Модуль подсистемы “Пользовательские интерфейсы” <VCAEngine>

Модуль:	VCAEngine
Имя:	Движок среды визуализации и управления
Тип:	Пользовательские интерфейсы
Источник:	ui_VCAEngine.so
Версия:	0.8.10
Автор:	Роман Савоченко
Описание:	Основной движок среды визуализации и управления.
Лицензия:	GPL

Оглавление

Модуль подсистемы “Пользовательские интерфейсы” <VCAEngine>.....1
Введение.....3
1. Назначение.....4
2. Конфигурация и формирование интерфейсов СВУ.....5
3. Архитектура.....5
3.1. Кадры и элементы отображения(виджеты).....7
3.2. Проект.....11
3.3. Темы отображения.....15
3.4. События, их обработка и карты событий.....15
3.5. Сигнализация.....18
3.6. Управление правами.....19
3.7. Связывание с динамикой.....19
3.8. Примитивы виджетов.....24
3.8.1. Элементарные графические фигуры (ElFigure).....26
3.8.2. Элементы формы (FormEl).....28
3.8.3. Элемент текста (Text).....30
3.8.4. Элемент отображения медиа-материалов (Media).....30
3.8.5. Элемент построения диаграмм/трендов (Diagram).....31
3.8.6. Элемент построения протоколов, на основе архивов сообщений (Protocol).....32
3.8.7. Элемент формирования отчётной документации (Document).....33
3.8.8. Контейнер (Box).....35
3.9. Использование БД для хранения библиотек виджетов и проектов.....36
3.10 API пользовательского программирования и сервисные интерфейсы OpenSCADA.....38
3.10.1. API пользовательского программирования.....38
Список виджетов (WdgList).....38
Присутствие узла (NodePresent).....38
Список атрибутов (AttrList).....38
Запрос атрибута (AttrGet).....38
Установка атрибута (AttrSet).....39
3.10.2. Сервисные интерфейсы OpenSCADA.....39
Доступ к значениям атрибутов элементов визуализации (виджетам).....39
Групповой доступ к значениям атрибутов элементов визуализации (виджеты).....39

Доступ к страницам сеанса.....40
Управление сигнализацией.....40
Манипуляция сеансами проектов.....40
Групповой запрос дерева библиотек виджетов.....41
4. Конфигурация модуля посредством интерфейса управления OpenSCADA.....42

Введение

Модуль VCAEngine предоставляет движок среды визуализации и управления (СВУ) в систему OpenSCADA. Сам модуль не реализует визуализации СВУ, а содержит данные в соответствии с идеологией «Модель/данные – Интерфейс». Визуализация данных этого модуля выполняется модулями визуализации СВУ например, модулем Vision и WebVision.

Среда визуализации и управления (СВУ) является неотъемлемой составляющей SCADA системы. Она применяется на клиентских станциях с целью доступного предоставления информации об объекте управления и выдачи управляющих воздействий на объект. В различных практических ситуациях и условиях могут применяться СВУ, построенные на различных принципах визуализации. Например, это могут быть библиотеки виджетов QT, GTK+, wxWidgets или гипертекстовые механизмы на основе технологий HTML, XHTML, XML, CSS и JavaScript или сторонние приложения визуализации, реализованные на различных языках программирования Java, Python и т.д. Любой из этих принципов имеет свои преимущества и недостатки, комбинация которых может стать непреодолимым препятствием в возможности использования СВУ в том или ином практическом случае. Например, технологии вроде библиотеки QT позволяют создавать высокореактивные СВУ, что несомненно важно для станций оператора управления технологическим процессом (ТП). Однако необходимость инсталляции данного клиентского ПО в отдельных случаях может сделать использование его невозможным. С другой стороны, Web-технологии не требуют инсталляции на клиентские системы и являются предельно многоплатформенными (достаточно создать ссылку на Web-сервер в любом Web-браузере), что наиболее важно для различных инженерных и административных станций, но реактивность и надёжность таких интерфейсов ниже, что практически исключает их использования на станциях оператора ТП.

Система OpenSCADA имеет предельно гибкую архитектуру, которая позволяет создавать внешние интерфейсы, в том числе и пользовательские, на любой основе и на любой вкус. Например, среда конфигурации системы OpenSCADA доступна как на QT библиотеке, так и на Web-основе.

В тоже время независимое создание реализаций СВУ на различной основе может повлечь за собой невозможность использования данных конфигурации одной СВУ в другой. Что неудобно и ограничено с пользовательской стороны, а также накладно в плане реализации и последующей поддержки. С целью избежать этих проблем, а также создать в кратчайшие сроки полный спектр различных типов СВУ основан проект создания концепции СВУ. Результатом этого проекта и стал данный модуль движка(модели данных) СВУ, а также модули непосредственной визуализации Vision и WebVision.

1. Назначение

Данный модуль движка(модели данных) СВУ предназначен для формирования логической структуры СВУ и исполнения сеансов отдельных экземпляров проектов СВУ. Также модуль предоставляет все необходимые данные конечным визуализаторам СВУ как посредством локальных механизмов взаимодействия OpenSCADA, так и посредством интерфейса управления OpenSCADA для удалённого доступа.

Финальная версии этого модуля СВУ, построенная на основе данного модуля, обеспечит:

- три уровня сложности в формировании интерфейса визуализации, позволяющие органично осваивать и применять инструментарий по методике от простого к сложному:
  - формирование из шаблонных кадров путём назначения динамики (без графической конфигурации);
  - графическое формирование новых кадров путём использования готовых элементов визуализации из библиотеки (мнемосхемы);
  - формирование новых кадров, шаблонных кадров и элементов отображение в библиотеки.
- построение интерфейсов визуализации различной сложности, начиная от простых плоских интерфейсов мониторинга и заканчивая полноценными иерархическими интерфейсами, используемыми в SCADA системах;
- предоставление различных способов формирования и конфигурации пользовательского интерфейса, основанных на различных интерфейсах графического представления (QT, Web, Java ...) или-же посредством стандартного интерфейса управления системой OpenSCADA;
- смену динамики в процессе исполнения;
- построение новых шаблонных кадров на уровне пользователя и формирование специализированных под область применения библиотек кадров (например включение кадров параметров, графиков и других элементов с увязкой их друг с другом) в соответствии с теорией вторичного использования и накопления;
- построение новых пользовательских элементов визуализации и формирование специализированных под область применения библиотек кадров в соответствии с теорией вторичного использования и накопления;
- описание логики новых шаблонных кадров и пользовательских элементов визуализации как простыми связями, так и лаконичным, полноценным языком пользовательского программирования;
- возможность включения в пользовательские элементы визуализации функций (или кадров вычисления функций) объектной модели OpenSCADA, практически связывая представление с алгоритмом вычисления (например, визуализируя библиотеку моделей аппаратов ТП для последующего визуального построения моделей ТП);
- разделение данных пользовательских интерфейсов и интерфейсов представления этих данных, позволяющее строить интерфейс пользователя в одной среде, а исполнять во многих других (QT, Web, Java ...);
- возможность подключения к исполняющемуся интерфейсу для наблюдения и коррекции действий (например, при обучении операторов и контроля в реальном времени за его действиями);
- визуальное построение различных схем с наложением логических связей и последующим централизованным исполнением в фоне (визуальное построение и исполнение математических моделей, логических схем, релейных схем и иных процедур);
- предоставление функций объектного API в систему OpenSCADA может использоваться для управления свойствами интерфейса визуализации из пользовательских процедур;
- построение серверов кадров, элементов визуализации и проектов интерфейсов визуализации с возможностью обслуживания множественных клиентских соединений;
- простая организация клиентских станций на различной основе (QT, Web, Java ...) с подключением к центральному серверу;
- полноценный механизм разделения полномочий между пользователями, позволяющий создавать и исполнять проекты с различными правами доступа к его компонентам;
- гибкое формирование правил сигнализаций и уведомления с учётом и поддержкой

различных способов уведомления;
- поддержка пользовательского формирования палитры и шрифтовых предпочтений для интерфейса визуализации;
- поддержка пользовательского формирования карт событий под различное оборудование управления и пользовательские предпочтения;
- поддержка профилей пользователей, позволяющая определять различные свойства интерфейса визуализации (цветовая гамма, шрифтовые особенности, предпочтительные карты событий);
- гибкое хранение и распространение библиотек виджетов, кадров и проектов интерфейсов визуализации в БД, поддерживаемых системой OpenSCADA; практически пользователю нужно только зарегистрировать полученную БД с данными.

2. Конфигурация и формирование интерфейсов СВУ

Сам модуль не содержит инструмента визуального формирования интерфейсов СВУ, основанного на одном из механизмов. Такие инструменты могут предоставляться модулями конечной визуализации СВУ например, модулем Vision такой инструмент предоставляется.

Хотя визуального инструмента формирования СВУ модулем не предоставляется, для управления логической структурой предоставляется интерфейс, реализованный на основе интерфейса управления OpenSCADA, а значит доступный для использования в любом конфигураторе системы OpenSCADA. Диалоги этого интерфейса рассмотрены далее в контексте рассмотрения архитектуры модуля и его данных.

3. Архитектура

Любая СВУ может работать в двух режимах – разработки и исполнения. В режиме разработки формируется интерфейс СВУ, его компоненты и определяются механизмы взаимодействия. В режиме исполнения выполняется формирование интерфейса СВУ и производится взаимодействие с конечным пользователем на основе разработанных СВУ.

Интерфейс СВУ формируется из кадров, каждый из которых, в свою очередь, формируется из элементов примитивов или пользовательских элементов интерфейса. При этом пользовательские элементы интерфейса также формируются из примитивов или других пользовательских элементов. Таким образом обеспечивается иерархичность и повторное использования уже разработанных компонентов.

Кадры и пользовательские элементы размещаются в библиотеках виджетов. Из элементов этих библиотек формируются проекты интерфейсов конечной визуализации СВУ. На основе же этих проектов формируются сеансы визуализации.

Описанная структура СВУ приведена на рис. 3.

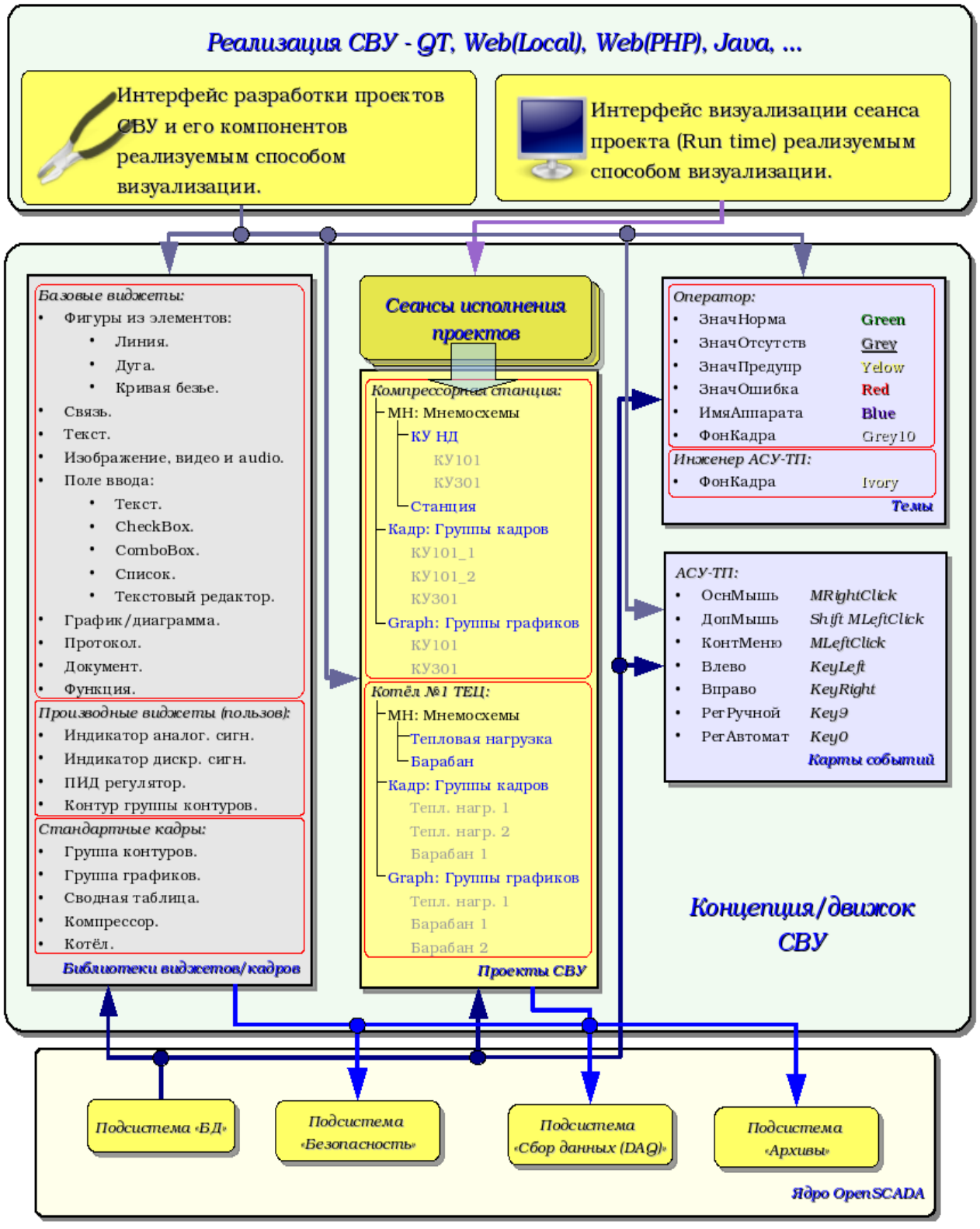


Рис.3 Обобщённая структура СВУ.

Данная архитектура СВУ позволяет реализовать поддержку трёх уровней сложности процесса разработки интерфейсов управления:

- Формирования интерфейса ВУ(визуализации и управления) с помощью библиотеки шаблонных кадров путём помещения шаблонов кадров в проект и назначения динамики.
- В дополнении к первому уровню производится формирование собственных кадров на основе библиотеки производных и базовых виджетов. Возможно как прямое назначение динамики в виджете, так и последующее её назначение в проекте.
- В дополнении ко второму уровню производится самостоятельное формирование

производных виджетов, новых шаблонных кадров, а также кадров с использованием механизма описания логики взаимодействия и обработки событий на одном из языков пользовательского программирования системы OpenSCADA.

3.1. Кадры и элементы отображения(виджеты)

Кадр – это окно, непосредственно предоставляющее информацию пользователю в графической и/или текстовой форме. Группа взаимосвязанных кадров формирует цельный пользовательский интерфейс ВУ.

Содержимое кадра формируется из элементов отображения(виджетов). Виджеты могут быть базовыми примитивами (различные плоские фигуры, текст, тренд и т.д.) и производными (сформированные из базовых или других производных виджетов). Все виджеты группированы по библиотекам. В процессе работы пользователь может формировать собственные библиотеки производных виджетов.

Собственно кадр также является виджетом, который используется в роли конечного элемента визуализации. А это значит, что библиотеки виджетов могут хранить и заготовки кадров, и шаблоны результирующих страниц пользовательского интерфейса.

Кадры и виджеты являются пассивными элементами, которые обычно не содержат связей с динамикой и другими кадрами, а только предоставляют информацию о свойствах виджета и характере динамики(конфигурации), подключаемой к свойствам кадра. Активированные кадры, т.е. содержащие ссылки на динамику и активные связи, формируют пользовательский интерфейс и хранятся в проектах. В некоторых случаях возможно прямое назначение динамики в заготовках кадров.

Производные кадры/виджеты могут содержать другие виджеты(вложенные), которые могут быть склеены(связаны) логикой один с другим с помощью одного из языков пользовательского программирования, доступного в системе OpenSCADA (рис.3.1.1).

