

Модуль подсистемы “Пользовательские интерфейсы” <VCAEngine>

Модуль:	VCAEngine
Имя:	Движок среды визуализации и управления
Тип:	Пользовательские интерфейсы
Источник:	ui_VCAEngine.so
Версия:	0.5.0
Автор:	Роман Савоченко
Описание:	Основной движок среды визуализации и управления.
Лицензия:	GPL

Оглавление

Модуль подсистемы “Пользовательские интерфейсы” <VCAEngine>	1
Введение	2
1 Назначение	3
2 Конфигурация и формирование интерфейсов СВУ	5
3 Архитектура	6
3.1 Кадры и элементы отображения(виджеты)	8
3.2 Проект	11
3.3 Темы отображения	14
3.4 События, их обработка и карты событий	15
3.5 Управление правами	16
3.6 Прimitives виджетов	16
3.7 Использование БД для хранения библиотек виджетов и проектов	24
3.8 Сервисные интерфейсы для доступа к элементам СВУ	26
4 Конфигурация модуля посредством интерфейса управления OpenSCADA	30
Заключение	38

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 ситуациях, 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. С другой стороны, 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) СБУ предназначен для формирования логической структуры СБУ и исполнения сеансов отдельных экземпляров проектов СБУ. Также, модуль предоставляет все необходимые данные конечным визуализаторам СБУ как посредством локальных механизмов взаимодействия OpenSCADA, так и посредством интерфейса управления OpenSCADA для удалённого доступа.

В финальной версии этого модуля СБУ 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 практически любой 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 включение 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 подключение 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 архитектура 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 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 повторность 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. 1.

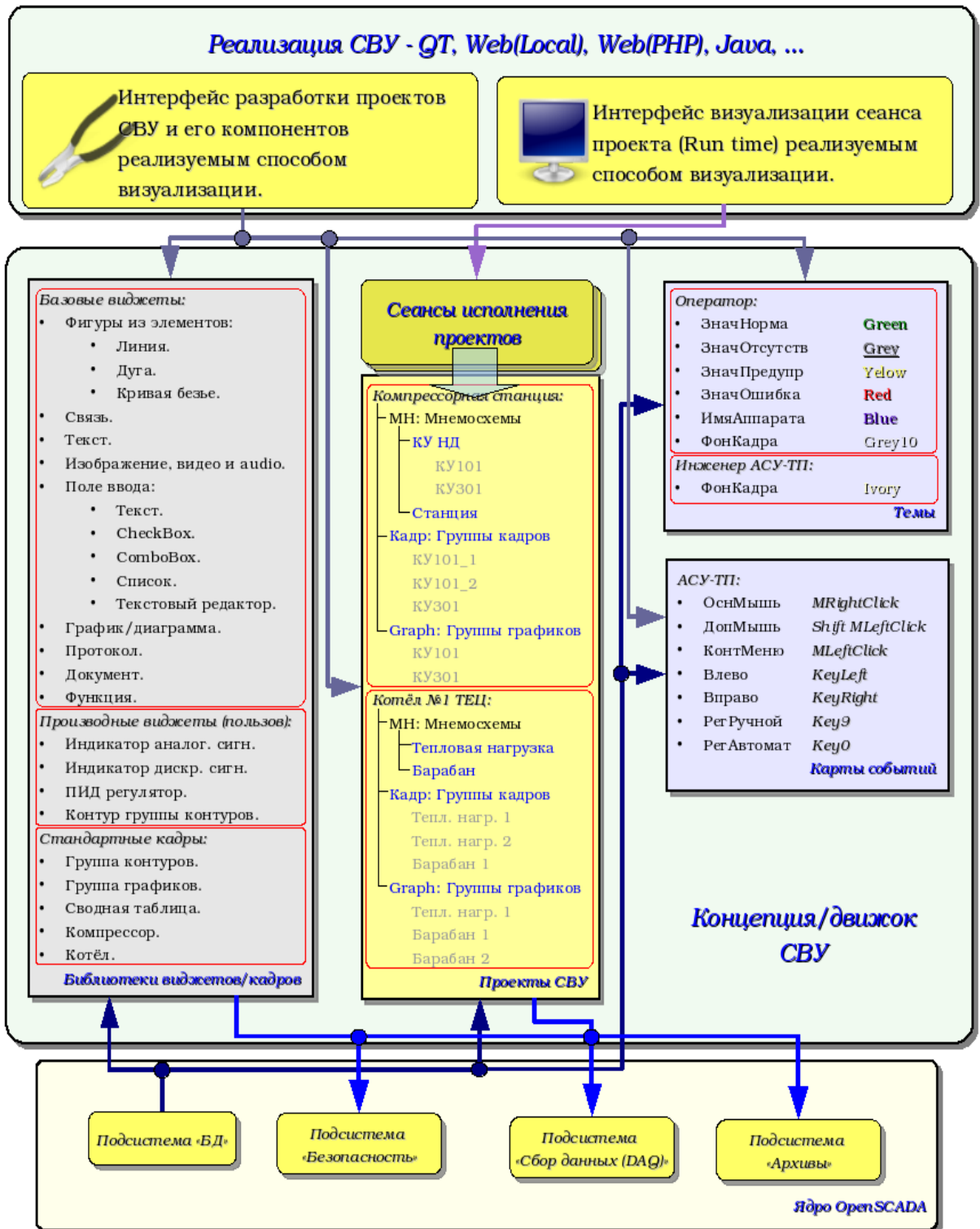


Рис.1 Обобщённая структура СВУ.

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 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(слотах) 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.2).

