API системы OpenSCADA

Статус:	Публикация
Версия:	0.5.0
Содержание:	Содержит исчерпывающее описание API системы OpenSCADA. Также содержится руководство по программированию ядра системы и созданию модулей для неё.

Оглавление

1 Внутренняя структура, API системы OpenSCADA.

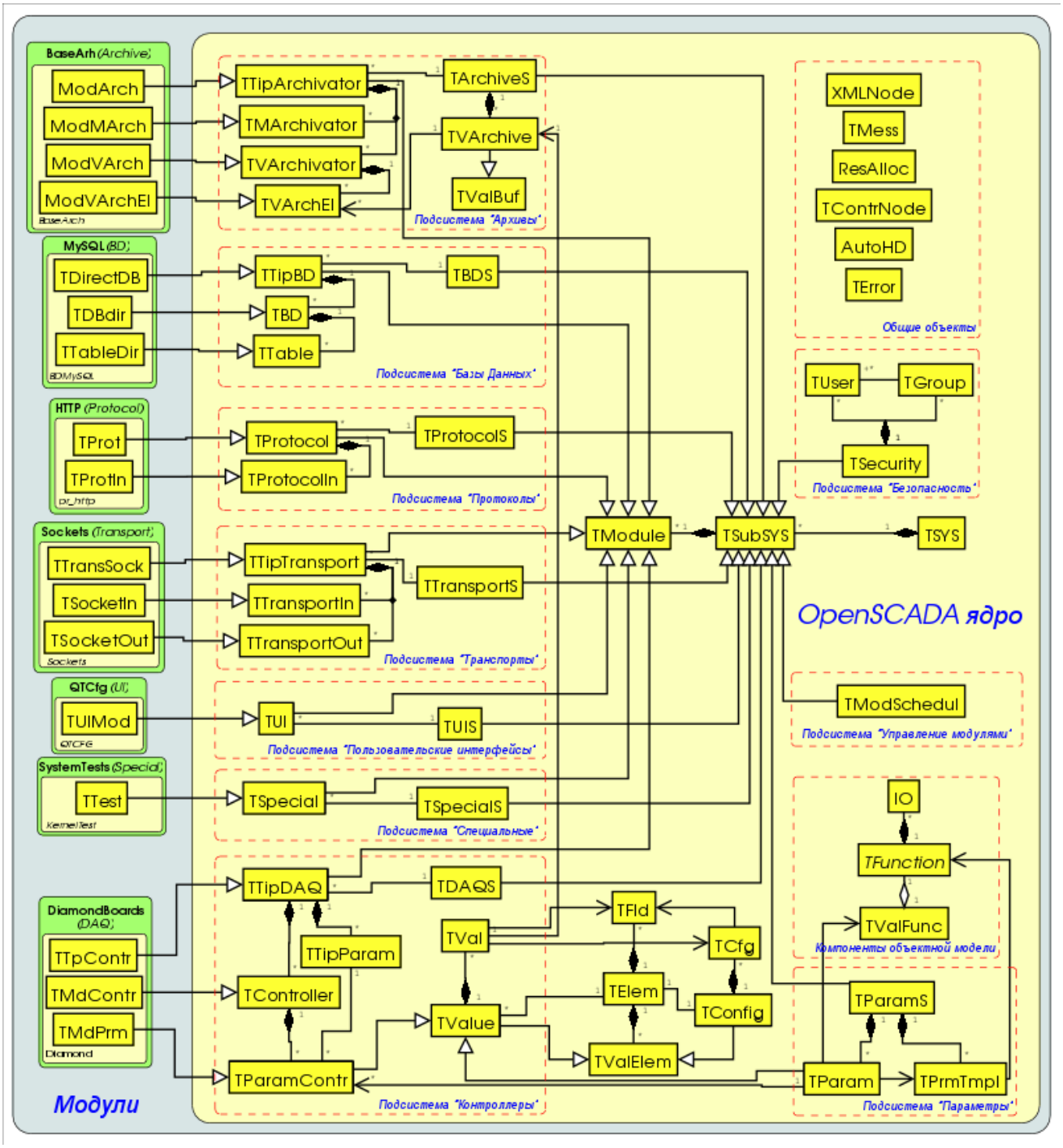

Рис. 1. Статическая диаграмма классов

2 Общая структура системы. Модульность (TSubSYS, TModule)

Корнем, от которого строится вся система, является объект TSYS. Корень содержит подсистемы (TSubSYS). Подсистемы могут быть: обычными и модульными. Отличие модульных подсистем четко прослеживается на рис. 1. Так, модульные подсистемы обязательно содержат список модульных объектов (TModule), например подсистема архивы TArchiveS содержит модульные объекты TTipArchivators. В тоже время обычная подсистема таких объектов не содержит. Например подсистема безопасности TSecurity (рис.2).

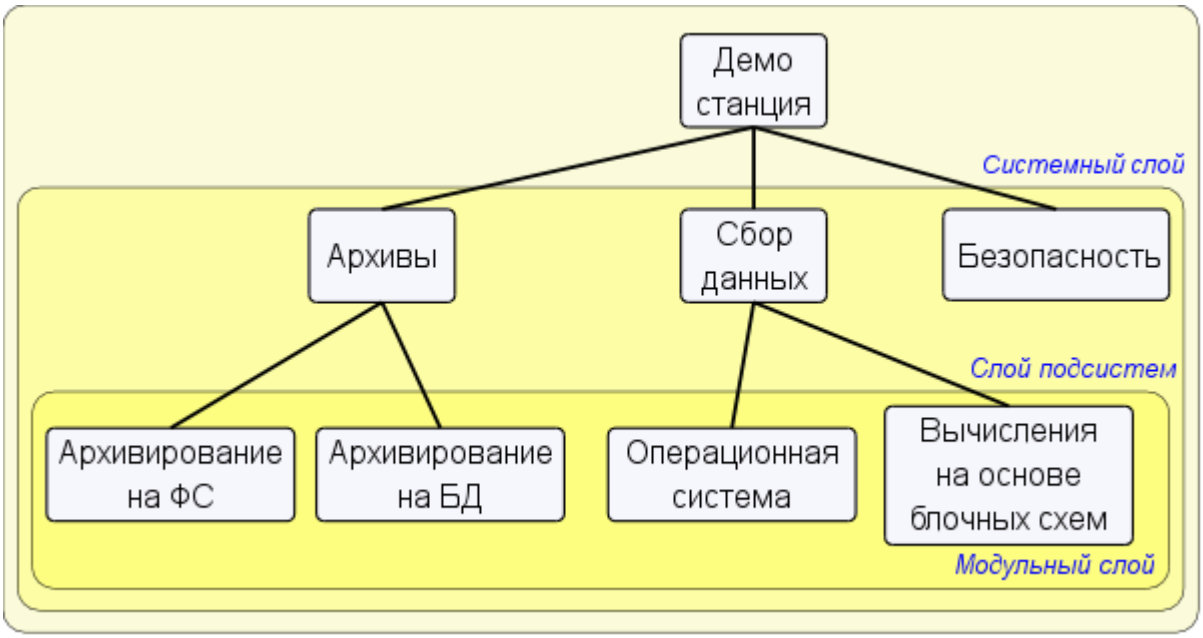

Рис. 2. Слоистая структура системы OpenSCADA.

В процессе инициализации корня (TSYS) определяется глобальная переменная SYS. Переменная SYS может использоваться для прямого обращения к корню системы из любого её узла. Инициализация корня выполняется единожды из главной вызывающей функции. После запуска управление захватывается объектом системы до остановки. Корневой объект концентрирует все обшесистемные функции системы OpenSCADA.

Продолжением корневого объекта (TSYS), выполняющего функции обслуживания потока системных сообщений, выступает объект TMess. Объект доступен посредством глобальной переменной Mess, которая инициализируется корнем системы. Объект содержит функции кодирования, декодирования и локализации сообщений.

В подсистемах (TSubSYS) реализуются функции характерные для каждой подсистемы индивидуально, с общим для всех подсистем доступом через объект TSubSYS. Модульная подсистема имеет возможность расширять функциональность посредством модулей. Для этой цели модульная подсистема предоставляет доступ к модулям своего типа в виде модульных объектов.

Модуль – составная часть модульной подсистемы. В общем, для всех модулей и их подсистем, модуль предоставляет информацию о себе, своём происхождении и экспортируемых функциях. Отдельно взятый модуль реализует функциональность в соответствии со своими потребностями.

2.1 Корневой объект системы (TSYS)

Наследует:	*TCntrNode.*

Данные:

Способы кодирования символьных последовательностей (enum – IO::Code):

- *PathEl* — элемент пути (символы: '/' и '%' к виду '%2f');
- *HttpURL* — адрес браузера (http url);
- *Html* — специальных символов для использования в html;
- *JavaSc* — символов конца строки для JavaScript;
- *SQL* — значения SQL-запросов;
- *Custom* — выборочное кодирование указанных символов.
- *base64* — Mime кодирование в стандарте Base64.

Виды представления целого в функциях TSYS::int2str() и TSYS::ll2str() (enum – IO::IntView):

- *Dec* — десятичное;
- *Oct* — восьмеричное;
- *Hex* — шестнадцатеричное.

Шаблоны:

- *__func__* — Полное имя вызывающей функции.
- *vmin(a,b)* — Определение минимального значения.
- *vmax(a,b)* — Определение максимального значения.

Публичные методы:

- *TSYS(int argi, char **argb, char **env);* — Инициализирующий конструктор.
- *void load( );* — Загрузка системы.
- *void save( );* — Сохранение системы.
- *int start( );* — Запуск системы. Функция завершается только с завершением работы системы. Возвращается код возврата.
- *void stop( );* — Команда остановки системы.
- *int stopSignal( );* — Код возврата в случае останова системы. Может использоваться как признак «Останов системы».
- *string id();* — Идентификатор станции.
- *string name();* — Локализованное имя станции.
- *string user();* — Системный пользователь от имени которого запущена система.
- *void list(vector<string> &list);* — Список подсистем зарегистрированных в системе.
- *bool present(const string &name);* — Проверка на наличия указанной подсистемы.
- *void add(TSubSYS *sub);* — Добавление/регистрация подсистемы.
- *void del(const string &name);* — Удаление подсистемы.
- *AutoHD<TSubSYS> at(const string &name);* — Подключение к указанной подсистеме.
- *AutoHD<TUIS> ui();* — Прямой доступ к подсистеме «Пользовательские интерфейсы».
- *AutoHD<TArchiveS> archive();* — Прямой доступ к подсистеме «Архивы».
- *AutoHD<TBDS> db();* — Прямой доступ к подсистеме «Базы данных».
- *AutoHD<TControllerS> daq();* — Прямой доступ к подсистеме «Сбор данных».
- *AutoHD<TProtocolS> protocol();* — Прямой доступ к подсистеме «Протоколы».
- *AutoHD<TTransportS> transport();* — Прямой доступ к подсистеме «Транспорты».
- *AutoHD<TSpecialS> special();* — Прямой доступ к подсистеме «Специальные».
- *AutoHD<TParamS> param();* — Прямой доступ к подсистеме «Параметры».

- *AutoHD<TModSchedul> modSchedul( );* — Прямой доступ к подсистеме «Управление модулями».
- *AutoHD<TSecurity> security( );* — Прямой доступ к подсистеме «Безопасность».
- *string cfgFile( );* — Имя конфигурационного файла системы.
- *XMLNode &cfgRoot( );* — Разобранная структура конфигурационного файла.
- *string workDB( )* — Полное имя рабочей БД.
- *bool sysOptCfg( )* — Признак – «Загружать системные параметры только из конфигурационного файла».
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.
- *static void sighandler(int signal);* — Функция стандартного обработчика сигналов.
- *unsigned long long sysClk( );* — Расчётная частота процессора на котором функционирует система (Гц).
- *void clkCalc( );* – Расчёт частоты процессора на котором работает система. Вызывается периодически для систем с переменной частотой процессора.
- *unsigned long long shrtCnt( );* — Функция замера малых интервалов времени по счетчику тактов процессора. Возвращает значение счетчика тактов процессора.
- *static long TZ( );* — Время системного тика процессора.
- *static long long curTime( );* — Текущее время в микросекундах с начала эпохи (01.01.1970).
- *static bool eventWait(bool &m_mess_r_stat, bool exempl, const string &loc, time_t time = 0);* — Функция ожидания события <exempl> для переменной <m_mess_r_stat> в течении указанного интервала времени <time> для источника <loc>.
- *static string int2str(int val, IntView view = Dec);* — Функция преобразования целого в строку в виде <*view*>.
- *static string ll2str(long long val, IntView view = Dec);* — Функция преобразования длинного целого (64бит) в строку в виде *view*.
- *static string real2str(double val);* — Функция преобразования вещественного в строку.
- *static string fNameFix(const string &fname);* — Преобразование относительных имён файлов к абсолютным.
- *static bool strEmpty(const string &val);* — Проверка строки на пустоту, с учётом пробелов и других незначащих символов в строке.
- *static string strSepParse(const string &path, int level, char sep);* — Разбор строки <path> на составляющие отделённые разделительным символом <sep>.
- *static string pathLev(const string &path, int level, bool encode = true);* — Выделение элементов пути <path> с возможностью их декодирования.
- *static string strCode(const string &in, Code tp, const string &symb = " \t\n");* — Кодирование строки по указанному правилу <tp>.
- *static string strEncode(const string &in, Code tp = Custom);* — Декодирование строки по указанному правилу <tp>.

2.2 Объект сообщений системы (TMess)

Данные:

Типы (уровни) сообщений (enum – TMess::Type):

- *Debug* — отладка;
- *Info* — информация;
- *Notice* — замечание;
- *Warning* — предупреждение;
- *Error* — ошибка;

- *Crit* — критическая ситуация;
- *Allert* — сигнализация;
- *Emerg* — авария.

Структура сообщения (struct – TMess::SRec):

- *time_t time;* — время сообщения;
- *string categ;* — категория сообщения (обычно путь внутри системы);
- *Type level;* — уровень сообщения;
- *string mess;* — сообщение.

Публичные методы:

- *void load( );* — Загрузка.
- *void save( );* — Сохранение.
- *string codeConv(const string &fromCH, const string &toCH, const string &mess);* — Конвертация кодировки сообщения.
- *string codeConvIn(const string &fromCH, const string &mess);* — Конвертация кодировки сообщения во внутреннюю кодировку системы.
- *string codeConvOut(const string &toCH, const string &mess);* — Конвертация кодировки сообщения из внутренней кодировки системы.
- *static const char *I18N(const char *mess, const char *d_name = NULL);* — Получение сообщения на языке системы.
- *static string I18Ns(const string &mess, const char *d_name = NULL);* — Получение сообщения на языке системы.
- *static bool chkPattern(const string &val, const string &patern);* — Проверка принадлежности строки к шаблону. Поддерживаются специальные символы обобщения '*' и '?'.
- *string lang( );* — Язык системы (локализация).
- *string &charset( );* — Системная кодировка.
- *int logDirect( );* — Приемники которым направляются системные сообщения (stdout, stderr, syslog, archive);
- *int messLevel( );* — Уровень, ниже которого сообщения игнорируются.
- *void lang(const string〈);* — Установка языка системы (локализации).
- *void logDirect(int dir);* — Установка приемников которым направляются системные сообщения. Для <dir> используется битовая маска. Где:

  1 – в syslog;

  2 – в stdout;

  4 – в stderr;

  8 – в архив.

- *void messLevel(int level);* — Установка минимального уровня обрабатываемых сообщений.
- *void put(const char *categ, Type level, const char *fmt, ...);* — Сформировать сообщение за текущее время.
- *void get(time_t b_tm, time_t e_tm, vector<TMess::SRec> & recs, const string &category = "", Type level = Debug);* — Запросить сообщения из архива за промежуток времени <*b_tm*> – *e_tm* в соответствии с шаблоном категории <*category*> и минимальным уровнем <*level*>.

2.3 Объект подсистемы (TSubSYS)

Наследует:	*TCntrNode.*
Наследуется:	*TArchiveS, TProtocolS, TBDS, TFunctionS, TSecurity, TModShedul, TTransportS, TUIS, TSpecialS, TControllerS, TParamS.*

Публичные методы:

- *TSubSYS(char *id, char *name, bool modi = false);* — Инициализирующий конструктор. Признак <modi> указывает, что подсистема модульная.
- *string subId( );* — Идентификатор подсистемы.
- *string subName( );* — Локализованное имя подсистемы.
- *bool subModule( );* — Признак модульности подсистемы.
- *int subSecGrp( );* — Идентификатор группы пользователей данной подсистемы.
- *virtual int subVer( );* — Версия подсистемы.
- *virtual void subLoad( );* — Загрузка подсистемы.
- *virtual void subSave( );* — Сохранение подсистемы.
- *virtual void subStart( );* — Запуск подсистемы.
- *virtual void subStop( );* — Останов подсистемы.
- *void modList(vector<string> &list);* — Список <*list*> модулей модульной подсистемы.
- *bool modPresent(const string &name);* — Проверка на наличие указанного модуля <*name*>.
- *void modAdd(TModule *modul);* — Добавление/регистрация модуля <*modul*>.
- *void modDel(const string &name);* — Удаление модуля <*name*>.
- *AutoHD<TModule> modAt(const string &name);* — Подключение к модулю <*name*>.
- *TSYS &owner( );* — Система – владелец подсистемы.

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.

2.4 Объект модуля (TModule)

Наследует:	*TCntrNode.*
Наследуется:	*TProtocol, TTipBD, TTipArchive, TTipTransport, TUI, Tspecial, TTipController.*

Данные:

Структура данных идентифицирующей модуль (struct – TModule::SAt):

- *string id;* — идентификатор модуля;
- *string type;* — тип модуля (подсистема);
- *int t_ver;* — версия типа модуля (подсистемы) для которой модуль разработан.

Структура экспортируемых функций (class – TModule::ExpFunc):

- *string prot;* — прототип функции;
- *string dscr;* — локализованное описание функции;
- *void (TModule::*ptr)( );* — относительный адрес функции (относительно объекта модуля).

Публичные методы:

- *const string &modId( );* — Идентификатор модуля.
- *string modName( );* — Локализованное имя модуля.
- *virtual void modLoad( );* — Загрузка модуля.

- *virtual void modSave( );* — Сохранение модуля.
- *virtual void modStart( );* — Запуск модуля.
- *virtual void modStop( );* — Останов модуля.
- *virtual void modInfo(vector<string> &list);* — Список <*list*> информационных элементов модуля.
- *virtual string modInfo(const string &name);* — Получение содержимого указанного информационного элемента <*name*>.
- *void modFuncList(vector<string> &list);* — Список <*list*> экспортируемых функций модуля.
- *bool modFuncPresent(const string &prot);* — Проверка на наличие указанной функции по её прототипу <*prot*>.
- *ExpFunc &modFunc(const string &prot);* — Получить информацию об экспортируемой функции модуля <*prot*>.
- *void modFunc(const string &prot, void (TModule::**offptr)());* — Получение относительного адреса <*offptr*> экспортируемой функции <*prot*>.
- *const char *I18N(const char *mess);* — Локализации модульного сообщения <*mess*> в соответствии с текущей локалью.
- *string I18Ns(const string &mess);* — Локализации модульного сообщения <*mess*> в соответствии с текущей локалью.
- *TSubSYS &owner();* — Подсистема – владелец модуля.

Защищённые атрибуты:

- *string mId;* — Идентификатор модуля.
- *string mName;* — Имя модуля.
- *string mDescr;* — Описание модуля.
- *string mType;* — Тип модуля.
- *string mVers;* — Версия модуля.
- *string mAutor;* — Автор модуля.
- *string mLicense;* — Лицензия модуля.
- *string mSource;* — Источник/происхождение модуля.

Защищённые методы:

- *void postEnable();* — Вызывается после подключение модуля к динамическому дереву объектов.
- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void modFuncReg(ExpFunc *func);* — Регистрация экспортируемых модулем функций.

