

РЕФЕРАТ

Выпускная квалификационная работа 142 страниц, 39 рисунков, 46 таблиц, 36 источников, 5 приложений.

Ключевые слова: производство, мнемосхема, векторная графика, редактор, библиотека.

Объектом исследования являются методологии проектирования векторных графических редакторов, а так же построение модульных систем.

Цель работы — разработать программные средства для работы с технологическими мнемосхемами с возможностью встраивания их в другие системы.

В процессе исследования использовались методы сбора и анализа информации, методы создания технических документов, методы проектирования и разработки программных средств.

В результате исследования был а создана карта бизнес-процессов работы с мнемосхемами на предприятиях. Данная карта позволила выделить ключевые бизнес-процессы, их взаимодействие друг с другом, а так же определить потоки данных в них. Выделение ключевых бизнес-процессов и их взаимодействие позволило формализовать требования для программного обеспечения для работы с мнемосхемами.

Основные конструктивные, технологические и технико-эксплуатационные характеристики: редактор позволяет создавать и редактировать мнемосхемы. Обозреватель позволяет выполнить интеграцию его в стороннюю систему через провайдер, и выполнять демонстрацию мнемосхем. Указанные выше средства позволяют выполнять анимацию мнемосхем в зависимости от данных источника.

Степень внедрения: разработанный набор графических средств MnemoEditorPro внедрены на предприятии ООО «СибХайТекЦентр», а так же включены в состав MES-системы «APM технолога» предприятия ООО «СибМетаХим».

Область применения: редактор используется на предприятиях для повышения эффективность создания и использования мнемосхем, необходимых для оперативного отображения состояния функционирования систем добычи и транспортировки нефти и газа.

В будущем планируется доработка набора средств для распространения в качестве коробочной версии программного продукта.

ОГЛАВЛЕНИЕ

Введение	4
1 Анализ предметной области	5
1.1 Графические средства для контроля процессов производства	5
1.2 Общий процесс создания мнемосхем и их использования на производстве	9
1.2.1 Процесс создания мнемосхем	9
1.2.2 Процесс использования мнемосхем	11
1.2.3 Внутренняя структура мнемосхем	11
1.3 Обзор средств для работы с мнемосхемами на рынке	13
1.3.1 Критерии обзора средств работы с мнемосхемами	13
1.3.2 Оценка графических средств	13
2 Разработка графических средств	17
2.1 Командная работа, методы и инструменты проектирования, разработки программного обеспечения	17
2.1.1 Команда	17
2.1.2 Методологии разработки программного обеспечения в команде	18
2.1.3 Инструменты разработки программного обеспечения	19
2.2 Используемые методы и инструменты	21
2.3 Векторная графическая библиотека	22
2.3.1 Редактор и обозреватель мнемосхем: общее и различия	22
2.3.2 Проектирование векторной графической компоненты	23
2.3.3 Объектная модель векторной графической библиотеки	30
2.4 Обозреватель мнемосхем	32
2.4.1 Проектные решения для разработки обозревателя мнемосхем	32
2.4.2 Программная реализация обозревателя мнемосхем	37
2.5 Редактор мнемосхем	39
2.5.1 Проектные решения для разработки редактора мнемосхем	39
2.5.2 Программная реализация редактора мнемосхем	51
3 Задачи интеграции графических средств с внешними системами	58
4 Финансовый менеджмент, ресурсоэффективность и ресурсосбережение	Ошибка! За
5 Социальная ответственность	Ошибка! Залка не определена.
Заключение	60
Список публикаций	61
Список используемых источников	62

Приложение А.....	66
Приложение Б.....	79
Приложение В.....	80
Приложение Г.....	81
Приложение Д.....	82

ВВЕДЕНИЕ

В большинстве предприятий для моделирования и анализа бизнес-процессов используют графические схемы. Графическими схемами может быть представлен оборот средств компаний, производственные цеха предприятия, или например, процесс добычи нефти.

Разработка выполнялась для компании ООО «СибХайТекЦентр», которая занимается разработкой программного обеспечения для предприятий нефтегазовой и химической отрасли. Такими предприятиями являются ОАО «Востокгазпром», ООО «Сибметакхим», ООО «Газпром трансгаз Томск». Для моделирования процессов производства в разрабатываемом программном обеспечении компании необходимо применение мнемосхем [3]. Мнемосхема представляет собой графическую модель объекта управления, динамически отображающая его функционирование [1]. Такими схемами изображают технологические процессы, энергетические системы, цеха станков с числовым программным управлением и т.п.

В производственном процессе мнемосхема имеет свой жизненный цикл: создание, эксплуатация и вывод из эксплуатации. Создание и модификация мнемосхем осуществляется с помощью специализированных редакторов. Эксплуатация и вывод из эксплуатации мнемосхем осуществляется, как правило, в информационных системах управления производством [3].

В ООО «СибХайТекЦентр» для создания и модификации мнемосхемам используется лицензионный векторный графический редактор Pro Grapher, разработанный на основе ActiveX компонента ProGrapherControl. Компонента ProGrapherControl является свободно распространяемым продуктом и не имеет лицензионных ограничений на применение (на встраивание в другие системы). При этом сам редактор Pro Grapher имеет лицензионное ограничение на количество используемых копий (в ООО «СибХайТекЦентр» одна копия). Таким образом, хотя данное технологическое решение позволяет выполнять задачи создания и использования мнемосхем, существуют лицензионные ограничения использования имеющихся инструментальных средств уже на этапе создания мнемосхем. Это в значительной степени замедляет процесс создания мнемосхем для крупного производства. Кроме того, существующее решение имеет недостатки в методике изготовления и использования мнемосхем [1].

Учитывая недостатки существующего технологического решения, а также требования, предъявляемые к нему (см. приложение Д), была поставлена цель работы: разработать графические программные средства для создания и использования мнемосхем. Одним из основным требований предприятия к разработке графических программных средств является платформа разработки Microsoft .Net и фреймворк WPF.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Графические средства для контроля процессов производства

В большинстве промышленных предприятий для управления процессами производства и их мониторинга создаются АСУ ТП, использующие специализированные средства, обладающих специфическим графическим интерфейсом. Спецификой графического интерфейса таких средств является представление информации о производственных процессах производства в виде мнемосхем.

Мнемосхема — это графическая модель объекта управления, динамически отображающая его функционирование (рис. 1) [1]. Это может быть технологический процесс, энергетическая система, цех станков с числовым программным управлением и т.п. Иначе говоря, мнемосхема – это условная информационная модель производственного процесса или системы, выполненная как комплекс символов, изображающих элементы системы (или процесс) с их взаимными связями.

Мнемосхемы эффективно используют в случаях, когда:

- управляемый объект имеет сложную технологическую схему и большое число контролируемых параметров;
- отображаемая информация технологических схем объекта может оперативно изменяться в процессе работы.

Мнемосхемы могут отражать как общую картину состояния системы, технологических процессов, так и состояние отдельных агрегатов, устройств, значения параметров и т.п. Мнемосхемы помогают оператору, работающему в условиях большого количества поступающей информации, облегчить процесс информационного поиска, подчинив его определенной логике, диктуемой реальными связями параметров контролируемого объекта. Они облегчают оператору логическую систематизацию и обработку поступающей информации,

помогают осуществлению технической диагностики при отклонениях процесса от нормы, обеспечивают внешнюю опору для выработки оптимальных решений и формирования управляющих воздействий.

Мнемосхемы по своему представлению могут быть выполнены в различном виде. Это может быть стенд с графическими схемами и лампами, или графическое представление в компьютерной программе. В данной работе рассматриваются мнемосхемы как графическое представление в программной системе. При работе с мнемосхемами необходимо опираться на принципы векторной графики. Одним из главных принципов векторной графики является факт, что такой способ представления объектов и изображений в компьютерной графике основан на математическом описании элементарных геометрических объектов, называемых примитивами: точки, полилинии, полигоны, прямоугольники, дуги.

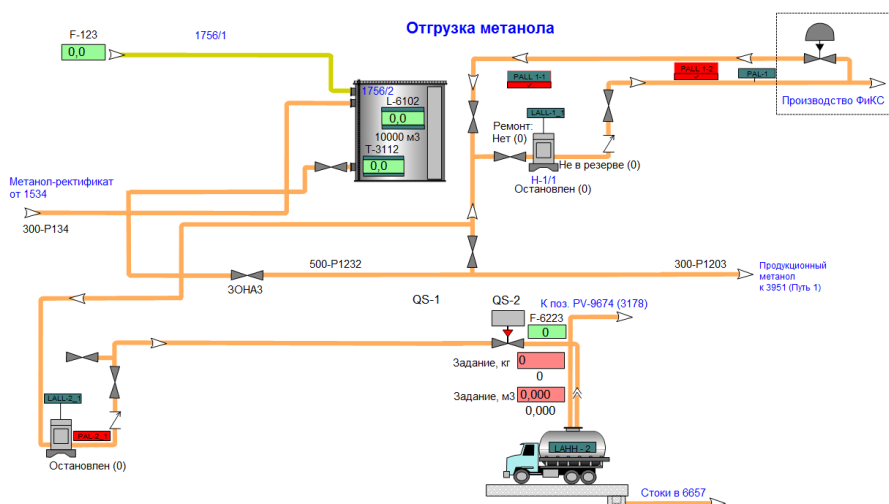


Рис. 1. Пример изображения мнемосхемы

Задачи, упомянутые выше, в полной мере решаются продуктами класса SCADA. В тоже время, на различных уровнях предприятия имеется потребность в получении информации о состоянии технологических процессов производства (например, на вышестоящем уровне MES [3] (см. рис. 2). Однако доступность таких данных между уровнями может быть ограничена политикой безопасности, территориальной распределённости предприятия или другими факторами.

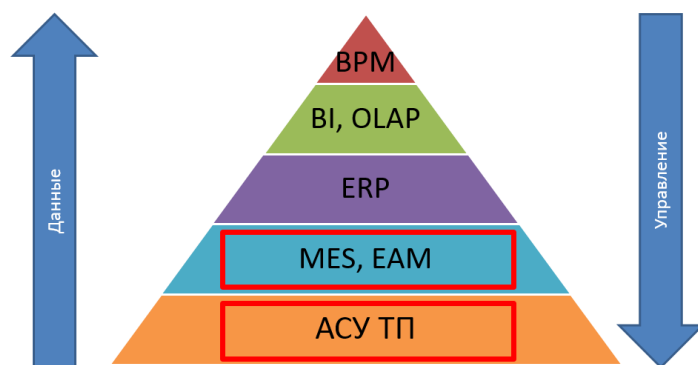


Рис. 2. Информационно-управляющая структура производственного предприятия

Заказчиком разработки является компания ООО «СибХайТекЦентр». В данной компании для создания и модификации мнемосхем используется векторный графический редактор Pro Grapher. Данный разработан на основе ActiveX компонента ProGrapherControl. Графическая ActiveX компонента является свободно распространяемым продуктом и не имеет лицензионных ограничений на применение (на встраивание в другие системы). При этом сам редактор Pro Grapher имеет лицензионное ограничение на количество используемых копий. Несмотря на то что данное технологическое решение позволяет выполнять задачи создания и использования мнемосхем, существуют лицензионные ограничения на использование данного редактора. степени замедляет процесс создания мнемосхем для крупного производства. Кроме того, существующее решение имеет недостатки в методике изготовления и использования мнемосхем [1]. Таким образом, используемые инструментальные средства на предприятии уже на этапе усложняют и ограничивают возможности создания мнемосхем.

Учитывая недостатки существующего технологического решения, а также требования, предъявляемые к нему (см. приложение Д), была поставлена цель работы: разработать графические программные средства для создания и использования мнемосхем. Одним из основным требований предприятия к разработке графических программных средств является платформа разработки Microsoft .Net и фреймворк WPF.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) Изучить предметную область работы с технологическими мнемосхемами на производстве
- 2) Изучить методы и инструменты командной работы над проектом

- 3) Выполнить проектные работы по созданию средств работы с мнемосхемами
- 4) Выполнить программную реализацию средства для работы с мнемосхемами
- 5) Интегрировать разработанные программные средства с системами на предприятии

1.2 Общий процесс создания мнемосхе и их использования на производстве

В данном разделе будет рассмотрен общий процесс создания мнемосхемы для производства. На начальном этапе жизненного цикла, мнемосхема создаётся дизайнером в специализированном графическом редакторе. Затем мнемосхема проходит этап проверки оператором-диспетчером. В случае если мнемосхема удовлетворяет требованиям, её переводят в опытную/промышленную эксплуатацию. Иначе, отправляется на доработку дизайнеру.

1.2.1 Процесс создания мнемосхем

Для наиболее полного выделения необходимых функциональных возможностей графических средств необходимо исследовать процесс создания и использования абстрактной мнемосхемы дизайнером.

Процесс создания мнемосхемы начинается с создания графической модели схемы. На данном этапе дизайнер с помощью специальных инструментов создаёт графические примитивы, такие как: полилиния, полигон, прямоугольник, дуга, текст. При создании схемы из графических примитивов дизайнер может их группировать, объединяя в одну фигуру. Это делается с целью упрощения процесса редактирования схемы, путём восприятия группы фигур как единую фигуру.

При создании фигур схемы дизайнер может изменять её геометрию, размеры, угол наклона на плоскости и место расположения на плоскости. Каждой фигуре дизайнер может настроить индивидуальный стиль оформления фигур. Кроме того, по желанию дизайнер может задать единый стиль оформления некоторым фигурам. Под стилем оформления фигуры схемы понимается его цветное оформление, стиль заливки, наличие и направление градиента, стиль и цвет границ фигуры. Дизайнеру доступно одиночное, или групповое удаление фигур схем.

Дизайнер может изменять порядок отрисовки фигур схемы на плоскости, таким образом, обеспечивая их взаимную видимость. Стоит отметить, что фигур на схеме может быть большое количество, что усложняет процесс настройки их взаимного расположения на плоскости. Решением данной проблемы является перемещение фигур на другой слой. Таким образом, благодаря послойной

отрисовке мнемосхем, фигуры на различных слоях автоматически располагаются в разных зонах видимости друг относительно друга.

Мнемосхема может быть достаточно сложной, потому дизайнер может настраивать видимость отдельных фигур, групп фигур или же слоёв для нанесения и настройки других фигур.

Мнемосхему от иного векторного графического изображения отличает наличие свойств анимации. Анимация мнемосхемы заключается в динамическом изменении её состояния в зависимости от изменения состояния технологического объекта, который описывает данная мнемосхема.

Для связи мнемосхемы с данными, дизайнер может связать фигуру схемы с источником данных в базе данных (далее БД). Таким образом, у фигуры схемы установится ссылка на источник данных в базе данных, с которого на фигуру схемы будут поступать данные. Дизайнер может произвольным образом связывать различные фигуры схемы с одним источником данных в базе данных. Следует отметить, что бывают случаи, когда данные одной фигур схемы должны быть определены как логическое выражение между различными источником данных базы данных. Таким образом, дизайнер может задать вычисление значения фигуры схемы на основе различных выражений для источников данных БД.

После настройки связи фигур схемы с источником данных БД дизайнер может настроить стиль отображения каждой фигуры схемы, связанной с БД, в зависимости от значения приходящих данных. Для этого, в первую очередь, определяется природа данных, которые приходят к фигуре схемы из БД. Под природой данных понимается форма сигнала, которая была зарегистрирована автоматикой до попадания данных в БД: дискретный или аналоговый сигнал. Дискретные сигналы отличаются от аналоговых тем, что их сигнал принимает конкретный набор значений во все моменты времени. Примером сигнала дискретной природы является состояние крана (открыт/закрыт), сигнала аналоговой природы является изменение температуры жидкости в котле (в пределах определённого диапазона значений). Таким образом, в случае использования дискретного сигнала дизайнер должен указать стиль оформления фигуры схемы в зависимости от конкретных значений источника данных БД. В случае использования аналогового сигнала, дизайнеру следует определить стиль оформления фигуры в различных диапазонах значений источника данных БД.

Для настроек стиля оформления фигуры схемы дизайнеру доступны определённые заранее стили. Однако возможно создание уникального стиля для конкретной фигуры схемы.

1.2.2 Процесс использования мнемосхем

Использование мнемосхемы выполняется на производственном предприятии для наблюдения за процессом производства оператором-диспетчером.

Определение функциональных возможностей, необходимых при использовании мнемосхем, достигается путём полного описания процесса использования мнемосхемы. Использование мнемосхемы диспетчером начинается с работы среды представления схем.

Диспетчеру отображается перечень мнемосхем в специальном представлении. В этом представлении мнемосхемы можно открывать, менять порядок, удалять, переименовывать, добавлять новые, выполнять сортировку.

Диспетчер может открыть мнемосхему в системе. При работе с мнемосхемой возможно выполнение навигации по ней, увеличивать и уменьшать отдельные участки мнемосхемы и выполняя сдвиг центра схемы. Диспетчеру доступен просмотр детальной информации по каждой фигуре схемы. Информация фигуры схемы включает: имя объекта, тип информации, текущее значение, дату обновления, достоверность значения.

Диспетчер может одновременно обозревать несколько мнемосхем, переключаясь между ними. В случае если на одной из мнемосхем у фигур включается критическое состояние, мнемосхема должна моментально отобразиться, выделяя фигуры с критическим состоянием.

В результате исследования общего процесса создания мнемосхемы дизайнером и её использования диспетчером были созданы диаграммы вариантов использования (см. Приложение Б)

1.2.3 Внутренняя структура мнемосхем

Учитывая, что мнемосхема должна представлять собой векторное изображение (см. раздел. 1), а также опираясь на результаты анализа диаграмм вариантов использования мнемосхем при создании и эксплуатации, была определена внутренняя структура мнемосхемы.

Мнемосхема должна представлять собой графический документ, содержащей в себе векторное изображение (рис. 3). Организация внутренней

структуры данных является иерархической. Основными структурами данных мнемосхем должны быть векторные геометрические примитивы (фигуры), такие как: дуги, прямоугольники, полигоны, полилинии, текст, групповые фигуры. Так как мнемосхема рассматривается на плоскости, то для пространственного размещения необходимо использовать относительную декартову систему координат. Мнемосхема может иметь произвольное количество слоёв для размещения на них фигур.

Для обеспечения связи фигур мнемосхемы с данными, фигуры должны иметь специальные идентификаторы, по которым можно их связать с объектами базы данных. Для решения задачи использования вычисляемых значений техпараметров может рассматриваться два варианта. Первый вариант решения задачи предполагает вычисление значений техпараметра на стороне сервера с использованием функции в БД. Такой подход обеспечивает высокое быстродействие, однако вносит сложность в редактирование таких техпараметров. Второй вариант предполагает получение вычисляемых значений техпараметров с использованием части клиента. Такой подход прост для редактирования выражений вычисляемых техпараметров, но более сложный в реализации.

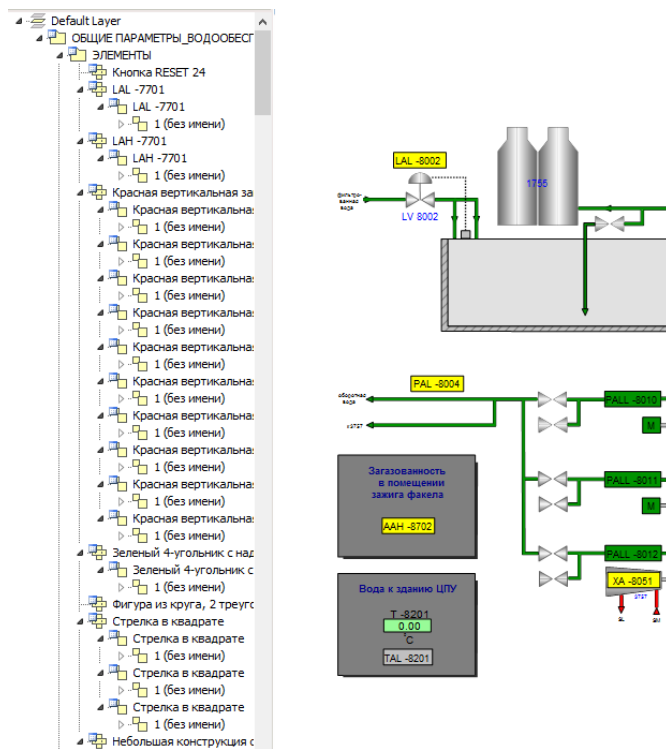


Рис. 3. Иерархическая внутренняя структура мнемосхемы

1.3 Обзор средств для работы с мнемосхемами на рынке

1.3.1 Критерии обзора средств работы с мнемосхемами

В настоящее время рынок программного обеспечения для работы с векторной графикой достаточно многообразен. Специфика работы с мнемосхемами достаточно сильно сужает круг возможных для использования программных средств. Основными требованиями, при выборе программных средств являются:

- наличие графического редактора, позволяющего создавать мнемосхемы (как упоминалось выше, отличительной чертой мнемосхемы является возможность её анимации в зависимости от поступающих данных к её элементам);
- наличие обозревателя, способного интегрироваться с внешними системами;
- цена приобретения графических средств.

1.3.2 Оценка графических средств

Из всего многообразия средств работы с векторной графикой были выделены такие средства, как:

- InTouch V10 (Wonderware) [4];
- Trace mode V6 (AdAstra Research Group, Ltd) [5];
- AutoCAD 2016 (AUTODESK) [6];
- CorelDraw X6 (Corel Corporation) [7];
- Unity3d (Unity Technologies) [8].

Для оценки данных графических средств необходимо было сделать их обзор и оценить согласно ранее определённым критериям. За полное соответствие критерию даётся 5 баллов. За частичное соответствие критерию даётся пропорциональная величина от максимального балла. За несоответствие критерию присваивается ноль баллов.

Все выбранные графические редакторы позволяют создавать графические документы (см. рис. 4). Однако если InTouch V10 и Trace mode V6 являются SCADA системами и априори имеют поддержку работы с мнемосхемами, то редакторы CorelDraw X6 и AutoCAD 2016 значительно проигрывают в этом. Последние системы являются профессиональными инструментами дизайна и автоматизированного проектирования соответственно. Потому они не содержат по умолчанию возможности анимации графического документа в зависимости от

поступающих на него данных. Хотя данные редакторы и имеют встроенное API, позволяющее создавать плагины расширения функциональных возможностей, этого будет недостаточно для создания мнемосхем. Unity3d является графическим движком, позволяющим создавать трёхмерные видео сцены. Данный игровой движок позволяет выполнить связь графического изображения с БД и выполнить анимацию изображения, что обеспечивает возможность создания мнемосхем.

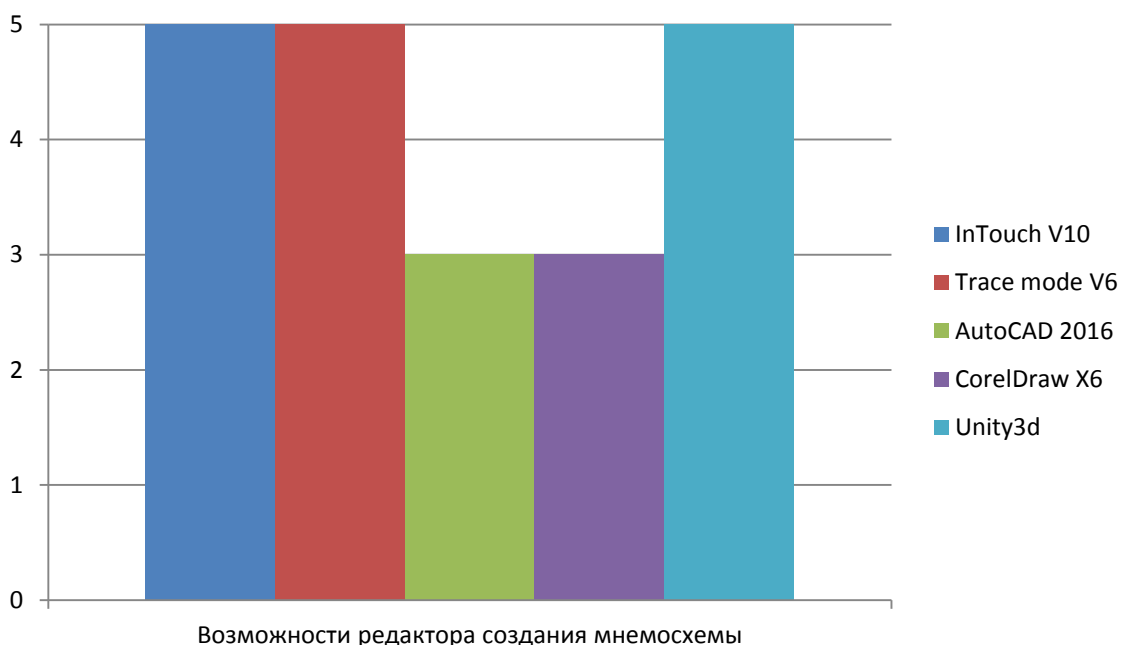


Рис. 4. Диаграмма «возможности редактора создания мнемосхем»

Рассматриваемые системы поддерживают широкие возможности по редактированию графических документов. Как писалось ранее, CorelDraw X6 и AutoCAD 2016 не являются редакторами мнемосхем и не содержат возможности создания анимационных графических документов в зависимости от входящих данных. Потому данные редакторы далее не будут рассматриваться по другим критериям. Графический движок Unity3d позволяет создавать трёхмерные сцены, выполнять их анимацию, в том числе в зависимости от данных получаемых с БД. Однако, как упоминалось ранее, мнемосхема является двумерным графическим изображением. Поэтому функциональные возможности данного графического движка излишни для решения поставленных задач.

Наличием обозревателя, который можно встраивать в сторонние системы, не может выделиться ни одна из представленных систем. InTouch V10 и Trace mode V6 являются самостоятельными системами, поставляющимися в виде

коробочной версии, которая разворачивается на предприятии. Данные системы способны интегрироваться по данным с внешними системами класса MES и EAM (см. рис. 5), однако не позволяют быть частью данных систем и предоставлять пользователям вне АСУ ТП возможностей наблюдения за мнемосхемами. Графический движок unity3d вообще не имеет возможности интеграции в сторонние системы. Потому из дальнейшего обзора исключается.