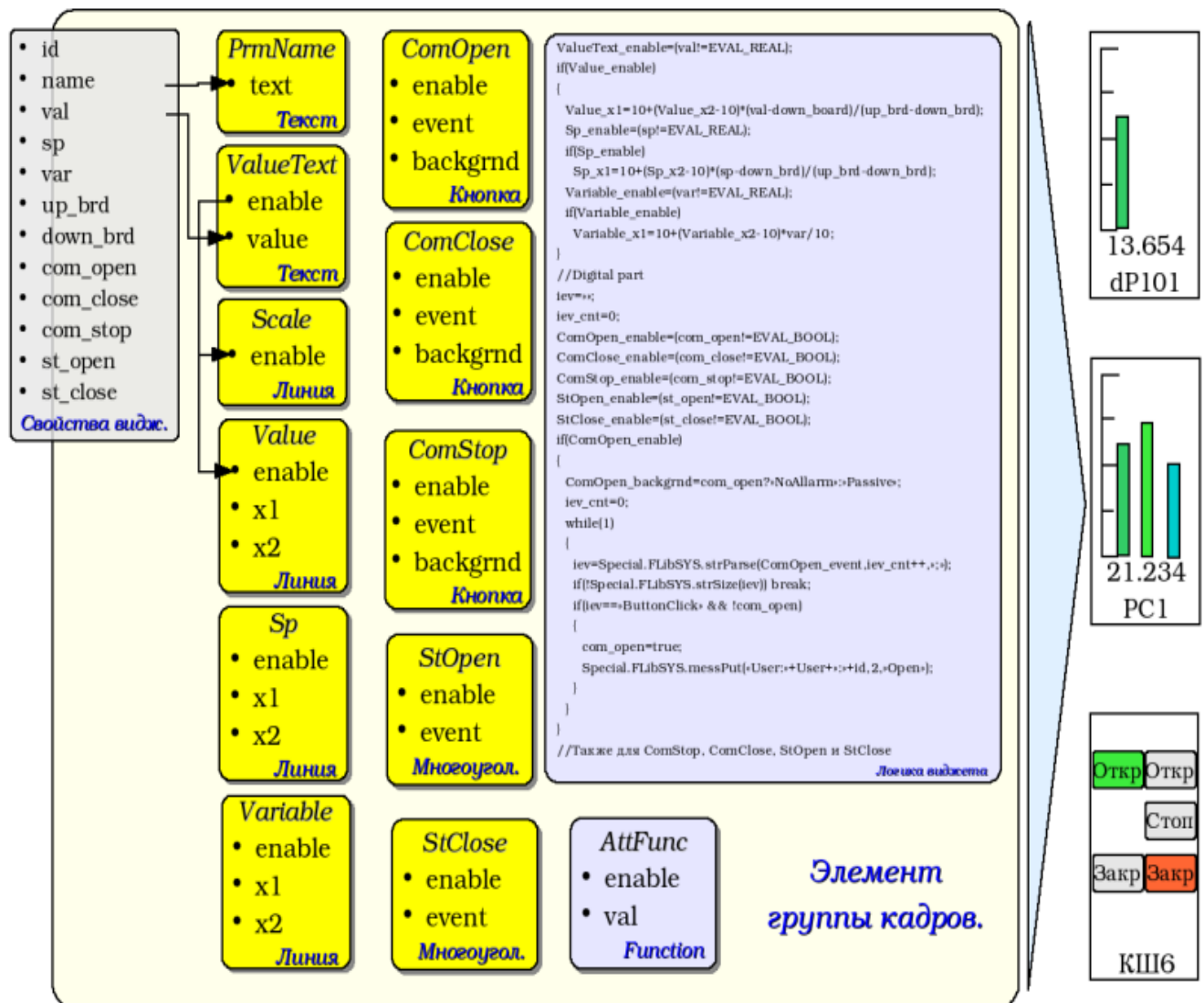


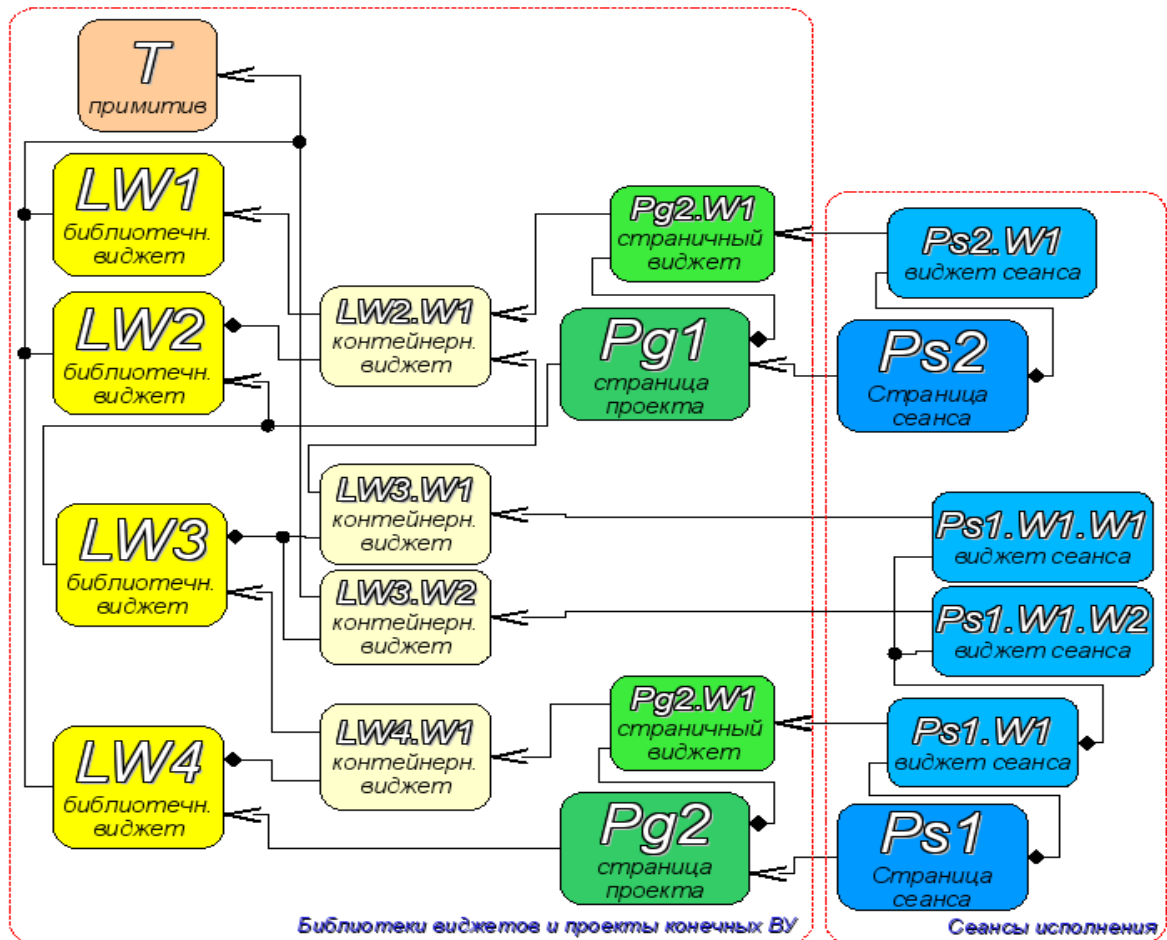
Рис.2 Пример структуры производного виджета.

Виджет является единственным 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. Свойства производных виджетов формируются отдельно и 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 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 для конечного пользователя.

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

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

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

Рис.3 Карта «использующего» наследования компонентов концепции/движка

На уровне сеансов виджет содержит кадр значений процедуры обчёта. Этот кадр инициируется и используется в случае наличия процедуры обчёта. 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 обчёта. 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 не обработанных на уровень выше.

3.2 Проект

Непосредственная конфигурация и свойства конечного интерфейса визуализации содержатся 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. 4.

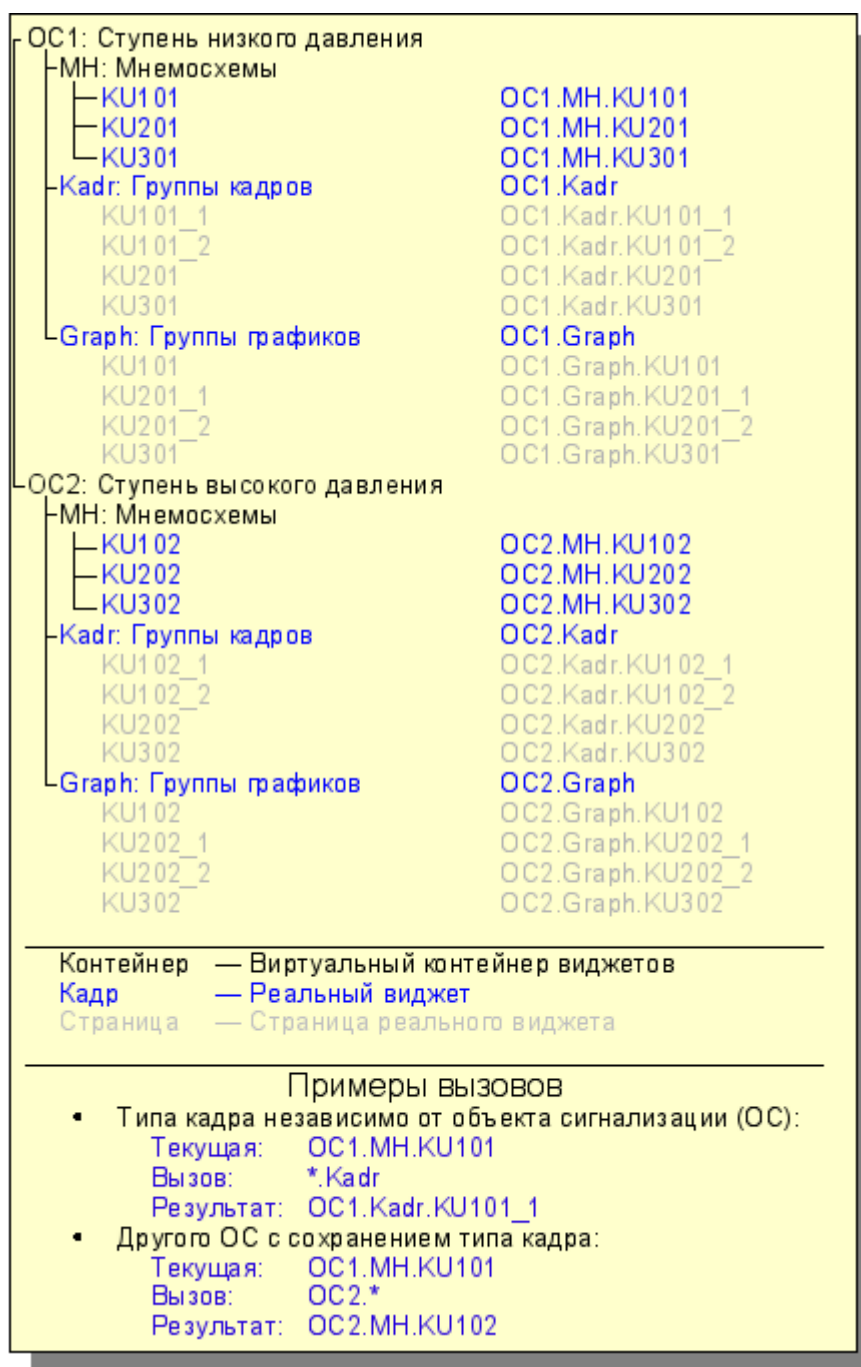


Рис.4 Иерархическое представление компонентов проекта классического интерфейса ВУ технологического процесса.

В соответствии с рис.3 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).

В дополнении к стандартным свойствам абстрактного виджета («Widget») элементы страницы, и сами страницы, сессии получают свойства: хранения кадра значений вычислительной процедуры, обчёта процедур и механизм обработки событий. Страницы сессии, в дополнение ко всему, содержат контейнер следующих по иерархии страниц. Сессия в целом обчитывается 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, не активная 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 реализовано на основе обработки событий по сценарию, в атрибуте активного виджета “*evProc*”. Сценирий этого атрибута записывается в виде списка команд с синтаксисом: *<event>:<com>:<prm>* где:

event — ожидаемое событие;

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* как на прямую, так и в виде шаблона (например: */pg_so/*/*/\$*).

prev — открытие предыдущей страницы. Открываемая страница указывается в параметре *prm* как на прямую, так и в виде шаблона (например: */pg_so/*/*/\$*).

Специальные символы шаблона расшифровываются следующим образом:

pg_so — прямое имя требуемой страницы, с префиксом. Требуется обязательного соответствия и используется для идентификации предыдущей открытой страницы;

I — имя новой страницы в общем пути, без префикса. Игнорируется при обнаружении предыдущей открытой страницы;

*** — страницы 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 связке с выше описанным 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.

pgGrp — Группа страниц. Используется для связки контейнеров страниц со страницами.

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 ни кто 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 отличаться. 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 обчёта виджетов;
- мапированные события (префикс: `map_`), события полученные из карты событий.

В таблице 1 приведён перечень стандартных событий, поддержка которых должна быть обеспечена в визуализаторах СВУ.