3 Подсистема “Базы Данных” (TBDS)

Подсистема «Базы Данных» представлена объектом *TBDS* который содержит модульные объекты типов БД *TTipBD*. Каждый тип базы данных содержит объекты отдельно взятых баз данных данного типа *TBD*. Каждая БД, в свою очередь, содержит объекты своих таблиц *TTable* (рис. 3).

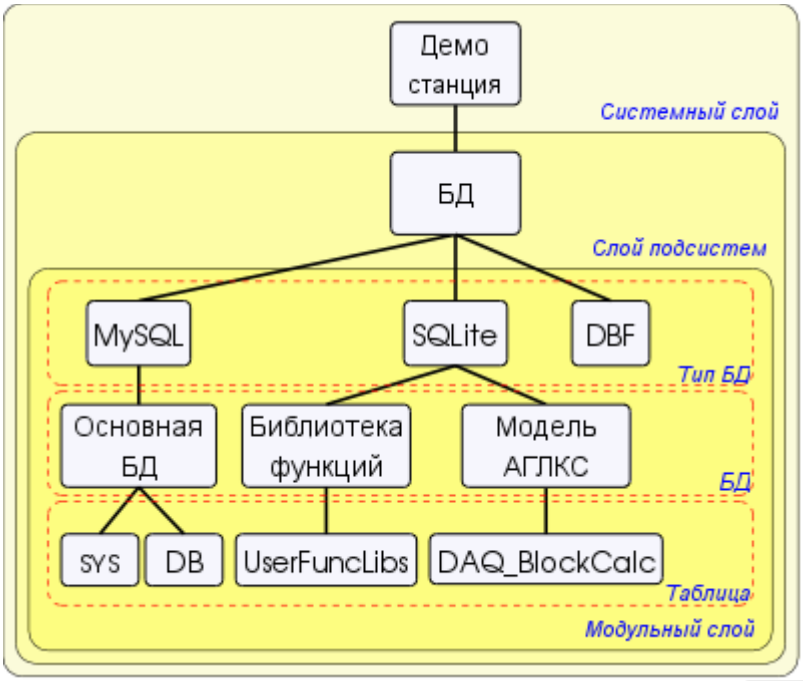

Рис. 3. Слоистая структура подсистемы БД.

Подсистема представляет базовые функции для доступа к типам БД, а также обобщающие функции для манипуляции с базами данных и таблицами. Так, для скрытия источника данных, которым может быть и конфигурационный файл, предоставляются функции абстрактного доступа к источнику данных. А для хранения обще-системных данных предоставляется системная таблица и функции абстрактного доступа к ней. Следовательно, обще-системные данные могут храниться как в конфигурационном файле, так и в таблице БД. Приоритетным источником, в таком случае, является таблица БД.

Являясь модульным объектом, тип БД (TTipBD) содержит доступ к реализации механизма той или иной БД. Доступ производится посредством открытых БД модуля отдельно взятого типа БД. Открываемые/регистрированные БД описываются в таблице открываемых БД или в конфигурационном файле. Существует, так называемая, рабочая БД, которая открывается всегда и указывается в конфигурационном файле. БД поддерживающие SQL-запросы могут предоставлять доступ основанный на прямых SQL-запросах.

В процессе использования, компоненты системы OpenSCADA открывают таблицы (TTable) в доступных БД и работают с ними.

3.1 Объект подсистемы «Базы Данных» (TBDS)

Наследует:	*TSubSYS.*

Публичные методы:

- *int subVer( );* — Версия подсистемы.
- *void subLoad( );* — Загрузка подсистемы.
- *void subSave( );* — Сохранение подсистемы.

- *AutoHD<TTable> open(const string &bdn, bool create = false);* — Открытие таблицы <*bdn*> БД по её полному пути с созданием <*create*> в случае отсутствия.
- *void close(const string &bdn, bool del = false);* — Закрытие таблицы <*bdn*> БД по её полному пути с возможностью удаления после закрытия <*del*>.
- *bool dataSeek(const string &bdn, const string &path, int lev, TConfig &cfg);* — Общее сканирование записей источника данных. В качестве источника выступает конфигурационный файл или БД. В случае отсутствия БД используется конфигурационный файл.
- *bool dataGet(const string &bdn, const string &path, TConfig &cfg);* — Получение записи из источника данных (БД или конфигурационный файл).
- *void dataSet(const string &bdn, const string &path, TConfig &cfg);* — Установить/сохранить запись в источнике данных (БД или конфигурационный файл).
- *void dataDel(const string &bdn, const string &path, TConfig &cfg);* — Удаление записи из источника данных (БД или конфигурационный файл).
- *static string genDBGet(const string &path, const string &oval = "", bool onlyCfg = false);* — Получить обще-системные данные из конфигурационного файла или системной таблицы. Если данные отсутствуют то возвращается значений <*oval*>.
- *static void genDBSet(const string &path, const string &val);* — Установить/сохранить обще-системные данные в конфигурационном файле или системной таблице.
- *string SysBD();* — Полное имя системной таблицы.
- *string openBD();* — Полное имя таблицы с описанием зарегистрированных БД.
- *TElem &openDB_E()* — Структура таблицы зарегистрированных БД.
- *AutoHD<TTipBD> at(const string &iid)* — Обращение к модулю БД (типу БД).
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.

3.2 Модульный объект типов баз данных (TTipBD)

Наследует:	*TModule.*
Наследуется:	Корневыми объектами модулей подсистемы «БД».

Публичные методы:

- *bool fullDeleteDB();* — Признак полного удаления БД.
- *void list(vector<string> &list);* — Список зарегистрированных (открытых) БД.
- *bool openStat(const string &idb);* — Проверка на наличие указанной открытой БД.
- *void open(const string &iid);* — Открытие БД.
- *void close(const string &iid, bool erase = false);* — Закрытие БД. Если установлен признак <*erase*> то БД будет полностью удалена.
- *AutoHD<TBD> at(const string &name);* — Подключение к открытой БД.
- *TBDS &owner();* — Подсистема – владелец модуля.

3.3 Объект базы данных (TBD)

Наследует:	*TCntrNode.*
Наследуется:	Объектами баз данных модулей подсистемы «БД».

Публичные методы:

- *TBD(const string &iid, TElem *cf_el);* — Инициализирующий конструктор.
- *const string &id();* — Идентификатор БД.
- *string name();* — Имя БД.
- *string dscr();* — Описание БД.

- *string addr( );* — Адрес БД. Форма записи отлична для каждого типа БД.
- *bool create( );* — Признак: «Создавать БД».
- *bool enableStat( );* — Состояние БД: «Включена».
- *bool toEnable( );* — Признак БД: «Включать».
- *void name(const string &inm);* — Установка имени БД.
- *void dscr(const string &idscr);* — Установка описания БД.
- *void addr(const string &iaddr);* — Установка адреса БД.
- *void create(bool ivl);* — Установка признака: «Создавать БД».
- *void toEnable(bool ivl);* — Установка признака: «Включать».
- *virtual void enable( );* — Включение БД.
- *virtual void disable( );* — Отключение БД.
- *virtual void load( );* — Загрузка БД.
- *virtual void save( );* — Сохранение БД.
- *void list(vector<string> &list);* — Список открытых таблиц.
- *bool openStat(const string &table);* --Признак указывающий на то, что запрошенная таблица открыта.
- *void open(const string &table, bool create);* — Открытие таблицы. Если установлен признак <create>, то в случае отсутствия таблица будет создана.
- *void close(const string &table, bool del = false);* — Закрытие таблицы. Если установлен признак <del>, то таблица будет полностью удалена.
- *AutoHD<TTable> at(const string &name);* — Подключение к таблице.
- *virtual void sqlReq(const string &req, vector< vector<string> > *tbl = NULL);* — Отправка SQL-запроса <*req*> на БД и получение результата в виде таблицы <*tbl*>.
- *TTipBD &owner( );* — Тип базы данных – владелец данной БД.

Защищённые методы:

- *virtual TTable *openTable(const string &table, bool create);* — Модульный метод открытия таблицы.
- *void preDisable(int flag);* — Вызывается перед исключением узла из динамического дерева.
- *void postDisable(int flag);* — Вызывается после исключения узла из динамического дерева.
- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обработка команд интерфейса управления системой.

3.4 Объект таблицы (TTable)

Наследует:	*TCntrNode.*
Наследуется:	Объектами таблиц модулей подсистемы «БД».

Публичные методы:

- *TTable(const string &name);* — Инициализирующий конструктор.
- *string &name( );* — Имя таблицы.
- *virtual bool fieldSeek(int row, TConfig &cfg);* — Сканирование записей таблицы.
- *virtual void fieldGet(TConfig &cfg);* — Запрос указанной записи. Запрашиваемая запись определяется значениями ключевых ячеек исходной записи <*cfg*>.
- *virtual void fieldSet(TConfig &cfg);* — Установка значений указанной записи. В случае отсутствия запись будет создана.
- *virtual void fieldDel(TConfig &cfg);* — Удаление указанной записи.
- *TBD &owner( );* — БД – владелец данной таблицы.

4 Подсистема “Сбор данных” (TDAQS)

Подсистема “Сбор данных” представлена объектом *TDAQS*, который содержит модульные объекты типов источников данных *TTipDAQ*. Объект типов источников данных содержит объекты контроллеров TController и объекты типов параметров *TTipParam*. Объекты типов параметров предоставляются модулем контроллера и содержат структуру БД отдельных типов параметров (аналоговые, дискретные ...). Объекты контроллеров содержат объекты параметров *TParamContr*. Каждый параметр ассоциируется с одним из типов параметров. Для хранения атрибутов, параметр наследуется от объекта значений *TValue*, который и содержит значения атрибутов *TVal* (рис. 4).

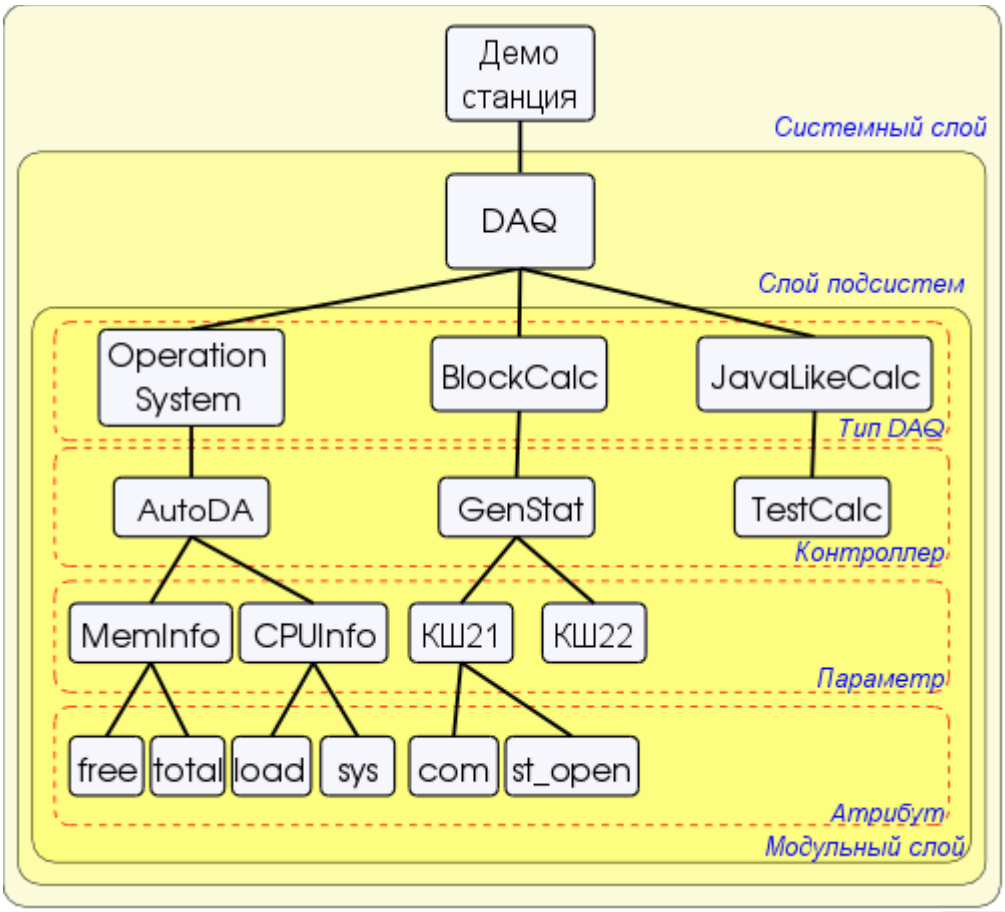

Рис. 4. Слоистая структура подсистемы сбора данных.

Подсистема содержит типы источников данных. Источником может выступать практически любая сущность предоставляющая какие либо данные. Тип источника может делится на отдельные источники (контроллеры) в пределах конкретного типа. Например, если взять данные из операционной системы (ОС), то под отдельным источником можно понимать операционную систему отдельного ПК.

Источник данных (контроллер) делится, или содержит, параметры. Под параметром подразумевается какая-то часть источника данных. В случае с ОС это будет, например: расход оперативной памяти, частота процессора и много других составных частей.

Параметр, в свою очередь, содержит атрибуты, которые и предоставляют данные. Кроме основных данных атрибутами могут предоставляться и сопутствующие или детализирующие данные. В случае тойже ОС и расхода памяти, атрибутами может предоставляться не только занятая память, а также и сколько её всего, сколько в свопе и т.д.

4.1 Объект подсистемы «Сбор данных» (TDAQS)

Наследует:	*TSubSYS, TElem.*

Публичные методы:

- *int subVer( );* — Версия подсистемы.
- *void subLoad( );* — Загрузка подсистемы.
- *void subSave( );* — Сохранение подсистемы.
- *virtual void subStart( );* — Запуск подсистемы.
- *virtual void subStop( );* — Останов подсистемы.
- *AutoHD<TTipDAQ> at(const string &name);* — Подключение к типу источника данных.
- *TElem &errE( );* — Структура атрибута(ов) ошибок параметров.

4.2 Модульный объект типа контроллера (TTipDAQ)

Наследует:	*TModule, TElem.*
Наследуется:	Корневыми объектами модулей подсистемы «Сбор данных».

Публичные методы:

- *void modStart( );* — Запуск подсистемы.
- *void modStop( );* — Останов подсистемы.
- *void list(vector<string> &list);* — Список контроллеров.
- *bool present(const string &name);* — Проверка на наличие указанного контроллера.
- *void add(const string &name, const string &daq_db = "*.*");* — Добавить контроллер.
- *void del(const string &name);* --Удалить контроллер.
- *AutoHD<TController> at(const string &name, const string &who = "");* — Подключиться к контроллеру.
- *unsigned tpPrmToId(const string &name_t);* — Получение индекса типа параметров по имени.
- *int tpParmAdd(const char *id, const char *n_db, const char *name);* — Добавление/регистрация типа параметров.
- *unsigned tpPrmSize( );* — Количество типов параметров.
- *TTipParam &tpPrmAt(unsigned id);* — Получить объект типа параметров.

Защищённые методы:

- *virtual TController *ContrAttach(const string &name, const string &daq_db);* — Подключение контроллера. Обязательно переопределяется в потомке модуля.
- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса контроля системы.

4.3 Объект контроллера (TController)

Наследует:	*TModule, TConfig.*
Наследуется:	Объектами контроллеров модулей подсистемы «Сбор данных».

Публичные методы:

- *TController(const string &name_c, const string &daq_db, TElem *cfgelem);* — Инициализирующий конструктор контроллера.
- *const string &id( );* — Идентификатор контроллера.
- *string name( );* — Имя контроллера.

- *string descr( );* — Описание контроллера.
- *void name(const string &nm);* — Установить имя контроллера.
- *void descr(const string &dscr);* — Установить описание контроллера.
- *bool toEnable( );* — Признак «Включать контроллер».
- *bool toStart( );* — Признак «Запускать контроллер».
- *bool enableStat( );* — Состояние «Включен».
- *bool startStat( );* — Состояние «Запущен».
- *virtual void load( );* — Загрузка контроллера.
- *virtual void save( );* — Сохранение контроллера.
- *virtual void start( );* — Запуск контроллера.
- *virtual void stop( );* — Останов контроллера.
- *void enable( );* — Включение контроллера.
- *void disable( );* — Отключение контроллера.
- *void list(vector<string> &list);* — Список параметров в контроллере.
- *bool present(const string &name);* — Проверка на наличие параметра <*name*>.
- *void add(const string &name, unsigned type);* — Добавление параметра <*name*> типа <*type*>.
- *void del(const string &name, bool full = false);* — Удаление параметра <*name*>. Если указано поле <*full*> то контроллер будет удалён полностью.
- *AutoHD<TParamContr> at(const string &name, const string &who = "th_contr");* — Подключение к параметру <*name*>.
- *string genBD( );* — Основная БД контроллера.
- *string BD( );* — Полное имя таблицы содержащей конфигурацию контроллера.
- *TTipController &owner( );* — Тип источника данных (модуль) – владелец данным контроллером.

Защищённые атрибуты:

- *bool en_st;* — Признак «Включено».
- *bool run_st;* — Признак «Запущено».

Защищённые методы:

- *virtual void enable_( );* — Включение контроллера.
- *virtual void disable_( );* — Отключение контроллера.
- *virtual TParamContr *ParamAttach(const string &name, int type);* — Модульный метод создания/открытия нового параметра.
- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void preDisable(int flag);* — Вызывается перед отключением от динамического дерева.
- *void postDisable(int flag);* — Вызывается после отключением от динамического дерева.

4.4 Объект типа параметров (TTipParam)

Наследует:	*TElem.*

Публичные методы:

- *TTipParam(const char *id, const char *name, const char *db);* — Инициализирующий конструктор.
- *string name( );* — Имя типа параметра.
- *string lName( );* — Описание типа параметра.
- *string BD( );* — БД типа параметра.

4.5 Объект параметра физического уровня (TParamContr)

Наследует:	*TConfig, TValue.*
Наследуется:	Объектами параметров модулей подсистемы «Сбор данных».

Публичные методы:

- *TParamContr(const string &name, TTipParam *tpprm);* — Инициализирующий конструктор.
- *const string &id();* — Идентификатор параметра (шифр).
- *string name();* — Имя параметра.
- *string descr();* — Описание параметра.
- *void name(const string &inm);* — Установка имени параметра.
- *void descr(const string &idsc);* — Установка описания параметра.
- *TTipParam &type();* — Тип параметра.
- *bool toEnable();* — Признак «Включать параметр».
- *bool enableStat()* — Состояние «Включен».
- *virtual void enable( );* — Включить параметр.
- *virtual void disable( );* — Отключить параметр.
- *void load( );* — Загрузить параметр.
- *void save( );* — Сохранить параметр.
- *bool operator==(TParamContr & PrmCntr);* — Сравнение параметров.
- *TParamContr &operator=(TParamContr & PrmCntr);* — Копирование параметра.
- *TController &owner();* — Контроллер – владелец параметра.

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void postEnable();* — Вызывается после включения узла в динамическое дерево.
- *void preDisable(int flag);* — Вызывается перед отключением узла от динамического дерева.
- *void postDisable(int flag);* — Вызывается после отключением узла от динамического дерева.

4.6 Объект значения (TValue)

Наследует:	*TValElem, TCntrNode.*
Наследуется:	*TParamContr.*

Публичные методы:

- *void vlList(vector<string> &list);* — Получение списка атрибутов.
- *bool vlPresent(const string &name);* — Проверка на наличия указанного атрибута.
- *AutoHD<TVal> vlAt(const string &name);* — Подключение к атрибуту.

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обработка команд интерфейса управления системой.
- *TConfig *vlCfg()* — Получение связанного объекта конфигурации. Если возвращается NULL то отсутствует связанный объект конфигурации.
- *void vlCfg(TConfig *cfg);* — Установка связанного объекта конфигурации <*cfg*>.
- *bool vlElemPresent(TElem *Val El);* — Проверка на включенность элемента атрибутов <*ValEl*>.