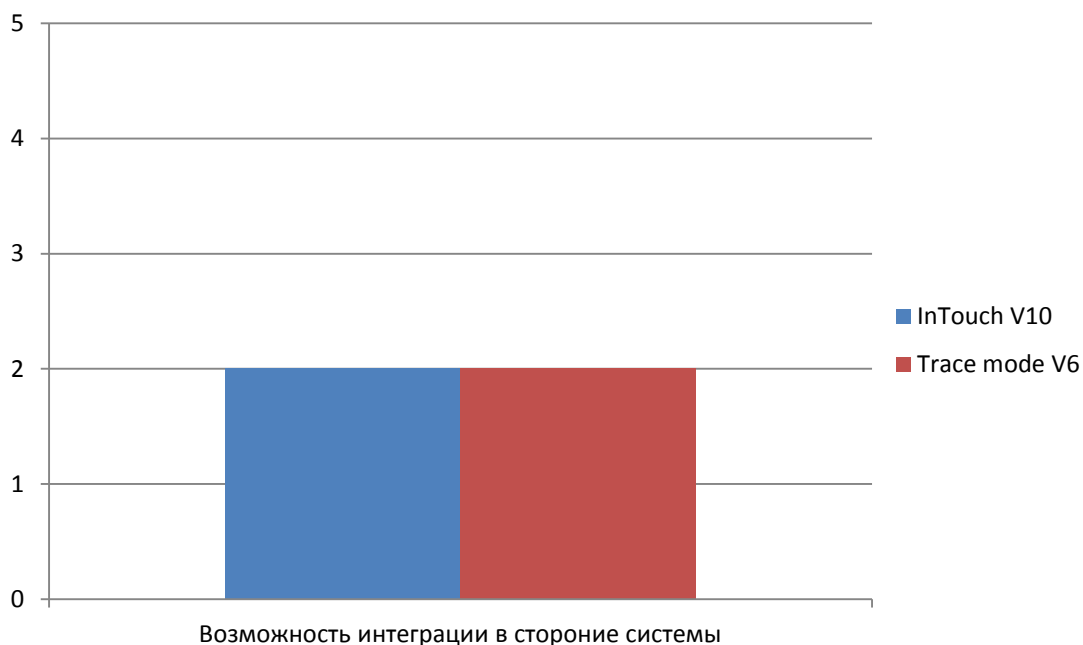


Рис. 5. Диаграмма «возможности интеграции в сторонние системы»

InTouch V10 и Trace mode V6 рассчитаны на корпоративных клиентов (на большое количество пользователей системы) (см. рис. 7). Кроме того, данные системы поставляются большим набором взаимосвязанных модулей, содержащие в себе много лишних функциональных возможностей (для решения поставленных в работе задач). Извлечение только нужных функциональных возможностей из данных систем не представляется возможным. Потому цена за необходимый объём функциональных возможностей является достаточно высокой.

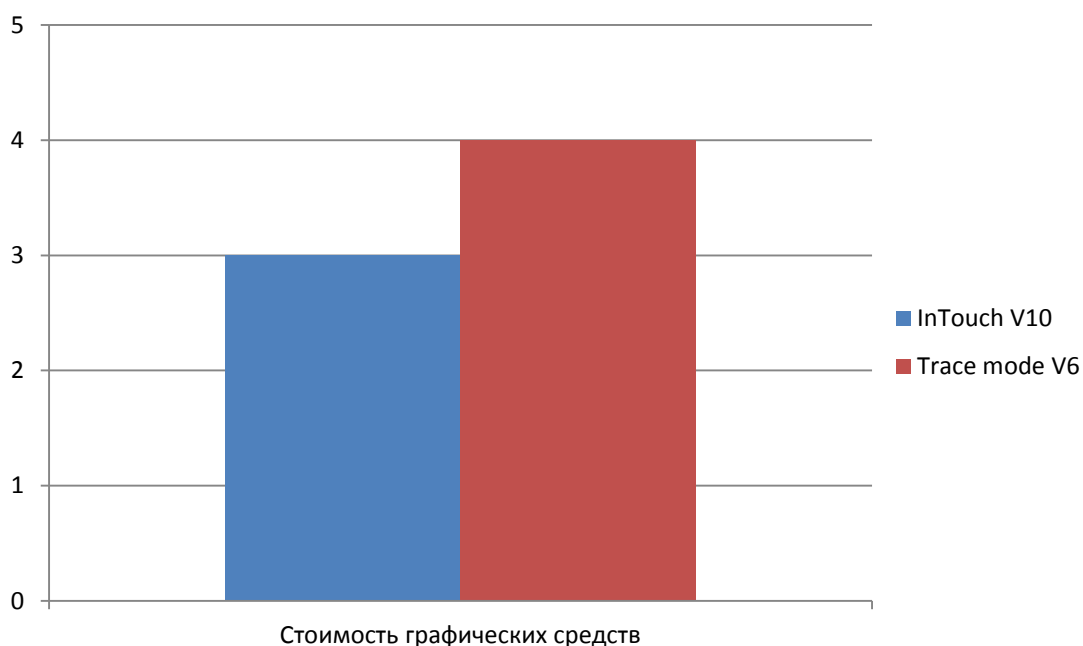


Рис. 6. Диаграмма «Стоимость графических средств»

Подводя итог обзора, следует отметить, что ни одна из рассмотренных систем полностью не удовлетворяет компанию ООО «СибХайТекЦентр» в возможностях по работе с мнемосхемами. Высший балл получила SCADA система Trace Mode V6 (см. рис 7, 11 баллов). Все рассмотренные системы позволяют перенять опыт накопленный многолетней практикой компаниями разработчиками решений для работы с векторной графикой и мнемосхемами в частности. Из SCADA систем полезными является функциональные возможности по настройке правил анимации мнемосхем. Графический редактор CorelDraw X6 и CAD система AutoCAD 2016 полезны для изучения функциональных возможностей по работе графическим документом и графическими примитивами. Графический движок Unity3d представляет большой интерес для изучения функциональных возможностей по настройке стиля оформления графических фигур.

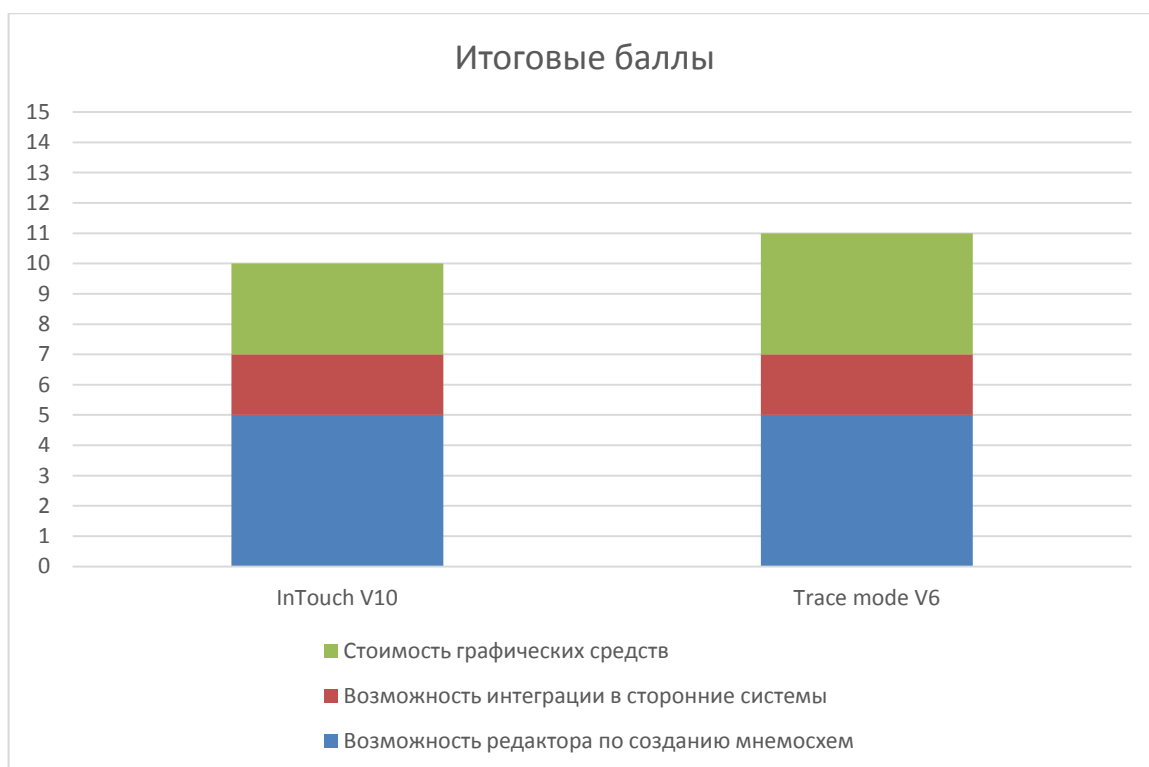


Рис. 7. Диаграмма «Стоимость графических средств»

2 РАЗРАБОТКА ГРАФИЧЕСКИХ СРЕДСТВ

2.1 Командная работа, методы и инструменты проектирования, разработки программного обеспечения

2.1.1 Команда

Компания ООО «СибХайТекЦентр» имеет широкий штат специалистов для разработки программного обеспечения. В него входят: проектные менеджеры, разработчики, сотрудники обеспечения качества, дизайнер, администрация.

Администрация предприятия решает вопросы финансов, продвижения, маркетинга и другие. Проектные менеджеры выстраивают план работ над проектами. Разработчики под управлением руководителей команд занимаются разработкой проектов. Сотрудники обеспечения качества под управлением руководителя обеспечивают соблюдение качества разрабатываемых программных продуктов. Дизайнер занимается созданием дизайна пользовательского интерфейса разрабатываемых продуктов.

Выполнение проекта начинается с инициации его административным отделом, которые смогли договориться с заказчиком, или приняли решение о

необходимости собственной разработки. Затем наиболее опытные разработчики выполняют ознакомление с техническим заданием, или созданием технического задания, если внутренняя разработка. По техническому заданию составляется иерархическая структура работ. Данная структура определяет ресурсы, которые должны быть использованы при выполнении проекта.

Проектные менеджеры, используя ИСР выполняют планирование выполнения работ согласно выбранной методологии разработки. По окончании планирования выполняется запуск проекта в разработку.

По готовности первых функциональных единиц проекта подключается отдел обеспечения качества, который выполняет проверку выполненных работ.

По окончании выполнения проекта выполняется приёмочное тестирование и процесс приёмо-сдаточных испытаний внутри компании административным отделом и отделом обеспечения качества.

2.1.2 Методологии разработки программного обеспечения в команде

Методология разработки, используемая в команде, зависит от модели взаимодействия с заказчиком. Команда ООО «СибХайТекЦентр» пользуется: Waterfall (водопадная модель), Scrum, LEAN (бережливая разработка), Extreme Programming (экстремально программирование).

Водопадная модель отличается разделением всего процесса разработки ПО на последовательные этапы, причем последующий этап начинается после того, как полностью завершён предыдущий. Как правило, такая модель используется в случаях взаимодействия с госзаказчиками и крупными корпорациями. У таких заказчиков, бюджет закладывается задолго до начала проекта. В таких проектах предполагается минимальное изменение требований [9].

Scrum — это методология управления проектами, которая построена на принципах тайм-менеджмента. Основной ее особенностью является вовлеченность в процесс разработки всех участников. Данная методология основана на, позволяющих в жёстко фиксированные и небольшие по времени итерации, называемые спринтами (sprints), предоставлять конечному пользователю работающее ПО с новыми возможностями, для которых определён наибольший приоритет. Возможности ПО к реализации в очередном спринте определяются в начале спринта на этапе планирования и не могут изменяться на

всём его протяжении. При этом строго фиксированная небольшая длительность спринта придаёт процессу разработки предсказуемость и гибкость [10].

LEAN — это модель управления разработкой ПО, основанная на постоянном стремлении к устранению всех видов потерь. LEAN предполагает вовлечение в процесс оптимизации бизнес-процессов каждого сотрудника и максимальную ориентацию на пользователя. В данной модели выполняется оценка пользы продукта для конечного пользователя, на каждом этапе его создания. В качестве основной задачи предполагается создание процесса непрерывного устранения любых действий, которые потребляют ресурсы, но не являются важными для конечного пользователя [11].

Extreme Programming – методология, которая предполагает повышение доверия заказчика к программному продукту путем предоставления реальных доказательств успешности развития процесса разработки и резкое сокращение сроков разработки продукта. При этом в данной методологии обеспечивается минимизация ошибок на ранних стадиях разработки. Это позволяет добиться максимальной скорости выпуска готового продукта и даёт возможность говорить о прогнозируемости работы. Практически все приемы данной методологии направлены на повышение качества программного продукта [12].

Все эти методологии применяются в различных проектах компании и обеспечивают удовлетворение требований заказчика к разрабатываемому ПО.

2.1.3 Инструменты разработки программного обеспечения

В процессе разработки программного обеспечения в компании используют различные инструменты. Все рассмотренные далее инструменты выбраны опытным путём в течении в многолетней практики разработки ПО в компании.

Одним из основных инструментов командной работы является Microsoft Team Foundation Server (далее TFS). TFS - продукт корпорации Microsoft, представляющий собой комплексное решение, объединяющее в себе систему управления версиями, сбор данных, построение отчётов, отслеживание статусов и изменений по проекту и предназначенное для совместной работы над проектами по разработке программного обеспечения (см. рис. 8) [13]. Данный продукт доступен как в виде отдельного приложения, так и в виде серверной платформы для Visual Studio Team System.

TFS содержит в себе следующие компоненты (см. рис. 8): система управления версиями, система построения сборки, система автоматической

сборки проекта, система тестирования, система управления рабочими элементами, отчетность и мониторинг.

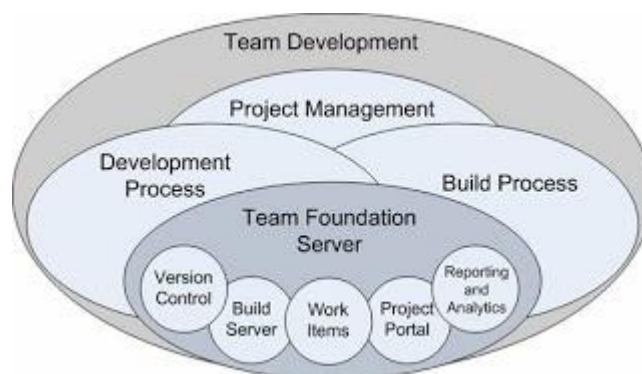


Рис. 8. Диаграмма «Инструменты в TFS»

Кроме прочего TFS имеет средство Builds — используется разработчиками для выполнения полной сборки большинства последних версий программного обеспечения, используемых в контроле кода. Записи каждой сборки сохраняются вне зависимости от её успешности или не успешности, так что разработчики и тестировщики могут отслеживать прогресс проекта.

Для выполнения проектных работ необходимо использовать инструменты, помогающие создавать документы с различным наполнением, как текстовым, так и графическим. Форма данного документа зависит от характера проекта. Если разработка заказная, то, как правило, требуется отчуждаемый проектный документ. Такой документ готовится с использованием средств MS Office [14]. Для внутренних документов, в компании используется Wiki-движок, который позволяет создавать документы в виде статей с последующим экспортом в необходимый формат. Работа над статьями может осуществляться несколькими членами команды одновременно.

Для выполнения графических схем используется редактор MS Visio и бесплатный редактор yEdit [15]. Данные редакторы обладают мощными функциональными возможностями для создания графических схем. Поддерживают большое количество форматов экспортирования.

Для создания графических эскизов пользовательского интерфейса командой используется бесплатный графический редактор Pencil [16]. Данный редактор в своей библиотеке содержит большое количество элементов пользовательского интерфейса, что позволяет быстро создавать эскизы пользовательского интерфейса для проектируемой системы.

Кроме того, в компании используется портал MS Share Point [17], позволяющий накапливать всю документацию по проектам на портале. Данный сервис имеет возможности по накоплению документов компании с учётом истории изменений.

Процесс разработки выполняется в средах Microsoft Visual Studio 2015 [18] и Embarcadero RAD Studio XE7 [19]. Для обеспечения эффективной программной реализации приобретаются наборы готовых компонент. Такие компоненты могут содержать как визуальные элементы, так и расчётные модули.

Как правило, выполняемые командой проекты содержат сложную бизнес-логику. Такие проекты требуют подходов к обеспечению качества с применением автоматизированного тестирования. Это позволяет снизить риск пропуска ошибки сотрудником QA в наиболее важных функциональных единицах разрабатываемой системы. Автоматизированные тесты выполняются на стадии continues integration при сборке проекта.

Для организации автоматизированного тестирования используются такие средства автоматизации как MS Unit Testing Framework [21], Test Complete [22] и другие. Автоматизированные тесты представляют собой скрипты, которые выполняют проверку выбранного бизнес-процесса или части кода. Кроме того, используются автоматизированные тесты пользовательского интерфейса, с применением записи действий QA-инженера и многократного их повторения системой автоматизированного тестирования.

2.2 Используемые методы и инструменты

Проект, выполненный в данной работе, разрабатывался командой с использованием TFS. Средой разработки была выбрана MS Visual Studio 2015, поскольку одним из основных требований к разрабатываемым программным средствам было: использование фреймворка WPF [22]. Все проектные работы выполнялись с помощью MS Word, Visio и редактора Pencil.

В качестве методологии разработки ПО была выбрана гибкая методология SCRUM. В Scrum-проектах для управления требованиями следует использовать подход “just-in-time” (“в нужный момент”). Этот подход предполагает заблаговременное описание функциональных возможностей лишь общего характера, которые будут постепенно уточняться по мере выполнения проекта через многократные совещания с обсуждениями.

При обсуждении у команды имеется возможность постепенно уточнять смысл функциональных и нефункциональных возможностей продукта. Это позволяет не полагаться только на информацию, изложенную в документе, которая кажется читателю априори верной.

Автор настоящей работы внёс вклад в каждый из этапов разработки данного ПО. Автором совместно с командой опытных разработчиков, а так же научным руководителем, было составлено техническое задание. На основе технического задания была создана ИСР. Совместно с проектным менеджером автор выполнил планирование выполнения работ. Автор под руководством научного руководителя и опытных разработчиков выполнил проектные работы разрабатываемых программных средств. Далее выполняя роль командного лидера автор выполнял разработку программных средств с командой разработчиков. В процессе разработки был подключен отдел обеспечения качества, который выполнял тестирование разработанных функциональных единиц. Автор также выполнил часть задач по тестированию проекта. На завершающей стадии проекта автор выполнял приёмо-сдаточные испытания, демонстрируя работу разработанных программных средств административному отделу компании.

2.3 Векторная графическая библиотека

2.3.1 Редактор и обозреватель мнемосхем: общее и различия

Рассматриваемый подход к процессу работы с мнемосхемами предполагает разделение среды редактирования и среды отображения мнемосхем (подробно рассмотрены в научно-техническом отчёте выполнения научно-исследовательской практики в июле 2015 года). Среда отображения может являться автономной частью, или встраиваемой автономной частью, например подсистемой MES [3]. Среда редактирования предназначена не только для отображения, но и для модификации мнемосхем, и может быть представлена в виде автономного редактора.

Исходя из описанного выше процесса, наличие общей функциональности у редактора и средства отображения мнемосхем является основанием для создания универсальной графической компоненты (рис. 9). Требования, предъявляемые к графической компоненте, находятся на пересечении требований, предъявляемых к графическому редактору и средству отображения

мнемосхем. Данная универсальная векторная компонента не зависит от предметной области.



Рис. 9. Место графической компоненты в средствах работы с мнемосхемами

Такой подход позволяет сэкономить время на разработку программных средств работы с мнемосхемами. Кроме наличия графической компоненты подход вносит универсальность, позволяющую создавать различные программные средства, использующие одну объектную модель. Использование общей объектной модели для средств редактирования и представления мнемосхем даёт возможность использовать один формат хранения данных схем, что значительно облегчает их использование.

2.3.2 Проектирование векторной графической компоненты

2.3.2.1 Описание бизнес-процессов

В данном разделе выполнено описание бизнес-процессов работы с мнемосхемами, подлежащих автоматизации (см. рис. 10).

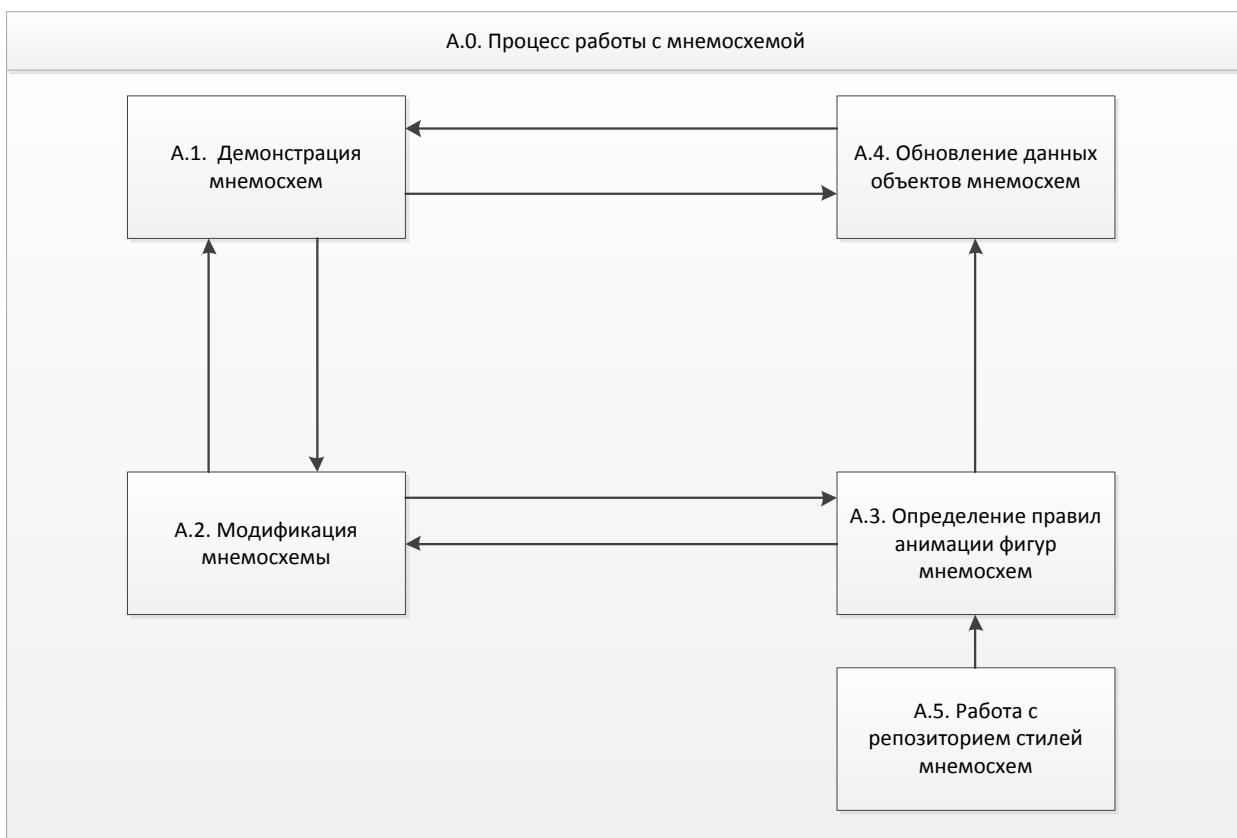


Рис. 10. Описание бизнес-процессов

Процесс работы с мнемосхемами включает в себя следующие бизнес-процессы:

- А1. Демонстрация мнемосхем;
- А2. Модификация мнемосхемы;
- А3. Определение правил анимации фигур мнемосхем;
- А4. Обновление данных объектов мнемосхем;
- А5. Работа с репозиторием стилей мнемосхем.

