26.51.70.190

УТВЕРЖДЕН

АДИГ.421457.005 РЭ5-ЛУ

КОМПЛЕКСЫ ПРОГРАММНО-ТЕХНИЧЕСКИЕ «СУРА»

Руководство по эксплуатации

Принципы проектирования

АДИГ.421457.005 РЭ5

Вер.3.3

Листов 38

2021

Литера O_1

Содержание

Содержание данного документа может изменяться без предварительного уведомления.

В связи с непрерывной работой по улучшению характеристик изделия программное обеспечение может также меняться. Тем не менее, содержание документации регулярно отслеживается и все корректировки вносятся в последующие издания.

Настоящее руководство содержит сведения о принципах создания проектов АСУ ТП на базе программно-технического комплекса «СУРА», используя инструментальные средства из состава ПО верхнего уровня ПТК «СУРА» (менеджер приложений «СФЕРА», в дальнейшем по тексту – **СФЕРА**).

Перед началом проектирования необходимо также ознакомиться с документами «Комплексы программно-технические «СУРА». Контроллеры программируемые Эликонт-100. Руководство по эксплуатации. Часть 1. Состав и функциональные возможности. АДИГ.421457.004 РЭ», «Комплексы программно-технические «СУРА». Руководство по эксплуатации. АДИГ.421457.005 РЭ», «Комплексы программно-технические «СУРА». Руководство по эксплуатации. Администрирование проектов АСУ ТП. АДИГ.421457.005 РЭ1».

1 Задачи проектирования

1.1 Суть проектирования

Программно-технический комплекс «СУРА» (в дальнейшем по тексту - СУРА) – это не готовая АСУ ТП, а лишь набор универсальных средств, на основе которых АСУ ТП может быть построена. Чтобы автоматизировать конкретный объект на базе универсальных аппаратных и программных решений СУРЫ, необходимо создать проект (рисунок 1).

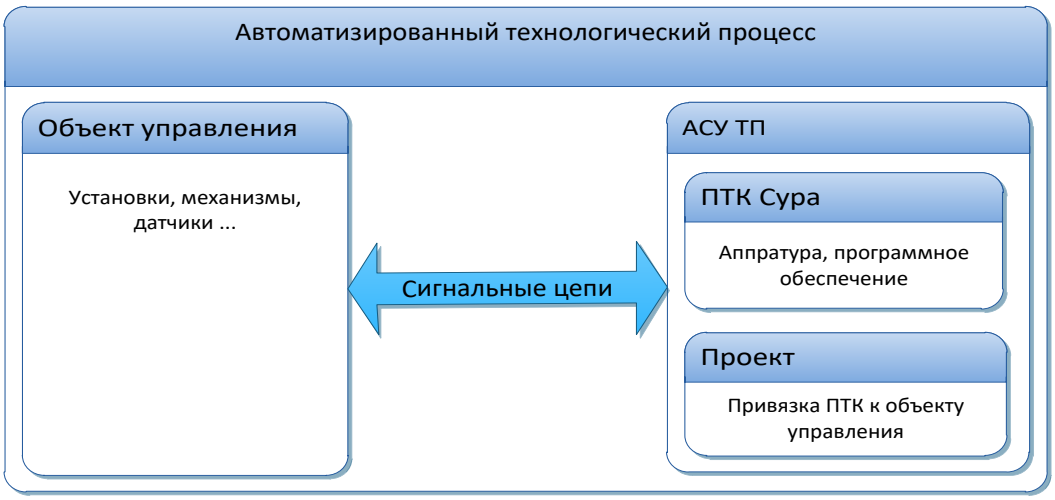

Рисунок 1 - Суть проектирования

Проект содержит большое количество разнородной информации, для формирования которой нужна команда пользователей, обладающих различной квалификацией: технологи, программисты, дизайнеры, системные администраторы. Успех разработки проекта АСУ ТП во многом зависит от согласованности действий и четкого понимания основных задач проектирования.

Примечание. Отметим, что слово «**объект**» будет использоваться в двух разных смыслах:

– Объект управления (например, энергоблок).
– Просто Объект (например, задвижка), как элемент проекта СУРЫ.

1.2 Перечень задач

В самом общем виде, можно выделить следующие задачи проектирования, решаемые средствами СУРЫ.

Таблица 1 – Перечень задач проектирования

Задача	Пояснения
Описание объектов	Ввод информации обо всех механизмах, датчиках и др. элементах, с которыми взаимодействует АСУ ТП
Описание аппаратуры	Ввод информации об используемых аппаратных компонентах СУРЫ: контроллерах, модулях УСО, станциях верхнего уровня и др.
Распределение сигналов	Описание связей сигнальных цепей с каналами модулей УСО
Технологическое программирование	Включает в себя программирование контроллера и расчетного сервера
Технологическое моделирование	Разработка моделей объекта управления в целом или его элементов для отладки АСУ ТП и создания тренажеров
Подготовка видеоизображений	Проектирование операторского интерфейса
Обеспечение безопасности	Регистрация пользователей, назначение прав доступа и другие меры

Конечно, есть много других задач, например, задача интеграции с другими АСУ, но они являются более специфическими и будут рассматриваться в соответствующих разделах.

Перечисленные задачи не являются независимыми; на схеме (рисунок 2) зависимости обозначены стрелками:

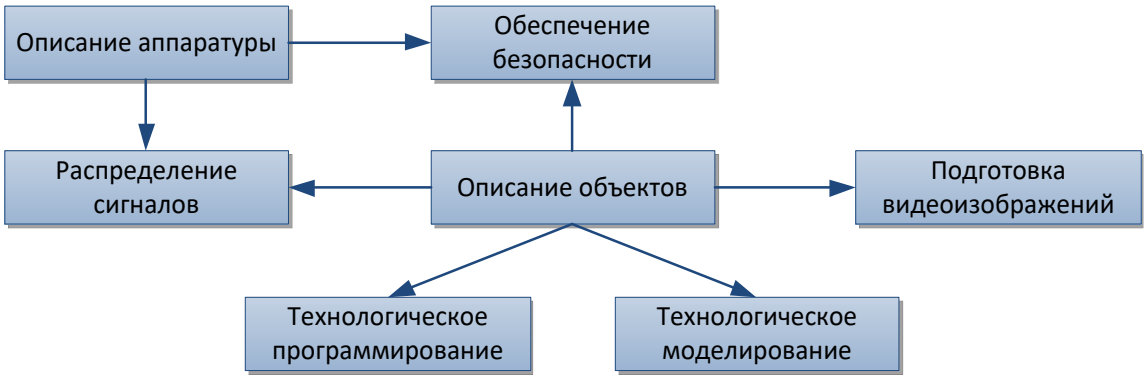

Рисунок 2 – Схема зависимости задач проектирования

Одним из ориентиров при создании СУРЫ было уменьшение количества таких зависимостей.

1.3 Последовательность действий

В данном разделе описывается подход к проектированию, рекомендуемый в СУРЕ. Мы предлагаем определенную последовательность шагов. Это не означает, что эти шаги нужно пройти один раз с начала до конца. Во-первых, реальный процесс носит итеративный характер: вы будете возвращаться к однажды пройденным шагам и вносить необходимую коррекцию. Во-вторых, шаги могут распараллеливаться между несколькими проектантами.

1.3.1 Создание проекта

Мы предполагаем, что СУРА уже установлена и настроена. Запустите **СФЕРУ** и выберите локальную среду выполнения (это облегчит дальнейшую отладку).

При создании проекта рекомендуем выбирать опцию «**Создать пустой проект**». В СУРЕ нет необходимости копировать многократно используемые конструкции из проекта в проект – для этого есть более удобный способ: подключение проектов.

Если у вас есть готовые библиотеки, подключите их к созданному проекту. В любом случае к проекту будет автоматически подключена «**Библиотека Библио**», содержащая большое количество типовых изображений и макросов.

1.3.2 Ввод объектов

В СУРЕ информацию об объектах можно не только набивать вручную, но и импортировать из таблиц Microsoft Excel (у заказчика обычно уже имеется готовая информация). Если импорт из однотипных таблиц производится многократно, имеет смысл применять сохраненные настройки импорта.

Если в импортируемых таблицах описаны важные свойства объектов, не укладывающиеся в стандартные поля СУРЫ, можно определить в СУРЕ **дополнительные атрибуты** и импортировать в них значения данных свойств.

Вводимые или импортируемые объекты удобно сразу группировать по узлам подобно тому, как файлы группируются по папкам.

1.3.3 Объединение простых объектов в составные

Этот шаг не обязателен, но он может значительно сократить объем последующих рутинных операций.

Часто в совокупности объектов прослеживаются однотипные связки, состоящие из нескольких объектов. Подобные связки могут не только единообразно представляться в операторском интерфейсе, но и иметь одинаковый алгоритм обработки в контроллере и модели. В этом случае рекомендуется описать такую связку как **составной тип**, затем для каждой связки

добавить экземпляр составного типа и указать, из каких простых объектов он состоит. Если предполагается только общность операторского интерфейса, составной тип следует объявить, как **слабый**, а если предполагается также общность алгоритмов – как **сильный**.

Если подобный составной тип уже встречался в других проектах, имеет смысл не описывать его в каждом проекте, а внести в библиотеку, которую можно будет подключать к проектам.

1.3.4 Описание аппаратуры и распределение сигналов

После того как в **Администраторе БД** добавлен один или несколько контроллеров, в **Базисе** можно быстро распределить объекты по этим контроллерам. Такая процедура называется **полупривязкой**, поскольку она задает для каждого объекта только контроллер, в который будут поступать сигналы от данного объекта, но не алгоблок, который будет обрабатывать эти сигналы. Поддерживается групповая полупривязка.

Хотя объекты только полупривязаны, их сигналы уже можно распределить по каналам ввода-вывода в **Полисе**. Для этого надо описать **дерево устройств ввода-вывода (УВВ)** и связать сигналы с каналами путем перетаскивания. Поддерживается групповое связывание с автоматическим распределением сигналов по свободным каналам.

Важно, что процедура распределения является сравнительно быстрой, и ее можно выполнять до составления технологических программ. Таким образом, можно на раннем этапе проектирования оценить необходимый объем аппаратуры.

1.3.5 Технологическое программирование и моделирование

В СУРЕ создание управляющих и моделирующих технологических программ выполняется единообразно, в одной среде – **Полисе**. Вместе с тем, эти задачи независимы и могут выполняться разными пользователями в разное время.

Задача моделирования не является обязательной, если в проекте не предусмотрены тренажеры. Но мы рекомендуем выполнять моделирование хотя бы на уровне элементарных объектов. Это потребует минимальных трудозатрат, но облегчит в дальнейшем наладку АСУ ТП.

После выполнения шага 4 объекты уже распределены по контроллерам (хотя и не связаны с технологической программой). Чтобы добавить управляющий алгоблок и связать его с объектом, достаточно в **Полисе** выполнить одну операцию перетаскивания. Чтобы добавить моделирующий алгоблок, потребуется вторая операция перетаскивания. Проще всего делать эти операции сразу, одну за другой.

1.3.6 Подготовка видеоизображений

Этот шаг можно выполнять сразу после шага 3, независимо от шагов 4 и 5. Эта задача решается средствами **Контура** с рядом возможностей:

- возможность использования данных из подключённых проектов;
- возможность привязки аниматоров и рецепторов к модели;
- прочие возможности, описанные в документе по Контуру.

Напомним, что **Контур** поддерживает создание типовых изображений для составных объектных типов, поэтому группировка объектов на шаге 3 существенно сократит шаг 6.

1.3.7 Обеспечение безопасности

Безопасность работы АСУ ТП достигается комплексом мероприятий, выполняемых проектантами АСУ ТП и системными администраторами предприятия. Этот шаг поставлен в конец не из-за малой значимости. Дело в том, что многие вопросы безопасности могут решаться только на предприятии заказчика, в то время как предыдущие шаги могут выполняться географически в любом месте.

Тем не менее, некоторые вопросы безопасности можно и нужно решать на ранней стадии проектирования. К ним относятся: настройка доступа к БД проекта, обеспечение резервного копирования проекта, описание групп пользователей АСУ ТП, назначение прав этим группам и т.д.

2 Проект

2.1 Понятие проекта

Проект – это совокупность вводимой пользователем информации, описывающей конкретную АСУ ТП на базе СУРЫ. В состав проекта входят, в частности:

– все технологические программы, включая расчетные задачи и модели объекта;

– все изображения для операторских станций;

– описание абонентов сети и пользователей, участвующих в работе АСУ ТП.

Проектант – это роль пользователя, обязанностью которой является разработка проекта. Допускается одновременная работа нескольких проектантов над одним проектом.