- *void vlElemAtt(TElem *Val El);* — Подключение структуры данных <*ValEl*>.
- *void vlElemDet(TElem *Val El);* — Отключение структуры данных <*ValEl*>.
- *TElem &vlElem(const string &name);* — Получить структуру данных по её имени <*name*>.
- *virtual void vlSet(TVal &val);* — Упреждающая функция установки значения. Используется для прямой записи.
- *virtual void vlGet(TVal &val);* — Упреждающая функция получения значения. Используется для прямого чтения.

4.7 Объект атрибута (TVal).

Наследует:	*TCntrNode.*

Данные:

Дополнительные флаги к объекту TFld (define):

- *FLD_DRD* — флаг прямого чтения значения;
- *FLD_DWR* — флаг прямой записи значения.

Значения ошибки для различных типов данных:

- *EVAL_BOOL* — Значение ошибки логического (2);
- *EVAL_INT* — Значение ошибки целого (-2147483647);
- *EVAL_REAL* — Значение ошибки вещественного (-3.3E308);
- *EVAL_STR* — Значение ошибки строкового ("<EVAL>").

Публичные методы:

- *TVal(TFld &fld, TValue *owner);* — Инициализация как хранилище динамических данных.
- *TVal(TCfg &cfg, TValue *owner);* — Инициализация как отражение статических данных (БД).
- *const string &name();* — Имя атрибута.
- *TFld &fld();* — Описатель структуры атрибута.
- *string getSEL(long long *tm = NULL, bool sys = false);* — Запрос значения выборочного типа на указанное время <*tm*>. Если NULL то возвратится последнее значение.
- *string getS(long long *tm = NULL, bool sys = false);* — Запрос значения строкового типа на указанное время <*tm*>. Если NULL то возвратится последнее значение.
- *double getR(long long *tm = NULL, bool sys = false);* — Запрос значения вещественного типа на указанное время <*tm*>. Если NULL то возвратится последнее значение.
- *int getI(long long *tm = NULL, bool sys = false);* — Запрос значения целого типа на указанное время <*tm*>. Если NULL то возвратится последнее значение.
- *char getB(long long *tm = NULL, bool sys = false);* — Запрос значения логического типа на указанное время <*tm*>. Если NULL то возвратится последнее значение.
- *void setSEL(const string &value, long long tm = 0, bool sys = false);* — Установка значения выборочного типа <*value*>.
- *void setS(const string &value, long long tm = 0, bool sys = false);* — Установка значения строкового типа <*value*>.
- *void setR(double value, long long tm = 0, bool sys = false);* — Установка значения вещественного типа <*value*>.
- *void setI(int value, long long tm = 0, bool sys = false);* — Установка значения целого типа <*value*>.

- *void setB(char value, long long tm = 0, bool sys = false);* — Установка значения логического типа <*value*>.
- *AutoHD<TVArchive> &arch();* — Получение ассоциированного со значением архива.
- *void arch(const AutoHD<TVArchive> &vl);* — Установка ассоциированного со значением архива.

Защищённые методы:

- *void vlSet( );* — Упреждающая функция установки значения. Используется для прямой записи.
- *void vlGet( );* — Упреждающая функция получения значения. Используется для прямого чтения.

5 Подсистема “Архивы” (TArchiveS)

Подсистема «Архивы» представлена объектом TArchiveS который содержит, на уровне подсистемы, модульные объекты типов архиваторов TTipArchivator. Каждый объект типа архиватора содержит объекты архиваторов сообщений TMArchivator и архиваторов значений TVArchivator. Кроме этого объект подсистемы архивы содержит методы архива сообщений и объекты архивов значений TVArchive. Объект архива значений TVArchive содержит буфер значений путём наследования объекта буфера TValBuf. Для связи архива значений с архиваторами предназначен объект элемента значения TVArchEl. Этот объект содержится в архиваторе и на него ссылается архив. Структура подсистемы «Архивы» представлена на рис. 5.

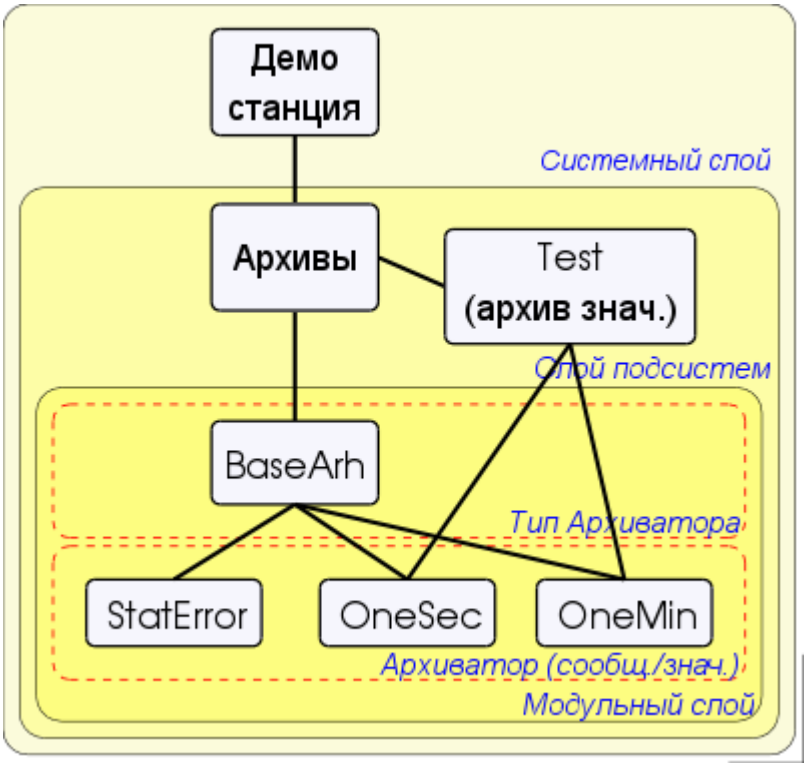

Рис. 5. Слоистая структура подсистемы архивов.

Подсистема «Архивы». Содержит механизмы архивирования сообщений и значений. Непосредственно содержит архив сообщений вместе с его буфером. Содержит методы доступа к архивам значений и архиваторам значений и сообщений. Кроме этого выполняет задачу активного сбора данных из источников значений для архивов значений, а также архивирование архива сообщений по архиваторам.

Архив значений (TVArchive) содержит буфер (TValBuf) для промежуточного накопления значений перед архивированием. Связывается с источником значений в лице параметров системы OpenSCADA в активном или пассивном режиме, а также с другими источниками в пассивном режиме. Для архивирования на физические хранилища связывается с архиваторами значений различных типов.

Объект буфера TValBuf содержит массив значений основных типов системы OpenSCADA: строка, целое, вещественное и логичное. Поддерживается хранение значений в режимах жесткой, мягкой сетки и режиме свободного доступа. Предусмотрен, также, режим времени высокого разрешения (микросекунды). Используется как для непосредсвенного хранения больших массивов значений, так и для обмена с большими масивами методом покадрового доступа.

Корневой объект модуля подсистемы «Архивы» (TTipArchivator) содержит информацию

о конкретно взятом типе модуля. В рамках отдельных модулей может реализовывать собственные общемодульные функции. В общем, для модулей этого типа, содержит методы доступа к хранилищам значений и сообщений.

Объект архиватора сообщений (TMArchivator) содержит конкретную реализацию хранилища сообщений. В общем, для архиваторов сообщений, предоставляется интерфейс для доступа к реализации механизма архивирования.

Объект архиватора значений (TVArchivator) содержит конкретную реализацию хранилища значений. В общем, для архиваторов значений, предоставляется интерфейс для доступа к реализации механизма архивирования и назначение архивов значений на обслуживание архиватором.

Объект элемента архива TVArchEl связывает объекты архивов с архиваторами. Используется для доступа к архиваторам из архива, а также к архивам из архиватора, т.е. для перекрёстных вызовов.

5.1 Объект подсистемы «Архивы» (TArchiveS)

Наследует:	*SubSYS.*

Публичные методы:

- *int subVer( );* — Версия подсистемы.
- *int messPeriod( );* — Период архивирование сообщений из буфера (секунд).
- *int valPeriod( );* — Период сбора значений для активных архиваторов (милисекунд).
- *void messPeriod(int ivl);* — Установка периода архивирования сообщений из буфера (секунд).
- *void valPeriod(int ivl);* — Установка периода сбора значений для активных архиваторов (милисекунд).
- *void subLoad( );* — Загрузка подсистемы.
- *void subSave( );* — Сохранение подсистемы.
- *void subStart( );* — Запуск подсистемы.
- *void subStop( );* — Останов подсистемы.
- *void valList(vector<string> &list);* — Список архивов значений в подсистеме.
- *bool valPresent(const string &iid);* — Проверка на наличие архива значений <*iid*>.
- *void valAdd(const string &iid, const string &idb = "*.*");* — Добавление нового архива значений <*iid*>.
- *void valDel(const string &iid, bool db = false);* — Удаление архива значений <*iid*>.
- *AutoHD<TVArchive> valAt(const string &iid);* — Подключение/обращение к архиву значений <*iid*>.
- *void setActValArch(const string &id, bool val);* — Установка архива <*id*> в активное состояние <*val*>. Активный архив будет обеспечиваться периодическим потоком значений (определяется периодичностью архива) подсистемой.
- *AutoHD<TTipArchivator> at(const string &name);* — Подключение/обращение к типу архиватора (модулю) <*name*>.
- *void messPut(time_t tm, const string &categ, TMess::Type level, const string &mess);* — Помещение значения <*mess*> с уровнем <*level*> категории <*categ*> и время <*tm*> в буфер, а затем в архив сообщений.
- *void messPut(const vector<TMess::SRec> &recs);* — Помещение группы значений <*recs*> в буфер, а затем в архив сообщений.
- *void messGet(time_t b_tm, time_t e_tm, vector<TMess::SRec> & recs, const string &category = "", TMess::Type level = TMess::Debug, const string &arch = "");* — Запрос значений <*reqs*> за указанный период времени <*b_tm*>, *e_tm* для указанной категории (по шаблону) <*category*> и уровня <*level*> из архиватора <*arch*>.

- *TElem &messE( );* — Структура БД архиваторов сообщений.
- *TElem &valE( );* — Структура БД архиваторов значений.
- *TElem &aValE( );* — Структура БД архивов значений.

5.2 Объект архива значений (TVArchive)

Наследует:	*TCntrNode, TValBuf, TConfig*

Данные:

Режим сбора данных/источник (struct – TVArchive::SrcMode):

- *Passive* — пассивный режим сбора данных, источник самостаятельно помещает данные в архив;
- *PassiveAttr* — пассивный режим сбора данных у атрибута параметра, атрибут параметра самостаятельно помещает данные в архив;
- *ActiveAttr* — активный режим сбора данных у атрибута параметра, атрибут параметра периодически опрашивается подсистемой «Архивы»;

Публичные методы:

- *TVArchive(const string &iid, const string &idb, TElem *cf_el);* — Инициализирующий конструктор архива. Где <*iid*> — идентификатор архива, <*idb*> — БД для хранения и <*cf_el*> — структура БД архивов значений.
- *const string &id( );* — Идентификатор архива.
- *string name( );* — Имя архива.
- *string dscr( );* — Описание архива.
- *SrcMode srcMode( );* — Режим связывания с источником данных.
- *bool toStart( );* — Признак: «Запускать архив при включении».
- *bool startStat( );* — Состояние: «Архив запущен».
- *long long end(const string &arch = BUF_ARCH_NM);* — Время окончания архива в целом (arch="") или указанного архиватора, буфера (arch="<bufer>").
- *long long begin(const string &arch = BUF_ARCH_NM);* — Время начала архива в целом (arch="") или указанного архиватора, буфера (arch="<bufer>").
- *TFld::Type valType( );* — Тип архивируемого значения.
- *bool hardGrid( );* — Использование жесткой сетки в буфере архива.
- *bool highResTm( );* — Использование высокого разрешения времени в буфере архива (микросекунды).
- *int size( );* — Размер буфера архива (еденицы).
- *long long period( );* — Периодичность буфера архива (микросекунд).
- *void name(const string &inm);* — Установка имени архива.
- *void dscr(const string &idscr);* — Установка описания архива.
- *void srcMode(SrcMode vl, const string &isrc = "");* — Установка режима связывания с источником данных.
- *void toStart(bool vl);* — Установка признака: «Запускать архив при включении».
- *void valType(TFld::Type vl);* — Установка типа архивируемого значения.
- *void hardGrid(bool vl);* — Установка использования жесткой сетки в буфере архива.
- *void highResTm(bool vl);* — Установка использования высокого разрешения времени в буфере архива (микросекунды).
- *void size(int vl);* — Установка размера буфера архива (едениц).
- *void period(long long vl);* — Установка периодичности буфера архива.
- *void load( );* — Загрузка архива.
- *void save( );* — Сохранение архива.
- *void start( );* — Запуск архива.

- *void stop(bool full_del = false);* — Останов архива, с полным удалением <*full_del*>.
- *void getVal(TValBuf &buf, long long beg = 0, long long end = 0, const string &arch = "");* — Запрос кадра значений <*buf*> за время от <*beg*> до <*end*> из указанного архиватора <*arch*>, буфера (arch="<bufer>") или всех архиваторов по мере падения качества (arch="").
- *string getS(long long *tm = NULL, bool up_ord = false, const string &arch = "");* — Запрос одного значения строкового типа за время <*tm*> и признаком притягивания к верху <*up_ord*> из указанного архиватора <*arch*>, буфера (arch="<bufer>") или всех архиваторов по мере падения качества (arch="").
- *double getR(long long *tm = NULL, bool up_ord = false, const string &arch = "");* — Запрос одного значения вещественного типа за время <*tm*> и признаком притягивания к верху <*up_ord*> из указанного архиватора <*arch*>, буфера (arch="<bufer>") или всех архиваторов по мере падения качества (arch="").
- *int getI(long long *tm = NULL, bool up_ord = false, const string &arch = "");* — Запрос одного значения целого типа за время <*tm*> и признаком притягивания к верху <*up_ord*> из указанного архиватора <*arch*>, буфера (arch="<bufer>") или всех архиваторов по мере падения качества (arch="").
- *char getB(long long *tm = NULL, bool up_ord = false, const string &arch = "");* — Запрос одного значения логического типа за время <*tm*> и признаком притягивания к верху <*up_ord*> из указанного архиватора <*arch*>, буфера (arch="<bufer>") или всех архиваторов по мере падения качества (arch="").
- *void setVal(TValBuf &buf, long long beg, long long end, const string &arch);* — Загрузка кадра значений <*buf*> за время от <*beg*> до <*end*> в указанный архиватор <*arch*>, буфер (arch="<bufer>") или все архиваторы (arch="").
- *void getActiveData();* — Опросить источник данных. Используется подсистемой для периодического сборе данных активными архивами.
- *void archivatorList(vector<string> &ls);* — Список архиваторов которыми обслуживается архив.
- *bool archivatorPresent(const string &arch);* — Проверка архиватора на обслуживание данного архива.
- *void archivatorAttach(const string &arch);* — Подключение данного архива на обслуживание указанным архиватором.
- *void archivatorDetach(const string &arch, bool full = false);* — Отключение данного архива от обслуживание указанным архиватором.
- *void archivatorSort();* — Сортировка обслуживающих архиваторов в порядке ухудшения качества.
- *string makeTrendImg(long long beg, long long end, const string &arch, int hsz = 650, int vsz = 230);* — Формирование изображения (pdf) тренда за указанный промежуток времени <*beg*>, <*end*> и указанного архиватора <*arch*>.
- *string BD();* — Полное имя таблицы в БД содержащей архив.
- *TArchiveS &owner();* — Подсистема «Архивы» – владелец архива значений.

5.3 Объект буфера значений (TValBuf)

Наследуется:	*TVArchive*

Публичные методы:

- *TValBuf( );* — Инициализатор буфера с установками по умолчанию.
- *TValBuf(TFld::Type vtp, int isz, long long ipr, bool ihgrd = false, bool ihres = false);* — Инициализатор буфера с указанными параметрами.
- *void clear();* — Очистка буфера.
- *TFld::Type valType();* — Тип значения хранимого буфером.

- *bool hardGrid( );* — Работа буфера в режиме жесткой сетки.
- *bool highResTm( );* — Работа буфера в режиме времени высокого разрешения (микросекунды).
- *int size( );* — Максимальный размер буфера (едениц).
- *int realSize( );* — Реальный размер буфера (едениц).
- *long long period( );* — Периодичность значений в буфере (микросекунд). Если периодичность нулевая то буфер функционирует в режиме свободного доступа.
- *long long begin( );* — Время начала буфера (микросекунд).
- *long long end( );* — Время окончания буфера (микросекунд).
- *void valType(TFld::Type vl);* — Установка типа значения хранимого буфером.
- *void hardGrid(bool vl);* — Установка режима жесткой сетки.
- *void highResTm(bool vl);* — Установка режима времени высокого разрешения (микросекунды).
- *void size(int vl);* — Установка размера буфера (едениц).
- *void period(long long vl);* — Установка периодичности значений в буфере (микросекунд).
- *bool vOK(long long ibeg, long long iend);* — Проверка наличия значений в буфере за указанный промежуток времени от <*ibeg*> до <*iend*>.
- *virtual void getVal(TValBuf &buf, long long beg = 0, long long end = 0);* — Запрос кадра значений <*buf*> за время от <*beg*> до <*end*>.
- *virtual string getS(long long *tm = NULL, bool up_ord = false);* — Запрос значения строкового типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual double getR(long long *tm = NULL, bool up_ord = false);* — Запрос значения вещественного типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual int getI(long long *tm = NULL, bool up_ord = false);* — Запрос значения целого типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual char getB(long long *tm = NULL, bool up_ord = false);* — Запрос значения логического типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual void setVal(TValBuf &buf, long long beg = 0, long long end = 0);* — Установка кадра значений из <*buf*> за время от <*beg*> до <*end*>.
- *virtual void setS(const string &value, long long tm = 0);* — Установка значения строкового типа с временем <*tm*>.
- *virtual void setR(double value, long long tm = 0);* — Установка значения вещественного типа с временем <*tm*>.
- *virtual void setI(int value, long long tm = 0);* — Установка значения целого типа с временем <*tm*>.
- *virtual void setB(char value, long long tm = 0);* — Установка значения логического типа с временем <*tm*>.

Защищённые методы:

- *void makeBuf(TFld::Type v_tp, int isz, long long ipr, bool hd_grd, bool hg_res);* — Пересоздание буфера для указанных параметров.

5.4 Модульный объект типа архиватора (TTipArchivator)

Наследует:	*TModule.*
Наследуется:	Корневыми объектами модулей подсистемы «Архивы».

Публичные методы:

- *void messList(vector<string> &list);* — Список архиваторов сообщений.
- *bool messPresent(const string &iid);* — Проверка на наличие указанного архиватора

сообщений.

- *void messAdd(const string &iid, const string &idb = "*.*");* — Добавление архиватора сообщений.
- *void messDel(const string &iid, bool full = false);* — Удаление архиватора сообщений.
- *AutoHD<TMArchivator> messAt(const string &iid);* — Подключение к архиватору сообщений.
- *void valList(vector<string> &list);* — Список архиваторов значений.
- *bool valPresent(const string &iid);* — Проверка на наличие указанного архиватора значений.
- *void valAdd(const string &iid, const string &idb = "*.*");* — Добавление архиватора значений.
- *void valDel(const string &iid, bool full = false);* — Удаление архиватора значений.
- *AutoHD<TVArchivator> valAt(const string &iid);* — Подключение к архиватору значений.
- *TArchiveS &owner();* — Подсистема «Архивы» – владелец типа архиватора.

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *virtual TMArchivator *AMess(const string &iid, const string &idb);* — Модульный метод создание архиватора сообщений.
- *virtual TVArchivator *AVal(const string &iid, const string &idb);* — Модульный метод создание архиватора значений.

5.5 Объект архиватора сообщений (TMArchivator)

Наследует:	*TCntrNode, TConfig*
Наследуется:	Объектами архиваторов сообщений модулей подсистемы «Архивы».