2.3.2.2 Описание функций

2.3.2.2.1 Демонстрация мнемосхем

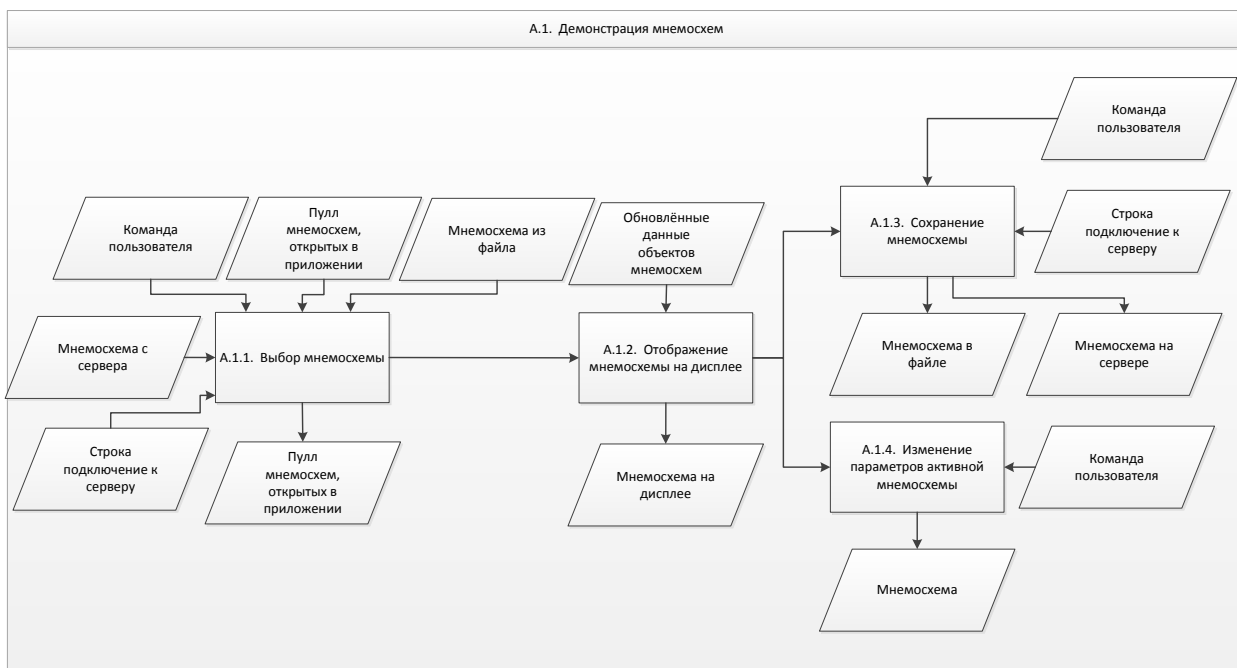


Рис. 11. Детализация бизнес-процесса «Демонстрация мнемосхем»

Таблица 2.1 – Описание бизнес-процесса «демонстрация мнемосхем»

Код	Функция	Входные данные	Выходные данные
A.1.1.	Выбор мнемосхемы	1. Команда пользователя 2. Пулл мнемосхем, открытых в приложении 3. Мнемосхема из файла 4. Мнемосхема с сервера 5. Строка подключения к серверу	Пулл мнемосхем, открытых в приложении
A.1.2.	Отображение мнемосхемы на дисплее	Обновлённые данные объектов мнемосхем	Мнемосхема на дисплее
A.1.3.	Сохранение мнемосхемы	1. Команда пользователя 2. Строка подключения к серверу	1. Мнемосхема в файле 2. Мнемосхема на сервере

			сервере 3. Строка подключения к серверу
A.1.4.	Изменение параметров активной мнемосхемы	Команда пользователя	Мнемосхема

2.3.2.2.2 Модификация мнемосхемы

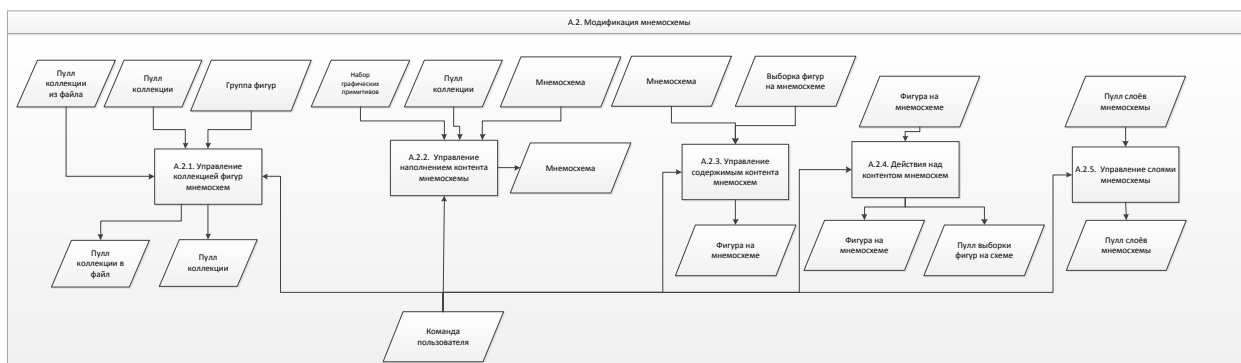


Рис. 12. Детализация бизнес-процесса «Модификация мнемосхемы»

Таблица 2.2 – Описание бизнес-процесса «модификация мнемосхем»

Код	Функция	Входные данные	Выходные данные
A.2.1.	Управление коллекцией фигур мнемосхем	1. Пулл коллекции 2. Группа фигур 3. Пулл коллекции из файла 4. Команда пользователя	1. Пулл коллекции 2. Пулл коллекции в файл
A.2.2.	Управление наполнением контента мнемосхемы	1. Набор графических примитивов 2. Пулл коллекции 3. Мнемосхема 4. Команда пользователя	Мнемосхема

A.2.3.	Управление содержимым контента мнемосхем	1. Мнемосхема 2. Выборка фигур на мнемосхеме 3. Команда пользователя	Фигура на мнемосхеме
A.2.4.	Действия над контентом мнемосхем	1. Фигура на мнемосхеме 2. Команда пользователя	3. Фигура на мнемосхеме 4. Пулл выборки фигур на схеме
A.2.5.	Управление слоями мнемосхемы	1. Пулл слоёв мнемосхемы 2. Команда пользователя	Пулл слоёв мнемосхемы

2.3.2.2.3 Определение правил анимации фигур мнемосхем



Рис. 13. Детализация бизнес-процесса «Определение правил анимации фигур»

Таблица 2.3 – Описание бизнес-процесса «определение правил анимации фигур»

Код	Функция	Входные данные	Выходные данные
A.3.1.	Изменение в параметрах фигур идентификатора объекта источника данных	1. Выборка фигур на мнемосхеме 2. Идентификатор	

		3. Команда пользователя	
A.3.2.	Выбор правила анимации	1. Пулл стилей анимации 2. Команда пользователя	
A.3.3.	Управление пуллом связанных фигур	Пулл связанных фигур	Пулл связанных фигур

2.3.2.2.4 Обновление данных объектов мнемосхем



Рис. 14. Детализация бизнес-процесса «Обновление данных объектов мнемосхем»

Таблица 2.4 – Описание бизнес-процесса «обновление данных объектов мнемосхем»

Код	Функция	Входные данные	Выходные данные
A.4.1.	Получение обновлённых	1. Пулл связанных	Данные от

	данных от источника данных	фигур 2. Команда пользователя 3. Строка подключения к источнику данных 4. Команда пользователя	источника данных
A.4.2.	Обновление данных каждой фигуры пулла связанных фигур		Пулл связанных фигур

2.3.2.2.5 Работа с репозиторием стилей мнемосхем

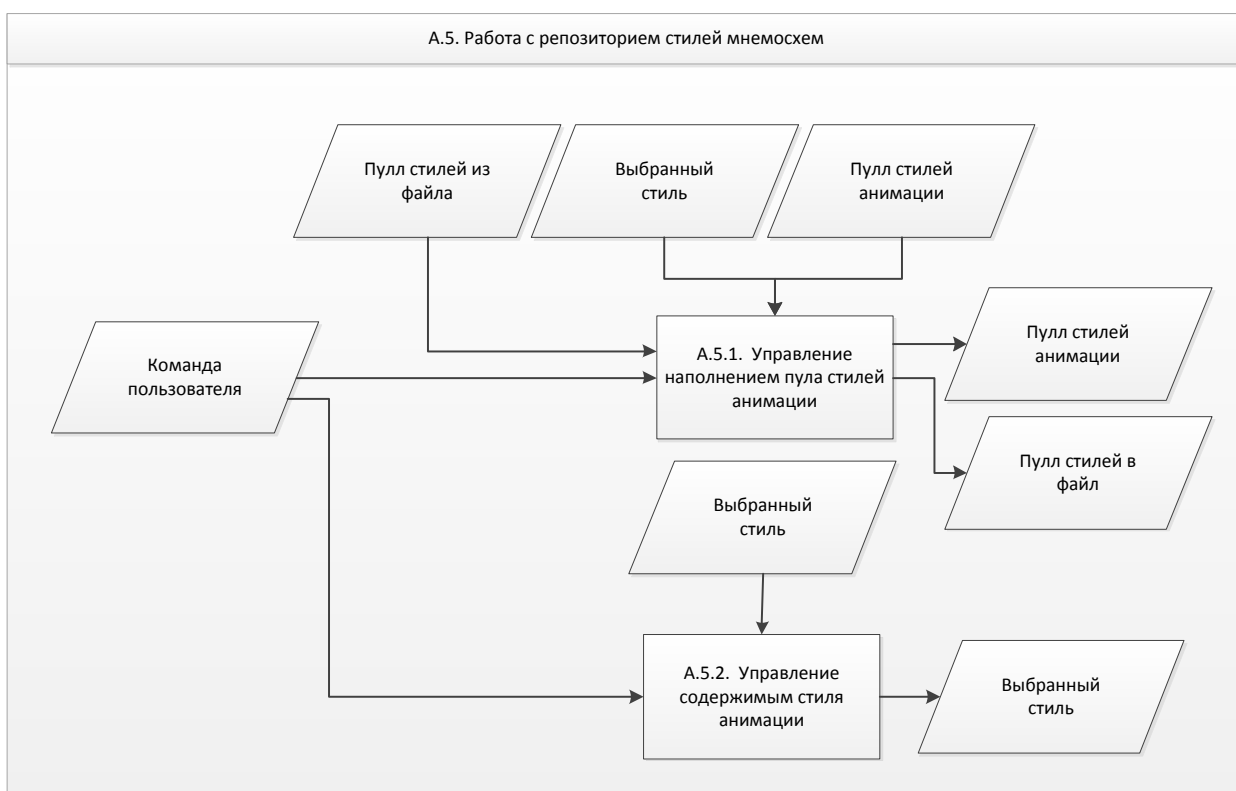


Рис. 15. Детализация бизнес-процесса «Работа с репозиторием стилей мнемосхем»

Таблица 2.5 – Описание бизнес-процесса «Работа с репозиторием стилей мнемосхем»

Код	Функция	Входные данные	Выходные данные
A.5.1.	Управление наполнением пула	1. Выбранный	1. Пулл

	стилей анимации	стиль 2. Пулл стилей анимации 3. Команда пользователя 4. Пулл стилей из файла	стилей анимации 2. Пулл стилей в файл
A.5.2.	Управление содержимым стиля анимации	1. Выбранный стиль 2. Команда пользователя	Выбранный стиль

2.3.3 Объектная модель векторной графической библиотеки

В результате процесса проектирования, была создана объектная модель векторной графической компоненты, структурная схема которой представлена на рисунке 16, а описание классов объектной представлено в приложении В.

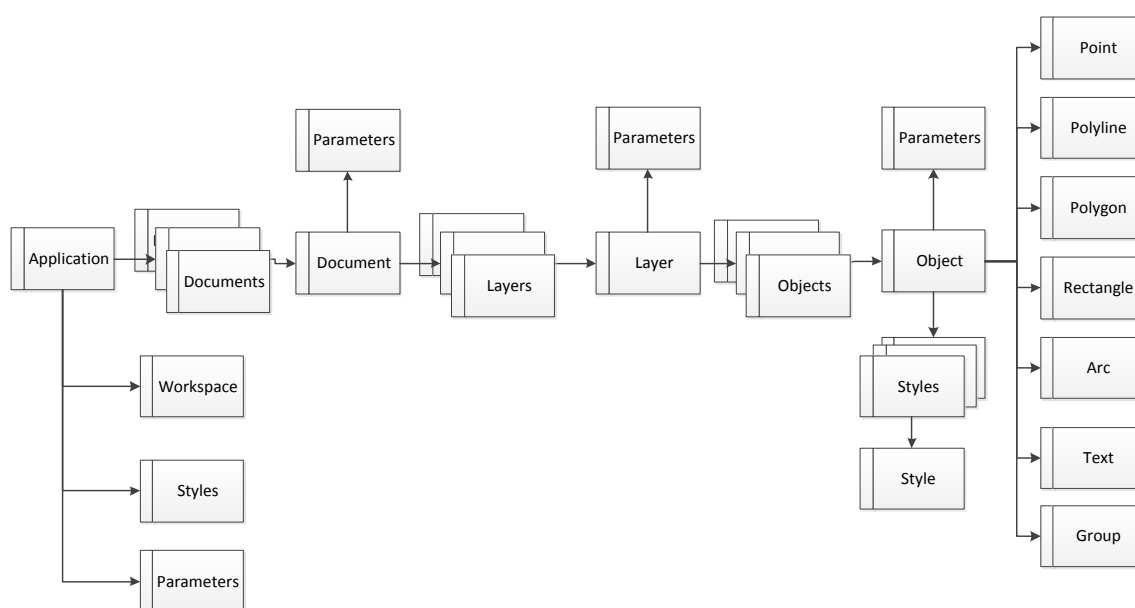


Рис. 16. Структурная схема объектной модели векторной графической библиотеки

Компанией было поставлено требование необходимости реализации векторной графической компоненты с использованием Фреймворка WPF [22], что предполагает использование платформы Microsoft .Net 4.5 [24, 25]. Технология WPF имеет все необходимые функциональные возможности для работы в 2D графикой [22]. Данная технология позволяет легко работать с XML форматом, в котором можно хранить созданные графические мнемосхемы.

Далее будет приведено описание функциональных возможностей векторной графической компоненты.

Поскольку рассматривается объектная модель универсальной векторной графической компоненты (рис. 16), на основе которой будут создаваться средства работы с мнемосхемами, была необходимость создания класса Application, который является базой для всей спроектированной компоненты.

Для обеспечения возможности мульти документного редактирования мнемосхем, была предусмотрена подчинённая коллекция Documents у класса Application. Коллекция Documents позволяет в себе хранить набор классов Document. Класс Document является основой графического документа. Он содержит коллекцию класса Layers, что обеспечивает послойную организацию данных на схеме, для удобства её редактирования.

Класс Layer является объединением классов Object в один слой по определённому признаку. Наличие класса Layer позволяет облегчить редактирование схемы путём объединения объектов в один слой. Объединение объектов в слое достигается коллекцией Objects классов Object.

Класс Object является одним из базовых классов графической компоненты. Именно он является родителем для всех классов графических примитивов. Его наличие в компоненте объясняется наличием базовых полей, методов и свойств у каждого из классов графических примитивов, таких как: Point, Polyline, Polygon, Rectangle, Text, Group. Таким образом общие для выше перечисленных классов поля, методы и свойства не создаются у каждого класса в отдельности, а наследуются от родительского класса Object.

Для обеспечения связи фигуры схемы и объектов БД, у классов Application, Document, Layer, Object, предусмотрена связь с классом Parameters. Класс Parameter обеспечивает сохранение уникальных свойств для каждой фигуры, таких как, например, уникальный идентификатор.

Поскольку основным свойством мнемосхем, является их динамическое отображение процесса производства, то нельзя обойтись без графического оформления схем. Для этих целей у класса Object была создана коллекция Styles класса Style. Класс Style имеет набор графических свойств, который позволяет настроить окраску фигуры уникальным стилем. Кроме того, наличие коллекции Styles позволяет создавать наборы стилей отображения фигур. Присваивая фигуре определённый стиль, его изменение влечёт изменение всех фигур окрашенных данным стилем.

Поскольку мнемосхема является функциональной схемой какого-то реально объекта, то является необходимым наличие функциональной возможности создания сложных графических объектов состоящих из нескольких графических примитивов. Такая функциональная возможность обеспечивается наличием класса Group. Данный класс является наследником класса Object, и содержит в себе набор ссылок на объекты, которые в себя включает. Изменение пространственных свойств, и некоторых геометрических свойств фигуры класса Group влечёт изменение этих свойств и у подчинённых фигур.

Поскольку на основе данной компоненты будет возможно построение графического редактора, то для обеспечения инструментов редактирования схемы был создан класс Workspace. Данный класс имеет поля, методы и свойства, обеспечивающие изменение размеров, положения в пространстве и угла поворота фигур на схеме. Кроме того, данный класс позволяет обработать действия над фигурами вызываемые клавишами мыши и клавиатуры.

Согласно спроектированной объектной модели была выполнена программная реализация векторной графической компоненты, представляющей DLL библиотеку, реализованную на 3231 строках кода на языке C#.

2.4 Обзоратель мнемосхем

2.4.1 Проектные решения для разработки обзорателя мнемосхем

Согласно подходу, рассмотренному в разделе 2.3.1, обзоратель мнемосхем представляет собой автономный встраиваемый модуль. Функциональная часть данного модуля обеспечивается средствами графической библиотеки. Этот модуль должен представлять собой динамический интерпретатор данных источника в графическом виде.

В глобальном смысле обзоратель должен содержать в себе всю бизнес-логику, а так же правила, способствующие отображению мнемосхемы в графическом виде.

Обзоратель должен содержать в себе открытые интерфейсы, которые позволяют выполнить его интеграцию с другой системой (см. рис. 17). Сторонняя система, в свою очередь, должна иметь механизм для встраивания в неё расширяющих модулей провайдеров (plug-in). Стороннее приложение предоставляет API, которые плагин может использовать. К ним относится предоставляемая плагину возможность зарегистрировать себя в стороннем приложении, а также протокол обмена данными с другими плагинами. Плагины

являются зависимыми от API, предоставляемых сторонним приложением и зачастую отдельно не используются. Стороннее приложение независимо оперирует плагинами, предоставляя конечным пользователям возможность динамически добавлять и обновлять плагины без необходимости внесения изменений в основное приложение. Провайдер, являющийся посредником между обозревателем и сторонней системой, позволяет им осуществлять двустороннее взаимодействие.

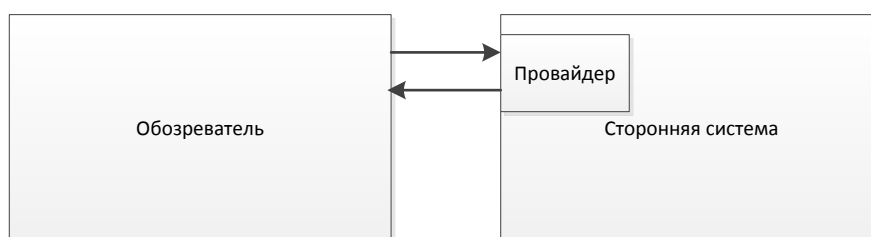


Рис. 17. Архитектура обозревателя мнемосхем

Поскольку обозреватель представляет собой встраиваемый модуль, способный лишь динамически отображать данные мнемосхем, логика и интерфейсы взаимодействия с ним пользователя должны быть реализована в сторонней системе (навигация по мнемосхеме, просмотр свойств мнемосхемы и др.).

Проектные решения, описывающие функциональные возможности обозревателя, раскрыты ниже.

2.4.1.1 Демонстрация мнемосхем обозревателем

Обозреватель способен отобразить мнемосхему на рабочей области (область пространства дисплея, отведённая для обозревателя). Под отображением мнемосхемы подразумевает графическая прорисовка её контента, а также прорисовка дополнительных системных объектов (например, границ выделения фигуры). В процессе демонстрации мнемосхемы обозреватель позволяет выбрать мнемосхему для отображения, изменить параметры активной мнемосхемы, выполнить публикацию мнемосхемы на сервер и удалить опубликованную мнемосхему с сервера.

2.4.1.1.1 Выбор мнемосхемы

Обозреватель позволяет выбрать мнемосхему для работы. Доступно открытие мнемосхемы из файла, открытие опубликованной мнемосхемы, активация текущей мнемосхемы, закрытие текущей мнемосхемы.

2.4.1.1.1 Открытие мнемосхемы из файла

Обозреватель позволяет открыть мнемосхему из файла формата XML. Для выполнения открытия мнемосхемы из файла векторная графическая библиотека принимает путь к файлу. Открытая мнемосхема становится активной после открытия. Контент мнемосхемы отображается в рабочей области.

2.4.1.1.2 Открытие опубликованной мнемосхемы с сервера

Обозреватель позволяет открыть мнемосхему, опубликованную на сервере. Для этого выполняется запрос провайдеру на получение мнемосхемы с сервера. Мнемосхемы на сервере хранятся в формате XML. При получении мнемосхемы она отображается в рабочей области.

2.4.1.1.3 Активация мнемосхемы среди открытых мнемосхем

Обозреватель позволяет одновременно работать с несколькими мнемосхемами. Все открытые мнемосхемы содержатся в пулле мнемосхем. Отображаемая мнемосхема является активной в пулле мнемосхем. Активность мнемосхем в пулле может быть изменена.

2.4.1.1.4 Закрытие текущей мнемосхемы

Обозреватель позволяет закрыть мнемосхему. При закрытии мнемосхемы выполняется её удаление из пулла мнемосхем (при этом активной становится мнемосхема, находящаяся в начале пулла).

2.4.1.1.2 Изменение параметров активной мнемосхемы

Обозреватель позволяет настроить параметры демонстрации мнемосхем. Такими параметрами являются: изменение название мнемосхемы, изменение масштаба мнемосхемы, сдвиг мнемосхемы.

2.4.1.1.2.1 Изменение названия мнемосхемы

Обозреватель позволяет изменить название мнемосхемы. Доступно назначение названия мнемосхемы отличное от названия файла, или опубликованной мнемосхемы на сервере.

2.4.1.1.2.2 Изменение масштаба мнемосхемы

Обозреватель позволяет изменить масштаб мнемосхемы в рабочей области. В процессе изменения мнемосхемы выполняется изменение метрики координат: при увеличении становится детальней, при уменьшении становится более общим.

2.4.1.1.2.3 Сдвиг мнемосхемы

Обозреватель позволяет изменить центр мнемосхемы в пределах координатной плоскости рабочей области.

2.4.1.1.3 Публикация мнемосхемы на сервере

Обозреватель позволяет опубликовать мнемосхему на сервере. При публикации мнемосхемы на сервер выполняется преобразование её структур в формат XML. Мнемосхема передаётся провайдеру, который выполняет её публикацию.

2.4.1.1.4 Удаление опубликованной мнемосхемы с сервера

Обозреватель позволяет удалить мнемосхему с сервера. Для этого провайдеру передаётся команда с идентификатором мнемосхемы, которая должна быть удалена.

2.4.1.2 Анимация мнемосхем обозревателем

Неотъемлемой частью работы с мнемосхемами является их динамическое представление. Другими словами, анимация. Обозреватель позволяет выполнить анимацию мнемосхемы с заданными параметрами. Анимация подразумевает получение данных технологических параметров от источников, их верификацию, и перерисовку мнемосхемы согласно правил анимации.

2.4.1.2.1 Правила анимации и стили анимации

Правила анимации содержат в себе сопоставление стилей и значение технологических параметров, при которых фигура должна быть окрашена данным стилем. Правила анимации поставляются совместно с мнемосхемой.

2.4.1.2.1.1 Обновление стилей анимации

Стили анимации загружаются в обозреватель из файла или распространяются с сервера. Стили анимации хранятся в XML формате в файле или на сервере. Набор стилей содержит версию. Для обновления набора стилей провайдеру посылается запрос на обновление стилей.

2.4.1.2.2 Обновление данных объектов мнемосхем

Обновление данных фигур выполняется согласно заданному периоду обновления. При наступлении времени обновления выполняется опрос провайдера на получение данных фигур. Полученные данные содержат значения технологических параметров, время прихода.

2.4.1.2.2.1 Верификация полученных данных

Обозреватель выполняет верификацию полученных данных от провайдера. Если разница между временем прихода данных и временем последнего значения данных фигуры, больше чем интервал обновления, то техпараметру присевается флаг: недостоверно.

2.4.1.2.3 Анимация

После обновления данных выполняется анимация мнемосхемы. Анимация предполагает обновление изображения всех фигур, согласно поступившим данным по установленным правилам анимации для данных фигур.

2.4.1.3 Работа провайдера

Как было описано ранее, обозреватель выполняет взаимодействие с внешней системой через посредника – провайдера. Провайдер представляет собой плагин, который встраивается во внешнюю систему и обеспечивает её взаимодействие с обозревателем.

2.4.1.3.1 Опрос провайдера

Обозреватель с заданной периодичностью опрашивает провайдер о наличии свежих данных технологических параметров из внешней системы. Провайдер в свою очередь содержит в себе механизмы взаимодействия с внешней системой. Провайдер выполняет опрос внешней системы.

2.4.1.3.2 Передача команд обозревателю от внешней системы

Поскольку обозреватель представляет собой модуль, первоначально несущий пользователю только графическую информацию, то задачи управления обозревателем со стороны пользователя ложатся на внешнюю систему. Во внешней системе должен быть реализован пользовательский интерфейс для управления обозревателем. Команды от данного пользовательского интерфейса передаются обозревателю через провайдер.

2.4.1.3.3 Подключение к источнику данных

Подключение обозревателя к источнику данных осуществляется через провайдер, в который передаётся строка подключения. В случае если подключение не установлено, возвращается ошибка.

2.4.2 Программная реализация обозревателя мнемосхем

Программная реализация обозревателя заключалась в реализации самого обозревателя и провайдера, который будет встраиваться в внешнюю систему как плагин.

Поскольку в векторной графической библиотеке уже реализованы все необходимые методы для работы с мнемосхемами, программная реализация обозревателя в большей степени свелась к созданию графической части и механизма анимации мнемосхем.

Поскольку разработка выполняется с использованием паттерна MVVM [24], для графического отображения рабочей области необходимо было создать View, ViewModel и Model.

Компонент Model в своей сути содержит методы и свойства, которые обращаются к методам и свойствам библиотеки, добавленной в проект. Библиотека имеет открытые методы, к которым выполнена привязка в компоненте Model.

2.4.2.1 Создание графической части обозревателя

При реализации графической части в первую очередь следовало обратить внимание на координатную плоскость полотна, на котором должна выполняться отрисовка мнемосхемы. В качестве координатной плоскости был выбран компонент WPF Canvas. Canvas позволяет размещать элементы, используя относительную декартову систему координат, что является ценным инструментом, когда требуется построить графический объект с координатной привязкой. Для позиционирования элемента в контейнере Canvas устанавливаются присоединенные свойства Canvas.Left и Canvas.Top. Свойство Canvas.Left задает количество единиц измерения между левой гранью элемента и левой границей Canvas. Свойство Canvas.Top устанавливает количество единиц измерения между вершиной элемента и верхней границей Canvas. Как всегда, эти значения выражаются в независимых от устройства единицах измерения, которые соответствуют обычным пикселям, когда системная установка DPI составляет 96 dpi.

Для отрисовки графических примитивов использовался компонент Graphics Drawing Tool. Данный компонент содержит в себе весь необходимый набор графических примитивов. Для каждого графического примитива доступен широкий набор свойств стилистического оформления.

2.4.2.2 Создание механизма анимации мнемосхем

С векторной графической библиотекой поставляется набор классов для работы с правилами анимации и стилями. Однако реализация самого механизма анимации полностью ложится на обозреватель.

Для того, чтобы обеспечить анимацию мнемосхемы, согласно проекту необходимо выполнять запрос технологических данных фигур с заданной периодичностью, выполнять их верификацию и отрисовывать фигуры мнемосхемы стилями согласно правил анимации. В проект был добавлен компонент `Timer`. Данный компонент выполняет вызов метода `MnemonicRefresh()`, с заданной периодичностью. Для обеспечения возможности работы в приложении независимо от анимации компонент `Timer` выполняет работу асинхронно.

Асинхронность работы анимации достигается за счёт использование класса `BackgroundWorker`. Компонент `BackgroundWorker` предоставляет почти возможность запуска длительно выполняющихся задач в отдельном потоке. Он использует диспетчер "за кулисами" и абстрагирует сложности маршализации с помощью модели событий. `BackgroundWorker` также поддерживает два дополнительных удобства: события продвижения и сообщения отмены. В обоих случаях детали многопоточности скрыты, что облегчает кодирование [N].

