УДК 681.58.011.56 А.С. Гристан, В.Г. Джулгаков, К.И. Руденко, В.Г.Сухоребрый

Среда разработки объектного ПО АСУТП

Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ»

Предложен проект среды разработки программного обеспечения (ПО) распределенной АСУТП, которая поддерживает новую технологию, базирующуюся на полномасштабном использовании объектно-ориентированного проектирования. Эта технология включает использование независимых от конфигурации АСУ объектов, которые допускают децентрализацию и настройку под конкретный проект. Она применяет объектный подход для автоматизации традиционно ручных этапов проектирования структуры локальной сети и определения потоков сетевых данных.

Ключевые слова: ПО АСУ ТП, среда разработки ПО, объектная модель ПО, управляющие компоненты, конфигурация АСУ, объектно-ориентированное программирование

Введение

Цель настоящей работы – определить концепцию построения, структуру и интерфейсы среды разработки ПО АСУТП, в рамках которой разработка программ распределенной АСУ будет реализована как построение структуры локальной сети из объектов аппаратуры и размещение на ней элементов программных объектов. Все объекты, программные и аппаратные – это настраиваемые экземпляры библиотечных классов.

В работе [1] предложен новый подход к применению объектно-ориентированного программирования (ООП) для разработки ПО АСУ ТП. Его суть заключается в использовании распределяемых управляющих компонентов (РУК) – объектов, охватывающих полный контур управления и не зависящих от структуры распределенной системы. При распределении элементов объекта РУК по сети часть его внутренних связей превращается в сетевые, а соответствующие атрибуты – в данные, передаваемые между узлами сети.

В этой работе проведен анализ возможности и особенности систематического применения методики ООП и сформулированы требования к среде, поддерживающей новую технологию. Новая среда разработки должна обеспечить решение следующих основных задач:

- создание библиотеки классов РУК, адекватной определенной прикладной области;
- построение локальной сети из объектов аппаратуры и размещение на ней элементов РУК;
- автоматическая генерация сетевых потоков данных и программ распределенной системы в соответствии с ее структурой и содержанием, определенными двумя первыми задачами.

Ниже будем пользоваться набором понятий и этапами разработки, которые определены в работе [2].

Визуализация

Для визуализации элементов объектной модели АСУТП предлагается применить общепринятый способ, который можно считать негласным стандартом. Это две связанные между собой формы:

- дерево, на котором показана «вертикальная» структура одного или нескольких экземпляров рассматриваемого понятия;
- множество «горизонтальных» представлений, каждое из которых соответствует одной вершине дерева и отображает либо связи между вершинами следующего уровня, либо свойства текущей вершины.

В табл. 1 показано, каким образом элементы объектной модели и другие составляющие проекта ложатся на эти формы.

Отметим, что элемент конфигурации представляет собой объект оборудования, поэтому он имеет двойственное «горизонтальное» представление: как элемент конфигурации он может иметь схему коммуникаций (если вложенные элементы связаны между собой), а как объект обладает набором свойств. То же самое можно сказать и о таких программных элементах, как метод класса, процедура и функция.

Таблица 1

Элементы модели	«Вертикаль»	«Горизонталь»
Классы	Наследование	
Класс	Атрибуты и методы	Схема класса
Атрибут, переменная, константа, параметр составного типа, объект	Внутренняя структура	Свойства
Атрибут, переменная, константа, параметр простого типа		Свойства
Метод, процедура, функция	Параметры, локальные данные, вложенные процедуры и методы	FBD-схема, текстовое представление
Типы данных	Классификация	
Пользовательский тип или его элемент	Внутренняя структура	Свойства
Программа	Разделы, глобальные переменные, объекты	
Раздел	Локальные данные, вложенные процедуры и методы	FBD-схема
Конфигурация, элемент конфигурации	Внутренняя структура	Схема коммуникаций

Необходимо, чтобы среда позволяла редактировать обе формы представления и обеспечивала синхронность: изменение одной формы должно автоматически отслеживаться на другой. Если элемент имеет два «горизонтальных» представления, они должны отображаться оба, возможно, на разных страницах панели.

Анализ табл. 1 приводит к списку необходимых редакторов, который использован ниже.

Этапы разработки и инструменты

В табл. 2 этапам разработки, изложенным в работе [2], ставятся в соответствие структурные компоненты среды.

Как видно из этой таблицы, номенклатура редакторов относительно невелика: благодаря «всепроникающему» объектному подходу редактирование РУК,

задач, устройств, сигналов и типов данных обеспечивает один редактор – редактор объектов. Поэтому он используется на всех этапах, кроме последнего.

Последний этап состоит из операций, выполняемых автоматически по командам пользователя.