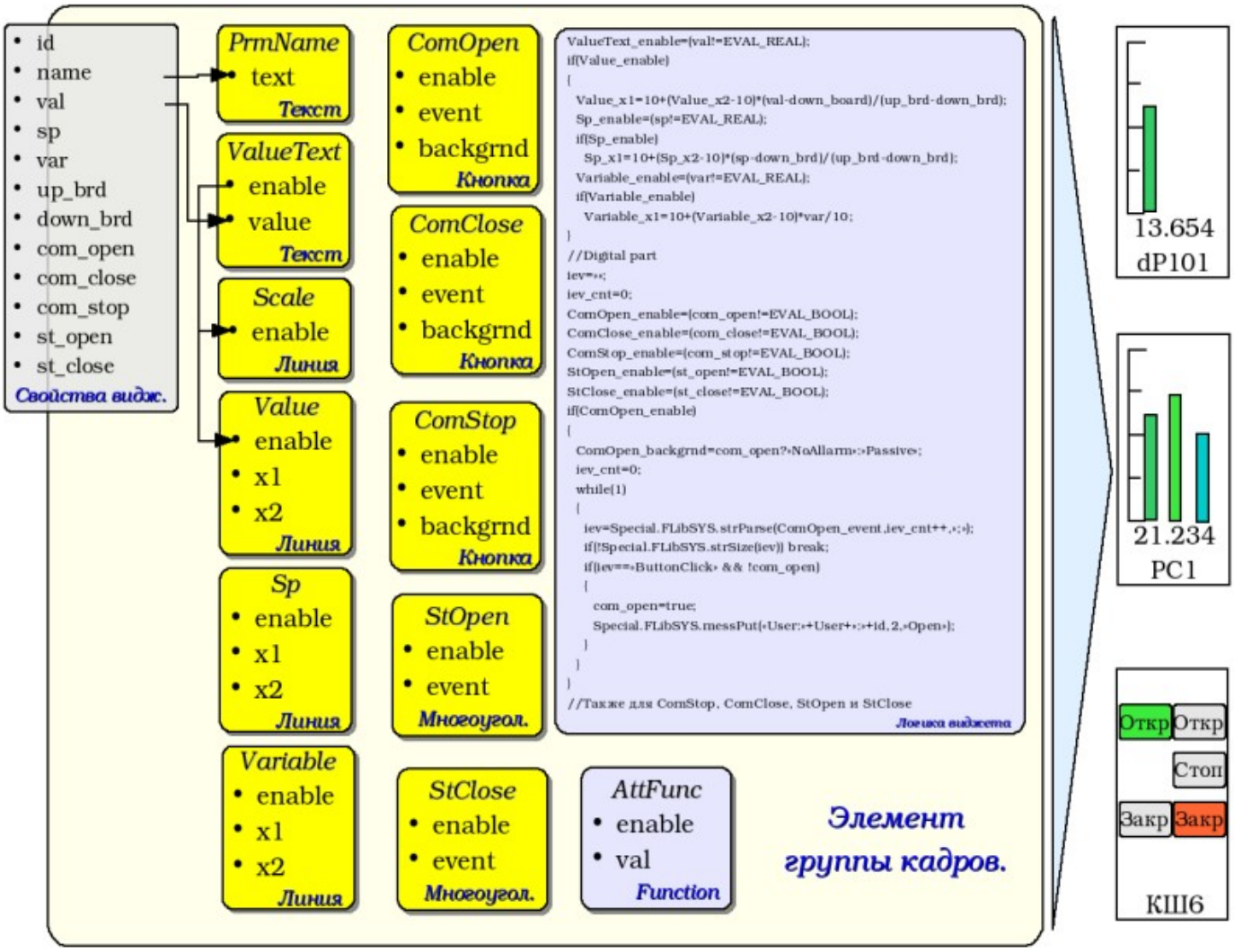


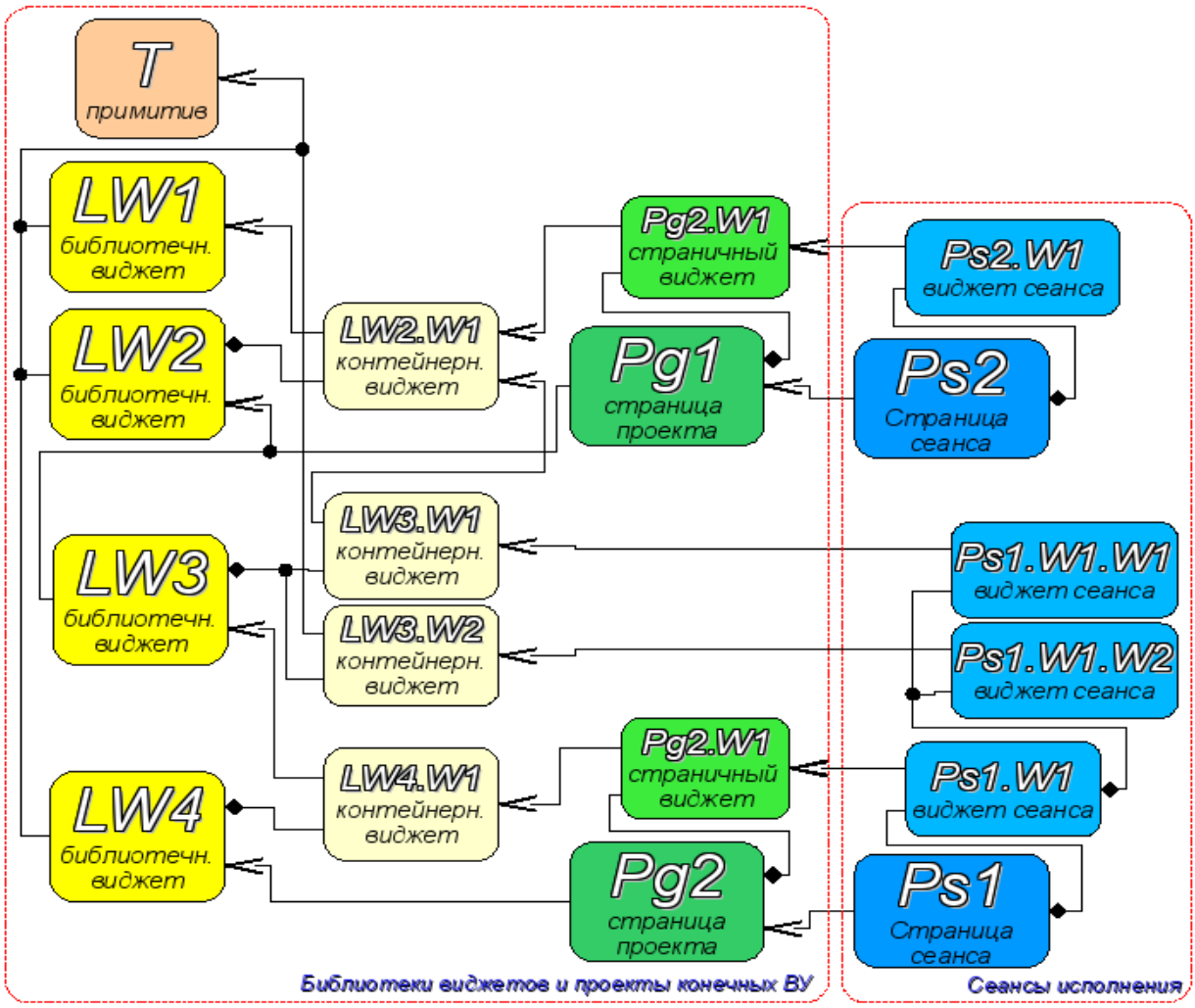
Рис.3.1.1 Пример структуры производного виджета.

Виджет является элементом, посредством которого обеспечивается:

- визуализация оперативной и архивной информации ведения ТП;
- сигнализация про нарушения ведения ТП;
- переключение между кадрами ТП;
- управление технологическим оборудованием и параметрами ведения ТП.

Настройка и связывание виджетов производится посредством их свойств. Родительский виджет и виджеты, содержащиеся в нем, могут дополняться пользовательскими свойствами. В последствии пользовательские и статические атрибуты связываются со свойствами вложенных виджетов посредством внутренней логики. Для отображения динамики (т.е. текущих и архивных данных) свойства виджетов динамизируются, т.е связываются с атрибутами параметров OpenSCADA или свойствами других виджетов. Использование для связывания вложенных виджетов внутренней логикой доступных языков пользовательского программирования системы OpenSCADA снимает вопрос реализации сложной логики визуализации, обеспечивая тем самым высокую гибкость. Практически можно создавать полностью динамизированные кадры со сложными взаимосвязями на уровне пользователя.

Между виджетами на различных уровнях иерархии выстраиваются достаточно сложные наследственные связи, которые определяются возможностью использования одних виджетов другими, начиная с библиотечного виджета и заканчивая виджетом сеанса. Для разъяснения этих особенностей взаимодействия на рис. 3.1.2 изображена исчерпывающая карта «использующего» наследования.



Терминальный виджет – Конечный элемент визуализации, или примитив. На стороне визуализации приобретает соответствующий визуальный образ.

Библиотечный виджет – Хранимый библиотекой виджет. Обязательно наследует визуальный образ терминального виджета и переопределяет его данные. Наследование терминального виджета может быть как прямое, так и посредством нескольких промежуточных элементов.

Контейнерный виджет библиотеки – Фактически является ссылкой на другой виджет в библиотеке (LW2.W1 -> LW1) или ссылку контейнера библиотеки (LW3.W1 -> LW2.W1).

Страница проекта – Элемент интерфейса визуализации и управления (ВУ) - страница, используется для построения иерархического интерфейса ВУ конечного пользователя.

Страничный виджет – Элемент страницы, доопределяющий данные библиотечного виджета к нуждам страницы проекта.

Страница сеанса – Страница сеанса для исполнения страницы проекта в контексте цельного интерфейса ВУ.

Виджет сеанса – Элемент конечной визуализации. Выстраиваются в иерархическую зависимость, соответствующую наследованию терминальных виджетов в контейнерных виджетах библиотеки виджетов и проекта.

Рис.3.1.2 Карта «использующего» наследования компонентов концепции/движка

На уровне сеансов виджет содержит кадр значений процедуры обсчёта. Этот кадр инициируется и используется в случае наличия процедуры обсчёта. В момент инициализации создаётся перечень параметров процедуры и выполняется компиляция процедуры с этими параметрами в модуле, реализующем выбранный язык программирования и закодированным полным именем виджета.

Скомпилированная функция подключается к кадру значений процедуры обсчёта. Далее выполняется вычисление с периодичностью сеанса.

Вычисление и обработка виджета в целом выполняется в следующей последовательности:

- выбирается события, доступные на момент вычисления из атрибута “event” виджета;
- события загружаются в параметр “event” кадра вычисления;
- загружаются значения по входным связям в кадр вычисления;
- загружаются значения специальных переменных в кадр вычисления (f_frq, f_start и f_stop);
- загружаются значения выбранных параметров виджета в кадр вычисления;
- вычисление;
- выгрузка значений кадра вычисления в выбранные параметры виджета;
- выгрузка события из параметра “event” кадра вычисления;
- обработка событий и передача необработанных на уровень выше.

3.2. Проект

Непосредственная конфигурация и свойства конечного интерфейса визуализации содержаться в проекте интерфейса визуализации СВУ. Может быть создано множество проектов интерфейсов визуализации.

Каждый проект включает кадры из библиотек кадров/виджетов. Кадр предоставляет инструмент для привязки динамики к описанным в нём свойствам. Все свойства кадра могут быть связаны с динамикой или разрешены константами, а могут выступать в роли шаблона для формирования производных страниц. Фактически каждый кадр может содержать множество страниц с собственной динамикой. Данный механизм позволяет предельно упростить процесс создания однотипных кадров инженером АСУ-ТП или пользователем системы OpenSCADA для простого мониторинга. Примером таких однотипных кадров могут быть: группы контуров, группы графиков, протоколы и различные сводные таблицы. Мнемосхемы технологических процессов редко подпадают под такую схему и будут формироваться прямо в описании кадра.

Для предоставления возможности создания сложных иерархических интерфейсов ВУ кадры, помещённые в проект, могут группироваться путём именования в иерархическом виде и соответствующей визуализации в виде дерева. В придачу к этому предусматривается механизм ассоциативного описания вызова кадров посредством регулярных выражений.

Пример иерархического представления компонентов проекта классического интерфейса ВУ технологического процесса с описанием выражений стандартных вызовов приведен на рис. 3.2.

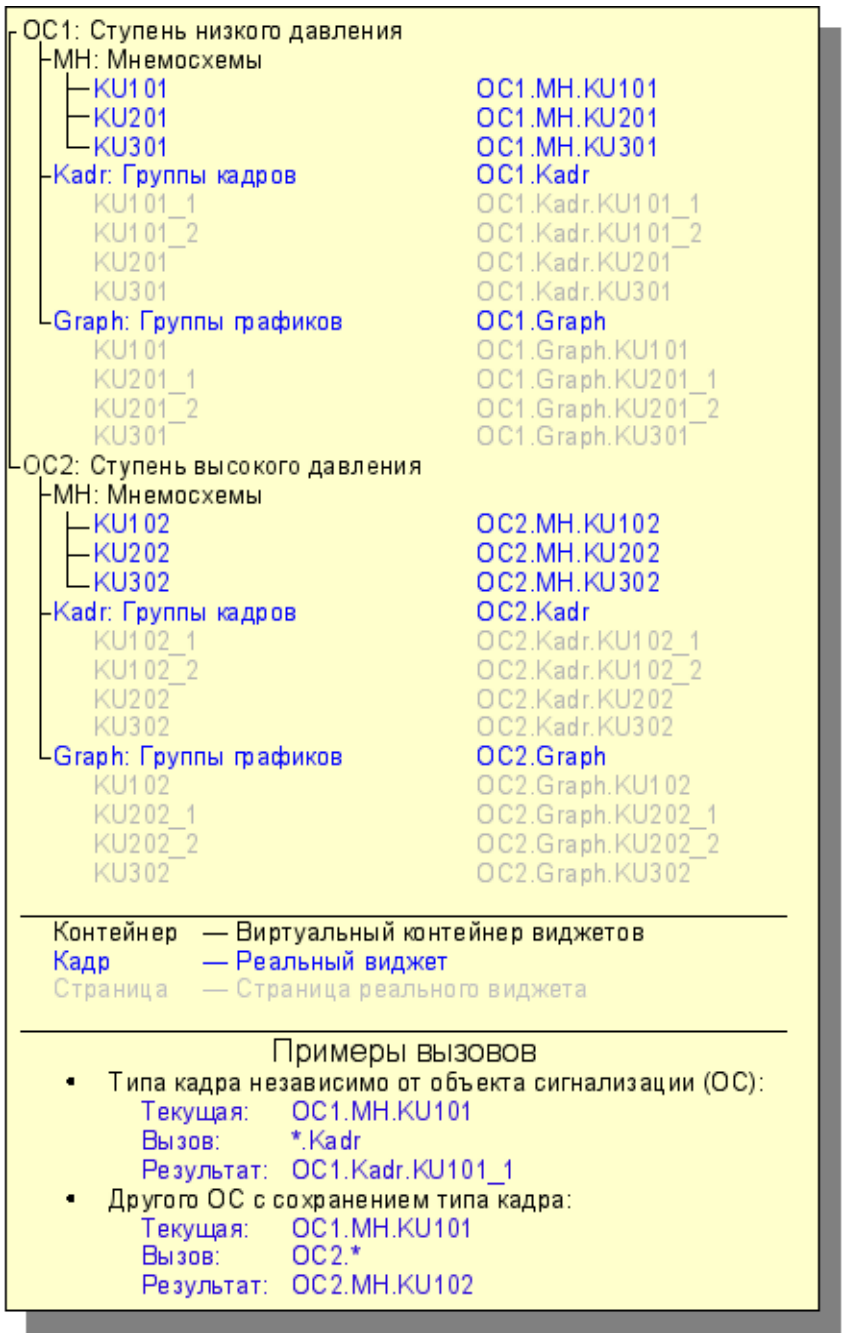


Рис.3.2 Иерархическое представления компонентов проекта классического интерфейса ВУ технологического процесса.

В соответствии с рис.3.1.2 объекты сессии проекта наследуются от абстрактного объекта “Widget” и используют соответствующие объекты проекта. Так, сессия («Session”) использует проект («Project”) и формирует развёрнутое дерево на основе него. Страница проекта “Page” прямо используется страницей сессии «SessPage". Остальные объекты («SessWdg") разворачиваются в соответствии с иерархией элементов страницы (рис.3.1.2).

В дополнение к стандартным свойствам абстрактного виджета («Widget») элементы страницы и сами страницы сессии получают свойства: хранения кадра значений вычислительной процедуры, обсчёта процедур и механизм обработки событий. Страницы сессии в дополнение ко всему содержат контейнер следующих по иерархии страниц. Сессия в целом обсчитывается с указанной периодичностью и в последовательности:

- «Страница верхнего уровня» -> «Страница нижнего уровня»
- «Виджет нижнего уровня» -> «Виджет верхнего уровня»

Такая политика позволяет обходить страницы в соответствии с иерархией, а событиям в виджетах всплывать на верх за одну итерацию.

В сессии реализована поддержка специальных свойств страниц:

- *Container* — страница является контейнером для нижележащих страниц;
- *Template* — страница является шаблоном для нижележащих страниц;
- *Empty* — пустая, неактивная, страница; это свойство используется совместно со свойством *Container* для организации логических контейнеров.

На основе этих свойств реализованы следующие типы страниц:

- *Standard* — Стандартная страница (не установлено ни одно из свойств). Является полноценной конечной страницей.
- *Container* — Полноценная страница со свойством контейнера (*Container*).
- *Logical container* — Логический контейнер, фактически не являющийся страницей (*Container*|*Empty*). Выполняет свойство промежуточного и группирующего элемента в дереве страниц.
- *Template* — Страница шаблон (*Template*). Чистая шаблонная страница используется для описания общих свойств и доопределения их в частном порядке во вложенных страницах.
- *Container and template* — Страница шаблон и контейнер (*Template*|*Container*). Совмещает функции шаблона и контейнера.

Переключение, открытие, замещение и навигация по страницам реализованы на основе обработки событий по сценарию в атрибуте активного виджета “evProc”. Сценарий этого атрибута записывается в виде списка команд с синтаксисом: **<event>:<evSrc>:<com>:<prm>**. Где:

- *event* — ожидаемое событие;
- *evSrc* — путь вложенного виджета-источника события;
- *com* — команда сессии;
- *prm* — параметр команды;

Реализованы следующие команды:

- *open* — Открытие страницы. Открываемая страница указывается в параметре <prm> как на прямую, так и в виде шаблона (например: /pg_so/1/*/*).
- *next* — Открытие следующей страницы. Открываемая страница указывается в параметре <prm> в виде шаблона (например: /pg_so/*/*/$).
- *prev* — Открытие предыдущей страницы. Открываемая страница указывается в параметре <prm> в виде шаблона (например: /pg_so/*/*/$).

Специальные символы шаблона расшифровываются следующим образом:

- *pg_so* — прямое имя требуемой страницы с префиксом. Требует обязательного соответствия и используется для идентификации предыдущей открытой страницы;
- *1* — имя новой страницы в общем пути без префикса. Игнорируется при обнаружении предыдущей открытой страницы;
- * — страница берётся с имени предыдущей открытой страницы или подставляется первая доступная страница, если предыдущая открытая страница отсутствует;
- *$* — указывает на место открытой страницы, относительно которой необходимо искать следующую или предыдущую.

Для понимания работы механизма шаблонов приведём несколько реальных примеров:

- *Переключение объекта сигнализации:*

Команда: open:/pg_so/2/*/* Было: /pg_so/pg_1/pg_mn/pg_1 Стало: /pg_so/pg_2/pg_mn/pg_1

- *Переключение вида:*

Команда: open:/pg_so/*/gkadr/* Было: /pg_so/pg_1/pg_mn/pg_1 Стало: /pg_so/pg_1/pg_gkadr/pg_1

- *Следующая/предыдущая страница вида:*

Команда: next:/pg_so/*/*/$ Было: /pg_so/pg_1/pg_mn/pg_1 Стало: /pg_so/pg_1/pg_mn/pg_2

В качестве примера приведём сценарий обеспечения работы главной страницы интерфейса пользователя:

```
ws_BtPress:/prev:prev:/pg_so/*/*/$
ws_BtPress:/next:next:/pg_so/*/*/$
ws_BtPress:/go_mn:open:/pg_so/*/mn/*
ws_BtPress:/go_graph:open:/pg_so/*/ggraph/*
ws_BtPress:/go_cadr:open:/pg_so/*/gcadr/*
ws_BtPress:/go_view:open:/pg_so/*/gview/*
ws_BtPress:/go_doc:open:/pg_so/*/doc/*
```

```
ws_BtPress:/go_resg:open:/pg_so/rg/rg/*
ws_BtPress:/so1:open:/pg_so/1/*/*
ws_BtPress:/so2:open:/pg_so/2/*/*
ws_BtPress:/so3:open:/pg_so/3/*/*
ws_BtPress:/so4:open:/pg_so/4/*/*
ws_BtPress:/so5:open:/pg_so/5/*/*
ws_BtPress:/so6:open:/pg_so/6/*/*
ws_BtPress:/so7:open:/pg_so/7/*/*
ws_BtPress:/so8:open:/pg_so/8/*/*
ws_BtPress:/so9:open:/pg_so/9/*/*
ws_BtPress:*:open:/pg_control/pg_terminator
```

В связке с вышеописанным механизмом на стороне визуализации (RunTime) построена логика регулирующая, каким образом открывать страницы. Логика построена на следующих атрибутах базового элемента “Box”:

- *pgOpen* — Признак «Страница открыта».
- *pgNoOpenProc* — Признак «Исполнять страницу даже если она не открыта».
- *pgOpenSrc* — Содержит адрес виджета или страницы, открывшей текущую. В случае вложенного контейнерного виджета здесь содержится адрес включаемой страницы. Для открытия страницы из скрипта достаточно здесь указать адрес виджета-источника открытия.
- *pgGrp* — Группа страниц. Используется для связки контейнеров страниц со страницами в соответствии с общей группой.

Логика определения способа открытия страниц работает следующим образом:

- если страница имеет группу “main” или совпадает с группой страницы в главном окне или нет страницы на главном окне, то открывать страницу в главном окне;
- если страница имеет группу, которая совпадает с группой одного из контейнеров текущей страницы, то открыть в этом контейнере;
- если источник открытия страницы совпадает с текущей страницей, то открыть в виде дополнительного окна над текущей страницей;
- передать вызов на запрос открытия дополнительным окнам с обработкой у каждого по первым трем пунктам;
- если никто из родственных окон не открыл новую страницу, то открыть её как родственное окно главного окна.

3.3. Темы отображения

Известно, что человек может иметь индивидуальные особенности в восприятии графической информации. Если эти особенности не учитывать, то можно получить неприятие и отторжение пользователя к интерфейсу ВУ. Такое неприятие и отторжение может привести к фатальным ошибкам при управлении ТП, а также травмировать человека постоянной работой с таким интерфейсом. В SCADA системах приняты соглашения, которые регламентируют требования по созданию унифицированного интерфейса ВУ нормально воспринимаемого большинством людей. При этом практически отсутствует учёт особенностей людей с некоторыми отклонениями.

С целью учесть это обстоятельство и предоставить возможность централизованно и просто изменять визуальные свойства интерфейса в модуле планируется реализация менеджера тем интерфейса визуализации.

Пользователем может быть создано множество тем, каждая из которых будет хранить цветовые, шрифтовые и другие свойства элементов кадра. Простая смена темы позволит быстро преобразить интерфейс ВУ, а возможность назначения индивидуальной темы в профиле пользователя позволит учесть его индивидуальные особенности.