Таблица 1. Стандартные события

Id	Описание
<i>Клавиатурные события:</i> key_[pres rels][Ctrl Alt Shift]{Key}	
*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 – “v”.
*PageUp	Страницы вверх – «PageUp”.
*PageDown	Страницы вниз – «PageDown”.
*F1 – *F35	Функциональная клавиша от “F1” до “F35”.
*Space	Пробел – “ «.”

Id	Описание
*Apostrophe	Апостроф – "'".
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 – "'".
<i>Мышиные события:</i>	
key_mouse[Pres Rels]Left	Нажата/отпущена левая кнопка мыши.
key_mouse[Pres Rels]Right	Нажата/отпущена правая кнопка мыши.
key_mouse[Pres Rels]Midle	Нажата/отпущена средняя кнопка мыши.
key_mouseDbClick	Двойное нажатие левой кнопки мыши.
<i>События базового элемента элементов формы FormEl:</i>	
ws_LnAccept	Установлено новое значение в строке ввода.
ws_TxtAccept	Изменено значение редактора текста.
ws_ChkOn	Флажок установлен.
ws_ChkOff	Флажок снят.
ws_BtPress	Кнопка нажата.
ws_BtRelease	Кнопка отпущена.
ws_BtToggleOn	Кнопка вдавлена.
ws_BtToggleOff	Кнопка освобождена.
ws_CombChange	Изменено значение поля выбора.
ws_ListChange	Изменен текущий элемент списка.

3.5 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 использование виджета как при разработке так и при исполнении проекта;
- “w” — право на модификацию при разработке;
- “x” — право на обработку модифицирующих событий при исполнении.

3.6 Прimitives 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 2 приведён перечень примитивов виджетов(базовых элементов отображения).

Таблица 2. Библиотека примитивов виджетов(базовых элементов отображения)

Id	Наименование	Функция
ElFigure	Элементарные графические фигуры	<p>Примитив является основой для отрисовки элементарных графических фигур со всевозможной комбинацией их в одном объекте. Предусматривается поддержка следующих элементарных фигур:</p> <ul style="list-style-type: none"> • Линия. • Дуга. • Кривая безье. • Градиент замкнутого пространства. <p>Для всех фигур содержащихся в виджете устанавливаются единые свойства толщины, цвета, стрелок и т.д.</p>
FormEl	Элементы формы.	<p>Включает поддержку стандартных компонентов формы:</p> <ul style="list-style-type: none"> • Редактирование строки. • Редактирование текста. • Флажок. • Кнопка. • Поле выбора из списка. • Список.
Text	Текст	Элемент текста(метки). Характеризуется типом шрифта, цветом, ориентацией и выравниванием.
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 поддержку OpenGL!
Diagram	Диаграмма	Элемент диаграммы с поддержкой возможности отображения нескольких потоков трендов и различных режимов отображения, от минималистического до полноэкранного, двухмерного, трёхмерного, кругового и т.д.
Protocol	Протокол	Элемент протокола с поддержкой различных режимов работы при различных размерах и установках.
Document	Документ	Элемент формирования отчётных и других документов на основе указанных данных.
Function	Функция API объектной модели OpenSCADA	Невизуальный, на стороне исполнения, виджет, позволяющий включать вычислительные функции объектной модели OpenSCADA в CBY.

Id	Наименование	Функция
Box	Контейнер	Содержит механизм размещения других виджетов с целью формирования новых, более сложных виджетов и страниц конечной визуализации.
Link	Связующая линия	Данный элемент является не только визуальным, но и логическим, и служит для визуальной и логической связи элементов контейнера (Box) между собой. Связи накладываются между узловыми точками, описанными для виджетов индивидуально. Узловые точки могут описывать только визуальные связи или же содержать логические свойства для наложения связей.

Каждый примитив, и виджет вообще, содержит общий набор свойств/атрибутов в составе приведённом в таблице 3:

Таблица 3. Общий набор свойств/атрибутов в виджете

Id	Имя	№	Значение
id	Id	1	Идентификатор элемента. Атрибут только для чтения, призванный предоставить информацию об идентификаторе элемента.
name	Name	2	Имя элемента. Модифицируемое имя элемента.
dscr	Description	3	Описание элемента. Текстовое поле для прикрепления к виджету краткого описания.
path	Path	4	Полный путь к данному элементу. Атрибут только для чтения, призванный предоставить информацию о полном(абсолютном) пути к данному элементу.
en	Enabled	5	Состояние элемента – «Включен». Отключенный элемент не отображается при исполнении.
active	Active	6	Состояние элемента – «Активный». Активные элементы могут 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 фигуры по горизонтали.
geomYsc	Geometry:y scale	14	Масштаб фигуры по вертикали.
geomZ	Geometry:z	11	Геометрия, координата “z” (уровень) элемента на странице.
geomMargin	Geometry:margin	12	Геометрия, поля элемента.
<i>Дополнительные атрибуты активного виджета (атрибут active установлен)</i>			
event	Event	–	Специальный атрибут для сбора событий виджета в списке вида: <i>ws_BtPress;key_pres1;key_presCtrl2</i> . Данный атрибут доступен только в сеансе. Доступ к атрибуту защищён ресурсом, с целью избежания потери событий.

Id	Имя	№	Значение
evProc	Events process	—	Атрибут для хранения сценария обработки событий непосредственного управления пользовательским интерфейсом. Сценарий представляет собой список команд интерфейсу визуализации, генерируемых при поступлении события (атрибут event).

3.6.1 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 градиент. Основываясь уже на этих базовых фигурах можно строить производные фигуры комбинируя базовыми.

Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 4.


Таблица 4. Набор дополнительных свойств/атрибутов в примитиве ElFigure

Id	Имя	№	Значение
lineWdth	Line:width	20	Ширина линии.
lineClr	Line:color	21	Цвет линии.
lineDecor	Line:decorate	22	Оформление линии (Без декорации, Труба).
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.
elLst	Element's list	27	<p>Список графических примитивов в формате:</p> <ul style="list-style-type: none"> • Линия. Форма записи в списке: <line:p1:p2:width:color:border_width:border_color> • Дуга. Форма записи в списке: <arc:p1:p2:p3:p4:p5:width:color:border_width:border_color> • Кривая Безье. Форма записи в списке: <bezier:p1:p2:p3:p4:width:color:border_width:border_color> • Заливка. Форма записи в списке: <fill:p1:p2:....:pn:fillClr:fillImg>
<i>Атрибуты для каждой точки из списка графических фигур elLst</i>			
p{n}x	Point {n}:x	30+n*2	Координата “x” точки {n}.
p{n}y	Point {n}:y	30+n*2+1	Координата “y” точки {n}.

3.6.2 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 5.

Таблица 5. Набор дополнительных свойств/атрибутов в примитиве FormEl

Id	Имя	№	Значение
elType	Element type	20	Тип элемента (Строка редактирования; Редактор текста; Флажок; Кнопка; Выбор из списка; Список). От его значения зависит перечень дополнительных атрибутов.
<i>Строка редактирования:</i>			
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: <i>Текст</i> — указывается шаблон ввода в формате  библиотеки QT . <i>Комбобокс</i> — содержит список значений редактируемого комбобокса. <i>Целое</i> — содержит конфигурацию поля ввода целочисленного представления в формате: <Минимум>:<Максимум>:<Шаг изменения>:<Префикс>:<Суффикс>. <i>Вещественное</i> — содержит конфигурацию поля ввода вещественного представления в формате: <Минимум>:<Максимум>:<Шаг изменения>:<Префикс>:<Суффикс>:<Число знаков после запятой>.
<i>Редактор текста:</i>			
value	Value	21	Содержимое редактора.
wordWrap	Word wrap	22	Автоматический перенос текста по словам.
<i>Флажок:</i>			
name	Name	2	Имя/метка флажка.
value	Value	21	Значение флажка.
<i>Кнопка:</i>			
name	Name	2	Имя, надпись на кнопке.
value	Value	21	Значение для фиксированной кнопки.
img	Image	22	Изображение на кнопке.
color	Color	23	Цвет кнопки.
checkable	Checkable	24	Признак функционирования как фиксированная кнопка.
<i>Выбор из списка:</i>			
value	Value	21	Текущее значение списка.
items	Items	22	Перечень элементов списка.
<i>Список:</i>			
value	Value	21	Выбранное значение списка.
items	Items	22	Перечень элементов списка.

3.6.3 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 простой рамкой. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 6.

Таблица 6. 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.
font	Font:full	24	Шрифт текста в полной записи (Arial 11 1 0). Имеет приоритет выше, чем свойства шрифта по отдельности.
fontFamily	Font:family	25	Семейство шрифта.
fontSize	Font:size	26	Размер шрифта.
fontBold	Font:bold	27	Усиление шрифта.
fontItalic	Font:italic	28	Наклонность шрифта.
fontUnderline	Font:underline	29	Подчёркивание текста.
fontStrikeout	Font:strikeout	30	Перечёркивание текста.
color	Color	31	Цвет текста.
orient	Orientation angle	32	Ориентация текста, поворот на угол.
wordWrap	Word wrap	33	Автоматический перенос текста по словам.
alignment	Alignment	34	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	35	Значение текстового поля.
numbArg	Arguments number	36	Количество аргументов .
<i>Атрибуты аргументов</i>			
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: <ul style="list-style-type: none"> • <i>строка</i> : [len] — ширина строки; • <i>вещественное</i>: [width];[form];[prec] — ширина значения, форма значения («g», "f"); • <i>целое</i>: [len] — ширина значения.