Публичные методы:

- *TMArchivator(const string &iid, const string &idb, TElem *cf_el);* — Инициализирующий конструктор архиватора сообщений с идентификатором <*iid*>, для хранения на БД <*idb*> со структурой <*cf_el*>.
- *const string &id();* — Идентификатор архиватора.
- *string workId();* — Рабочий идентификатор, включает имя модуля.
- *string name();* — Имя архиватора.
- *string dscr();* — Описание архиватора.
- *bool toStart();* — Признак «Запускать архиватор».
- *bool startStat();* — Состояние архиватора «Запущен».
- *void name(const string &vl);* — Установка имени архиватора.
- *void dscr(const string &vl);* — Установка описание архиватора.
- *void toStart(bool vl);* — Установка признака «Запускать архиватор».
- *virtual void load( );* — Загрузка архиватора.
- *virtual void save( );* — Сохранение архиватора.
- *virtual void start( );* — Запуск архиватора.
- *virtual void stop( );* — Останов архиватора.
- *string &addr();* — Адрес хранилища архиватора.
- *int &level();* — Уровень сообщений обслуживаемых архиватором.
- *void categ(vector<string> &list);* — Категории (шаблоны) сообщений обслуживаемых архиватором.
- *virtual void put(vector<TMess::SRec> &mess);* — Поместить группу сообщений

в архив сообщений данного архиватора.

- *virtual void get(time_t b_tm, time_t e_tm, vector<TMess::SRec> &mess, const string &category = "", char level = 0);* — Получить сообщения из архива данного архиватора для указанных параметров фильтра.
- *string BD( );* — Полное имя таблицы БД хранящей архиватор сообщений.
- *TTipArchivator &owner( );* — Тип архиватора – владелец архиватором сообщений.

Защищённые атрибуты:

- *bool run_st;* — Признак «Запущен».

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void postEnable( );* — Вызывается после подключения к динамическому дереву.
- *void postDisable(int flag);* — Вызывается после отключением от динамического дерева.
- *bool chkMessOK(const string &icateg, TMess::Type ilvl);* — Проверка сообщения на соответствие условиям фильтра.

5.6 Объект архиватора значений (TVArchivator)

Наследует:	*TCntrNode, TConfig*
Наследуется:	Объектами архиваторов значений модулей подсистемы «Архивы».

Публичные методы:

- *TVArchivator(const string &iid, const string &idb, TElem *cf_el);* — Инициализирующий конструктор архиватора значений с идентификатором <*iid*>, для хранения на БД <*idb*> со структурой <*cf_el*>.
- *const string &id( );* — Идентификатор архиватора.
- *string workId( );* — Рабочий идентификатор, включает имя модуля.
- *string name( );* — Имя архиватора.
- *string dscr( );* — Описание архиватора.
- *string addr( );* — Адрес хранилища архиватора.
- *double valPeriod( );* — Периодичность значений архиватора (микросекунд).
- *int archPeriod( );* — Периодичность архивирования значений архиватором. Время через которое архиватор производит архивирование кадра значений из буфера архива.
- *bool toStart( );* — Признак «Запускать архиватор».
- *bool startStat( );* — Состояние архиватора «Запущен».
- *void name(const string &inm);* — Установка имени архиватора.
- *void dscr(const string &idscr);* — Установка описания архиватора.
- *virtual void valPeriod(double iper);* — Установка периодичности значений архиватора (микросекунд).
- *virtual void archPeriod(int iper);* — Установка периодичности архивирования значений архиватором. Время через которое архиватор производит архивирование кадра значений из буфера архива.
- *virtual void load( );* — Загрузка архиватора.
- *virtual void save( );* — Сохранение архиватора.
- *virtual void start( );* — Запуск архиватора.
- *virtual void stop(bool full_del = false);* — Останов архиватора с возможностью полного удаления <*full_del*>.
- *void archiveList(vector<string> &ls);* — Список архивов обслуживаемых

архиватором.

- *bool archivePresent(const string &iid);* — Проверка на обслуживаемость указанного архива архиватором.
- *string BD( );* — Полное имя таблицы БД хранящей архиватор значений.
- *TTipArchivator &owner( );* — Тип архиватора – владелец архиватором значений.

Защищённые методы:

- *TVArchEl *archivePlace(TVArchive &item);* — Включить архив <*item*> в обработку архиватором.
- *void archiveRemove(const string &iid, bool full = false);* — Удалить архив <*item*> из обработки архиватором, с возможностью полного удаления <*full*>.
- *virtual TVArchEl *getArchEl(TVArchive &arch);* — Получение объекта элемента архива для указанного архива.
- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void postEnable( );* — Вызывается после подключения к динамическому дереву.
- *void preDisable(int flag);* — Вызывается перед отключением от динамического дерева.
- *void postDisable(int flag);* — Вызывается после отключения от динамического дерева.

Защищённые атрибуты:

- *int a_res* — Идентификатор ресурса процесса архивирования.
- *bool run_st* — Признак «Архив запущен».
- *vector<TVArchEl *> arch_el;* — Массив элементов архивов.

5.7 Объект элемента архива в архиваторе (TVArchEl)

Наследуется:	Объектами архиваторов значений модулей подсистемы «Архивы».

Публичные методы:

- *TVArchEl(TVArchive &iachive, TVArchivator &iarchivator);* — Инициализирующий конструктор для связи архива <*iachive*> с архиватором <*iarchivator*>.
- *virtual void fullErase( );* — Полное удаление элемента.
- *virtual long long end( );* — Время конца данных (микросекунды).
- *virtual long long begin( );* — Время начала данных (микросекунды).
- *long long lastGet( );* — Время последнего сброса данных из буфера в хранилище.
- *virtual void getVal(TValBuf &buf, long long beg = 0, long long end = 0);* — Запрос кадра значений <*buf*> за время от <*beg*> до <*end*>.
- *virtual string getS(long long *tm, bool up_ord);* — Запрос значения строкового типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual double getR(long long *tm, bool up_ord);* — Запрос значения вещественного типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual int getI(long long *tm, bool up_ord);* — Запрос значения целого типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual char getB(long long *tm, bool up_ord);* — Запрос значения логического типа за время <*tm*> и признаком притягивания к верху <*up_ord*>.
- *virtual void setVal(TValBuf &buf, long long beg = 0, long long end = 0);* — Установка кадра значений из <*buf*> за время от <*beg*> до <*end*>.
- *TVArchive &archive( );* — Архив элемента.
- *TVArchivator &archivator( );* — Архиватор элемента.

6 Подсистема «Транспорты» (TTransportS)

Подсистема «Транспорты» представлена объектом TTransportS который содержит, на уровне подсистемы, модульные объекты типов транспортов TTipTransport. Каждый тип транспорта содержит объекты входящих TTransportIn и исходящих TTransportOut транспортов. Общая структура подсистемы приведена на рис. 6.

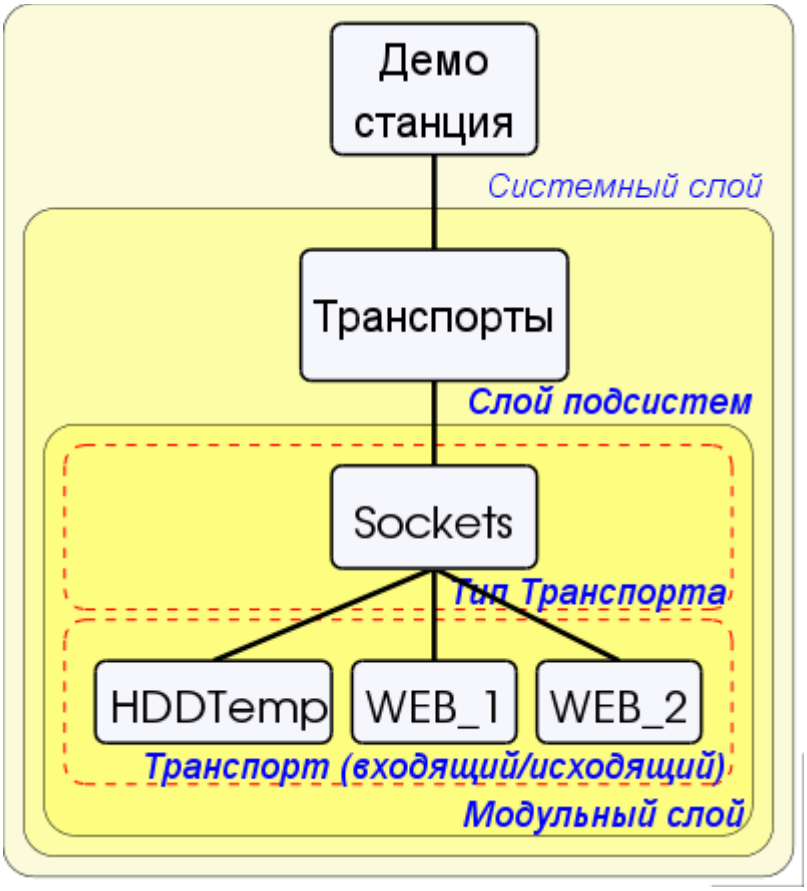

Рис. 6. Слоистая структура подсистемы транспортов.

Корневой объект модуля подсистемы «Транспорты» содержит информацию о конкретно взятом типе модуля. В рамках отдельно взятого модуля может реализовывать собственные общемодульные функции. В общем, для всех модулей, содержит методы доступа к входящим и исходящим транспортам конкретно взятого модуля.

Объект входящего транспорта TTransportIn предоставляет интерфейс к реализации модульного метода входящего транспорта.

Объект исходящего транспорта TTransportOut предоставляет интерфейс к реализации модульного метода исходящего транспорта.

6.1 Объект подсистемы «Транспорты» (TTransportS)

Наследует:	*TSubSYS.*

Публичные методы:

- *int subVer( );* — Версия подсистемы.
- *void subLoad( );* — Загрузка подсистемы.
- *void subSave( );* — Сохранение подсистемы.
- *void subStart( );* — Запуск подсистемы.

- *void subStop( );* — Останов подсистемы.
- *TElem &inEl( );* — Структура БД входящих транспортов.
- *TElem &outEl( );* — Структура БД исходящих транспортов.
- *AutoHD<TTipTransport> at(const string &iid);* — Обращение/подключение к типу транспорта *<iid>*.
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.

6.2 Модульный объект типа транспортов (TTipTransport)

Наследует:	*TModule.*
Наследуется:	Корневыми объектами модулей подсистемы «Транспорты».

Публичные методы:

- *void inList(vector<string> &list);* — Список входящих транспортов.
- *bool inPresent(const string &name);* — Проверка на наличие входящего транспорта.
- *void inAdd(const string &name, const string &idb = "*.*");* — Добавление входящего транспорта.
- *void inDel(const string &name);* — Удаление входящего транспорта.
- *AutoHD<TTransportIn> inAt(const string &name);* — Подключение к входящему транспорту.
- *void outList(vector<string> &list);* — Список исходящих транспортов.
- *bool outPresent(const string &name);* — Проверка на наличие исходящего транспорта.
- *void outAdd(const string &name, const string &idb = "*.*");* — Добавление исходящего транспорта.
- *void outDel(const string &name);* — Удаление исходящего транспорта.
- *AutoHD<TTransportOut> outAt(const string &name)* — Подключение к исходящему транспорту.
- *TTransportS &owner( );* — Подсистема «Транспорты» – владелец типом транспорта.

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *virtual TTransportIn *In(const string &name, const string &idb);* — Модульный метод создания/открытия нового «входящего» транспорта.
- *virtual TTransportOut *Out(const string &name, const string &idb);* — Модульный метод создания/открытия нового «исходящего» транспорта.

6.3 Объект входящих транспортов (TTransportIn)

Наследует:	*TCntrNode, TConfig.*
Наследуется:	Объектами входящих транспортов модулей подсистемы «Транспорты».

Публичные методы:

- *TTransportIn(const string &id, const string &idb, TElem *el);* — Инициализирующий конструктор.
- *const string &id( );* — Идентификатор транспорта.
- *string name( );* — Имя транспорта.
- *string dscr( );* — Описание транспорта.
- *string addr( );* — Адрес.

- *string protocol();* — Связанный транспортный протокол.
- *bool toStart();* — Признак «Запускать транспорт».
- *bool startStat();* — Состояние «Транспорт запущен».
- *void name(const string &inm);* — Установка имени транспорта <*inm*>.
- *void dscr(const string &idscr);* — Установка описания транспорта <*idscr*>.
- *void addr(const string &addr);* — Установка адреса транспорта <*addr*>.
- *virtual void start();* — Запуск транспорта.
- *virtual void stop();* — Останов транспорта.
- *void load( );* — Загрузка транспорта.
- *void save( );* — Сохранение транспорта.
- *string BD();* — Полное имя таблицы БД хранящей транспорт.
- *TTipTransport &owner();* — Тип транспорта – владелец входящим транспортом.

Защищённые атрибуты:

- *bool run_st;* — Признак «Запущен».

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void preEnable();* — Вызывается перед включением узла в динамическое дерево.
- *void postDisable(int flag);* — Вызывается после отключения от динамического дерева.

6.4 Объект исходящих транспортов (TTransportOut)

Наследует:	*TCntrNode, TConfig.*
Наследуется:	Объектами исходящих транспортов модулей подсистемы «Транспорты».

Публичные методы:

- *TTransportOut(const string &id, const string &idb, TElem *el);* — Инициализирующий конструктор.
- *const string &id();* — Идентификатор транспорта.
- *string name();* — Имя транспорта.
- *string dscr();* — Описание транспорта.
- *string addr();* — Адрес транспорта.
- *bool toStart();* — Признак «Запускать транспорт».
- *bool startStat();* — Состояние «Транспорт запущен».
- *void name(const string &inm);* — Установка имени транспорта.
- *void dscr(const string &idscr);* — Установка описания транспорта.
- *void addr(const string &addr);* — Установка адреса транспорта.
- *void toStart(bool val);* — Установка признака «Запускать транспорт».
- *virtual void start( );* — Запуск транспорта.
- *virtual void stop( );* — Останов транспорта.
- *void load( );* — Загрузка транспорта.
- *void save( );* — Сохранение транспорта.
- *virtual int messIO(const char *obuf, int len_ob, char *ibuf = NULL, int len_ib = 0, int time = 0);* — Отправка данных через транспорт.
- *string BD();* — Полное имя таблицы БД хранящей транспорт.
- *TTipTransport &owner();* — Тип транспорта – владелец исходящим транспортом.

Защищённые атрибуты:

- *bool run_st;* — Признак «Запущен».

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void preEnable( );* — Вызывается перед включением узла в динамическое дерево.
- *void postDisable(int flag);* — Вызывается после отключением от динамического дерева.

7 Подсистема “Протоколы коммуникационных интерфейсов” (TProtocolS)

Подсистема «Протоколы коммуникационных интерфейсов» представлена объектом TProtocolS, который содержит, на уровне подсистемы, модульные объекты отдельных протоколов TProtocol. Каждый протокол содержит объекты открытых сеансов входящих протоколов TProtocolIn.

Объект TProtocolS предоставляет доступ к входящим протоколам отдельно взятых типов транспортных протоколов.

7.1 Объект подсистемы «Протоколы коммуникационных интерфейсов» (TProtocolS)

Наследует:	*TSubSYS.*

Публичные методы:

- *int subVer( );* — Версия подсистемы.
- *void subLoad( );* — Загрузка подсистемы.
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.

7.2 Модульный объект протокола (TProtocol)

Наследует:	*TModule.*
Наследуется:	Корневыми объектами модулей подсистемы «Протоколы».

Публичные методы:

- *void list(vector<string> &list);* — Список открытых входящих сеансов.
- *bool openStat(const string &name);* — Проверка на открытость входящего сеанса с указанным именем.
- *void open(const string &name);* — Открытие входящего сеанса.
- *void close(const string &name);* — Закрытие входящего сеанса.
- *AutoHD<TProtocolIn> at(const string &name);* — Подключение к открытому входящему сеансу.

7.3 Объект сеанса входящего протокола (TProtocolIn)

Наследует:	*TCntrNode.*
Наследуется:	Объектами сеанса входящего протокола модулей подсистемы «Протоколы».

Публичные методы:

- *TProtocolIn(const string &name);* — Инициализирующий конструктор.
- *string &name( );* — Имя входящего сеанса.
- *virtual bool mess(const string &request, string &answer, const string &sender);* — Передача неструктурированных данных на обработку протоколу.
- *TProtocol &owner( );* — Протокол – владелец входящим сеансом.

8 Подсистема “Пользовательские интерфейсы” (TUIS)

Подсистема «Пользовательские интерфейсы» представлена объектом TUIS который содержит, на уровне подсистемы, модульные объекты пользовательских интерфейсов TUI.

8.1 Объект подсистемы «Пользовательские интерфейсы» (TUIS)

Наследует:	*TSubSYS.*

Публичные методы:

- *int subVer( );* — Версия подсистемы.
- *void subLoad( );* — Загрузка подсистемы.
- *static string getIco(const string &inm, string *tp = NULL);* — Загрузка изображения иконки <*inm*> из стандартной директории. Имя иконки указывается без расширения. Расширение/тип загруженного изображения помещается в <tp>.
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.

8.2 Модульный объект пользовательского интерфейса (TUI)

Наследует:	*TModule.*
Наследуется:	Корневыми объектами модулей подсистемы «Пользовательские интерфейсы».

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.

Защищённые атрибуты:

- *bool run_st;* — Признак «Модуль запущен».

9 Подсистема “Специальные” (TSpecialS)

Подсистема «Системные» представлена объектом TSpecialS, который содержит, на уровне подсистемы, модульные объекты специальных TSpecial.

9.1 Объект подсистемы «Специальные» (TSpecialS)

Наследует:	*TSubSYS.*

Публичные методы:

- *int subVer( );* — Версия подсистемы.
- *void subLoad( );* — Загрузка подсистемы.
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.

9.2 Модульный объект специальных (TSpecial)

Наследует:	*TModule.*
Наследуется:	Корневыми объектами модулей подсистемы «Специальные».

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.

Защищённые атрибуты:

- *bool run_st;* — Признак «Модуль запущен».

10 Подсистема “Безопасность” (TSecurity)

Подсистема безопасности представлена объектом TSecurity, который содержит объекты групп TGroup и пользователей TUser.

Объект пользователя TUser содержит пользовательскую информацию и выполняет проверку аутентичности пользователя в соответсвии с указанным паролем.

Объект пользователя TGroup содержит информацю о группе пользователей и выполняет проверку на принадлежность пользователя к группе.

10.1 Объект подсистемы «Безопасность» (TSecurity)

Наследует:	*TSubSYS.*

Публичные методы:

- *void subLoad( );* — Загрузка подсистемы.
- *void subSave( );* — Сохранение подсистемы.
- *bool access(const string &user, char mode, int owner, int group, int access);* — Проверка доступа для пользователя <*user*> с правами <*mode*> к ресурсу с владельцем <*owner*> и группой <*access*>.
- *string usr(int id);* — Имя пользователя в соответствии с идентификатором <*id*>.
- *int usr(const string &sid);* — Идентификатор пользователя <*sid*>.
- *void usrList(vector<string> &list);* — Список пользователей <*list*>.
- *bool usrPresent(const string &name);* — Проверка на наличие указанного пользователя <*name*>.
- *int usrAdd(const string &name, const string &idb = "*.*");* — Добавление пользователя <*name*> с хранением в БД <*idb*>.
- *void usrDel(const string &name);* — Удаление пользователя <*name*>.
- *AutoHD<TUser> usrAt(const string &name);* — Подключение к пользователю <*name*>.
- *string grp(int id);* — Имя группы пользователей в соответствии с идентификатором <*id*>.
- *int grp(const string &sid);* — Идентификатор группы пользователей <*sid*>.
- *void grpList(vector<string> &list);* — Список групп пользователей <*list*>.
- *bool grpPresent(const string &name);* — Проверка на наличие указанной группы пользователей <*name*>.
- *int grpAdd(const string &name, const string &idb = "*.*");* — Добавление группы пользователей <*name*> с хранением в БД <*idb*>.
- *void grpDel(const string &name);* — Удаление группы пользователей <*name*>.
- *AutoHD<TGroup> grpAt(const string &name);* — Подключение к группе пользователей <*name*>.
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.