Для поддержки этой возможности при создании кадров необходимо указывать цвета, шрифты и другие свойства визуализации не прямо, а через имя свойства/группы в теме.

3.4. События, их обработка и карты событий

Учитывая спектр задач для которых может использоваться система OpenSCADA, нужно предусмотреть и механизм управления интерактивными пользовательскими событиями. Это связано с тем, что при решении отдельных задач встраиваемых систем устройства ввода и управления могут значительно отличатся. Впрочем достаточно взглянуть на обычную офисную клавиатуру и клавиатуру ноутбука что бы снять любые сомнения о необходимости менеджера событий.

Менеджер событий должен работать, используя карты событий. Карта событий – это список именованных событий с указанием его происхождения. Происхождением события может быть клавиатура, манипулятор мыши, джойстик и т.д. При возникновении события менеджер событий ищет его в активной карте и сопоставляет с именем события. Сопоставленное имя события помещается в очередь на обработку. Виджеты в этом случае должны обрабатывать полученную очередь событий.

Активная карта событий указывается в профиле каждого пользователя или устанавливается по умолчанию.

В целом предусмотрены четыре типа событий:

- события образов СВУ (префикс: ws_), например, событие нажатия кнопки – *ws_BtPress*;
- клавишные события (префикс: key_) – все события от клавиатуры и мыши в виде – *key_presAlt1*;
- пользовательские события (префикс: usr_) генерируются пользователем в процедурах обсчёта виджетов;
- мапированные события (префикс: map_) – события, полученные из карты событий.

Само событие представляет мало информации, особенно если его обработка происходит на уровнях выше. Для однозначной идентификации события и его источника событие в целом записывается следующим образом: «ws_BtPress:/curtime». Где:

ws_BtPress — событие;
/curtime — путь к дочернему элементу, сгенерировавшего событие.

В таблице 3.4 приведён перечень стандартных событий, поддержка которых должна быть обеспечена в визуализаторах СВУ.

Таблица 3.4. Стандартные события

Id	**Описание**
Клавиатурные события: **key_[pres|rels][Ctrl|Alt|Shift]{Key}**	
*SC#3b	Скан код клавиши.
*#2cd5	Код не именованной клавиши.
*Esc	“Esc”.
*BackSpace	Удаления предыдущего символа – “<--».
*Return, *Enter	Ввод – “Enter”.
*Insert	Вставка – “Insert”.
*Delete	Удаление – “Delete”.
*Pause	Пауза – “Pause”.
*Print	Печать экрана – “Print Screen”.
*Home	Дом – “Home”.
*End	Конец – “End”.
*Left	Влево – “<-».
*Up	Вверх – "^".
*Right	Вправо – “->".
*Down	Вниз – "\/".
*PageUp	Страницы вверх – «PageUp".
*PageDown	Страницы вниз – «PageDown".
*F1 – *F35	Функциональная клавиша от “F1” до “F35”.
*Space	Пробел – “ «.
*Apostrophe	Апостраф – "`".
*Asterisk	Звёздочка на дополнительном поле клавиатуры – "*".
*Plus	Плюс на дополнительном поле клавиатуры – "+".
*Comma	Запятая – “,".
*Minus	Минус – “-".
*Period	Точка – ".".
*Slash	Наклонная черта – "\".
*0 – *9	Цифра от “0” до “9”.
*Semicolon	Точка с запятой – “;".
*Equal	Равно – "=".
*A – *Z	Клавиши букв латинского алфавита от “A” до “Z”.
*BracketLeft	Левая квадратная скобка – "[".
*BackSlash	Обратная наклонная линия – «/».
*BracketRight	Правая квадратная скобка – "]".
*QuoteLeft	Левая кавычка – “"".
События клавиатурного фокуса.	
ws_FocusIn	Фокус получен виджетом.
ws_FocusOut	Фокус утерян виджетом.
Мышиные события:	
key_mouse[Pres|Rels][Left|Right|Midle]	Нажата/отпущена кнопка мыши.
key_mouseDblClick	Двойное нажатие левой кнопки мыши.
События примитива элементарной фигуры **ElFigure***:*	
ws_Fig{n}[Left|Right|Midle]	Активация фигуры {n} клавишей мыши.

Id	**Описание**
События примитива элементов формы **FormEl***:*	
ws_LnAccept	Установлено новое значение в строке ввода.
ws_TxtAccept	Изменено значение редактора текста.
ws_ChkChange	Состояние флажка изменено.
ws_BtPress	Кнопка нажата.
ws_BtRelease	Кнопка отпущена.
ws_BtToggleChange	Изменена вдавленность кнопки.
ws_CombChange	Изменено значение поля выбора.
ws_ListChange	Изменен текущий элемент списка.
ws_SliderChange	Изменение положения слайдера.
События примитива медиа-контента **Media***:*	
ws_MapAct{n}[Left|Right|Midle]	Активирована медиа-область с номером {n} клавишей мыши.

События являются основным механизмом уведомления и активно используются для осуществления взаимодействия с пользователем. Для обработки событий предусмотрены два механизма: сценарии управления открытием страниц и вычислительная процедура виджета.

Механизм «Сценарии управления открытием страниц» основан на базовом атрибуте виджета “evProc” и детально описан в разделе 3.2.

Механизм «Обработка событий с помощью вычислительной процедуры виджета» основан на атрибуте “event” и пользовательской процедуре вычисления на одном из языков пользовательского программирования OpenSCADA. События по мере поступления аккумулируются в атрибуте “event” до момента вызова вычислительной процедуры. Вычислительная процедура вызывается с указанной периодичностью вычисления виджета и получает значение атрибута “event” в виде списка событий. В процедуре вычисления пользователь может: проанализировать, обработать и исключить обработанные события из списка, а также добавить в список новые события. Оставшиеся после исполнения процедуры события, анализируются на предмет соответствия условиям вызова сценарием первого механизма после чего оставшиеся события передаются на верхний по иерархии виджет для обработки им, при этом осуществляется коррекция пути событий в соответствии с иерархией проникновения события.

Содержимое атрибута “event” является списком событий формата **<event>:<evSrc>**, с событием в отдельной строке. Приведём пример процедуры обработки событий на Java-подобном языке пользовательского программирования OpenSCADA:

```
using Special.FLibSYS;
ev_rez = "";
off = 0;
while(true)
{
      sval = strParse(event,0,"\n",off);
      if( sval == "" ) break;
      else if( sval == "ws_BtPress:/cvt_light" ) alarmSt = 0x1000001;
      else if( sval == "ws_BtPress:/cvt_alarm" ) alarmSt = 0x1000002;
      else if( sval == "ws_BtPress:/cvt_sound" ) alarmSt = 0x1000004;
      else ev_rez+=sval+"\n";
}
event=ev_rez;
```

3.5. Сигнализация

Важным элементом любого интерфейса визуализации является уведомление пользователя про нарушения – сигнализация. Для упрощения восприятия, а также в виду тесной связности визуализации и уведомления (как правило уведомление дополняет визуализацию) решено интегрировать интерфейс уведомления в интерфейс визуализации. Для этого во всех виджетах предусматриваются два дополнительных атрибута (уровня сеанса): 'alarm' и 'alarmSt'. Атрибут 'alarm' используется для формирования сигнала виджетом в соответствии с его логикой, а атрибут 'alarmSt' используется для контроля за фактом сигнализации ветви дерева сеанса проекта.

Атрибут 'alarm' является строкой и имеет следующий формат: *{lev|categ|message|type|tp_arg}* Где:

- *lev* — уровень сигнализации: число от 0 до 255;
- *categ* — категория сигнала: параметр подсистемы сбора, объект, путь или комбинация.
- *message* — сообщение сигнализации; для помещение в строку статуса, отображения в протоколе и помещения в архив сообщений;
- *type* — типы уведомления (визуальное, гудок и речь): формируется в виде целого числа содержащего флаги способов уведомлений:
  - *0x01* — визуальная;
  - *0x02* — гудок, часто производится через PC-speaker;
  - *0x04* — звуковой сигнал из файла звука или синтез речи; если в <tp_arg> указано имя ресурса звукового файла, то воспроизводится именно он, иначе выполняется синтез речи из текста указанного в <message>.
- *tp_arg* — аргумент типа; используется в случае осуществления звуковой сигнализации для указания ресурса звукового сигнала (файл звукового формата).

Атрибут 'alarmSt' является целым числом, которое отражает максимальный уровень сигнала и факт квитации ветви дерева сеанса проекта. Формат числа имеет следующий вид:

- первый байт (0–255) характеризует уровень сигнала ветви;
- второй байт указывает тип уведомления (также как и в атрибуте 'alarm');
- третий байт указывает тип несквитированного уведомления (также как и в атрибуте 'alarm');
- первый бит четвёртого байта имеет специальное назначение, установка этого бита является фактом квитации уведомлений указанных первым байтом.

Формирование сигнала и получение его визуализатором. Формирование сигнала производится самим виджетом путём установки собственного атрибута 'alarm' нужным образом, и в соответствии с ним устанавливается атрибут 'alarmSt' текущего и вышестоящих виджетов. Визуализаторы получают уведомление о сигнале с помощью стандартного механизма уведомления об изменении атрибутов виджетов.

Такой механизм предоставляет возможность формировать интерфейсы сигнализации как на уровне подсистемы «Сбор данных», так и прямо на уровне представления.

Учитывая то, что обработка условий сигнализации осуществляется в виджетах, страницы, содержащие объекты сигнализации, должны исполняться в фоне, не зависимо от открытости их в данный момент. Это осуществляется путём установки флага исполнения страницы в фоне.

Хотя механизм сигнализации и построен в среде визуализации, возможность формирования невизуальных элементов сигнализации остаётся например, путём создания страницы которая никогда не будет открываться.

Квитация Квитация производится путём указания корня ветви виджетов и типов уведомления. Это позволяет реализовать квитацию на стороне визуализатора как по группам например, по объектам сигнализации, так и индивидуально по объектам. При этом можно независимо квитировать разные типы сигнализаций. Установка квитации производится простой модификацией атрибута 'alarmSt'.

Пример скрипта для работы с сигналами приведён ниже:

```
//Выделение факта наличия сигнализаций разных способов уведомления
```

```
cvt_light_en = alarmSt&0x100;
cvt_alarm_en = alarmSt&0x200;
cvt_sound_en = alarmSt&0x400;
//Выделение факта наличия несквитированных сигнализаций разных способов уведомления
cvt_light_active = alarmSt&0x10000;
cvt_alarm_active = alarmSt&0x20000;
cvt_sound_active = alarmSt&0x40000;
//Обработка событий кнопок квитации и квитация разных способов уведомлений
ev_rez = "";
off = 0;
while(true)
{
      sval = strParse(event,0,"\n",off);
      if( sval == "" ) break;
      else if( sval == "ws_BtPress:/cvt_light" ) alarmSt = 0x1000001;
      else if( sval == "ws_BtPress:/cvt_alarm" ) alarmSt = 0x1000002;
      else if( sval == "ws_BtPress:/cvt_sound" ) alarmSt = 0x1000004;
      else ev_rez+=sval+"\n";
}
event=ev_rez;
```

3.6. Управление правами

Для разделения доступа к интерфейсу ВУ и его составляющим каждый виджет содержит информацию о владельце его группе и правах доступа. Права доступа записываются, как принято в системе OpenSCADA, в виде триады: <пользователь><группа><остальные>, где каждый элемент состоит из трёх признаков доступа. Для элементов СВУ принята следующая их интерпретация:

- *“r”* — право на просмотр виджета;
- *“w”* — право на контроль над виджетом.

В режиме разработки используется простая схема доступа «root.UI:RWRWR_", что означает – все пользователи могут открывать и просматривать библиотеки, их компоненты и проекты; а редактировать могут все пользователи группы “UI” (пользовательские интерфейсы).

В режиме исполнения работают права, описанные в компонентах интерфейса.

3.7. Связывание с динамикой

Для предоставления актуальных данных в интерфейс визуализации должны использоваться данные подсистемы «Сбор данных (DAQ)". Природа этих данных следующая:

1. параметры, содержащие некоторое количество атрибутов;
2. атрибуты параметра могут предоставлять данные четырёх типов: Логический, Целый, Вещественный и Строковый;
3. атрибуты параметра могут иметь историю (архив);
4. атрибуты параметра могут быть на чтение, запись и с полным доступом.

Учитывая первый пункт, нужно обеспечить возможность группового назначения ссылки. Для этого используем концепцию логического уровня.

В соответствии с пунктом 2 связи обеспечивают прозрачное преобразование типов и не требуют специальной конфигурации.

Для удовлетворения возможности доступа к архивам в соответствии с пунктом 3 связи выполняют проверку типа атрибута и, в случае подключения к «Адресу», в значение помещается адрес связи.

В терминах СВУ, динамические связи и конфигурация динамики являются одним процессом, для описания конфигурации которого предусматривается вкладка «Обработка» виджетов (рис.3.7.а). Вкладка содержит таблицу конфигурации свойств атрибутов виджета и текст процедуры вычисления виджета.

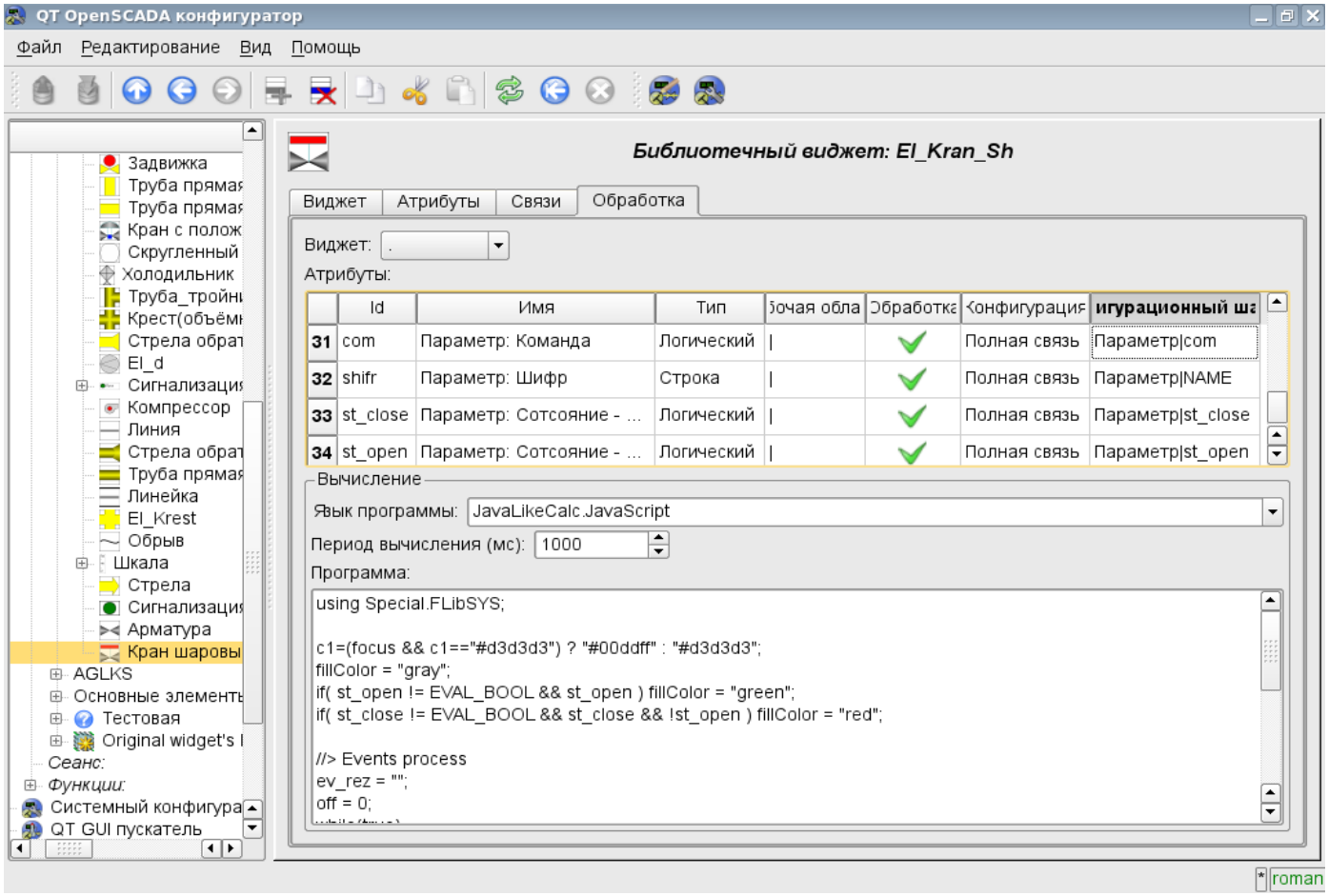


Рис. 3.7.а Вкладка «Обработка» страницы конфигурации виджета.

Кроме полей конфигурации атрибутов в таблице предусматривается колонка «Обработка», для избирательного использования атрибутов виджетов в вычислительной процедуре виджета, и колонки «Конфигурация» и «Конфигурационный шаблон» для описания конфигурации связей.

Колонка «Конфигурация» позволяет указать тип связи для атрибута виджета:

- *Постоянная* — во вкладке связей виджета появляется поле указания постоянной например, особого цвета или заголовка для шаблонных кадров;
- *Входная связь* — связь с динамикой только для чтения;
- *Выходная связь* — связь с динамикой только для записи;
- *Полная связь* — полная связь с динамикой (чтение и запись).

Колонка «Конфигурационный шаблон» позволяет описать группы динамических атрибутов. Например, это могут быть разные типы параметров подсистемы “DAQ”. Кроме того, при корректном формировании этого поля работает механизм автоматического назначения атрибутов при указании только параметра подсистемы “DAQ”, что упрощает и ускоряет процесс конфигурации. Значение этой колонки имеет следующий формат: **<Параметр>|<Идентификатор>**, где:

- *<Параметр>* — группа атрибута;
- *<Идентификатор>* — идентификатор атрибута, именно это значение сопоставляется с атрибутами параметров DAQ при автоматическом связывании после указания групповой связи.

Установка связей может быть нескольких типов, который определяется префиксом:

- *val:* — Прямая загрузка значения через механизм связей. Например, связь: "val:100" загружает в атрибут виджета значение 100. Часто используется в случае отсутствия конечной точки связи с целью прямой установки значения.
- *prm:* — Связь на атрибут параметра или параметр в целом, для группы атрибутов, подсистемы «Сбор данных». Например, связь "prm:/LogicLev/experiment/Pi/var" осуществляет доступ атрибута виджета к атрибуту параметра подсистемы «Сбор данных».

- *wdg:* — Связь на атрибут другого виджета или виджет в целом для группы атрибутов. Например, связь "wdg:/ses_AGLKS/pg_so/pg_1/pg_ggraph/pg_1/a_bordColor" осуществляет доступ атрибута одного виджета к атрибуту другого. На данный момент этот тип связи не предназначен для установки пользователем вручную, а устанавливается автоматически в режиме динамического связывания!

Обработка связей происходит с периодичностью вычисления виджета в порядке:

- Получение данных входных связей.
- Выполнение вычисления скрипта.
- Передача значений по выходным связям.

На рис. 3.7.b представлена вкладка связей с групповым назначением атрибутов путём указания только параметра, а на рис. 3.7.c с индивидуальным назначением атрибутов.

