ЯЗЫК ABIS: ЯЗЫКОВАЯ СРЕДА ДЛЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ЭЛЕМЕНТАМИ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

А.Г. Полетыкин, М.Е., Бывайков (ИПУ РАН им. В.А. Трапезникова)

Аннотация: Работа посвящена языку ABIS как языковому средству для разработки программного обеспечения с элементами искусственного интеллекта. Язык ABIS является полностью российской разработкой, созданной в Институте проблем управления РАН им. В.А. Трапезникова. История использования языка насчитывает 35 лет. В настоящее время язык используется только внутри Института. Главной областью применения является интегрирующая платформа – система Оператор, которая применяется для интеграции АСУ ТП АЭС. В статье содержатся основные ссылки на доступные публикации и ссылки на место для скачивания языка ABIS с исходными текстами.

Ключевые слова: Искусственный интеллект, АСУ ТП, АЭС.

Введение

В 60-70 годах 20го века одной из многообещающих парадигм Искусственного интеллекта (ИИ) в СССР и за рубежом были экспертные системы, при помощи которых пытались создавать программы на основе неформальных знаний экспертов. Для этого создавались языковые среды, имитирующие рассуждения людей. В частности, логический вывод.

Одной из таких попыток, приведших к созданию полезного продукта, является язык ABIS. Дата его рождения – 1989 год. Место рождения – Институт проблем управления РАН им. В.А.Трапезникова, лаборатория 31. Автор идеи и первый программист - Зуенков Михаил Анатольевич.

За прошедшие годы язык претерпел ряд изменений и к 2000г окончательно сформировался как основной языковый инструмент высокого уровня для интегрирующей платформы – системы Оператор [1,2] (далее Оператор).

Ныне язык ABIS является продуктом для внутреннего применения внутри коллектива программистов ИПУ РАН, поддерживается ими и портируется на современные операционные системы, используется для создания проблемно ориентированных программ с ИИ.

Однако авторы считают, что потенциал «старого» языка ABIS не раскрыт и намереваются ознакомить с ним специалистов - программистов, которые занимаются автоматизацией на основе ИИ. Продвижение и популяризация языка ABIS является целью данной публикации.

1. Основы языка

В языке реализована идея имитации «человеческого» способа рассуждения на основе «эвристических» правил «Если …, то …».

Применяется «машина логических выводов» прямого типа.

В качестве базовой модели данных была выбрана расширенная реляционная модель баз. данных Данные хранятся в виде таблиц с типизированными полями и возможностью прямых вложенных между кортежами. СУБД является составной неотъемлемой частью языка.

Для имитации коллективных рассуждений используются модель взаимодействия людей посредством писем. Для этого в языке созданы средства сетевого взаимодействия между отдельными программами в виде «посылок» высокого уровня переменной величины. Они позволяют обмениваться данными в виде подмножеств таблиц произвольного переменного состава и объема.

2. Реализация языка

Язык реализован в виде интерпретатора-компилятора, который представляет собой программу на языке “C ANSI”, адаптированную в среду UNIX/Linux.

Полное описание языка и работа с интерпретатором-компилятором доступны по ссылке [3].

Интерпретатор-компилятор с исходными текстами в виде виртуального образа доступен для скачивания по ссылке [4].По ссылке [5] содержится инструкция по инсталляции с правилами работы и примерами (электронный депозитарий, зык моделирования киберугроз и рисков в системах управления КАЛЬКИБЕР, редактор edit) можно ознакомиться по ссылке.

3. Особенности, место и преимущества языка

ABIS поддерживает основные типы данных, исключая битовые операции, позволяет не только «имитировать рассуждения», но и выполнять вычисления любой сложности с любыми входными данными в виде текстовых и бинарных файлов и потоков данных по сети.

Реализация встроенных средств ввода-вывода в виде структурированных файлов и сетевых посылок позволяет разгрузить разработчика от рутины разбора входных и формирования выходных данных. Эти средства позволяют очень быстро и удобно создавать базы данных, метаязыки и протоколы прикладного уровня. Они реализованы, в том числе, на бинарном уровне и достаточно эффективны. Хотя остается и обмен в текстовой форме, которая удобна для анализа человеком, импорта и экспорта.