Метод `MnemonicRefresh()` выполняет получение данных от источника через провайдер, вызовом метода `GetTechparamsData()`. По получению данных выполняется их верификация методом `VerifyTechparamsData()`. Затем выполняется обход всех фигур мнемосхемы, у которых имеется привязка к данным и изменение их стиля согласно правилам анимации используя метод `RefreshFiguresStyle()`. Те фигуры, которые имеют данные технологических параметров «недостоверные», отмечаются особым стилем.

2.4.2.3 Создание провайдера

Провайдер является контейнером-переводчиком. Он выполняет связь внешней системы с обозревателем. Провайдер работает асинхронно, используя компонент `BackgroundWorker` [24]. Провайдер одновременно принимает запросы от обозревателя и внешней системы. Получение команд от внешней системы выполняется методом `GetParentSystemCommands()`. При получении команды, провайдер выполняет её парсинг и вызывает соответствующее методы обозревателя.

При регистрации провайдера как плагина во внешней системе, ему передаётся строка подключения к серверу, от которого он будет получать данные техпараметров.

По запросу от обозревателя провайдер вызывает метод `GetTechParamsFromServer(ConnectionString)` используя строку подключения. Получение техпараметров осуществляется пачками. Размер пачки определяется в настройках провайдера. Данная особенность необходимо для снижения нагрузки на сервер. Полученные пачки техпараметров тут же передаются обозревателю.

2.5 Редактор мнемосхем

2.5.1 Проектные решения для разработки редактора мнемосхем

Проектные работы, связанные с редактором мнемосхем в первую очередь заключаются в создании интерфейсной части для обеспечения доступа к бизнес-логике работы с мнемосхемами, заложенной в векторной графической библиотеке. Проектные работы выполнялись с использованием классических паттернов проектирования [26].

В процессе проектной деятельности были созданы эскизы пользовательского интерфейса графического редактора мнемосхем. Пользовательский интерфейс графического редактора мнемосхем должен содержать: главное окно (см. рис. 18), окно настроек стиля фигуры (см. рис. 19), окно атрибутов фигуры (см. рис. 20), окно настроек стилей анимации (см. рис. 21), окно настроек правил анимации (см. рис. 22).

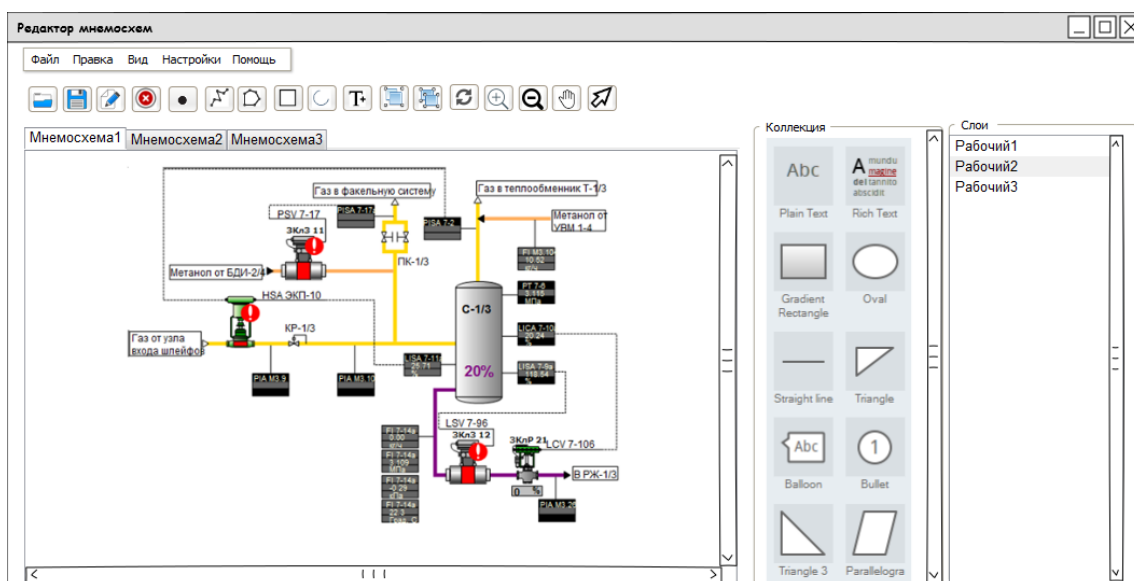


Рис. 18. Эскиз главного окна редактора мнемосхем

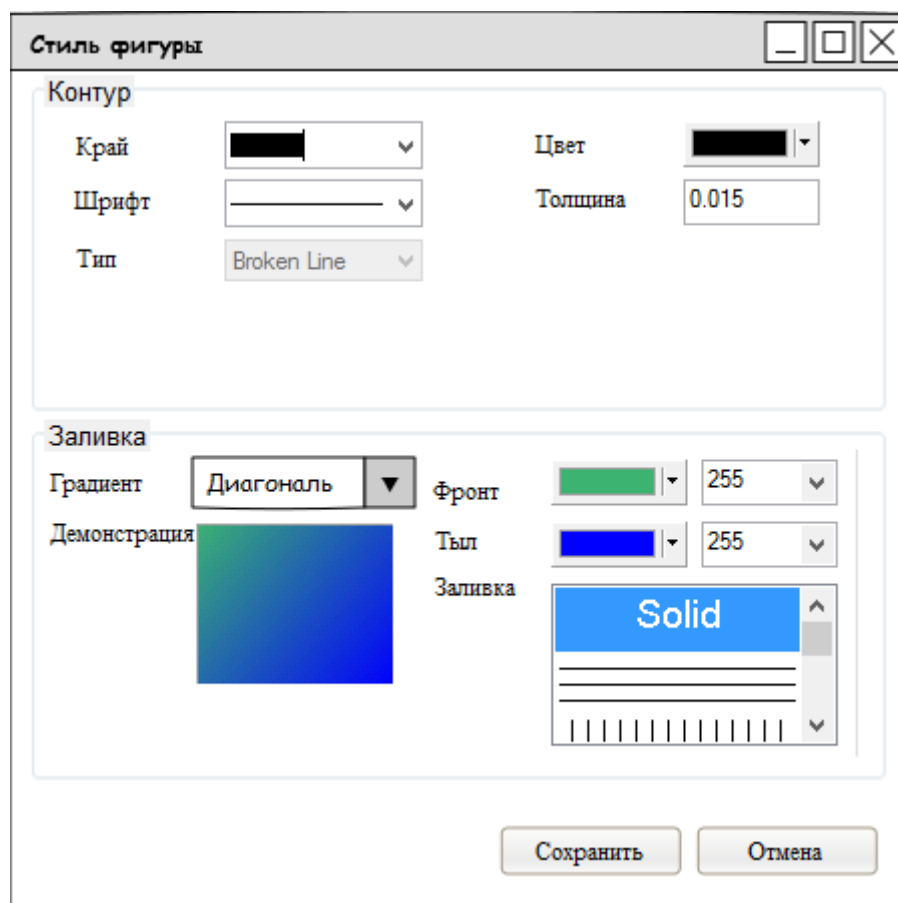


Рис. 19. Эскиз окна настроек стиля фигуры

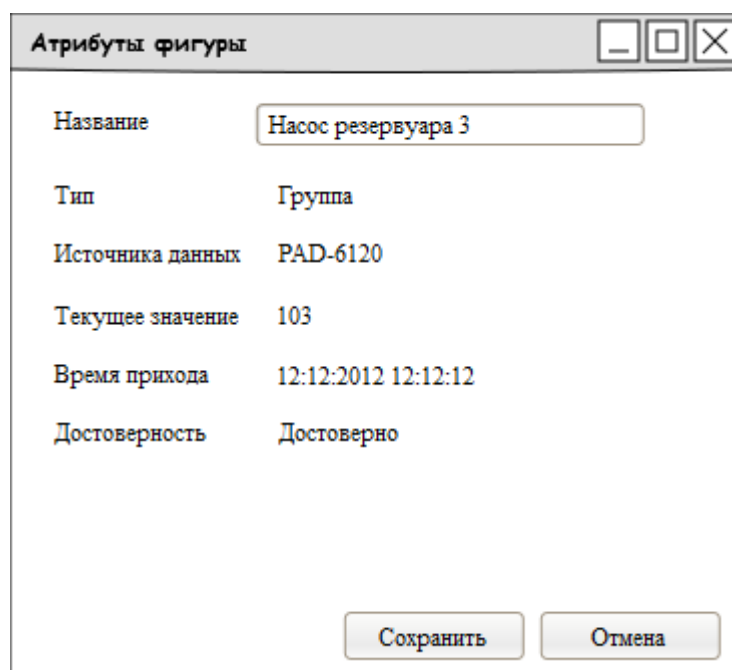


Рис. 20. Эскиз окна атрибутов фигуры

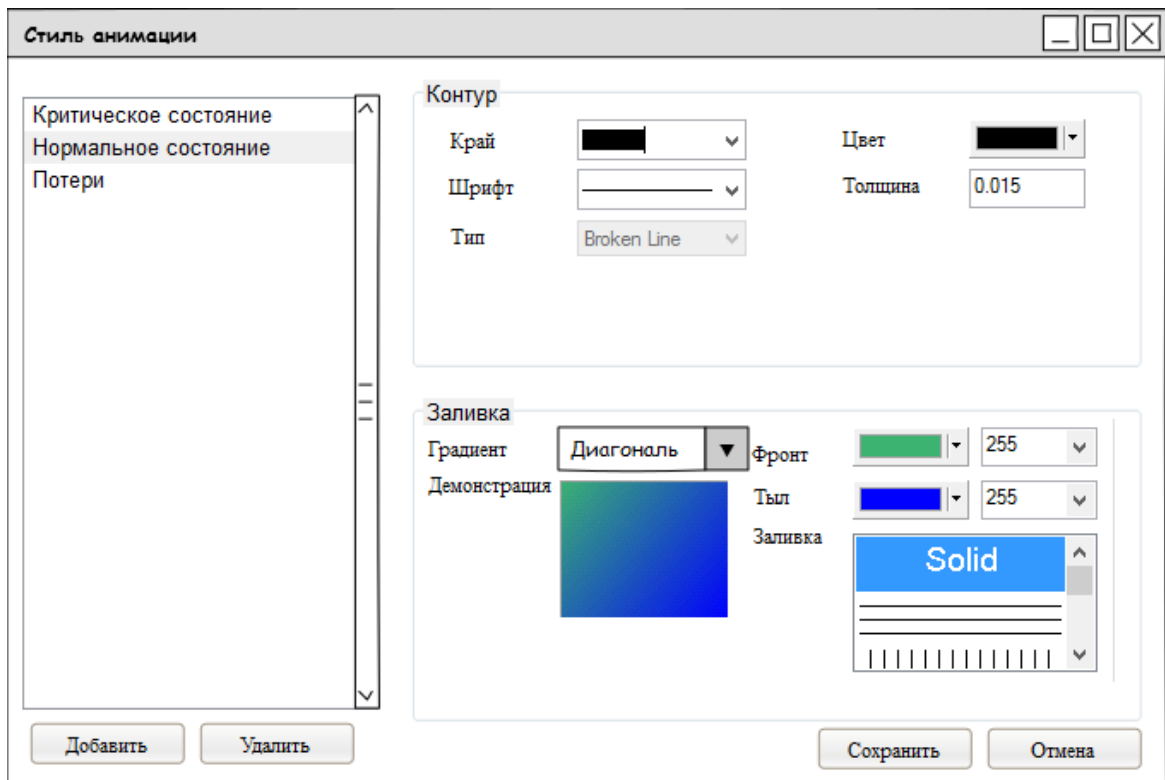


Рис. 21. Эскиз окна настроек стилей анимации

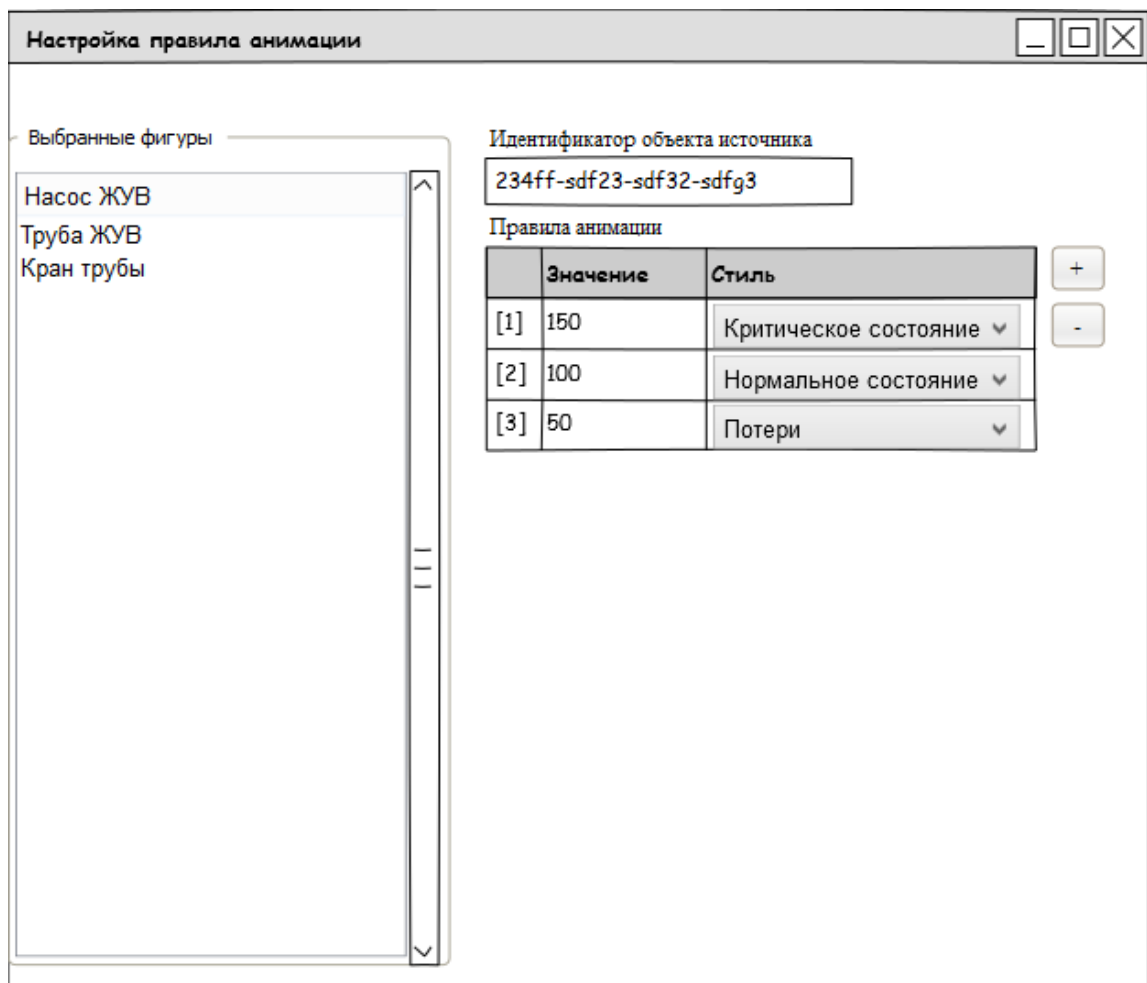


Рис. 22. Эскиз окна настроек правил анимации

Логика демонстрации и анимации мнемосхем заложена в обозревателе. Поэтому обозреватель наравне с графической библиотекой включается в проект редактора мнемосхем. Однако, редактор мнемосхем обладает более расширенными функциональными возможностями, чем обозреватель мнемосхем. Таким образом, функциональные возможности редактора мнемосхем являются в своём роде надстройкой над обозревателем мнемосхем. Далее будет выполнено описание проектных решений в реализации функциональных возможностей редактора мнемосхем. Информация из данного раздела также выполнена на английском языке (см. приложение А).

2.5.1.1 Демонстрация мнемосхем

2.5.1.1.1 Отображение мнемосхемы в рабочей области

Редактор позволяет отображать текущую мнемосхему в рабочей области. Отображение мнемосхемы подразумевает отображение её контента, а также отображение системных объектов, содержащихся на слое подложке. Подробнее об отображении мнемосхемы описано в проектных решениях к обозревателю мнемосхем.

2.5.1.1.2 Выбор мнемосхемы

В зависимости от пользовательской команды редактор позволяет выбрать мнемосхему для работы. Помимо заложенных в обозреватель функций открытия мнемосхемы из файла, открытия опубликованной мнемосхемы, активации мнемосхемы среди текущих, закрытия мнемосхемы, редактор позволяет создать новую мнемосхему.

2.5.1.1.2.1 Создание новой мнемосхемы

Редактор позволяет создать мнемосхему. Созданная новая мнемосхема содержит в себе минимум два слоя: рабочий и косметический. Новая мнемосхема становится активной после создания.

2.5.1.1.2.2 Открытие мнемосхемы из файла

Редактор позволяет открыть мнемосхему из файла формата XML. Особенности открытия мнемосхемы описаны в проектных решениях к обозревателю мнемосхем.

2.5.1.1.2.3 Открытие опубликованной мнемосхемы с сервера

Редактор позволяет открыть мнемосхему, опубликованную на сервере. Особенности открытия мнемосхемы опубликованной на сервере описаны в проектных решения к обозревателю мнемосхем.

2.5.1.1.2.4 Активация мнемосхемы среди открытых мнемосхем

Редактор позволяет выполнить активацию мнемосхемы в пулле мнемосхем. Особенности активации мнемосхемы описаны в проектных решения к обозревателю мнемосхем.

2.5.1.1.2.5 Закрытие текущей мнемосхемы

Редактор позволяет закрыть одну из мнемосхем в пулле. Особенности закрытия мнемосхемы описаны в проектных решения к обозревателю мнемосхем.

2.5.1.1.3 Сохранение мнемосхемы

Сохранение мнемосхемы осуществляется при подаче пользовательских команд сохранения мнемосхемы в файл или публикация мнемосхемы на сервере.

2.5.1.1.3.1 Сохранение мнемосхемы в файл

Редактор позволяет выполнить выгрузку в файл контента мнемосхемы. Для выполнения выгрузки библиотеке подаётся путь сброса файла формата XML, в который выполняется сохранение мнемосхемы.

2.5.1.1.3.2 Публикация мнемосхемы на сервере

Редактор позволяет опубликовать мнемосхему на сервер. Особенности публикации мнемосхемы описаны в проектных решения к обозревателю мнемосхем.

2.5.1.1.4 Изменение параметров активной мнемосхемы

Изменение параметров активной мнемосхемы осуществляется после подачи пользовательских команд: изменить название мнемосхемы, изменить масштаб мнемосхемы, сдвиг мнемосхемы.

2.5.1.1.4.1 Изменение названия мнемосхемы

Редактор позволяет изменить название мнемосхемы. Особенности изменения названия мнемосхемы описаны в проектных решения к обозревателю мнемосхем.

2.5.1.1.4.2 Изменение масштаба мнемосхемы

Редактор позволяет изменить масштаб мнемосхемы в рабочей области. Особенности публикации мнемосхемы описаны в проектных решения к обозревателю мнемосхем.

2.5.1.1.4.3 Сдвиг мнемосхемы

Редактор позволяет изменить центр мнемосхемы в пределах координатной плоскости рабочей области. Особенности публикации мнемосхемы описаны в проектных решения к обозревателю мнемосхем.

2.5.1.2 Модификация мнемосхемы

2.5.1.2.1 Управление коллекцией фигур мнемосхем

Коллекция фигур мнемосхем представляет собой библиотеку, в которой доступно хранение готовых пользовательских групповых фигур с целью последующего многократного использования. При первом выполнении управления коллекцией фигур мнемосхем после запуска редактора, в который встроена библиотека, выполняется наполнение пулла коллекции из файла формата XML, хранящегося в предопределённой системной папке. При завершении работы в редакторе, компонента выполняет выгрузку пулла коллекции в файл формата XML, хранящегося в предопределённой системной папке. Управление коллекцией фигур осуществляется при подаче пользовательских команд: добавить групповую фигуру на мнемосхему из коллекции, создать групповую фигуру на мнемосхеме, выбрать групповую фигуру в коллекции, удалить групповую фигуру из коллекции.

2.5.1.2.1.1 Добавление групповую фигуру в коллекцию

Редактор позволяет добавить групповую фигуру в коллекцию. Наличие в коллекции фигур с одинаковыми названиями не допускается.

2.5.1.2.1.2 Удаление групповую фигуру из коллекции

Редактор позволяет удалить выбранную фигуру из пулла коллекции.

2.5.1.2.1.3 Выбор групповую фигуру в коллекции

Редактор позволяет выбрать фигуру в коллекции, при этом доступен выбор только одной фигуры.

2.5.1.2.2 Управление наполнением контента мнемосхемы

Управление наполнением контента мнемосхемы осуществляется, если поступили пользовательские команды: добавить фигуру на мнемосхему из коллекции, создать фигуру на мнемосхеме, удалить фигуру на мнемосхеме, создать групповую фигуру из выбранных фигур, разделить групповую фигуру.

2.5.1.2.2.1 Добавление групповую фигуру на мнемосхему из коллекции

Редактор позволяет добавить выделенную групповую фигуру на мнемосхему из коллекции. При добавлении фигуры из коллекции на мнемосхему добавляется фигура типа «Группа».

2.5.1.2.2.2 Создание фигуру на мнемосхеме

Редактор позволяет создать на мнемосхеме фигуру из набора графических примитивов. Графическими примитивами являются (см. рис. 23): точка, полилиния, полигон, прямоугольник, дуга, текст. При создании фигура помещается на верхний слой.

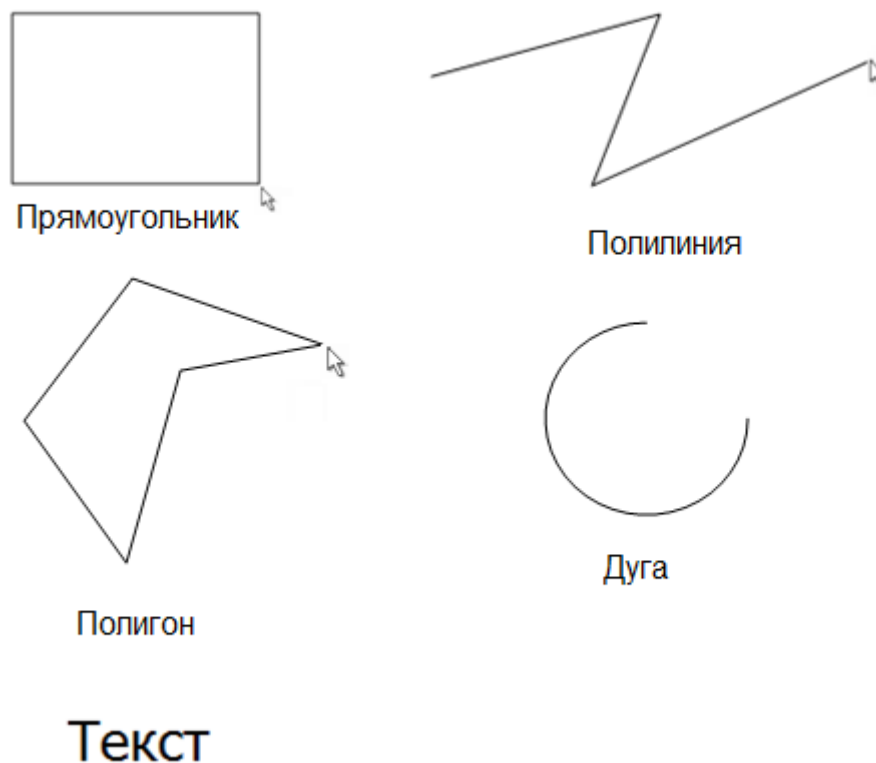


Рис. 23. Графические примитивы

2.5.1.2.2.3 Удаление фигуры на мнемосхеме

Редактор позволяет выполнять удаление фигуры выделенной фигуры на мнемосхеме. При удалении фигуры удаляются ссылки на неё в пуллах: выборка фигур на схеме, пулл связанных фигур. Редактор позволяет удалить несколько фигур, используя выборку фигур на мнемосхеме. При групповом удалении выполняется поочерёдное удаление фигур, находящихся в выборке мнемосхемы.

2.5.1.2.2.4 Создание групповой фигуры из выбранных фигур

Редактор позволяет создать групповую фигуру на мнемосхеме. Групповая фигура создаётся из выбранных фигур на мнемосхеме. Компонента позволяет создавать групповую фигуру только для фигур, содержащихся на одном слое. Если часть фигур выборки находятся на другом слое, то система выполняется перемещение меньшую часть этих фигур на слой, в котором расположена большая часть этих фигур. После создания групповая фигура представляет собой единую фигуру. Выбор фигур включённых в группу приводит к выбору самой групповой фигуры. Изменение геометрических свойств групповой фигуры влечёт изменение этих свойств у фигур, включённых в эту группу.

2.5.1.2.2.5 Разделение групповой фигуры

Редактор позволяет выполнить разделение групповой фигуры. При разделении выполняется удаление групповой фигуры, а все включённые фигуры становятся самостоятельными. После разделения группы доступен выбор отдельно каждой фигуры, ранее включённой в группу. Геометрические свойства группы распространяются на все включённые фигуры в группу.

2.5.1.2.3 Управление содержимым контента мнемосхем

Управление содержимым контента мнемосхемы выполняется, если поступила пользовательская команда: изменить геометрию фигуры на мнемосхеме, изменить стиль оформления фигуры, просмотр и изменение атрибутивной информации фигур.

2.5.1.2.3.1 Изменение геометрии фигуры на мнемосхеме

Редактор позволяет изменять геометрию фигур на мнемосхеме. Изменение геометрии включает в себя: изменение угла поворота фигуры относительно её центра (для фигур: точка, полилиния, полигон, прямоугольник, дуга, текст, группа); для группы фигур центр выбирается согласно центру области занимаемой всеми фигурами группы на рабочей плоскости; изменение ширины,

длины, точки левого края занимаемой области (для фигур полигон, прямоугольник, дуга, текст); места расположения точки центра фигуры (для фигур: точка, полилиния, полигон, прямоугольник, дуга, текст, группа); положения узлов (для полилиния, полигон, дуга, точка); тип геометрии полилиний и полигонов (ломанная, кривая Безье, сплайн); тип дуги (дуга, сектор, сегмент).