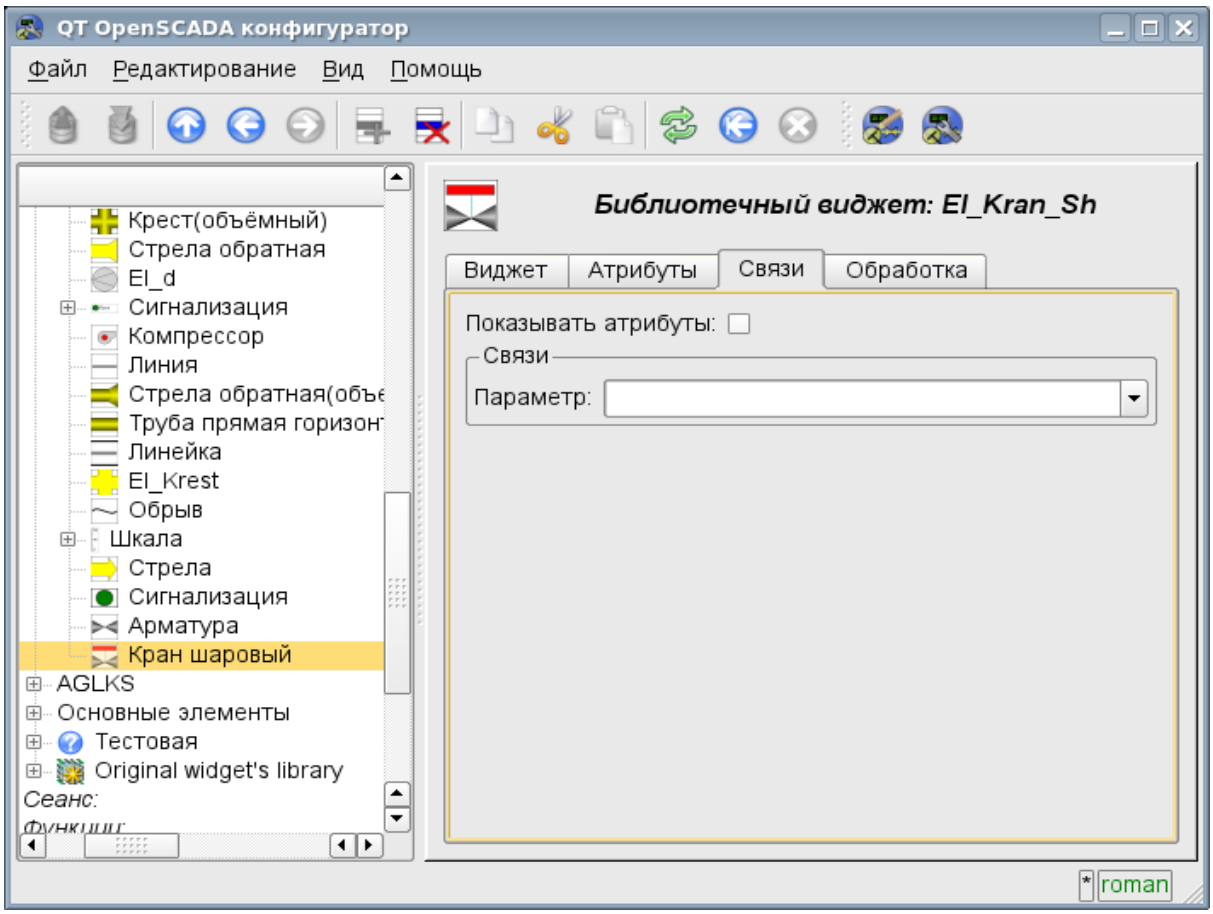


Рис. 3.7.b Вкладка «Связи» страницы конфигурации виджета с групповым назначением атрибутов путём указания только параметра.

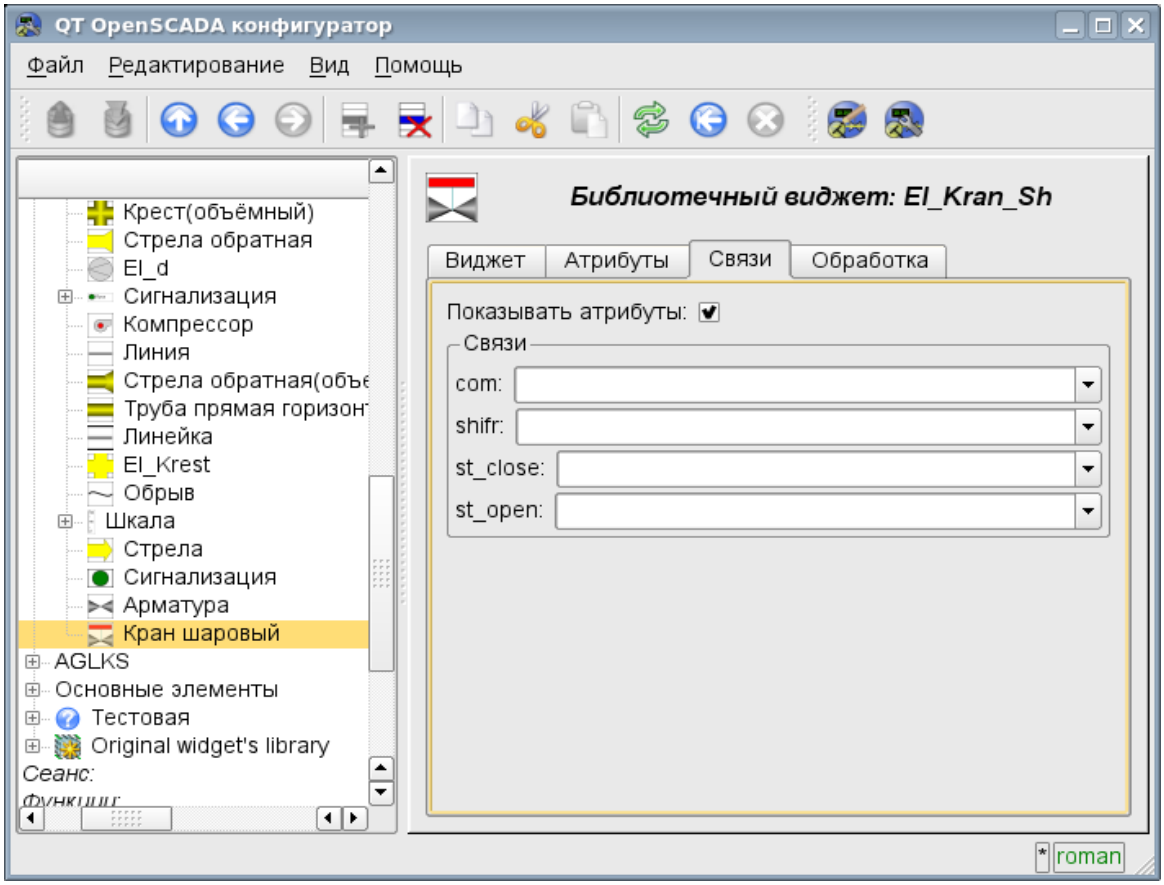


Рис. 3.7.c Вкладка «Связи» страницы конфигурации виджета с индивидуальным назначением атрибутов.

При размещении виджета, содержащего конфигурацию связей, в контейнер виджетов все связи исходного виджета добавляются в список результирующих связей контейнера виджетов (рис. 3.7.d)

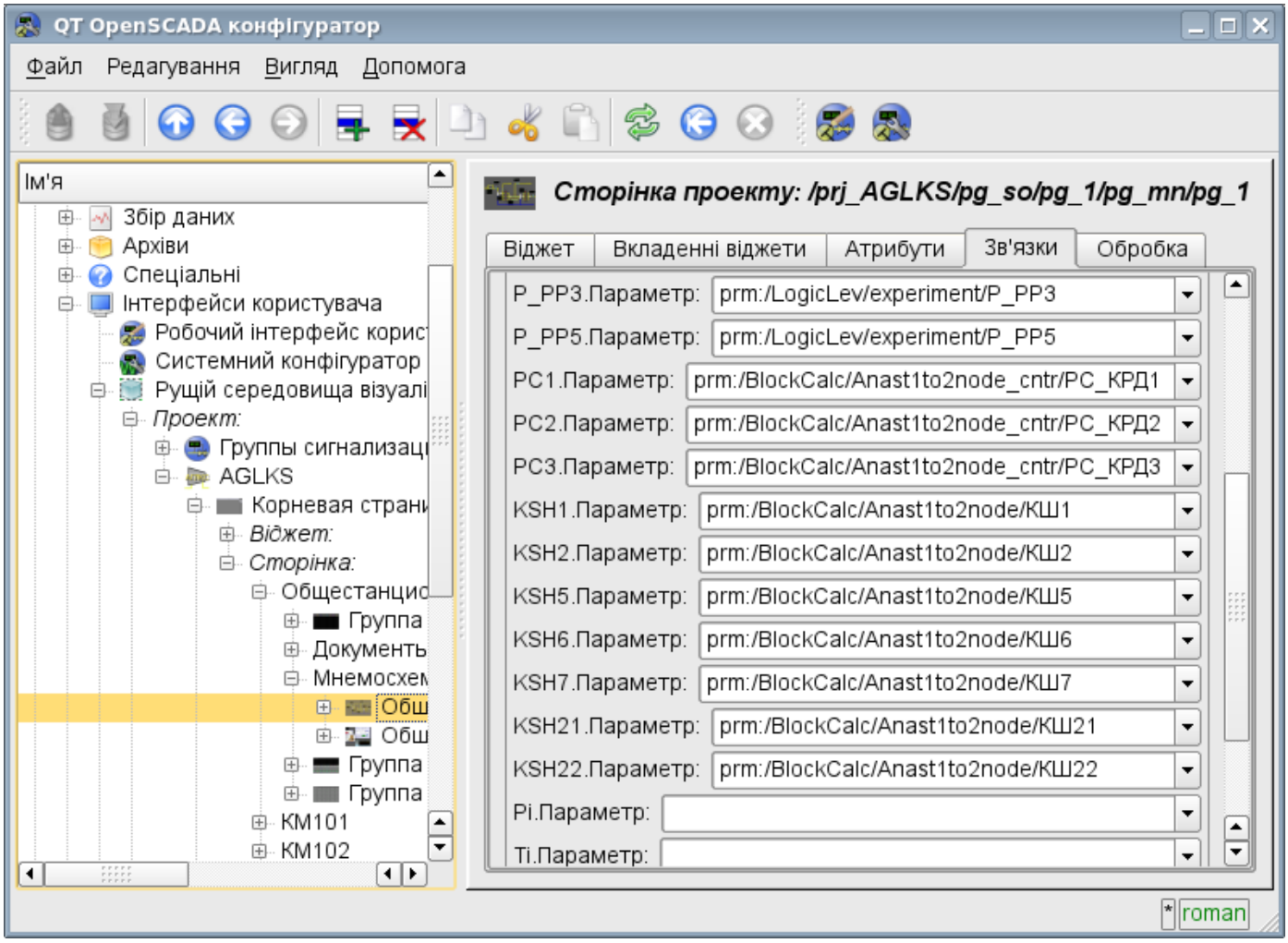


Рис. 3.7.d Вкладка «Связи» страницы конфигурации контейнера виджетов, включающего виджеты со связями.

Из вышесказанного видно, что связи устанавливаются пользователем в процессе конфигурации интерфейса. Однако, для предоставления возможности создания кадров общего назначения с функцией предоставления детализированных данных разных источников одного типа необходим механизм динамической установки связей. Такой механизм предусматривается посредством зарезервированного ключевого идентификатора '**<page>**' группы атрибутов связей у кадров общего назначения и динамическое назначение связей с идентификатором '**<page>**' в процессе открытия кадра общего назначения сигналом от другого виджета.

Рассмотрим пример, когда имется кадр общего назначения «Панель контроля графиком» и множество «Графиков» на разных кадрах. «Панель контроля графиком» имеет связи с шаблонами:

- tSek --> '<page>|tSek'
- tSize --> '<page>|tSize'
- trcPer --> '<page>|trcPer'
- valArch --> '<page>|valArch'

При этом каждый виджет «График» имеет атрибуты tSek, tSize, trcPer и valArch. В случае вызова сигнала открытия «Панели контроля графиком» из любого виджета «График» происходит связывания атрибутов «Панели контроля графиком» в соответствии атрибуту указанного в шаблоне с атрибутом виджета «График». Как результат, все изменения на «Панели контроля графиком» будут отражаться на графике посредством связи.

В случае наличия у виджета «График» внешних связей на параметры подсистемы «Сбор данных», связи «Панели контроля графиком» будут устанавливаться на внешний источник. Кроме этого, если у «Панели контроля графиком» будут заявлены связи на отсутствующие непосредственно у виджета «График» атрибуты, то будет производится поиск на наличие таких атрибутов у внешнего источника, первого на который установлена прямая связь, выполняя, тем

самым, дополнение недостающих связей.

Для наглядного изображения этого механизма приведена таблица 3.7.

Таблица 3.7. Механизм динамической линковки.

Атрибуты «Панели контроля графиком» (шаблон динамической связи)	Атрибуты «Графика»	Атрибуты внешнего «Параметра»	Результирующая связь или значение связующегося атрибута
tSek (<page>|tSek)	tSek	–	«График».tSek
tSize (<page>|tSize)	tSize	–	«График».tSize
trcPer (<page>|trcPer)	trcPer	–	«График».trcPer
valArch (<page>|valArch)	valArch	–	«График».valArch
var (<page>|var)	var	var	«Параметр».var
ed (<page>|ed)	–	ed	«Параметр».ed
max (<page>|max)	–	–	EVAL
min (<page>|min)	–	–	EVAL

3.8. Примитивы виджетов

Любой вновь создаваемый виджет основывается на одном из нескольких примитивов(конечный элемент визуализации) путём установки родственной связи как прямо на примитив, так и посредством нескольких промежуточных пользовательских виджетов. Каждый из примитивов содержит механизм (логику) модели данных. Экземпляр виджета хранит значения свойств конфигурирования примитива специально для себя.

В задачи интерфейса визуализации входит поддержка и работа с моделью данных примитивов виджетов. Примитивы виджетов должны быть тщательно проработаны и унифицированы с целью охватить как можно больше возможностей в как можно меньшем количестве слабо связанных друг с другом по назначению примитивов.

В таблице 3.8.а приведён перечень примитивов виджетов(базовых элементов отображения).

Таблица 3.8.а. Библиотека примитивов виджетов(базовых элементов отображения)

Id	**Наименование**	**Функция**
ElFigure	Элементарные графические фигуры	Примитив является основой для отрисовки элементарных графических фигур со всевозможной комбинацией их в одном объекте. Предусматривается поддержка следующих элементарных фигур: • Линия. • Дуга. • Кривая безье. • Заливка замкнутого пространства. Для всех фигур, содержащихся в виджете устанавливаются единые свойства толщины, цвета и т.д., но это не исключает возможность указания вышеперечисленных атрибутов для каждой фигуры отдельно.
FormEl	Элементы формы.	Включает поддержку стандартных компонентов формы: • Редактирование строки. • Редактирование текста. • Флажок. • Кнопка. • Поле выбора из списка. • Список. • Слайдер. • Строка прокрутки.
Text	Текст	Элемент текста(метки). Характеризуется типом шрифта, цветом, ориентацией и выравниванием.

Id	Наименование	Функция
Media	Медиа	Элемент отображения растровых и векторных изображений различных форматов, проигрывания анимированных изображений, проигрывание аудио фрагментов и просмотр видео-фрагментов. Возможно в него стоит включить и поддержку OpenGL!
Diagram	Диаграмма	Элемент диаграммы с поддержкой возможности отображения: нескольких потоков трендов, частотного спектра ... .
Protocol	Протокол	Элемент протокола, визуализатора системных сообщений с поддержкой несколько режимов работы.
Document	Документ	Элемент формирования отчётов, журналов и другой документации на основе доступных в системе данных.
Box	Контейнер	Содержит механизм размещения других виджетов с целью формирования новых более сложных виджетов и страниц конечной визуализации.
Function	Функция API объектной модели OpenSCADA	Невизуальный на стороне исполнения виджет, позволяющий включать вычислительные функции объектной модели OpenSCADA в СВУ.

Каждый примитив и виджет вообще содержит общий набор свойств/атрибутов в составе приведенном в таблице 3.8.b:

Таблица 3.8.b. Общий набор свойств/атрибутов в виджете

Id	Имя	№	Значение
id	Id	–	Идентификатор элемента. Атрибут только для чтения, призванный предоставить информацию об идентификаторе элемента.
path	Path	–	Путь виджета. Атрибут только для чтения и предоставления информации об расположения элемента.
parent	Parent	–	Предок или родитель виджета. Атрибут только для чтения и предоставления информации об расположении предка, от которого наследован виджет.
root	Root	1	Идентификатор виджета-примитива (базового элемента) который лежит в основе образа визуализации виджета.
name	Name	–	Имя элемента. Модифицируемое имя элемента.
dscr	Description	–	Описание элемента. Текстовое поле для прикрепления к виджету краткого описания.
en	Enabled	5	Состояние элемента – «Включен». Отключенный элемент не отображается при исполнении.
active	Active	6	Состояние элемента – «Активный». Активный элементы может получать фокус при исполнении, а значит получать клавиатурные и иные события с последующей их обработкой.
geomX	Geometry:x	7	Геометрия, координата “x” положения элемента.
geomY	Geometry:y	8	Геометрия, координата “y” положения элемента.
geomW	Geometry:width	9	Геометрия, ширина элемента.
geomH	Geometry:height	10	Геометрия, высота элемента.
geomXsc	Geometry:x scale	13	Масштаб элемента по горизонтали.
geomYsc	Geometry:y scale	14	Масштаб элемента по вертикали.
geomZ	Geometry:z	11	Геометрия, координата “z” (уровень) элемента на странице. Также определяет порядок передачи фокуса между активными элементами.

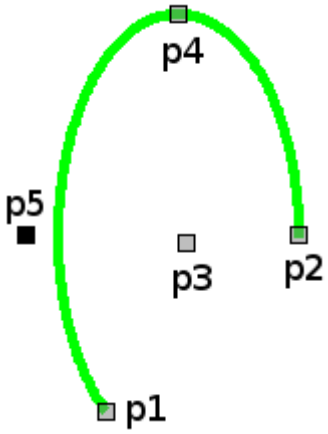
Id	Имя	№	Значение
geomMargin	Geometry:margin	12	Геометрия, поля элемента.
tipTool	Tip:tool	15	Текст краткой помощи или подсказки по данному элементу. Обычно реализуется как всплывающая подсказка при удержании курсора мыши над элементом.
tipStatus	Tip:status	16	Текст информации о состоянии элемента или руководства к действию над элементом. Обычно реализуется в виде сообщения в строке статуса при удержании курсора мыши над элементом.
contextMenu	Context menu	17	Конфигурация собственного контекстного меню элемента. Конфигурация записывается в виде строк пунктов контекстного меню формата: **<Имя пункта>:<ИдСобытия>**. Где: • *<Имя пункта>* — Имя пункта меню. • *<ИдСобытия>* — Идентификатор события, генерируемого виджету (usr_<ИдСобытия>) при выборе пункта меню.
evProc	Events process	–	Атрибут для хранения сценария обработки событий непосредственного управления пользовательским интерфейсом. Сценарий предстовляет собой список команд интерфейсу визуализации, генерируемых при поступлении события (атрибут **event**).
Дополнительные атрибуты для элементов, помещённых в проект в роли страницы.			
pgOpen	Page:open state	–	Признак «Страница открыта».
pgNoOpenProc	Page:no open process	–	Признак «Исполнять страницу даже если она закрыта».
pgOpenSrc	Page:open source	3	Полный адрес страницы открывшей данную.
pgGrp	Page:group	4	Группа страницы.
Дополнительные атрибуты режима исполнения.			
event	Event	–	Специальный атрибут для сбора событий виджета в списке, разделённом новой строкой. Данный атрибут доступен только в сеансе. Доступ к атрибуту защищён ресурсом с целью избежания потери событий. Атрибут всегда доступен в скрипте виджета.
load	Load	-1	Виртуальная команда групповой загрузки данных.
focus	Focus	-2	Специальный атрибут индикации факта получения фокуса активным виджетом. Данный атрибут доступен только в сеансе. Атрибут этого виджета и вложенных виджетов доступен в скрипте виджета.
perm	Permition	-3	Виртуальный атрибут проверки прав активного пользователя на просмотр и контроль над виджетом.

