полнения лучше делать в модели Simulink. Это поддерживает весь проект в актуальном и документированном состоянии через графическую структуру Simulink и Stateflow. Пользователь может транслировать код в машинный язык и переносить его в ПЛК в знакомой среде Automation Studio.

Согласованность рабочего процесса приводит к колоссальному росту эффективности. Любые изменения осуществляются непосредственно на модели и быстро переносятся в систему простым нажатием кнопки. Это существенно сокращает время разработки. Возможность повторного использования существующих моделей Simulink в последующих проектах позволяет машиностроителю экономить и время, и деньги.

Автоматическая генерация кода и проектирование на основе моделей в действии

Многие машиностроители уже используют продукт B&R Automation Studio Target for Simulink. Например, известный в мире производитель пневматических роботов – компания FerRobotics Compliant Robot Technology GmbH убедилась в существенном сокращении времени разработки своей продукции. Ведущий производитель газотурбинных двигателей GE Jenbacher использует автоматическую генерацию кода для тестирования новых типов структур контроллеров. Но есть и такие компании, как ControllerSolution, которые продолжают использовать привычную среду проектирования MATLAB и Simulink для реализации алгоритмов на промышленных контроллерах.

Независимо от субъективных мнений об эффективности проектирования на основе имитационного моделирования эти инновационные методы разработки станут ключом к успеху при создании будущих средств промышленной автоматизации. Доминировать на рынке машиностроения в ближайшие годы будет тот, кто сможет реализовать этот потенциал.

Контактный телефон (495) 657-95-01.
Http://www.br-automation.com

ОА-архитектура построения и моделирования распределенных систем автоматизации

С.М. Салибекян, П.Б. Панфилов (МИЭМ)

Приводится описание применения объектно-атрибутной (ОА) архитектуры вычислительной среды к реализации распределенных систем автоматизации, имеющих в составе вычислительные узлы (компьютеры или ПЛК) различной аппаратной архитектуры. Показаны особенности ОА моделирования распределенных средств автоматизации, основные приемы ОА-моделирования, программирования и отладки подобных систем.

Ключевые слова: объектно-атрибутная архитектура, милликоманда, информационная пара, капсула, параллельная модель программирования.

Введение

В Московском институте электроники и математики в настоящее время ведется разработка новой архитектуры вычислительных систем, относящейся к классу dataflow (управление вычислительным процессом с помощью потока данных), получившей название *объектно-атрибутной* или *ОА-архитектуры* [1-3]. Управление данными в ОА-архитектуре обеспечивается, во-первых, с помощью нового способа описания алгоритма: описываются не команды, а данные, циркулирующие в вычислительной системе (подобное решение достаточно широко применяется на практике); а во-вторых (это уже новшество) – посредством иной организации работы исполнительных (функциональных) устройств, выполняющих вычислительные операции (арифметические, логические и т.д.): устройство принимает не команды, а информационные пары (совокупность данных и описывающего их тега/атрибута, именуемого милликомандой). Отсюда следует и другое название предлагаемой архитектуры – *милликомандная архитектура* в отличие от более традиционных микрокомандных архитектур.

Предлагаемый ОА подход к реализации вычислительного процесса создает множество новых возможностей, включая создание вычислительных систем с децентрализованным управлением вычислительным процессом и реализацию системы на базе вычислительных узлов различной аппаратной архитектуры, программирование системы, состоящей из нескольких разнородных вычислительных узлов как единого целого и обеспечение высокого параллелизма выполнения программ без необходимости прикладывания дополнительных усилий на решение задач распараллеливания вычислений и их распределения по вычислительным ресурсам (которые в большинстве практических случаев не являются тривиальными). Системы автоматизации – одна из сфер применения ОА-архитектуры. В данной области применение ОА-архитектуры обеспечивает множество преимуществ, в том числе предоставляет удобные механизмы проектирования, отладки и имитационного моделирования распределенных систем автоматизации.