2.5.1.2.3.2 Изменение стиля оформления фигуры

Редактор позволяет изменить стиль оформления фигуры. Для площадных объектов доступно изменение: цвет линии границы, толщина линии границы, тип концов границы, цвет сплошной заливки, цвет градиентной заливки. Для линейных объектов доступно следующее изменение стиля: цвет линии границы, толщина линии границы, тип концов границы. Площадными объектами являются: полигон, прямоугольник, дуга, текст. Линейными объектами являются: точка, полилиния

2.5.1.2.3.3 Просмотр и изменение атрибутивной информации фигур

Редактор позволяет изменить атрибутивную информацию фигур. Атрибутивная информация содержится в метаданных фигур мнемосхем.

2.5.1.2.4 Действия над контентом мнемосхем

Действие над контентом пользователя выполняется при поступлении пользовательских команд: изменить порядок фигуры на слое, изменить выборку фигур на мнемосхеме, изменить видимость фигуры, переместить фигуру на другой слой.

2.5.1.2.4.1 Изменение порядка фигур на слое

Редактор позволяет менять порядок фигур на слое, как следствие меняется их взаимная видимость на рабочей области.

2.5.1.2.4.2 Изменение выборки фигур на мнемосхеме

Редактор позволяет выполнять выборку фигур на мнемосхеме. Выборка фигур выполняется со всех слоёв. При повторном выборе фигуры она удаётся из выборки.

2.5.1.2.4.3 Изменение видимости фигур

Редактор позволяет изменять видимость фигур. Фигуры, у которых видимость отключена, не отображаются на рабочей области.

2.5.1.2.4.4 Перемещение фигуры на другой слой

Редактор позволяет перемещать фигуры на другой слой.

2.5.1.2.5 Управление слоями мнемосхемы

По умолчанию мнемосхема имеет минимум два слоя: рабочий и подложку. Рабочий слой предназначен для создания фигур на мнемосхеме. Косметический слой предназначен для создания объектов системы. Объектами системы являются рамки выделения выборки фигур, узлы границ объектов. Управление слоями мнемосхемы должно осуществляться при поступлении пользовательских команд: добавить новый слой к мнемосхеме, удалить выбранный слой на мнемосхеме, изменить порядок слоёв мнемосхемы, изменить видимость слоя на мнемосхеме.

2.5.1.2.5.1 Добавление нового слоя к мнемосхеме

Редактор позволяет добавлять дополнительный рабочий слой к мнемосхеме. Количество рабочих слоёв не ограничено. Каждый добавляемый рабочий слой помещается на последнее место в пуле слоёв.

2.5.1.2.5.2 Удаление выбранного слоя на мнемосхеме

Редактор позволяет удалять выбранный слой мнемосхемы. При удалении выбранного слоя мнемосхемы выполняется удаление всех фигур слоя. Удаление выбранного слоя выполняется со смещением всех нижестоящих слоёв в пуле на позицию вверх.

2.5.1.2.5.3 Изменение порядка слоёв мнемосхемы

Редактор позволяет менять порядок слоёв на мнемосхеме. Самый верхний слой можно переместить только вниз. Самый нижний слой можно переместить только вверх. Серединные слои могут быть перемещены как вверх, так и вниз.

2.5.1.2.5.4 Изменение видимости слоя на мнемосхеме

Редактор позволяет менять видимость слоя на схеме. Фигуры, содержащиеся на невидимом слое, не отображаются на рабочей области.

2.5.1.3 Определение правил анимации фигур мнемосхем

2.5.1.3.1 Изменение в параметрах фигур идентификатора объекта источника данных

Редактор позволяет изменить идентификатор объекта источника для выбранных фигур на мнемосхеме. Для удаления идентификатора объекта источника для выбранных фигур на мнемосхеме необходимо, передаётся пустой идентификатор. Изменение в параметрах фигур объекта источника выполняется при поступлении пользовательской команды изменить идентификатор объекта источника данных фигуры.

2.5.1.3.2 Выбор правила анимации

Редактор позволяет выбрать правила для анимации для выбранных фигур на мнемосхеме. Для выбора правил анимации компонента позволяет сопоставить значение объекта источника данных фигуры со стилем анимации отображения фигуры на схеме. Редактор позволяет настроить не ограниченное количество правил для фигуры мнемосхемы. Выбор правила анимации осуществляется при поступлении пользовательских команд: управление пуллом связанных фигур, добавить фигуру в пулл связанных фигур, удалить фигуру из пулла связанных фигур.

2.5.1.3.3 Управление пуллом связанных фигур

При открытии мнемосхемы редактор автоматически формирует пулл связанных фигур мнемосхемы обходом всего контента мнемосхемы с чтением информации идентификатора объекта источника данных у фигур мнемосхем.

2.5.1.3.3.1 Добавление фигуры в пулл связанных фигур

Редактор позволяет автоматически добавляет фигуру в пулл связанных фигур после добавления к фигуре идентификатора источника данных.

2.5.1.3.3.2 Удаление фигуры из пулла связанных фигур

Редактор позволяет автоматически удаляет фигуру из пулла связанных фигур после добавление к фигуре пустого идентификатора источника данных.

2.5.1.4 Обновление данных объектов мнемосхем

Обновление данных будет выполняться автоматически после вызова пользовательской команды включения обновления данных. При повторном вызове команды процедура обновления прекращается.

2.5.1.4.1 Получение обновлённых данных от источника данных

Редактор будет выполнять запрос обновления к источнику данных по идентификаторам объектов источников данных из пулла связанных фигур. Редактор позволяет выполнить получение обновлённых данных от объекта источника данных. При получении данных выполняется верификация.

2.5.1.4.1.1 Подключение к источнику данных

Редактор будет выполнять подключение к источнику данных по данным строки подключения. В случае если подключение не установлено, выводится ошибка.

2.5.1.4.1.2 Получение обновлённых данных

Редактор будет выполнять получение обновлённых данных от источника данных по идентификаторам пулла связанных фигур.

2.5.1.4.1.3 Верификация полученных данных

Редактор будет выполнять верификацию полученных данных. Если разница между временными данными объекта источника данных и данных фигуры пулла связанных фигур больше, чем интервал обновления, то данные считать недостоверными.

2.5.1.4.2 Обновление данных каждой фигуры пулла связанных фигур

Редактор будет выполнять обновление данных фигур, находящихся в пулле связанных фигур, на основе пришедших данных от объекта источника.

2.5.1.5 Работа с репозиторием стилей мнемосхем

2.5.1.5.1 Управление наполнением пула стилей анимации

Управление наполнением пулла стилей анимации должно выполняться при поступлении пользовательских команд: добавить новый стиль в пулл стилей анимации. Удалить выбранный стиль из пулла стилей анимации, выбрать стиль.

2.5.1.5.1.1 Добавление нового стиля в пулл стилей анимации

Редактор позволяет добавить новый стиль в пулл стилей анимации. Редактор запретит добавление стиля, если стиль с таким именем уже есть в репозитории.

2.5.1.5.1.2 Удаление выбранного стиля из пулла стилей анимации

Редактор позволяет удалить выбранный стиль из пулла стилей.

2.5.1.5.1.3 Загрузка пулла стилей из файла

При первом выполнении управления репозиторием стилей мнемосхем после запуска редактора выполняется наполнение пулла стилей из файла формата XML, хранящегося в предопределённой системной папке.

2.5.1.5.1.4 Выгрузка пулла стилей в файл

При завершении работы в системе редактор будет выгружать пулл стилей в файл формата XML, хранящегося в предопределённой системной папке.

2.5.1.5.1.5 Выбор стиля

Редактор позволяет выбрать стиль в репозитории стилей мнемосхем. Доступен выбор только одного стиля в репозитории.

2.5.1.5.2 Управление содержимым стиля анимации

Редактор позволяет изменять атрибуты выбранного стиля. Под атрибутами понимаются настройки стиля, которые изменяются у фигуры в зависимости от правила анимации. Управление содержимым стиля анимации выполняется при поступлении пользовательской команды на управление содержимым стиля анимации.

2.5.2 Программная реализация редактора мнемосхем

Программная реализация редактора мнемосхем заключалась в обеспечении возможности создания и редактирования мнемосхем на основе векторной графической библиотеки и обозревателя мнемосхем.

В проект редактора мнемосхем были включены графическая библиотека и обозреватель. Все методы, связанные с отображением и анимацией мнемосхем полностью реализуются средствами обозревателя.

Разработанный редактор мнемосхем имеет главное окно (см. рис. 24), которое содержит рабочую зону, панель объектов мнемосхемы, панель коллекций фигур, панель инструментов, главное меню и строку состояния. Редактор позволяет выполнить просмотр и редактирование атрибутивной информации фигуры (см. рис. 25), а также выполнить настройку стиля отображения фигуры (см. рис. 26). Для решения задач обеспечения анимации фигур редактор позволяет настроить коллекцию стилей анимации фигур (см. рис. 27) и коллекцию правил анимации фигур (см. рис. 28).

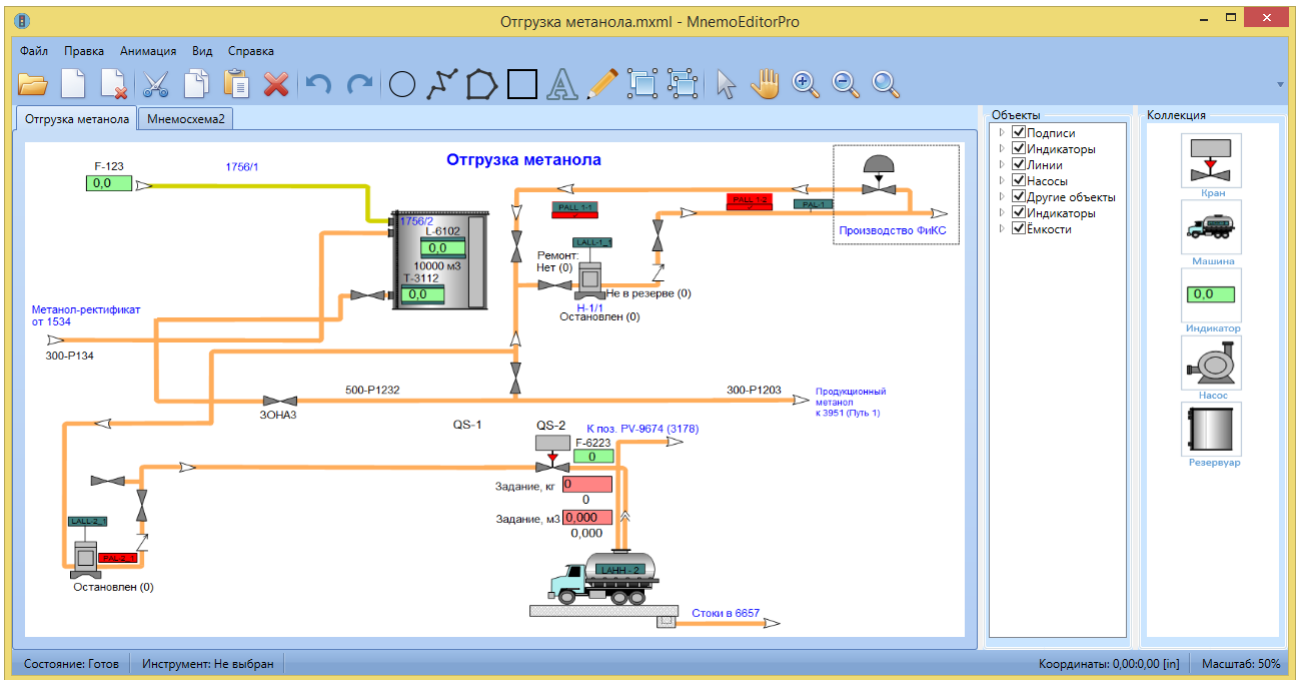


Рис. 24. Главное окно

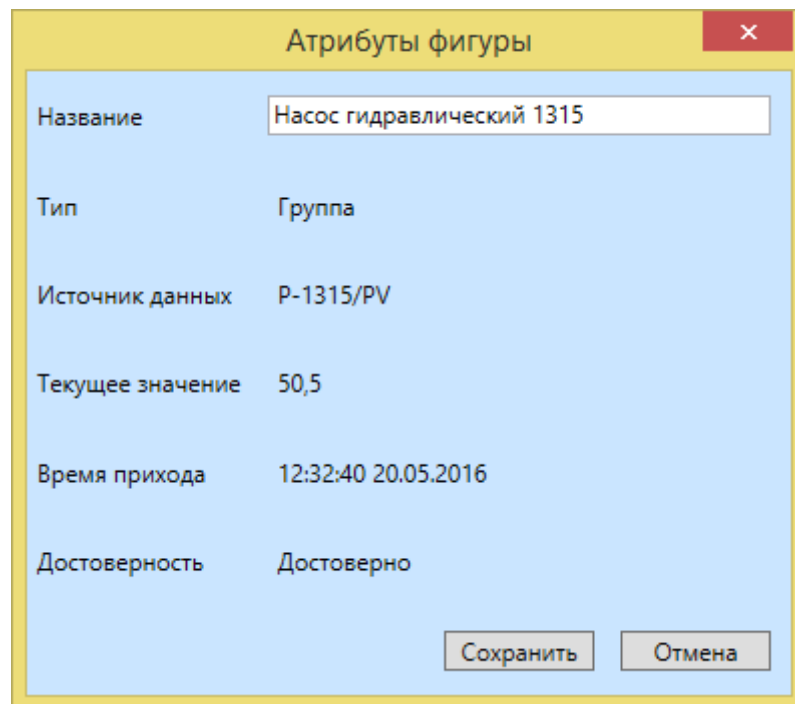


Рис. 25. Окно атрибуты фигуры

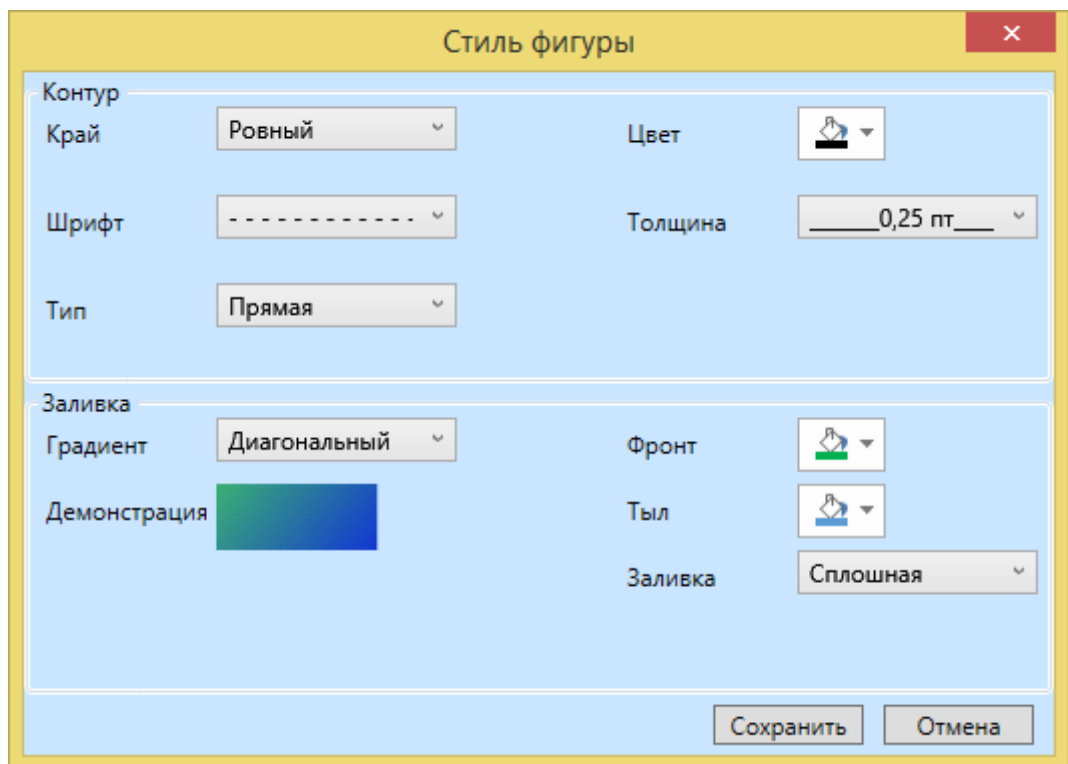


Рис. 26. Окно стиль фигуры

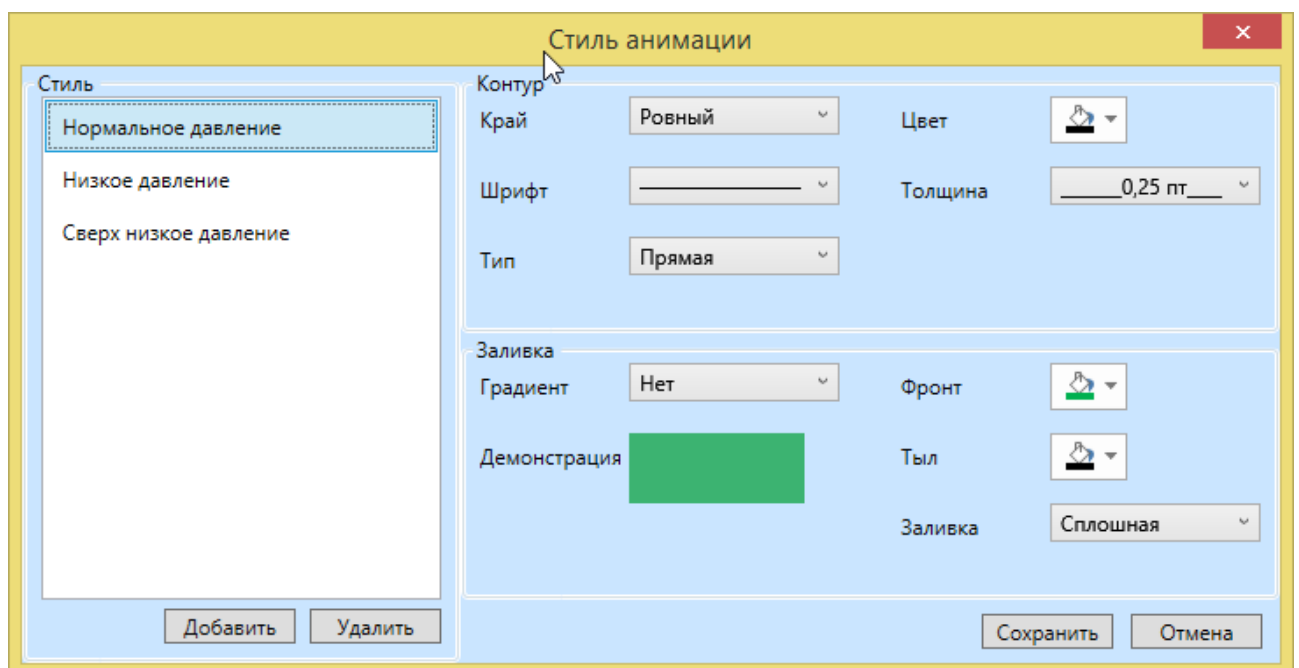


Рис. 27. Окно стиль анимации

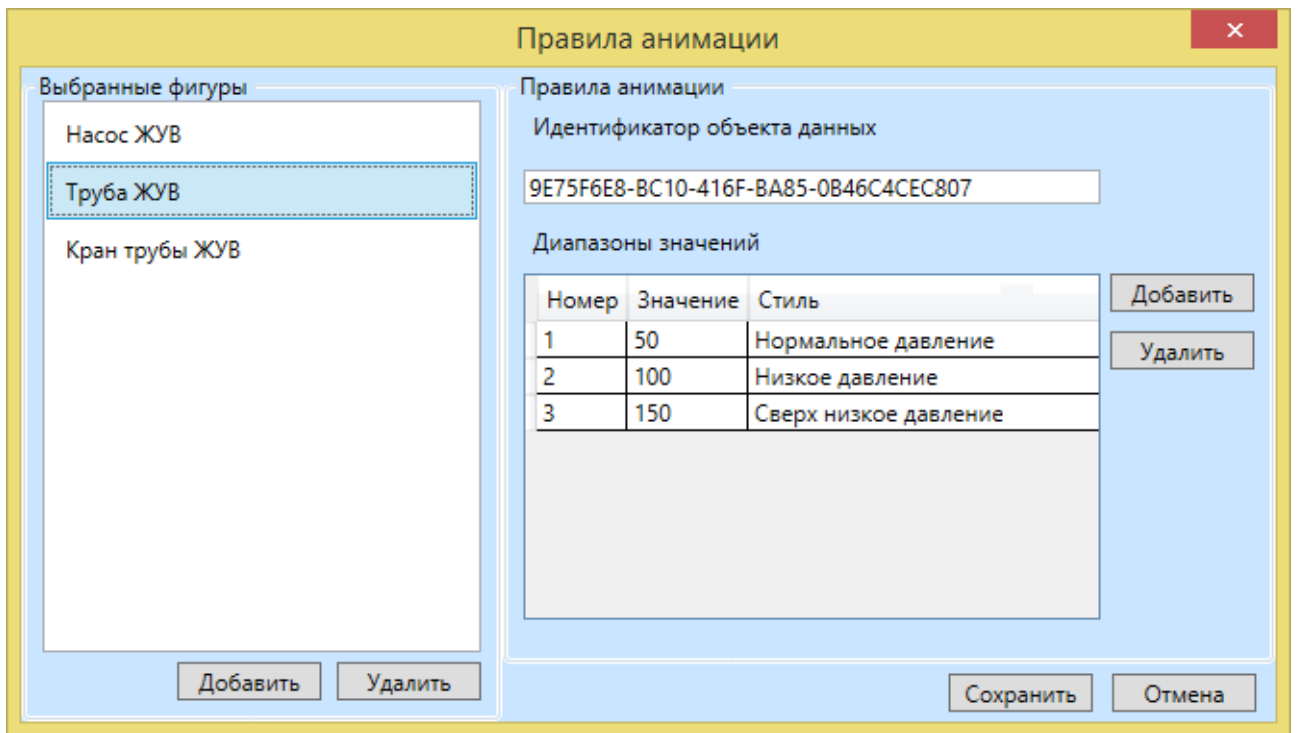


Рис. 28. Окно правила анимации

2.5.2.1 Демонстрация мнемосхем

Задачи отображение мнемосхем в полной мере обеспечиваются обозревателем мнемосхем. Однако редактор добавляет возможность сохранения мнемосхемы в файл и создания новой мнемосхемы. Это обеспечивается методами `Mnemonic.Save()` и `Mnemonic.New()`. При этом при создании мнемосхемы открывается пустая мнемосхема в рабочей области, которая содержит один рабочий слой. При сохранении мнемосхемы пользователь указывает папку сброса и имя мнемосхемы. Мнемосхема сохраняется в формате XML файла.

2.5.2.2 Модификация мнемосхемы

Возможность создания фигур обеспечивается векторной графической библиотекой. Поэтому в редакторе мнемосхем реализованы лишь инструментальные методы для создания фигур. Инструменты доступны пользователю из панели инструментов в пользовательском интерфейсе редактора. При активации пользователем инструмента, в редакторе изменяется поле `CurrentTool`. При выполнении какого либо действия в рабочей области вызывается метод обработки событий рабочей области `Events()`. В данном методе определяется текущий инструмент и вызывается метод для действия конкретным текущим инструментом. Кроме того, в редакторе реализована возможность

создания фигуры из коллекции. Пользовательский интерфейс редактора содержит панель с коллекцией пользовательских фигур. Выполнение перемещение фигур из панели коллекции на рабочую область выполняется методом `DragAndDrop()`. При этом в структурах объектов мнемосхемы создаётся групповой объект. Реализованы функциональные возможности создания групповых фигур. Группируются фигуры, находящиеся в выборке `Selection`. При группировке создаётся объект типа `Group`. Данный объект содержит в себе ссылки на включённые в группу фигуры. При этом выбранные фигуры удаляются из выборки и добавляются в неё групповая фигура. При разделении выборки выполняется удаление групповой фигуры со схемы и включённые в неё фигуры становятся отдельными фигурами. Для разгруппировки создан метод `Ungroup()`, который вызывается при вызове команды пользователем. Удаление объектов с мнемосхемы выполняется каскадным методом. Т.е. удаляется не только сам объект, но и все подчинённые к нему объекты. Для удаления фигуры создан метод `RemoveObject()`, который вызывается при выполнении удаления фигуры. Редактор содержит возможность массового удаления. Для этого удаление фигур осуществляется среди фигур выборки. Таким образом, удаляются те фигуры, которые помещены выборку.

Возможность редактирования мнемосхем реализована группой методов. Такие методы задействуют возможности компоненты по созданию выборки объектов мнемосхем. Для этого реализован метод `ChangeSelection()` который вызывает при отлавливании нажатия клавиш мыши в рабочей области. В момент нажатия пользователем клавиши мыши, выполняет отлавливание положения курсора в рабочей области и проверка наличия в данной точке фигур. При выборе фигур всегда выбирается та, что лежит выше на схеме.

При выборе фигур выполняется добавление её в коллекцию выборки, а так же выделение области, занимаемой фигурой, рамкой выделения. Рамка выделения содержит точки редактирования. При воздействии указателем мыши на данные точки выполняется редактирование геометрических свойств фигуры. Для этого реализованы такие методы как `Resize()` и `Rotated()`, для изменения размера и поворота фигуры на плоскости соответственно.

Как писалось выше, редактор имеет механизм создания коллекций пользовательских фигур. В коллекцию доступно добавление только групповых фигур. В общем случае коллекция фигур представляет собой не визуальную мнемосхему, на которой сохранены в отдельных слоях групповые фигуры. При

добавлении групповой фигуры в коллекцию выполняется вызов метода `CollectionControl.Add()`. При его выполнении выполняется автоматическое сохранение коллекции. Коллекция хранится в папке с пользовательскими настройками средствами Windows. При удалении фигуры из коллекции выполняется метод `CollectionControl.Remove()`.

Изменение стиля оформления фигур выполняется с помощью специального окна в интерфейсе пользователя. Изменение стиля фигуры выполняется вызовом метода `ChangeObjectStyle()`, который выполняет изменение всех стилистических свойств фигуры в зависимости от выбранных в окне настроек.