3.6.4 Элемент отображения медиа-материалов (Media)

Данный примитив предназначен для проигрывания различных медиа-материалов, начиная с 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 имплементации 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 7.

Таблица 7. 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.
src	Source	24	Источник медиа-данных.
type	Type	25	Тип медиа (Image;Movie).
fit	Fit to widget size	26	Признак «Согласовать содержимое с размером виджета».
<i>Атрибуты видеоролика (Movie)</i>			
speed	Play speed	27	Скорость проигрывания, в процентах от оригинальной скорости. Если значение меньше 1%, то проигрывание прекращается.

3.6.5 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:

- Графики/тренды – позволяет строить графики из значений параметров подсистемы «Сбор данных», 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 значение. 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 8.

Таблица 8. 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.
trcPer	Tracing period (s)	24	Режим и периодичность слежения.
type	Type	25	Тип диаграммы: "Trend".
<i>Атрибуты тренда/графика (Trend)</i>			
tSek	Time:sek	26	Текущее время, секунд.
tUSek	Time:usek	27	Текущее время, микросекунды.
tSize	Size, sek	28	Размер тренда, секунды.
curSek	Cursor:sek	29	Положение курсора, секунды.
curUSek	Cursor:usek	30	Положение курсора, микросекунды.
curColor	Cursor:color	36	Цвет курсора.
sclColor	Scale:color	31	Цвет шкалы/решетки.
sclHor	Scale:horizontal	32	Режим горизонтальной шкалы/решетки: "No draw", "Grid;Markers" и "Grid and markers".
sclVer	Scale:vertical	33	Режим вертикальной шкалы/решетки: "No draw", "Grid", "Markers", "Grid and markers", «Grid (log)", «Marker (log)", «Grid and markers (log)".

Id	Имя	№	Значение
sclMarkColor	Scale:Markers:color	37	Цвет маркеров шкалы/решетки.
sclMarkFont	Scale:Markers:font	38	Шрифт маркеров шкалы/решетки.
valArch	Value archivator	34	Архиватор архивов параметров.
parNum	Parameters number	35	Количество параметров, отображаемых на одном тренде.
<i>Индивидуальные атрибуты параметров тренда/графика</i>			
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 параметра {X} под курсором.

3.6.6 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 9.

Таблица 9. Набор дополнительных свойств/атрибутов в примитиве Protocol

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.
arch	Archival	24	Архиватор архива.
tmpl	Template	25	Шаблон запроса в архиве.
time	Time, sek	26	Текущее время, секунд.
tSize	Size, sek	27	Размер запроса, секунды.
trcPer	Tracing period (s)	28	Режим и периодичность слежения.
viewOrd	View order	29	Порядок отображения: “On time”, “On level”, “On level and trigered”.
col	View columns	30	Отображаемые колонки.
itProp	Items properties	31	Количество элементов протокола с индивидуальными свойствами.
<i>Свойства элементов протокола</i>			
it{X}lev	Item {X} :levels	50+10*{X}	Уровни элемента.
it{X}tmpl	Item {X} :template	50+10*{X} +1	Шаблон элемента.
it{X}fnt	Item {X} :font	50+10*{X} +2	Шрифт элемента.

Id	Имя	№	Значение
it{X}color	Item {X}:color	50+10*{X} +3	Цвет элемента.
it{X}blink	Item {X}:blink	50+10*{X} +4	Мигание элемента.

3.6.7 Контейнер (Box)

Примитив контейнера, используется для формирования составных виджетов и/или страниц пользовательского интерфейса. Перечень дополнительных свойств/атрибутов данного примитива приведён в таблице 10.

Таблица 10. Набор дополнительных свойств/атрибутов в примитиве Box

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.
pgOpen	Page:open state	24	Признак «Страница открыта». Используется при выполнении примитивом роли страницы.
pgNoOpenProc	Page:no open process	25	Признак «Исполнять страницу даже если она закрыта». Используется при выполнении примитивом роли страницы.
pgOpenSrc	Page:open source	26	Полный адрес страницы, открывшей данную. Используется при выполнении примитивом роли страницы.
pgGrp	Page:group	27	Группа страницы. Используется при выполнении примитивом роли страницы.
pgFullScr	Page:full screen	28	Режим открытия страниц, основанных на даном примитиве, на полный экран.

3.7 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, принадлежащий библиотеке виджетов входят следующие:

- {DB_TBL} — Таблица с виджетами, принадлежащими библиотеке (структура «LibWigets”).
- {DB_TBL}_io — Таблица с рабочими свойствами виджетов этой библиотеки и вложенных виджетов контейнерных виджетов (структура «LibWidgetIO”).
- {DB_TBL}_uio — Таблица с пользовательскими свойствами виджетов этой библиотеки и вложенных виджетов 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, USER, GRP, PERMIT) — Библиотеки виджетов <ID>.
 - *ID* — идентификатор;
 - *NAME* — имя;
 - *DSCR* — описание;
 - *DB_TBL* — БД с виджетами;
 - *ICO* — закодированное (Base64) изображение иконки библиотеки;
 - *USER* — владелец библиотеки;
 - *GRP* — группа пользователей библиотеки;
 - *PERMIT* — права доступа к библиотеке.
- LibWidgets(ID, ICO, PARENT, PROC, USER, GRP, PERMIT) — 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;
 - *USER* — владелец виджета;
 - *GRP* — группа пользователей виджета;
 - *PERMIT* — права доступа к виджету.
- LibWidgetIO(IDW, ID, SELF_FLG, CFG_TMPL, CFG_VAL) — Рабочие атрибуты <ID> виджета <IDW>.
 - *IDW* — идентификатор виджета;
 - *ID* — идентификатор IO;
 - *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) — Включенные в контейнер <IDW> виджеты <ID>.
 - *IDW* — идентификатор виджета;
 - *ID* — идентификатор экземпляра вложенного виджета;
 - *PARENT* — адрес виджета основы в виде /wlb_originals/wdg_Box .
- LibWidgetMime(ID, MIME, DATA) — Audio, video, media и другие ресурсы виджетов библиотеки.
 - *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) — 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.

- ProjPage(OWNER, ID, ICO, PARENT, PROC, USER, GRP, PERMIT, FLGS) — 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;
 - *USER* — владелец страницы;
 - *GRP* — группа пользователей страницы;
 - *PERMIT* — права доступа к странице;
 - *FLGS* — флаги страницы.
- ProjPageIO(IDW, ID, 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) — Включенные на страницы виджеты. Структура фактически совпадает с таблицей LibWidgetIncl.

3.8 Сервисные интерфейсы для доступа к элементам СВУ

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.

3.8.1 Архивные данные

Для доступа к архивным данным системы OpenSCADA из элементов СВУ, например из визуализаторов, для построения трендов и других диаграмм необходим гибкий и оптимизированный интерфейс, в виду большого объема и широты возможных режимов работы подсистемы архивирования значений. В основе интерфейса управления OpenSCADA лежит язык XML, на котором передаются команды и получаются ответы. Локально, работа с XML сводится к работе с деревом объектов (тегов) XML, исключая стадию преобразования в символьный поток и парсинг.

Для запроса данных архива, в подсистеме архивирования, объекта архива значения, и объекте атрибута параметра, подсистемы сбора данных, предусмотрим команды <info path="{a_p_addr}/%2fserve%2f0"/> и <get path="{a_p_addr}/%2fserve%2f0"/> для запроса информации об архиве и значений архива, соответственно. Атрибуты данных команд, предусматривающие различные механизмы запроса, представим в таблице 11.

Таблица 11. Атрибуты команд запроса информации об архиве и архивных данных

Id	Имя	Значение
Команда запроса информации об архиве: <info path="{a_p_addr}/%2fserve%2f0"/>		
arch	Установка имени архиватора архива	Архиватор архива для которого определять параметры.
end	Контроль вершины архива в данном архиваторе	В результате запроса указывает на реальную вершину архива значений в данном архиваторе <arch>. В случае отсутствия архива атрибут устанавливается в "0".

Id	Имя	Значение
beg	Контроль глубины архива в данном архиваторе	В результате запроса указывает на реальную глубину архива значений в данном архиваторе <arch>. В случае отсутствия архива атрибут устанавливается в "0".
per	Контроль периодичности архива в данном архиваторе	В результате запроса указывает на реальную периодичность значений в данном архиваторе <arch>. В случае отсутствия архива атрибут устанавливается в "0".
vtp	Контроль типа значений архива/параметра	Возвращает код типа значений архивных данных: 0 – Boolean, 1 – Integer, 4 – Real, 5 – String. Только данное свойство доступно в случае запроса у параметра без архива!
<i>Команда запроса архивных и/или текущих данных: <get path="{a_p_addr}/%2fserv%2f0"/></i>		
tm	Установка и контроль времени	Время запрашиваемого значения или вершины блока архива значений. Если атрибут не указан или равен "0", то возвращается последнее значение. Значение данного атрибута устанавливается в значение времени полученного значения или вершины блока архивных данных.
tm_grnd	Установка и контроль времени основания/начала архива	Указывает основание/начало архива. Если атрибут не указан или равен "0", то производится запрос одного значения. Значение данного атрибута устанавливается в значение времени основания блока архивных данных.
per	Установка и контроль периодичности получаемых значений	Периодичность запрашиваемых значений. Если атрибут не указан, равен "0" или меньше чем реальная периодичность архива, то значения будут возвращаться с реальной периодичностью архива и значение данного атрибута установится в значение реальной периодичности. Если значение атрибута больше реальной периодичности, то значения архива будут усредняться к указанной периодичности.
arch	Установка и контроль архиватора архива	Указывает у какого архиватора запрашивать значения. Если архиватор не указан, то запрос будет производиться последовательно, с приоритетностью от лучшего к худшему. Имя архиватора, у которого получены значения будет указано в данном атрибуте при возврате результата. Если архиватор для данного параметра не установлен или такого архиватора нет, то значений данного атрибута будет обнулено. В случае выполнения запроса пересекающего несколько архиваторов выполняется обработка только первого архиватора с указанием его имени и размерности в соответствующих атрибутах. Для запроса данных последующих архиваторов запрос повторяется отталкиваясь от размерности предыдущего запроса.

