенностью модели является послойная организация: каждый уровень адресует сущности предыдущего по идентификаторам, не дублируя их содержание, что обеспечивает компактность сериализованного представления и непротиворечивость при изменении общих компонентов.

Разработан декларативный язык описания примитивов: графическое представление элемента задаётся структурированным набором геометрических примитивов и булевых операций над ними. Базовым строительным блоком является примитив, описываемый объектом с собственным идентификатором, углом поворота и параметрами отрисовки — цветом, толщиной контура, цветом заливки и прочими визуальными характеристиками. Реализовано девятнадцать видов примитивов, объединённых в шесть групп: линии, четырёхугольники, треугольники, эллипсы, многоугольники и составные элементы (шкала, текстовая надпись, SVG-изображение, таблица). Корневой структурой выступает запрос на построение составной формы, содержащий список исходных примитивов и список булевых операций над ними; поддерживаются пять типов операций — объединение, пересечение, разность, симметрическая разность и операция отсутствия комбинирования, — результат которых может выступать операндом последующих операций, что позволяет выстраивать иерархию композиций произвольной глубины. Параметрическая настройка выполняется через механизм связных переменных: вместо непосредственного значения параметра указывается ссылка на свойство элемента, выраженная строкой с фиксированным префиксом. Процедура разрешения непосредственно перед отрисовкой проходит по всем полям описания формы и подставляет на место таких строк актуальные значения свойств; она реализована как чистая функция, возвращающая новое описание при наличии хотя бы одной подстановки и исходное в противном случае, что обеспечивает иммутабельность исходного описания и сокращает излишние перерисовки.

Самостоятельно спроектирована и реализована подсистема отрисовки на основе библиотек Konva и Paper.js, включающая классы Painter, PainterCache и ScriptExecutor. Класс Painter инкапсулирует работу с группой графических узлов библиотеки Konva и реализует основной алгоритм построения сцены по

описанию формы. Цикл отрисовки одного кадра построен по схеме «подготовка — формирование сцены — завершение»: метод начала кадра переустанавливает счётчик использованных идентификаторов, а метод завершения удаляет из кэша и со сцены узлы, не помеченные как использованные в текущем кадре. Подобный приём управления ресурсами, называемый mark-and-sweep, обеспечивает корректное удаление примитивов, исчезнувших из описания между кадрами, без явного отслеживания списка изменений. Для удержания приемлемого быстродействия при сотнях одновременно отображаемых параметров реализован двухуровневый кэш геометрии и графических узлов. Решение о применимости кэша принимается процедурой двухступенчатого сравнения: на первом шаге выполняется сравнение по ссылке на объект описания примитива, и если описание не изменилось со времени предыдущего кадра, кэш применим без дополнительных проверок; в противном случае вычисляются хэши геометрических и визуальных параметров и сравниваются с сохранёнными в записи кэша значениями. Разделение геометрического и визуального хэшей позволяет повторно использовать построенную геометрию при изменении лишь визуального оформления и наоборот. Хэширование выполняется через детерминированную сериализацию структуры в строку и вычисление 53-битного хэша алгоритмом cyrb53, а нормализация цветовых значений перед хешированием обеспечивает совпадение записей при семантически эквивалентных представлениях цвета.

Класс ScriptExecutor расширяет возможности декларативного описания формы за счёт допустимости программной генерации описания на
языке JavaScript для случаев, когда геометрия элемента или его внешний вид определяются нетривиальной вычислительной логикой — например, при построении шкалы измерительного прибора с переменным числом делений или диаграммы по табличным данным. Класс реализует два пути формирования описания. Текстовый путь компилирует строковое представление скрипта в исполняемую функцию и вызывает её в подготовленном контексте, формируемом из ссылки на экземпляр Painter, словаря свойств элемента, размеров холста и набора переменных свойств, обеспечивающих доступ к значениям непосредственно по их идентификаторам; все элементы контекста передаются скомпилированной функции как именованные аргументы, что исключает неявные глобальные зависимости. Прямой путь применяется, когда описание получено декларативным способом, и сводится к разрешению связных переменных и непосредственной

передаче готового описания в подсистему отрисовки. Защитные ветви на случай некорректного исходного представления значения обеспечивают устойчивость подсистемы к ошибкам в описании библиотеки и предотвращают аварийное завершение исполнения скрипта.

Разработан протокол обмена с сервером значений на основе WebSocket с тремя типами сообщений: snapshot — полное состояние при подключении, update — частичные обновления, write — запрос на запись со стороны клиента. Применение начального снапшота приводит к полной замене внутреннего отображения значений, а применение обновления — к наложению дельты с проверкой каждого значения на действительное изменение, что сокращает излишние перерисовки. Клиент протокола обеспечивает идемпотентные операции установления и разрыва соединения, автоматическое переподключение при разрыве и защиту от устаревших событий, возникших на ранее закрытом сокете, посредством сравнения сохранённой в замыкании ссылки с актуальной ссылкой объекта.