У фигур мнемосхем имеется атрибутивная информация: тип, название, время прихода данных, достоверность. Изменение атрибутивной информации выполняется методом `ChangeAttributeInf()`. При этом пользователю доступно изменение только названия фигуры. Остальная атрибутивная информация изменяется программно.

Редактором доступно изменение порядка фигур на слое. Это обеспечивается методом `ChangeObjectOrder()`. При этом фигура перемещается в перечне объектов мнемосхемы на позицию выше или ниже (в зависимости от команды пользователя). Изменение видимости фигуры доступно только в пределах слоя.

Редактором обеспечивается изменение видимости фигур. Изменение видимости фигур осуществляется методом `ChangeObjectVisible()`, который изменяет свойство видимости у фигуры.

Пользователю доступно перемещение фигуры на другой слой. Для это по пользовательской команде вызывается метод `ChangeObjectLayer()`. При этом фигура помещается выше всех фигур выбранного слоя.

Пользовательский интерфейс редактор имеет панель управления слоями фигур мнемосхемы. Мнемосхем должна содержать минимум один рабочий слой. Кроме того, редактор мнемосхем во время работы добавляет к мнемосхеме рабочий слой, на который помещаются служебные объекты, такие как рамки выделения и т.п. Добавление слоя в коллекцию осуществляется методом `Layers.Add()`, удаление методом `Layers.Remove()`. Редактором доступно изменение порядка слоёв мнемосхемы. Это обеспечивается методом `ChangeLayerOrder()`. При этом слоё перемещается в перечне слоёв мнемосхемы на позицию выше или ниже (в зависимости от команды пользователя).

Редактором обеспечивается изменение видимости слоёв. Изменение видимости слоёв осуществляется методом `ChangeLayerVisible()`, который изменяет свойство видимости у слоя.

2.5.2.3 Работа с репозиторием стилей мнемосхем

Редактор позволяет создать стили оформления фигур мнемосхем. Это позволяет настроить единый стиль оформления для нескольких фигур. Стили хранятся в коллекции стилей, называемой репозиторием. Для настройки стилей оформления фигур мнемосхем редактор содержит специализированный интерфейс. Данный интерфейс позволяет создать стиль и выполнить настройки стиля, которые должна принимать фигура данного стиля. Интерфейс позволяет добавлять стили, при этом вызывается метод `StyleCollection.Add()`. Кроме того, доступно удаление стиля из коллекции, при этом вызывается метод `StyleCollection.Remove()`.

При старте редактора выполняется загрузка стиля из файла вызовом метода `LoadAnimationStyles()`. Данный файл храниться средствами Windows в папке пользовательских настроек. Сам файл имеет структуру XML, в которой сохранены стили пользователя. При завершении редактирования стилей выполняется их выгрузка в файл настроек, методом `SaveAnimationStyles()`.

2.5.2.4 Механизм настройки анимации фигур

Редактор обеспечивает механизм создания правил анимации фигур. Данный механизм позволяет выполнить сопоставление фигуры мнемосхемы идентификатору объекта мнемосхемы. При этом указывается стиль оформления фигуры и значение при котором фигура принимает данный стиль. Добавление стиля анимации выполняется методом `AnimationCollection.Add()`, и удаление методом `AnimationCollection.Remove()`. При этом все настройки анимации сохраняются в структурах данных мнемосхемы.

2.5.2.5 Анимация фигур мнемосхем

Анимация фигур мнемосхем в полной мере обеспечивается средствами обозревателя мнемосхем. Плагин встраивается в редактор мнемосхем и обеспечивает анимацию мнемосхемы при создании мнемосхем. В интерфейсе пользователя указывается сервер и база данных, которой необходимо выполнить подключение.

3 ЗАДАЧИ ИНТЕГРАЦИИ ГРАФИЧЕСКИХ СРЕДСТВ С ВНЕШНИМИ СИСТЕМАМИ

Созданный обозреватель мнемосхем был интегрирован с MES-системой «АРМ Технолога», разрабатываемой ООО «СибХайТекЦентр» для предприятия ООО «СибМетаХим». Обозреватель включён в MES-систему как среда отображения для подсистемы «Мнемосхем».

В процессе интеграции необходимо было решить некоторое количество проблем. Как писалось ранее (см. раздел 1), до настоящего момента в компании ООО «СибХайТекЦентр» для работы с мнемосхемами использовался векторный графический редактор Pro Grapher, который построен на свободно распространяемой компоненте ActiveX ProGrapherControl. На базе данной компоненты был построен «АРМ Технолога» (см. рис. 29), который позволял отображать мнемосхемы созданные в редакторе Pro Grapher.

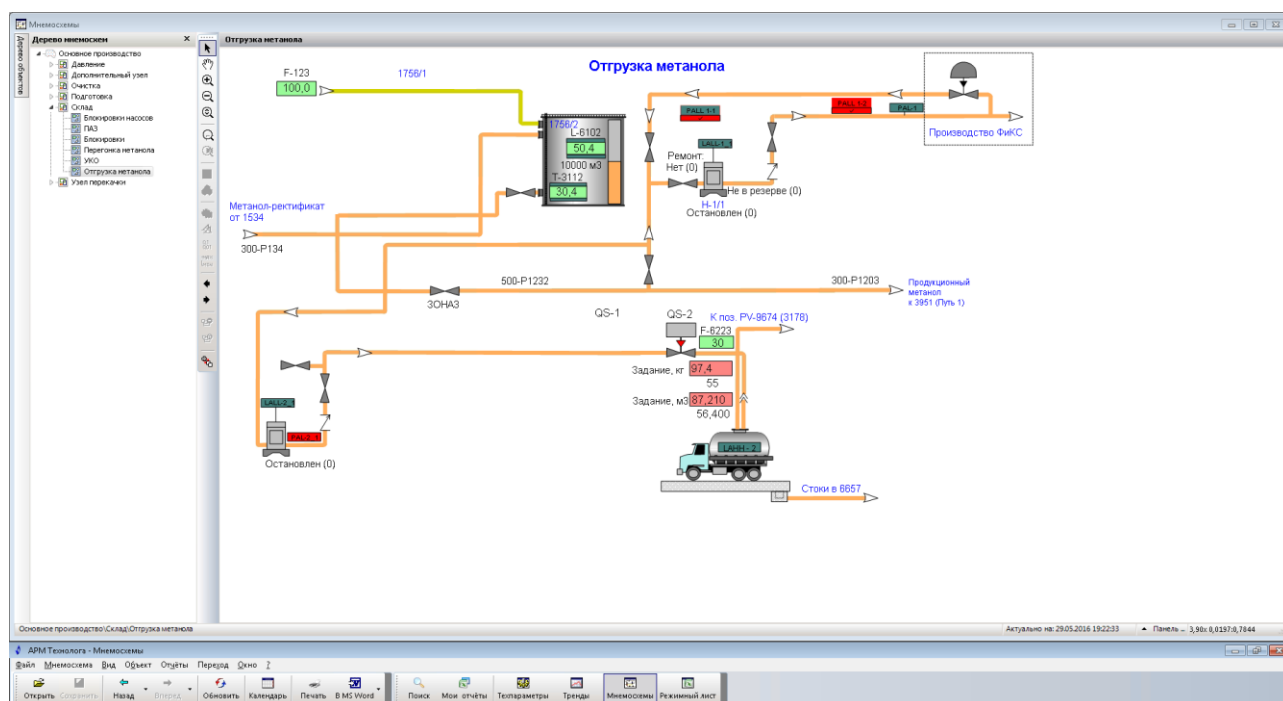
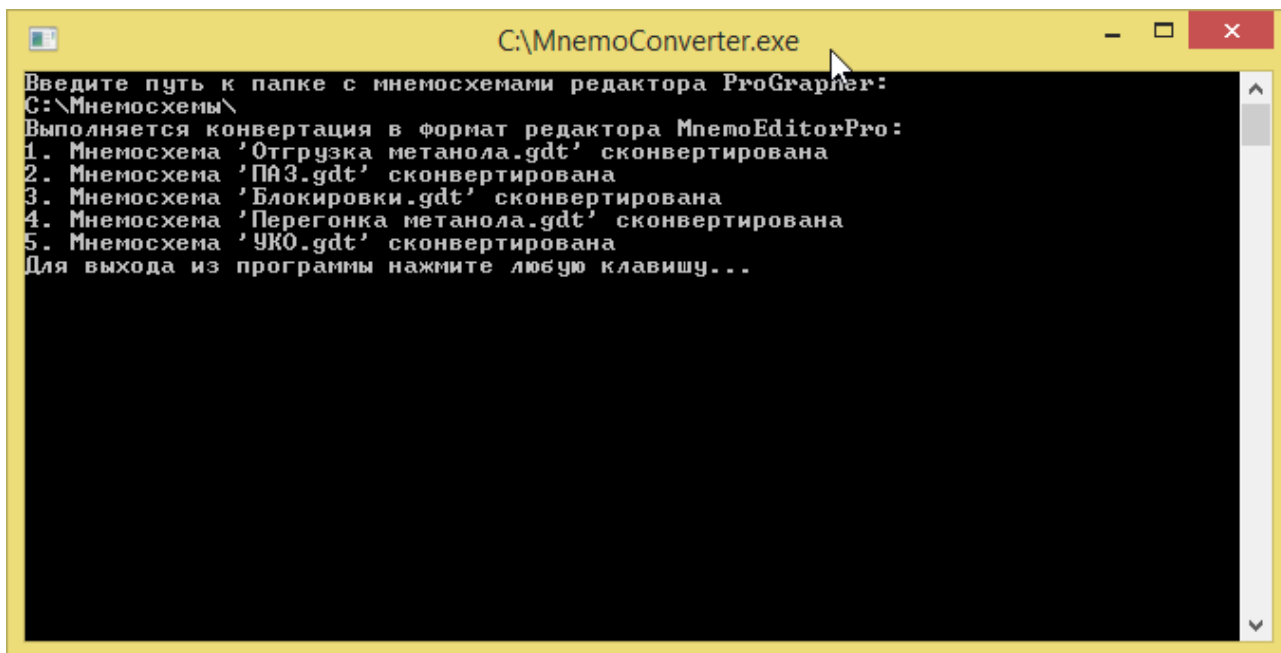


Рис. 29. Интеграция обозревателя в «АРМ Технолога»

В процессе промышленной эксплуатации «АРМ Технолога» ранее было создано порядка 150 мнемосхем. Их полное воссоздание в разработанных графических средствах потребовало бы большое количество ресурсов. Для решения данной задачи был спроектирован и создан специализированный конвертер мнемосхем (см. рис. 30). Данный конвертер построен на базе двух графических библиотек: Mmemo Editor Library и ActiveX Pro Grapher Control.

Данный конвертер позволяет сконвертировать мнемосхемы, созданные в редакторе Pro Grapher, и сохранить их в формате XML в структурах данных редактора MnemoEditorPro.

В процессе конвертации мнемосхем выполняется создание всех фигур мнемосхемы с её свойствами, в том числе привязки к технологическим параметрам.



```
C:\MnemoConverter.exe
Введите путь к папке с мнемосхемами редактора ProGrapher:
C:\Мнемосхемы\
Выполняется конвертация в формат редактора MnemoEditorPro:
1. Мнемосхема 'Отгрузка метанола.gdt' сконвертирована
2. Мнемосхема 'ПАЗ.gdt' сконвертирована
3. Мнемосхема 'Блокировки.gdt' сконвертирована
4. Мнемосхема 'Перегонка метанола.gdt' сконвертирована
5. Мнемосхема 'УКО.gdt' сконвертирована
Для выхода из программы нажмите любую клавишу...
```

Рис. 30. Конвертер мнемосхем

Изменениям также подверглась и база данных для «АРМ Технолога». Ранее, мнемосхемы хранились в таблице в виде бинарного типа данных. В таблицу было добавлено поле с типом данных XML, что позволит загрузить туда новые мнемосхемы. Данная особенность позволит в период опытной эксплуатации существовать двум системам одновременно.

По результатам внедрения программных средств на предприятии ООО «СибХайТекЦентр» был получен акт внедрения (см. приложение Г).

ЗАКЛЮЧЕНИЕ

Магистерская диссертация посвящена решению актуальной задачи разработки графических средств для работы с технологическими мнемосхемами. Работа выполнялась на предприятии ООО «СибХайТекЦентр», занимающейся коммерческой разработкой инженерного программного обеспечения. К разрабатываемому ПО в компании отнесется, в том числе, программного обеспечения для предприятий нефтегазовой и химической отрасли. В таком ПО необходимо применение мнемосхем для моделирования процессов производства.

В работе выполнен подробный анализ предметной области, включающий детальное исследование принципов построения и использования мнемосхем на производственных предприятиях. На каждом этапе работы рассмотрены различные варианты решения возникающих задач, выявлены их достоинства и недостатки, обоснованы принимаемые решения.

В процессе разработки были формализованы требования и составлено техническое задание на разработку. По результатам анализа предметной области и техническому заданию были спроектированы графические средства работы с мнемосхемами.

Особенностью разработки является создание единой графической библиотеки, как основы для создания средств редактирования и использования мнемосхем.

На базе векторной графической библиотеки созданы обозреватель и редактор мнемосхем, которые интегрированы с действующими системами на предприятии, о чем свидетельствует полученный акт внедрения.

В процесс выполнения работы были проработаны вопросы финансового менеджмента, ресурсоэффективности и ресурсосбережения выполненного проекта. Кроме того, были рассмотрены аспекты социальной ответственности данной работы.

В процессе выполнения работы выполнялись выступления с докладами на XII и XIII Международной научно-практической конференции студентов, аспирантов и молодых учёных «Молодежь и современные информационные технологии». По теме работе опубликованы две статьи.

СПИСОК ПУБЛИКАЦИЙ

1. Сергеев Д.А., Мирошниченко Е.А. Специализированный векторный редактор схем // XII Международная научно-практическая конференция студентов, аспирантов и молодых учёных «Молодёжь и современные информационные технологии». — 2014. — №33. — С. 82-83
2. Д.А. Сергеев, Р.В. Ковин; Универсальная векторная графическая компонента для работы с мнемосхемами технологических процессов // Молодежь и современные информационные технологии: сборник трудов XIII Международной научно-практической конференции студентов, аспирантов и молодых ученых (г. Томск, 9-13 ноября 2015 г) / Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2015. – С. 103-104.
3. Ибраева Н.С.; Сергеев Д.А.; Кудинов А.В.; Использование технологий Business Intelligence для анализа данных в сфере государственных закупок // Технологии Microsoft в теории и практике программирования: сборник трудов XII Всероссийской научно-практической конференции студентов, аспирантов и молодых ученых (Томск, 25–26 марта 2015 г.) / Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2015. – 251 с.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. ГОСТ Р 54369-2011. Проектирование, изготовление и введение в эксплуатацию систем управления электрооборудованием для обеспечения технологического процесса судопропуска на вновь вводимых, реконструируемых и подлежащих капитальному ремонту судовых шлюзах. — М.: Стандартинформ, 2012. — 56 с.
2. Сергеев Д.А., Мирошниченко Е.А. Специализированный векторный редактор схем // XII Международная научно-практическая конференция студентов, аспирантов и молодых учёных «Молодёжь и современные информационные технологии». — 2014. — №33. — С. 82-83
3. Марков Н.Г., А.В. Кудинов. MES «Магистраль-Восток» в управлении производством газодобывающих компаний // Энергетика. Энергоснабжение. Экология. — 2013. — №33. — С. 63-68
4. Wondeware Solutionos www.wonderwarepacwest.com [Электронный ресурс].— режим доступа: URL: <https://wonderwarepacwest.com/solutions/#Visualization>, свободный (01.06.16). – Загл. с экрана
5. Adastra products www.adastra.ru [Электронный ресурс].— режим доступа: URL: <http://www.adastra.ru/products/>, свободный (01.06.16). – Загл. с экрана
6. Autodesk products www.autodesk.ru [Электронный ресурс].— режим доступа: URL: <http://www.autodesk.ru/products/autocad/overview>, свободный (01.06.16). – Загл. с экрана
7. CorelDraw Graphics www.coreldraw.com [Электронный ресурс].— режим доступа: URL: <http://www.coreldraw.com/ru/?topNav=ru>, свободный (01.06.16). – Загл. с экрана
8. Unity описание www.unity3d.com [Электронный ресурс].— режим доступа: URL: <https://unity3d.com/ru/unity>, свободный (01.06.16). – Загл. с экрана
9. Мирошниченко Е.А. Технологии программирования: учебное пособие – 2е изд. – Томск: Изд-во ТПУ, 2008. – 124 стр.
10. Scrum. Гибкая разработка ПО.: пер. с английского / Майк Кон — М.: Вильям, 2015. — 576 с.

11. Evaluation of the Most Used Agile Methods (XP, LEAN, SCRUM):. англ. / Bodje N’Kauh Nathan Regis — LAP Lambert Academic Publishing, 2012. — 136 с.
12. Совершенный код.: пер. с английского / Стив Макконнелл — М.: Русская Редакция, 2015. — 896 с. <http://www.ozon.ru/context/detail/id/5508646/>
13. Настройка Team Foundation Server.: пер. с английского / Гордон Биминг — ЭКОМ Паблишерз, 2014. — 88 с.
14. Новый Office www.office.com [Электронный ресурс].— режим доступа: URL: <https://products.office.com/ru-ru/home>, свободный (01.06.16). — Загл. с экрана
15. Yworks products www.yworks.com [Электронный ресурс].— режим доступа: URL: <http://www.yworks.com/>, свободный (01.06.16). — Загл. с экрана
16. Pencil products www.pencil.evolus.vn [Электронный ресурс].— режим доступа: URL: <http://pencil.evolus.vn/>, свободный (01.06.16). — Загл. с экрана
17. SharePoint. Новые возможности для совместной работы www.products.office.com [Электронный ресурс].— режим доступа: URL: <https://products.office.com/ru-ru/sharepoint/collaboration>, свободный (01.06.16). — Загл. с экрана
18. Microsoft Visual Studio 2105 Community www.microsoft.com [Электронный ресурс].— режим доступа: URL: <https://www.microsoft.com/ru-ru/download/details.aspx?id=48146>, свободный (01.06.16). — Загл. с экрана
19. RAD Studio: Самый быстрый способ разработать кросс-платформенные нативные приложения с гибкой поддержкой облачных сервисов и IoT www.embarcadero.com [Электронный ресурс].— режим доступа: URL: <https://www.embarcadero.com/ru/products/rad-studio>, свободный (01.06.16). — Загл. с экрана
20. RAD Studio: Самый быстрый способ разработать кросс-платформенные нативные приложения с гибкой поддержкой облачных сервисов и IoT www.embarcadero.com [Электронный ресурс].— режим доступа: URL: <https://www.embarcadero.com/ru/products/rad-studio>, свободный (01.06.16). — Загл. с экрана

21. Unit Testing Framework for MS Visual Studio 2015 www.msdn.microsoft.com
[Электронный ресурс].– режим доступа: URL: [https://msdn.microsoft.com/ru-ru/library/ms243147\(vs.80\).aspx](https://msdn.microsoft.com/ru-ru/library/ms243147(vs.80).aspx), свободный (01.06.16). – Загл. с экрана
22. Automated Software Testing www.smartbear.com/ [Электронный ресурс].– режим доступа: URL: <https://smartbear.com/product/testcomplete/overview/>, свободный (01.06.16). – Загл. с экрана
23. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов: пер. с английского / Мэтью Макдональд — М.: Вильямс, 2013. — 1024 с.
24. Язык программирования C# 5.0 и платформа .NET 4.5.: пер. с английского / Эндрю Троелсен — М.: Вильямс, 2015. — 1312 с.
25. C# 5.0 и платформа .NET 4.5 для профессионалов: пер. с английского / Кристиан Нагел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер — М.: Вильямс, 2014. — 1440 с.
26. Приемы объектно-ориентированного проектирования. Паттерны проектирования: пер. с английского / Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влссидес — П.: Питер, 2016. — 366 с.
27. Белов С.В. Безопасность жизнедеятельности и защита окружающей среды: учебник для вузов. – М.: Изд-во Юрайт, 2013. – 671с.
28. СанПиН 2.2.4.1294-03 «Гигиенические требования к аэроионному составу воздуха производственных и общественных помещений»
29. СанПиН 2.2.2/2.4.1340 – 03 «Гигиенические требования к персональным электронно–вычислительным машинам и организации работы». – М.: Госкомсанэпиднадзор, 2003.
30. СН 2.2.4/2.1.8.562 – 96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории застройки.
31. ГОСТ 12.1.030 -81. Защитное заземление, зануление.
32. СНиП 21 – 01 – 97. Пожарная безопасность зданий и сооружений. М.: Гострой России, 1997. – с.12.
33. Федеральный закон Российской Федерации от 22 июля 2008 г. N 123-ФЗ "Технический регламент о требованиях пожарной безопасности".

34. Гусельников М.Э., Методические указания по разработке раздела «Производственная и экологическая безопасность» выпускной квалификационной работы для студентов всех форм обучения.
35. Козлитин А.М., Яковлев Б.Н. Чрезвычайные ситуации техногенного характера, Учеб. / Под. ред. А.И.Попова, Саратов: Сар. гос. тех. ун-т, 2000. – 124с.
36. СанПиН 2.2.4.548 – 96. Гигиенические требования к микроклимату производственных помещений. М.: Минздрав России, 1997.

ПРИЛОЖЕНИЕ А

Разделы на иностранном языке

Раздел 5.4.1

Project solutions for building mnemonic editor

Студент:

Группа	ФИО	Подпись	Дата
8ИМ4А	Сергеев Дмитрий Алексеевич		

Консультант кафедры ВТ:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент каф. ВТ	Мирошниченко Е.А.	к.т.н.		

Консультант – лингвист кафедры иностранных языков ИК:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Старший преподаватель	Шепетовский Д.В.			

A.1 Project solutions for building mnemonic editor

Design works related to a mnemonic editor primarily consist of creation of the front end to allow access to behavior of the mnemonic embedded in vector graphics library.

In the process of design activity, user interface sketches of graphical mnemonic editor have been created. Design works were performed using classical design patterns [7].

User interface of graphical mnemonic editor must include: main window (fig. 31), figure style settings window (fig. 32), figure attributes window (fig. 33), animation style settings window (fig. 34), animation rules settings window (fig. 35).