ОА-архитектура на примере задачи промышленной автоматизации

Объектно-атрибутная архитектура вычислительной системы (ВС) базируется на следующих понятиях:

- информационная пара (ИП) – совокупность данных (*нагрузки*) и атрибута, описывающего данные (например, {ТемператураПомещения=19}, где "=" – знак сопоставления атрибута ярлыку);

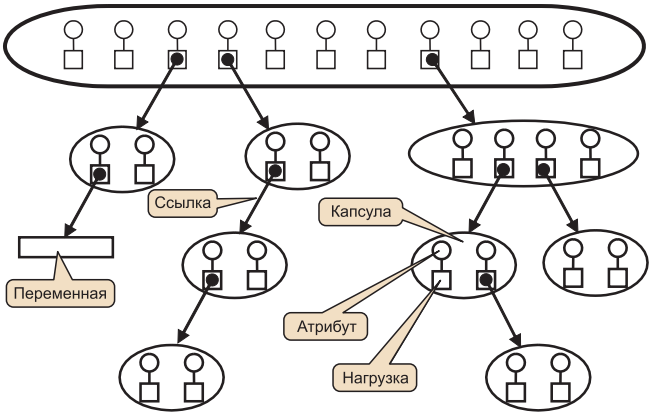

Рис. 1. Дерево абстракций

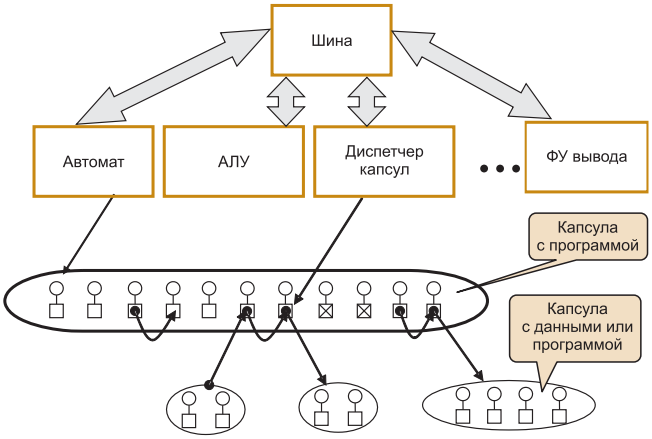

Рис. 2. Вычислительная ОА-среда

- капсула – объединение нескольких ИП, которые описывают один объект (например, {Объект=ящик, Материал=картон, ВысотаВСантиметрах=20, ШиринаВСантиметрах=50, ГлубинаВСантиметрах=40}).

С помощью ИП, сгруппированных в капсулы, можно не только описывать различные объекты, но и задавать алгоритмы и строить сложные абстрактные модели сложных управляемых объектов и систем. Похожие возможности реализованы в популярном ныне объектно-ориентированном программировании (ООП).

Для реализации модели, описывающей некий сложный объект, достаточно в качестве нагрузки ИП определить не только константу, но и ссылку на ячейку оперативной памяти, где хранится капсула, описывающая другой объект. Например, возможна следующая конструкция:

Объект{Название = "ЗаводЭлектроприбор",
Помещения = *ПомещенияЗавод*}
ПомещенияЗавод{Насосная=**НасоснаяОборудование**, ТП=***ТПОборудование***}
НасоснаяОборудование{НасосДренажный=
НасосДренажСостояние}
ТПОборудование{Трансф1=Трансф1Состояние,
Трансф2=Трансф2Состояние}
НасосДренажСостояние{ВклВыкл=Да}
Трансф1Состояние{НапряжениеВольты=220,
ТокАмпер=50,
ТемператураГрад=25}
Трансф2Состояние{НапряжениеВольты=221,
ТокАмпер=60,
ТемператураГрад=23},

где "{...}" – обозначение капсулы; "=" – знак сопоставления атрибута и нагрузки (данные или ссылка на данные). Здесь перед знаком "{" стоит имя капсулы, которое задает ссылку на адрес оперативной памяти, где находится капсула.