Эта особенность предопределяет место, где использование языка дает наибольший эффект. Это сложные логические вычисления с большим числом однотипных алгоритмов работы с разнотипными данными, особенностями, исключениями. Например, для реализации логики человеко-машинного интерфейса, который претерпевает огромное число доделок, переделок и т.п. Технические операции по обрисовке графики, вводу с клавиатуры, контроль за трекболом нужно реализовывать на более эффективных языках среднего уровня. Например, на “C”.

В Оператор на языке ABIS реализованы логические алгоритмы обработки в клиент-серверной архитектуре программ обработки данных от систем низовой автоматики АСУ ТП и программ дистанционного управления АЭС. При этом для связи с периферийными устройствами используются высокоэффективны программы на языке “C”, интегрированные с программами на языке ABIS через сетевые каналы. Информацию о внедрении на АЭС можно получить по ссылке [6].

Примером эффективного использования является программа электронного депозитария (см. [5]). В ней логическая часть написана на ABIS, а интерфейс реализован при помощи Web-технологий. Программа с широкими возможностями ввода информации/поиска включает всего 705 строк. Программа была разработана за один месяц без отрыва от основной деятельности одним человеком. Ее простота позволяет легко вносить изменения и добавления, не затрачивая больших усилий и средств. Авторы считают ABIS хорошей альтернативой Microsoft Access для быстрой разработки небольших специализированных баз данных.

Встроенная СУБД ABIS с логикой поиска/порождения/удаления на основе правил позволяет реализовывать сложные алгоритмы обработки информации в компактной форме. Это характерно для САПР. В частности, САПР в Оператор в виде многопользовательской системы с векторной графикой также разработана на ABIS.

Особенности встроенного сетевого протокола высокого уровня позволяют распараллеливать создание сложных программ между различными разработчиками: сложный алгоритм обработки информации делится на последовательные этапы, которые распределяются между вычислительными процессами, устанавливается общая модель данных (форматы таблиц) для обмена информацией и для разработки каждого отдельного вычислительного процесса выделяется человек-разработчик. Далее разработчики автономно создают свои куски программы, пользуясь для отладки тестовыми файлами, которые по структуре и использованию идентичны посылкам по сети (это тоже особенность ABIS). На этапе сборки на основе кусков создается вычислительный конвейер в виде отдельных процессов, связанных сетевыми каналами. Так создавалась Оператор.

4. Пример базы данных документов программ на ABIS

Специализированный электронный депозитарий для электронных копий документов по программному обеспечению, разрабатываемому в ИПУ РАН для АСУ ТП АЭС. Специфика документов состоит в том, что они выполняются по российским стандартам с учетом требований иностранных заказчиков. Каждый документ включает от одного до шести файлов разных форматов. Структура комплекса показана на рисунке 1.