Удельный вес первого этапа, разработки и/или пополнения библиотек снижается по мере накопления опыта разработок в определенной области. В идеале, после создания адекватных библиотек, он может вовсе отсутствовать. Этап создания монопрограммы для отладки на одном процессоре, как отмечалось в работе [2], не обязателен – после создания необходимых объектов РУК можно сразу переходить к их распределению по устройствам сети (децентрализации). Однако в определенных случаях отладка монопрограммы с использованием комфортабельного визуального отладчика может существенно сократить общий цикл разработки проекта.

Таблица 2

Этап	Содержание работы	Компоненты среды
Пополнение библиотеки	Определение структуры класса Разработка методов. Создание тестовых объектов и отладка	Редактор классов, редактор программ, редактор объектов, автономный отладчик
Создание объектов РУК	Создание и настройка РУК для данного проекта	Редактор объектов
Построение монопрограммы	Разработка разделов монопро-граммы с использованием членов созданных РУК и распределение их по задачам. Отладка монопрограммы на одном процессоре с использованием моделей	Редактор программ, редактор объектов, автономный отладчик
Разработка конфигурации	Создание объектов оборудова-ния, вставка в конфигурацию и размещение в них задач с разделами. Заполнение разделов частями РУК или монопрогра-ммы	Редактор конфигураций, редактор объектов, редактор программ
Получение исполняемых модулей	Натурализация элементов РУК или монопрограммы, генерация программ на выходном языке, компиляция и компоновка	Генератор, компилятор и компоновщик

Таким образом, при устоявшейся технологии разработки и наличии хороших библиотек разработка нового проекта сводится, по существу, к двум этапам – созданию и настройке объектов РУК на базе библиотечных классов, и их децентрализации в распределенную систему.

По характеру работ первые три этапа табл. 2 можно объединить в одну подсистему, скажем, *программатор*, а два оставшихся – в другую подсистему, *конфигуратор*.

Рассмотрим функции компонентов среды более подробно.

Программатор

Здесь не будем подробно останавливаться на требованиях к таким традиционным компонентам среды разработки ПО АСУТП, как редактор программ и автономный отладчик. Ясно, что редактор программ должен предоставить средства программирования для всех языков, поддерживаемых средой, а отладчик – обеспечить визуальную отладку на уровне исходного представления программы - графического и текстового.

Редактор классов должен выполнять следующие функции:

- создание класса на дереве классов с возможностью наследования;
- редактирование схемы класса: создание/перемещение/удаление атрибутов и методов;
- определение и настройка атрибутов с использованием редактора объектов;
- определение и программирование методов с использованием редактора программ;
- определение и программирование специальных методов (пользовательских конструктора и деструктора, обработчиков событий) средствами редактора программ;
- автоматическое дополнение схемы класса линиями связи между атрибутами и методами.

Возможный вид окна редактора классов показан на рис. 3

Редактор объектов на самом деле должен редактировать переменные любого типа. Объект – это частный случай переменной.

Функции редактора объектов:

- определение общих свойств переменной (имя, назначение, тип, область действия),
- определение начальных значений переменных любых типов (в том числе классов, массивов, структур, строк, множеств, перечислений) и их элементов любого уровня вложенности;
- возможность выбора начальных значений для множеств и перечислимых типов из списка (простого или древовидного);
- возможность редактирования общих свойств нескольких объектов одновременно.

На рис. 1 показан возможный вид окна редактора объектов. Четыре поля в верхней части окна относятся к объекту верхнего уровня. Поле **Имя** демонстрирует возможность одновременного редактирования нескольких объектов. На рисунке показан момент редактирования свойств объекта второго уровня, а именно сигнала **Prab** типа **TAI**. Переход к объекту/элементу следующего уровня осуществляется двойным щелчком по соответствующему полю значения, которое в этом случае указывает категорию типа (например, класс, массив, структура). Кнопка с изображением стрелки служит для возврата на предыдущий уровень. Свойство **Lin** перечислимого типа **TSensorTypes** снабжено структурированным списком значений.