10.2 Объект пользователя (TUser)

Наследует:	*TCntrNode, TConfig.*

Публичные методы:

- *TUser(const string &name, const string &idb, unsigned id, TElem *el);* — Инициализирующий конструктор.

- *const string &name( );* — Имя пользователя.
- *const string &lName( );* — Полное имя пользователя.
- *const int &id( );* — Идентификатор пользователя.
- *const string &grp( );* — Основная группа пользователя.
- *bool auth(const string &n_pass);* — Проверка аутентичности пользователя по паролю <*n_pass*>.
- *void name(const string &nm);* — Установка имени пользователя <*nm*>.
- *void lName(const string &nm);* — Установка полного имени пользователя <*nm*>.
- *void id(unsigned n_id);* — Установка идентификатора пользователя <*n_id*>.
- *void pass(const string &n_pass);* — Установка пароля пользователя <*n_pass*>.
- *void grp(const string &nm_grp);* — Установка основной группы пользователя <*nm_grp*>.
- *void load( );* — Загрузка пользователя.
- *void save( );* — Сохранение пользователя.
- *string BD( );* — Полное имя таблицы БД хранящей пользователя.
- *TSecurity &owner( );* — Подсистема «Безопасность» – владелец пользователя.

10.3 Объект группы пользователей (TGroup)

Наследует:	*TCntrNode, TConfig.*

Публичные методы:

- *TGroup(const string &name, const string &idb, unsigned id, TElem *el);* — Инициализирующий конструктор.
- *const string &name( );* — Имя группы пользователей.
- *const string &lName( );* — Полное имя группы пользователей.
- *const int &id( );* — Идентификатор группы пользователей.
- *void name(const string &nm);* — Установка имени группы пользователей <*nm*>.
- *void lName(const string &nm);* — Установка полного имени группы пользователей <*nm*>.
- *void id(unsigned n_id);* — Установка идентификатора группы пользователей <*n_id*>.
- *bool user(const string &name);* — Проверка на принадлежность пользователя к группе <*name*>.
- *void load( );* — Загрузка пользователя.
- *void save( );* — Сохранение пользователя.
- *string BD( );* — Полное имя таблицы БД хранящей группу пользователей.
- *TSecurity &owner( );* — Подсистема «Безопасность» – владелец группой пользователей.

11 Подсистема “Управление модулями” (TModSchedul)

Подсистема «Управление модулями» представлена объектом TModSchedul.

Подсистема содержит механизм управления модулями содержащимися в разделяемых библиотеках.

11.1 Объект подсистемы «Управление модулями» (TModSchedul)

Наследует:	*TSubSYS.*

Данные:

Структура информации о разделяемой библиотеке (struct – TModSchedul::SHD):

- *void *hd;* — заголовок разделяемой библиотеки (если NULL то библиотека присутствует но не подключена);
- *vector<SUse> use;* — список подключенных модулей;
- *time_t m_tm;* — время модификации библиотеки;
- *string name;* — полное имя/путь разделяемой библиотеки.

Структура информации о модуле (struct – TModSchedul::SUse):

- *string mod_sub;* — имя модульной подсистемы;
- *string n_mod;* — имя модуля.

Публичные методы:

- *void chkPer(int per);* — Установка периода проверки директории с модулями (сек). Если периодичность равна нуль, то проверка будет отключена.
- *void subLoad( );* — Загрузка подсистемы.
- *void subSave( );* — Сохранение подсистемы.
- *void subStart( );* — Запуск подсистемы.
- *void subStop( );* — Останов подсистемы.
- *void loadLibS( );* — Загрузка разделяемых библиотек и инициализация модулей,
- *SHD &lib(const string &name);* — Получение объекта разделяемой библиотеки <*name*>.
- *void libList(vector<string> &list);* — Список разделяемых библиотек <*list*>.
- *void libLoad(const string &path, bool full);* — Загрузка разделяемых библиотек по указанному пути <*path*>.
- *void libAtt(const string &name, bool full = false);* — Подключение указанной разделяемой библиотеки <*name*>.
- *void libDet(const string &name);* — Отключение разделяемой библиотеки <*name*>.
- *string optDescr( );* — Локализованная помощь по опциям командной строки и параметрам конфигурационного файла.

12 Подсистема “Параметры” (TParamS)

Подсистема «Параметры» представлена объектом TParamS, который содержит объекты параметров логического уровня TParam и шаблоны параметров TPrmTempl.

Подсистема выполняет периодический обсчёт параметров построенных на основе шаблонов.

Объект параметра логического уровня TParam предоставляет механизм связывания с параметрами физического уровня, а также рекурсивное связывание с параметрами логического уровня. Реализованы механизмы: прямого отражения и связывание по шаблону.

Общий принцип формирования параметра логического уровня представлено на рис.7.

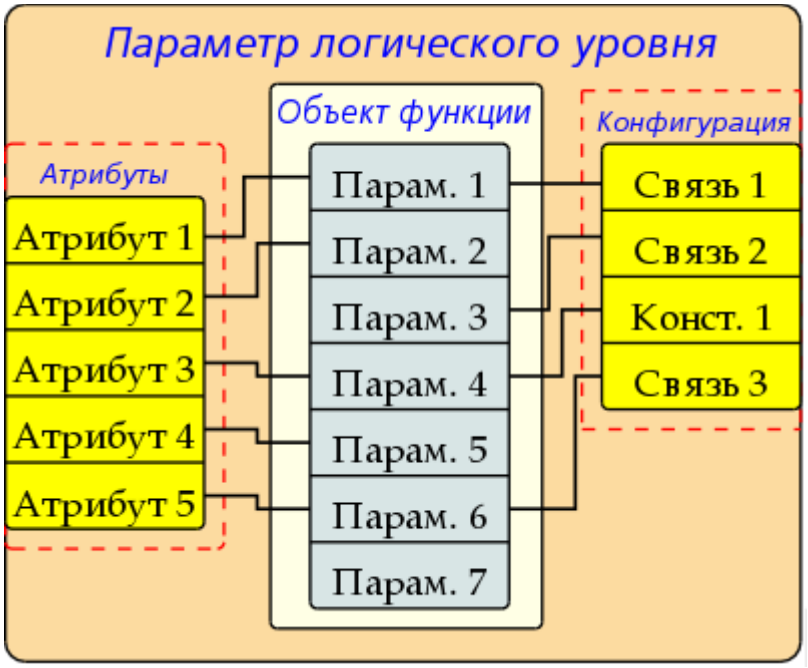

Рис. 7. Формирование параметра логического уровня по шаблону.

Объект шаблона параметра содержит структуру шаблона параметра основанную на функциях объектной модели системы OpenSCADA, добавляя к структуре функции описание связей параметра функции.

12.1 Объект подсистемы «Параметры» (TParamS)

Наследует:	*TSubSYS*

Публичные методы:

- *int calcPer( );* — Период обсчёта параметров основанных на шаблонах (мс).
- *void calcPer(int iper);* — Установка периода обсчёта параметров основанных на шаблонах (мс).
- *void subLoad( );* — Загрузка подсистемы.
- *void subSave( );* — Сохранение подсистемы.
- *void subStart( );* — Запуск подсистемы.
- *void subStop( );* — Останов подсистемы.
- *void list(vector<string> &list);* — Список доступных параметров логического уровня *<list>*.
- *bool present(const string &param);* — Проверка на наличие параметра логического уровня *<param>*.

- *void add(const string &id, const string &idb = "*.*");* — Добавление параметра логического уровня <*id*> с хранением в БД <*idb*>.
- *void del(const string &id);* — Удаление параметра логического уровня <*id*>.
- *AutoHD<TParam> at(const string &name, const string &who = "");* — Подключение к параметру логического уровня <*name*>.
- *void tplList(vector<string> &list);* — Список доступных шаблонов параметров <*list*>.
- *bool tplPresent(const string &tpl);* — Проверка на наличие шаблона параметров <*tpl*>.
- *void tplAdd(const string &tpl, const string &idb = "*.*");* — Добавление шаблона параметров <*tpl*>.
- *void tplDel(const string &tpl);* — Удаление шаблона параметров <*tpl*>.
- *AutoHD<TPrmTempl> tplAt(const string &tpl, const string &who = "");* — Подключение к шаблону параметров <*tpl*>.
- *TElem &prmE();* — Структура БД параметра.
- *TElem &prmIOE();* — Структура БД атрибута параметра.
- *TElem &tplE();* — Структура БД шаблона параметра.
- *TElem &tplIOE();* — Структура БД атрибута шаблона параметра.

12.2 Объект параметра логического уровня (TParam)

Наследует:	*TValue, TConfig*

Данные:

Режимы функционирования параметра логического уровня (enum – TParam::Mode):

- *Clear* — пустой или не сконфигурированный;
- *DirRefl* — прямое отражение параметра(ов) физического уровня с возможностью полного резервирования (Планируется);
- *Template* — работа по шаблону параметра.

Публичные методы:

- *TParam(const string &iid, const string &idb, TElem *cf_el);* — Инициализирующий конструктор параметра с хранением в БД <*idb*> структурой <*cf_el*>.
- *const string &id( );* — Идентификатор параметра.
- *string name( );* — Имя параметра.
- *string descr( );* — Описание параметра.
- *void name(const string &inm);* — Установка имени параметра <*inm*>.
- *void descr(const string &idsc);* — Установка описания параметра <*idsc*>.
- *bool toEnable( );* — Признак «Включать параметр».
- *bool enableStat( );* — Состояние «Параметр включен».
- *Mode mode( );* — Режим параметра логического уровня.
- *void mode(Mode md, const string &prm = "");* — Установка режима <*md*> с параметром <*prm*>.
- *void enable( );* — Включение параметра.
- *void disable( );* — Отключение параметра.
- *void load( );* — Загрузка параметра.
- *void save( );* — Сохранение параметра.
- *void calc();* — Вычисление, для параметра использующего шаблон.
- *string BD();* — Полное имя таблицы БД хранящей параметр.
- *TParamS &owner();* — Подсистема «Параметры» – владелец параметра.

12.3 Объект шаблона параметра (TPrmTempl)

Наследует:	*TCntrNode, TConfig.*

Данные:

Режим отражения параметра функции на атрибут параметра логического уровня (enum – TPrmTempl::AttrMode):

- *NoAttr* — не атрибут параметра логического уровня;
- *ReadOnly* — атрибут с доступом только на чтение;
- *FullAccess* — атрибут с полным доступом.

Режим доступа к параметру функции шаблона (enum – TPrmTempl::AccMode):

- *Const* — постоянная уровня шаблона, на уровне параметра не доступна;
- *PublConst* — публичная константа, доступна везде;
- *Link* — связь на атрибут внешнего параметра.

Структура параметров функции для шаблона (class – TPrmTempl::SIOPrm):

- *SIOPrm(const string iid, AttrMode iattr, AccMode iaccs, const string ival);* — Инициализирующий конструктор структуры.
- *string id;* — идентификатор связи;
- *AttrMode attr;* — режим атрибута параметра функции;
- *AccMode accs;* — режим доступа к параметру функции;
- *string val;* — значение параметра функции.

Публичные методы:

- *TPrmTempl(const string &id, const string &idb, TElem *cf_el);* — Инициализирующий конструктор шаблона с хранением в БД <*idb*> структурой <*cf_el*>.
- *const string &id();* — Идентификатор шаблона.
- *string name();* — Имя шаблона.
- *string descr();* — Описание шаблона.
- *void name(const string &inm);* — Установка имени шаблона <*inm*>.
- *void descr(const string &idsc);* — Установка описания шаблона <*idsc*>.
- *void load( );* — Загрузка шаблона.
- *void save( );* — Сохранение шаблона.
- *bool enable();* — Состояние шаблона «Включен».
- *void enable(bool val);* — Включение/выключение <*val*> шаблона.
- *AutoHD<TFunction> func();* — Функция ассоциированная с шаблоном.
- *void attrUp();* — Загрузка атрибутов.
- *void attrSave();* — Сохранение атрибутов.
- *int attrSize();* — Количество атрибутов.
- *int attrId(const string &id);* — Индекс атрибута по его идентификатору <*id*>.
- *SIOPrm &attr(int id);* — Получение параметров атрибута <*id*>.
- *string BD();* — Полное имя таблицы БД содержащей шаблон.
- *TParamS &owner();* — Подсистема «Параметры» – владелец шаблона.

13 Компоненты объектной модели системы OpenSCADA

Объектная модель системы OpenSCADA строится на основе объекта функции *TFunction*, параметрах функции *IO* и кадре значений функции *TValFunc*. В последствии, объекты функции включаются в объектное дерево формируя объектную модель системы. Использование функций объектной модели производится путём связывания кадра значений *TValFunc* с функцией.

Идея в целом доступно представленна на рис. 8.

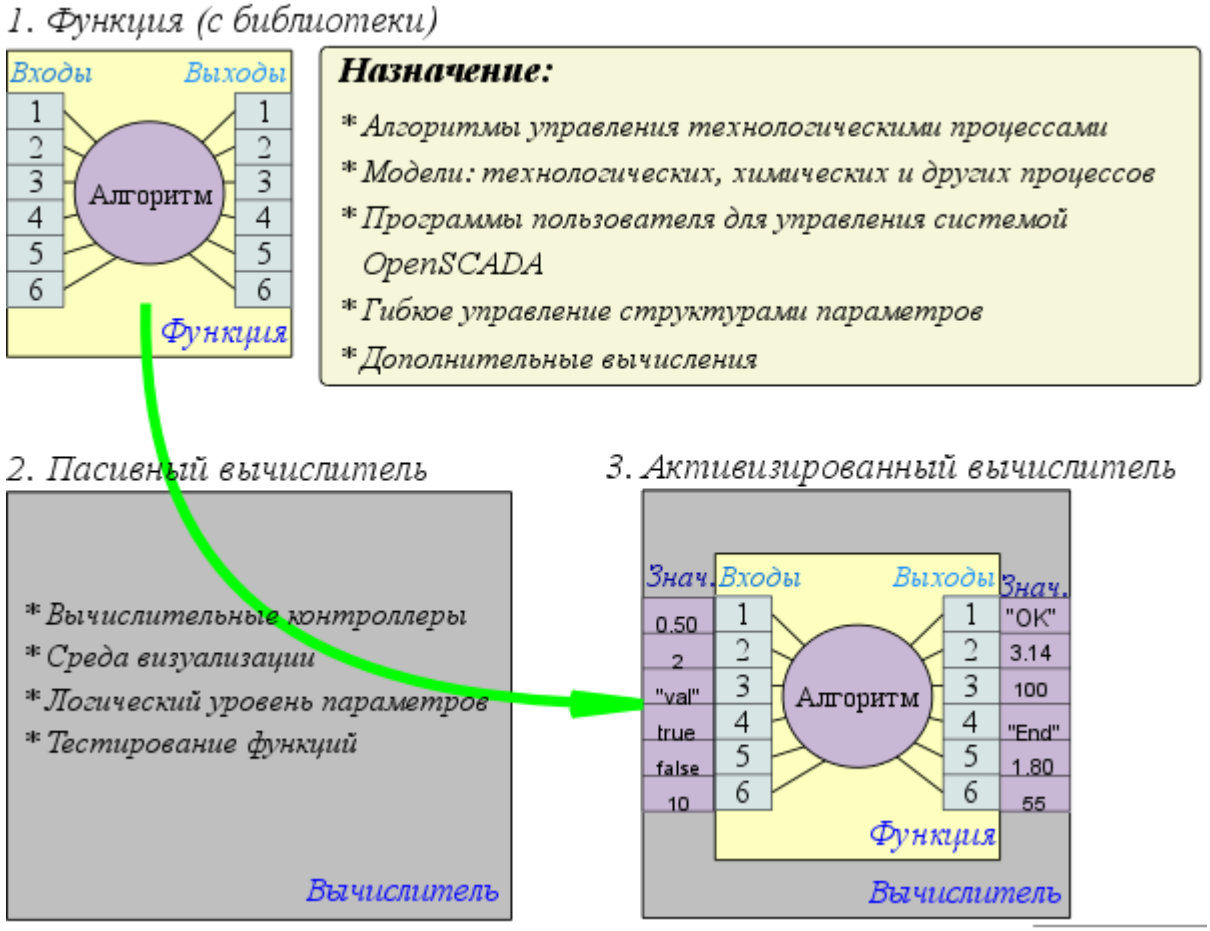

Рис. 8. Основа среды программирования системы Open SCADA?.

Объект функции (TFunction) предоставляет интерфейс для формирования параметров функции и алгоритма вычисления в объекте наследующем его.

Объект параметра функции (IO) содержит конфигурацию отдельно взятого параметра.

Объект кадра значений (TValFunc) содержит значения в соответствии со структурой связанной функции. При исполнении алгоритма ассоциированой функции используются значения этого объекта.

13.1 Объект функции (TFunction)

Наследует:	*TCntrNode*
Наследуется:	Модулями и узлами систем содержащими функции для публикации в объектную модель системы.

Публичные методы:

- *TFunction(const string &iid);* — Инициализирующий конструктор функции с идентификатором <*id*>.
- *string &id();* — Идентификатор функции.
- *virtual string name() = 0;* — Локализованное имя функции.
- *virtual string descr() = 0;* — Описание функции.
- *bool startStat();* — Состояние «Функция запущена».
- *virtual void start(bool val);* — Запуск/останов <*val*> функции.
- *void ioList(vector<string> &list);* — Список параметров функции <*list*>.
- *int ioId(const string &id);* — Получение идентификатора параметра <*id*>.
- *int ioSize();* — Количество параметров.
- *IO *io(int id);* — Получение параметра по идентификатору <*id*>.
- *virtual void calc(TValFunc *val) = 0;* — Вычисление алгоритма функции над указанными значениями <*val*>.
- *void valAtt(TValFunc *vfnc);* — Вызывается объектом кадра значений <*vfnc*> в случае связывания с функцией.
- *void valDet(TValFunc *vfnc);* — Вызывается объектом кадра значений <*vfnc*> в случае развязывания от функции.
- *virtual void preIOCfgChange();* — Вызывается перед изменением конфигурации.
- *virtual void postIOCfgChange();* — Вызывается после изменения конфигурации.

Защищённые атрибуты:

- *string m_id;* — Идентификатор функции.
- *bool run_st;* — Признак «Запущен».
- *TValFunc *m_tval;* — Ссылка на объект значений используемый для тестирования функции. Может отсутствовать.
- *static int n_tcalc;* — Количество итераций тестового вычисления.

Защищённые методы:

- *void cntrCmd_(const string &a_path, XMLNode *opt, TCntrNode::Command cmd);* — Обслуживание команд интерфейса управления системой.
- *void ioAdd(IO *io);* — Добавить параметр <*io*>.
- *void ioIns(IO *io, int pos);* — Вставить параметр <*io*> в позицию <*pos*>.
- *void ioDel(int pos);* — Удалить параметр с позиции <*pos*>.
- *void ioMove(int pos, int to);* — Переместить параметр с позиции <*pos*> в позицию <*to*>.
- *void preDisable(int flag);* — Выполняется перед исключением из динамического дерева.

13.2 Объект параметра функции (IO)

Данные:

Типы параметра (enum – IO::Type):

- *IO::String* — строка/текст;
- *IO::Integer* — целое;
- *IO::Real* — вещественное;
- *IO::Boolean* — логический;
- *IO::Vector* — вектор (не используется).

Режимы параметра (enum – IO::Mode):

- *IO::Input* — вход;
- *IO::Output* — выход;

- *IO::Return* — возврат.