Проект является базой данных клиент-серверного типа. Это означает, что приложения СУРЫ **Сервер БД проекта**. Эта служба обеспечивает контроль прав доступа и блокирует одновременное редактирование одного элемента проекта несколькими пользователями.

2.2 Открытие существующего проекта

Один проектант может участвовать в разработке нескольких проектов, но в каждый конкретный момент времени он работает с одним **текущим проектом**. Путь к этому проекту отображается в верхней панели СУРЫ. Здесь же можно сменить текущий проект, выбрав нужный элемент из выпадающего списка. При смене текущего проекта потребуется закрыть все запущенные из СУРЫ приложения.

Если требуемый проект отсутствует в выпадающем списке, его необходимо туда добавить. Для этого нажмите кнопку «…» справа от списка, затем в диалоговом окне нажмите кнопку Добавить. В открывшемся окне мастера выберите опцию **Открыть существующий проект**, укажите имя компьютера-сервера, выберите нужный проект и нажмите кнопку **Готово**. Требуемый проект будет добавлен в список.

В некоторых случаях может возникнуть несоответствие между установленной версией ПО СУРЫ и форматом хранящихся в проекте данных. Если версия ПО более новая, чем версия формата проекта, СУРА предложит обновить формат проекта (при запуске **Администратора БД**). Если версия ПО – более старая, обновление проекта невозможно: вместо этого нужно обновить версию ПО СУРЫ.

Внимание! Обновление версии проекта – необратимая операция. После обновления проекта с ним не смогут работать пользователи, у которых установлено более старое ПО СУРЫ. Поэтому перед обновлением всегда создавайте архивную копию проекта.

2.3 Создание проекта

Для создания нового проекта используется мастер баз данных. Для его вызова нажмите кнопку  , сверху, в строке с адресом текущего проекта **СФЕРЫ**, затем в открывшемся диалоговом окне нажмите кнопку **Добавить**. Далее необходимо добавить сервер, в основном окне щёлкнуть правой кнопкой мыши по добавленному серверу и выбрать создать проект.

2.3.1 Выбор расположения проекта

Если с проектом будет работать только один пользователь, можно установить службу «Сервер БД проекта» на локальном компьютере и хранить проект на диске этого компьютера. При многопользовательской работе с проектом рекомендуется размещать проект на одном из серверов локальной сети (например, на контроллере домена). В любом случае служба «Сервер БД проекта» должна быть установлена и запущена на том компьютере (или компьютерах), где хранятся проекты.

Каждый сервер проекта хранит все проекты в подпапках одной корневой папки, называемой **Основным хранилищем**. По умолчанию, основное хранилище размещается по пути: **C:\ProgramData\Sura\DbServer**, но этот путь можно изменить.

Необходимо учитывать, что жесткие диски не являются абсолютно надежными устройствами, поэтому рекомендуем в качестве хранилища проектов использовать не одиночный жесткий диск, а RAID-массив (хотя бы простейшее зеркало из двух дисков – RAID 1).

2.3.2 Новый проект или резервная копия существующего?

СУРА позволяет подключать к проекту другие проекты, более того, к любому проекту автоматически подключается проект **Библио** (входит в дистрибутив фирменного программного обеспечения), содержащий большое количество типовых изображений и макросов. Поэтому сделать из пустого проекта «работающее нечто» можно менее чем за час.

В некоторых случаях рекомендуется использовать резервную копию существующего проекта, например:

– объект управления, для которого создается проект, технологически похож на другой объект, для которого проект уже разработан и отлажен;

– создается временная копия проекта для проведения тестов, апробирования новых решений и т.п.;

– проект переносится из одной локальной сети в другую, например, из проектной организации на предприятие заказчика.

В остальных случаях рекомендуется создавать новый (пустой) проект.

2.3.3 Задание прав доступа

При создании нового проекта его автор (пользователь создавший проект) автоматически получает роль Администратора и, тем самым, получает все права на этот проект. Другие пользователи не будут иметь доступа к созданному проекту, пока автор не задаст для них роли. Задавать роли можно (и рекомендуется) не для отдельных пользователей, а для доменных групп.

Если для всех проектов на данном сервере предполагается одинаковое распределение прав, можно поступить проще: распределить роли пользователей для сервера в целом, а для созданного проекта указать флажок «**Наследовать права**».

2.3.4 Задание доменного имени

В интегрированной АСУ ТП в одной сети доступны данные, относящиеся к разным проектам. Технически невозможно добиться того, чтобы имена контроллеров, объектов, типов были бы уникальны глобально по всем проектам. Чтобы обеспечить уникальную идентификацию, каждому проекту присваивается **доменное имя**. Проектантам достаточно проследить, чтобы все участвующие в интегрированной АСУ ТП проекты имели различные доменные имена. Теперь, если марки двух объектов из двух разных проектов совпадут, конфликта не будет, поскольку в идентификации объекта участвуют также доменные имена проектов.

Доменное имя может состоять из одного или нескольких фрагментов, разделенных точками. Каждый фрагмент должен начинаться с буквы и может содержать только латинские буквы, цифры и символ тире. Если на предприятии заказчика каждый проект эксплуатируется в своем Windows-домене, можно рекомендовать использование имени Windows-домена в качестве доменного имени проекта.

2.3.5 Добавление начальных данных

Чтобы с пустым проектом можно было начать работать, обычно нужно выполнить ряд типовых операций в **Администраторе БД**, в том числе:

– добавить в проект текущего пользователя и дать ему нужные права;

– добавить в проект текущий компьютер;

– добавить корневой узел.

Мастер баз данных позволяет выполнить эти операции сразу при создании проекта.

2.4 Конструктор

Как уже говорилось, проектные данные обычно расположены не на локальном компьютере, а на сервере, для того чтобы все проектанты могли редактировать проект. Такой подход удобен на этапе проектирования, но не годится для оперативных приложений (терминал, архивной станции и т.д.). Дело в том, что получение проектных данных по сети может занять недопустимое для

оперативных приложений время (более 1 секунды). Более того, при неработоспособности сервера проекта ни одно оперативное приложение нельзя будет запустить.

Поэтому каждое оперативное приложение получает проектные данные из **dat-файла** – специального файла с расширением dat, расположенного на локальном диске и содержащего проектные данные в упакованном виде. При работе с dat-файлом проектные данные доступны только для чтения.

Dat-файл находится специальной папке, называемой **нишей проекта**. Ниша предназначена для хранения связанных с определенным проектом файлов на локальном диске. Ниша создается автоматически для каждого проекта, добавляемого в список проектов **СФЕРЫ**.

Изначально dat-файл в нише отсутствует. Для его создания нужно запустить **Конструктор** – инструмент, позволяющий упаковать проектные данные в файл. Есть две формы запуска конфигуратор: явная и неявная. Различия между ними приведены в таблице 2.

Таблица 2 – Формы запуска Конструктора

Форма запуска	Как запускается	Сравнение возможностей	Когда используется
Неявная	Автоматически при запуске любого оперативного приложения из **СФЕРЫ**	Может создавать dat-файл только для локального компьютера	При проектировании
Явная	Из дерева приложений **СФЕРЫ** (пункт **Утилиты – Конструктор**)	Может создавать dat-файл для любого описанного в проекте компьютера и копировать его на тот компьютер	При эксплуатации АСУ ТП

2.5 Структурирование проекта

2.5.1 Узлы

Как правило, проект содержит описания многих тысяч объектов. Чтобы в них легче было ориентироваться, можно распределять объекты по узлам (аналогично тому, как файлы распределяются по папкам). **Узел** – это описываемая в проекте сущность, позволяющая иерархически классифицировать объекты и другие элементы проекта.

Один узел может быть дочерним по отношению к другому, родительскому узлу. Тем самым, узлы образуют древовидную иерархию. Допускается наличие нескольких корневых (т.е. не имеющих родителя) узлов. Узлы описываются в приложении **Администратор БД**, форма **Узлы**.

Кроме организационной функции, узлы выполняют ряд других важных функций:

– наследование прав доступа. Задать право доступа (например, право оперативного управления) можно индивидуально для объекта, а можно для узла в целом;

– групповая сигнализация. В операторской станции для каждого узла выполняется подсчет общего количества ошибок по всем объектам, принадлежащим этому узлу и его потомкам. Это позволяет выводить индикаторы ошибок по узлам (обычно такие индикаторы размещаются в верхней панели);

– переопределение атрибутов ошибок и событий. Заданные разработчиками СУРЫ тексты и другие атрибуты ошибок и событий могут быть переопределены проектантом. Это переопределение можно сделать индивидуально для объекта, а можно для узла в целом;

– фильтрация проектных данных при формировании dat-файла – см. в следующем разделе.

2.5.2 Срезы

В небольшой АСУ ТП всем станциям верхнего уровня обычно доступна вся проектная информация. Однако в крупных системах бывает необходимо ограничить информационное пространство, которое доступно оперативным приложениям СУРЫ. Приведем несколько примеров:

– если создается единая база данных для нескольких энергоблоков, то на операторских станциях одного энергоблока не должна быть «видна» информация, относящаяся к другим энергоблокам. В то же время, некоторым станциям может понадобиться информация (возможно, не

вся), принадлежащая нескольким энергоблокам, - например, это может быть общая архивная станция, операторская станция центрального щита или терминал руководителя;

– иногда даже в пределах одного агрегата бывает полезно скрыть часть информации, которая не нужна оператору на определенном рабочем месте;

– если требуется экспорт информации из СУРЫ в другие системы (например, с помощью OPC сервера СУРЫ), необходимо четко ограничить объем экспортируемых данных из соображений безопасности и производительности.

В приложениях, используемых на этапе проектирования, всегда отображаются все элементы проекта. Но при формировании dat-файла может производиться фильтрация проектных данных. Для задания этой фильтрации в **Администраторе БД** вводятся **срезы**. Каждый срез имеет уникальный номер (от 1 до 32) и уникальное название.

Далее, для таких элементов проекта как объекты, узлы, абоненты сети и др. можно задать принадлежность к срезам. Обычно это задается в том же приложении, где описывается элемент. Один элемент проекта может входить сразу в несколько срезов (а может ни в один срез не входить). По умолчанию, элемент входит во все срезы.

Кроме того, для каждого описанного в проекте компьютера задается набор **рабочих срезов**. Когда для этого компьютера запускается конфигуратор, он исключает из результирующего dat-файла те элементы, которые не входят ни в один из рабочих срезов для этого компьютера.

Задача. В проекте имеются объекты Датчик1 и Датчик2. Показания первого датчика должны публиковаться для сторонних систем с помощью OPC сервера, запускаемого на компьютере OS1. Показания второго датчика не должны публиковаться. На всех остальных станциях верхнего уровня должны быть доступны оба датчика.

Решение. Создаем в **Администраторе БД** два среза под названием «Внутренние» и «Публикация». В форме Абоненты сети для компьютера OS1 в списке рабочих срезов оставляем только срез «Публикация». Открываем в **Базисе** форму **Объекты** и закладку **Флаги и срезы**. Для объекта Датчик2 в списке срезов оставляем только срез «Внутренние». В результате Датчик2 будет недоступен на компьютере OS1, потому что он не входит в его рабочий срез.

Задавать принадлежность срезам индивидуально для каждого объекта было бы очень трудоемкой задачей. Ее можно упростить, воспользовавшись имеющимся в СУРЕ механизмом исключения: при исключении элемента из среза все его дочерние элементы автоматически исключаются из этого среза. Например, если узел исключить из среза «Публикация», то все объекты этого узла не попадут в dat-файл для компьютера OS1.

2.6 Подключённые проекты

2.6.1 Назначение

При внедрении СУРЫ на предприятии объектом автоматизации является, как правило, отдельный энергоблок или более мелкая подсистема. В дальнейшем обычно возникает задача информационной интеграции отдельных АСУ ТП, выполненных на СУРЕ.

Самым простым решением было бы объединение проектных данных по всем подсистемам в одном проекте. Однако это решение может быть неприемлемым по следующим причинам:

– модернизация подсистем производится в разное время. Если одна из подсистем находится в работе, модификация ее проекта может быть недопустимой;

– проекты подсистем могут создаваться разными группами проектантов в разных организациях;

– подсистемы могут использовать разные версии СУРЫ.

С учетом этих факторов в СУРЕ была разработана технология **подключения проектов**. Вкратце: проектант может подключить к текущему проекту любое число других проектов, созданных в СУРЕ или в более ранних версиях СУРЫ. После этого он может использовать данные подключенных проектов, но только для чтения. Примеры:

– на мнемосхеме текущего проекта можно разместить мнемосимвол и привязать его к марке из подключённого проекта;

– в технологической программе текущего проекта можно использовать макросы из подключённого проекта;

– в таблицу или график станции анализа можно добавлять марки из подключённых проектов.