Id	Имя	Значение
mode	Установка и контроль режима передачи данных архива	<p>Указывается форма в которой желательно получать данные архива. Предусматриваются следующие режимы/формы передачи данных архивов:</p> <p>0 — Простая запись: одно значение – одна строка, без упаковки смежных значений. В случае архива строк, значения кодируются на предмет исключения символов перевода строки. Пример формы записи:</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> 34.5678 23.6543 65.8754 34.6523 </div> <p>1 — Упакованная запись по принципу сворачивания смежных значений: одно значение – одна запись. Перед каждым значением указывается его позиционный номер, отделённый пробелом:</p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> 0 34.5678 1 23.6543 4 65.8754 6 34.6523 </div> <p>2 — Массив бинарных, не упакованных значений, кодированный Mime Base64. Не может быть использован для строковых архивов.</p>
real_precision	Установка точности вещественных чисел	Указывает с какой точностью передавать данные значений вещественного типа, в режиме “0” и “1”. По умолчанию эта точность составляет 6 значащих цифр.
round_perc	Установка процента округления	Указывает процент округления смежных числовых значений, для режима “1”.

3.8.2 Доступ к значениям атрибутов элементов визуализации (виджеты)

С целью предоставления унифицированного, группового и сравнительно быстрого доступа к значениям атрибутов визуальных элементов предусмотрена сервисная функция визуального элемента «/serv/0» и команды получения/установки значений атрибутов: `<get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2f0"/>` и `<set path="/UI/VCAEngine/{wdg_addr}/%2fserv%2f0"/>`. Атрибуты данных команд, предусматривающие различные механизмы запроса, представим в таблице 12.

Таблица 12. Атрибуты команд получения/установки атрибутов визуальных элементов

Id	Имя	Значение
<i>Команда запроса визуальных атрибутов виджета: <get path="/UI/VCAEngine/{wdg_addr}/%2fserv%2f0"/></i>		
tm	Время/счётчик изменений	Установка времени/счётчика изменений для запроса только изменившихся атрибутов.
<code><el id="{attr}" pos="{a_id}">{val}</el></code>	Формирование дочерних элементов с результатами атрибутов	В дочернем элементе указываются: строковых идентификатор {attr} атрибута, индекс {a_id} атрибута и его значение {val}.
<i>Команда установки визуальных атрибутов виджета: <set path="/UI/VCAEngine/{wdg_addr}/%2fserv%2f0"/></i>		

Id	Имя	Значение
<el id="{attr}">{val}</el>	Установка атрибутов	В дочерних элементах указывается идентификатор атрибута {attr} и его значение {val}.

3.8.3 Доступ к открытым страницам сеанса

С целью унификации и оптимизации доступа к открытым страницам предусмотрена сервисная функция сеанса «/serv/0» и команда запроса перечня открытых страниц: <openlist path="/UI/VCAEngine/ses_{Session}/%2fserv%2f0"/>. Результатом запроса перечня открытых страниц являются дочерние элементы <el>{OpPage}</el> содержащие полный путь открытой страницы.

Кроме перечня открытых страниц данный 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.

3.8.4 Манипуляция сеансами проектов

Для предоставления унифицированного механизма манипуляции сеансами, визуализаторам СВУ, в модуле движка СВУ (VCAEngin) предусмотрена сервисная функция «/serv/0» и команды запроса перечня открытых сеансов, подключения/создания нового сеанса и отключения/удаления сеанса: <list path="/UI/VCAEngine/%2fserv%2f0"/>, <connect path="/UI/VCAEngine/%2fserv%2f0"/> и <disconnect path="/UI/VCAEngine/%2fserv%2f0"/> соответственно. Атрибуты данных команд, предусматривающие различные механизмы запроса, представим в таблице 13.

Таблица 13. Атрибуты команд механизма манипуляции сеансами

Id	Имя	Значение
<i>Команда запроса перечня открытых сеансов для проекта: <list path="/UI/VCAEngine/%2fserv%2f0"/></i>		
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.
<i>Команда подключения/открытия сеанса: <connect path="/UI/VCAEngine/%2fserv%2f0"/></i>		
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.
prj	Установка имени проекта	Используется для открытия нового сеанса для указанного проекта и если атрибут {sess} не указан.
<i>Команда отключения/закрытия сеанса: <disconnect path="/UI/VCAEngine/%2fserv%2f0"/></i>		
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.

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. 5).

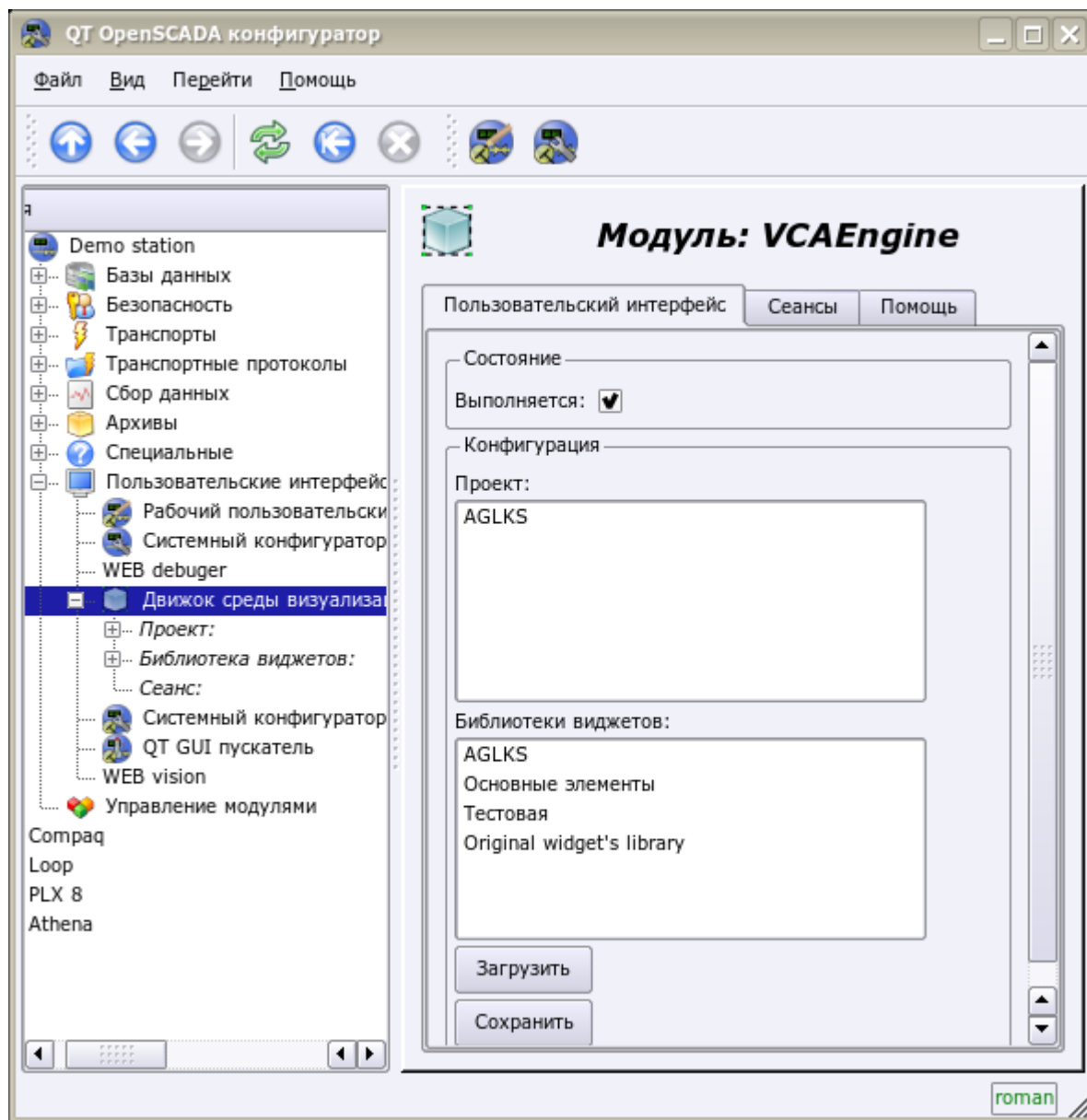


Рис.5 Главная конфигурационная страница модуля.

Конфигурация контейнеров виджетов, в лице библиотек виджетов и проектов фактически не отличается и выполняется посредством страницы на рис. 6. Единственное между ними различие в том, что библиотека виджетов содержит – виджеты, а проект – страницы. Кроме этого библиотека виджетов содержит вкладку конфигурации Mime-данных, используемых виджетами (рис.7).