3.8.1. Элементарные графические фигуры (ElFigure)

Примитив является основой для отрисовки элементарных графических фигур со всевозможной комбинацией их в одном объекте. Учитывая широкий спектр всевозможных фигур, которые должен поддерживать примитив, и в тоже время являться достаточно простым в использовании и, по возможности, в реализации, решено было ограничить перечень базовых фигур, используемых для построения результирующих графических объектов до таких фигур: линия, дуга, кривая Безье и заливка замкнутых контуров. Основываясь уже на этих базовых фигурах, можно строить производные фигуры, комбинируя базовые. В рамках примитива существует возможность задания прозрачности цвета в диапозоне [0..255], где “0” – полная прозрачность.

Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.1.

Таблица 3.8.1. Набор дополнительных свойств/атрибутов в примитиве ElFigure

Id	**Имя**	**№**	**Значение**
lineWdth	Line:width	20	Ширина линии.
lineClr	Line:color	21	Цвет линии. (Прозрачность цвета задается следующим способом: red-127 | #ff0000–127, где “127” и есть значение прозрачности.)
lineStyle	Line:style	22	Стиль линии (сплошная, пунктир, точечная).
bordWdth	Border:width	23	Ширина бордюра линии. Нулевая ширина указывает на отсутствие бордюра.
bordClr	Border:color	24	Цвет бордюра.
fillColor	Fill:color	25	Цвет заливки.
fillImg	Fill:image	26	Изображение заливки.
orient	Orientation angle	28	Угол поворота содержимого виджета.
elLst	Element's list	27	Список графических примитивов в формате: • Линия. Форма записи в списке: *<line:p1 | (x | y):p2 | (x | y):[width | w{n}]:[color | c{n}]:[border_width | w{n}]:[border_color | c{n}]:[line_style | s{n}]>* • Дуга. Форма записи в списке: *<arc:p1 | (x | y):p2 | (x | y):p3 | (x | y):p4 | (x | y):p5 | (x | y):[width | w{n}]:[color | c{n}]:[border_width | w{n}]:[border_color | c{n}]:[line_style | s{n}]>*  • Кривая Безье. Форма записи в списке: *<bezier:p1 | (x | y):p2 | (x | y):p3 | (x | y):p4 | (x | y):[width | w{n}]:[color | c{n}]:[border_width | w{n}]:[border_color | c{n}]:[line_style | s{n}]>* • Заливка. Форма записи в списке: *<fill:p1 | (x | y),p2 | (x | y),...,pn | (x | y):[fillClr | c{n}]:[fillImg | i{n}]>*
Атрибуты для каждой точки из списка графических фигур **elLst**			
p{n}x	Point {n}:x	30+n*6	Координата “x” точки {n}.
p{n}y	Point {n}:y	30+n*6+1	Координата “y” точки {n}.
w{n}	Width {n}	30+n*6+2	Ширина {n}.
c{n}	Color {n}	30+n*6+3	Цвет {n}.
i{n}	Image {n}	30+n*6+4	Изображение {n}.
s{n}	Style {n}	30+n*6+5	Стиль {n}.

3.8.2. Элементы формы (FormEl)

Примитив, предназначенный для предоставления стандартных элементов формы в распоряжение пользователя. Общий перечень атрибутов зависит от типа элемента. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.2.

Таблица 3.8.2. Набор дополнительных свойств/атрибутов в примитиве FormEl

Id	**Имя**	**Номер**	**Значение**
elType	Element type	20	Тип элемента (Строка редактирования; Редактор текста; Флажок; Кнопка; Выбор из списка; Список; Слайдер; Полоса прокрутки). От его значения зависит перечень дополнительных атрибутов.
Строка редактирования:			
value	Value	21	Содержимое строки.
view	View	22	Вид строки редактирования (Текст; Комбобокс; Целое; Вещественное; Время; Дата; Дата и время).
cfg	Config	23	Конфигурация строки. Формат значения данного поля для различных видов строки: *Текст* — указывается шаблон ввода в формате библиотеки QT. *Комбобокс* — содержит список значений редактируемого комбобокса. *Целое* — содержит конфигурацию поля ввода целочисленного представления в формате: **<Минимум>:<Максимум>:<Шаг изменения>:<Префикс>:<Суффикс>**. *Вещественное* — содержит конфигурацию поля ввода вещественного представления в формате: **<Минимум>:<Максимум>:<Шаг изменения>:<Префикс>:<Суффикс>:<Число знаков после запятой>**. *Время*, *Дата*, *Дата и время* — формировать дату по шаблону с параметрами: **d** — номер дня (1–31); **dd** — номер дня (01–31); **ddd** — сокращённое наименование дня ('Mon' ... 'Sun'); **dddd** — полное наименование дня ('Monday' ... 'Sunday'); **M** — номер месяца (1–12); **MM** — номер месяца (01–12); **MMM** — сокращённое имя месяца ('Jan' ... 'Dec'); **MMMM** — полное имя месяца ('January' ... 'December'); **yy** — последние две цифры года; **yyyy** — год полностью; **h** — час (0–23); **hh** — час (00–23); **m** — минуты (0–59); **mm** — минуты (00–59); **s** — секунды (0–59); **ss** — секунды (00–59); **AP,ap** — отображать AM/PM или am/pm.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Редактор текста:			
value	Value	21	Содержимое редактора.
wordWrap	Word wrap	22	Автоматический перенос текста по словам.

Id	**Имя**	**Номер**	**Значение**
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Флажок:			
name	Name	26	Имя/метка флажка.
value	Value	21	Значение флажка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Кнопка:			
name	Name	26	Имя, надпись на кнопке.
value	Value	21	Значение для фиксированной кнопки.
img	Image	22	Изображение на кнопке.
color	Color	23	Цвет кнопки.
colorText	Color:text	27	Цвет текста.
checkable	Checkable	24	Признак функционирования как фиксированная кнопка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Выбор из списка:			
value	Value	21	Текущее значение списка.
items	Items	22	Перечень элементов списка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Список:			
value	Value	21	Выбранное значение списка.
items	Items	22	Перечень элементов списка.
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
Слайдер и полоса прокрутки:			
value	Value	21	Положение слайдера.
cfg	Config	22	Конфигурация слайдера в формате: "<VertOrient>:<Min>:<Max>:<SinglStep>:<PageStep>". Где: • *VertOrient* — признак вертикальной ориентации, по умолчанию ориентация горизонтальная; • *Min* — минимальное значение; • *Max* — максимальное значение; • *SinglStep* — размер одиночного шага; • *PageStep* — размер страничного шага.

3.8.3. Элемент текста (Text)

Данный примитив предназначен для вывода простого текста, используемого в роли меток и различных подписей. С целью простого создания частых декоративных оформлений примитив должен поддерживать обвод текста рамкой. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.3.

Таблица 3.8.3. Набор дополнительных свойств/атрибутов в примитиве Text

Id	**Имя**	**№**	**Значение**
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.
bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None;Dotted;Dashed;Solid;Double;Groove;Ridge;Inset;Outset).
font	Font	25	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
color	Color	26	Цвет текста.
orient	Orientation angle	27	Ориентация текста, поворот на угол.
wordWrap	Word wrap	28	Автоматический перенос текста по словам.
alignment	Alignment	29	Выравнивание текста (Вверху слева; Вверху справа; Вверху по центру; Вверху по ширине; Внизу слева; Внизу справа; Внизу по центру; Внизу по ширине; По центру слева; По центру справа; По середине; По центру по ширине).
text	Text	30	Значение текстового поля.
numbArg	Arguments number	40	Количество аргументов.
Атрибуты аргументов			
arg{x}val	Argument {x}:value	50+10*x	Значение аргумента
arg{x}tp	Argument {x}:type	50+10*x+1	Тип аргумента: “Integer”, “Real”, “String”
arg{x}cfg	Argument {x}:config	50+10*x+2	Конфигурация аргумента: • *строка* : **[len]** — ширина строки; • *вещественное*: **[wdth];[form];[prec]** — ширина значения, форма значения («g»,"f"); • *целое*: **[len]** — ширина значения.

3.8.4. Элемент отображения медиа-материалов (Media)

Данный примитив предназначен для проигрывания различных медиа-материалов, начиная от простых изображений и заканчивая полноценными аудио и видео потоками. Учитывая многообразность способов и библиотек проигрывания полноценных аудио и видео потоков, а также достаточно серьёзную трудоёмкость по имплиминтации всех этих механизмов в данный виджет, решено было на первоначальном этапе реализовать только работу с изображениями и простыми анимационными форматами изображений и видео. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.4.

Таблица 3.8.4. Набор дополнительных свойств/атрибутов в примитиве Media

Id	**Имя**	**№**	**Значение**
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.

bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None; Dotted; Dashed; Solid; Double; Groove; Ridge; Inset; Outset).
src	Source	25	Источник медиа-данных.
fit	Fit to widget size	26	Признак «Согласовать содержимое с размером виджета.
type	Type	27	Тип медиа (Image; Movie).
areas	Map areas	28	Количество активных областей.
Атрибуты видеоролика (Movie)			
speed	Play speed	29	Скорость проигрывания, в процентах от оригинальной скорости. Если значение меньше 1%, то проигрывание прекращается.
Активные области			
area{x}shp	Area {x}:shape	40+3*x	Вид области (Rect; Poly; Circle).
area{x}coord	Area {x}:coordinates	40+3*x+1	Координаты областей. Через запятую идут координаты: “x1,y1,x2,y2,xN,yN”
area{x}title	Area {x}:title	40+3*x+2	Заголовок области.

3.8.5. Элемент построения диаграмм/трендов (Diagram)

Данный примитив предназначен для построения различных диаграмм, включая и графики/тренды отображения текущего процесса и архивных данных. На данный момент реализованы следующие типы диаграмм:

- 'График' — позволяет строить одномерные графики из значений параметров подсистемы 'Сбор данных' по времени, а также прямое использование архивных данных для построения графиков. Поддерживается режим отслеживания как текущих значений, так и значений по архиву. Поддерживается также возможность построения графиков параметров, не имеющих архива значений.
- 'Спектр' — строит частотный спектр из значений параметров подсистемы 'Сбор данных'. Окно данных частотного спектра формируется, исходя из размера виджета по горизонтали в пикселах и доступных данных параметров, наложенных на сетку горизонтального размера. В связи с этим минимальная частота определяется значением атрибута tSize (1/tSize), а максимальная частота выделяемых частот определяется половинной шириной графика в пикселах умноженной на минимальную частоту (width/(2*tSize)). Поддерживается возможность формирования спектра в режиме слежения.

Процесс доступа к архивным данным оптимизирован путём ведения промежуточного буфера для отображения, а также упаковки трафика данных при запросе. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.5.

Таблица 3.8.5. Набор дополнительных свойств/атрибутов в примитиве Diagram

Id	**Имя**	**№**	**Значение**
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.
bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None; Dotted; Dashed; Solid; Double; Groove; Ridge; Inset; Outset).
trcPer	Tracing period (s)	25	Режим и периодичность слежения.
type	Type	26	Тип диаграммы: “Trend”, 'Спектр'.
Атрибуты тренда/графика (Trend)			
tSek	Time:sek	27	Текущее время, секунд.
tUSek	Time:usek	28	Текущее время, микросекунды.

Id	Имя	№	Значение
tSize	Size, sek	29	Размер тренда, секунды.
curSek	Cursor:sek	30	Положение курсора, секунды.
curUSek	Cursor:usek	31	Положение курсора, микросекунды.
curColor	Cursor:color	32	Цвет курсора.
sclColor	Scale:color	33	Цвет шкалы/решетки.
sclHor	Scale:horizontal	34	Режим горизонтальной шкалы/решетки: “No draw”, “Grid;Markers” и “Grid and markers”.
sclVer	Scale:vertical	35	Режим вертикальной шкалы/решетки: “No draw”, “Grid”, “Markers”, “Grid and markers”, «Grid (log)", «Marker (log)", «Grid and markers (log)".
sclMarkColor	Scale:Markers:color	36	Цвет маркеров шкалы/решетки.
sclMarkFont	Scale:Markers:font	37	Шрифт маркеров шкалы/решетки в виде {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
valArch	Value archivator	38	Архиватор архивов параметров.
parNum	Parameters number	39	Количество параметров, отображаемых на одном тренде.
Индивидуальные атрибуты параметров тренда/графика			
prm{X}addr	Parametr {X} :address	50+10*{X}	Полный адрес к параметру {X} или архиву значений.
prm{X}bordL	Parametr {X} :view border:lower	50+10*{X} +1	Нижняя граница значений параметра {X}.
prm{X}bordU	Parametr {X} :view border:upper	50+10*{X} +2	Верхняя граница значений параметра {X}.
prm{X}color	Parametr {X} :color	50+10*{X} +3	Цвет отображения тренда параметра {X}.
prm{X}val	Parametr {X} :value	50+10*{X} +4	Значение параеметра {X} под курсором.

3.8.6. Элемент построения протоколов, на основе архивов сообщений (Protocol)

Данный примитив предназначен для визуализации данных архива сообщений путём формирования протоколов с различными способами визуализации, начиная от статического сканирующего просмотра и заканчивая динамическим отслеживанием протокола сообщения. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.6.

Таблица 3.8.6. Набор дополнительных свойств/атрибутов в примитиве Protocol

Id	Имя	№	Значение
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
font	Font	22	Шрифт текста в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
time	Time, sek	24	Текущее время, секунд.
tSize	Size, sek	25	Размер запроса, секунды.
trcPer	Tracing period (s)	26	Режим и периодичность слежения.
arch	Archival	27	Архиватор архива сообщений.
tmpl	Template	28	Шаблон запроса в архиве.
lev	Level	29	Уровень сообщений.
col	View columns	31	Отображаемые колонки.

3.8.7. Элемент формирования отчётной документации (Document)

Примитив предназначен для формирования отчётной, оперативной и иной документации на основе шаблонов документов.Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.7.

Таблица 3.8.7. Набор дополнительных свойств/атрибутов в примитиве Document

Id	**Имя**	**№**	**Значение**
style	CSS	20	Стиль документа (CSS).
tmpl	Template	21	XHTML исходный шаблон документа.
doc	Document	22	Псевдо-виртуальный атрибут текущего (выбранного) документа.
font	Font	26	Базовый шрифт текста документа в полной записи {<Family> <Size> <Bold> <Italic> <Underline> <Strikeout>}.
bTime	Time:begin	24	Время начала документа, секунд.
time	Time:current	23	Время генерации документа, секунд.
n	Archive size	25	Количество документов или глубина архива.
Атрибуты включеного режима архивирования			
aCur	Cursor:archive	–	Позиция текущего документа в архиве. Запись значения <0 производит архивацию текущего документа.
vCur	Cursor:view	–	Текущий визуализируемый документ архива. Запись значения -1 — выбор следующего документа, -2 — выбор предыдущего документа.
Атрибуты архива			
doc{X}	Document {X}	–	Архивный документ X (0...(n-1))

Возможности примитива «Документ»:

- Гибкое формирования структуры документа на основе языка гипертекстовой разметки. Это предоставит поддержку широких возможностей форматирования документов с последующей реализацией обёртки графического интерфейса формирования документа.
- Формирования документов по команде или по плану в архив с последующим просмотром архива.
- Формирование документа в режиме реального времени полностью динамически и на основе архивов за указанное время.
- Использование атрибутов виджета для передачи значений и адресов на архивы в документ. Позволяет использовать виджет документа как шаблон для формирования отчётов с другими входными данными.

В основе любого документа лежит XHTML-шаблон. XHTML-шаблон это тег 'body', WEB-страницы, содержащий статику документа в стандарте XHTML 1.0 и элементы исполняемых инструкций на одном из языков пользовательского программирования OpenSCADA в виде **<?dp <procedure> ?>**. Результирующий документ формируется путём исполнения процедур и вставки их результата в документ.

Источником значений исполняемых инструкций являются атрибуты виджета этого примитива, а также все механизмы языков пользовательского программирования OpenSCADA. Атрибуты могут добавляться пользователем и линковаться на реальные атрибуты параметров или-же являться автономными, значения которых будут формироваться в скрипте виджета. В случае со слинкованными атрибутами могут извлекаться значения из истории, архива.

На рис. 3.7.7.а изображена структурная схема виджета примитива «Документ». Согласно этой структуре «Документ» содержит: XHTML-шаблон, результирующие документы и скрипт обработки данных. Источником данных для скрипта и результирующих документов являются атрибуты виджета.

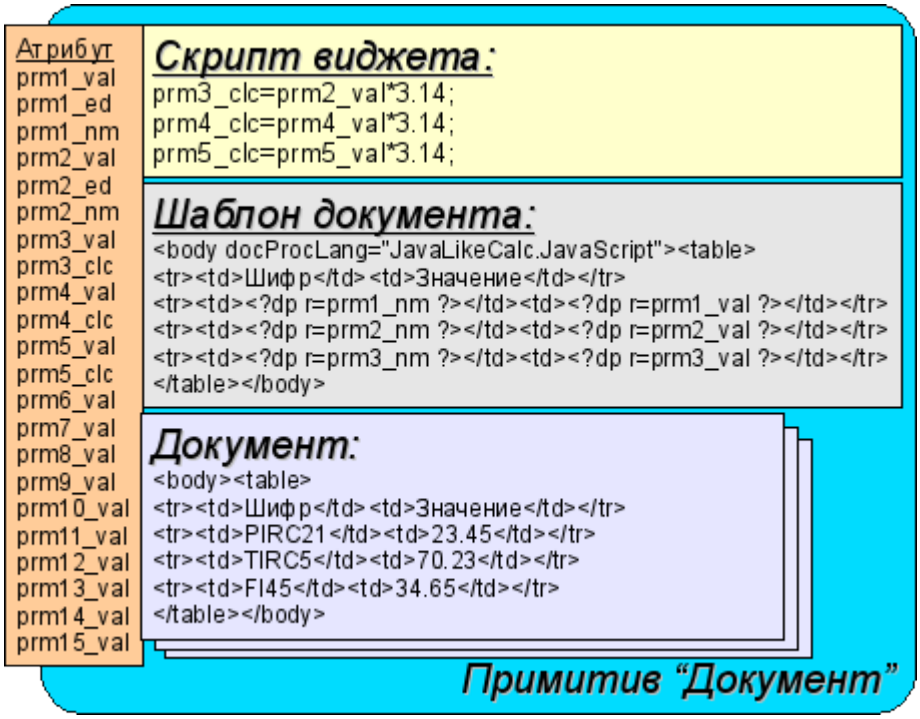


Рис.3.7.7.а Структурная схема примитива «Документ».

Предусматривается работа виджета в двух режимах: Динамический и Архивный. Отличие архивного режима заключается в наличии архива указанной глубины и атрибутов позволяющих управлять процессом архивирования и просмотра указанного документа в архиве.

Генерация документа всегда производится в момент установки атрибута времени <time> относительно установленного начального времени документа в атрибуте <bTime>. При выключенном архиве результирующий документ помещается непосредственно в атрибут <doc>. При включенном архиве результирующий документ помещается в ячейку под курсором, атрибут <aCur>, а так-же в <doc> если значение курсора архива <aCur> и курсора визуализируемого документа <vCur> совпадают. Атрибуты архивных курсоров предусматривают несколько командных значений:

- *aCur<0* — Перемещает курсор архиватора на следующую позицию, тем самым оставляя предыдущий документ в архиве и очищая документ под курсором.
- *vCur==-1* — Выбор следующего документа для отображений. Выбранный документ копируется в атрибут <doc>.
- *vCur==-2* — Выбор предыдущего документа для отображений. Выбранный документ копируется в атрибут <doc>.