Публичные методы:

- *IO(const char *iid, const char *iname, IO::Type itype, IO::Mode imode, const char *idef = "", bool ihide = false, const char *ivect = "");* — Инициализирующий конструктор.
- *const string &id( );* — Идентификатор параметра функции.
- *const string &name( );* — Локализованное имя параметра функции.
- *const Type &type( );* — Тип параметра функции.
- *const Mode &mode( );* — Режим доступа к параметру функции.
- *const string &def( );* — Значение по умолчанию.
- *const string &vector( );* — Вектор к которому принадлежит параметр функции.
- *bool hide( );* — Признак «Скрыто».
- *void id(const string &val);* — Установить идентификатор <*val*>.
- *void name(const string &val);* — Установить имя <*val*>.
- *void type(Type val);* — Установить тип <*val*>.
- *void mode(Mode val);* — Установить режим <*val*>.
- *void def(const string &val);* — Установить значение по умолчанию <*val*>.
- *void vector(const string &val);* — Установит вектор <*val*>.
- *void hide(bool val);* — Установить/снять <*val*> признак «Скрыто».

13.3 Объект значения функции (TValFunc).

Публичные методы:

- *TValFunc(const string &iname = "", TFunction *ifunc = NULL, bool iblk = true);* — Инициализирующий конструктор.
- *const string &name( );* — Имя объекта значений.
- *void name(const char &inm);* — Установить имя <*inm*> объекта значений.
- *void ioList(vector<string> &list);* — Список параметров функции <*list*>.
- *int ioId(const string &id);* — Получение идентификатора параметра <*id*>.
- *int ioSize( );* — Общее количество параметров.
- *IO::Type ioType(unsigned id);* — Тип параметра <*id*>.
- *IO::Mode ioMode(unsigned id);* — Режим параметра <*id*>.
- *bool ioHide(unsigned id);* — Признак параметра <*id*> «Скрыт».
- *string getS(unsigned id);* — Получить значение (строка) параметра <*id*>.
- *int getI(unsigned id);* — Получить значение (целое) параметра <*id*>.
- *double getR(unsigned id);* — Получить значение (вещественное) параметра <*id*>.
- *bool getB(unsigned id);* — Получить значение (логическое) параметра <*id*>.
- *void setS(unsigned id, const string &val);* — Установить значение <*val*> (строка) параметра <*id*>.
- *void setI(unsigned id, int val);* — Установить значение <*val*> (целое) параметра <*id*>.
- *void setR(unsigned id, double val);* — Установить значение <*val*> (вещественное) параметра <*id*>.
- *void setB(unsigned id, bool val);* — Установить значение <*val*> (логическое) параметра <*id*>.
- *bool blk( );* — Признак «Блокирование изменений параметров функции».
- *bool dimens( );* — Признак «Измерять время вычисления».
- *void dimens(bool set)* — Установка/сброс <*set*> признака «Измерять время вычисления».
- *virtual void calc( );* — Вычислить.
- *double calcTm( );* — Время вычисления.

- *TFunction *func( );* — Связанная функция.
- *void func(TFunction *ifunc, bool att_det = true);* — Связать с указанной функцией <*ifunc*>.
- *virtual void preIOCfgChange( );* — Вызывается перед изменением конфигурации.
- *virtual void postIOCfgChange( );* — Вызывается после изменения конфигурации.

14 Данные в системе OpenSCADA и их хранение в БД (TConfig)

Хранение данных в системе построено на объектах *TConfig* и *TElem*. Эти объекты хранят структуру и значения полей БД, что позволяет выполнять прямую загрузку и сохранение конфигурации через подсистему «БД».

Объект *TElem* содержит структуру записи БД. Структура записи содержит исчерпывающую информацию о элементах, их типах, размерах и остальных параметрах. Информации в данной структуре достаточно для создания, контроля и управления реальной структурой БД. Элементарной единицей записи является ячейка *Tfld*.

Объект *TConfig* является наследником от *TElem* и содержит реальные значения элементов. *TConfig* используется в качестве параметра в функциях манипуляции с записями таблиц в подсистеме «БД». Элементарной единицей записи является ячейка *TCfg*.

Для предоставления возможности предупреждения хранилища данных о смене структуры предусмотрен объект *TValElem*, от которого наследуется хранилище *TConfig* и список которых содержится в структуре *TElem*.

14.1 Объект данных (TConfig)

Наследует:	*TValElem*
Наследуется:	*TParamContr, TController, TArchiveMess, TArchiveVal, TTransportIn, TTransportOut, TUser, TGroup, TParamS, а также модульные объекты хранящие свои данные в БД.*

Публичные методы:

- *TConfig(TElem *Elements = NULL);* — Инициализирующий конструктор.
- *TConfig &operator=(TConfig &cfg);* — Копирование из <*cfg*>.
- *void cfgList(vector<string> &list);* — Список элементов <*list*>.
- *TCfg &cfg(const string &n_val);* — Получение элемента <*n_val*>.
- *void elem(TElem *Elements, bool first = false);* — Назначение структуры <*Elements*>.
- *TElem &elem();* — Структура.

Защищённые методы:

- *virtual bool cfgChange(TCfg &cfg);* — Вызывается в случае изменения содержимого ячейки.
- *void cntrMake(XMLNode *fld, const char *req, const char *path, int pos);* — Генерация формы данных для интерфейса управления системой.
- *void cntrCmd(const string &elem, XMLNode *fld, TCntrNode::Command cmd);* — Обработка команд интерфейса управления системой для формы данных.

14.2 Ячейка данных (TCfg)

Данные:

Дополнительные флаги к *TFld* (define):

- *FLD_NOVAL* — не отражать на значения (TValue);
- *FLD_KEY* — ключевое поле.

Публичные методы:

- *TCfg(TFld &fld, TConfig &owner);* — Инициализирующий конструктор.
- *const string &name();* — Имя ячейки.

- *bool operator==(TCfg &cfg);* — Сравнение ячеек.
- *TCfg &operator=(TCfg &cfg);* — Копирование ячеек.
- *bool view( );* — Признак «Ячейка видима».
- *void view(bool vw);* — Установка признака «Ячейка видима» <*vw*>.
- *TFld &fld( );* — Конфигурация ячейки.
- *string getSEL( );* — Получить значение выборочного типа.
- *string getS( );* — Получить значение строкового типа.
- *double getR( );* — Получить значение вещественного типа.
- *int getI( );* — Получить значение целого типа.
- *bool getB( );* — Получить значение логического типа.
- *string &getSd( );* — Получить прямой доступ к значению строкового типа.
- *double &getRd( );* — Получить прямой доступ к значению вещественного типа.
- *int &getId( );* — Получить прямой доступ к значению целого типа.
- *bool &getBd( );* — Получить прямой доступ к значению логического типа.
- *void setSEL(const string &val);* — Установить значение <*val*> выборочного типа.
- *void setS(const string &val);* — Установить значение <*val*> строкового типа.
- *void setR(double val);* — Установить значение <*val*> вещественного типа.
- *void setI(int val);* — Установить значение <*val*> целого типа.
- *void setB(bool val);* — Установить значение <*val*> логического типа.

14.3 Объект структуры данных (TElem)

Наследуется:	*TTipParam, TControllerS, TTipController*, а также модульными объектами совмещающими функции хранения структуры.

Публичные методы:

- *TElem(const string &name = "");* — Инициализация структуры с указанным именем <*name*>.
- *string &elName( );* — Имя структуры.
- *void fldList(vector<string> &list);* — Список ячеек в структуре <*list*>.
- *unsigned fldSize( );* — Количество ячеек в структуре.
- *unsigned fldId(const string &name);* — Получение индекса ячейки по её идентификатору <*name*>.
- *bool fldPresent(const string &name);* — Проверка на наличие указанной ячейки <*name*>.
- *int fldAdd(TFld *fld, int id = -1);* — Добавление/вставка ячейки <*fld*> в позицию <*id*> (-1 – вставка в конец).
- *void fldDel(unsigned int id);* — Удаление ячейки <*id*>.
- *TFld &fldAt(unsigned int id);* — Получение ячейки <*id*>.
- *void valAtt(TValElem *cnt);* — Вызывается автоматически в случае подключения структуры к хранилищу данных <*cnt*>.
- *void valDet(TValElem *cnt);* — Вызывается автоматически в случае отключения структуры от хранилища данных <*cnt*>.

14.4 Ячейка структуры данных (TFld)

Данные:

Тип ячейки (enum – TFld::Type):

- *TFld::Bool* — логический тип;
- *TFld::Dec* — десятичный тип;

- *TFld::Hex* — шестнадцатеричный тип;
- *TFld::Oct* — восьмеричный тип;
- *TFld::Real* — вещественный тип;
- *TFld::String* — строковый тип.

Флаги ячейки (define):

- *FLD_NOFLG* — флаги отсутствуют;
- *FLD_SELECT* — режим выборки из доступных значений;
- *FLD_SELF* — создавать собственную копию этой ячейки;
- *FLD_NWR* — не доступна для записи;
- *FLD_PREV* — предупреждать про модификацию значения ячейки.

Публичные методы:

- *TFld( );* — Инициализация по умолчанию.
- *TFld(const char *name, const char *descr, Type itype, unsigned char iflg, const char *valLen = "", const char *valDef = "", const char *vals = "", const char *nSel = "", int w_id = 0);* — Инициализация с указанной конфигурацией.
- *TFld &operator=(TFld &fld);* — Копирование ячейки из <*fld*>.
- *const string &name( );* — Имя ячейки.
- *string &descr( );* — Описание ячейки.
- *int len( );* — Размер значения ячейки (символов в символьном представлении).
- *int dec( );* — Размер дробной части для вещественного (символов в символьном представлении).
- *Type type( );* — Тип ячейки.
- *unsigned char flg( );* — Флаги ячейки.
- *const string &def( );* — Значение по умолчанию.
- *int workId( );* — Рабочий идентификатор. Ключевое значение, может использоваться в служебных целях потомками, например для быстрой связки этого поля со своими структурами.
- *vector<string> &selValS( );* — Список вариантов значений для строкового типа.
- *vector<int> &selValI( );* — Список вариантов значений для целого типа.
- *vector<double> &selValR( );* — Список вариантов значений для вещественного типа.
- *vector<bool> &selValB( );* — Список вариантов значений для логического типа.
- *vector<string> &selNm( );* — Список имён вариантов значений.
- *string selVl2Nm(const string &val);* — Получить выбранное имя по значению <*val*> строкового типа.
- *string selVl2Nm(int val);* — Получить выбранное имя по значению целого <*val*> типа.
- *string selVl2Nm(double val);* — Получить выбранное имя по значению <*val*> вещественного типа.
- *string selVl2Nm(bool val);* — Получить выбранное имя по значению <*val*> логического типа.
- *string &selNm2VlS(const string &name);* — Получить значение строкового типа по выбранному имени <*name*>.
- *int selNm2VlI(const string &name);* — Получить значение целого типа по выбранному имени <*name*>.
- *double selNm2VlR(const string &name);* — Получить значение вещественного типа по выбранному имени <*name*>.
- *bool selNm2VlB(const string &name);* — Получить значение логического типа по выбранному имени <*name*>.
- *void cntrMake(XMLNode *fld, const char *req, const char *path, int pos);* — Создать

элемент формы в соответствии с параметрами ячейки.

14.5 Объект упреждения про смену структуры (TValElem)

Наследуется:	*TValue, TConfig.*

Защищённые методы:

- *virtual void detElem(TElem *el);* — Уведомление элеменом <*el*> контейнера про желание отключиться.
- *virtual void addFld(TElem *el, unsigned id) = 0;* — Уведомление про добавление ячеки <*id*> елемента <*el*>.
- *virtual void delFld(TElem *el, unsigned id) = 0;* — Уведомление про удаление ячеки <*id*> елемента <*el*>.

15 Интерфейс управления системой и динамическое дерево объектов системы (TCntrNode)

Для полного покрытия ключевых компонентов системы сетью объектов единой структуры предназначен объект узла динамического дерева TCntrNode. На этот объект возлагаются функции:

- единообразного доступа к компонентам системы;
- построение распределённого интерфейса управления.

Любой объект имеющий потребность в предоставлении динамического доступа к себе или своим компонентам должен наследоваться от объекта узла динамического дерева TCntrNode. Данное родство автоматически включает узел в динамическое дерево объектов охваченное как прямой так и обратной связью, а также предоставляет возможность создания контейнеров под собственные дочерние узлы. Также, узел получает возможность упреждения про включение и исключение/удаление узла из дерева, с возможностью отказа от исключения/удаления.

Интерфейс управления системы включён в состав объекта TCntrNode и соответственно охватывает все узлы динамического дерева системы позволяя единообразно управлять системой в не зависимости от используемого инструмента. Интерфейс управления системой выполнен на основе языка разметки XML. Можно придумать множество способов использования интерфейса управления системой, в качестве примера отметим следующие наиболее яркие решения:

- Web интерфейс конфигурирования;
- GUI интерфейс конфигурирования (QT, GTK+, ...);
- отражение конфигурации в сеть, для распределённого управления множеством OpenSCADA-станций из единой среды администрирования;
- использование в роли протокола для доступа к данным объектов из сети;
- LDAP.

Интерфейс управления системой реализован посредством двух составляющих:

- информационной статической структуры конфигурационной страницы;
- динамической части, запросы на получение и модификацию данных.

Информационная иерархическая структура содержит информацию о публичных элементах управления и может быть использована для построения диалогов управления узлами системы.

Динамическая часть содержит сценарии обслуживания запросов к элементам управления описанным в информационной структуре, а также скрытые элементы управления, используемые для унифицированного доступа к узлу внешних элементов.

Общий интерфейс управления строится из отдельных узлов динамического дерева. Иерархичность наследования от объекта TCntrNode позволяет реализовывать многоуровневое дополнение конфигурации интерфейса управления. Общий вид динамического дерева узлов представлен на рис. 9.

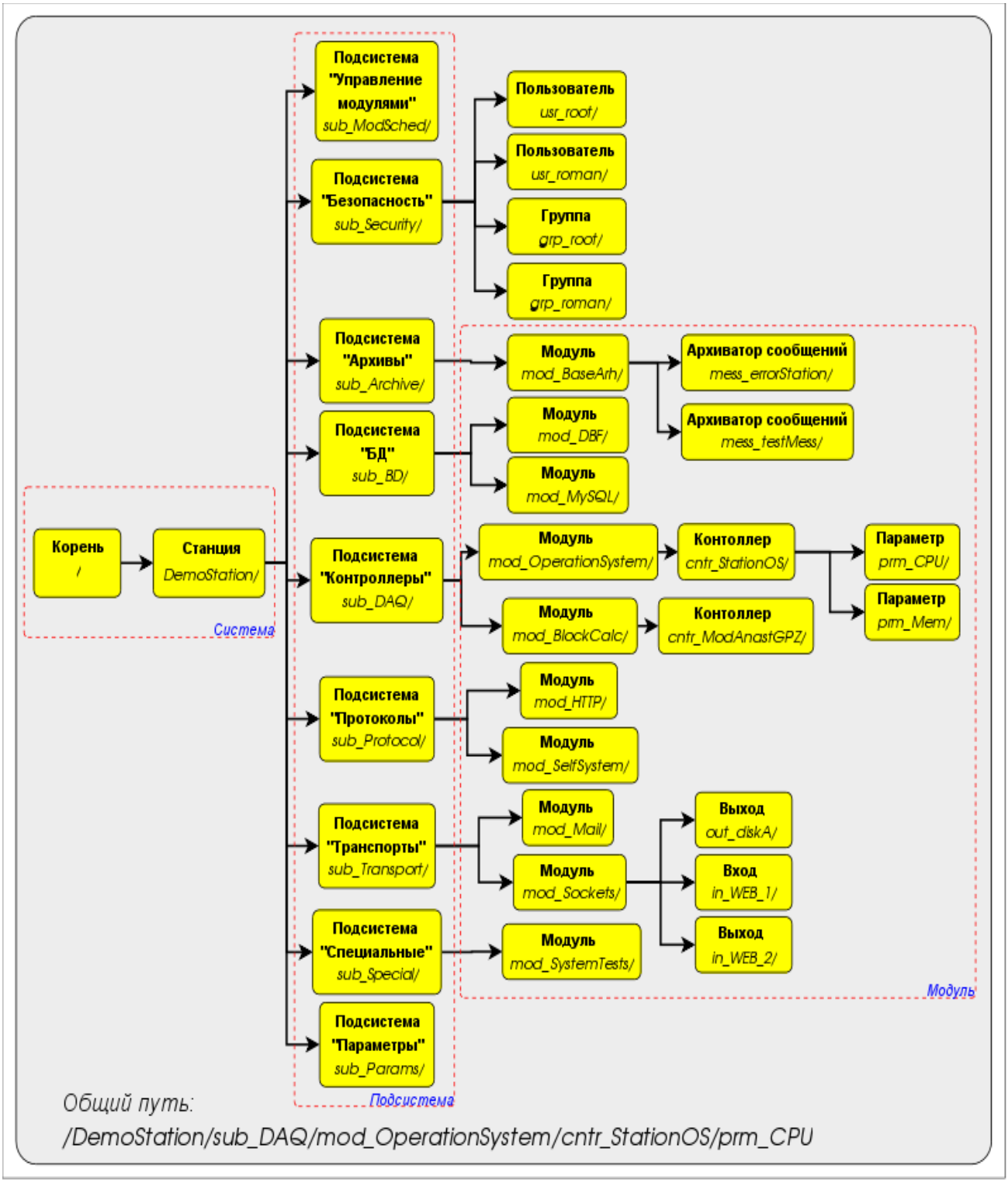

Рис. 9. Пример динамического дерева узлов системы OpenSCADA.

Узлы системы, содержащие данные для интерфейса управления системой, также, должны подключаться в динамическое дерево объектов. Подключение узла к динамическому дереву производится следующим образом:

- наследование объекта *TCntrNode* или его потомка;
- формирование информационной структуры;
- обслуживание запросов к динамическим данным.

15.1 Информационные теги интерфейса управления системой

Информационные теги для языка XML составляют алфавит формирования описания конфигурационных диалогов.

Тег области <area>

Область описываются тегом <area> и предназначены для группировки элементов по различным признакам. Область может включать другие элементы, и области. Корневые области формируют закладки, в представлении пользовательского интерфейса.

```
<area id='base' dscr='Base information'>
 <fld id='host' dscr='Host name' acs='0444' tp='str'/>
 <fld id='user' dscr='Operated user' acs='0444' tp='str'/>
 <fld id='sys' dscr='Station system' acs='0444' tp='str'/>
 <area id='other' dscr='Other options'>
   <fld id='val' dscr='Value' acs='0664' tp='real'/>
 </area>
</area>
```

Теги данных

Теги описывающие данные приведены в таблице 1.

Таблица 1. **Теги описывающие данные**

Тег	**Описание**
<fld>	Простейшие данные строкового, целого, вещественного и логического типов.
<list>	Списки с данными строкового, целого, вещественного и логического типов.
<table>	Таблицы с данными в ячейках строкового, целого, вещественного и логического типов.
<img>	Изображения.

Тег <fld>

```
<fld id='host' dscr='Host name' acs='0444' tp='str'/>
```

<fld id='user' dscr='Operated user' acs='0444' tp='str'/>

<fld id='sys' dscr='Station system' acs='0444' tp='str'/>

Тип элемента описываемого тегом <fld> указывается атрибутом <tp> (таблица 2).

Таблица 2. **Значения атрибута <tp> тега <fld>.**

Тег <tp>	**Описание**
str	Строковый тип. *<fld id='host' dscr='Host name' acs='0444' tp='str'> server </fld>*
dec	Целое число в десятичном представлении.

Тег <tp>	Описание
	<fld id='debug' dscr='Debug level' acs='0660' tp='dec' min='0' max='8'> 7 </fld>
oct	Целое число в восьмеричном представлении. *<fld id='cr_file_perm' dscr='Make files permissions(default 0644)' acs='0660' tp='oct' len='3'/> 0600 </fld>*
hex	Целое число в шестнадцатеричном представлении.
real	Вещественное число.
bool	Логический признак («false»|"true"). *<fld id='log_sysl' dscr='Direct messages to syslog' acs='0660' tp='bool'> true </fld>*
time	Время в секундах (от 01/01/1970). *<fld id='v_beg' tp='time'>FF4556DA</fld>*

Таблица 3. **Действия над элементом описанным тегом <fld>.**

Операция	Действие
Опрос	• Запрос: запрос содержимого, команда «Get». • Результат: значение в тексте тега.