– Технология подключённых проектов имеет несколько применений:

– объединение нескольких АСУ ТП в единое информационное пространство;

– создание проектантом библиотеки типовых решений (алгоритмы, изображения и т.д.). Библиотека создаётся в виде проекта, который затем подключается к разным рабочим проектам;

– разработка фирменной библиотеки, входящей в дистрибутив СУРЫ и автоматически подключаемой ко всем проектам.

Если проект **B** подключён к проекту **A**, будем говорить, что проект **B** экспортирует проектные данные, а проект **A** – импортирует.

2.6.2 Принципы

– Проект идентифицируется сетевым путем, например: \\SRV1.poligon.local\проекты\ТЭЦ.

– К текущему проекту может быть подключено любое количество других проектов, но все они должны быть различны (иметь разные пути).

– Проект не может быть подключён сам к себе.

– Подключение не симметрично: если проект **B** подключён к проекту **A**, то автоматически это не означает, что **A** подключён к **B**. Если нужна двусторонняя связь, следует дополнительно подключить **A** к **B**.

– Подключение не транзитивно: если **B** подключён к **A**, а **C** подключён к **B**, это не означает, что **C** подключён к **A**. Если нужно использовать данные из проекта **C** в проекте **A**, следует явно подключить **C** к **A**.

– Вся проектная информация, импортированная из подключённых проектов, хранится в упакованном виде в текущем проекте. Это обеспечивает постоянную доступность этой информации. Импорт производится только по специальной команде из «**Администратора БД**». Нигде более обращений к базам данных подключённых проектов не производится.

Каждому подключённому проекту следует назначить понятный пользователю псевдоним, по которому подключённый проект будет идентифицироваться в пользовательском интерфейсе.

2.6.3 Управление подключёнными проектами

Управление подключёнными проектами производится в приложении «**Администратор БД**», форма «**Подключенные проекты**».

Внимание! Операции, производимые в этой форме, требуют высокого уровня ответственности и могут привести к неработоспособности текущего проекта. Обязательно сделайте резервную копию текущего проекта перед их выполнением.

Подключённые проекты представлены в форме в виде дерева. Узел уровня 0 представляет текущий проект, узлы уровня 1 - подключённые проекты. Узлы следующих уровней отображают проекты, подключённые неявно, т.е. которые подключены к подключённым проектам.

Зеленым значком помечается текущий проект. Желтый значок узла означает, что данные из подключённого проекта были успешно импортированы, красный - что эти данные отсутствуют.

Если узел встречается в дереве повторно, то он помечается значком со стрелкой и его подузлы не раскрываются.

В тексте узла выводится псевдоним и путь каждого проекта. Кроме того, для выбранного в дереве проекта можно увидеть дату последнего импорта и служебные идентификаторы.

2.6.3.1 Добавление

Для подключения нового проекта нажмите кнопку  . В диалоговом окне укажите:

– путь к БД подключаемого проекта. Рекомендуется всегда указывать путь, начинающийся с полного сетевого имени сервера: \\ИмяСервера.ИмяДомена.local\;

– псевдоним подключаемого проекта. Нужно придумать псевдоним достаточно краткий и понятный пользователю, например, ЭБ1;

– способ соединения рабочих станций с контроллерами из подключаемого проекта: напрямую или через прокси-сервер. **Внимание!** Соединение напрямую будет увеличивать нагрузку на контроллеры и может превысить лимит числа сеансов контроллера. Поэтому рекомендуется использовать соединение через прокси, кроме того случая, когда суммарное количество контроллеров и абонентов во всех проектах достаточно мало и требуется одинаково высокая скорость коммуникаций в рамках объединенной АСУ ТП;

– принимать или нет ошибки от контроллеров из подключаемого проекта. Имеются в виду текущие приборные и технологические ошибки, отображаемые в списках на операторской станции. Приём ошибок из подключённых проектов может привести к неоправданному «раздуванию» списка ошибок.

2.6.3.2 Изменение свойств

Указанные в предыдущем разделе свойства подключённого проекта могут всегда быть скорректированы. Выберите в дереве один из подключённых проектов и нажмите кнопку ▲. В результате будет выдано такое же диалоговое окно свойств.

2.6.3.3 Удаление

Для удаления (точнее, отключения) подключённого проекта выберите его в дереве и нажмите кнопку ▬. В результате из текущего проекта будет удалена ссылка на выбранный проект, а также импортированные из него данные. На исходную БД подключённого проекта эта операция никакого влияния не окажет.

Перед отключением будет автоматически выполнен поиск ссылок из текущего проекта на данные из подключённого проекта. Если ссылки есть, будет выдано дополнительное предупреждение.

Внимание! Операция отключения необратима. Перед отключением обязательно создайте резервную копию текущего проекта, а после отключения выполните диагностику проекта.

2.6.3.4 Импорт

Импорт данных из подключённого проекта – это единственная операция, требующая доступа к исходной БД подключённого проекта. Процедура импорта схожа с конфигуратором – результатом являются упакованные проектные данные – но эти данные сохраняются не в dat-файле, а в БД текущего проекта (в виде одного поля типа BLOB).

БД подключённого проекта может иметь версию, более старую, чем версия текущего проекта. В этом случае процедура вначале создаст временную копию БД, произведёт её обновление, выполнит импорт, а затем удалит временную копию. Затраченное время при этом будет существенно больше, чем при совпадении версий.

Для запуска импорта используйте кнопки «**Импортировать выбранный**» или «**Импортировать все**».

Внимание! Операция импорта необратима. Перед импортом обязательно создайте резервную копию текущего проекта, а после импорта выполните диагностику проекта.

2.6.4 Срезы «для экспорта»

Автор проекта может ограничить объем экспортируемой проектной информации. Для каждого среза в «**Администраторе БД**» можно указать флажок «**Для экспорта**». Если объект не входит ни в один срез «**Для экспорта**», этот объект будет недоступен из других проектов.

Типовые данные (объектные окна, макросы и т.п.) доступны из подключающих проектов всегда.

2.6.5 Работа с данными из подключённых проектов

Проектные данные из подключённых проектов доступны в большинстве приложений СУРЫ. Все эти данные доступны только на чтение.

Примечание. Хотя проектные данные из подключённых проектов доступны только для чтения, соответствующие им оперативные данные могут, при наличии прав, быть доступны на запись. Например, название двигателя из подключённого проекта изменить нельзя, но включить этот двигатель – можно (если разработчик подключённого проекта предоставил вам право оперативного управления).

В таблице 3 приведены примеры связей в БД, которые могут ссылаться на сущности из подключённых проектов.

Таблица 3 – Примеры связей с подключенными проектами

Сущность в текущем проекте	Может импортировать из подключённого проекта	Приложение
Объект (составного типа)	Тип объекта	Базис
Составной тип	Тип подобъекта	Администратор БД
Пользователь	Группа пользователей, в которую он входит	Администратор БД
Право	Пользователь, которому дано это право	Администратор БД
Макроблок	Тип алгоритма (макрос)	Полис
Переменная техпрограммы	Тип переменной	Полис
Типовое изображение	Тип изображаемого объекта	Контур
Вставка (мнемосимвол или другое вставленное изображение)	Исходное изображение	Контур
	Марка объекта	
Аниматоры, рецепторы	Марка объекта	Контур
Рабочий стол	Изображение для панели	Контур
Монитор операторской станции	Рабочий стол	Контур
Кривая на графике	Марка объекта	Контур, Метрика
Параметр для Станции анализа	Марка объекта	Контур, Метрика

В диалоговых окнах записи из подключённого и из текущего проекта никогда не смешиваются в одном списке: вначале нужно выбрать проект по его псевдониму, а затем нужную запись. Вот пример выбора изображения для мнемосимвола аналогового датчика в **Контуре**:

Текущий проект обозначается зеленым значком, подключённые проекты – желтым, а фирменная библиотека – синим.

2.7 Диагностика проекта

2.7.1 Для чего нужна диагностика

В идеальной ситуации, при любых операциях с проектом все проектные данные должны сохранять полноту и непротиворечивость, иначе говоря, быть свободными от дефектов. В 99% случаях так и происходит. Но полностью избежать появления дефектов в проекте было бы возможно только ценой потери существенной части функциональных возможностей СУРЫ. Приведем примеры:

Для каждого объектного параметра числового типа должны быть заданы диапазоны и размерности. Допустим, в **Базисе** было добавлено несколько объектов типа **К**, после чего в описании типа **К** был добавлен еще один числовой параметр **Р**. Для каждого объекта типа **К** образуется дефект: не заданы диапазоны и размерности для параметра **Р**.

В **Полисе** была создана переменная, тип которой был взят из подключенного проекта. Затем в подключенном проекте этот тип удалили. После повторного импорта подключенного проекта возникнет дефект: неизвестен тип переменной.

Поэтому в СУРЕ принят толерантный подход: дефекты могут возникать, но их можно эффективно диагностировать и исправлять.

2.7.2 Терминология

Дефект – изъян или недочет в данных проекта. Любой дефект характеризуется обязательным параметром – **уровнем критичности**. В данный момент определено 3 уровня критичности:

– **Ошибка** – серьезный дефект способный привести к неработоспособности или некорректной работе некоторых приложений СУРЫ;

– **Предупреждение** – дефект не способный привести к некорректной работе приложений СУРЫ, но влияющий на потребительские качества СУРЫ (неоднозначность восприятия информации, снижение производительности и т.д.);

– **Информация** – дефект никак себя не проявляющий, но противоречащий логике проекта.

Тест – автоматическая процедура выявления дефектов проекта по определенным параметрам.

Диагностика – выполнение набора тестов.

Коррекция – процедура исправления дефекта. В том числе:

– **Ручная коррекция** – коррекция, производимая пользователем с помощью соответствующего средства проектирования (Базиса, Полиса и т.д.);

– **Автокоррекция** – коррекция, выполняемая приложением **Диагностика проекта** по команде пользователя.

2.7.3 Запуск диагностики

Диагностику можно запустить из дерева **СФЕРЫ** (пункт **Администрирование проекта / Диагностика проекта**). При этом можно отобрать необходимые тесты, проанализировать результаты их выполнения и выполнить коррекцию.

Кроме этого, диагностика запускается автоматически каждый раз при формировании dat-файла **Конструктором**. Этот вариант существенно более быстрый, но имеет ряд ограничений:

– тестируется не весь проект, а только данные, попавшие в dat-файл;

– запускаются все тесты, кроме теста корректности файлов таблиц и теста изображений;

– выполняется только диагностика, коррекция невозможна;

– работа ведется до первой ошибки.

2.8 Резервное копирование проекта

Проектные данные могут быть частично или полностью испорчены по различным причинам: сбои жесткого диска или сети, ошибки программного обеспечения, неправильные действия пользователей.

Рекомендация. Регулярно выполняйте резервное копирование проекта. Без этого результат многодневной работы коллектива может пропасть.

В СУРЕ предусматривается 2 рода резервных копий. Копии первого рода создаются и хранятся на сервере, путь к ним указывать не требуется. Эти копии служат «точками восстановления» проекта. При восстановлении отображается список всех сделанных копий первого рода для этого проекта.

Копия второго рода представляет собой один файл с расширением kdbbak. Имя и путь файла указывается пользователем при архивировании. Копии второго рода нужны для передачи проектов вне рамок локальной сети (через интернет, диск USB-флэш и т.п.).

В процессе создания резервной копии автоматически устанавливается блокировка на БД проекта. Это делается для того, чтобы другие пользователи не могли вносить изменения в проект, пока идет архивирование.

3 Абоненты сети и приложения

3.1 Абоненты сети

Под **абонентами сети** мы будем понимать вычислительные устройства, которые:

– участвуют в работе АСУ ТП.
– подключены к сети Ethernet.
– исполняют программные приложения СУРЫ.

Примеры устройств, НЕ являющихся абонентами сети:

– компьютер из бухгалтерии (не участвует в АСУ ТП);
– модуль АЦП (не подключен к Ethernet).

В новом проекте на СУРЕ применяются следующие типы абонентов сети:

– Контроллеры «Эликонт-100»
– Рабочие станции (ПК в промышленном или обычном исполнении).
– Коммутаторы.

Как правило, все абоненты сети должны быть описаны в БД проекта. Более подробно (см. раздел 3.3).

3.2 Приложения

3.2.1 Классификация приложений СУРЫ

Приложение – это программный элемент, который может быть индивидуально, независимо от других приложений, запущен по команде пользователя, либо автоматически. Приложения СУРЫ подразделяются на визуальные и серверные. Визуальные приложения реализуют человеко-машинный интерфейс, серверные – обеспечивают выполнение функций, не требующих вмешательства пользователя. Серверные приложения называются кратко серверами.

Классификация приложений СУРЫ представлена в таблице 5 (устаревшие приложения не показаны).