Как было указано выше динамика шаблона документа определяется вставками исполняемых инструкций вида **<?dp <procedure> ?>**. В процедурах могут использоваться одноимённые атрибуты виджета и функции пользовательского интерфейса программирования OpenSCADA. Кроме атрибутов виджета зарезервированы специальные атрибутами (табл. 3.7.7.а).

Кроме специальных атрибутов в XHTML шаблоне зарезервированы теги и атрибуты тегов специального назначения (табл. 3.7.7.а).

Таблица 3.7.7.а. Специальные и зарезервированные элементы шаблона.

Имя	**Назначение**
Атрибуты	
rez	Атрибут результата исполнения процедуры содержимое которого помещается дерево документа.
lTime	Время последнего формирования. Если документ формируется впервые то <lTime> равен <bTime>.
rTime	Содержит время для перебираемых значений в секундах. Определяется внутри тегов с атрибутом 'docRept'.

Имя	Назначение
rTimeU	Содержит время для перебираемых значений в микросекундах. Определяется внутри тегов с атрибутом 'docRept'.
rPer	Содержит периодичность перебора значений (атрибут 'docRept').
mtime, mTimeU, mLev, mCat, mVal	Определяются внутри тегов с атрибутом 'docAMess' при разборе сообщений архива сообщений: mTime – время сообщения; mTimeU – время сообщения, микросекунды; mLev – уровень сообщения; mCat – категория сообщения; mVal – значение сообщения.
Специальные теги	
Специальные атрибуты стандартных тегов	
body.docProcLang	Язык исполняемых процедур документа. По умолчанию это JavaLikeCalc.JavaScript.
*.docRept="1s"	Тег с указанным атрибутом при формировании размножается путём смещения времени в атрибуте 'rTime' на значение указанное в данном атрибуте.
.docAMess="1:PLC"	Указывает на необходимость размножения тега с атрибутом сообщениями из архива сообщений за указанный интервал времени и в соответствии с уровнем (1) и шаблоном запроса (PLC*). Для данного тега, в процессе размножения, определяются атрибуты: mTime, mTimeU, mLev, mCat и mVal
*.docRevers="1"	Указывает на инвертирование порядок размножения, последний сверху.
*.docAppend="1"	Признак необходимости добавления результата выполнения процедуры в тег процедуры. Иначе результат исполенеия заменяет содержимое тега.
body.docTime	Время формирования документа. Используется для установки атрибута <lTime> при следующем формировании документа. Не устанавливается пользователем!

3.8.8. Контейнер (Box)

Примитив контейнера используется для формирования составных виджетов и/или страниц пользовательского интерфейса. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 3.8.8.

Таблица 3.8.8. Набор дополнительных свойств/атрибутов в примитиве Box

Id	Имя	№	Значение
pgOpenSrc	Page:open source	3	Полный адрес страницы, которая включена внутрь данного контейнера.
pgGrp	Page:group	4	Группа контейнера страниц.
backColor	Background:color	20	Фоновый цвет.
backImg	Background:image	21	Фоновое изображение.
bordWidth	Border:width	22	Ширина бордюра.
bordColor	Border:color	23	Цвет бордюра.
bordStyle	Border:style	24	Стиль бордюра (None;Dotted;Dashed;Solid;Double;Groove;Ridge;Inset;Outset).

3.9. Использование БД для хранения библиотек виджетов и проектов

Хранение данных виджетов и библиотек виджетов реализовано в БД, доступных системе OpenSCADA. БД организована по принадлежности данных к библиотеке. Т.е. отдельная библиотека хранится в отдельной группе таблиц одной или разных БД. Перечень библиотек виджетов хранится в индексной таблице библиотек с именем «VCALibs» и структурой “Libs”. Экземпляр этой таблицы создаётся в каждой БД, где хранятся данные этого модуля с перечнем библиотек, содержащихся в конкретно взятой БД. В состав таблиц, принадлежащих библиотеке виджетов, входят следующие:

- {DB_TBL} — Таблица с виджетами, принадлежащими библиотеке (структура «LibWigets").
- {DB_TBL}_io — Таблица с рабочими свойствами виджетов этой библиотеки и вложенными виджетами контейнерных виджетов (структура «LibWidgetIO").
- {DB_TBL}_uio — Таблица с пользовательскими свойствами виджетов этой библиотеки и вложенными виджетами контейнерных виджетов (структура «LibWidgetUserIO", раздела БД).
- {DB_TBL}_incl — Таблица с перечнем вложенных виджетов в виджеты-контейнеры данной библиотеки (структура «LibWidgetIncl").
- {DB_TBL}_mime — Таблица с ресурсами библиотеки и её виджетов (структура «LibWidgetMime").

Проекции (структуры) основных таблиц таковы :

- Libs(ID, NAME, DSCR, DB_TBL, ICO) — Библиотеки виджетов <*ID*>.
  ID — идентификатор;
  NAME — имя;
  DSCR — описание;
  DB_TBL — БД с виджетами;
  ICO — закодированное (Base64) изображение иконки библиотеки.
- LibWigets(ID, ICO, PARENT, PROC, PROC_PER, USER, GRP, PERMIT, ATTRS) — Виджеты <*ID*> библиотеки.
  ID — идентификатор;
  ICO — закодированное (Base64) изображение иконки виджета;
  PARENT — адрес виджета основы в виде */wlb_originals/wdg_Box* ;
  PROC — внутренний сценарий и язык сценария виджета;
  PROC_PER — периодичность вычисления сценария виджета;
  USER — владелец виджета;
  GRP — группа пользователей виджета;
  PERMIT — права доступа к виджету;
  ATTRS — перечень атрибутов виджета, модифицированных пользователем.
- LibWidgetIO(IDW, ID, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) — Рабочие атрибуты <*ID*> виджета <*IDW*>.
  IDW — идентификатор виджета;
  ID — идентификатор IO;
  IO_VAL — значение атрибута;
  SELF_FLG — внутренние флаги IO;
  CFG_TMPL — шаблон элемента конфигурации, основанного на данном атрибуте;
  CFG_VAL — значение элемента конфигурации (ссылка, константа ...).
- LibWidgetUserIO(IDW, ID, NAME, IO_TP, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) — Пользовательские атрибуты <*ID*> виджета <*IDW*>.
  IDW — идентификатор виджета;
  ID — идентификатор IO;
  NAME — имя IO;
  IO_TP — тип и главные флаги IO;
  IO_VAL — значение IO;
  SELF_FLG — внутренние флаги IO;
  CFG_TMPL — шаблон элемента конфигурации, основанного на данном атрибуте;

CFG_VAL — значение элемента конфигурации (ссылка, константа ...).

- LibWidgetIncl(IDW, ID, PARENT, ATTRS, USER, GRP, PERMIT) — Включенные в контейнер <*IDW*> виджеты <*ID*>.

IDW — идентификатор виджета;
ID — идентификатор экземпляра вложенного виджета;
PARENT — адрес виджета основы в виде */wlb_originals/wdg_Box* ;
ATTRS — перечень атрибутов виджета, модифицированных пользователем;
USER — владелец виджета;
GRP — группа пользователей виджета;
PERMIT — права доступа к виджету.

- LibWidgetMime(ID, MIME, DATA) — Audio, video, media и другие ресурсы виджетов библотеки.

ID — Идентификатор ресурса.
MIME — Mime тип данных ресурса (в формате – <mimeType;Size>).
DATA — Данные ресурса кодированные Base64.

- Project(ID, NAME, DSCR, DB_TBL, ICO, USER, GRP, PERMIT, PER, FLGS) — Проекты интерфейсов визуализации <*ID*>.

ID — идентификатор проекта;
NAME — имя проекта;
DSCR — описание проекта;
DB_TBL — БД со страницами проекта.
ICO — закодированное (Base64) изображение иконки проекта;
USER — владелец проекта;
GRP — группа пользователей проекта;
PERMIT — права доступа к проекту;
PER — периодичность вычисления проекта;
FLGS — флаги проекта.

- ProjPage(OWNER, ID, ICO, PARENT, PROC, PROC_PER, USER, GRP, PERMIT, FLGS, ATTRS) — Страницы <*ID*> содержащиеся в проекте/странице <*OWNER*>.

OWNER — проект/страница – владелец данной страницы (в виде – «/AGLKS/so/1/gcadr»)
ID — идентификатор страницы;
ICO — закодированное (Base64) изображение иконки страницы;
PARENT — адрес виджета основы страницы в виде */wlb_originals/wdg_Box* ;
PROC — внутренний сценарий и язык сценария страницы;
PROC_PER — периодичность вычисления сценария виджета;
USER — владелец страницы;
GRP — группа пользователей страницы;
PERMIT — права доступа к странице;
FLGS — флаги страницы;
ATTRS — перечень атрибутов виджета, модифицированных пользователем.

- ProjSess(IDW, ID, IO_VAL) — Таблица проекта <*IDW*> для хранения данных сеансов, исполняющих проект.

IDW — полный путь элемента проекта;
ID — атрибут элемента;
IO_VAL — значение атрибута.

- ProjPageIO(IDW, ID, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) — Рабочие атрибуты страниц. Структура фактически совпадает с таблицей LibWidgetIO.
- ProjPageUserIO(IDW, ID, NAME, IO_TP, IO_VAL, SELF_FLG, CFG_TMPL, CFG_VAL) — Пользовательские атрибуты страниц. Структура фактически совпадает с таблицей LibWidgetUserIO.
- ProjPageWIncl(IDW, ID, PARENT, ATTRS, USER, GRP, PERMIT) — Включенные на страницы виджеты. Структура фактически совпадает с таблицей LibWidgetIncl.

3.10 API пользовательского программирования и сервисные интерфейсы OpenSCADA

3.10.1. API пользовательского программирования

API пользовательского программирования движка визуализации представлено группой функций непосредственно в модуле движка СВУ. Вызов этих функций из скриптов виджетов может осуществляться прямо по идентификатору функции, поскольку их область имён указывается для контекста скриптов виджетов.

Список виджетов (WdgList)

Описание: Возвращает список виджетов в контейнере виджетов или список дочерних виджетов. Если установлено <pg>, то возвращается список страниц для проектов и сеансов.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
list	Список	Строка	Возврат	
addr	Адрес	Строка	Вход	
pg	Страницы	Bool	Вход	0

Присутствие узла (NodePresent)

Описание: Проверка на присутствие узла, включая виджеты, атрибуты и другие.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
rez	Результат	Bool	Возврат	
addr	Адрес	Строка	Вход	

Список атрибутов (AttrList)

Описание: Возвращает список атрибутов виджета. Если установлен <noUser> тогда возвращаются только не пользовательские атрибуты.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
list	Список	Строка	Возврат	
addr	Адрес	Строка	Вход	
noUser	Без пользовательских	Bool	Вход	1

Запрос атрибута (AttrGet)

Описание: Запрос значения атрибута виджета. Запрос может осуществляться как указанием полного адреса атрибута в <addr>, так и отдельно адреса виджета в <addr>, а идентификатора атрибута в <attr>.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
val	Значение	Строка	Возврат	
addr	Адрес	Строка	Вход	
attr	Атрибут	Bool	Вход	

Установка атрибута (AttrSet)

Описание: Установка значения атрибута виджета. Установка может осуществляться как указанием полного адреса атрибута в <addr>, так и отдельно адреса виджета в <addr>, а идентификатора атрибута в <attr>.

Параметры:

ID	Имя	Тип	Режим	По умолчанию
addr	Адрес	Строка	Вход	
val	Значение	Строка	Вход	
attr	Атрибут	Bool	Вход	

3.10.2. Сервисные интерфейсы OpenSCADA

Сервисные интерфейсы это интерфейсы доступа к системе OpenSCADA посредством интерфейса управления OpenSCADA из внешних систем. Данный механизм положен в основу всех механизмов обмена внутри OpenSCADA, реализованных посредством слабых связей и стандартного протокола обмена OpenSCADA.

Доступ к значениям атрибутов элементов визуализации (виджетам)

С целью предоставления унифицированного, группового и сравнительно быстрого доступа к значениям атрибутов визуальных элементов предусмотрена сервисная функция визуального элемента «/serv/attr» и команды получения/установки значений атрибутов: <get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/> и <set path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/>. Атрибуты данных команд, предусматривающие различные механизмы запроса, представим в таблице 4.13.1.

Таблица 3.10.2.а. Атрибуты команд получения/установки атрибутов визуальных элементов

Id	Имя	Значение
Команда запроса визуальных атрибутов виджета: <get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/>		
tm	Время/счётчик изменений	Установка времени/счётчика изменений для запроса только изменившихся атрибутов.
<el id="{attr}" p="{a_id}">{val}</el>	Формирование дочерних элементов с результатами атрибутов	В дочернем элементе указываются: строковых идентификатор {attr} атрибута, индекс {a_id} атрибута и его значение {val}.
Команда установки визуальных атрибутов виджета: <set path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattr"/>		
<el id="{attr}">{val}</el>	Установка атрибутов	В дочерних элементах указывается идентификатор атрибута {attr} и его значение {val}.

Групповой доступ к значениям атрибутов элементов визуализации (виджеты)

С целью оптимизации трафика сетевого взаимодействия путём исключения мелких запросов, а использования одного, но большого запроса создан групповой запрос значений атрибутов визуальных элементов. Группировка данного запроса подразумевает запрос атрибутов всей ветви виджета, включая и вложенные элементы. Для данного запроса предусмотрена сервисная команда «/serv/attrBr». Запрос данной сервисной команды эквивалентен сервисной команде «/serv/attr» и выглядит следующим образом: *<get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2fattrBr"/>*

tm — Время/счётчик изменений. Установка времени/счётчика изменений для запроса только изменившихся атрибутов.

Результат:

<el id="{attr}" p="{a_id}">{val}</el> — Элементы с результатами атрибутов. В элементе указываются: строковый идентификатор {attr} атрибута, индекс {a_id} атрибута и его

значение {val}.

<w id="{wid}" lnkPath="{lnk_path}">{childs+attrs}</w> — Элементы с дочерними виджетами и их атрибутами. В элементе указываются идентификатор дочернего виджета {wid} и путь виджета на который ссылается данный виджет если он является ссылкой {lnk_path}.

Доступ к страницам сеанса

С целью унификации и оптимизации доступа к страницам предусмотрена сервисная функция сеанса «/serv/pg» и команды запроса перечня открытых страниц (<openlist path="/UI/VCAEngine/ses_{Session}/%2fserv%2fpg"/>); открытия страницы (<open path="/UI/VCAEngine/ses_{Session}/%2fserv%2fpg"/>); и закрытия страницы <close path="/UI/VCAEngine/ses_{Session}/%2fserv%2fpg"/>).

Результатом запроса перечня открытых страниц являются дочерние элементы <el>{OpPage}</el> содержащие полный путь открытой страницы. Кроме перечня открытых страниц запрос возвращает значение текущего счётчика вычисления сеанса в атрибуте <tm>. Если данный атрибут устанавливается при запросе, то для каждой открытой страницы возвращается список изменённых с момента указанного значения счётчика виджетов открытой страницы.

Управление сигнализацией

Для предоставления механизма глобального контроля за сигнализацией сеанса предусмотрена сервисная функция сеанса «/serv/alarm» и команды запроса статуса сигналов (<get path="/UI/VCAEngine/ses_{Session}/%2fserv%2falarm"/>); и квитации сигналов (<quittance path="/UI/VCAEngine/ses_{Session}/%2fserv%2falarm"/>).

Запрос статуса сигналов возвращает обобщённое состояние сигналов, а так-же дополнительную информацию для звуковой сигнализации. Дополнительная информация звуковой сигнализации предоставляет текущий ресурс, звуковой файл, для воспроизведения и обеспечивает отслеживание последовательности сигнализации и квитации отдельных файлов звуковых сообщений.

Запрос на квитацию выполняет квитацию указанного виджета, атрибут *<wdg>*, в соответствии с шаблоном, *атрибут <tmpl>*.

Манипуляция сеансами проектов

Для предоставления унифицированного механизма манипуляции сеансами визуализаторам СВУ, в модуле движка СВУ (VCAEngin) предусмотрена сервисная функция «/serv/sess» и команды запроса перечня открытых сеансов, подключения/создания нового сеанса и отключения/удаления сеанса: <list path="/UI/VCAEngine/%2fserv%2fsess"/>, <connect path="/UI/VCAEngine/%2fserv %2fsess"/> и <disconnect path="/UI/VCAEngine/%2fserv%2fsess"/> соответственно. Атрибуты данных команд, предусматривающие различные механизмы запроса, представим в таблице 4.13.5.

Таблица 3.10.2.b. Атрибуты команд механизма манипуляции сеансами

Id	**Имя**	**Значение**
Команда запроса перечня открытых сеансов для проекта: <list path="/UI/VCAEngine/%2fserv %2fsess"/>		
prj	Указание проекта	Указывает проект, для которого возвращать перечень открытых сеансов.
<el>{Session}</el>	Контроль перечня сеансов	В дочерних элементах указываются сеансы, открытые для запрошенного проекта.
Команда подключения/открытия сеанса: <connect path="/UI/VCAEngine/%2fserv%2fsess"/>		
sess	Установка и контроль имени сеанса	Если атрибут определён, то производится подключение к существующему сеансу, иначе создание нового сеанса. В случае открытия нового сеанса в данный атрибут помещается его имя.

Id	Имя	Значение
prj	Установка имени проекта	Используется для открытия нового сеанса для указанного проекта и если атрибут {sess} не указан.
Команда отключения/закрытия сеанса: <disconnect path="/UI/VCAEngine/%2fserv%2fsess"/>		
sess	Установка имени сеанса	Указывает имя сеанса от которого выполняется отключение или закрытие. Сеансы, не являющиеся фоновыми и к которым ни один из визуализаторов не подключен автоматически закрываются.

Групповой запрос дерева библиотек виджетов

С целью оптимизации производительности локального и особенно сетевого взаимодействия предусмотрена сервисная функция «/serv/wlbBr» и команда запроса дерева библиотек виджетов: <get path="/UI/VCAEngine/%2fserv%2fwlbBr"/>. Результатом запроса является дерево с элементами библиотек виджетов, теги <*wlb*>. Внутри тегов библиотек виджетов содержаться тег иконки <*ico*> и теги виджетов библиотеки <*w*>. Теги виджетов, в свою очередь, содержат тег иконки и теги дочерних виджетов <*cw*>.

4. Конфигурация модуля посредством интерфейса управления OpenSCADA

Посредством интерфейса управления OpenSCADA компоненты, которые его используют, можно конфигурировать из любого конфигуратора системы OpenSCADA. Данным модулем предоставляется интерфейс для доступа ко всем объектам данных СВУ. Главная конфигурационная страница модуля предоставляет доступ к объектам верхнего уровня, а именно к библиотекам виджетов, проектам и открытым сеансам проектов (рис. 4.1). Для настройки движка синтеза речи предусмотрена соответствующая страница (рис. 4.2).