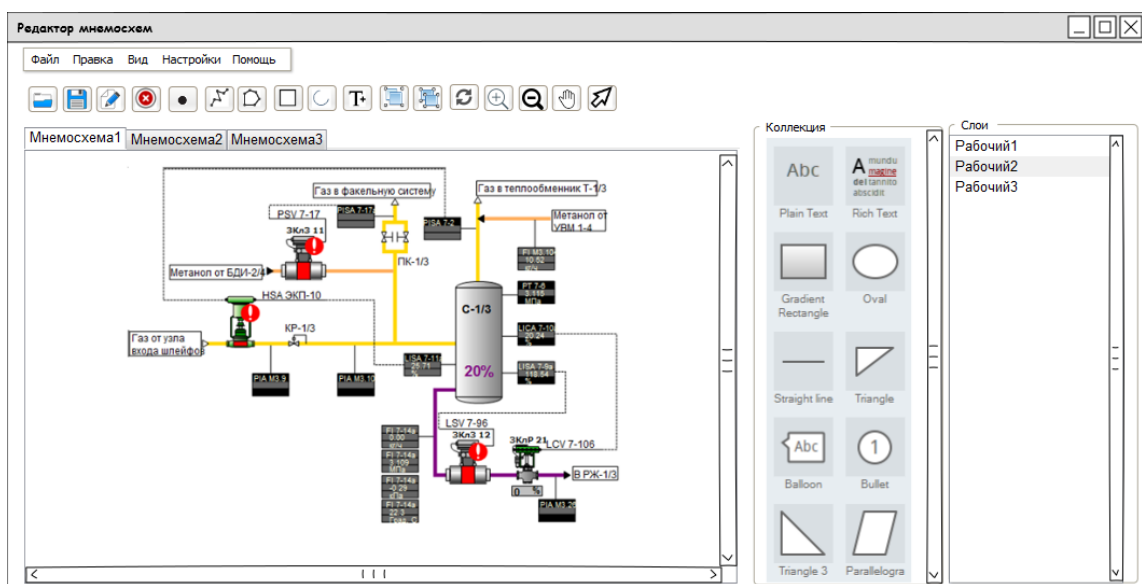


Fig. 31. The main window of mnemonic editor

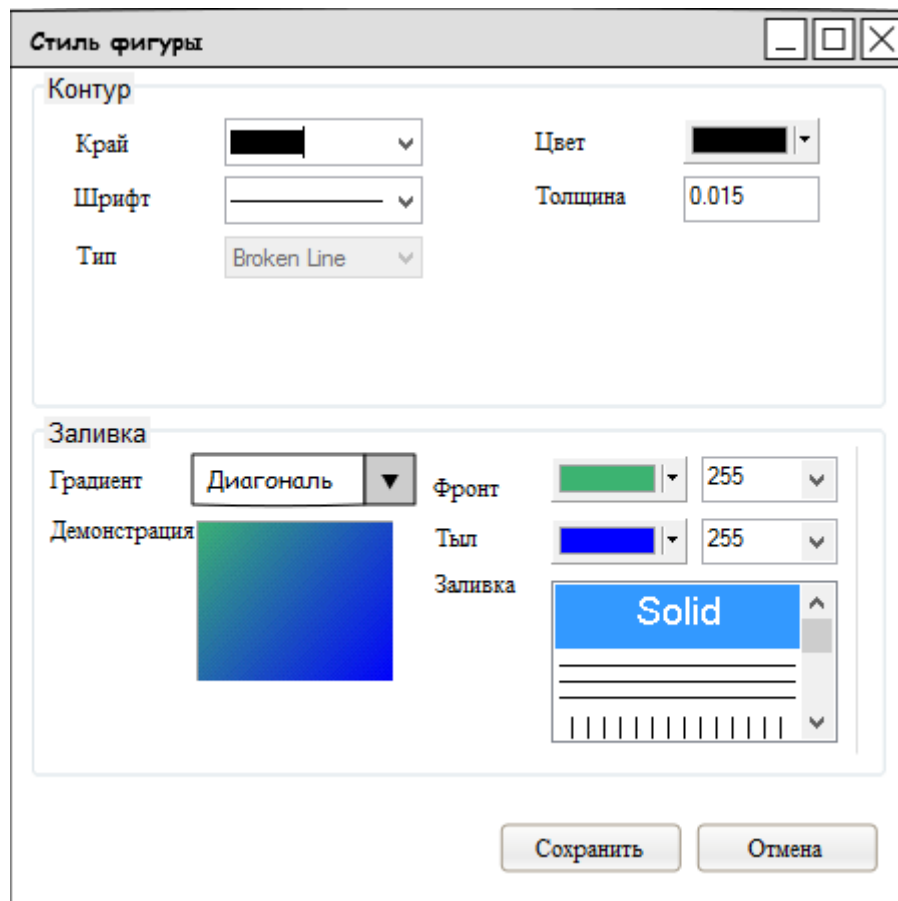


Fig. 32. Figure style settings window

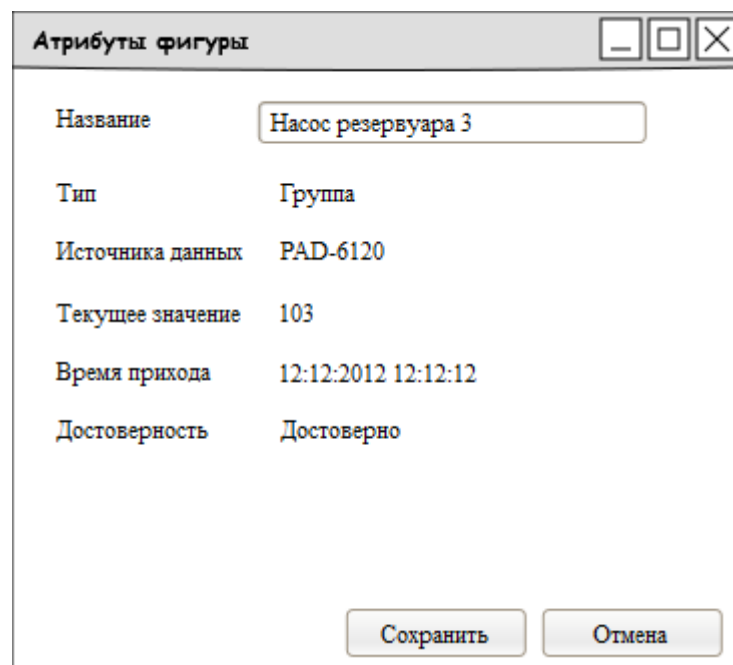


Fig. 33. Figure attributes window

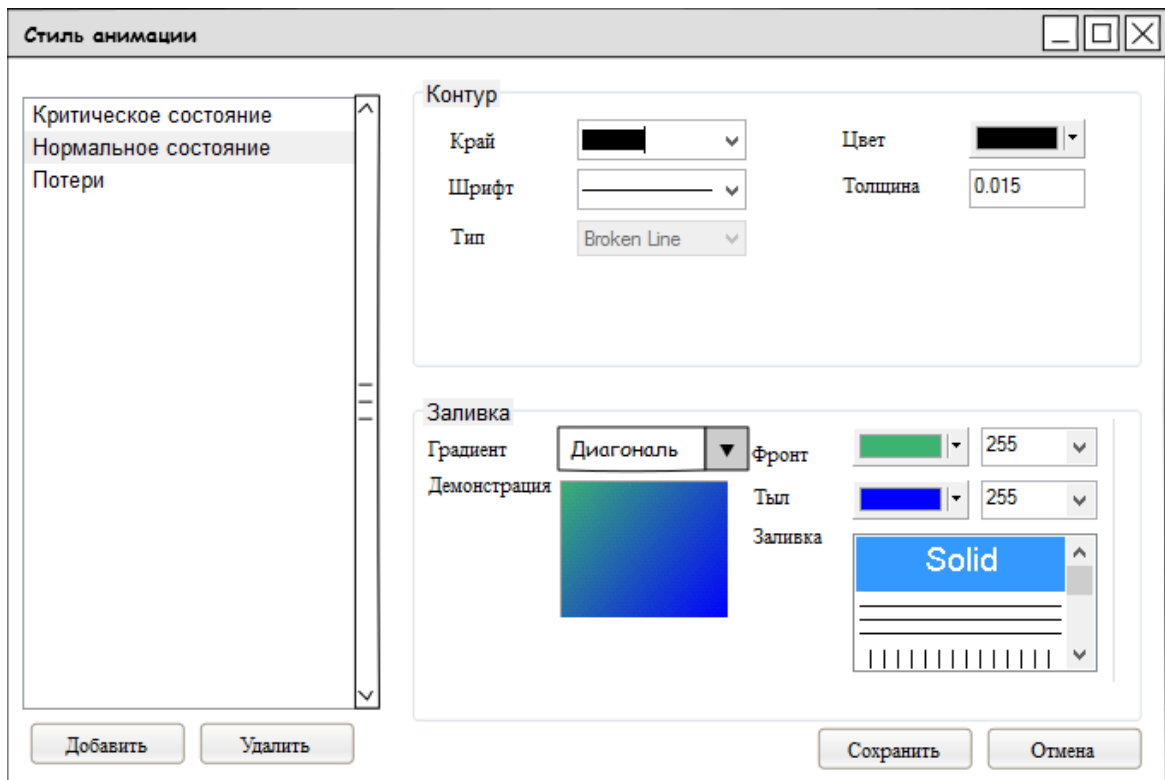


Fig. 34. Animation style settings window

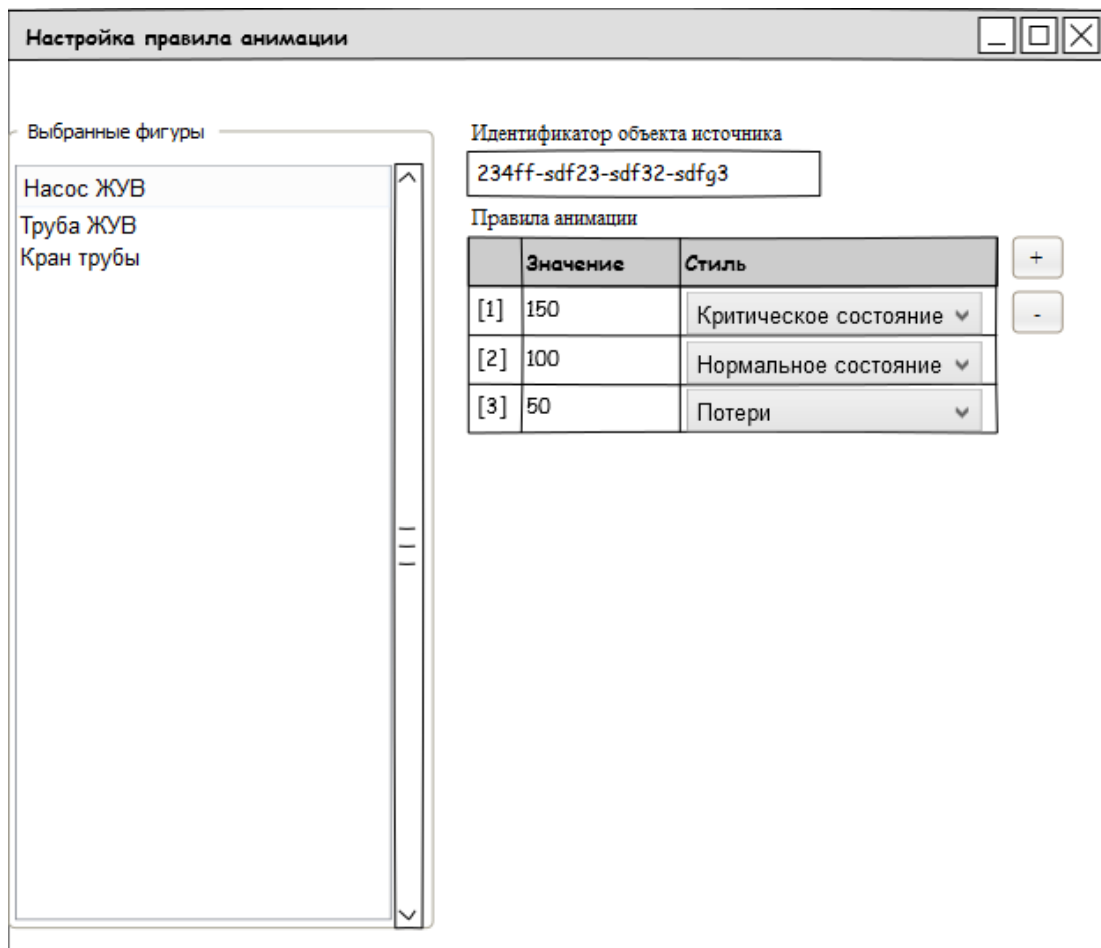


Fig. 35. Animation rules settings window

Further will be description of the mnemonic editor's functionality, which were formulated as a result of requirements analysis for the mnemonic editor.

A.1.1 Mnemonics' demonstration

A.1.1.1 Mnemonic selection

Editor will allow choosing the mnemonic for a work depending on user's command. It is possible to create a new mnemonic, open a mnemonic from a file, activate a current mnemonic, close a current mnemonic.

A.1.1.1.1 Mnemonic creation

The editor will allow creating a mnemonic. Creating a new mnemonic contains at least two layers: working layer and substrate. New mnemonic becomes active after its creation.

A.1.1.1.2 Open a mnemonic from a file

The editor will allow opening a mnemonic in the *.xml file format. A vector graphics library takes the path to file on a hard drive to open a mnemonic file. The open

mnemonic becomes active after the opening. A mnemonic's content becomes displayed in the workspace.

A.1.1.1.3 Activation of mnemonic among opened mnemonic

Editor must have a mnemonic pool for simultaneous operation with several mnemonics. The displayed mnemonic is active in mnemonic pool. Activation of mnemonic in pool must be performed after a user command of mnemonic activation.

A.1.1.1.4 Closing the current mnemonic

The editor will allow closing the mnemonic. If a user closes the mnemonic, it is removed from the mnemonic pool in a system; mnemonic which located at the beginning of the pool becomes active.

A.1.1.1.5 Opening published mnemonic on server

The editor will allow opening the mnemonic published on a server. For this purpose the vector graphics library connects to the server using the connection string. Editor will allow downloading of mnemonic structures in XML format and will make them open in the workplace. Open mnemonic becomes active.

A.1.1.2 Showing mnemonic on a display

The editor will allow displaying the current mnemonic on display in the workplace. Displaying mnemonic involves the showing of its content and displaying system objects located on the substrate layer.

A.1.1.3 Current mnemonic updating

The editor will allow updating the current mnemonic. Update includes redrawing a content in the workplace taking into account the visualization rules of style of figure and data of these figures.

A.1.1.4. Saving the mnemonic

Saving of the mnemonic will be performed when a user sends the command to save a file or publish a mnemonic on the server.

A.1.1.1.4.1 Mnemonic saving to file

The editor will allow uploading a mnemonic content to file. Library receives a path name where the file in *.xml format is sent. Mnemonic is saved to this file.

A.1.1.1.4.2 Publication of mnemonic on the server

The editor will allow to publish a mnemonic on the server using connection string for the server. When the mnemonic is publishing on server, it is uploaded into the database table using XML data format.

A.1.1.5. Changing active mnemonic parameters

Changing active mnemonic parameters will be performed after the following user commands: change name of the mnemonic, change mnemonic scale, move mnemonic.

Changing mnemonic name

The editor will allow changing the mnemonic name. The system allows specifying mnemonic name different from mnemonic file on the hard disk.

Changing mnemonic scale

The editor will allow changing a mnemonic scale in the workspace. In the process of changing mnemonic scale coordinate metrics are modified: it is becoming more detailed with increasing, it is becoming more general with decreasing.

Mnemonic shear

The editor will allow changing mnemonic center within the coordinate space on the workspace.

A.1.2 Mnemonic modification

A.1.2.1 Mnemonic figures collection control

Mnemonics figures collection is a library, which provides storage of ready user group figures for subsequent multiple use. During the first run of the mnemonic figures collection control after the launch of the editor with the integrated library, collection is filled from the collection pool of the files in * .xml format, stored in a predefined system folder. Figures collection control is carried out by applying user commands: add the group figure from collection to the mnemonic, create a group shape on mnemonic, select a group figure in the collection, remove the group figure from collection.

A.1.2.1.1 Adding the group figure to mnemonic to collection

The editor will allow adding a group figure to collection. The presence of collection figures with the same name is not allowed.

A.1.2.1.2 Removing group figure from collection

The editor will allow removing selected figure from collection pull.

A.1.2.1.3 Selecting group figure from collection

The editor will allow selecting a group figure from collection. Only one figure may be selected.

A.1.2.2 Mnemonic content management

Mnemonic content management is performed in the case of incoming user commands: add figure from collection to mnemonic, create figure on mnemonic, remove figure on mnemonic, add group figure from collection to mnemonic, create group figure from selected figures, ungroup a group figure.

A.1.2.2.1 Adding group figure from collection to mnemonic

The editor will allow adding a group figure from collection. At adding a group figure from collection figure with type group adds on mnemonic.

A.1.2.2.2 Figure Creation on mnemonic

The editor will allow creating a figure on a mnemonic from the set of graphic primitives. Graphic primitives are (see fig. 36): point, polyline, polygon, rectangle, arc, text. At creating a figure is placed on the top layer.

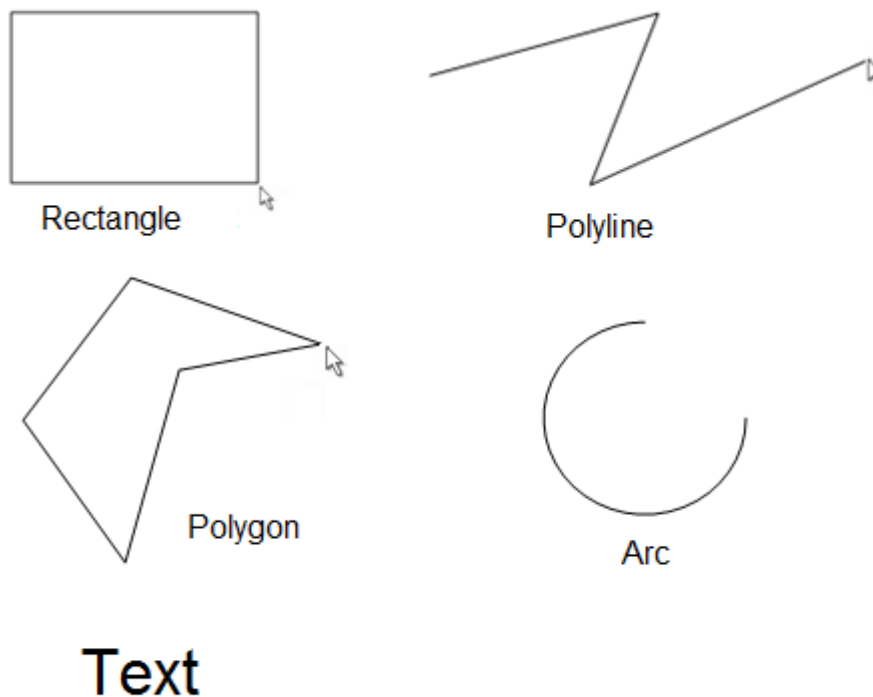


Fig. 36. Graphic primitives

A.1.2.2.3 Removing figure on mnemonic

The editor will allow removing selected figure from mnemonic. At removing figure references to it are removed in the following pools: selected figures on the scheme, pool of related figures. Mnemonic editor will allow removing several figures, using selected figures on a scheme. A group removal is a queued removing of figures in the selection mnemonic.

A.1.2.2.4 Creation of group figure from selected figures

The editor will allow creating group figure on a mnemonic. Group figure is created from selected figures on mnemonic. The component allows creation of group figure only from figures, located on the same layer. If some of the selected figures are located on other layer, then the system moves a smaller part of this group figure to a layer which contains a larger part of this group. After its creation the group figure

represents a single figure. Selecting figure included in the group figure leads to group figure selection. Changing geometric properties of group figure entails changing these properties of figures included in this group.

A.1.2.2.5 Ungrouping group figure

The editor will allow ungrouping group figure. When ungrouping a removing group figure is performed, and all related figures become independent. After ungrouping each figure may be selected, previously included in the group, individually. The geometric properties of figures defined in the group must be the same after group separation.

A.1.2.3 Mnemonics figures management

Mnemonics figures management is performed after the following user commands: changing geometric properties of figure on mnemonic, changing style of figure, view and edit attribute information of figure.

A.1.2.3.1 Changing geometric properties of figure on mnemonic

The editor will allow changing geometric properties of figure on a mnemonic. It includes: changing of rotation angle of figure relative to its center (for figures: point, polyline, polygon, rectangle, arc, text and group). For a group figure the center is defined according to the center of the area, took by this figure; changing of width, length, point of the upper left edge of taking area (for figures: polygon, rectangle, arc and text); location of figure center (for figures: point, polyline, polygon, rectangle, arc, text and group); positions of nodes (for figures: point, polyline, polygon and arc); geometric type of polylines and polygons (angled line, Bezier, spline); type of arc (arc, sector, segment).

A.1.2.3.2 Changing the style of a figure

The editor will allow changing style of a figure. For areal objects the following may be changed: color of border line, width of border line, type of borders ends, color of flood fill, color of gradient fill. For linear objects is available to change: color of border line, width of border line, type of borders ends. Areal objects are: polygon, rectangle, arc and text. Linear objects are: point, polyline.

A.1.2.3.3 View and edit attribute information of a figure

The editor will allow changing attribute information of a figure. Attribute information is contained in metadata of mnemonic figures.

A.1.2.4 Operations on mnemonic figures

Operations on mnemonic figures is performed after the following user command: changing order of figures on a layer, changing figure selection on a mnemonic, changing visibility of a figure, moving figure on another layer.

A.1.2.4.1 Changing order of figures on a layer

The editor will allow changing order of figures on a layer; as a consequence of this their visibility in the workspace is changing as well.

A.1.2.4.2 Changing figure selection on mnemonic

The editor will allow changing figure selection on mnemonic. Figure selection may be performed from all layers. When a figure is reselected, selection is removed.

A.1.2.4.3 Changing visibility of figures

The editor will allow changing visibility of figure. The figures that have their visibility turned off are not displayed in the workspace.

A.1.2.4.4 Moving figure on other layers

The editor will allow moving figure to other layers.

A.1.2.5 Mnemonics layers management

Default mnemonic has at least two layers: working and substrate. Working layer is assigned to create figures on mnemonic. Substrate layer is assigned to create system objects. System objects are selection frames of figures selection, nodes of boundaries of objects. Mnemonics layers management is performed after the following user commands: adding new layer to mnemonic, removing selected layer on mnemonic, changing selected layer on mnemonic, changing accessibility of layers on mnemonic, changing layer visibility on mnemonic.

Adding new layer to mnemonic

The editor will allow adding additional working layer to mnemonic. Number of working layers must not exceed 20. Each added working layer is placed in the last position in the layers pool.

Removing selected layer on mnemonic

The editor will allow removing selected layer on mnemonic. At removing selected layer all figures from the layer are removed. Removal of the selected layer is performed with shifting the lower layers in the pool one position up.

Changing accessibility of layers on mnemonic

The editor will allow changing accessibility of layers on mnemonic. The top layer can be moved only down. The lowest layer can be moved only up. The middle layers can be moved both up and down.

Changing layer visibility on mnemonic

The editor will allow changing layer visibility on mnemonic. The layers that have visibility turned off are not displayed in the workplace.

A.1.3 Determination of animation rules for mnemonics figures

A.1.3.1 Changing object identifier of data source into parameters of figures

The editor will allow changing object identifier of data source into parameters of figures on a mnemonic. To remove object identifier of a data source for selected figures on mnemonic it is necessary to enter an empty identifier. The object identifier of data source parameters is changed after the user command.

A.1.3.2 Selecting animation rule

The editor will allow selection of animation rules for selected figures on mnemonic. To select animation rules component allows assigning a value of object of data source of figure in accordance with animation style of displaying figures on the scheme. The editor will allow configuring up to 10 rules per mnemonic figure. Selection of an animation rule is performed after the following user commands: configure pool of connected figures, adding figure into pool of related figures, removing figure from pool of related figures.

A.1.3.3 Configure pool of connected figures

At opening of mnemonic editor automatically generates pool of connected figures by going through all mnemonic content and reading information about object identifier of data source in mnemonic figures.

A.1.3.3.1 Adding figure into pool of related figures

The editor will allow automatic adding of figure into a pool of related figures after adding object identifier of data source to figure.

A.1.3.3.2 Removing figure from pool of related figures

The editor will allow removing figure from pool of related figures after adding an empty identifier of data source to the figure.

A.1.4 Updating data of mnemonics objects

Updating data will be performed automatically after user command enabling the update. When user repeats the command the update procedure is terminated.

A.1.4.1 Receiving updated data from data source

The editor will make an update request to data source of objects identifiers pool of connected figures. The editor will allow receiving updated data from the data source. When data is received verification is performed.

A.1.4.1.1 Connecting to data source

The editor will connect to data source according to the connection string. If the connection is not established, the error is displayed.

A.1.4.1.2 Receiving updated data

The editor will receive updated data from data source according to data of identifiers of pool of related figures.

A.1.4.1.3 Verification of received data

The editor will verify the received data. If difference of time information between object of data source and data of figure from pool of related figures is bigger than the update interval, the data is considered unreliable.

A.1.4.2 Updating data of each figure from pool of related figures

The editor will update figures data, for those in the pool of related figures, based on data coming from a source object.

A.1.5 Working with repository of mnemonic styles

A.1.5.1 Management of pool of animation styles filling

Management of pool of animation styles must be performed after the following user commands: adding new style into pool of animation styles, removing selected style from pool of animation styles, downloading pool of animation styles, uploading pool of animation styles to file selecting style.

A.1.5.1.1 Adding new style into the pool of animation styles

The editor will allow adding new style into pool of animation styles. The editor will prohibit adding style, if style with that name already exists in the repository.

A.1.5.1.2 Removing selected style from the pool of animation styles

The editor will allow removing selected style from pool of animation styles.

A.1.5.1.3 Downloading pool of animation styles

At the first execution of repository of mnemonic styles management after starting the editor, pool of animation styles is filled from file in the *.xml format, stored in a predefined system folder.

A.1.5.1.4 Uploading pool of animation styles to file

At the completion of working in the system the editor will upload pool of animation styles to file in the *.xml format, stored in a predefined system folder.

A.1.5.1.5 Selecting style

The editor will allow selecting style in repository of mnemonic styles. It is available to select only one style in the repository.

A.1.5.2 Content of animation style management

The editor will allow changing attributes of selected style. Attributes are figure style settings which are changing depending on animation rules. Content management of animation style is performed after user command to manage the content of animation style is issued.

ПРИЛОЖЕНИЕ Б

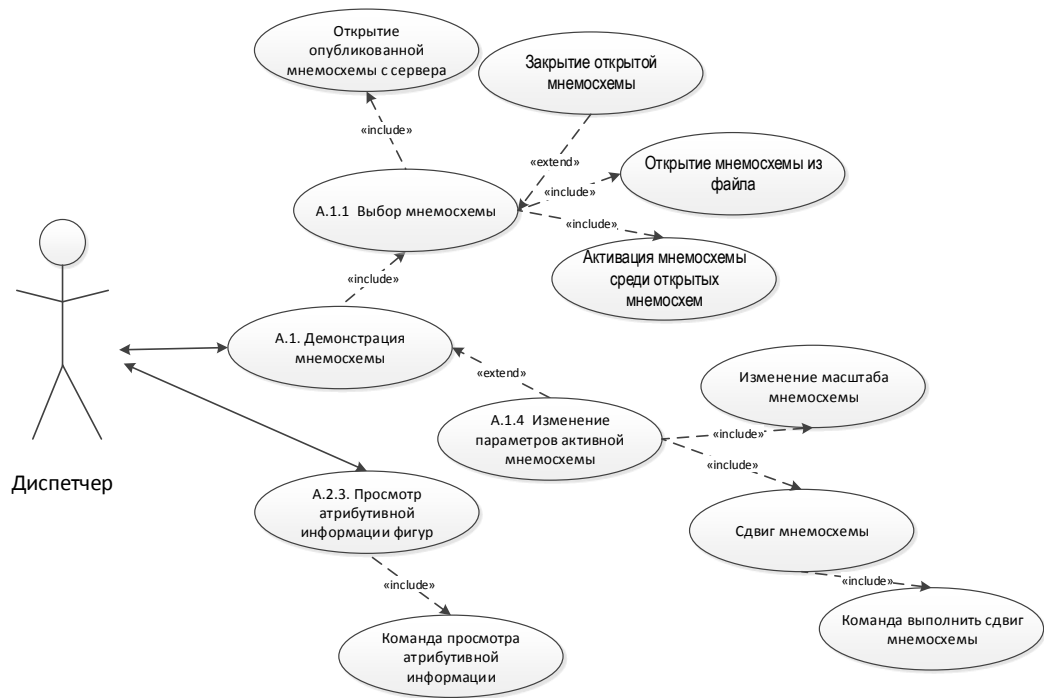


Рис. 37. Диаграмма вариантов использования роли Диспетчер

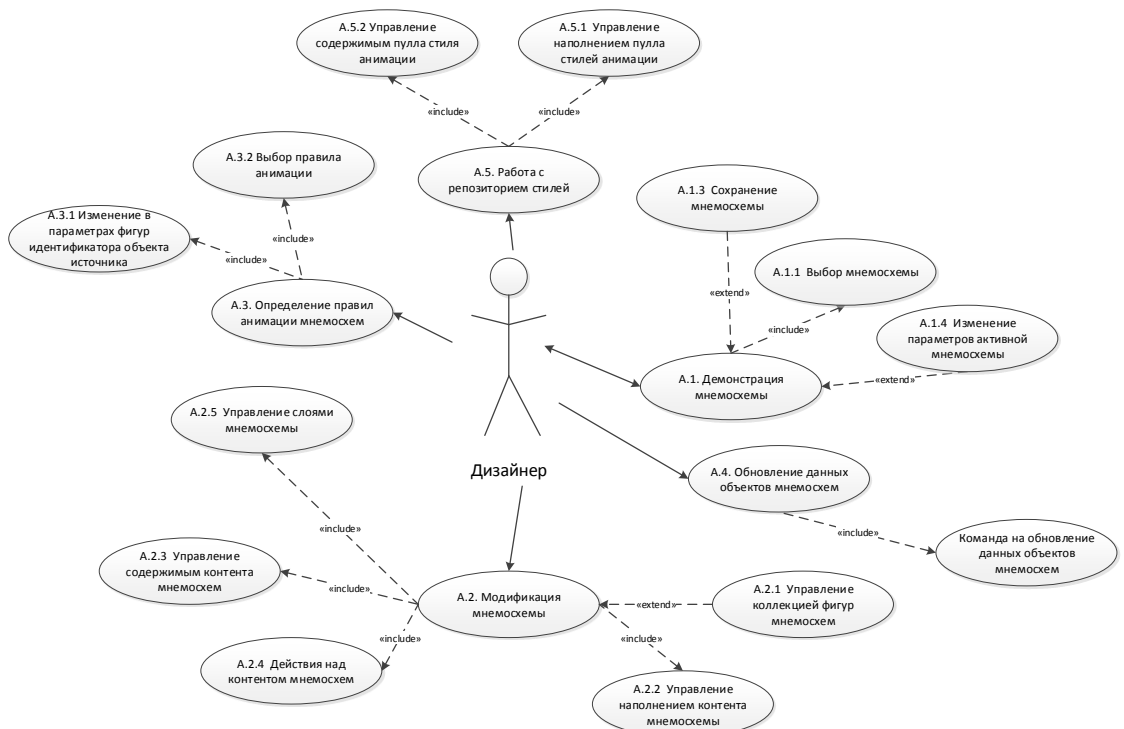


Рис. 38. Диаграмма вариантов использования роли Дизайнер

ПРИЛОЖЕНИЕ В

В данном приложении представлена спроектированная объектная модель универсальной векторной графической библиотеки (см. рис. 39).