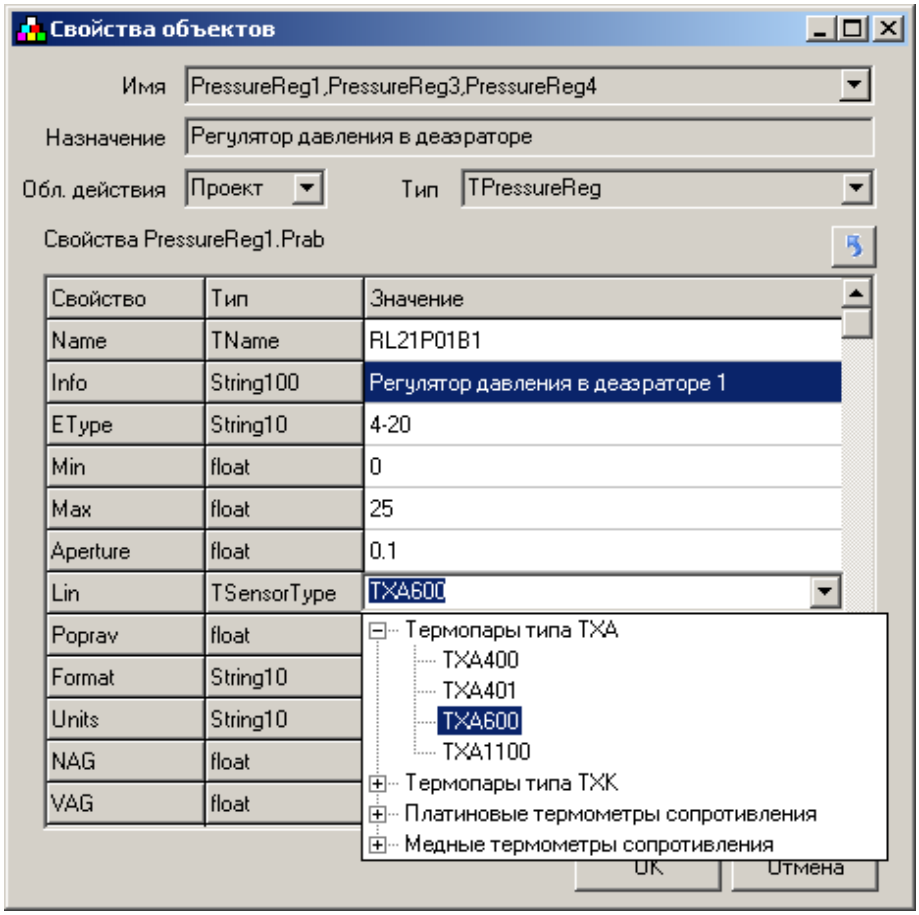


Рис. 1. Окно редактора объектов

Конфигуратор

Конфигуратор отвечает за следующие задачи новой среды: построение дерева конфигурации и заселение его частями РУК. Другими словами, за сборку программно-аппаратного комплекса из готовых настраиваемых объектов.

В соответствии с принятой объектной моделью и способом ее визуализации дерево структуры представляет собой список объектов верхнего уровня, например шкафов управления. Каждый объект раскрывается в структуру вложенных элементов (аппаратных и программных).

Выбор на дереве объекта, имеющего схему, вызывает ее отображение. На схеме в виде вершин показаны все элементы следующего уровня. Схема в отличие от структуры, содержит дополнительную информацию: она представляет связи между элементами, а также внешние связи рассматриваемого объекта.

Интерфейс любой операции над элементами конфигурации может использовать как элементы дерева структуры, так и элементы схем, причем для любой операции допускается смешанный выбор: часть операндов указывается на дереве, часть – на схеме.

Поскольку конфигурация полностью представляет разрабатываемый программно-аппаратный комплекс, конфигуратор должен обеспечить удобный доступ ко всем ее компонентам. Например, в ответ на двойной щелчок по разделу задачи, вставленной в процессор на дереве, на панели схем появляется FBD-схема[3] этого раздела, нарисованная редактором программ и доступная для редактирования.

Процесс построения конфигурации заключается в последовательном создании объектов библиотечных классов (аппаратных и программных), настройке их свойств и переносе на дерево структур. Параллельно среда разработки строит иерархию схем и генерирует пакеты сигналов, передаваемых по линиям связи.

Несмотря на автоматическую генерацию связей между частями объектов и построение схем всех уровней конфигурации, следует сохранить возможность «ручного» создания связей между элементами конфигурации и средства редактирования схем.