С помощью механизма капсул и нагрузок-ссылок можно описывать любой объект (как в ООП), только подобные информационные конструкции не задаются изначально, а синтезируются по определенному алгоритму уже в процессе работы программы: ВС может создавать ОА-конструкции, описывающие еще неизвестный программисту объект с помощью формирования капсул и задания ссылок между ними, что существенно повышает гибкость системы автоматизации и делает ее интеллектуальной. Назовем подобную конструкцию *объектно-атрибутным деревом* или *деревом абстракций*, так как в большинстве случаев смысловое описание сложного объекта является иерархической структурой, представляющей собой граф типа "дерево" (рис. 1).

Основу ВС, построенной в соответствии с ОА-архитектурой, составляют виртуальные функциональные устройства (ФУ), объединенные общей шиной[1] (рис. 2). Информация по шине передается в виде ИП, где атрибут описывает операнд, прикрепленный к нему в качестве нагрузки (в роли операнда может выступать как константа, так и ссылка на переменную или капсулу). Такой атрибут будем называть "милликомандой", поскольку он задает для ФУ инструкцию о том, какую операцию следует проводить с данными. Милликоманды, сгруппированные в информационные капсулы, легко встраиваются в дерево абстракций и превращают его из структуры, пассивно описывающий объект автоматизации, в мощную базу знаний, гибко реагирующую на любые события, фиксируемые системой.

Для выполнения алгоритма в ОА-системе имеется ФУ *Автомат*. Последовательность вычислений в ОА-архитектуре представляется в виде капсулы, например:

{Сумматор1.ПервСлаг = 10,
Сумматор1.ВтороеСлаг = 15,
Сумматор.Результат = ПеременнаяСуммы},

где перед точкой указывается мнемоника, обозначающая определенное ФУ, а после точки – милликоманда, которую должно выполнять данное ФУ. В состав Автомата входит регистр, где хранится ссылка на капсулу с последовательностью милликоманд, описывающих миллипрограмму, и указатель на ИП, выдаваемую в данный момент на шину. *Автомат* последовательно выдает милликоманды из капсулы с программой на *шину*. Например, чтобы выдать результат вычисления ФУ *ЦелочисленноеАЛУ* на консоль (эк-

[1] *Шина* – ФУ, специализирующееся на управлении другими ФУ и передаче информации между ними.

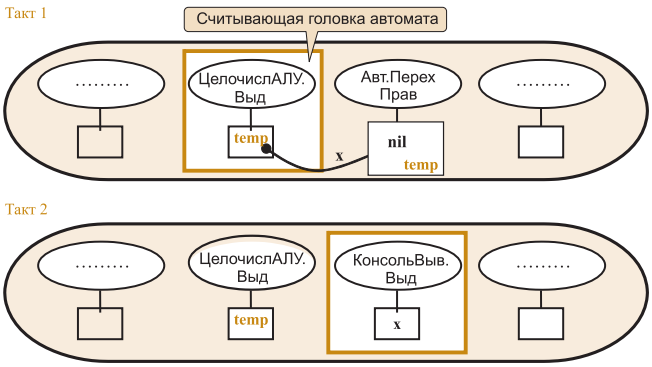

Рис. 3. Работа Автомата (Вывод результата вычислений АЛУ на экран)

ран), следует задать миллипрограмму: {*ЦелАЛУ. РезультатВыдать=temp, Консоль.Вывести=temp(0)*}, где *temp(0)* – адрес нагрузки милликоманды (рис. 3), в скобках указывается первоначальное значение, записанное в нагрузке. На первом такте считывающая головка выдает на шину милликоманду {*ЦелАЛУ. РезультатВыдать=temp*}. *Шина* принимает милликоманду и передает ее для ФУ *ЦелочисленноеАЛУ*, которое записывает результат вычислений в ячейку *temp*, теперь в нагрузку следующей информационной пары будет записан результат вычислений *ЦелАЛУ*. На другом такте считывающая головка *Автомата* передвигается на следующую милликоманду, в нагрузку которой на предыдущем такте уже записано число, которое нужно выдать на консоль. *Автомат* выдает данную милликоманду вместе с нагрузкой на *шину* и ФУ *КонсольВывода* выводит число на экран (рис. 3).