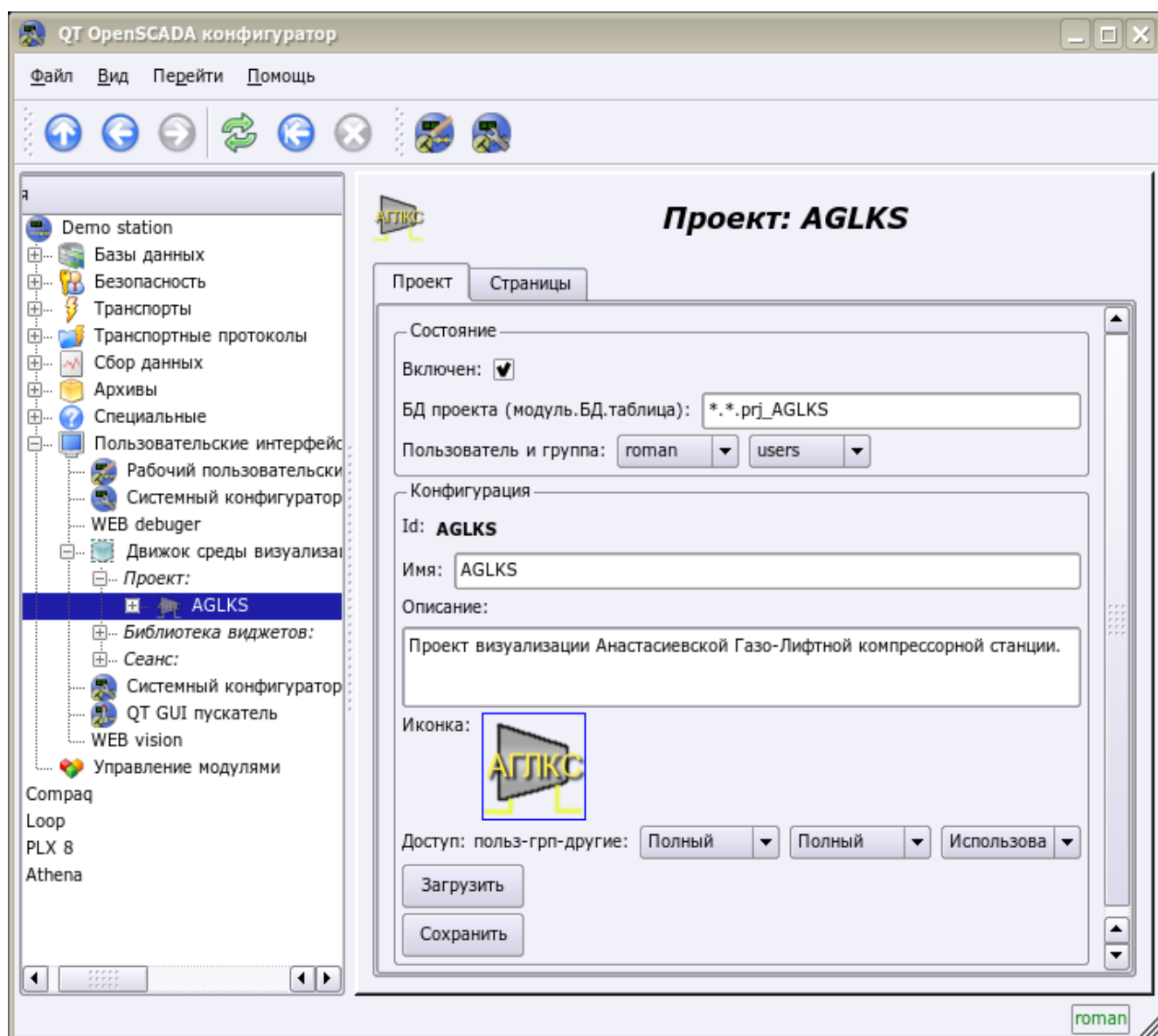


Рис.6 Страница конфигурации библиотек виджетов и проектов.

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.
- Сохранить/загрузить контроллер в БД.

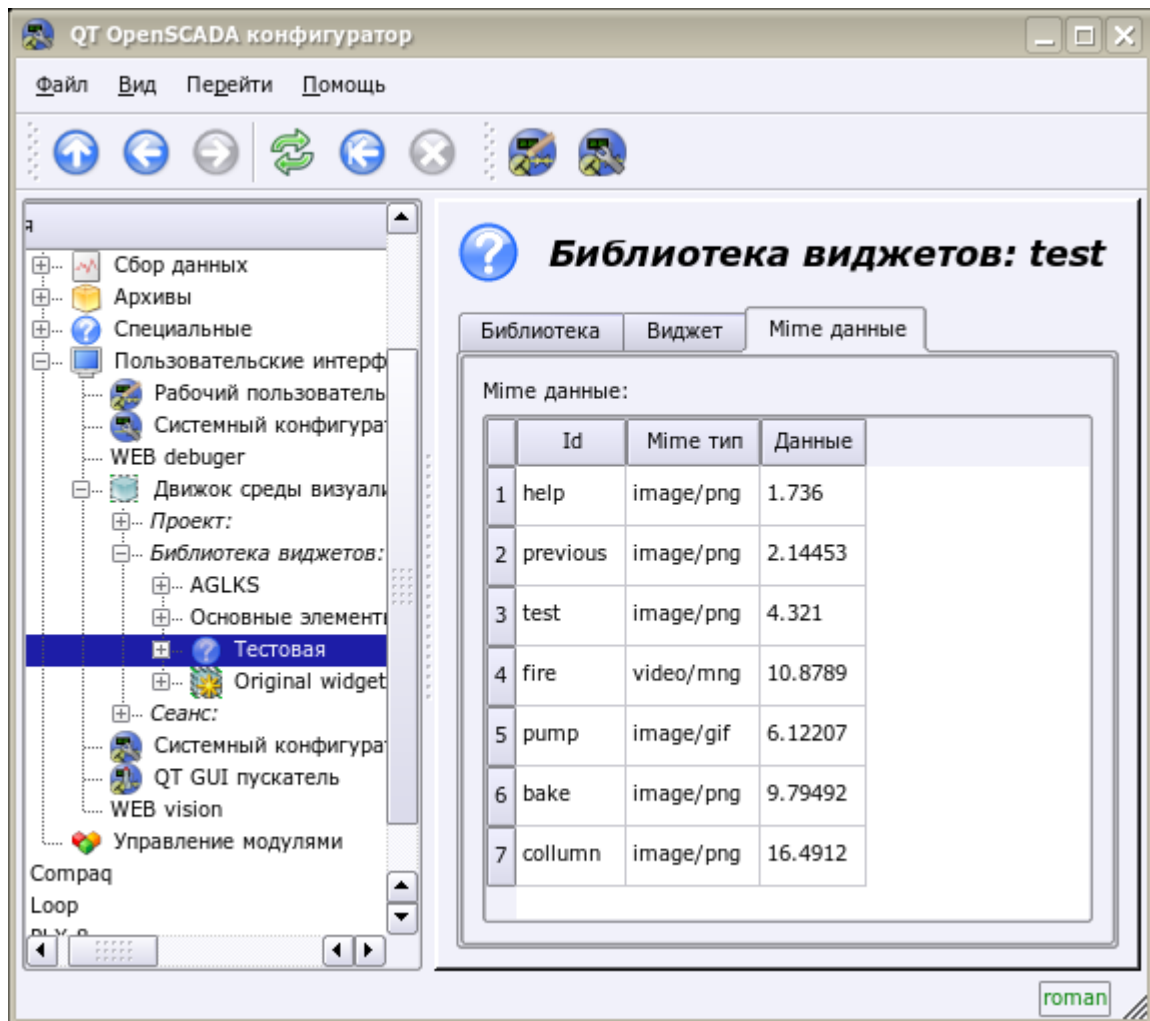


Рис.7 Вкладка конфигурации mime-данных страницы библиотек виджетов.

Конфигурация сеанса проекта значительно отличается от конфигурации проекта (рис. 8), однако, также, содержит страницы проекта.