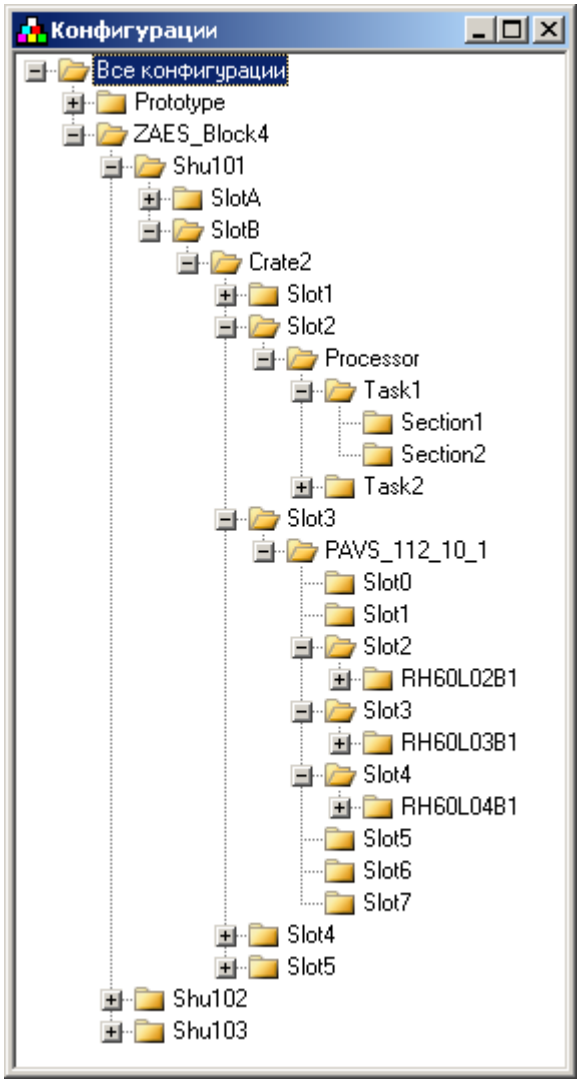


Рис. 2. Пример конфигурации

При построении конфигурации используются:

- окно конфигуратора;
- одно или несколько окон программатора с библиотеками классов.

Пользователь создает очередной объект нужного класса (если класс отсутствует, то он создается по мере необходимости), и размещает его (полностью либо по частям) на дереве структуры конфигурации. При размещении вложенных объектов в контейнерах используются «слоты».

«Слоты» сигналов заполняются сигналами автоматически. При этом среда сообщает о наличии или нехватке свободных мест, чтобы пользователь мог добавить в структуру нужные устройства.

На рис. 2 приведен пример дерева конфигурации с именем ZAES_Block4. Эта конфигурация включает три шкафа управления (Shu101, Shu102, Shu103). Shu101 имеет два слота - SlotA и SlotB. SlotB содержит Crate2, в котором размещены платы процессора с программными объектами и платы модулей связи.

Модель процессора как контейнера программных объектов не содержит слотов.

Показана начинка одного из модулей (PAVS_112_10_1) сигналами типа AIN.

Следует отметить, что в отличие от обычного дерева структуры, отображаемого, например, редактором классов, здесь ряд атрибутов замаскирован. Отображаются только слоты, причем их внутренняя структура также замаскирована, за исключением атрибута Content.

Построение «аппаратной» части дерева конфигурации особых проблем не вызывает. Пользователь «перетаскивает» созданный объект на выбранный слот нужной вершины дерева и настраивает его свойства с помощью редактора объектов, а конфигуратор регистрирует его на данном слоте, отображает на дереве и создает вершину на схеме.

Размещение на дереве частей РУК и других программных объектов требует от конфигуратора более «интеллектуальных» действий.

В результате определения класса среда получает информацию об использовании атрибутов и их членов методами класса. При распределении объекта между процессами и/или процессорами она может определить все разорванные связи и создать в каждом узле частный класс и его объект, который содержит только необходимые члены распределенного объекта. В тех процессорах, где память не является критическим ресурсом, можно создавать не частные, а полные исходные классы с блокировкой вызова неиспользуемых методов.

Рассмотрим пример. На рис. 3 показан исходный класс до распределения его членов по локальной сети. Слева показана структура, справа – схема класса. Он включает 3 метода и 5 полей. Одно из полей, Field1 является объектом класса Class2, который, в свою очередь, включает 2 поля. Первое из них, FieldA, используется методом Class1.Method1.