Для осуществления безусловного перехода *Автомат* выполняет милликоманду *Переход*: считывающая головка перемещается по адресу, записанному в нагрузке милликоманды. Для организации условного перехода существуют милликоманды: *УстановитьАдрУсловного-Перехода, ПереходПоПравде, ПереходПоЛжи*. С помощью первой милликоманды во внутренний регистр адреса условного перехода Автомата записывается адрес условного перехода. В нагрузку к двум другим милликомандам дается логическая переменная, определяющая условие перехода. Если переменная в нагрузке к *ПереходПоПравде* имеет значение "Правда", то считывающая головка *Автомата* начинает выборку милликоманд по адресу, который хранится в *регистре адреса условного перехода*. В противном случае перехода не происходит, и считывающая головка на другом такте передвигается на следующую милликоманду. Аналогично переход осуществляется, если в нагрузке к *ПереходПоЛжи* записано "Ложь". Например, следующий фрагмент миллипрограммы выводит сообщение о том, равняется ли число, находящееся в аккумуляторе *ЦелочисленногоАЛУ*, положительным или отрицательным:

{Автомат.АдрЛогПерехУстановить = МеткаОтрицат, ЦелАЛУ.АккумулУстановить= Переменная, ЦелАЛУ.Вычесть=0, ЦелАЛУ.ПризнакМеньшеНуляВыдать=temp, ЦелАЛУ.ПереходПоЛжи = temp(Ложь), ВывКонсоль.Вывод = "Положительное число", Автомат.Стоп, МеткаОтрицат [ВывКонсоль.Вывод = "Отрицательное число"]},

где *ЦелАЛУ.ПризнакМеньшеНуляВыдать* – милликоманда выдачи признака отрицательного результата; "[...]" – обозначение информационной пары (мнемоника, распложенная перед квадратными скобками является обозначением данной информационной пары; МеткаОтрицат[...] – адрес в оперативной памяти, где расположена информационная пара.

ФУ в системе автоматизации делятся на два больших класса: устройства обмена данными с "внешним миром" (считыватели сигналов с датчиков, устройства вывода сигналов управления оборудованием и т.д.) и ФУ, осуществляющие реализацию алгоритма работы системы (сумматоры, умножители, диспетчеры оперативной памяти, ячейки хранения данных и т.д.).

Шина, через которую ФУ осуществляют обмен милликомандами, может быть реализована как программно (виртуальная шина), так и аппаратно, причем физически шина может иметь любую реализацию: внутренняя шина процессора, коммутационная линия на печатной плате, локальная компьютерная сеть, модемная или беспроводная линия и т.д., что позволяет создавать, в том числе и распределенные системы автоматизации любой топологии.

Распределенная модель программирования

В настоящее время объектно-ориентированные языки способны создавать программные модели, функционирующие лишь на одной вычислительной машине. Правда существуют средства передачи параметров объектов по вычислительной сети Corba, DCOM, однако реализация программной модели, которая работала бы как единое целое на нескольких вычислительных узлах, объединенных сетью любой топологии, до настоящего времени не осуществлена. ОА-архитектура такую возможность дает.

Для описания работы распределенной ОА-модели необходимо понять, каким образом в ОА-архитектуре осуществляется синтез абстрактных данных. Синтез начинается с так называемых *атомов* (элементарных абстракций): применительно к системам автоматизации *атом* – это снятый с датчика сигнал, снабженный соответствующим атрибутом, по которому в ОА-системе происходит идентификация данных (рис. 3). Например, *{"Температура помещения №1"=22}, {"Расход воды в центральном трубопроводе"=12.4}, {"Сработка пожарного датчика в холле"=Ложь}* и т.д. Виртуальное *УстройствоСбораДанных* считывает показание датчика, приписывает ему соответствующий атрибут или милликоманду и получившуюся ИП выдает на шину. Последняя передает ИП соответствующему ФУ, которое будет заниматься ее обработкой. Определенные ФУ ОА-системы, получая атомы, осуществляют их анализ и синтез капсул, описывающих абстракции более высокого порядка и управляющие