Таблица 5 – Классификация приложений

<table>
<tr><td rowspan="8">Серверные</td><td rowspan="3">Программируемые серверы</td><td rowspan="2">Контроллеры</td><td>Эликонт</td></tr>
<tr><td>РС-контроллеры</td></tr>
<tr><td colspan="2">Моделирующие серверы</td></tr>
<tr><td rowspan="5">Специализированные серверы</td><td colspan="2">Архивные станции</td></tr>
<tr><td colspan="2">Прокси-серверы</td></tr>
<tr><td colspan="2">Мониторы приложений</td></tr>
<tr><td colspan="2">Серверы лицензий</td></tr>
<tr><td colspan="2">Серверы БД проектов</td></tr>
<tr><td rowspan="4">Визуальные</td><td rowspan="2">Средства персонала АСУ ТП</td><td colspan="2">Терминал</td></tr>
<tr><td colspan="2">Метрика</td></tr>
<tr><td>Средства проектанта</td><td colspan="2">Базис, Полис, Контур и др.</td></tr>
<tr><td>Вспомогательные средства</td><td colspan="2"></td></tr>
</table>

3.2.2 Контроллеры

Под **Эликонтом-100** здесь понимается программное обеспечение, выполняемое контроллером Эликонт (поскольку никаких других приложений СУРЫ Эликонт не выполняет, можно объединить под обозначением Эликонт-100 устройство и выполняемое на нём программное обеспечение). ПО Эликонта-100 можно подразделить на:

– системное ПО (ОС Microsoft Windows 10 с драйверами устройств);

– фирменное ПО (ядро, различные подсистемы, встроенная библиотека алгоритмов);

– прикладное ПО (скомпилированная технологическая программа).

Системное и фирменное ПО Эликонта не зависят от конкретного проекта. Специфика сосредоточена в технологической программе (сокращенно техпрограмме), которая создаётся проектантами в **Полисе**, а затем компилируется и загружается по сети в контроллер.

Рядом с Эликонт-100 в таблице расположен **РС-контроллер**. Логически это аналог Эликонта-100 выполняемый на персональном компьютере. Техпрограмма для РС-контроллера составляется и загружается точно так же, как для Эликонта. В то же время, между Эликонтом и РС-контроллером имеется ряд отличий:

– РС-контроллер не подключен к шинам Profibus и INEL и, следовательно, не может работать с УСО.

– РС-контроллер не является устройством реального времени: время цикла в нем не гарантировано.

– Скорость вычислений у РС-контроллера обычно намного выше, чем у Эликонта.

– Некоторые алгоритмы могут выполняться только на РС-контроллере, например, алгоритмы для работы с файлами.

Эти отличия определяют **СФЕРУ** применения РС-контроллеров: это ресурсоёмкие расчётные задачи и задачи связи с другими системами по нестандартным протоколам.

3.2.3 Моделирующие серверы

Моделирующие серверы используются не в реальной работе АСУ ТП, а в виртуальной среде. Их назначение – тренажеры для операторов и средства наладки для проектантов.

Моделирующий сервер выполняет одновременно 2 задачи:

– Выполняет техпрограмму одного или нескольких контроллеров.

– Имитирует значения сигналов УСО для этих техпрограмм с помощью модели объекта.

При описании контроллеров в проекте каждому контроллеру назначается моделирующий сервер (нескольким контроллерам может быть назначен один моделирующий сервер). Этот моделирующий сервер будет выполнять техпрограммы всех контроллеров, которым он назначен. Кроме того, он будет выполнять техпрограмму модели. Эта техпрограмма создается проектантом в Полисе точно теми же средствами, что и техпрограммы для контроллеров.

При большом количестве контроллеров в проекте от моделирующего сервера требуется высокая производительность. Желательно использовать компьютер с несколькими процессорами. Если необходимо, можно завести в проекте несколько моделирующих серверов и организовать сетевой обмен между ними.

3.2.4 Архивные станции

Основные функции архивной станции:

– сохранение на жестком диске хронологии работы автоматизированного технологического процесса: значений технологических параметров, ошибок и событий, действий персонала;

– предоставление сохраненной информации клиентским приложениям по запросу.

Каждому контроллеру и каждому моделирующему серверу в проекте назначается архивная станция (нескольким контроллерам может быть назначена одна архивная станция). Эта станция будет архивировать информацию, формируемую теми контроллерами, которым она назначена.

Каждая архивная станция может быть либо одиночной, либо дублированной. Чтобы описать в проекте дублированную архивную станцию, нужно для абонента сети **А** указать флажок **Архивная станция**, для абонента сети **В** – флажок **Резервный архив** и абоненту **В** в качестве архивной станции назначить абонент **А**.

3.2.5 Прокси-серверы

Одной из важных особенностей АСУ ТП на базе СУРЫ является отсутствие центрального сервера оперативных данных. Это повышает живучесть и масштабируемость системы и облегчает

обновление ее версий. Отсутствия сервера данных означает, что рабочие станции могут обмениваться информацией непосредственно с контроллерами. Если количество рабочих станций велико, поддержка контроллером сеансов с ними всеми может стать узким местом производительности системы. Избежать этого позволяют прокси-серверы: они агрегируют сеансы и передаваемые данные.

Каждый контроллер Эликонт-100 поддерживает не более 20 сеансов с рабочими станциями. Для АСУ ТП малого и среднего масштаба этого обычно достаточно. Если количество рабочих станций больше 20, их необходимо разделить на 2 эшелона (Рисунок 3). В 1-й эшелон включаются станции, непосредственно участвующие в управлении, для которых надежность и скорость взаимодействия с контроллерами принципиально важна. Во 2-й эшелон включаются все прочие станции – они будут получать информацию через прокси-сервер.

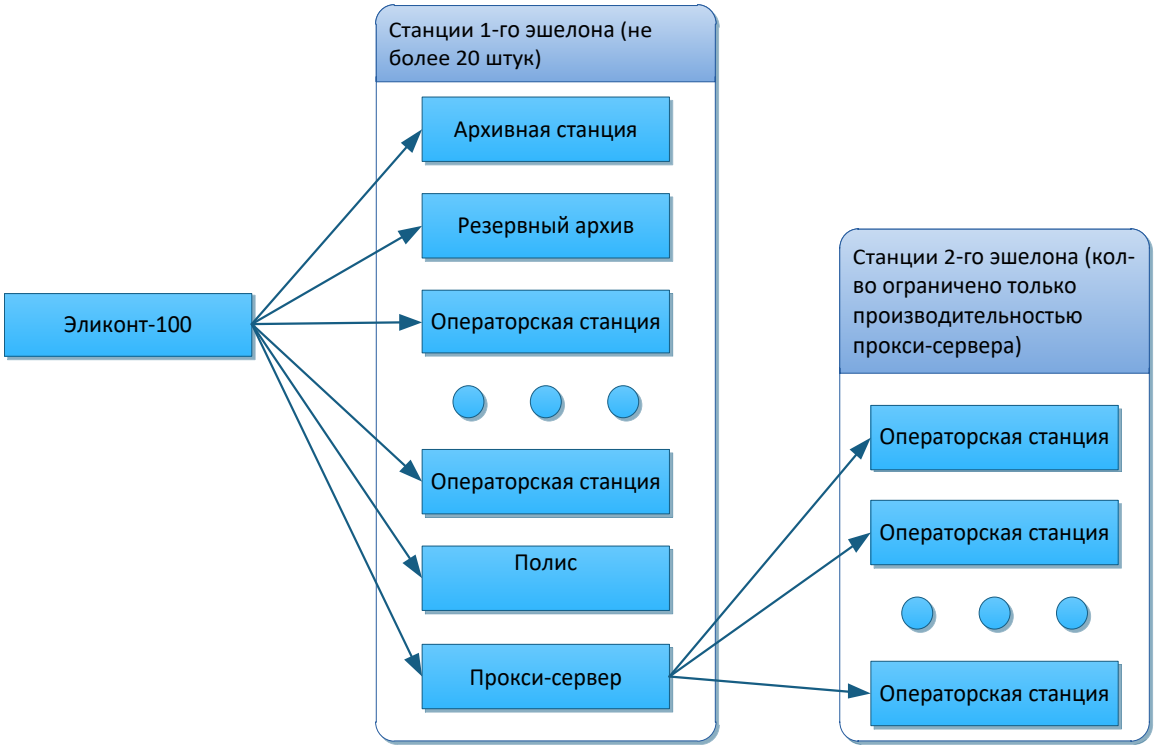

Рисунок 3 – Разделение Операторских станций на 2 эшелона

Примечание. Через Прокси-сервер могут работать операторские станции СУРЫ и сторонние клиенты OPC UA. **Архивная станция** и **Полис** через Прокси не работают.

В проекте необходимо:

– Указать флажок **Прокси-сервер** для одного или нескольких абонентов сети, на которых будут запускаться прокси-серверы.

– Для каждой рабочей станции задать опцию **Соединение с контроллерами**: для станций 1-го эшелона – напрямую, для станций 2-го эшелона – через прокси-сервер.

– Указывать конкретный прокси-сервер для каждой станции 2-го эшелона не нужно – станция автоматически выберет один из доступных в данный момент серверов. Тем самым обеспечивается резервирование прокси-серверов.

Сам прокси-сервер всегда подключается к контроллерам напрямую, даже если для этого компьютера указана опция соединения **через прокси-сервер**, но другие приложения, запускаемые на этом компьютере, будут работать через прокси.

Прокси-сервер может понадобиться не только для обеспечения масштабируемости системы. Второй его важной функцией является расширение возможностей взаимодействия со сторонними системами. Стандартный протокол OPC UA позволяет сторонним системам взаимодействовать непосредственно с Эликонтами. Но наладить работу по OPC UA с прокси-сервером может оказаться проще, потому что прокси-сервер предоставляет расширенную поддержку стандарта OPC UA, в

частности, реализует модель OPC UA Data Access (специализированная модель для работы с оперативными данными АСУ ТП).

3.2.6 Мониторы приложений

Монитор приложений обеспечивает:

– Автоматический запуск приложений СУРЫ.

– Автоматический перезапуск приложений СУРЫ в случае их зависания или нештатного завершения.

– Сохранение диагностической информации в файлах журналов.

– Удаленное управление приложениями.

Служба монитора приложений должна работать на каждой рабочей станции. На этапе проектирования используется только третий пункт.

3.2.7 Серверы лицензий

Для защиты от несанкционированного использования приложений в СУРЕ применяются аппаратные ключи Sentinel, подключаемые к USB-портам. В ключе хранится информация о количестве лицензий по каждому типу программ. Один ключ можно использовать сразу на несколько компьютеров. На том компьютере, в который вставлен ключ, должна работать служба **сервер лицензий** (или Sentinel-сервер). Эта служба контролирует количество выполняющихся программ.

Для бесперебойной работы обычно используется не одиночный, а дублированный ключ, т.е. 2 ключа, подключенные к двум разным компьютерам. На каждой рабочей станции нужно указать сетевые имена этих компьютеров.

Внимание! Данная настройка сохраняется не в БД проекта, а на локальном диске компьютера. Настройка будет действовать независимо от того, какой проект выбран в **СФЕРЕ**. Задается эта настройка в специальном приложении **Активация**.

Основные операции проектирования в СУРЕ могут выполняться без лицензий, но лицензии потребуются при переходе в режим обзора или при запуске РС-контролера.

3.2.8 Серверы БД проектов

Если на компьютере размещена БД проекта, необходимо наличие на этом компьютере службы **сервер БД проекта**. Например, можно базы данных всех проектов хранить на контроллере домена, в этом случае достаточно установить сервер БД только на нем. Управление базами данных на сервере БД и другие его настройки выполняются удаленно из мастера добавления проектов.

3.3 Описание абонентов и приложений в проекте

Описание абонентов производится в приложении **Администратор БД**, форма **Абоненты сети**. В левой части формы содержится дерево абонентов. Узлы первого уровня в дереве представляют типы абонентов. Эти узлы добавляются автоматически и не редактируются. Узлы второго уровня добавляются проектантом, каждый такой узел представляет одного абонента сети.

Если выбрать узел второго уровня, в правой части формы появится набор закладок, в которых можно редактировать свойства выбранного абонента.

4 Среда выполнения

В СУРЕ есть понятие **среды выполнения** (кратко: **среда**). Цель – облегчить наладку проектов и создание тренажеров.

Имеются три среды: **реальная**, **виртуальная** и **локальная**. Среда выбирается в **СФЕРЕ** и действует на все запущенные из него приложения.