Все три редактора подсистемы проектирования реализованы по единому архитектурному шаблону с явным разделением состояния документа и состояния пользовательского интерфейса в самостоятельные функции-редьюсеры, что позволяет выполнять переключение выделения без повторной валидации мнемосхемы. Редактор библиотечных элементов организован в виде трёхпанельного интерфейса: список примитивов, холст с интерактивным выделением и инспектор свойств на основе типизированных элементов управления, каждый из которых инкапсулирует валидацию и форматирование значений своего типа; булевы операции над контурами выполняются через отдельное диалоговое окно. Редактор мнемосхем повторяет трёхпанельную схему, но оперирует деревом подключённой библиотеки, экземплярами элементов на холсте и инспектором экземпляра или мнемосхемы в целом; размещение экземпляров реализовано механизмом drag-and-drop. Согласование поступающих от сервера значений со свойствами экземпляров выполняется в два этапа: преобразование значений по адресам в значения по идентификаторам внешних параметров и последующий сбор действующих значений свойств, используемых как словарь подстановок при разрешении связных переменных. Реализована семантическая валидация мнемосхемы, выявляющая привязки к несуществующим параметрам, ссылки на необъявленные источники данных, дубликаты идентификаторов и противо-

речия правилу взаимного исключения переопределения и привязки; отчёт о валидации предотвращает сохранение заведомо повреждённой мнемосхемы.

По результатам раздела разработан работоспособный программный комплекс, реализующий все поставленные задачи и проверенный на тестовой мнемосхеме.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был разработан программный комплекс Web-HMI, объединяющий визуальный редактор мнемосхем с библиотекой переиспользуемых графических компонентов и среду исполнения, обеспечивающую отображение значений параметров технологического оборудования в режиме реального времени.

Выполнен аналитический обзор предметной области и нормативной базы построения мнемосхем, а также сравнительный анализ ведущих коммерческих и открытых Web-HMI и SCADA-платформ, по результатам которого установлены характерные пробелы существующих решений и определена функциональная ниша разрабатываемого комплекса. Спроектирована трёхуровневая доменная модель «библиотека — мнемосхема — рантайм», обеспечивающая разделение между описанием графических компонентов, описанием конкретной диспетчерской сцены и динамическими значениями параметров и сохраняющая компактность сериализованного представления за счёт адресации сущностей по идентификаторам. Разработан декларативный язык описания примитивов с поддержкой булевых операций над контурами и механизмом параметрической настройки через связные переменные.

Реализованы три специализированных редактора подсистемы проектирования, построенные по единому архитектурному шаблону. Разработана клиентская часть среды исполнения, обеспечивающая поддержание соединения с сервером значений, приём начального снапшота и дельта-обновлений и применение поступающих значений к свойствам экземпляров мнемосхемы. Реализована подсистема отрисовки на основе библиотек Konva и Paper.js с двухуровневым кэшем геометрии и графических узлов.

Разработанный программный комплекс принят к внедрению на производственном предприятии-разработчике, что подтверждается соответствующим актом, и пригоден к дальнейшему применению как в составе внутренних диспетчерских систем предприятия, так и в качестве самостоятельного программного продукта. Перспективы дальнейшего развития связаны с введением механизмов аутентификации и разграничения полномочий, интеграцией с промышленными протоколами OPC-UA и MQTT и поддержкой управляющих воздействий.

Основные источники информации

1. Кангин, В. В. Программные аспекты проектирования SCADA-систем / В. В. Кангин, М. В. Кангин, Д. Н. Ямолдинов. — Нижний Новгород: Нижегородский государственный технический университет, 2010.
2. Полетыкин, А. Г. Разработка человеко-машинных интерфейсов для операторов промышленных систем управления на примере объектов тепловой энергетики / А. Г. Полетыкин [и др.]. — Москва: ИПУ РАН, 2023.
3. Макаренко, С. И. Интероперабельность человеко-машинных интерфейсов / С. И. Макаренко. — СПб.: Наукоемкие технологии, 2023.
4. Веревкин, А. П. Модели в задачах разработки автоматизированных систем управления технологическими объектами / А. П. Веревкин, Т. М. Муртазин. — Москва; Вологда: Инфра-Инженерия, 2025.
5. Bailey, D. Practical SCADA for Industry / D. Bailey, E. Wright. — Oxford: Newnes, 2003.
6. Sharma, K. L. S. Overview of Industrial Process Automation / K. L. S. Sharma. — Oxford: Elsevier, 2011.
7. Mahnke, W. OPC Unified Architecture / W. Mahnke, S. H. Leitner, M. Damm. — Berlin; Heidelberg: Springer, 2009.
8. Ware, C. Information Visualization: Perception for Design / C. Ware. — 4th ed. — Cambridge, MA: Morgan Kaufmann, 2020.