сигналы для оборудования. Например, пожарные датчики, фиксирующие факт возгорания в помещении, выдают следующие милликоманды для ФУ, контролирующего определенное помещение (*ПожКонтроль*):

{ПожКонтроль.ПринятьСигнал=
{ВозгаранеНомерДатчика=N}},

где *N* – номер пожарного датчика (капсула в качестве нагрузки ИП может задаваться и такой конструкцией: Атрибут={перечисление ИП, входящих в капсулу}).

Получив милликоманды, ФУ *ПожКонтроль* по номеру пожарного датчика определяет помещение, где фиксируется возгорание и передает для вывода на экран милликоманду:

*{Дисплей.Вывести={НомерПомещения=**M**,*
Состояние="Внимание"}}.

Если ФУ *ПожКонтроль* получает сообщение о срабатывании второго датчика, расположенного в помещении ***M***, то оно генерирует следующие абстракции:

*{Дисплей.Вывести={НомерПомещения=**M**,*
Состояние="Пожар"},
СистемаПожаротушения=
*{НомерПомещения=**M**, Команда=Включить}}.*

Виртуальные ФУ, участвующие в синтезе абстракций, и капсулы, полученные в результате такого синтеза, можно расположить в оперативной памяти нескольких вычислительных узлов (компьютеров или контроллеров), объединенных компьютерной сетью или иными коммутационными линиями (рис. 4): вычислительный узел, синтезировав абстракцию, передает ее для обработки на вышележащий уровень информационного дерева, расположенный либо в его оперативной памяти, либо в памяти другого вычислительного узла. Причем программная модель управляемого объекта для программиста представляется единой, и только отдельные части программы выделяются специальными конструкциями ОА-языка, определяющими, в оперативной памяти каких вычислительных узлов будут располагаться ФУ и синтезируемые капсулы. На вершине дерева абстракций находится вычислительная машина, которая реализует АРМ оператора системы. Причем графический интерфейс АРМ организуется также по принципу ОА-архитектуры (графические элементы на экране можно программно реализовать в виде все тех же виртуальных устройств, милликомандами которых будут: изменить цвет, изменить видимость и т.д.), что позволит избавиться от громоздкого OPC-интерфейса, служащего для стыковки системы автоматизации и SCADA-программы [5].

Программист пишет миллипрограмму автоматизации и визуализации полученных данных в единой среде программирования и может воспринимать ее целостно (в современных распределенных системах автоматизации программы создаются для каждого узла совершенно автономно [4-6]). И только на объекте автоматизации части ОА-программной модели распределяются по различным вычислительным устройствам: программа загружается на головную машину, а затем с головной машины необходимые части модели автоматически загружаются на остальные вычислительные узлы системы автоматизации.

Отладка программы и имитационное моделирования распределенной системы автоматизации

Рассмотрим процедуру отладки ОА-среды автоматизации. Программист запускает созданную программную модель объекта на своем компьютере: все виртуальные ФУ и капсулы создаются в оперативной памяти одного вычислительного устройства. Далее для эмуляции сигналов, которые в последствии должны приходить с датчиков, программист задает милликоманды или набор милликоманд, выводит их на шину ОА-среды и наблюдает за реакцией среды на поступающую информацию. Реакцию системы можно отследить по милликомандам, которыми по шине обмениваются между собой ФУ системы (на реальном объекте поставщиками информационных пар в ОА-среду будут датчики: они считывают данные и добавляют к ним соответствующие атрибуты). Для моделирования поведения более сложных объектов автоматизации составляются последовательности милликоманд, задается время поступления ИП в эмулятор системы, пишутся специальные программные модели, эмулирующие работу оборудования (модели оборудования меняют состояние в зависимости от ИП, которые описывают выходные сигналы системы автоматизации).