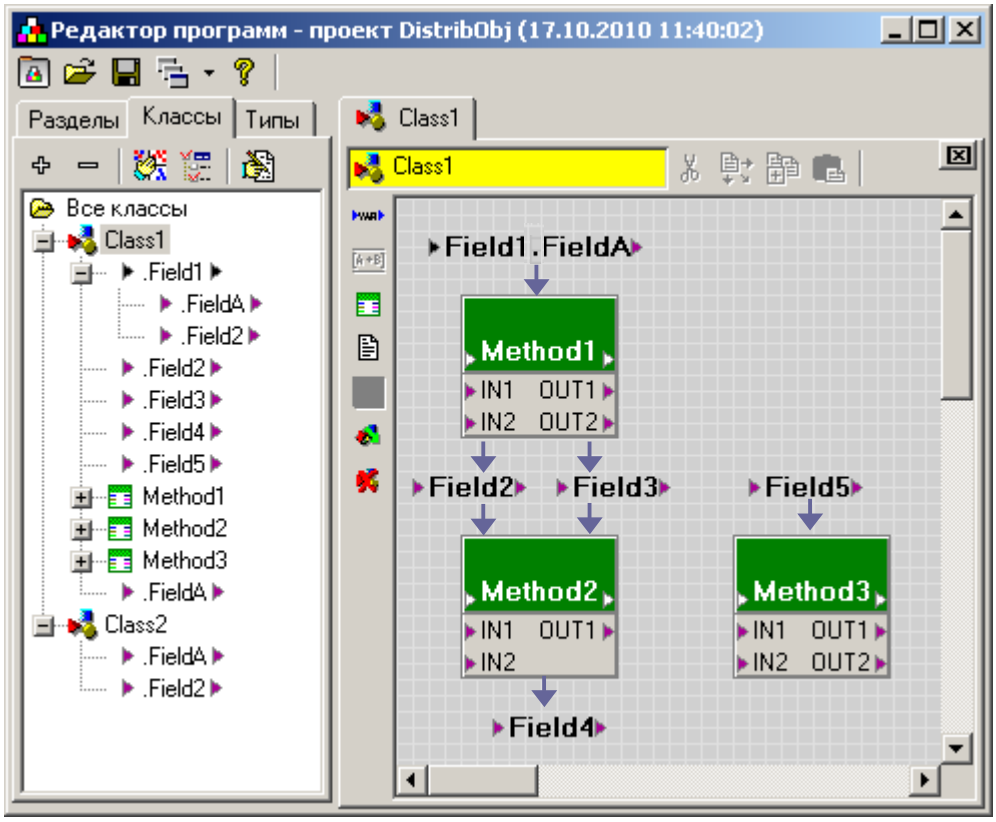
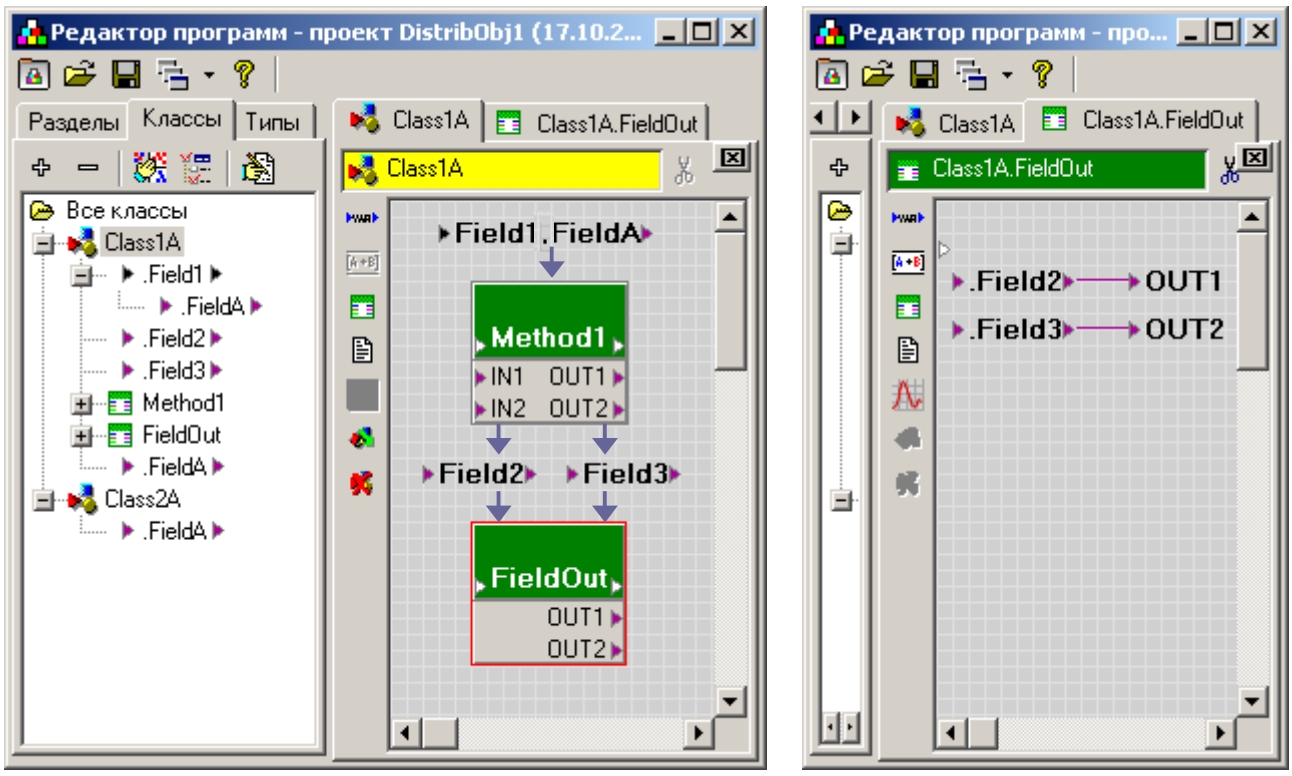