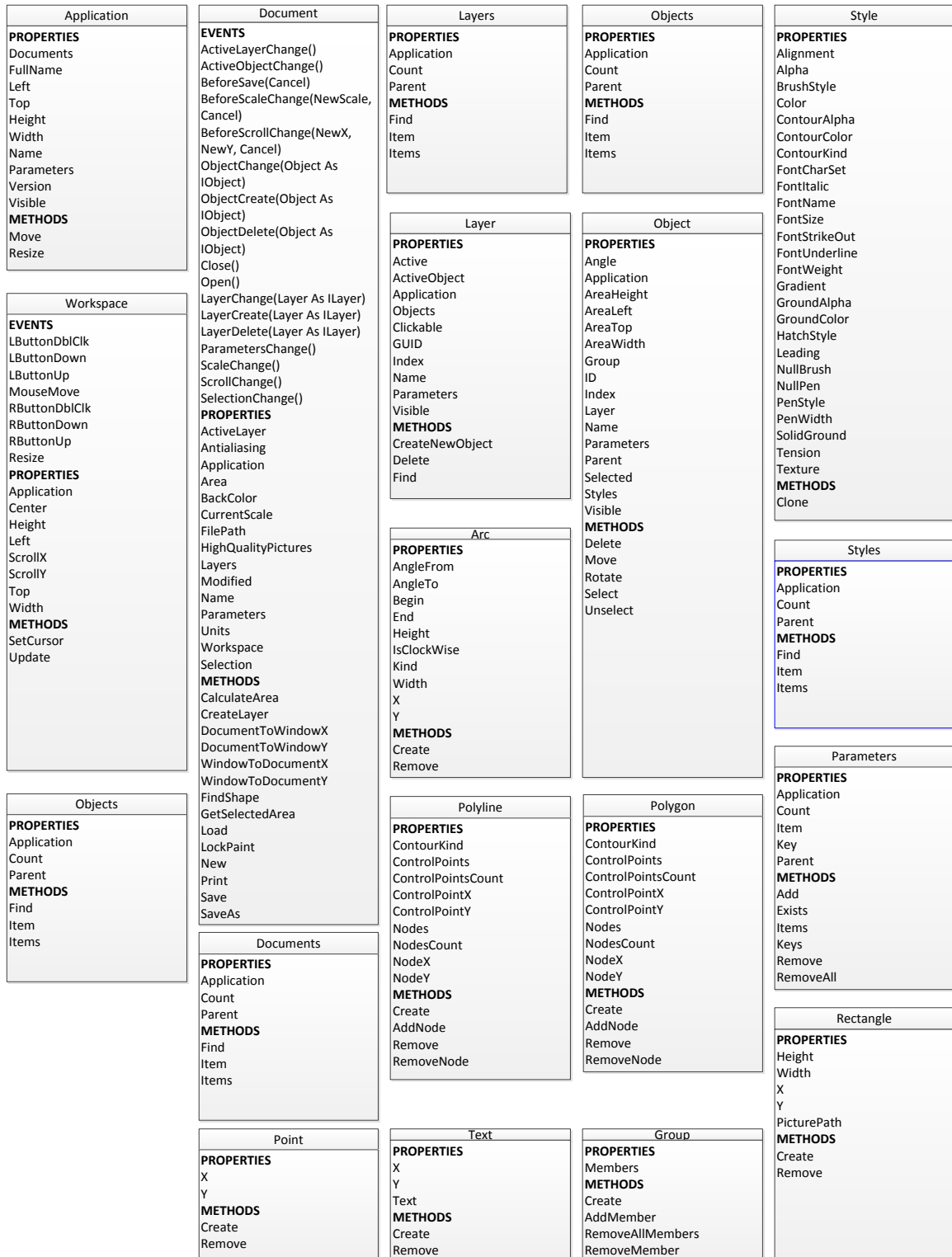


Рис. 39. Объектная модель универсальной векторной графической библиотеки

ПРИЛОЖЕНИЕ Г

Общество с ограниченной ответственностью «Сибирский центр высоких технологий»



ООО «СибХайТекЦентр»
Юридический адрес: 636039, Россия, Томская область, г. Северск, пр. Коммунистический, 70-43,
Фактический адрес: 634050, г. Томск, ул. Нахимова, 13/1, оф. 205,
тел. (3822) 21-30-93 e-mail: sibHTC@sibhtc.ru, http://www.sibhtc.ru
ИНН/КПП 7024023627/702401001, ОГРН1057004447807

УТВЕРЖДАЮ

Генеральный директор

ООО «СибХайТекЦентр»

А.В. Кудинов



Дата "20" мая 2016 г.

АКТ

о внедрении программного обеспечения

«Графические программные средства работы с мнемосхемами
«MnemoEditorPro»

Комиссия в составе генерального директора Кудинова Антона Викторовича и заместителя генерального директора Копнова Максима Валерьевича ООО «СибХайТекЦентр» рассмотрела НИР магистранта кафедры Вычислительной техники Института кибернетики Национального исследовательского Томского политехнического университета Сергеева Д.А. «Разработка графических средств для работы с технологическими мнемосхемами», выполненную под руководством доцента, к.т.н. Ковина Романа Владимировича.

Разработанные в рамках НИР Сергеева Д.А. графические программные средства работы с мнемосхемами «MnemoEditorPro» переданы в ООО «СибХайТекЦентр» с целью применения для создания и использования технологических мнемосхем в рамках «АРМ Технолога» ООО «СибМетаХим».

Зам. генерального директора  / Копнов М.В.

ПРИЛОЖЕНИЕ Д

В данном приложении представлена часть технического задания, описывающая требования к функциональным алгоритмам системы.

Приложение Д.[FA] Требования к алгоритмам работы функций

Приложение Д.[FA.01.00] А.1. - Демонстрация мнемосхем

Приложение Д.[FA.01.01] А.1.1- Выбор мнемосхемы

В зависимости от пользовательской команды система должна позволять выбрать мнемосхему для работы. Должно быть доступно создание новой мнемосхемы, открытие мнемосхемы из файла, активация текущей мнемосхемы, закрытие текущей мнемосхемы.

Приложение Д.[FA.01.01.01] А.1.1- Создание новой мнемосхемы

Система должна позволять создать мнемосхему в системе. Созданная новая мнемосхема должна содержать в себе минимум два слоя: рабочий и подложка. Новая мнемосхема должна становиться активной после создания.

Приложение Д.[FA.01.01.02] А.1.1- Открытие мнемосхемы из файла

Система должна позволять открыть мнемосхему из файла формата *.xml. Для выполнения открытия мнемосхемы из файла векторной графической компоненте должен быть передан путь к файлу на жестком диске. Открытая мнемосхема должна становиться активной после открытия. Контент мнемосхемы должен быть отображён в рабочей области.

Приложение Д.[FA.01.01.03] А.1.1- Активация мнемосхемы среди открытых мнемосхем

Система должна иметь пулл мнемосхем для обеспечения одновременной работы с несколькими мнемосхемами. Отображаемая мнемосхема является активной в пулле мнемосхем. Для активации мнемосхемы в пулле должна быть подана пользовательская команда активации мнемосхемы.

Приложение Д.[FA.01.01.04] А.1.1- Закрыть мнемосхему

Система должна позволять закрыть мнемосхему. При закрытии мнемосхемы она должна удаляться из пулла мнемосхем в системе. При закрытии должна стать активной мнемосхема, находящаяся в начале пулла.

Приложение Д.[FA.01.01.05] А.1.1- Открытие опубликованной мнемосхемы с сервера

Система должна позволять открыть мнемосхему, опубликованную на сервере. Для этого система должна подключиться к серверу используя строку

подключения. Система должна произвести выгрузку из структур данных мнемосхемы в формате XML и выполнить её открытие в рабочей области. Мнемосхема должна стать активной.

Приложение Д.[FA.01.02] А.1.2- Отображение мнемосхемы на дисплее

Система должна позволять отображать текущую мнемосхему на дисплее в рабочей области. При отображении мнемосхемы должно выполняться отображение контента, содержащегося на мнемосхеме, а также системных объектов содержащихся в слое подложки.

Приложение Д.[FA.01.02.01] А.1.2- Обновление текущей мнемосхемы

Система должна позволять выполнять обновление текущей мнемосхемы. Под обновлением подразумевается полная перерисовка контента мнемосхемы в рабочей области с учётом правил отображения стиля фигур и данных этих фигур.

Приложение Д.[FA.01.03] А.1.3- Сохранение мнемосхемы

Сохранение мнемосхемы должно осуществляться в случае, если поданы пользовательские команды сохранить мнемосхемы в файл или публикация мнемосхемы на сервере.

Приложение Д.[FA.01.03.01] А.1.3- Сохранение мнемосхемы в файл

Система должна позволять выполнять выгрузку в файл контента мнемосхемы. Для выполнения выгрузки системе должен быть подан путь сброса файла формата *.xml, в который должно быть выполнено сохранение мнемосхемы.

Приложение Д.[FA.01.03.02] А.1.3- Публикация мнемосхемы на сервере

Система должна позволять опубликовать мнемосхему на сервере используя строку подключения к серверу. При публикации мнемосхемы на сервер должна выполняться её выгрузка в структурах таблиц базы данных в формате данных XML.

Приложение Д.[FA.01.04] А.1.4- Изменение параметров активной мнемосхемы

Изменение параметров активной мнемосхемы должно осуществляться, если поданы пользовательские команды: изменить название мнемосхемы, изменить масштаб мнемосхемы, сдвиг мнемосхемы.

Приложение Д.[FA.01.04.01] А.1.4- Изменение названия мнемосхемы

Система должна позволять изменять название мнемосхемы. Система должна позволять задавать название мнемосхемы отличное от названия файла хранения мнемосхемы.

Приложение Д.[FA.01.04.02] А.1.4- Изменение масштаба мнемосхемы

Система должна позволять изменять масштаб мнемосхемы в рабочей области. При изменении масштаба должен изменяться шаг метрики координат: при увеличении становится детальней, при уменьшении становится более общим.

Приложение Д.[FA.01.04.03] А.1.4- Сдвиг мнемосхемы

Система должна позволять изменять центр мнемосхемы в рамках координатной плоскости рабочей области.

Приложение Д.[FA.02.00] А.2. - Модификация мнемосхемы

Приложение Д.[FA.02.01] А.2.1- Управление коллекцией фигур

мнемосхем

Коллекция фигур мнемосхем должна представлять из себя библиотеку, в которой доступно хранение готовых пользовательских групповых фигур с целью последующего многократного использования. При первом выполнении управления коллекцией фигур мнемосхем после запуска системы должно выполняться наполнение пулла коллекции из файла формата *.xml, хранящегося в предопределённой системной папке. При завершении работы в системе должна выполняться выгрузка пулла коллекции в файл формата *.xml, хранящегося в предопределённой системной папке. Управление коллекцией фигур должно осуществляться, если поданы пользовательские команды: добавить групповую фигуру на мнемосхему из коллекции, создать групповую фигуру на мнемосхеме, выбрать групповую фигуру в коллекции, удалить групповую фигуру из коллекции.

Приложение Д.[FA.02.01.01] А.2.2- Добавить групповую фигуру в

коллекцию

Система должна позволять добавлять групповую фигуру в коллекцию фигур. Система должна не допускать наличие в коллекции фигур с одинаковыми названиями.

Приложение Д.[FA.02.01.02] А.2.2- Удалить групповую фигуру из

коллекции

Система должна позволять удалить выбранную фигуру из пулла коллекции.

Приложение Д.[FA.02.01.03] А.2.2- Выбрать групповую фигуру в коллекции

Система должна позволять выбрать фигуру в коллекции. Должен быть доступен выбор только одной фигуры в коллекции.

Приложение Д.[FA.02.02] А.2.2- Управление наполнением контента мнемосхемы

Управление наполнением контента мнемосхемы должно осуществляться, если поступили пользовательские команды: добавить фигуру на мнемосхему из коллекции, создать фигуру на мнемосхеме, удалить фигуру на мнемосхеме, создать групповую фигуру из выбранных фигур, разделить групповую фигуру.

Приложение Д.[FA.02.02.01] А.2.2- Добавить групповую фигуру на мнемосхему из коллекции

Система должна позволять добавить выделенную групповую фигуру на мнемосхему из коллекции. При добавлении фигуры из коллекции на мнемосхему должна добавляться фигура типа группа.

Приложение Д.[FA.02.02.02] А.2.2- Создать фигуру на мнемосхеме

Система должна позволять создать на мнемосхеме фигуру из набора графических примитивов. Графическими примитивами являются: точка, полилиния, полигон, прямоугольник, дуга, текст. При создании фигура помещается на верхний слой.

Приложение Д.[FA.02.02.03] А.2.2- Удалить фигуру на мнемосхеме

Система должна позволять выполнять удаление фигуры выделенной фигуры на мнемосхеме. При удалении фигуры должно выполняться удаление ссылок на неё в пуллах: выборка фигур на схеме, пулл связанных фигур. Система должна позволять удалить несколько фигур, используя выборку фигур на мнемосхеме. При групповом удалении система должна поочерёдно удалять фигуры, находящиеся в выборке мнемосхемы.

Приложение Д.[FA.02.02.04] А.2.2- Создать групповую фигуру из выбранных фигур

Система должна позволять создать групповую фигуру на мнемосхеме. Групповая фигура создаётся из выбранных фигур на мнемосхеме. Система должна позволять создавать групповую фигуру только для фигур, содержащихся на одном слое. Если часть фигур выборки находятся на другом слое, то система должна перемещать меньшую часть этих фигур на слой, в котором расположена большая часть этих фигур. После создания групповая фигура должна быть как

единая фигура. Выбор фигур включённых в группу должен приводить к выбору самой групповой фигуры. Изменение геометрических свойств групповой фигуры должно повлечь изменение этих свойств у фигур, включённых в эту группу.

Приложение Д.[FA.02.02.05] А.2.2- Разделить групповую фигуру

Система должна позволять выполнить разделение групповой фигуры. При разделении система должна удалить групповую фигуру, а все включённые фигуры должна стать самостоятельными. После разделения группы доступен выбор отдельно каждой фигуры, ранее включённой в группу. Геометрические свойства группы распространяются на все включённые фигуры в группу.

Приложение Д.[FA.02.03] А.2.3- Управление содержимым контента мнемосхем

Управление содержимым контента мнемосхемы должно выполняться, если поступила пользовательская команда: изменить геометрию фигуры на мнемосхеме, изменить стиль оформления фигуры, просмотр и изменение атрибутивной информации фигур.

Приложение Д.[FA.02.03.01] А.2.3- Изменить геометрию фигуры на мнемосхеме

Система должна позволять изменять геометрию фигур на мнемосхеме. Изменение геометрии включает в себя: изменение угла поворота фигуры относительно её центра (для фигур: точка, полилиния, полигон, прямоугольник, дуга, текст, группа); для группы фигур центр выбирается согласно центру области занимаемой всеми фигурами группы на рабочей плоскости; изменение ширины, длины, точки левого края занимаемой области (для фигур полигон, прямоугольник, дуга, текст); места расположения точки центра фигуры (для фигур: точка, полилиния, полигон, прямоугольник, дуга, текст, группа); положения узлов (для полилиния, полигон, дуга, точка); тип геометрии полилиний и полигонов (ломанная, кривая безье, сплайн); тип дуги (дуга, сектор, сегмент).

Приложение Д.[FA.02.03.02] А.2.3- Изменить стиль оформления фигуры

Система должна позволять изменить стиль оформления фигуры. Для площадных объектов доступно изменение: цвет линии границы, толщина линии границы, тип концов границы, цвет сплошной заливки, цвет градиентной заливки. Для линейных объектов должно быть доступно следующее изменение стиля: цвет линии границы, толщина линии границы, тип концов границы. Площадными

объектами являются: полигон, прямоугольник, дуга, текст. Линейными объектами являются: точка, полилиния

Приложение Д.[FA.02.03.03] А.2.3- Просмотр и изменение атрибутивной информации фигур

Система должна позволять изменять атрибутивную информацию фигур. Атрибутивная информация должна содержаться в метаданных фигур мнемосхем. При встраивании векторной графической компоненты в обозреватель мнемосхем должен быть доступен только просмотр атрибутивной информации фигуры.

Приложение Д.[FA.02.04] А.2.4- Действия над контентом мнемосхем

Действие над контентом пользователя должно выполняться при поступлении пользовательских команд: изменить порядок фигуры на слое, изменить выборку фигур на мнемосхеме, изменить видимость фигуры, переместить фигуру на другой слой.

Приложение Д.[FA.02.04.01] А.2.4- Изменить порядок фигуры на слое

Система должна позволять менять порядок фигур на слое. Это должно позволять менять их взаимную видимость на рабочей области.

Приложение Д.[FA.02.04.02] А.2.4- Изменить выборку фигур на мнемосхеме

Система должна позволять выполнять выборку фигур на мнемосхеме. Выборка фигур должна выполняться со всех слоёв. При повторном выборе фигуры она должна удаляться из выборки.

Приложение Д.[FA.02.04.03] А.2.4- Изменить видимость фигуры

Система должна позволять изменять видимость фигур. Фигуры, у которых видимость отключена, не должны отображаться на рабочей области.

Приложение Д.[FA.02.04.04] А.2.4- Переместить фигуру на другой слой

Система должна позволять перемещать фигуры на другой слой.

Приложение Д.[FA.02.05] А.2.5- Управление слоями мнемосхемы

По умолчанию мнемосхема должна иметь минимум два слоя: рабочий и подложку. Рабочий слой предназначен для создания фигур на мнемосхеме. Слой подложка предназначен для создания объектов системы. Объектами системы являются рамки выделения выборки фигур, узлы границ объектов. Управление слоями мнемосхемы должно осуществляться при поступлении пользовательских команд: добавить новый слой к мнемосхеме, удалить выбранный слой на мнемосхеме, изменить порядок слоёв мнемосхемы, изменить видимость слоя на мнемосхеме.

Приложение Д.[FA.02.05.01] А.2.5- Добавить новый слой к мнемосхеме

Система должна позволять добавлять дополнительный рабочий слой к мнемосхеме. Количество рабочих слоёв должно быть не больше 20. Каждый добавляемый рабочий слой помещается на последнее место в пуле слоёв.

Приложение Д.[FA.02.05.02] А.2.5- Удалить выбранный слой на мнемосхеме

Система должна позволять удалять выбранный слой мнемосхемы. При удалении выбранного слоя мнемосхемы должно выполняться удаление всех

фигур слоя. Удаление фигур слоя должно осуществляться согласно требованиям к удалению фигур. Удаление выбранного слоя должно выполняться со смещением всех нижестоящих слоёв в пуле на позицию вверх.

Приложение Д.[FA.02.05.03] А.2.5- Изменить порядок слоёв мнемосхемы

Система должна позволять менять порядок слоёв на мнемосхеме. Самый верхний слой можно переместить только вниз. Самый нижний слой можно переместить только вверх. Серединные слои могут быть перемещены как вверх так и вниз.

Приложение Д.[FA.02.05.04] А.2.5- Изменить видимость слоя на мнемосхеме

Система должна позволять менять видимость слоя на схеме. Фигуры, содержащиеся на невидимом слое, не должны отображаться на рабочей области.

Приложение Д.[FA.03.00] А.3. - Определение правил анимации фигур мнемосхем

Приложение Д.[FA.03.01] А.3.1- Изменение в параметрах фигур идентификатора объекта источника данных

Система должна позволять изменить идентификатор объекта источника для выбранных фигур на мнемосхеме. Для удаления идентификатора объекта источника для выбранных фигур на мнемосхеме необходимо, чтобы был передан пустой идентификатор. Изменение в параметрах фигур объекта источника выполняется при поступлении пользовательской команды изменить идентификатор объекта источника данных фигуры.

Приложение Д.[FA.03.02] А.3.2- Выбор правила анимации

Система должна позволять выбрать правила для анимации для выбранных фигур на мнемосхеме. Для выбора правил анимации система должна позволять сопоставить значение объекта источника данных фигуры со стилем анимации отображения фигуры на схеме. Система должна позволять настраивать до 10 правил для фигуры мнемосхемы. Выбор правила анимации осуществляется при поступлении пользовательских команд: управление пуллом связанных фигур, добавить фигуру в пулл связанных фигур, удалить фигуру из пулла связанных фигур.

Приложение Д.[FA.03.03] А.3.3- Управление пуллом связанных фигур

При открытии мнемосхемы система должна автоматически формировать пулл связанных фигур мнемосхемы обходом всего контента мнемосхемы с чтением информации идентификатора объекта источника данных у фигур мнемосхем.

Приложение Д.[FA.03.03.01] А.3.3- Добавить фигуру в пулл связанных фигур

Система должна автоматически добавлять фигуру в пулл связанных фигур после добавления к фигуре идентификатора источника данных.

Приложение Д.[FA.03.03.02] А.3.3- Удалить фигуру из пулла связанных фигур

Система должна автоматически удалять фигуру из пулла связанных фигур после добавление к фигуре пустого идентификатора источника данных.

Приложение Д.[FA.04.00] А.4. - Обновление данных объектов мнемосхем

Обновление данных должно выполняться автоматически после вызова пользовательской команды включения обновления данных. При повторном вызове команды процедура обновления должна прекращаться.

Приложение Д.[FA.04.01] А.4.1- Получение обновлённых данных от источника данных

Система должна выполнять запрос обновления к источнику данных по идентификаторам объектов источников данных из пулла связанных фигур. Система должна выполнять получение обновлённых данных от объекта источника данных. При получении данных должна выполняться верификация.

Приложение Д.[FA.04.01.01] А.4.1- Подключение к источнику данных

Система должна выполнять подключение к источнику данных по данным строки подключения. В случае если подключение не установлено, система должна записать отобразить системную ошибку.

Приложение Д.[FA.04.01.02] А.4.1- Получение обновлённых данных

Система должна выполнить получение обновлённых данных от источника данных по идентификаторам пулла связанных фигур.

Приложение Д.[FA.04.01.03] А.4.1- Верификация полученных данных

Система должна выполнять верификацию полученных данных. Если разница между временными данными объекта источника данных и данных фигуры пулла связанных фигур больше, чем интервал обновления, то данные считать недостоверными.

Приложение Д.[FA.04.02] А.4.2- Обновление данных каждой фигуры пулла связанных фигур

Система должна выполнять обновление данных фигур, находящихся в пулле связанных фигур, на основе пришедших данных от объекта источника.

Приложение Д.[FA.05.00] А.5. - Работа с репозиторием стилей мнемосхем

Приложение Д.[FA.05.01] А.5.1- Управление наполнением пула стилей анимации

Управление наполнением пулла стилей анимации должно выполняться при поступлении пользовательских команд: добавить новый стиль в пулл стилей анимации. Удалить выбранный стиль из пулла стилей анимации, выбрать стиль.

Приложение Д.[FA.05.01.01] А.5.1- Добавить новый стиль в пулл стилей анимации

Система должна позволять добавить новый стиль в пулл стилей анимации. Система должна запрещать добавление стиля, если стиль с таким именем уже есть в репозитории.

Приложение Д.[FA.05.01.02] А.5.1- Удалить выбранный стиль из пулла стилей анимации

Система должна позволять удалить выбранный стиль из пулла стилей.

Приложение Д.[FA.05.01.03] А.5.1- Загрузить пулл стилей из файла

При первом выполнении управления репозиторием стилей мнемосхем после запуска системы должно выполняться наполнение пулла стилей из файла формата *.xml, хранящегося в предопределённой системной папке.

Приложение Д.[FA.05.01.04] А.5.1- Выгрузить пулл стилей в файл

При завершении работы в системе должна выполняться выгрузка пулла стилей в файл формата *.xml, хранящегося в предопределённой системной папке.

Приложение Д.[FA.05.01.05] А.5.1- Выбрать стиль

Система должна позволять выбрать стиль в репозитории стилей мнемосхем. Должен быть доступен выбор только одного стиля в репозитории.

Приложение Д.[FA.05.02] А.5.2- Управление содержимым стиля анимации

Система должна позволять изменять атрибуты выбранного стиля. Под атрибутами понимаются настройки стиля, которые должны изменяться у фигуры в зависимости от правила анимации. Управление содержимым стиля анимации должно выполняться при поступлении пользовательской команды на управление содержимым стиля анимации.