Например, для моделирования работы системы пожаротушения можно задать милликоманду с прикрепленной к ней капсулой:

{ПожКонтроль.ПринятьСигнал=
{ВозгаранеНомерДатчика=N}}

и выдать ее на шину, а затем через консоль, где отображаются проходящие по шине милликоманды, на-

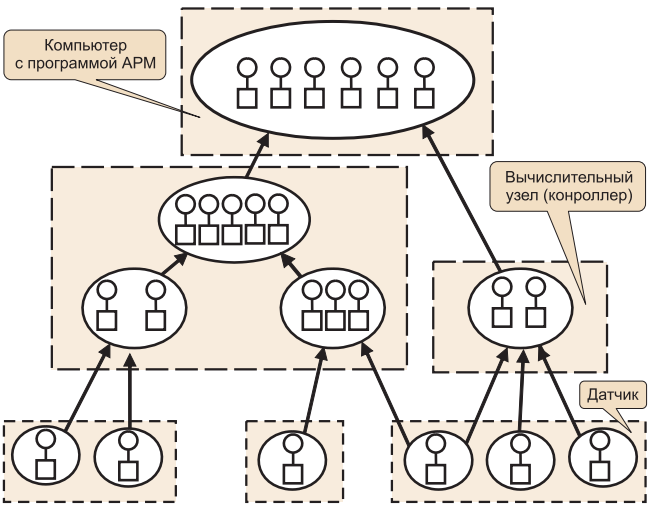

Рис. 4. Дерево абстракций, распределенное по нескольким вычислительным узлам

блюдать за реакцией системы. Если на консоли появится капсула

*{Дисплей.Вывести={НомерПомещения=**M**, Состояние="Внимание"}},*

то это значит, что ФУ *ПожКонтроль* сработало корректно (оно по номеру датчика определило номер помещения, где датчик расположен, и передало на устройство вывода сообщение о состоянии "Внимание").

Далее можно послать на *шину* информационную конструкцию

*ПожКонтроль.ПринятьСигнал= {Возгарание НомерДатчика=**N2**}.*

Если датчик расположен в помещении М, то ФУ *ПожКонтроль* должно выдать:

*{Дисплей.Вывести={НомерПомещения=**M**, Состояние="Пожар"}, СистемаПожаротушения= {НомерПомещения=**M**, Команда=Включить}}.*

Если же ожидаемая ИП на шине не появится, значит в программе допущена ошибка.

Преимущества моделирования работы системы автоматизации, обеспечиваемые ОА-архитектурой:

- существует возможность автономно (без применения средств полунатурного моделирования с подключением оборудования) моделировать и отлаживать распределенные системы автоматизации;

- не требуется заранее описывать информационные конструкции (например, в ООП это структуры данных и объекты): любую конструкцию можно составить "на лету" с помощью ОА-языка, что существенно упрощает процесс моделирования;

- имеется возможность сохранения тестовых примеров в текстовом файле, применения специальных редакторов не требуется;

- во время моделирования существует возможность отследить не только внешнюю реакцию системы автоматизации (то есть вывод на экран сообщений для оператора системы и сигналы управления оборудованием), но и при необходимости проследить внутреннюю логику работы программы, так как на шине отражаются и сообщения между ФУ, реализующими алгоритм работы системы;

- удобен просмотр результатов моделирования: чтобы вычленить необходимую информацию из передаваемого по шине информационного потока, применяются различные фильтры, которые выбирают и отображают на консоле лишь те ИП, которые удовлетворяют заданным условиям;

- повышается наглядность процесса моделирования: все сигналы, снимаемые с датчиков, снабжаются атрибутом, который отображается в виде мнемоники. Таким образом можно легко узнать, что за параметры при моделировании "запущены" в систему и какие управляющие сигналы система будет выдавать во "внешний мир". Аналогично по ярлыкам можно идентифицировать и целые информационные конструкции, которые будут синтезироваться в процессе моделирования;

- механизм дерева абстракций дает возможность проводить разработку и моделирование программы не только целиком, но и по частям: например, сначала создается код только для нижнего уровня абстракций (синтез из информационных атомов), а затем запускается моделирование и выявляется, какие информационные конструкции будут выдаваться на вышележащий уровень. Затем составляется программа для следующего уровня абстракции и проводится моделирование сразу двух уровней и т.д. Возможно проводить и разработку программы сверху вниз, когда сначала пишется верхний уровень абстракции, а поставщиком информации для него служит "заглушка", которая заменяет нижележащий уровень абстракции.