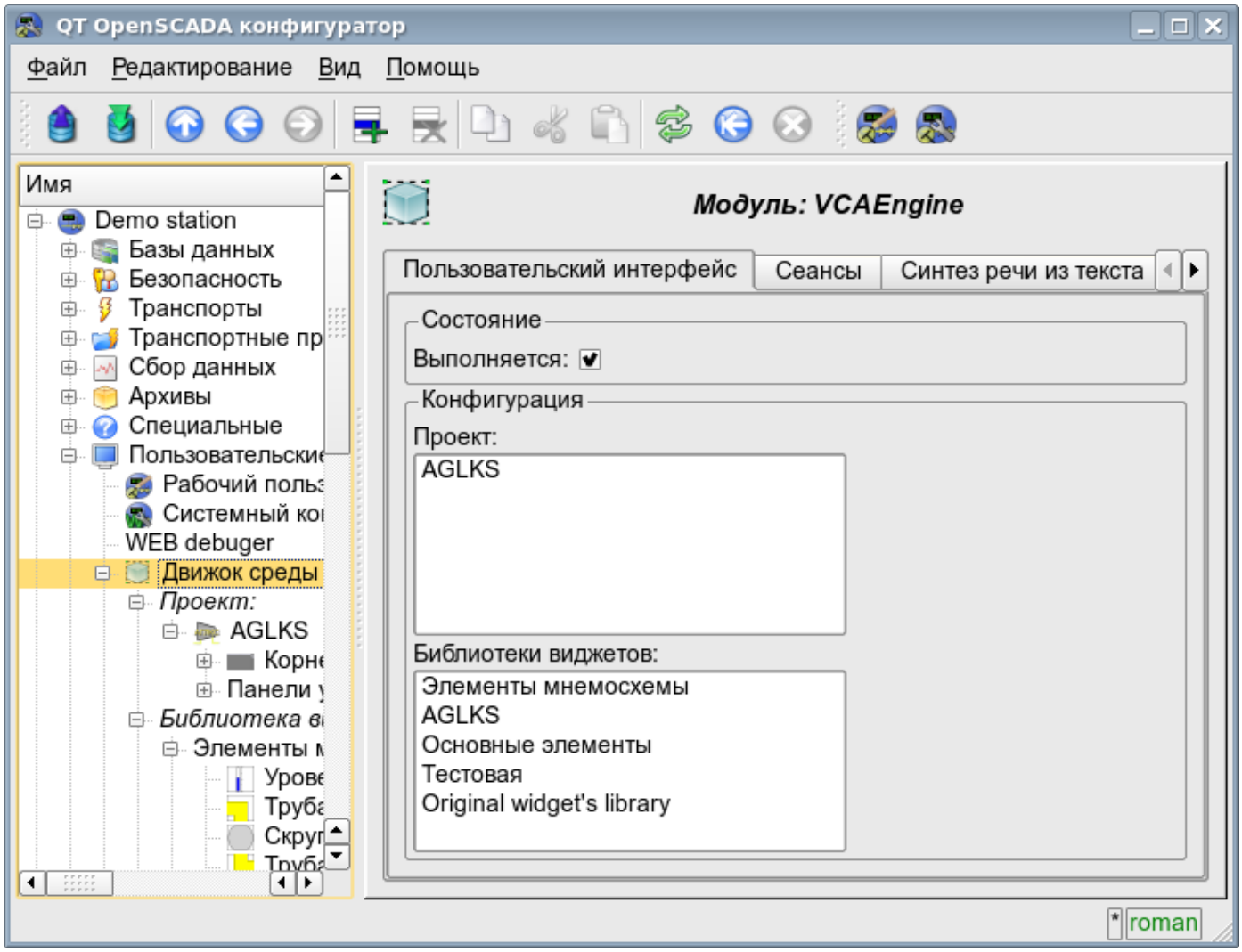


Рис.4.1 Главная конфигурационная страница модуля.

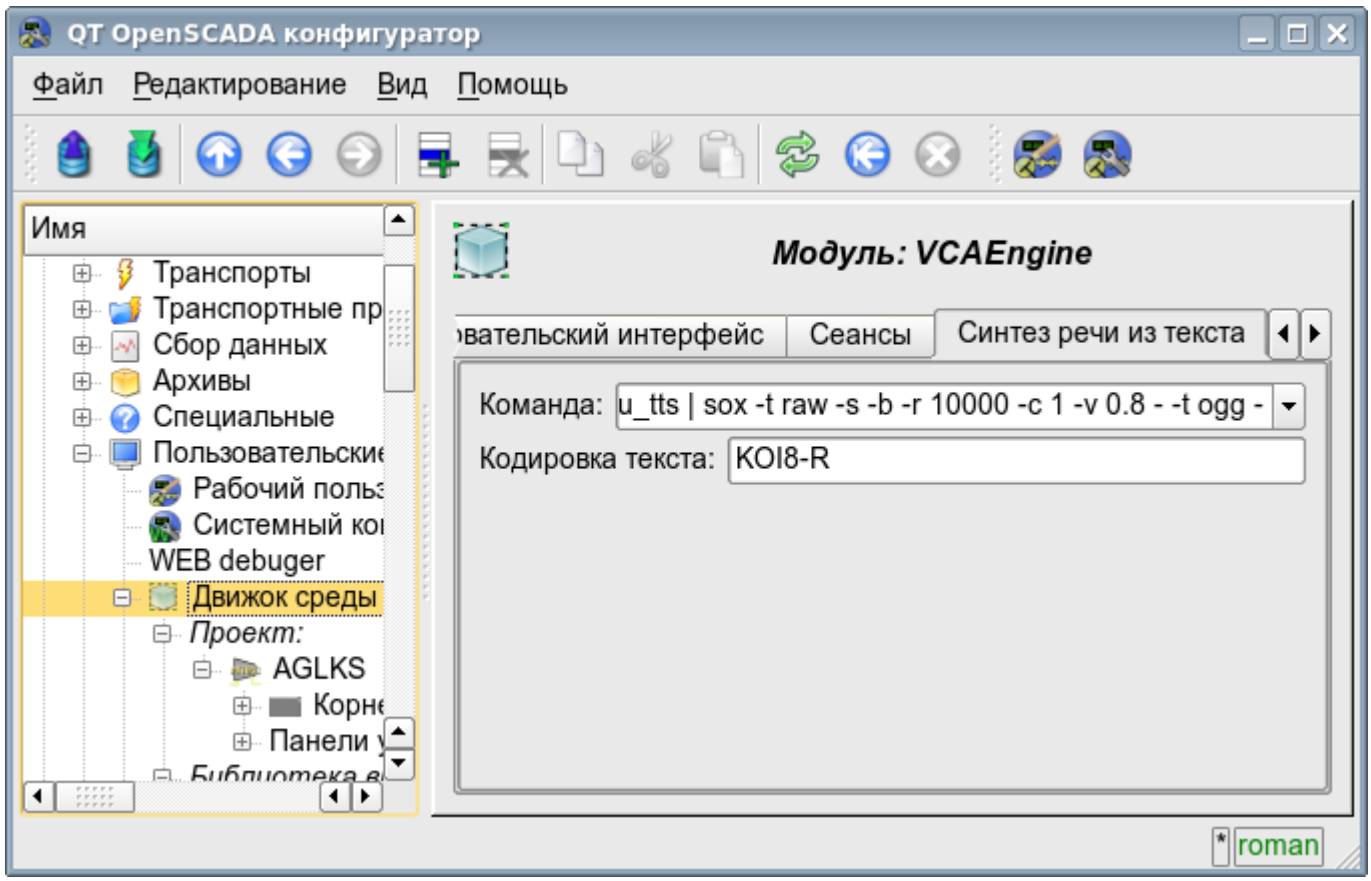


Рис.4.2 Вкладка конфигурации движка синтеза речи.

Конфигурация контейнеров виджетов, в лице библиотек виджетов и проектов выполняется посредством страниц на рис. 4.3 (для проекта) и рис.4.4 (для библиотеки виджетов). Библиотека виджетов содержит виджеты, а проект – страницы. Оба типа контейнера содержат вкладку конфигурации Mime-данных, используемых виджетами (рис.4.5).

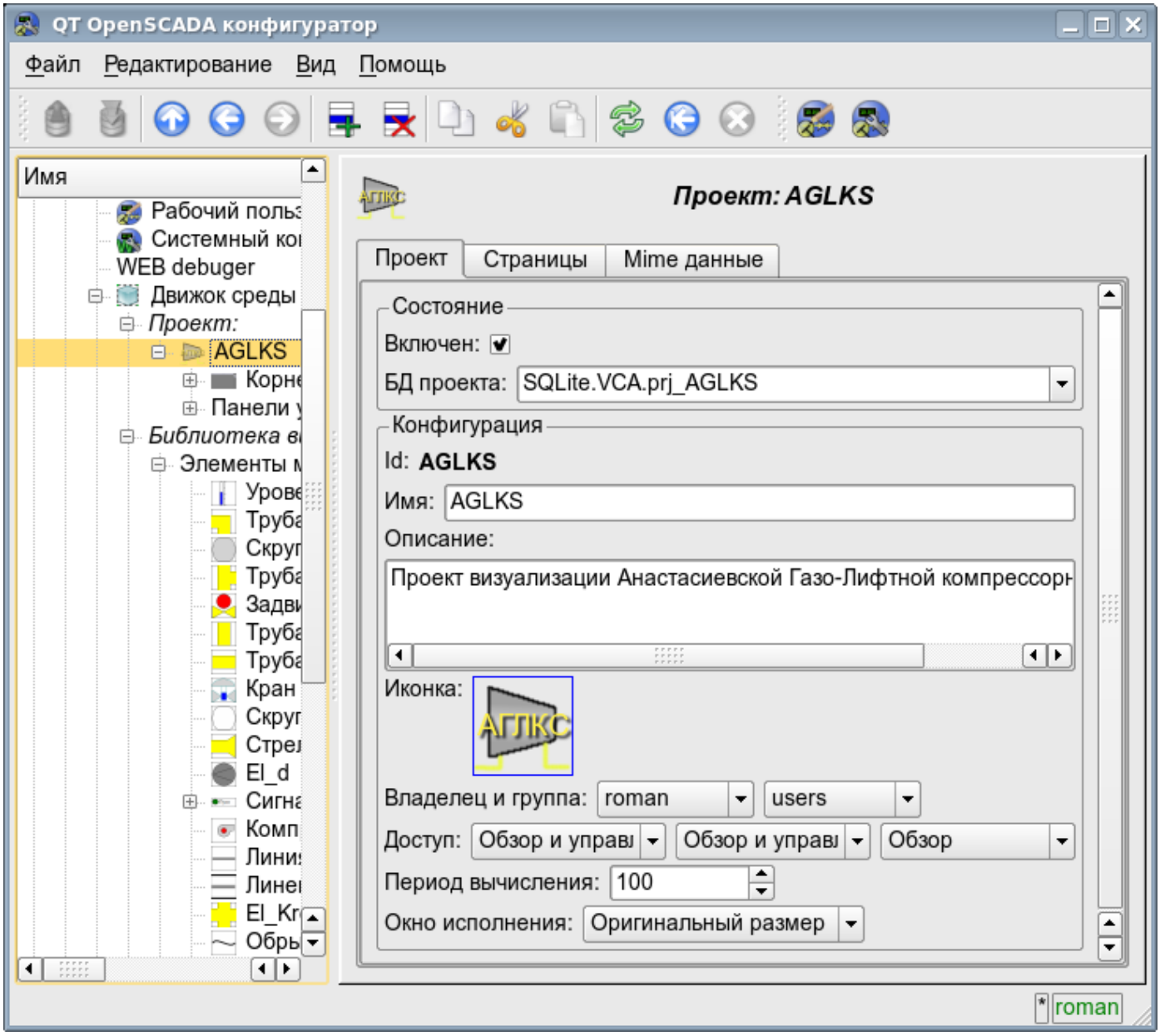


Рис.4.3 Страница конфигурации проектов.

С помощью этой страницы можно установить:

- Состояние контейнера, а именно: «Включен», имя БД, содержащей конфигурацию, владельца и группу контейнера.
- Идентификатор, имя, описание и иконку контейнера.
- Права доступа к контейнеру.
- Период вычисления сеансов основанных на данном проекте.
- Способ открытия главного окна исполнения (Оригинальный размер, максимизация и на весь экран).

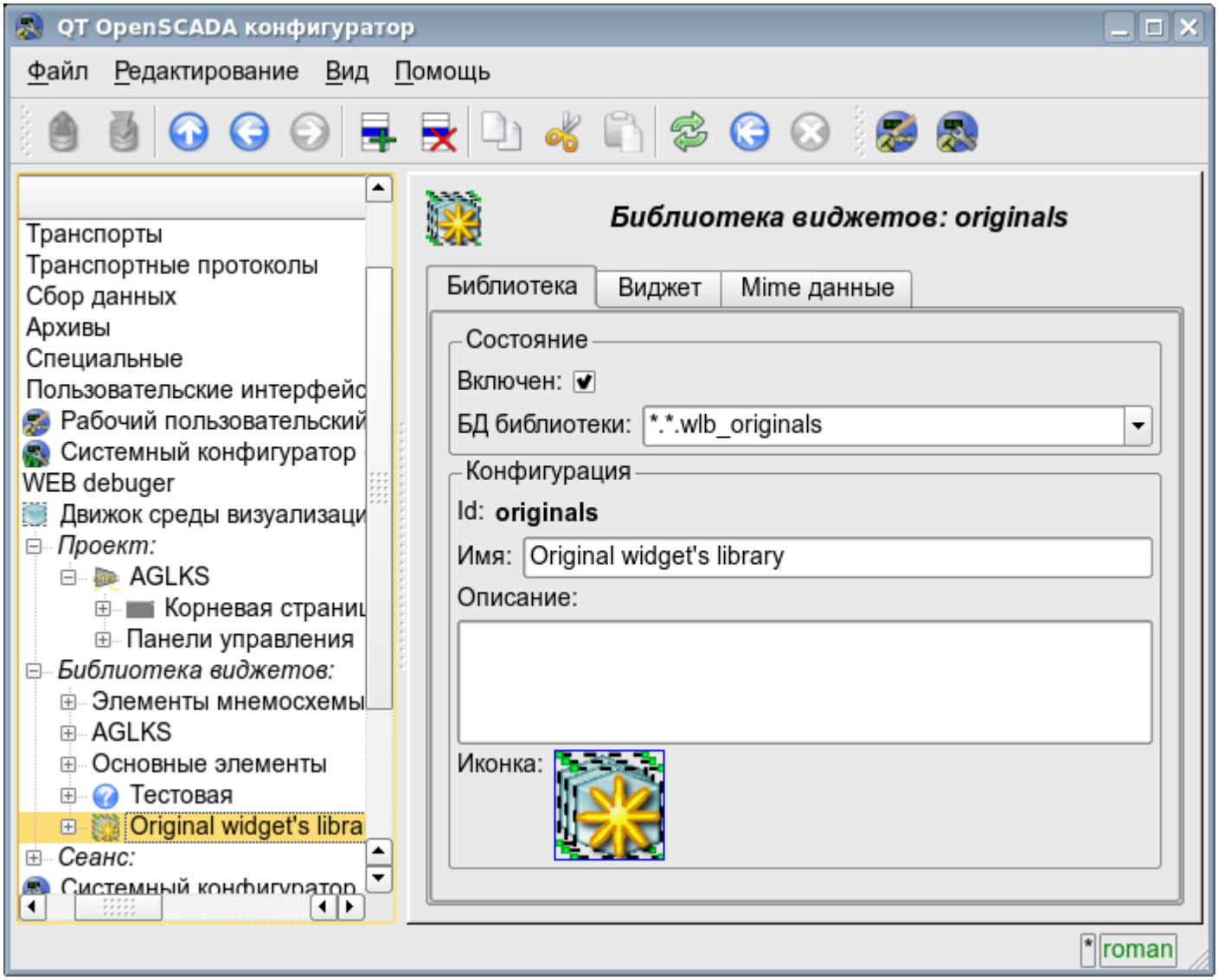


Рис.4.4 Страница конфигурации библиотек виджетов.

С помощью этой страницы можно установить:

- Состояние контейнера, а именно: «Включен», имя БД, содержащей конфигурацию.
- Идентификатор, имя, описание и иконку контейнера.

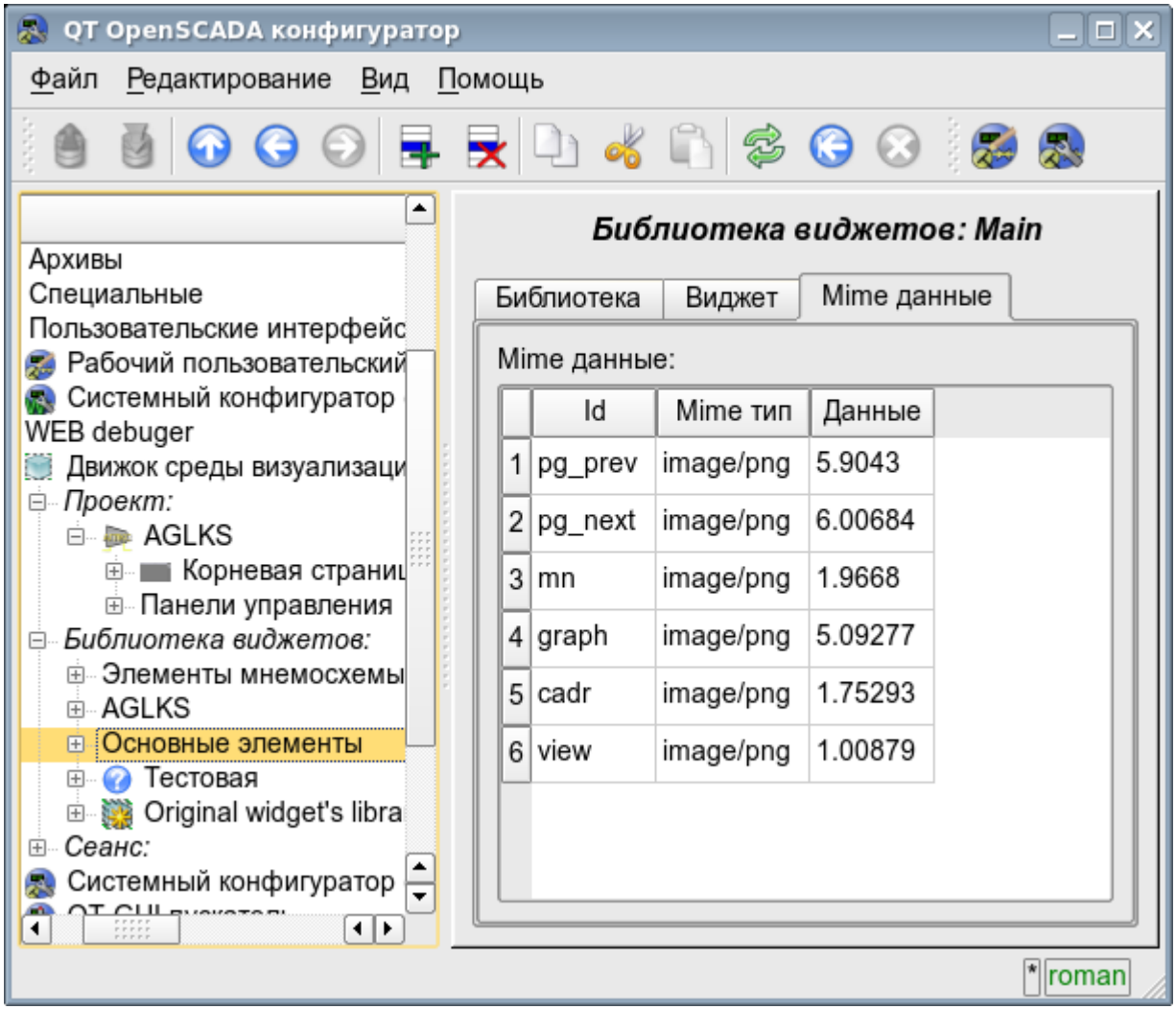


Рис.4.5 Вкладка конфигурации mime-данных контейнера.

Конфигурация сеанса проекта значительно отличается от конфигурации проекта (рис. 4.6), однако также содержит страницы проекта.