4.1 Назначение сред

Таблица 6 – Среды выполнения

Среда	Использование
Реальная	При оперативной работе с автоматизируемым объектом
Виртуальная	При организации тренажеров
Локальная	При проектировании и наладке

Эликонт-100 и обрабатываемые им данные используются только в реальной среде. В виртуальной и локальной среде эти данные подменяет моделирующий сервер. Отличие локальной среды в том, что клиентские приложения ищут моделирующий и архивный сервер не в сети, а на том же компьютере. Кроме того, в локальной среде ослаблены требования подсистемы безопасности.

4.2 Особенности сред

В таблице 7 перечислены основные отличия работы приложений в зависимости от среды. Под словом «локальный» понимается «находящийся на том же компьютере, что и клиентское приложение».

4.3 Установка среды

– каждое приложение работает в той среде, в которой оно было запущено (в ходе работы приложения среда не меняется);

– два приложения могут работать в разных средах, даже на одном компьютере;

– в настройках Монитора приложений для каждого приложения можно назначить среду, в которой Монитор должен запускать это приложение;

– задаваемая в **СФЕРЕ** среда не зависит от текущего пользователя и выбранного проекта, но зависит от компьютера (т.е., для каждого компьютера среду следует указывать отдельно).

Таблица 7 – Отличия работы приложений в зависимости от среды выполнения

Особенность	Реальная среда	Виртуальная среда	Локальная среда
Информация от датчиков и механизмов	Поступает от реальных объектов через контроллеры Эликонт-100	Поступает от моделирующего сервера	Поступает от локального моделирующего сервера
Информация от модели	Недоступна	Поступает от моделирующего сервера	Поступает от локального моделирующего сервера
Архивная информация	Выдается архивным сервером из хранящегося на нем реального архива	Выдается архивным сервером из хранящегося на нем виртуального архива	Выдается локальным архивным сервером из локального архива
Расположение архива	Папка X, указанная в настройках (по умолчанию - папка ниши)	Папка X\Model	Папка X\Local
Состав архивируемых параметров	Параметры от всех контроллеров, которым назначена данная архивная станция	То же плюс параметры от моделирующих серверов, которым назначена эта архивная станция	Параметры от всех контроллеров, независимо от того, какая архивная станция им назначена
Подсистема безопасности	Работает в полном объеме	Работает в полном объеме	Не требует, чтобы текущий компьютер был прописан в проекте

4.4 Принцип изоляции сред

Важно! Приложения, работающие в разных средах, не могут взаимодействовать друг с другом.

Основная цель изоляции сред в том, чтобы исключить случайное воздействие виртуальной АСУ ТП на реальные объекты. В виртуальной среде пользователь может посылать любые управляющие команды, не беспокоясь о том, что они могут попасть в реальный контроллер. То же правило действует для архивных станций: в виртуальной среде архивная станция ведет отдельный виртуальный архив, получает данные только от моделирующих серверов и отвечает на запросы только от клиентов, работающих в виртуальной среде. Таким образом, изолированными оказываются не только текущие, но и архивные данные.

В локальной среде приложение взаимодействует только с локальными серверами. Это позволяет отлаживать проект даже при отсутствии в сети контроллеров и архивных станций.

Внимание! Выбор среды влияет только на взаимодействие с контроллерами, архивными и моделирующими серверами. Обращения к серверу базы данных или серверу лицензий не зависят от среды и будут выполняться по сети даже в локальном режиме.

5 Объекты

Понятие объекта является одним из ключевых понятий СУРЫ. Прежде чем давать формальные определения, приведем пример объекта, так как он понимается в СУРЕ.

5.1 Поясняющий пример

Рассмотрим некоторый электродвигатель, например, вентилятор. Прежде всего, это реальное устройство с маркой, паспортными характеристиками и другими статическими **атрибутами**. Эти атрибуты можно занести в проект СУРЫ, для чего нужно, прежде всего, добавить в проект объект типа **Двигатель**.

Двигатель подключен к АСУ ТП проводами, по которым передаются **сигналы**, имеющие строго определенный смысл: **Включен**, **Отключен**, **Включить**, **Отключить**, **Ток** и др. Сигналы в контроллере принимаются подсистемой ввода-вывода и передаются в технологическую программу. В технологической программе эти сигналы обрабатывается алгоритмом **Двигатель**, который является **управляющим алгоритмом** для объектного типа **Двигатель**. Для каждого входного или выходного сигнала разработчиком задан соответствующий вход или выход алгоритма. Алгоритм обрабатывает значения на входах и формирует значения на выходах. Кроме входов и выходов, связанных с сигналами, в алгоритме есть много других входов и выходов, например, вход **Авт** для приема команд от других алгоритмов или выход **Состояние**, значение которого вычисляется алгоритмом в зависимости от значений нескольких входов.

Станции верхнего уровня оперируют не с сигналами, а с **параметрами** объекта. Параметр является динамическим свойством объекта, значение которого можно отобразить на терминале, а историю изменений получить на метрике. Примером параметра **Двигателя** является **Состояние**, его значение берется с одноименного выхода алгоритма. Параметр может быть связан также и с входом алгоритма. Значения некоторых параметров могут быть изменены оператором, в этом случае измененное значение поступает на соответствующий вход алгоритма. Примером может служить **Режим** двигателя (режим определяет приоритеты ручных и автоматических команд).

При работе в виртуальной среде помимо управляющего алгоритма используется также **моделирующий алгоритм ИмДвигателя** (имитатор двигателя). Моделирующий алгоритм имитирует значения сигналов, которые затем передаются на входы управляющего алгоритма вместо реальных сигналов. И наоборот, значения выходов управляющего алгоритма, связанных с выходными сигналами, передаются на входы моделирующего алгоритма. Передача этих значений выполняется автоматически, не требуя конфигурирования со стороны проектанта. Некоторые входы и выходы алгоритма **ИмДвигателя** также могут являться параметрами объекта Двигатель, например, параметр **Отказ** позволяет имитировать нештатную ситуацию непосредственно с операторской станции.

Мы видели, что источником значений одних параметров является управляющий алгоритм, других параметров – моделирующий алгоритм. Есть еще параметры, связанные с подсчетом наработки (моточасы). Значение наработки должно подсчитываться на протяжении многих лет, поэтому она хранится и рассчитывается не в контроллере, а на архивной станции. К числу подобных параметров объекта **Двигатель** относятся **ОбщаяНаработка**, **ДатаПоследТО** и др. Важно, что доступ к объектным параметрам, их анимация на терминале, представление на графиках и таблицах не зависят от того, берется параметр из управляющего, моделирующего или архивного алгоритма.

С точки зрения АСУ ТП все электродвигатели однотипны: имеют одинаковый состав атрибутов, параметров и сигналов, обрабатываются одинаковыми алгоритмами, имеют похожее визуальное представление. Это позволяет рассматривать **Двигатель** как **объектный тип**. Типизация позволяет описать все указанные выше связи на уровне типа, что радикально сокращает объем рутинных операций при проектировании. В Контуре можно подготавливать типовые изображения, которые визуализируют параметры и атрибуты объектного типа. Встроенная библиотека **Библио** содержит большое количество готовых типовых изображений для фирменных типов.

В СУРЕ проектант может также описывать собственные объектные типы, их параметры, сигналы, ассоциированные алгоритмы и прочие свойства.

Не все объекты столь сложны, как **Двигатель**. Некоторые объекты могут не иметь соответствующего физического устройства, например, **Регулятор** или **КаналЗащиты**. Такие объекты мы будем называть логическими, противопоставляя их физическим объектам. У логических объектов отсутствуют сигналы, они не нуждаются в модели.

5.2 Объекты и их типы

Таким образом, **объект** – это совокупность имеющейся в СУРЕ информации, относящейся к одному устройству, физическому или логическому.

Каждый объект имеет определенный **объектный тип**. Тип задается при создании объекта и не может быть изменен впоследствии. В СУРЕ имеется несколько десятков встроенных объектных типов (например, **ВводАналоговый** или **Двигатель**). Эти типы называются **простыми**.

Простые типы описываются разработчиками СУРЫ и не могут быть изменены проектантами. В то же время, проектант может конструировать свои объектные типы, называемые **составными**.

Описания составных типов сохраняются в проекте. Допускается импорт типов из подключённых проектов: при создании объекта можно выбрать его тип не только из текущего, но и из подключённого проекта. В Библиотеке Библио также имеются описания составных типов, доступные для импорта.

Приведем пример окна выбора объектного типа (Рисунок 4).

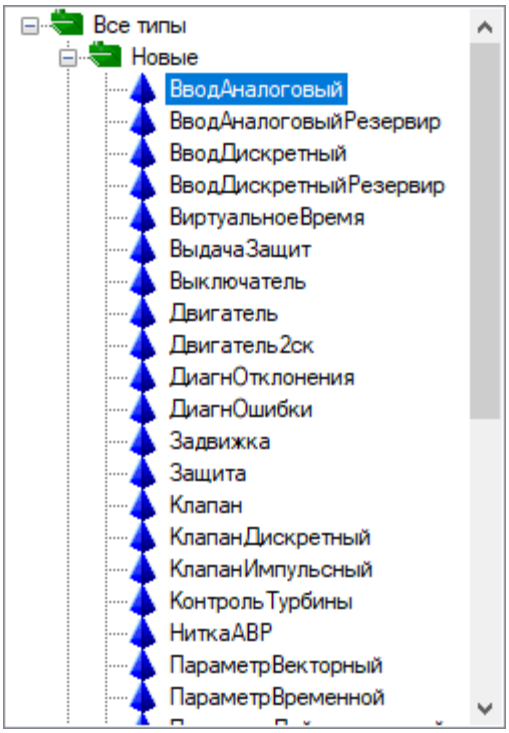

Рисунок 4 – Пример окна выбора объектного типа

Принадлежность объекта к типу предопределяет:

– ассоциированный управляющий и моделирующий алгоритмы;
– состав параметров, сигналов и дополнительных атрибутов объекта;
– состав подобъектов (только для составного типа).

Далее эти элементы будут рассмотрены подробнее.

5.3 Ассоциированные алгоритмы

Каждый объектный тип может быть **ассоциирован** с одним **управляющим** и одним **моделирующим** алгоритмом. Моделирующие алгоритмы нужны только для тех объектных типов, которые описывают реальное оборудование – датчики, механизмы. Для регуляторов или защит моделирующие алгоритмы не нужны, поскольку регуляторы и защиты являются чисто программными конструкциями, не связанными с «железом».

В реальной среде управляющие алгоритмы выполняются в контроллере, а моделирующие алгоритмы – не выполняются. В виртуальной и локальной среде и управляющие, и моделирующие алгоритмы выполняются моделирующим сервером.

Фирменные объектные типы могут быть ассоциированы только с встроенными алгоритмами, а составные объектные типы – только с пользовательскими алгоритмами (макросами).

Ассоциация объектного типа с алгоритмом означает, что каждый объект этого типа может быть привязан к алгоблоку только соответствующего типа. На Рисунке 5 представлен пример, в котором объектный тип **Двигатель** ассоциирован с управляющим алгоритмом **Двигатель** и с моделирующим алгоритмом **ИмДвигателя**. Эта ассоциация определяет, к каким алгоблокам можно привязать объект «Масляный насос №1» (но не требует обязательной привязки).

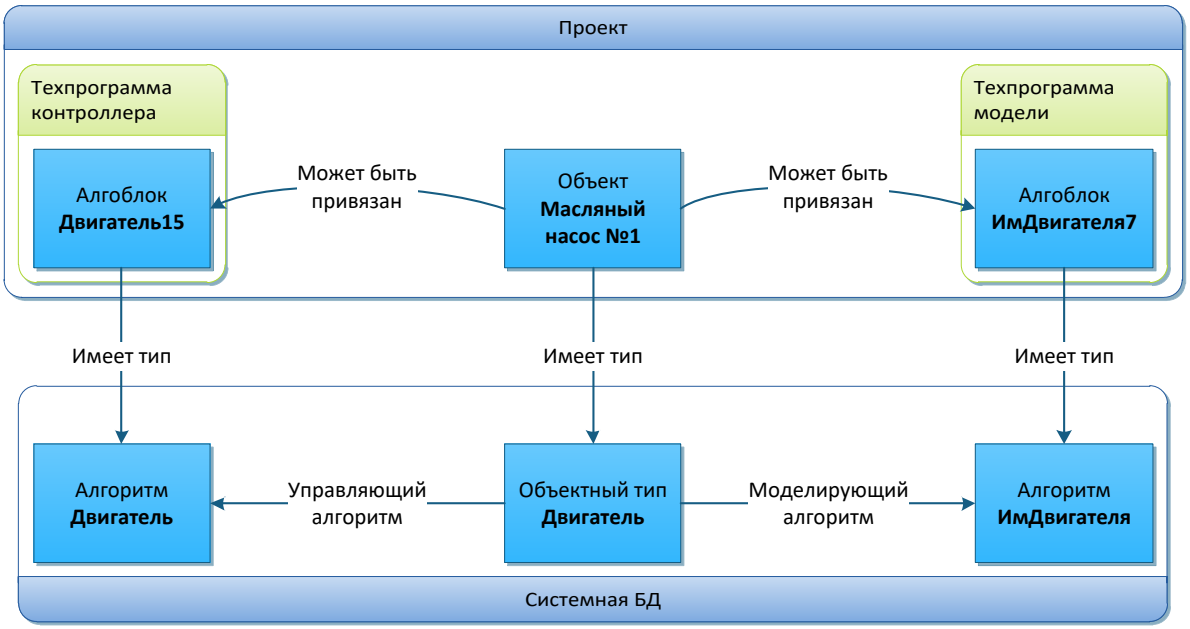

Рисунок 5 – Пример привязки объекта к алгоблоку

Алгоритм, с которым ассоциирован некоторый объектный тип, называется **объектным алгоритмом**.