Реализация ОА-архитектуры

Функциональные устройства могут быть реализованы как аппаратно, так и программно (виртуальные ФУ). Аппаратная реализация ОА-архитектуры в виде микросхем специализированных процессоров или электронных устройств является в настоящее время предметом разработок. Основная же экспериментальная работа и отладка прототипов систем ОА-архитектуры проводится в настоящее время на базе ее программной реализации, которая позволяет организовать виртуальную ОА-машину на ВС "традиционной" архитектуры, что существенно уменьшает финансовые затраты на создание экспериментальных образцов.

Для создания единого программного ОА-пространства на всех устройствах, входящих в состав системы, программисту необходимо написать программы реализации логики работы виртуальных устройств и программу виртуальной шины для обмена информацией между ФУ. Во время работы системы виртуальные устройства запускаются на компьютерах, где загружена программа АРМ, и на ПЛК и осуществляют вычисления и обмен информацией между собой. Совместимость различных аппаратных платформ достигается благодаря тому, что логика работы виртуальных устройств не зависит от архитектуры вычислительного узла.

Единое ОА-пространство создается следующим образом: на каждом вычислительном узле системы запускается ОА-платформа и настраиваются интерфейсы обмена информации между соседними узлами. Далее исполняемая программа загружается на головной машине системы, и программные блоки ОА-модели автоматически распределяются по всем вычислительным узлам, составляющим распределенную систему автоматизации, и вся система запускается на выполнение.

ОА-система автоматизации может работать с минимальными затратами на любых аппаратных платформах, подобно языку Java: чтобы перенести ОА-программу на другую платформу требуется лишь написать программы, реализующие алгоритмы функционирования для всех типов ФУ под эту платформу.

Заключение

Работы по развитию и продвижению объектно-атрибутной (милликомандной) архитектуры вычислительных систем ведутся по нескольким направлениям. В настоящий момент создана программная платформа ОА-среды, работающая на отдельной машине: в среде реализованы 35 классов виртуальных ФУ; язык ОА-программирования, с помощью которого осуществляется программирование управления ФУ; имеется возможность не только программирования, но и управления ОА-системой в реальном времени, то есть коррекция алгоритма работы системы без перезагрузки программы.

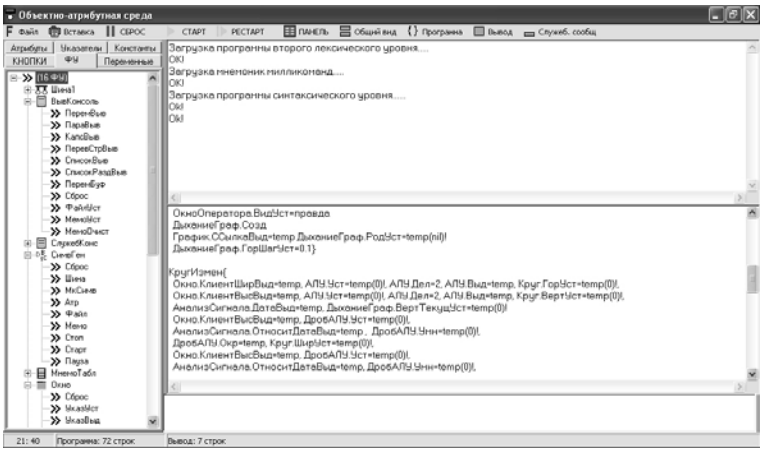

Рис. 5. Программная оболочка ОА-среды

Программная оболочка языка ОА-платформы (рис. 5) существенно облегчает программирование системы: интерфейс состоит из четырех текстовых окон (программы, вывода сообщений, вывода сообщений об ошибке при компиляции и командной строки, служащее для управления ФУ в РВ) и панели инструментов, где отображаются участвующие в вычислительном процессе ФУ вместе с набором их милликоманд, переменные и капсулы.