Модификация	• Запрос:
	  • инициализация тега: <set>value</set>;
	  • отправка запроса, команда «Set».

Тег <list>

```
  <list id='mod_auto' dscr='List of shared libs(modules)' tp='str' dest='file'>
    <el id='0'>./MODULES/arh_base.o</el>
    <el id='1'>./MODULES/cntr_sys.o</el>
  </list>
```

Тип элементов в списке указывается атрибутом *<tp>*. Значения атрибута *<tp>* приведены в таблице 1.

Таблица 4. **Действия над списком.**

Операция	Действие
Опрос	• Запрос: запрос содержимого, команда «Get». • Результат: строки в тегах *<el>*.
Добавление строки	• Запрос: инициализация тега: <add id='tst'>Test</add>

	Читается как – добавить строку с индексом «tst» и значением «Test». Если список неиндексированный то атрибут id отсутствует. • Отправка запроса, команда «Set».
Вставка строки	• Запрос: инициализация тега: <ins pos='3' p_id='tst1' id='tst' >Test</ins> Читается как – вставить строку с индексом «tst» и значением «Test» в позицию 3 со строкой «tst1». В случае индексного списка атрибут p_id содержит индекс иначе текст строки. Если список неиндексированный то атрибут id отсутствует. • Отправка запроса, команда «Set».
Удаление строки	• Запрос: инициализация тега: <del pos='3' id='tst'>Test</del> Читается как – удалить строку с индексом «tst» и значением «Test» в позиции 3. Если список неиндексированный то атрибут id отсутствует. • Отправка запроса, команда «Set».
Изменение строки	• Запрос: инициализация тега: <edit pos='3' p_id='tst1' id='tst' >Test</edit> Читается как – заменить строку в позиции 3 на строку с индексом «tst» и значением «Test». В случае индексного списка атрибут p_id содержит индекс иначе текст строки. Если список неиндексированный то атрибут id отсутствует. • Отправка запроса, команда «Set».
Перемещение строки	• Запрос: инициализация тега: <move pos='3' to='5'/> Читается как – переместить строку с позиции 3 в позицию 5. • Отправка запроса, команда «Set».

Тег <table>

```
<table id='a_mess' key='0'>
 <list id='0' tp='str'>
  <el id='0'>Sat Feb 21 18:04:16 2004</el>
 </list>
 <list id='1' tp='str'>
  <el id='0'>SYS</el>
 </list>
```

```
  <list id='2' tp='str'>
    <el id='0'>*:(TSYS)Broken PIPE signal allow!</el>
  </list>
</table>
```

Если указан атрибут <*key*> и в нём перечислены ключевые колонки, то работа с таблицей переходит в режим адресации по идентификаторам колонок и ключам.

Таблица 5. **Действия над таблицей.**

Операция	**Действие**
Опрос	• Запрос: Запрос содержимого. Команда «Get». • Результат: Группа списков <list> с строками в тегах <el>.
Добавление строки	• Запрос: Инициализация тега: <add/>. Добавление строки. • Отправка запроса. Команда «Set».
Вставка строки	• Запрос: Инициализация тега: <ins row='3'/>. Вставка строки в позицию 3. Данная команда не работает при установленном атрибуте <key>! • Отправка запроса. Команда «Set».
Удаление строки	• Запрос: Инициализация тега: <del row='3'/>. Удалить строку в позиции 3. Если определён атрибут <key>, то тег запишется: <del key_id='Test'/>. Удаление строки в позиции где значение колонки <id> равно 'Test'. • Отправка запроса. Команда «Set».
Перемещение строки	• Запрос: Инициализация тега: <move row='3' to='5'/>. Переместить строку с позиции 3 в позицию 5. Данная команда не работает при установленном атрибуте <key>! • Отправка запроса. Команда «Set».
Изменить ячейку	• Запрос: Инициализация тега: <set row='3' col='id'> Test </set>. Установить значение ячейки в строке 3 и колонке 'id' в «Test». Если определён атрибут <key>, то тег запишется: <set key_id='Test' col='id'>Test1</set>. Установка колонки с именем 'id' строки в позиции где значение колонки <id> равно 'Test' в значение 'Test1'. Практически, данная команда переименовывает ключевой элемент указанной строки. • Отправка запроса. Команда «Set».

Тег <img>

```
<img id='ico' descr='Иконка страницы'>
// Mime Base64 кодированный файл изображения иконки
</img>
```

Тег преднажначен для передачи изображений клиентам интерфейса управления. Под изображением могут выступать: иконки страниц, графики массивов значений и другие данные которые можно представить как изображение.

Команды с параметрами. Тег <comm>

```
<comm id='add'>
 <fld id='tm' tp='time'/>
 <fld id='cat' tp='str'/>
 <fld id='lvl' tp='dec' min='0' max='7'/>
 <fld id='mess' tp='str'/>
</comm>
```

Предназначен для передачи команд и действий узлу. Команды могут включать параметры. Параметры описываются тегом <fld>.

Команда инициализируется запросом на модификацию. В качестве аргументов запроса выступают путь к тегу <comm> и ветка этого тега с инициализированными параметрами в тегах <fld>.

Ветки (дочерние узлы)

```
<list id='k_br' dscr='Kernel branches' tp='br' mode='att'>
 <el id='0'>kern1<el/>
 <el id='1'>kern2<el/>
</list>
```

Ветки описываются обычным списком <list> со специальными атрибутами tp='br' и mode='att|at'.

Методика запроса и модификации веток полностью совпадает с методикой работы со списком <list>.

15.2 Иерархические зависимости элементов языка управления

Пример языка управления:

```
<oscada_cntr>
 <area id='a_gen' dscr='Generic control' acs='0440'>
  <fld id='config' dscr='Config file' acs='0660' com='1' tp='str' dest='file'/>
  <fld id='cr_file_perm' dscr='Files' acs='0660' cfg='1' tp='oct' len='3'/>
  <fld id='cr_dir_perm' dscr='Directories' acs='0660' cfg='1' tp='oct' len='3'/>
  <comm id='upd_opt' dscr='Update options(from config)'/>
  <comm id='quit' dscr='Quit'/>
```

```
  </area>
  <area id='a_kern' dscr='Kernels'>
   <list id='k_br' dscr='Kernels' tp='br' mode='att'>
    <el id='0'>kern1<el/>
   </list>
  </area>
</oscada_cntr>
```

Таблица 6. Иерархические зависимости элементов языка:

Тег	**Описание**	**Атрибуты**	**Содержимое**
oscada_cntr	Корневой элемент страницы. Является единственным и служит для идентификации принадлежности к языку интерфейса управления.	*id* — идентификатор; *dscr* — описание.	(area, img)
area	Группировка стандартных тегов.	*id* — идентификатор; *dscr* — описание; *acs* — права доступа; *own* — id владельца; *grp* — id группы.	(area, fld, list, table, comm, img)
comm	Команды узлу.	*id* — идентификатор; *dscr* — описание; *acs* — права доступа; *own* — id владельца; *grp* — id группы.	(fld)
fld	Описание данных стандартных типов.	*id* — идентификатор; *dscr* — описание; *acs* — права доступа; *own* — id владельца; *grp* — id группы; *tp* — тип элемента: *str(len, dest, cols, rows)* — строковый элемент;	Значение элемента.

Тег	Описание	Атрибуты	Содержимое
		dec(len, max, min, dest) — целое число в десятичном представлении; *oct(len, max, min, dest)* — целое число в восьмеричном представлении; *hex(len, max, min, dest)* — целое число в шестнадцатеричном; *real(len, max, min, dest)* — вещественное число; *bool* — логический признак (true|false); *time* — время/дата в секундах (от 01/01/1970). **Связные:** *len* — длина значения (симв.); *min* — минимум значения; *max* — максимум значения; *cols* — количество колонок; *rows* — количество строк; *dest* — способ ввода: *file* — полное имя файла; *dir* — полное имя директории; *select(select)* — выборный тип; *sel_ed(select)* — выборный тип с возможностью редактирования. *select* — путь к скрытому списку;	

Тег	Описание	Атрибуты	Содержимое
list	Список данных стандартных типов.	*id* — идентификатор; *dscr* — описание; *acs* — права доступа; *own* — id владельца; *grp* — id группы; *tp* — как в <fld> кроме: *br(br_pref)* — дочерние узлы. *idm* — индексированный список (0|1); *hide* — скрытый список (0|1); *s_com* — способы модификации списка [add][,ins][,edit][,del]: *add* — добавлять строки; *ins* — вставлять строки. *edit* — модифицировать строки; *del* — удалять строки. **Связные:** *br_pref* — префикс дочерних узлов; *dest* — как в <*fld*>.	(el)
table	Таблица данных стандартных типов.	*id* — идентификатор; *dscr* — описание; *acs* — права доступа; *own* — id владельца; *grp* — id группы;	(list)

Тег	Описание	Атрибуты	Содержимое
		key — ключевые колонки (key=«id,name,per»).	
img	Изображение.	*id* — идентификатор; *dscr* — описание; *acs* — права доступа; *own* — id владельца; *grp* — id группы; *tp* — формат изображения.	
el	Тег строки для списка <*list*>	*id* — номер строки.	Значение строки

15.3 Объект узла динамического дерева (TCntrNode)

Наследуется:	Всеми динамическими и управляемыми объектами, прямо или через потомков.

Данные:

Команды интерфейса управления (enum – TCntrNode::Command):

- *TCntrNode::Info* — запрос информации о элементе/элементах;
- *TCntrNode::Get* — запрос на получение значения элемента;
- *TCntrNode::Set* — запрос на модификацию значения элемента.

Состояния динамического узла (enum TCntrNode::Mode):

- *TCntrNode::MkDisable* — отключение;
- *TCntrNode::Disable* — отключен;
- *TCntrNode::MkEnable* — включение;
- *TCntrNode::Enable* — включен.

Публичные методы:

- *TCntrNode(TCntrNode *prev = NULL);* — Инициализация с указанием родительского узла <*prev*>.
- *void cntrCmd(const string &path, XMLNode *opt, TCntrNode::Command cmd, int lev = 0);* — Команда работа с интерфейсом управления системы. Поддерживаются транспортные переходы по полному пути вида: </sub_Security/usr_root/:gen> где :gen путь к конкретному полю страницы.
- *static XMLNode *ctrId(XMLNode *inf, const string &n_id);* — Получение узла XML по значению атрибута 'id' <*n_id*>. Поддерживаются запросы XML узла по полному пути к нему вида (node1/node2/node3).
- *static string ctrChk(XMLNode *fld, bool fix = false);* — Проверка валидности значения в полей <fld> (Устаревшее).
- *static XMLNode *ctrMkNode(const char *n_nd, XMLNode *nd, int pos, const char *req, const char *path, const string &dscr, int perm=0777, int uid=0, int gid=0, int n_attr=0, ...);* — Добавление элемента управления на страницу. Возможно указания множества

дополнительных атрибутов в количестве <*n_attr*> в виде: <атрибут1>,<значений1>,<атрибут2>,<значений2>,...,

- *static string ctrGetS(XMLNode *fld);* — Получение строки из <*fld*>.
- *static int ctrGetI(XMLNode *fld);* — Получение целого из <*fld*>.
- *static double ctrGetR(XMLNode *fld);* — Получение вещественного из <*fld*>.
- *static bool ctrGetB(XMLNode *fld);* — Получение логического признака из <*fld*>.
- *static void ctrSetS(XMLNode *fld, const string &val, const char *id=NULL);* — *Запись строки* <val> *в поле* <fld> *с идентификатором* <id>//, в случае строки.
- *static void ctrSetI(XMLNode *fld, int val, const char *id=NULL);* — Запись целого <*val*> в поле <*fld*> с идентификатором <*id*>, в случае строки.
- *static void ctrSetR(XMLNode *fld, double val, const char *id=NULL);* — Запись вещественного <*val*> в поле <*fld*> с идентификатором <*id*>, в случае строки.
- *static void ctrSetB(XMLNode *fld, bool val, const char *id=NULL);* — Запись логического признака <*val*> в поле <*fld*> с идентификатором <*id*>, в случае строки.
- *virtual string nodeName();* — Имя узла.
- *string nodePath(char sep = 0);* — Префикс узла. Используется для формирования имени пути в случае наличия более чем одного контейнера у узла.
- *void nodeList(vector<string> &list);* — Список дочерних узлов <*list*> у данного узла.
- *AutoHD<TCntrNode> nodeAt(const string &path, int lev = 0, char sep = 0);* — Подключение к дочернему узлу.
- *TCntrNode *nodePrev();* — Адрес родительского узла.
- *Mode nodeMode();* — Состояние узла.
- *unsigned nodeUse( );* — Количество подключений к узлу.
- *void connect();* — Подключение к узлу (захват ресурса).
- *void disConnect();* — Отключение от узла (освобождение ресурса).

Защищённые методы:

- *virtual void cntrCmd_(const string &path, XMLNode *opt, TCntrNode::Command cmd);* — Функция обслуживания запросов интерфейса управления. Должна переопределяться у потомка.
- *void nodeEn();* — Включение узла.
- *void nodeDis(long tm = 0,int flag = 0);* — Отключение узла с передачей флага.
- *void nodeDelAll( );* — Очистка всех контейнеров с дочерними узлами.
- *void nodePrev(TCntrNode *node);* — Установка родительского узла в <*node*>.
- *unsigned grpSize();* — Количество контейнеров с дочерними узлами.
- *unsigned grpAdd(const string &iid);* — Добавление контейнера дочерних узлов с префиксом <*iid*>. Возвращает идентификатор нового контейнера.
- *void chldList(unsigned igr, vector<string> &list);* — Список дочерних узлов <*list*> в указанном контейнере <*igr*>.
- *bool chldPresent(unsigned igr, const string &name);* — Проверка на присутствие указанного дочернего узла <*name*> в контейнере <*igr*>.
- *void chldAdd(unsigned igr, TCntrNode *node, int pos = -1);* — Добавление дочернего узла <*node*> в контейнер <*igr*>.
- *void chldDel(unsigned igr, const string &name, long tm = -1, int flag = 0);* — Удаление дочернего узла <*name*> из контейнера <*igr*> с флагом <*flag*>.
- *AutoHD<TCntrNode> chldAt(unsigned igr, const string &name, const string &user = "");* — Подключение к дочернему узлу <*name*> контейнера <*igr*> пользователя <*user*>.
- *virtual void preEnable();* — Предупреждение про подключение. Вызывается перед реальным подключением.
- *virtual void postEnable();* — Предупреждение про подключение. Вызывается после реального подключения.

- *virtual void preDisable(int flag);* — Предупреждение про отключение. Вызывается перед реальным отключением.
- *virtual void postDisable(int flag);* — Предупреждение про отключение. Вызывается после реального отключения (перед удалением).

16 XML в системе OpenSCADA (XMLNode)

XML в системе OpenSACDA представлен объектом XML-тега XMLNode.

16.1 XML-тег (XMLNode)

Данные:

Опции функции генерации XML-файла(define):

- *XML_BR_OPEN_PREV* — вставлять конец строки перед тегом открытия;
- *XML_BR_OPEN_PAST* — вставлять конец строки после тега открытия;
- *XML_BR_CLOSE_PAST* — вставлять конец строки после тега закрытия;
- *XML_BR_TEXT_PAST* — вставлять конец строки после текста тега;

Публичные методы:

- *XMLNode(const string &name = "");* — Инициализация тега с именем <*name*>.
- *XMLNode &operator=(XMLNode &prm);* — Копирование ветки XML-дерева из <*prm*>.
- *string name() const;* — Имя тега.
- *void name(const string &s);* — Установка имени тега.
- *string text() const;* — Текст тега.
- *void text(const string &s);* — Установка текста тега <*s*>.
- *void attrList(vector<string> & list) const;* — Список атрибутов в теге <*list*>.
- *string attr(const string &name) const;* — Получение атрибута *name*>.
- *XMLNode* attr(const string &name, const string &val);* — Установка/создание атрибута <*name*> со значением <*val*>.
- *XMLNode* attr_(const char *name, const char *val);* — Установка/создание атрибута <*name*> со значением <*val*>.
- *void load(const string &);* — Загрузка/парсинг XML-файла.
- *string save(unsigned char flgs = 0);* — Сохранение/создание XML-файла с параметрами форматирования <*flgs*>.
- *void clear();* — Очистка тега (рекурсивно и все вложенные).
- *int childSize() const;* — Количество вложенных тегов.
- *void childAdd(XMLNode *);* — Добавление вложенного тега.
- *XMLNode* childAdd(const string &name = "");* — Добавление вложенного тега.
- *int childIns(unsigned id, XMLNode *);* — Вставка вложенного тега в позицию <*id*>.
- *XMLNode* childIns(unsigned id, const string &name = "");* — Вставка вложенного тега с именем <*name*> в позицию <*id*>.
- *void childDel(const unsigned id);* — Удаление вложенного тега <*id*>.
- *void childClean(const string &name = "");* — Очистка вложенного тега <*name*>.
- *XMLNode* childGet(const int) const;* — Получение вложенного тега по порядковому номеру.
- *XMLNode* childGet(const string &name, const int numb = 0) const;* — Получение вложенного <*numb*> порядкового тега по имени тега <*name*>.
- *XMLNode* childGet(const string &attr, const string &name, bool noex = false) const;* — Получение вложенного <numb> порядкового тега по значению <*name*> атрибута <*attr*>. <*noex*> указывает на запрет генерации исклчения в случае отсутствия тега.

17 Ресурсы в системе OpenSCADA (ResAlloc, AutoHD)

Большинство узлов и подсистем системы OpenSCADA являются динамическими, т.е. допускают создание/удаление/конфигурацию в процессе функционирования системы. Учитывая многопоточность системы данная функциональность накладывает жесткие требование к синхронизации потоков. Для синхронизации в системе используются ресурсы функции которых локализованы в объекте <ResAlloc>. В объекте <ResAlloc> реализованы функции как классического захвата/освобождения ресурса так и функции с автоматическим освобождением ресурса. Автоматический ресурс подразумевает создание локального объекта ресурса с автоматическим его освобождением при разрушении (в деструкторе). Использование автоматических ресурсов значительно упрощает работу с ресурсами при использовании исключений.

Любой динамический объект системы наследуется от объекта TCntrNode, который содержит механизм подключения через шаблон AutoHD. Основной функцией шаблона является хранение ссылки на объект и захват ресурса исключающего удаление объекта на момент использования. Шаблон поддерживает копирование ресурса и автоматическое его освобождение в случае разрушения объекта шаблона. Для наглядности доступа к объектам порождённым от TCntrNode шаблон AutoHD поддерживает приведение типов основанное на динамическом приведении типов.

17.1 Объект ресурса (ResAlloc)

Публичные методы:

- *ResAlloc(unsigned id);* — Инициализация автоматически освобождающегося ресурса для ранее выделенного идентификатора <*id*>.
- *ResAlloc(unsigned id, bool write, long tm = 0);* — Инициализация автоматически освобождающегося ресурса для ранее выделенного идентификатора <*id*>. С указанием типа ресурса <*write*> (чтение/запись).
- *void request(bool write = false, long tm = 0);* — Запрос ресурса в указанном режиме <*write*> (чтение/запись).
- *void release( );* — Освобождение ресурса.
- *static unsigned resCreate(unsigned val = 1);* — Создание идентификатора ресурса на количество <*val*>.
- *static void resDelete(unsigned res);* — Освобождение идентификатора ресурса <*res*>.
- *static void resRequestW(unsigned res, long tm = 0);* — Запрос ресурса <*res*> на запись/модификацию.
- *static void resReleaseW(unsigned res);* — Освобождение ресурса <*res*> на запись/модификацию.
- *static void resRequestR(unsigned res, long tm = 0);* — Запрос ресурса <*res*> на чтение.
- *static void resReleaseR(unsigned res);* — Освобождение ресурса <*res*> на чтение.