5.4 Объектные атрибуты

Объектный атрибут – это статическая характеристика объекта, значение которой фиксируется в проекте и не меняется в оперативном режиме. На операторской станции доступны значения атрибутов для каждого объекта.

Атрибуты подразделяются на **основные** и **дополнительные**. Основные атрибуты присущи всем объектам, независимо от их типа. Примерами могут служить **Марка**, **Имя объекта**. Состав основных атрибутов предопределен разработчиками СУРЫ.

Список дополнительных атрибутов формируется проектантом в **Администраторе БД**. Список является общим для всего проекта, но для каждого атрибута указывается, к объектам каких типов он применим. Например, можно создать атрибут **Изготовитель**, который будет применим ко всем датчикам и механизмам.

Для каждого дополнительного атрибута нужно указать:

– имя, уникальное в пределах проекта;
– тип значения: строка, число или дата;
– минимальное и максимальное значение (не обязательно);
– значение по умолчанию (не обязательно);
– шаблон ввода (не обязательно);
– список возможных значений атрибута (не обязательно)4
– к объектам каких типов применим данный атрибут.

Значения основных и дополнительных атрибутов задаются для каждого объекта в **Базисе** (форма **Объекты**, закладка **Атрибуты**).

Чтобы значение того или иного атрибута стало отображаться на операторской станции, нужно в **Контуре** открыть типовое изображение объектного окна, добавить в него элемент «Реквизит» и указать, какой атрибут этот элемент отображает.

Есть ряд особенностей:

– состав применимых атрибутов не зависит от технологического подтипа объекта;

– можно задавать дополнительные атрибуты не только для простых, но и для составных типов;

– можно использовать дополнительные атрибуты, описанные в подключённых проектах;

– задание прав для редактирования атрибутов более не требуется (право редактирования значения атрибута определяется правом редактирования объекта).

5.5 Объектные параметры

Объектный параметр – это динамическая характеристика объекта, значение которой формируется в оперативном режиме тем или иным источником. На операторской станции можно наблюдать как текущие значения параметров, так и прошлые, сохраненные в архиве. Значения некоторых параметров могут быть изменены оператором, в этом случае параметр называется **управляемым**.

Все объекты, относящиеся к одному объектному типу, имеют одинаковый состав параметров. Этот состав определяется при описании типа объекта. Для составного типа, описываемого проектантом, состав параметров также определяет проектант.

Каждый объектный параметр имеет:

– имя;

– тип значения (см. 5.5.1 настоящего руководства);

– источник (Контроллер, Модель или Архив, см. 5.5.2 настоящего руководства);

– доступ (Чтение, Запись, Чтение + Запись);

– ассоциированный вход или выход алгоритма.

Подробно о задании этих свойств рассказывается в разделе Составные объектные типы.

5.5.1 Типы параметров

Тип параметра определяет, какие значения может принимать этот параметр, какими способами его можно визуализировать, какие дополнительные характеристики необходимо для него задать.

Таблица 8 – Типы параметров

Тип параметра	Описание	Примеры	
		Тип объекта	Параметр
Число	Значение является действительным числом и может меняться непрерывно в определенном диапазоне	ВводАналоговый	Значение
		Задвижка	Положение
Состояние	Имеется допустимый набор значений, каждое значение поименовано	Задвижка	Состояние
Время	Значение характеризует промежуток времени или астрономическую дату и время	Задвижка	Тайм-аут
		Двигатель	ДатаПоследТО (дата последнего техобслуживания)
Набор бит	Значение представляет собой совокупность логических элементов	Задвижка	Блокировки
Строка	Текст длиной до 255 символов	СигналСтроковый	Значение
Целый	Целое число	Задвижка	ЗаданиеОт (номер входа, с которого действует команда автоматики)
Логический	Да или Нет	Задвижка	ОперНапряж (наличие оперативного напряжения)

5.5.2 Источники параметров

При описании объектного типа для каждого параметра указывается его источник: контроллер, модель или архив.

Таблица 9 – Источники параметров

Источник параметра	Назначение параметра
Контроллер	Эти параметры используются оператором для получения информации об объекте и управления им. В виртуальной и локальной среде эти параметры формируются моделирующим сервером
Модель	Эти параметры используются инструктором для коррекции параметров модели и имитации нештатных ситуаций
Архив	Эти параметры позволяют отслеживать износ оборудования, предупреждать о необходимости ремонта

Когда клиентскому приложению (например, Терминал) нужно получить текущее значение параметра, приложение должно решить, с какого сервера запрашивать параметр. На это решение влияют три фактора:

– Режим запуска приложения.
– Текущая среда выполнения.
– Источник параметра.

Режим запуска выбирается при запуске приложения из **СФЕРЫ**.

Таблица 10 – Режимы запуска приложений

Режим запуска	Откуда берутся значения параметров
С эмулятором	Формируются программным эмулятором
С архивом	Запрашиваются из архивной станции (в локальной среде выполнения – из локальной архивной станции).
С контроллерами	В зависимости от среды выполнения и источника параметра – см. следующую таблицу

Таблица 11 – Среды выполнения и источники параметров

Среда \ Источник	Контроллер	Модель	Архив
Реальная	Из контроллера	Недоступны	Из архива
Виртуальная	Из моделирующего сервера		Из архива модели
Локальная	Из локального моделирующего сервера		Из локального архива

5.5.3 Преобразование значений

Для большинства параметров их значение параметра в точности совпадает со значением соответствующего входа или выхода. Однако параметры типа Число требуют дополнительного преобразования. Дело в том, что технологическая программа всегда оперирует с нормализованными значениями переменных, в то время как значение параметра должно предоставляться в физических единицах.

На Рисунок 6 представлена вся цепочка преобразований аналогового сигнала. Независимо от типа датчика, система ввода-вывода будет формировать значение входной переменной в диапазоне [0; 1], так называемое **нормализованное значение**. (В нашем примере 0 соответствует току 4 mA, а 1 – току 20 mA.) Входы и выходы алгоритмов также имеют нормализованные значения – алгоритму не нужно знать значение в физических единицах.

Преобразование в физические единицы выполняется сетевой подсистемой контроллера, когда станции верхнего уровня запрашивают у нее значение параметра. Это – линейное преобразование, выполняемое по формуле **Ф = (A100 – A0) * Н + A0**, где **Ф** – физическое значение, **Н** – нормализованное, **A0** и **A100** – коэффициенты, задаваемые для данного параметра в **Базисе** (форма **Объекты**, закладка **Диапазоны**).

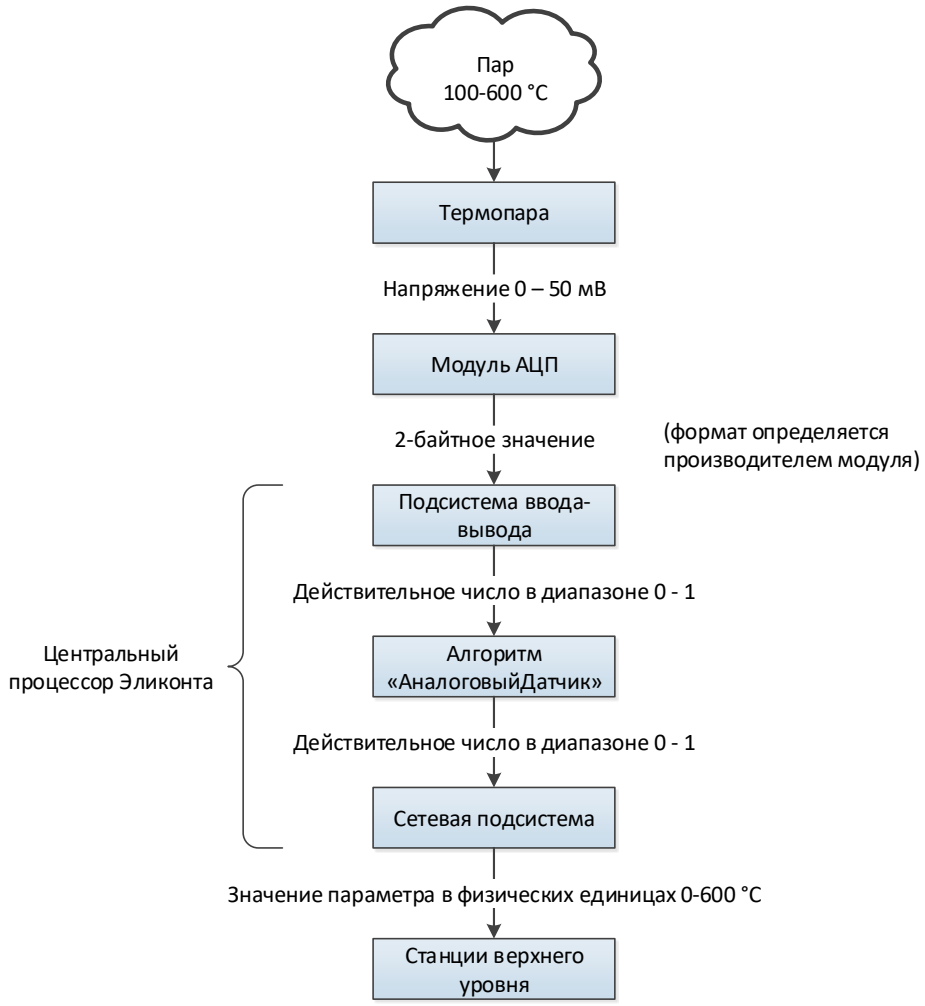

Рисунок 6 – Цепочка преобразований аналогового сигнала

Примечание. Неправильно интерпретировать коэффициенты **A0** и **A100**, как минимальное и максимальное значение параметра. На самом деле, это физические значения, соответствующие значениям 0 и 1 переменной в технологической программе. Ничто не запрещает переменной иметь значения вне диапазона [0; 1]. Часто бывает и так, что реальные значения переменной принадлежат небольшой части этого диапазона, например [0,6; 0,8]. Чтобы при отображении на графиках и барографах эта часть диапазона была растянута на всю высоту, следует заполнить поля **Минимум**, **Максимум** и **База** в той же закладке **Базиса**.

Все станции верхнего уровня, включая архив и станцию анализа, работают с физическими значениями параметров. Это касается и сторонних систем, подключающихся к Эликонтам по протоколу OPC UA.

5.6 Объектные сигналы

Каждый датчик или исполнительное устройство имеет одну или несколько цепей (проводов) для подключения к контроллеру. Информацию, передаваемую по такой цепи, традиционно называют сигналом. Кроме таких сигналов, СУРА также может потреблять информацию, поступающую по цифровым каналам связи.

Обобщенно, **сигнал** – это элементарная единица информации, поступающая в контроллер или из него через подсистему ввода-вывода. **Входные** сигналы передают информацию о состоянии устройства в контроллер. **Выходные** сигналы передают устройству управляющие команды от контроллера. Сигналы поступают через модули УСО в центральный процессор, где подсистема ввода-вывода передает их технологической программе в виде **сигнальных переменных**.

Программируя контроллер, проектант может явно добавлять входные и выходные сигнальные переменные и связывать их с каналами ввода-вывода. Но значительно быстрее и удобнее делать это с помощью объектов.

При описании объектного типа можно задать набор входных и выходных **объектных сигналов**. Элементы этого набора однозначно соответствуют проводам, связывающим устройство данного типа с контроллером. Для каждого объектного сигнала задается:

– имя сигнала;

– направление (входной или выходной);

– тип значения.

Для входного сигнала задается вход управляющего и выход моделирующего алгоритма. Для выходного сигнала задается выход управляющего и вход моделирующего алгоритма.

Если добавить в проект объект данного типа и в **Полисе** перетащить его марку в поле одной из задач контроллера, будут автоматически выполнены следующие действия:

– в задачу будет добавлен алгоблок, реализующий управляющий алгоритм для данного типа объекта;

– алгоблок будет привязан к экземпляру объекта;

– для каждого объектного сигнала будет создана сигнальная переменная;

– каждая такая переменная будет связана с соответствующим входом или выходом алгоблока.