Примером реального применения ОА-платформы является проект системы биологической обратной связи (БОС), реализуемый в Центре психолого-педагогической коррекции и реабилитации "Строгино". Данная система включает набор датчиков физиологических сигналов, снимаемых с пациента в РВ с помощью АЦП L-Card e-154, ПО анализа сигналов и средства визуальных и тактильных воздействий (нагревательные элементы) на пациента, с помощью которых осуществляется психокоррекция пациента. Для этого проекта были созданы и отлажены виртуальные ФУ, реализующие экранный интерфейс программы и управление реальным периферийным оборудованием (АЦП L-Card e-154), обработку (автокалибровка и фильтрация сигнала, сбор статистической информации о сигнале) и визуализацию сигнала, снимаемого с датчика. Аппаратная реализация, еще не воплощенная в макетных образцах, в настоящее время проходит имитационное моделирование в среде MLDesigner, что позволит выбрать наиболее оптимальные инженерные решения аппаратных ОА-устройств, определить характеристики будущих устройств и оценить выигрыш в производительности по сравнению с классической архитектурой.

Ближайшие перспективы проекта ОА-архитектуры связаны с реализацией распределенной версии ОА-платформы, а также реализации ОА-платформы на базе ПЛК, экспериментальная проверка их возможностей в составе типовых систем автоматизации. Доработка среды программирования ведется в направлении создания индексной системы адресации (она позволяет расширить адресное пространство ОА-среды на несколько вычислительных узлов), а также реализации механизма передачи капсул между вычислительными узлами и системы автоматической загрузки ОА-программы по вычислительным узлам системы автоматизации; и расширения возможностей ОА-языка. На базе совместной с компанией LEGOS лаборатории промышленной автоматики и научно-образовательного центра "Многопроцессорные вычислительные системы" (НОЦ МВС) на кафедре "Вычислительные системы и сети" МИЭМ ведутся работы по созданию экспериментального стенда для отладки программных и аппаратных решений ОА-архитектуры вычислительной системы и построения тестовых прикладных систем автоматизации, таких как, например, охранно-пожарные системы и системы пожаротушения, контроля доступа в помещение, климат-контроля помещений, контроля качества продукции на конвейере.

Список литературы

1. *Салибекян С.М.* Принципы милликомандной архитектуры как основа построения высокопроизводительных адаптивных вычислительных систем // Автоматизация и современные технологии. 2002. № 5.
2. *Салибекян С.М., Тарловский Г.Р.* Самопрограммирующийся адаптивный процессор, управляемый милликомандами (САПУМ). Информационные, сетевые и телекоммуникационные технологии. Сборник научных трудов. МИЭМ 2001.
3. *Салибекян С.М.* Вычислительная структура, управляемая милликомандами. Информационные, сетевые и телекоммуникационные технологи. Сборник научных трудов. МИЭМ. 2001.
4. *Парр Э.* Программируемые контроллеры: руководство для инженера / Э. Парр; Пер. 3-го англ. изд.- М.,: БИНОМ. Лаборатория знаний, 2007.
5. *Денисенко В. В.* Компьютерное управление технологическим процессом, экспериментом, оборудованием. М.: Горячая линия-Телеком, 2009.
6. *Федоров Ю.Н.* Справочник инженера по АСУТП: Проектирование и разработка. Учебно-практическое пособие. М. Инфра-Инженерия, 2008.

Салибекян Сергей Михайлович *– ст. преподаватель,*

Панфилов Петр Борисович *– канд. техн. наук, доцент, проф. кафедры "Вычислительные системы и сети"*

ГОУ ВПО "Московский институт электроники и математики (технический университет)"

Контактный телефон (495)916-89-09. E-mail: salibek@yandex.ru panfilov@miem.edu.ru