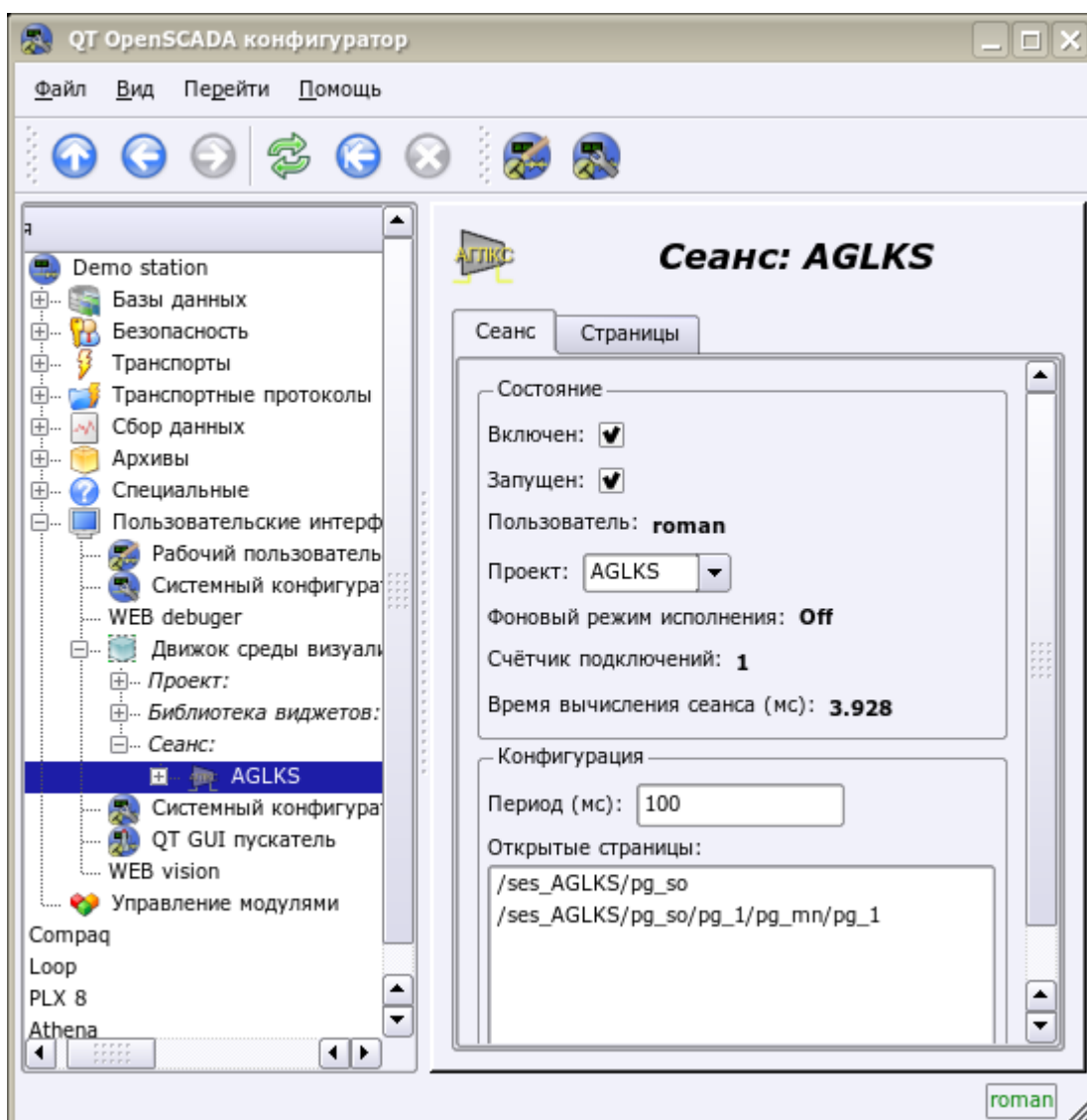


Рис.8 Страница конфигурации сеансов проектов.

С помощью этой страницы можно установить:

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

Страницы конфигурации визуальных элементов, расположенных в разных контейнерах, могут сильно отличаться, однако это отличие фактически заключается в наличии или отсутствии отдельных вкладок. Главная вкладка визуальных элементов фактически везде одинакова, отличаясь на одно конфигурационное поле (рис. 9). У страниц присутствуют вкладки дочерних страниц и вложенных виджетов. У контейнерных виджетов содержится вкладка вложенных виджетов. Все визуальные элементы содержат вкладку атрибутов (рис. 10), кроме логических контейнеров проектов. Элементы на уровне которых можно формировать пользовательскую процедуру и определять связи содержат вкладки «Обработка» (рис. 11) и «Связи» (рис.12).

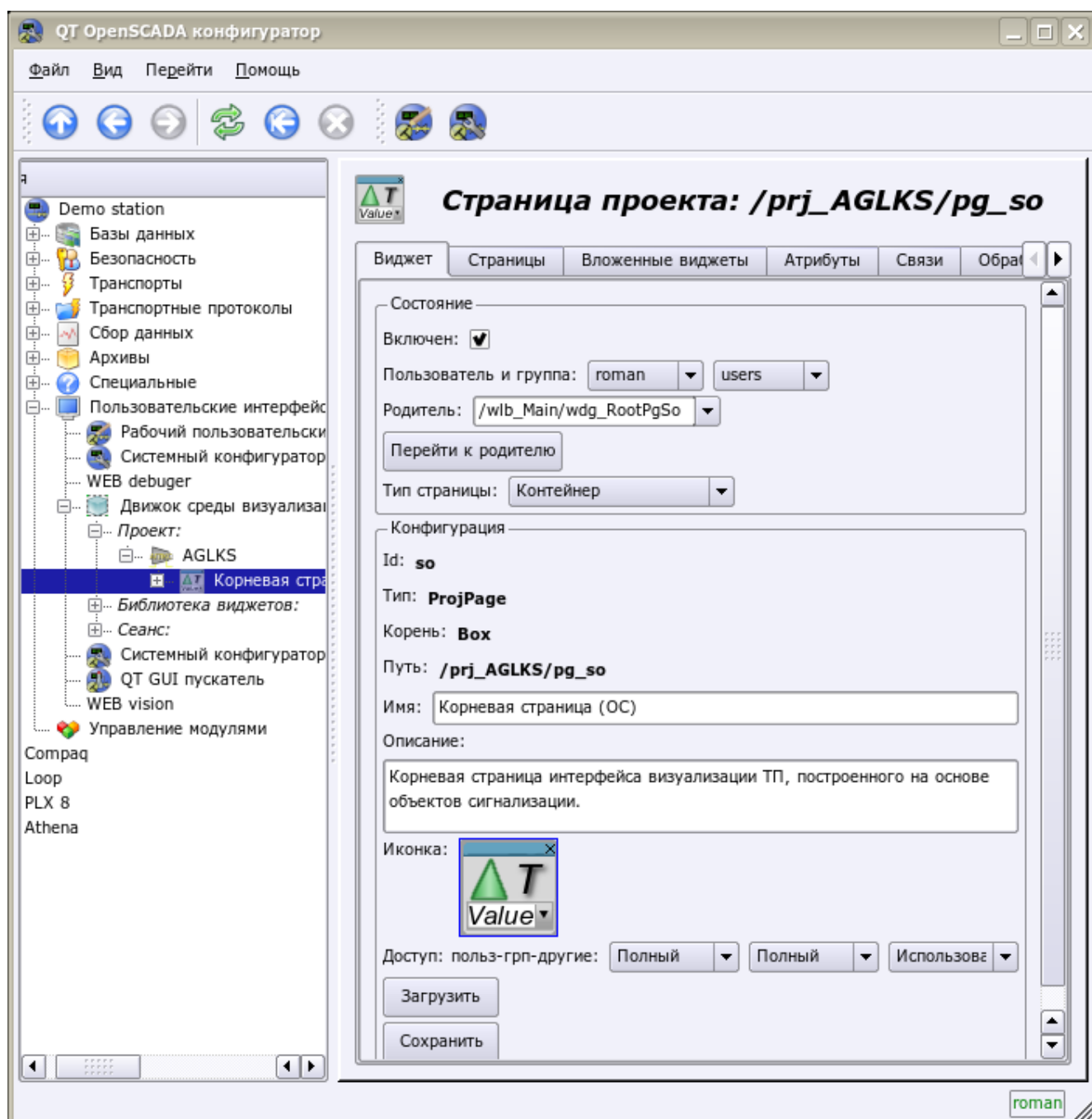


Рис.9 Главная вкладка конфигурации визуальных элементов.

С помощью этой страницы можно установить:

- Состояние элемента, а именно: «Включен», владельца, группу элемента, родительский элемент и переход к нему. Для страницы в состоянии указывается тип страницы.
- Идентификатор, тип, корень, путь, имя, описание и иконку элемента.
- Права доступа к элементу.
- Сохранить/загрузить контроллер в БД.