Рис. 3. Исходный класс

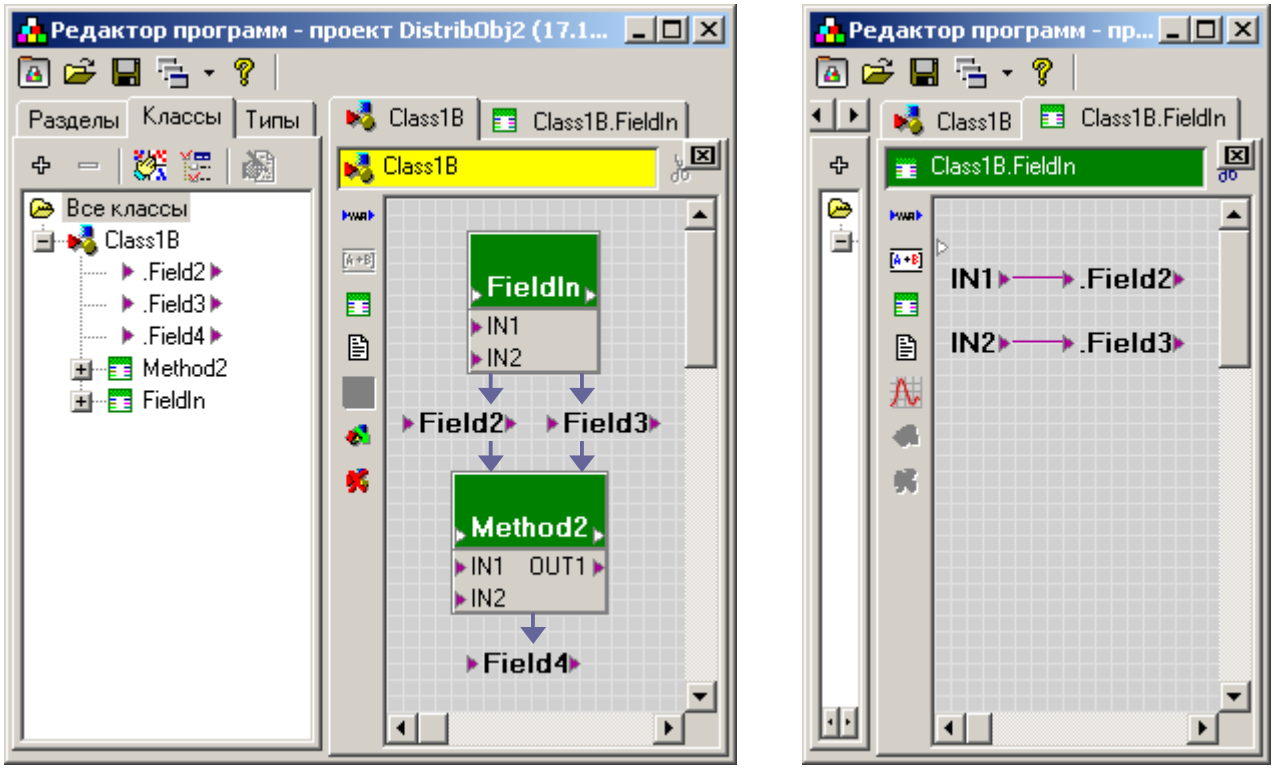
Предположим теперь, что мы хотим распределить объект этого класса, скажем, Obj1, между двумя процессорами. Obj1.Method1 размещаем в узле N1, а Obj1.Method2 – в узле N2. Obj1.Method3 пока не используем.

В таком варианте среда должна создать частные классы Class1A в узле N1 (рис. 4) и Class1B в узле N2 (рис.5), а также соответствующие объекты Obj1A и Obj1B. Кроме того, среда должна заменить разорванные внутренние связи между методами Method1 и Method2 исходного класса сетевой линией связи, по которой от узла N1 в узел N2 будут передаваться значения полей Field2 и Field3 с помощью сигналов SigField2 и SigField3 соответственно.



а б

Рис. 4. Частный класс в узле N1



а б

Рис. 5. Частный класс в узле N2

На рис. 4,а показаны структура и схема класса Class1A. Кроме вполне ожидаемого метода Method1 здесь появился дополнительный метод FieldOut. Его функция – передать вычисленные значения полей Field2 и Field3 на свои выходы Out1 и Out2, к которым в объекте Obj1A будут подключены глобальные переменные SigField2 и SigField3. Схема метода FieldOut показана на рис. 4,б.

На рис. 5,а представлена структура и схема класса Class1B. Здесь кроме исходного метода Method2 появился дополнительный метод FieldIn. Его функция – присвоить значения сигналов SigField2 и SigField3, подсоединенных к своим входам In1 и In2, полям Field2 и Field3 объекта Obj1B. Схема метода FieldOut показана на рис. 5,б.