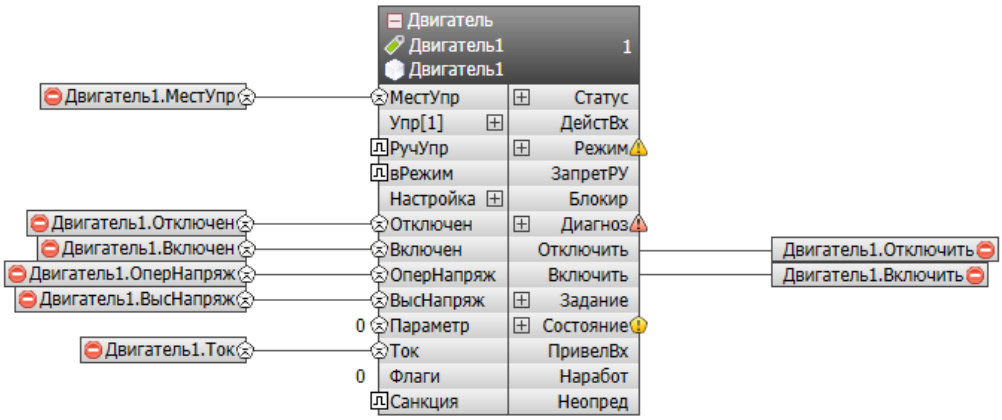

Рисунок 7. Добавление объекта в поле задачи Полиса

Проектанту останется только распределить эти переменные по каналам ввода-вывода.

Если теперь перетащить марку объекта в поле одной из задач моделирующего сервера, будут выполнены аналогичные действия, но не для управляющего, а для моделирующего алгоритма.

5.7 Ввод объектов

СУРА предоставляет несколько способов добавления объектов в проект. Выбор способа определяется решаемой задачей (таблица 12).

Таблица 12 – Способы ввода объектов в базу данных

Способ	Приложение	Описание
Добавление объектов вручную	**Базис**, форма **Объекты**	Основные атрибуты добавляемого объекта заполняются в диалоговом окне, открываемом кнопкой . Если нужно добавить подряд несколько объектов, рекомендуем снять флажок «Закрыть это окно после добавления»
Копирование объектов	**Базис**, форма **Объекты**	Кнопка создает копию выделенных объектов в том же проекте
Импорт имеющихся данных	**Импорт объектов**	Если у заказчика уже имеется описание объектов в виде таблиц Microsoft Excel того или иного формата, эту информацию можно импортировать в проект СУРЫ
Добавление объектов из Полиса	**Полис**, редактор диаграмм	Можно добавить объект из **Полиса**, выбрав алгоблок, вход или выход, и вызвав команду **Объект – Добавить и привязать**. Используется то же диалоговое окно, что и в **Базисе**. Этот способ чаще применяется для монопараметрических объектов

5.8 Привязка объектов

Как только объект добавлен в проект, его сразу можно использовать в различных целях, например, для построения мнемосхем в **Контуре**. Однако, при открытии мнемосхемы на операторской станции изображение объекта будет «мёртвым». Причина этого в том, что для объекта не задан источник данных. Чтобы объект «ожил», нужно выполнить для него привязку. Таким образом, привязка объектов – это процедура задания первоисточника данных для каждого объекта.

Каждый объект может находиться в одном из 4-х состояний, как указано в таблице 13 и на рисунке 8.

Таблица 13 – Состояния объектов при их привязке

Состояние	Описание
Не привязан	Для объекта не определен источник данных, он является «мертвым».
Полупривязан	Для объекта определен контроллер (или другой сервер), но не определен алгоблок техпрограммы. Сигналы полупривязанного объекта можно распределять по каналам ввода-вывода. Для станций верхнего уровня объект остается «мертвым»
Привязан	Для объекта определен алгоблок техпрограммы, который поставляет значения его параметров. Эти значения доступны на верхнем уровне
Суперпривязан	Объект входит в состав другого объекта, имеющего сильный тип. (Об этом подробнее будет сказано в разделе Составные объектные типы). Для суперпривязанных объектов источник данных определяется привязкой родительского объекта

В процессе проектирования состояние объекта можно изменять следующим образом (см. рисунок 8):

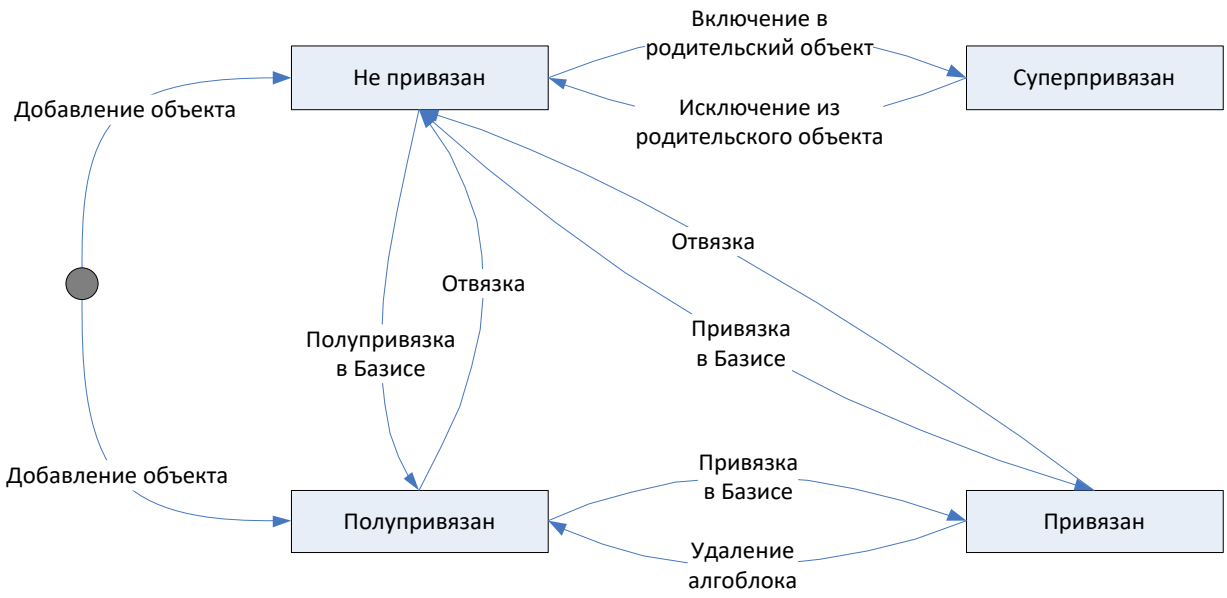

Рисунок 8 - Состояния объектов при их привязке

Опишем эти операции более подробно:

– при добавлении объекта можно сразу указать для него сервер-источник, а можно не указывать. Соответственно, объект получится полупривязанным или непривязанным;

– в **Базисе** непривязанные объекты можно полупривязать. Для этого нужно выделить один или несколько объектов (с помощью клавиши Ctrl), нажать кнопку  и выбрать нужный сервер-источник. В Полисе полупривязку можно произвести через меню;

– привязать объект можно только в **Полисе**. Для этого нужно перетащить значок объекта на нужный алгоблок или просто в пустое поле задачи (в последнем случае алгоблок будет автоматически добавлен);

– отвязать привязанный или полупривязанный объект можно в **Базисе** (кнопкой ) или **Полисе** (через меню). **Базис** поддерживает групповую отвязку;

– если объект был привязан к алгоблоку, то при удалении этого алгоблока (в **Полисе**) объект станет полупривязанным. Если же удалить целиком контроллер (в **Администраторе БД**), объект станет непривязанным;

– включение объекта в родительский объект (а также исключение) производится в **Базисе** (форма **Объекты**, закладка **Составные объекты**).

Если объектный тип ассоциирован с моделирующим алгоритмом, то объект этого типа можно, помимо привязки к управляющему алгоблоку, привязать к моделирующему алгоблоку. Эти две привязки выполняются независимо. В дереве проекта в **Базисе** для каждого объекта отображаются 2 значка с буквами **У** (управление) и **М** (модель).

5.9 Монопараметрические объекты

Как уже говорилось, станции верхнего уровня СУРЫ (терминал, архивная и т.д.) работают только с параметрами объектов, а не с входами или выходами алгоблоков. Объектный алгоритм можно привязать к объекту, при этом его основные входы и выходы становятся доступными как параметры объекта.

Иногда возникает необходимость представить на верхнем уровне вход или выход необъектного алгоритма. Например, мы складываем в контроллере значения двух аналоговых датчиков, и сумму хотим вывести на операторской станции. Сумма является выходом алгоритма **Сложение**. Алгоритм этот необъектный, и его ни с каким с объектом связать нельзя.

Для решения подобных задач введены специальные объектные типы, называемые **монопараметрическими**: **ПараметрЛогический**, **ПараметрЦелый** и т.д. Каждый такой тип

имеет один единственный параметр - **Значение**. Монопараметрический объект нельзя привязать к алгоблоку, зато его можно привязать к входу или выходу любого алгоблока, принадлежащего либо контроллеру, либо моделирующему серверу. Тип объекта должен при этом соответствовать типу входа/выхода, как описано в таблице 14.

Таблица 14 – Типы объектов

Тип объекта	Тип входа или выхода
ПараметрДействительный	Д (Действительное число)
ПараметрЦелый	Ц (Целое число)
ПараметрЛогический	Л (Логическое значение)
ПараметрВекторный	В (Вектор)
ПараметрСтроковый	С (Строка)

Если вход или выход является перечислением или набором, к нему нельзя привязать монопараметрический объект, но можно раскрыть его и привязать **ПараметрЛогический** к его элементу. На рисунке 9 объект «Перерасход» привязан к признаку выхода интеграла на верхний порог в необъектном алгоритме «Интегратор»:

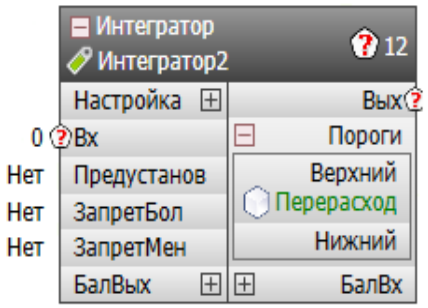

Рисунок 9 – Пример привязки объекта к входу

Если монопараметрический объект привязан к выходу, его значение на операторской станции доступно только на чтение. Если же монопараметрический объект привязан к входу, его значение доступно также и для записи, но только если вход не является связанным. Связанные входы постоянно получают свое значение с некоторого выхода, поэтому изменять это значение с операторской станции бессмысленно. Таким образом, монопараметрические объекты можно применять и для ввода значений, например, для коррекции параметров расчетов.

Монопараметрические объекты обеспечивают самый простой способ отображения произвольного входа или выхода на верхний уровень СУРЫ. Но самый простой способ не всегда является самым эффективным. Если вам приходится часто добавлять монопараметрические объекты, возможно, лучше описать новый составной тип и связать вход или выход с параметром этого типа. Этот подход описан в следующем разделе.

5.10 Составные объектные типы

5.10.1 Назначение

В СУРЕ составные типы делятся на **слабые** и **сильные**. Слабые типы создаются только для отображения его на экранах АРМ. Он содержит внутри себя только **подобъекты** с их ассоциированными библиотечными алгоритмами. Сильный тип – это, по существу, описываемый проектантом объектный тип со своим набором параметров, сигналов, своими ассоциированными алгоритмами. Сильные типы предоставляют проектанту практически столь же полный набор возможностей, как разработчику фирменных библиотек СУРЫ. При желании проектант может отказаться от использования встроенных простых типов, полностью заменив их объектными типами и алгоритмами собственной разработки.

Для чего же нужны составные типы? Единственная, по существу, причина их использования – это сокращение количества рутинных операций при проектировании и уменьшение

связанных с ними ошибок. Это касается и технологического программирования, и моделирования, и создания изображений.

Если вы обнаруживаете, что в проекте приходится многократно повторять некую конструкцию, возможно, это повод создать макрос, либо составной тип. Если конструкция проявляется только в **Полисе**, и входящие в нее алгоблоки не связаны с объектами, достаточно будет макроса. Если конструкция проявляется только в **Контуре** при создании изображений, следует создать слабый составной тип. Если же конструкция затрагивает и технологическую программу, и объекты, и изображения – нужно создавать сильный составной тип.

Создание составного типа – достаточно сложный процесс, требующий опыта и аккуратности. Но результат, как правило, окупает затраченные усилия.

5.10.2 Создание

Составные типы описываются в **Администраторе БД** (форма **Составные типы**). Создать новый составной тип несложно: нужно лишь дать ему уникальное имя, определить, сильный это тип или слабый и указать ассоциированный управляющий и моделирующий алгоритм (алгоритмы указываются только для сильного типа).