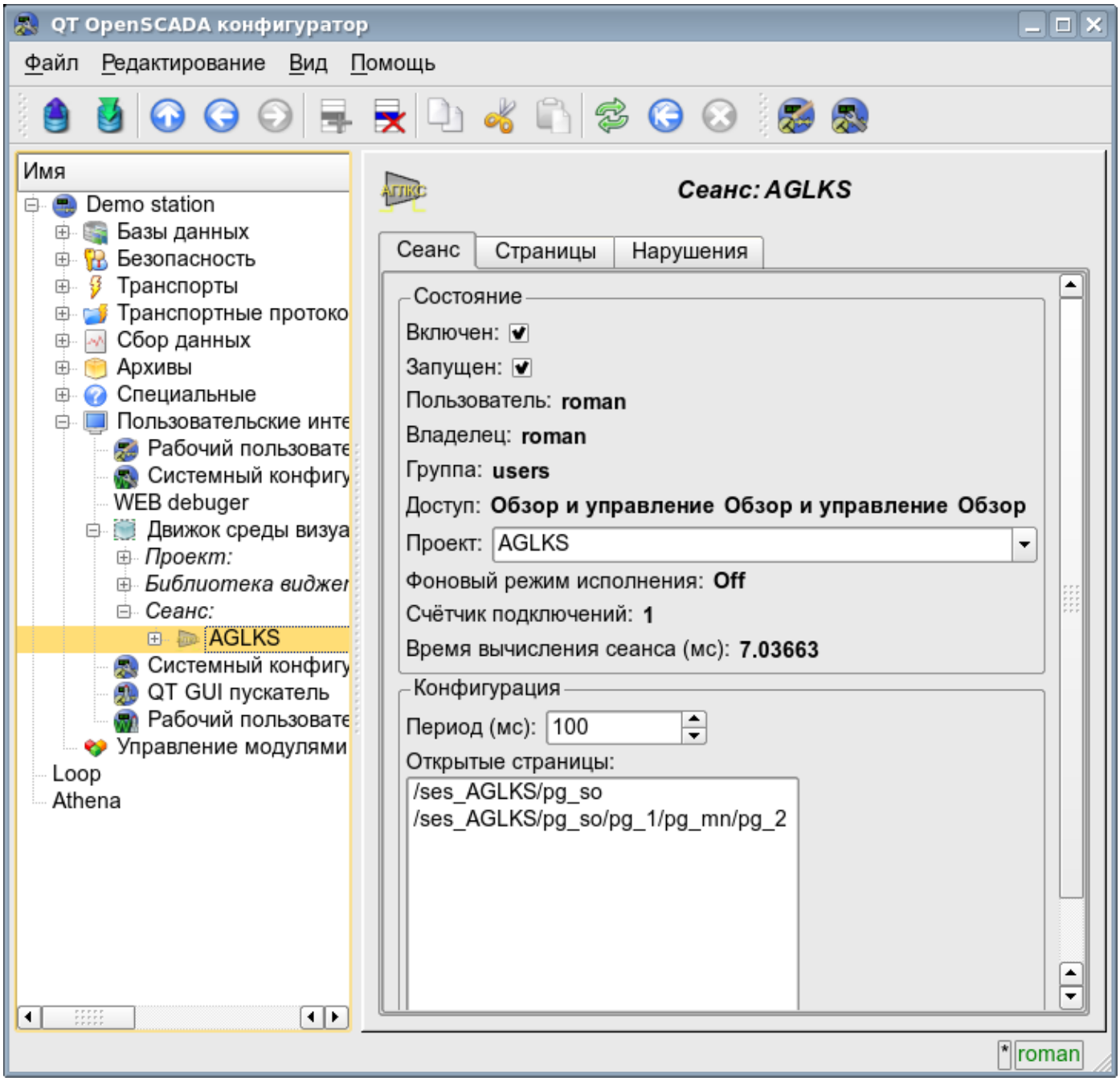


Рис.4.6 Страница конфигурации сеансов проектов.

С помощью этой страницы можно установить:

- Состояние сеанса, а именно: «Включен», «Запущен», пользователь, владелец, группа пользователей, доступ, исходный проект, режим исполнения в фоне, счётчик клиентских подключений и время исполения сеанса.
- Период обсчёта сеанса.
- Перечень открытых страниц.

Страницы конфигурации визуальных элементов, расположенных в разных контейнерах, могут сильно отличаться однако, это отличие заключается в наличии или отсутствии отдельных вкладок. Главная вкладка визуальных элементов фактически везде одинакова, отличаясь на одно конфигурационное поле (рис. 4.7). У страниц присутствуют вкладки дочерних страниц и вложенных виджетов. У контейнерных виджетов содержится вкладка вложенных виджетов. Все визуальные элементы содержат вкладку атрибутов (рис. 4.8), кроме логических контейнеров проектов. Элементы, на уровне которых можно формировать пользовательскую процедуру и определять связи, содержат вкладки «Обработка» (рис. 4.9) и «Связи» (рис.4.10).

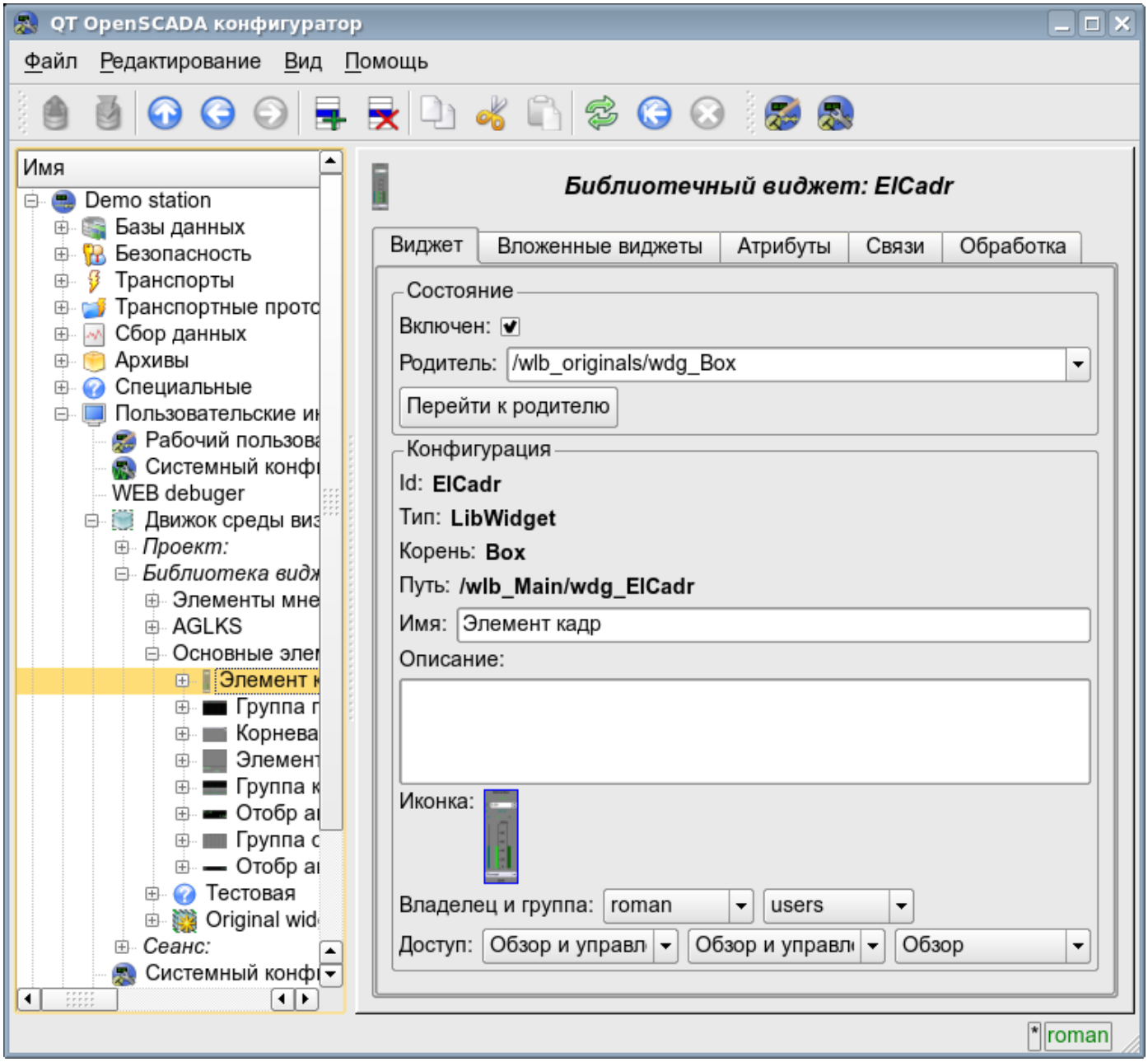


Рис.4.7 Главная вкладка конфигурации визуальных элементов.

С помощью этой страницы можно установить:

- Состояние элемента, а именно: «Включен», родительский элемент и переход к нему. Для страницы в состоянии указывается тип страницы.
- Идентификатор, тип, корень, путь, имя, описание и иконку элемента.
- Владелец, группа пользователей и права доступа к элементу.

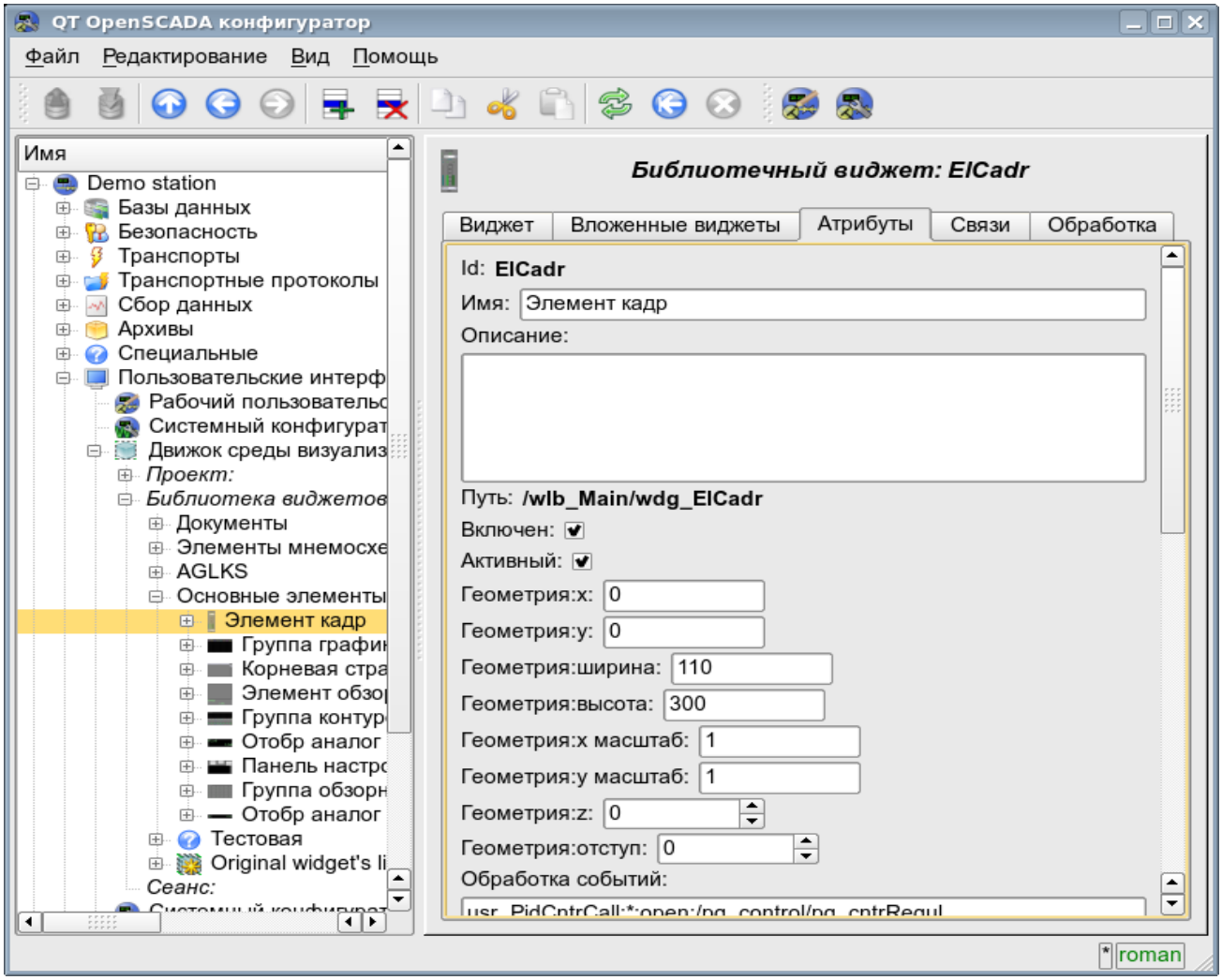


Рис.4.8 Вкладка атрибутов визуальных элементов.

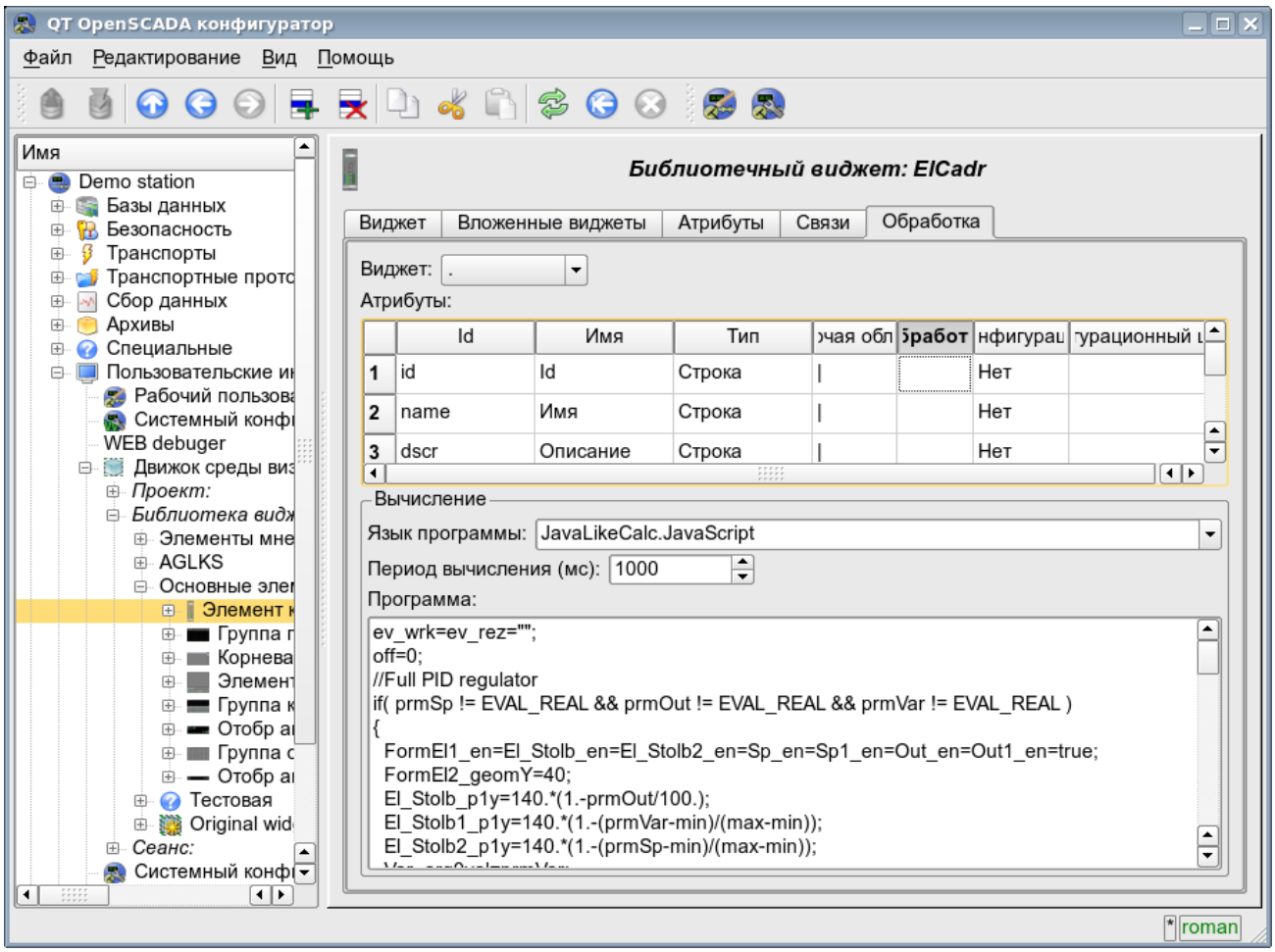


Рис.4.9 Вкладка обработки визуальных элементов.

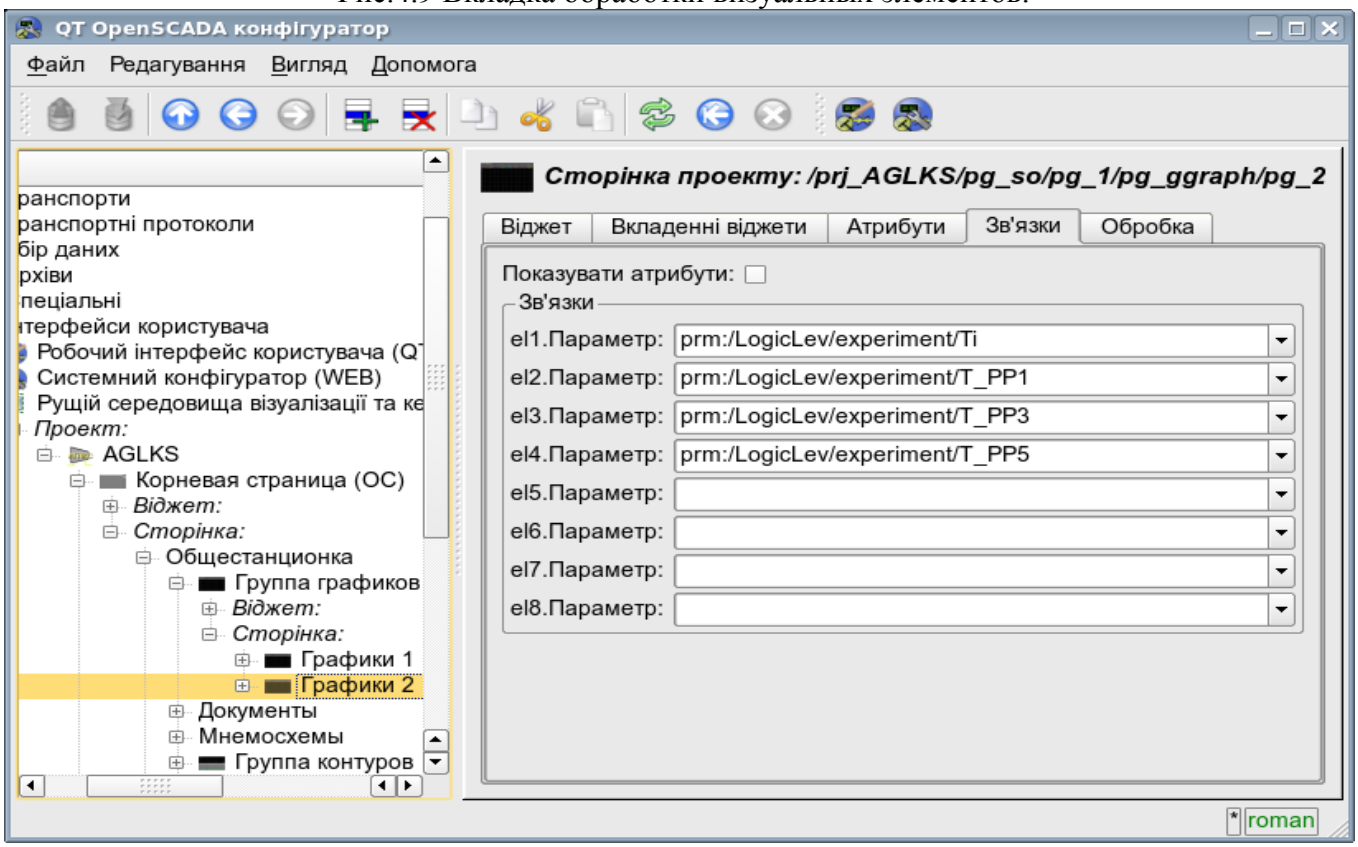


Рис.4.10 Вкладка связей визуальных элементов.