Отметим, что при создании частных классов среда должна учитывать только необходимые члены. В нашем примере не используемые в конфигурации метод Method3 и поля Field1.Field2 и Field5 не включены в частные классы. При создании частных классов среда должна также собирать частные классы одного процесса, полученные из общего исходного класса в один частный класс.

Приведенный пример иллюстрирует случай, когда одна часть РУК использует атрибуты, вычисленные другой частью. Для обеспечения взаимодействия в этом случае достаточно передать по сети соответствующие сигналы. Среда должна собрать все сигналы, вырабатываемые узлом N1, в один пакет, и передать их узлу N2 после окончания своих вычислений.

Если же одна часть РУК использует методы, размещенные в другой части, потребуется более тесное взаимодействие. Протокол OPC[4,5] подходит для любых случаев, хотя не исключено использование и других стандартных протоколов.

До начала процесса децентрализации РУК пользователь должен позаботиться о «программных слотах» процессоров, то есть создать и разместить в них необходимые объекты задач, а в задачи вставить объекты разделов. Теперь можно наполнять разделы частями РУК.

При определении класса РУК пользователь не обязан предвидеть, какие внутренние связи окажутся в конкретной конфигурации межузловыми. Поэтому для атрибутов, не относящихся к классу TSignal, необходимо предусмотреть автоматический запрос выбора типа сигнала во время децентрализации. В этом диалоге должна предоставляться возможность создания нового класса, если его нет среди ранее созданных. После выбора класса сигнала должно появляться окно редактора объекта для создания и настройки свойств конкретного сигнала.

В результате процесса децентрализации мы должны получить готовые исходные программы АСУ для каждого процессора сети. Получение исполняемых модулей происходит традиционным способом: генерация исходных программ с получением их эквивалентов на выходном языке, затем их обработка компилятором и компоновщиком.

Преимущества

Преимущества среды объектного проектирования перед традиционными средами разработки ПО АСУТП в концентрированном виде иллюстрируются рис. 6.

Четыре блока в верхней части рисунка показывают базовые принципы новой технологии разработки. Они соединены причинно-следственными связями. Пять блоков внизу демонстрируют преимущества среды разработки, вытекающие из базовых принципов.