Моделирующий алгоритм указывать не обязательно: это нужно только в том случае, если создаваемый тип описывает реальное оборудование.

Ассоциируемые алгоритмы должны быть выбраны из числа пользовательских алгоритмов (макросов) этого проекта. Если алгоритмы еще не созданы, диалоговое окно Мастера позволяет создать их одновременно с составным типом.

Составной тип может включать элементы трёх видов: **подобъекты**, **параметры** и **сигналы**. Слабый тип может содержать только подобъекты.

5.10.3 Подобъекты

Рассмотрим некоторое устройство, например, холодильник. Каждый холодильник имеет компрессор и (в том или ином виде) датчик температуры. Формально описывая холодильник как тип объекта, мы можем сказать, что он имеет 2 подобъекта (таблица 15):

Таблица 15 – Примеры подобъектов

Имя подобъекта	Тип подобъекта
Компрессор	Двигатель
ТемператураВКамере	ВводАналоговый

Таким образом, каждый подобъект имеет имя и тип. Имена подобъектам даются произвольно (лишь бы они были понятны и не совпадали). Типы подобъектов выбираются из имеющихся типов (простых или составных), в частности, можно выбрать тип из подключённого проекта. Имеются следующие ограничения на тип подобъекта:

– тип подобъекта не может совпадать с типом родителя или с типом предка в любом поколении (это привело бы к зацикливанию описаний типов);

– сильный составной тип не может иметь подобъектов слабого типа;

– сильный составной тип не может иметь подобъектов монопараметрического типа (вместо них удобнее использовать параметры).

Если составной тип – сильный, то каждый его подобъект нужно привязать к алгоблоку, входящему в состав ассоциированного макроса. Эта процедура похожа на привязку объекта к алгоблоку, принадлежащему задаче. Все подобъекты видны в **Полисе** в окне **Обзор проекта**. Достаточно перетащить значок подобъекта на нужный алгоблок или просто в пустое поле редактора макроса (в последнем случае алгоблок будет автоматически добавлен).

Если составной тип ассоциирован еще и с моделирующим алгоритмом, нужно дополнительно привязать его подобъекты к алгоблокам из моделирующего алгоритма. Эта процедура выполняется точно так же.

5.10.4 Параметры

На станциях верхнего уровня доступны все параметры всех подобъектов составного типа. Эти параметры обозначаются как **<Имя подобъекта>.<Имя параметра подобъекта>**, например, **ТемператураВКамере.Значение** или **Компрессор.Состояние**.

Однако подобъекты не всегда могут содержать всю необходимую на верхнем уровне информацию. На этот случай предусмотрена возможность добавления параметров в составной тип. Например, для **Холодильника** могут понадобиться параметры, перечисленные в таблице 16.

Таблица 16 - Примеры дополнительных параметров

Имя параметра	Тип	Назначение	Доступ
Задание	Число	Желаемая температура. Выставляется оператором	Чтение+Запись
Перегрузка	Логический	Устанавливается в **Да**, если компрессор работает больше часа подряд	Чтение
Разморозить	Логический	Принудительно отключает компрессор на 3 часа	Запись
ТемператураВнешняя	Число	Комнатная температура. Этот параметр используется только для имитации и связан не с управляющим, а с моделирующим алгоритмом	Чтение+Запись

Итак, для добавляемого параметра нужно задать:

– имя, уникальное среди параметров, принадлежащих этому составному типу;

– тип значения (см. Типы параметров);

– источник (Контроллер или Модель);

– доступ (Чтение, Запись, Чтение + Запись).

Если параметр – управляемый (т.е. имеет доступ Запись или Чтение + Запись), то для него дополнительно указывается тип права, которым должен обладать пользователь, чтобы изменить значение параметра. Для большинства параметров используется тип права «Управление объектами».

Далее, каждому параметру должен быть назначен вход или выход управляющего, или моделирующего алгоритма. В диалоговом окне добавления параметра можно назначить параметру, имеющийся вход/выход, либо создать для параметра новый вход/выход.

Тип этого входа/выхода должен коррелировать с типом параметра согласно, данным, приведенным в таблице 17.

Таблица 17 – Типы входов/выходов и соответствующие им типы параметров

Тип параметра	Тип входа или выхода				
	Д	Ц	Л	В	С
Число	●	●			
Целый		●			
Логический			●		
Состояние			●		
Время	●				
Строка					●
Набор бит				●	

Наличие качества на входе/выходе не влияет на возможность назначения. Объектные параметры всегда имеют признак качества. Если назначенный параметру вход/выход качества не имеет, параметру будет присвоено качество **Хор** (но при передаче параметра по сети его качество может ухудшиться).

Особенности назначения входов и выходов зависят от того, какой доступ имеет параметр.

Если параметр доступен только на чтение, ему можно назначить любой вход или выход (подходящего типа). Например, для параметра **Перегрузка** следует создать одноименный выход управляющего алгоритма.

Если параметр доступен только на запись, ему можно назначить только вход (на этот вход будет передаваться установленное оператором значение).

Если параметр доступен на чтение и запись, есть 2 возможности:

– для чтения и записи используется один и тот же вход алгоритма;

– для записи используется вход, а для чтения – выход того же алгоритма.

Рассмотрим, например, параметр **Задание** для **Холодильника**. Этот параметр доступен для чтения и записи. Самое простое – использовать **возможность 1**, т.е. добавить в управляющий алгоритм вход **Задание** и назначить его для чтения и записи параметра. Этот вариант годится, если нет никаких ограничений на изменения задания. Но предположим, что в некоторые моменты времени изменение задания нужно запрещать на уровне технологической программы. Тогда алгоритму дополнительно потребуется вход **Запрет** и выход **ПринятоеЗадание** (связанные формулой:

if not Запрет then ПринятоеЗадание := Задание).

Для параметра **Задание** следует использовать **возможность 2**: записывать его значение на вход **Задание**, а считывать – с выхода **ПринятоеЗадание**.

Отдельного рассмотрения требует случай командных входов и выходов. Назначить параметру командный выход нельзя (это лишено смысла), а командный вход – можно, но только для записи. Так, для **Холодильника**, следуя **возможности 2**, назначим для записи параметра **Задание** одноименный вход **Задание**, но придадим этому входу командный характер. Разница состоит в возможности одновременного управления Холодильником из технологической программы и с операторской станции (таблица 18)

Таблица 18 – Управление входами различного типа

Характер входа	Наличие связи на входе	Управление из ТП	Управление с ОС
Потенциальный	Связан	Возможно	Невозможно
	Не связан	Невозможно	Возможно
Командный	Связан	Возможно	Возможно
	Не связан	Невозможно	Возможно

Видно, что в случае командного входа можно управлять заданием как из ТП, так и с ОС.

При назначении параметрам командных входов действует то же правило соответствия типов, как и для потенциальных входов. Единственным отличием является дополнительная возможность: параметр логического типа, имеющий доступ только на запись, можно связать с командным входом без данных. Примером является параметр **Разморозить** – ему следует назначить одноименный командный вход без данных.

Если источником параметра является модель, параметру назначается вход или выход моделирующего алгоритма. Например, с помощью параметра **ТемператураВнешняя** можно тестировать работу алгоритма **Холодильник_Управление** при изменении температуры в помещении.

5.10.5 Диапазоны

В разделе 5.5.3 настоящего руководства говорилось о необходимости задания диапазонов для параметров типа Число. Всё сказанное в том разделе справедливо и для составных объектных типов.

Нередко возникает ситуация, когда объектный тип имеет 2 или более числовых параметра, диапазоны и единицы измерения которых заведомо должны быть идентичными в силу того, что они относятся к одной и той же физической величине. Примером могут служить уставки и значение в объекте **ВводАналоговый**.

Чтобы в **Базисе** не нужно было вводить несколько раз одни и те же данные для таких параметров, в СУРЕ используется понятие главного и подчиненного параметра. При описании объектного типа можно для объектного параметра X указать в качестве главного другой объектный параметр Y. Параметр X становится при этом подчиненным. Диапазоны и единицы измерения для X теперь задавать не нужно – они будут браться из параметра Y. Так, в объекте **ВводАналоговый** каждой уставке в качестве главного назначен параметр **Значение**. Итак, мы имеем:

Правило. Для подчиненных параметров диапазоны не задаются, а берутся из соответствующего главного параметра.

5.10.6 Сигналы

В состав сигналов для составного типа автоматически добавляются сигналы его подобъектов. Например, объектный тип Холодильник будет иметь сигналы, приведенные в таблице 19.

Таблица 19 – Пример сигналов объекта Холодильник

Входные сигналы	Выходные сигналы
ТемператураВКамере.А Компрессор.Включен Компрессор.Отключен Компрессор.ВысНапряж Компрессор.ОперНапряж Компрессор.Ток	Компрессор.Включить Компрессор.Отключить

Кроме этих автоматически формируемых сигналов, в составной тип можно добавить сигналы, не связанные с подобъектами. Для каждого сигнала нужно указать его направление: входной или выходной. Входному сигналу назначается вход управляющего и выход моделирующего алгоритма. Выходному сигналу назначается выход управляющего и вход моделирующего алгоритма.

5.11 Составные объекты

Для краткости будем называть объект составного типа **составным объектом**. Будем также использовать термины **сильный объект** и **слабый объект** для объектов **сильного** и **слабого** составного типа.

Прежде чем давать формальные описания, приведем одну метафору. Допустим, мы описали **сильный** составной тип **Квартет** следующим образом (таблица 20):

Таблица 20 – Пример описания составного объекта сильного типа

Имя подобъекта	Тип подобъекта
Скрипка1	Скрипач
Скрипка2	Скрипач
Альт	Альтист
Виолончель	Виолончелист

Введем теперь экземпляр объекта типа **Квартет** и назовем его, например, **КвартетБородина**. Этот составной объект будет являться лишь фантазией, пока мы не введем его участников (таблица 21).

Таблица 21 – Пример описания исполнителей для подобъектов

Имя подобъекта	Исполнитель
Скрипка1	Ростислав Дубинский
Скрипка2	Владимир Рабей
Альт	Рудольф Баршай
Виолончель	Валентин Берлинский

То есть, для каждого подобъекта мы должны указать **исполнителя**. Важно, что каждый исполнитель – это самостоятельный объект в нашей БД, для него можно задавать имя и другие атрибуты.

Дополним эту метафору привязкой объектов. Уподобим непривязанный объект безработному музыканту, а привязанный – получившему контракт. Если музыкант входит в состав квартета, он не может получить контракт сам по себе, но, когда контракт заключается с квартетом, все четверо исполнителей сразу получают работу. Аналогично с объектами, входящими в составной объект: индивидуально привязать или отвязать их нельзя, но при привязке составного объекта они получат привязку автоматически (а при отвязке будут автоматически отвязаны).

5.11.1 Ввод составных объектов

Составные объекты добавляются в **Базисе** таким же образом, как и простые объекты. (Импортировать составные объекты из Excel нельзя.) При выборе типа доступны все составные типы, описанные как в этом проекте, так и в подключённых проектах.

Просто добавить составной объект недостаточно: нужно для каждого подобъекта задать **исполнителя роли подобъекта** в виде ссылки на другой объект соответствующего типа. Исполнители задаются в подвале формы **Объекты** на закладке **Составные типы**. В качестве исполнителя для каждого подобъекта можно либо выбрать ранее введенный объект, либо полуавтоматически создать новый объект. Допускается оставлять некоторые из подобъектов без исполнителей.

Будем также говорить, что объект **А** входит в составной объект **Б**, если **А** исполняет роль одного из подобъектов **Б**. Один объект не может входить сразу в несколько сильных объектов, но может входить сразу в несколько слабых объектов.

5.11.2 Привязка составных объектов

Сильный объект можно привязывать и полупривязывать, как это было описано в разделе Привязка объектов. При этом все входящие в него объекты будут автоматически (полу)привязаны. Удобно ввести

Определение - Объект называется суперпривязанным, если он входит в состав сильного объекта.

Суперпривязанный объект нельзя привязать или отвязать индивидуально (можно только исключить из состава сильного объекта).

В **Базисе** суперпривязанные объекты легко отличить: в графе **Контроллер** для них указан текст **<Родительский>**. Список всех суперпривязанных объектов можно увидеть, переключившись в режим отбора **по контроллерам** и выбрав в дереве контроллеров узел **Суперпривязанные объекты**.

Слабый объект нельзя ни привязать, ни полупривязать. Зато входящие в него объекты допускают индивидуальную привязку.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Вх. № сопроводи-тельного документа и дата	Подпись	Дата
	изме-ненных	заме-ненных	новых	аннули-рованных					