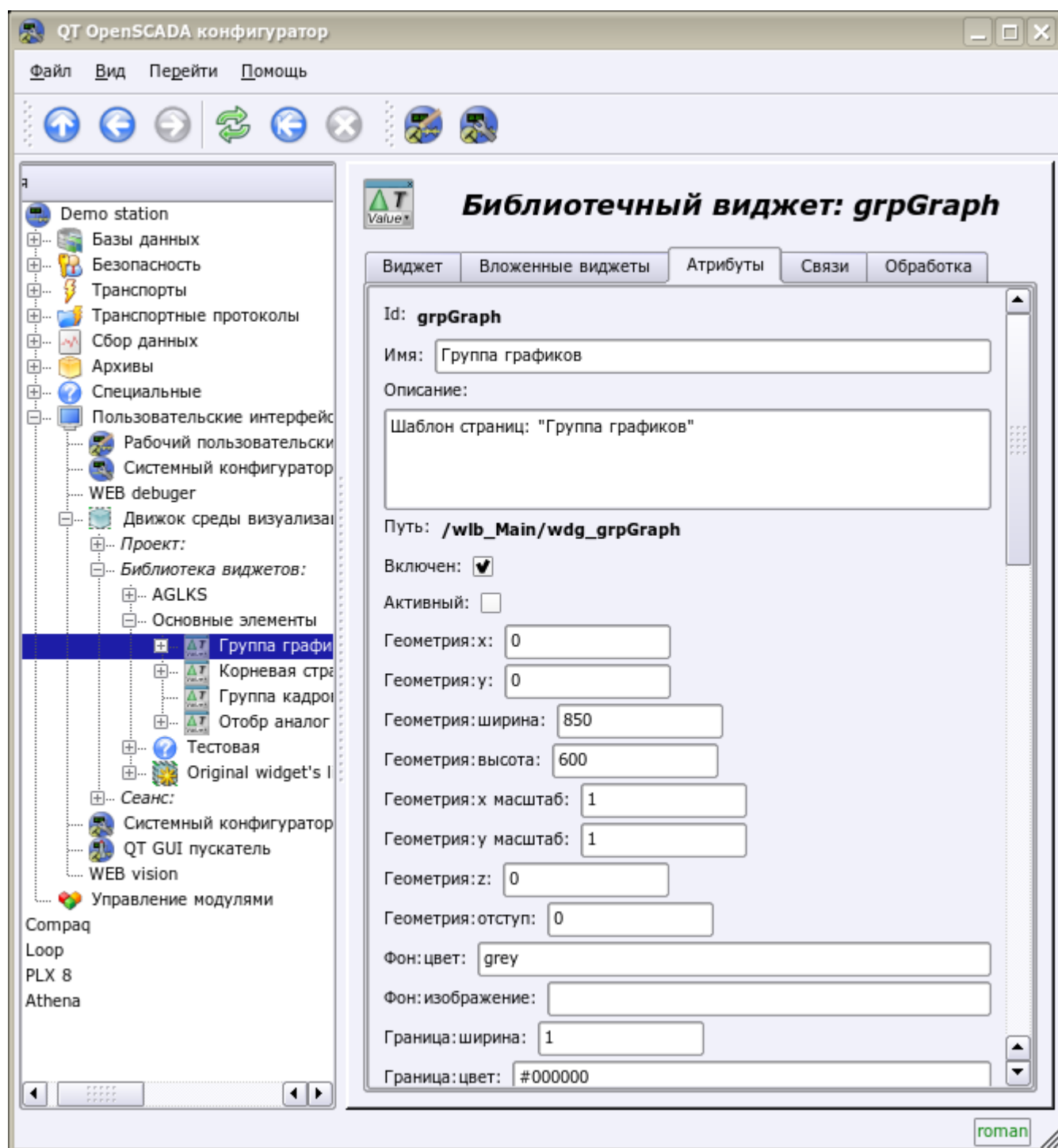


Рис.10 Вкладка атрибутов визуальных элементов.

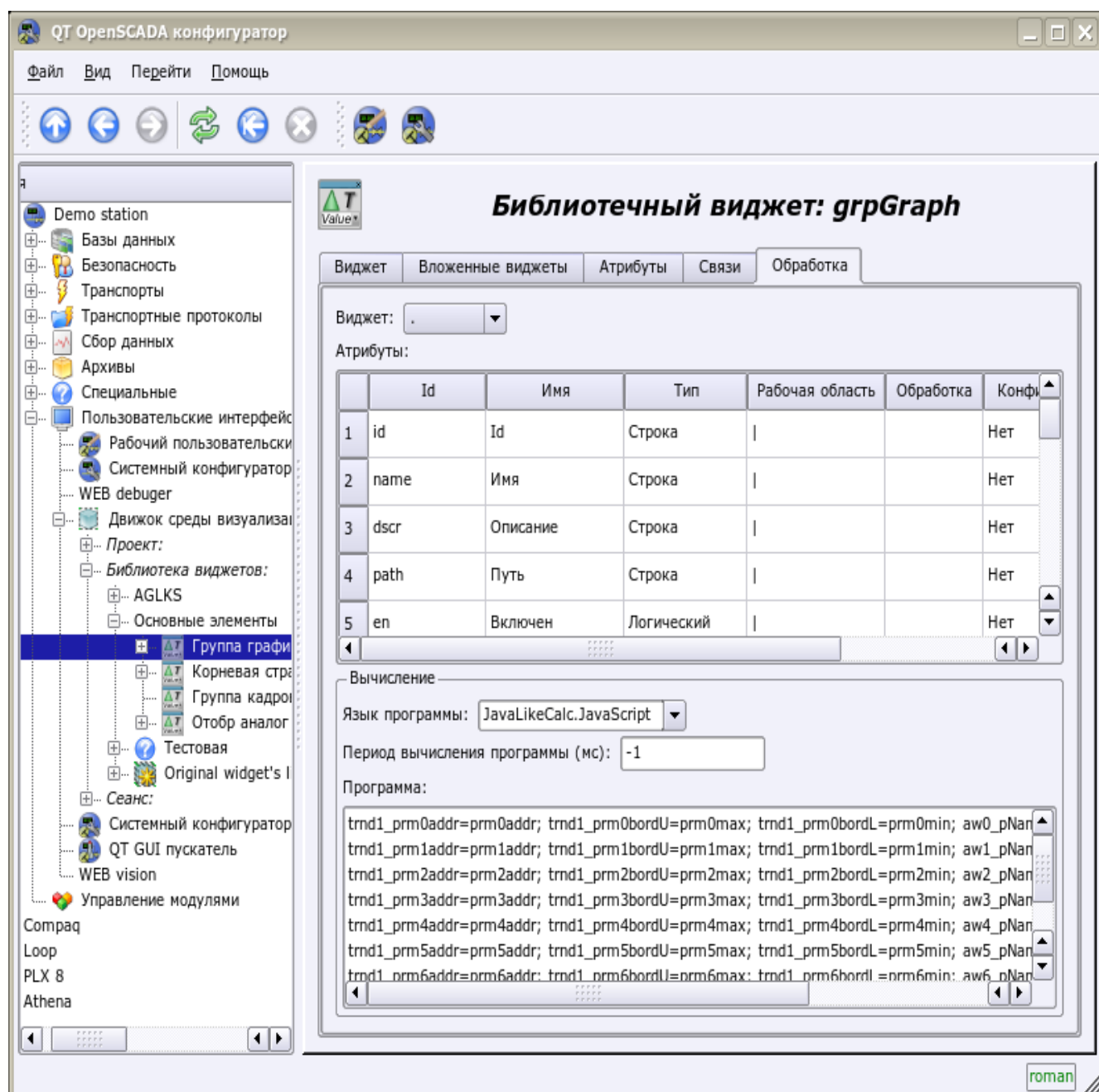


Рис.11 Вкладка обработки визуальных элементов.

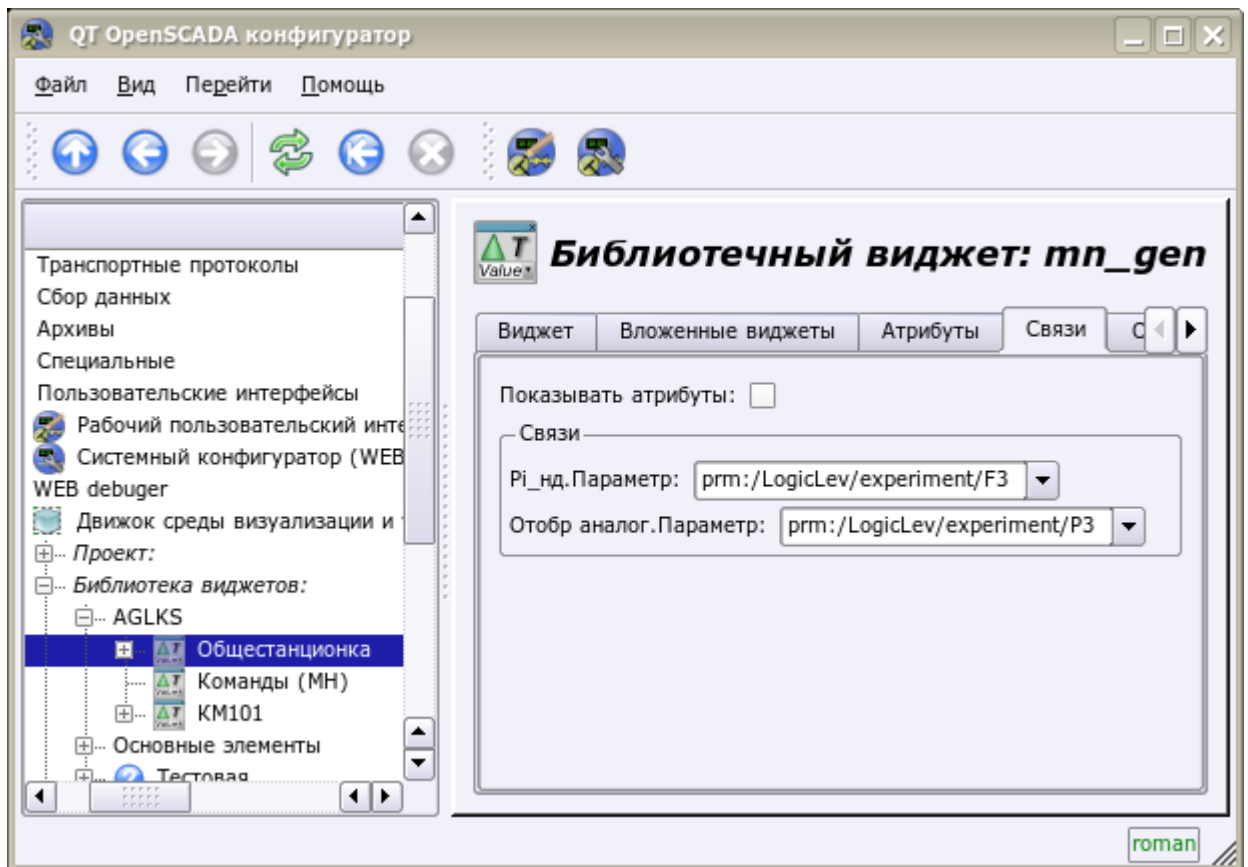


Рис.12 Вкладка связей визуальных элементов.

Заключение

Модуль ещё не реализует всех запланированных функций, однако уже позволяет выполнять базовый их набор, с помощью которого уже можно строить достаточно развитые интерфейсы ВУ.