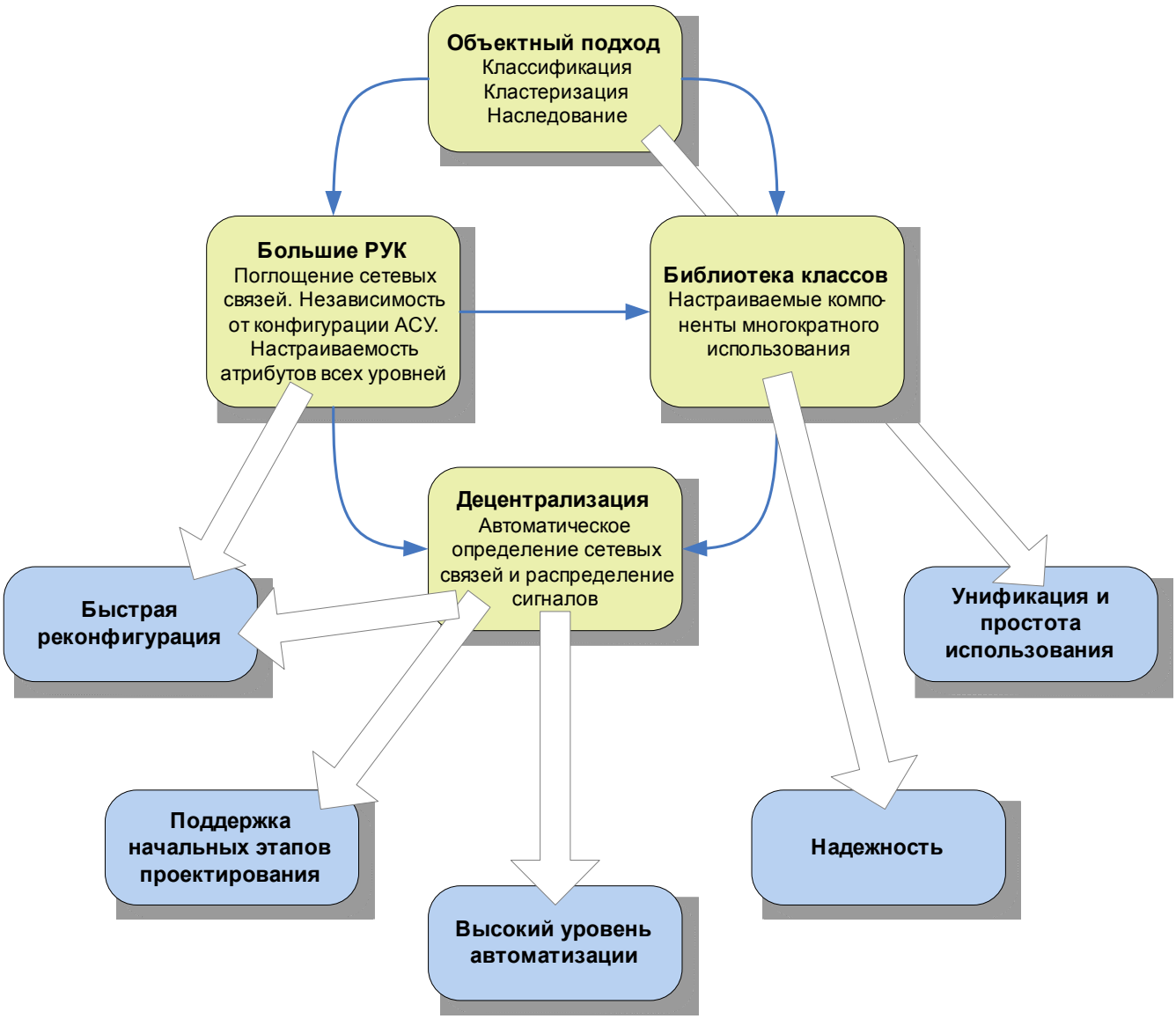


Рис. 6. Преимущества объектной среды разработки

Например, независимость распределяемых управляющих компонентов от конфигурации конкретной АСУ влечет за собой возможность создания библиотеки классов, которая будет пригодна для различных проектов. А применение проверенных элементов многократного использования гарантирует высокую надежность программного обеспечения. Аналогично унификация и простота использования среды объясняются общим объектным подходом к «изготовлению» и модификации компонентов проектируемого АСУ.

Выводы

Предложена концепция среды разработки ПО распределенной АСУТП, которая поддерживает новую технологию, базирующуюся на полномасштабном использовании объектно-ориентированного проектирования. Новая среда разработки обеспечивает решение следующих основных задач:

1. Создание библиотеки классов РУК, адекватной определенной прикладной области.
2. Построение локальной сети из объектов аппаратуры и размещение на ней элементов РУК.
3. Автоматическая генерация сетевых потоков данных и программ распределенной системы в соответствии с ее структурой и содержанием, определенными двумя первыми задачами.

Список литературы

1. Сухоребрый В.Г Объектное проектирование АСУ ТП [Текст]/ В.Г.Сухоребрый, А.С. Гристан, Ю.К. Швыдкий // Мир автоматизации, № 3, сентябрь 2010. – С. 50 - 54.
2. Гристан А.С. Объектная модель программного обеспечения АСУТП [Текст] /А.С. Гристан, В.Г. Сухоребрый // Открытые информационные и компьютерные интегрированные технологии.: сб. науч. тр. Нац. аэрокосм. ун-та «ХАИ». – Вып. 56 - Х. 2012. - С. 116 - 124.
3. International standard 11131-3, Part 3: Programming languages, IEC, Division Automatismes Programmables, First edition, 1993.
4. Интернет–ресурс: http://www.opcserver.ru/products.phtml?more=MODBUS.
5. Интернет–ресурс: http://ru.wikipedia.org/wiki/OPC.

Рецензент: доктор технических наук, профессор зав. каф. И.А. Фурман
Харьковский национальный технический университет сельского хозяйства им. Петра Василенко,

Середовище розроблення об'єктного ПО АСУТП

Запропоновано проект середовища розроблення програмного забезпечення (ПО) розподіленої АСУТП, яка підтримує нову технологію, що базується на повномасштабному використанні об'єктно-орієнтованого проектування. Ця технологія включає використання незалежних від конфігурації АСУ об'єктів, які допускають децентралізацію і настроювання під конкретний проект. Вона застосовує об'єктний підхід для автоматизації традиційно ручних етапів проектування структури локальної мережі і перерозподілу потоків мережних даних.

Ключові слова: ПО АСУ ТП, середовище розроблення ПЗ, об'єктна модель ПЗ, керуючі компоненти, конфігурація АСУ, об'єктно-орієнтоване програмування.

Software development environment object control system

Proposed project development environment (software) and distributed control, which supports the new technology which is based on the full use of object-oriented design. This technology involves the use of independent configuration of ACS objects that allow decentralization and the setting for a specific project. It uses the object approach to automation has traditionally manual steps: designing the structure network and determine the flow of network data.

Keywords: software control system, software development environment, object model, control components, the configuration process control, object-oriented programming.