17.2 Шаблон (AutoHD)

Публичные методы:

- *AutoHD( );* — Инициализация без привязки к объекту.
- *AutoHD(ORes *node, const string &who = "");* — Инициализация с привязкой к объекту <*node*>. Объект должен содержать функцию connect() и disConnect().
- *AutoHD(const AutoHD &hd);* — Копирующий конструктор.
- *template <class ORes1> AutoHD(const AutoHD<ORes1> &hd_s);* — Конструктор приведения типов.

- *ORes &at( ) const;* — Получение объекта за ресурсом.
- *void operator=(const AutoHD &hd);* — Копирование ресурсов.
- *void free( );* — Освобождение ресурса.
- *bool freeStat( ) const;* — Признак «Ресурс свободен».

18 Организация и структуры баз данных компонентов системы

Узлы и подсистемы системы OpenSCADA могут иметь собственные таблицы для хранения своих данных. При этом структура таблиц может быть индивидуальной, определяясь объектом <TConfig>. Узлы и подсистемы должны создавать и конфигурировать объект <TConfig> под свои требования.

18.1 Системные таблицы

Система OpenSCADA имеет две системные таблицы BD и SYS. Таблица BD содержит записи зарегистрированных БД, а таблица SYS содержит данные общесистемных параметров.

Таблица 7. **Структура таблицы общесистемных параметров (SYS).**

Идентификатор параметра <id>	**Значение параметра <val>**
/DemoStation/MessLev	0
/DemoStation/Workdir	/mnt/home/roman/work/OScadaD/share/OpenScada
/DemoStation/UI/QTStarter/StartMod	QTCfg

Таблица 8. **Структура таблицы зарегистрированных БД.**

Идентификатор <ID>	**Тип БД <TYPE>**	**Имя <NAME>**	**Описание <DESCR>**	**Адрес <ADDR>**	**Включать <EN>**
LibBD	MySQL	Библиотека функций		server.diya.org;roman;123456;oscadaUserLibs;;;KOI8-U	1
AnastModel	SQLite	Модель АГЛКС		./DATA/AGLKSModel.db	1
GenDB	MySQL	Основная БД		server.diya.org;roman;123456;oscadaDemoSt;;;KOI8-U	1

18.2 Таблицы подсистемы «Сбор данных»

Контроллеры (источники данных) подсистемы «Сбор данных» хранятся в таблицах своих подсистем с именем DQA_<ModName>. Структуры этих таблиц могут значительно отличаться, однако у всех них присутствуют обязательные поля одинаковые для всех них. Общая структура таблиц контроллеров представлена в таблице 5.

Таблица 9. **Общая структура таблиц контроллеров подсистемы «Сбор данных» (DQA_<ModName>).**

Идентификатор <ID>	**Имя контроллера <NAME>**	**Описание <DESCR>**	**Включать< ENABLE>**	**Запускать <START>**	**Индивидуальные параметры**
AutoDA	Автоматический источник	Сбор данных из активных источников с автоматическим их выявлением.	1	1	...

Также как и таблица контроллеров, таблицы параметров для различных типов источников данных могут значительно отличаться, но также и имеют обязательные поля. Кроме отличия характерного для типа источника данных, таблицы параметров ещё могут быть разными

для различных типов параметров. Общая структура таблицы параметров приведена в таблице 6.

Таблица 10. **Общая структура таблиц параметров подсистемы «Сбор данных».**

Шифр параметра <SHIFR>	**Имя параметра <NAME>**	**Описание параметра <DESCR>**	**Включать <EN>**	**Индивидуальные параметры**
P3	P3	Давление на диафрагме	1	...

18.3 Таблицы подсистемы «Параметры»

Для хранения данных подсистема «Параметры» использует четыре таблицы. Две таблицы для хранения шаблонов параметров (Params_tmpl) и их атрибутов (Params_tmpl_io) и две для хранения параметров логического уровня (Params) и их атрибутов (Params_io):

Таблица 11. **Структура таблицы параметров логического уровня (Params).**

Шифр <SHIFR>	**Имя <NAME>**	**Описание <DESCR>**	**Включать <EN>**	**Режим <MODE>**	**Шаблон <PRM>**
test	Тестовый параметр		1	2	test
F3	F3	Расход через диафрагму	1	2	border

Таблица 12. **Структура таблицы атрибутов параметров логического уровня (Params_io).**

Идентификатор параметра <PRM_ID>	**Идентификатор атрибута ID**	**Значений <VALUE>**
test	stOpen	DAQ.BlockCalc.Anast1to2node.КШ6.open
test	stClose	DAQ.BlockCalc.Anast1to2node.КШ6.st_close
test	tCmd	4

Таблица 13. **Структура таблицы шаблонов параметров логического уровня (Params_tmpl).**

Идентификатор <ID>	**Имя <NAME>**	**Описание <DESCR>**	**Функция объектной модели <FUNC>**
test	Тестовый шаблон		Special.FLibComplex1.digitBlock
border	Граничные условия		DAQ.JavaLikeCalc.lib_TemplFunc.GenBoard

Таблица 14. **Структура таблицы атрибутов шаблонов параметров логического уровня (Params_tmpl_io).**

Идентификатор шаблона <TMPL_ID>	**Идентификатор атрибута <ID>**	**Режим <ATTR_MODE>**	**Доступ <ACCS_MODE>**	**Значений <VALUE>**
test	stOpen	1	2	
test	stClose	1	2	
test	tCmd	0	1	5

18.4 Таблицы подсистемы “Транспорты”

Подсистема “Транспорты” делится на входящие и исходящие транспорты. Для каждого типа транспортов существует своя таблица с собственной структурой. Имена таблиц, соответсвенно: Transport_In и Transport_Out. Таблицы могут дополняться полями характерными для каждого типа транспорта.

Таблица 15. **Структура таблицы входящих транспортов (Transport_in).**

Идентификатор <ID>	**Имя <NAME>**	**Описание <DESCRIPT>**	**Тип <MODULE>**	**Адрес <ADDR>**	**Протокол <PROT>**	**Запускать <START>**	**Индивидуаль поля типов транспортов**
web1	Web 1	Work web transport for proced http requests.	Sockets	TCP::10002:0	HTTP	1	...
tcp1	TCP 1	Test TCP input socket!	Sockets	TCP::10001:1	SelfSystem	1	...

Таблица 16. **Структура таблицы исходящих транспортов (Transport_out).**

Идентификатор <ID>	**Имя <NAME>**	**Описание <DESCRIPT>**	**Тип <MODULE>**	**Адрес <ADDR>**	**Запускать <START>**	**Индивидуальные поля типов транспортов**
tcp_o1	TCP Out 1	Output TCP transport 1	Sockets	TCP::10001	1	...
tcp_o2	TCP Out 2	Output TCP transport 2	Sockets	TCP:127.0.0.1:10001	1	...

18.5 Таблицы подсистемы “Архивы”

Подсистема “Архивы” содержит три таблицы с предустановленными именами:

- Архивы значений: Archive_val;
- Архиваторы значений: Archive_val_proc;
- Архиваторы сообщений: Archive_mess_proc.

Таблицы архиваторов могут дополняться полями характерными для каждого типа архиватора.

Таблица 17. **Структура таблицы архивов значений (Archive_val).**

Идентификатор <ID>	**Имя <NAME>**	**Описание <DESCR>**	**Запускать <START>**	**Тип значений <VTYPE>**	**Периодичность буфера <BPER>**	**Размер буфера <BSIZE>**	**Жесткая сетка буфера <BHGR>**
test	Тестовый архив		1	4	1	100	1
MemInfo_use	Используемая память		1	1	1	100	1

Таблица 18. **Структура таблицы архиваторов значений (Archive_val_proc).**

Идентификатор <ID>	**Тип архиватора <MODUL>**	**Имя <NAME>**	**Описание <DESCR>**	**Запускать <START>**	**Адрес <ADDR>**	**Период значений <V_PER>**	**Период архивирован <A_PER>**
test	BaseArh	Test		1	ARCHIVES/VAL/test/	1	60
OneMinutes	BaseArh	Средний за минуту		1	ARCHIVES/VAL/OneMin/	60	60

Таблица 19. **Структура таблицы архиваторов сообщений (Archive_mess_proc).**

Идентификатор <ID>	**Тип архиватора <MODUL>**	**Имя <NAME>**	**Описание <DESCR>**	**Запускать <START>**	**Шаблон категории сообщений <CATEG>**	**Уровень сообщений <LEVEL>**	**Адрес <ADDR>**
StatErrors	BaseArh	Ошибки станции		1	/DemoStation*	4	ARCHIVE MESS/stEr r/
NetRequsts	BaseArh	Сетевые запросы		1	/DemoStation/Transport/Sockets*	1	ARCHIVE MESS/Net/

18.6 Таблицы подсистемы “Безопасность”

Подсистема “Безопасность” содержит две таблицы: таблица пользователей системы (Security_user) и групп системы (Security_grp).

Таблица 20. **Структура таблицы пользователей системы (Security_user).**

Имя <NAME>	**Описание <DESCR>**	**Индекс <ID>**	**Пароль <PASS>**	**Рабочая группа <GRP>**
root	Суперпользователь	0	openscada	root
user	Пользователь	1		root

Таблица 21. **Структура таблицы групп пользователей системы (Security_grp).**

Имя <NAME>	**Описание <DESCR>**	**Индекс <ID>**	**Пользователи в группе <USERS>**
root	Группа суперпользователей	0	root;user

18.7 Структура баз данных модулей

Каждый модуль может иметь собственные БД для хранения собственных данных. Структура БД модулей может формироваться свободно исходя из внутренних потребностей.

19 API модулей модульных подсистем

Первым шагом при подключении разделяемых (SO – shared object) библиотек является подключение функций инициализации. Эти функции должны быть определены как обычные “C” функции, для исключения искажения имен функций. Обычно это делается следующим образом:

```
//================== CUT ========================
extern "C"
{
  TModule::SAt module( int n_mod )
  {
  //Формирование описателя модуля из списка
  //доступных в библиотеке модулей
  }
  TModule *attach( const TModule::SAt &AtMod, const string &source )
  {
  //Подключить указанный модуль
  }
}
//================== CUT ========================
```

Функции для работы с SO библиотекой:

TModule::SAt module(int n_mod);

Функция предназначена для последовательного опроса информации о модулях содержащихся в SO библиотеке. Параметр <n_mod> указывает на порядковый номер запрашиваемого модуля и должен перебираться начиная с нуля. В случае отсутствия модуля с данным идентификатором функция должна возвращать структуру с именем модуля нулевой длины, что и служит окончанием процесса сканирования.

*TModule *attach(const TModule::SAt &AtMod, const string &source);*

Подключение к указанному модулю.

Практически все функции и данные системы сведены в API-системы описанного в данном документе. Однако, для упрощения и ускорения процесса написания модулей, основные функции переопределяемые в модулях приведены в таблице 22.

Таблица 22. Основные функции использующиеся при создании модулей

Общее API модулей (TModule): Атрибуты
• *string mId;* — Идентификатор модуля. • *string mName;* — Имя модуля. • *string mDescr;* — Описание модуля. • *string mType;* — Тип модуля.

- *string mVers;* — Версия модуля.
- *string mAutor;* — Автор модуля.
- *string mLicense;* — Лицензия модуля.
- *string mSource;* — Источник/происхождение модуля.

Методы

- *virtual void modLoad( );* — Загрузка модуля.
- *virtual void modSave( );* — Сохранение модуля.
- *virtual void modStart( );* — Запуск модуля.
- *virtual void modStop( );* — Останов модуля.
- *virtual void modInfo(vector<string> &list);* — Список доступных элементов информации о модуле. Предусмотрены следующие информационные элементы:
  Modul — идентификатор модуля;

  Name — локализованное имя модуля;

  Type — тип модуля;

  Source — источник модуля (контейнер);

  Version — версия модуля;

  Autors — автора модуля;

  Descript — описание модуля;

  License — лицензия модуля.

- *virtual string modInfo(const string &name);* — Запрос указанного элемента информации.
- *void void postEnable( );* — Подключение модуля к динамическому дереву объектов.
- *void modFuncReg(ExpFunc *func);* — Регистрация экспортируемой функции.

API модулей подсистемы “БД”.

Тип БД (потомок от TTipBD):

- *virtual TBD *openBD(const string &iid);* — Открыть/создать БД.

БД (потомок от TBD):

- *virtual void enable( );* — Включение БД.
- *virtual void disable( );* — Отключение БД.
- *virtual void load( );* — Загрузка БД.
- *virtual void save( );* — Сохранение БД.
- *virtual void sqlReq(const string &req, vector< vector<string> > *tbl = NULL);* — Обслуживание SQL-запросов. Для БД поддерживающих SQL-запросы.
- *virtual TTable *openTable(const string &table, bool create);* — Открыть/создать таблицу.

Таблица (потомок от TTable):

- *virtual bool fieldSeek(int row, TConfig &cfg);* — Последовательное сканирование записей таблицы.
- *virtual void fieldGet(TConfig &cfg);* — Получение указанной записи.
- *virtual void fieldSet(TConfig &cfg);* — Установка указанной записи.
- *virtual void fieldDel(TConfig &cfg);* — Удаление указанной записи.

API модулей подсистемы “Сбор данных”.

Тип контроллера (потомок от TTipDAQ):

- *virtual TController *ContrAttach(const string &name, const string &daq_db);* — Открытие/подключение контроллера.

Контроллер (потомок от TController):

- *virtual void load( );* — Загрузка контроллера.
- *virtual void save( );* — Сохранение контроллера.
- *virtual void start( );* — Запуск контроллера.
- *virtual void stop( );* — Останов контроллера.
- *virtual void enable_( );* — Включение контроллера.
- *virtual void disable_( );* — Отключение контроллера.
- *virtual TParamContr *ParamAttach(const string &name, int type);* — Создание/открытие нового параметра.

Параметр контроллера (потомок от TParamContr):

- *virtual void enable( );* — Включить параметр.
- *virtual void disable( );* — Отключить параметр.
- *virtual void load( );* — Загрузка параметра.
- *virtual void save( );* — Сохранение параметра.

API модулей подсистемы «Архивы».

Тип архиватора (потомок от TTipArchivator):

- *virtual TMArchivator *AMess(const string &iid, const string &idb);* — Создание архиватора сообщений.
- *virtual TVArchivator *AVal(const string &iid, const string &idb);* — Создание архиватора значений.

Архиватор сообщений (потомок от TMArchivator):

- *virtual void load( );* — Загрузка архиватора.
- *virtual void save( );* — Сохранение архиватора.
- *virtual void put(vector<TMess::SRec> &mess);* — Передать сообщение архиватору.
- *virtual void get(time_t b_tm, time_t e_tm, vector<TMess::SRec> &mess, const string &category = "", char level = 0, const string &arch = "");* — Запросить сообщение

из архиватора.

Архиватор значений (потомок от TVArchivator):

- *virtual void valPeriod(double iper);* — Установить периодичность значений архиватора.
- *virtual void archPeriod(int iper);* — Установить периодичность архивирования.
- *virtual void load( );* — Загрузить архиватор.
- *virtual void save( );* — Сохранить архиватор.
- *virtual void start( );* — Запустить архиватор.
- *virtual void stop(bool full_del = false);* — Остановить архиватор, с возможностью полного удаления если установлен <*full_del*>.
- *virtual TVArchEl *getArchEl(TVArchive &arch);* — Получение архива <*arch*> обслуживаемого архиватором.

Архивный элемент значений (потомок от TVArchEl):

- *virtual void fullErase( );* — Полное удаление части архива в архиваторе.
- *virtual long long end( );* — Время окончания архива в архиваторе.
- *virtual long long begin( );* — Время начала архива в архиваторе.
- *virtual void getVal(TValBuf &buf, long long beg = 0, long long end = 0);* — Получение кадра значений <*buf*> за время от <*beg*> и до <*end*> архива в архиваторе.
- *virtual string getS(long long *tm, bool up_ord);* — Получение строкового значения во время <*tm*> с притягиванием к верху <*up_ord*> из архива в архиваторе.
- *virtual double getR(long long *tm, bool up_ord);* — Получение вещественного значения во время <*tm*> с притягиванием к верху <*up_ord*> из архива в архиваторе.
- *virtual int getI(long long *tm, bool up_ord);* — Получение целого значения во время <*tm*> с притягиванием к верху <*up_ord*> из архива в архиваторе.
- *virtual char getB(long long *tm, bool up_ord);* — Получение логического значения во время <*tm*> с притягиванием к верху <*up_ord*> из архива в архиваторе.
- *virtual void setVal(TValBuf &buf, long long beg = 0, long long end = 0);* — Установка кадра значений <*buf*> за время от <*beg*> и до <*end*> в архив в архиваторе.

API модулей подсистемы «Протоколы».

Протокол (потомок от TProtocol):

- *virtual TProtocolIn *in_open(const string &name)* — Открыть/создать входной протокол.

Входной протокол (потомок от TProtocolIn):

- *virtual bool mess(const string &request, string &answer, const string &sender);* — Передача неструктурированного сообщения в протокол.

API модулей подсистемы «Транспорты».

Тип транспорта (потомок от TTipTransport):

- *virtual TTransportIn *In(const string &name, const string &idb);* — Создать/открыть новый «входящий» транспорт.
- *virtual TTransportOut *Out(const string &name, const string &idb);* — Создать/открыть

новый «исходящий» транспорт.

Входящий транспорт (потомок от TTransportIn):

- *virtual void start( );* — Запуск транспорта.
- *virtual void stop( );* — Останов транспорта.

Исходящий транспорт (потомок от TTransportOut):

- *virtual void start( );* — Запуск транспорта.
- *virtual void stop( );* — Останов транспорта.
- *virtual int messIO(const char *obuf, int len_ob, char *ibuf = NULL, int len_ib = 0, int time = 0);* — Передать запрос через транспорт. Возможно указание таймаута.

API модулей подсистемы «Пользовательские интерфейсы».

Пользовательский интерфейс (потомок от TUI):

Не содержит специфических функций!

API модулей подсистемы «Специальные”.

Специальные (потомок от TSpecial):

Не содержит специфических функций!

20 Отладка и тестирование проекта OpenSCADA

Для контроля за качеством кода и проверки работоспособности различных участков системы пишутся специальные модули выполняющие процедуру тестирования с выдачей протокола тестирования. Данные модули необходимо выполнять после завершения работы над любым участком проекта.

21 Правила оформления и комментирования исходных текстов OpenSCADA и его модулей

При написании и оформлении исходных текстов системы OpenSCADA и его модулей необходимо придерживаться следующих правил:

- отступ между уровнями вложений: 4 символа;
- Фигурные скобки открытия и закрытия должны располагаться в отдельных строках на уровне предыдущего текста;
- возможно написание вложений в одной строке с предыдущим уровнем вложения, в случае повышения читабельности кода;
- растояние между описаниями функций не менее одного символа;
- растояние между определением переменных и текстом программы не менее одного символа;
- допускается определение переменных в тексте при сохранении читабельности;
- избегать длины строки более 100 символов;
- команды препроцессора располагать на первом уровне вне зависимости от текущего уровня текста;
- для форматирования исходного текста наследованного у других свободных приложений и примеров рекомендуется использовать утилиту:

indent -bli0 -i4 -l100 -npsl -npcs -prs -nsaf -nsai -ts8 <filename>.

Правила комментирования исходных текстов OpenSCADA:

- обязательному комментированию и тщательному описания подлежат объявления классов;
- объявления публичных методов классов должны быть тщательно описаны с индивидуальным описанием каждого параметра;
- объявления публичных атрибутов также необходимо тщательно комментировать;
- текст функций не нуждаются в тщательном комментировании, однако тонкие и неявные места желательно комментировать.

22 Условные обозначения по тексту и в исходниках

???? — сомнение в целесообразности данного участка;

?!?! — участок не полностью реализован;

!!!! — тонкое место, участок требует переосмысления.