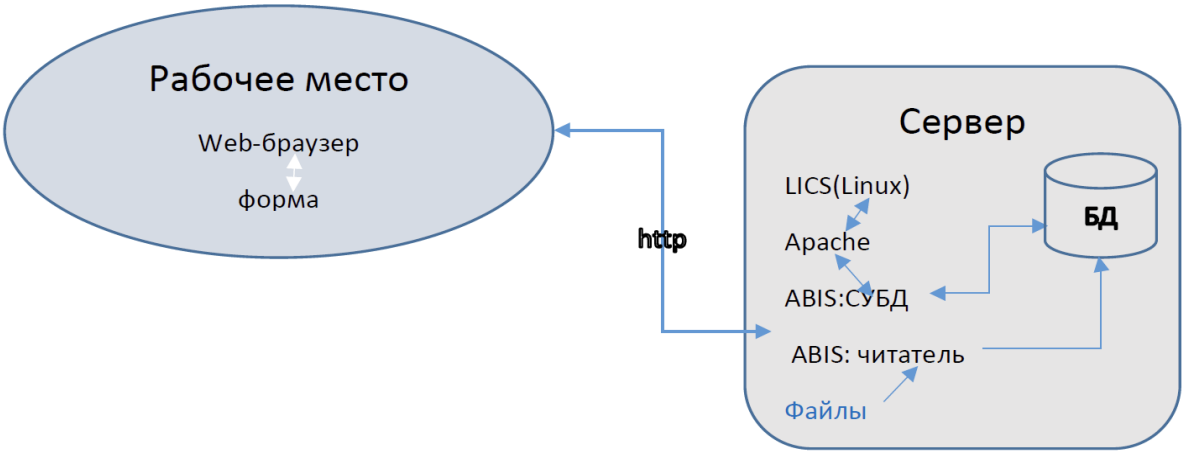

Рисунок 1. Структура ПО электронного депозитария

Комплекс включает две компоненты на языке ABIS. Первая компонента робот-читатель предназначена для анализа файлов. Робот-читатель определяет списки документов, находит файлы, относящиеся к разным документам, определяет их назначение, «читает» файлы и выделяет из них отдельные части: название документов, их коды по ЕСПД и KKS, ФИО разработчиков, аннотацию и другую информацию (всего 20 видов). Далее эта информация заносится роботом-читателем базу данных (БД на рисунке 1) поисковых характеристик. Робот-читатель разработан на языке ABIS, которая вызывается из консоли скриптом на B shell и использует свободно-распространяемые компоненты для преобразования файлов doc(x) и pdf в текстовый вид. Исходный код робота-читателя состоит из небольших частей технической подготовки текстов для автоматического анализа и загрузки в базу данных. Основная же часть исходного кода, которая анализирует содержимого файлов представляет собой формализованные знания сотрудников в виде эвристических правил. Оставляя открытым вопрос, можно ли считать робота-читателя ИИ, укажем, что он позволил решить вопрос о создании базы данных

(4 тыс. документов) в течении 3х месяцев. Однако робот-читатель, признаем, не лишен недостатков и ему далеко до человека. Он действует исходя из заложенных правил. А если документ содержит отклонения, ошибки, не точности, то в базе данных документов остаются лакуны. В этом случае можно исправить ошибки при помощи простого ЧМИ (см. рисунок 2).

Поисковая форма представляет собой мобильный код на языке html, которая загружается в среду Web-браузера на рабочее место из севера депозитария через Apache из состава ОС Lics. Сам сервер представляет собой виртуальную машину, функционирующую в ведомственной облачной среде. Вторая компонента на языке ABIS выполняет функции СУБД и отвечает за ведение диалога с пользователем. Основная ее часть включает технические операции, характерные для любой работы с СУБД. Но есть несколько «интеллектуальных» правил, препятствующих потере информации даже в случае, когда пользователь совершает ошибки. В частности, используя форму невозможно удалить файлы документов из БД. Даже если пользователь удалит или исказит поисковую информацию робот-читатель сможет восстановить ее. На рисунке 2 представлена форма для поиска документов.

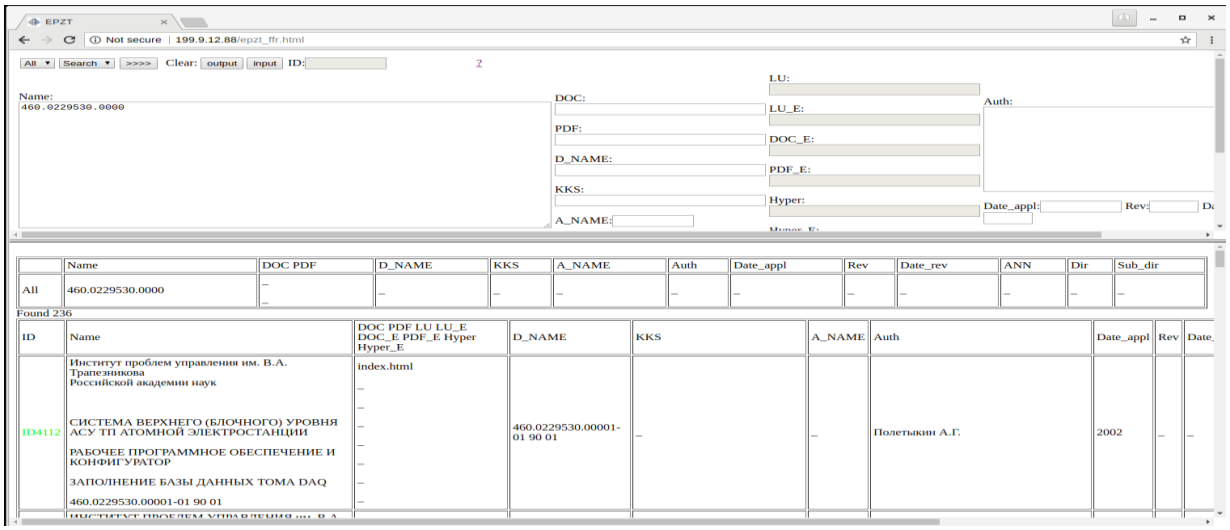

Рисунок 2. Поисковая форма

Данный пример служит иллюстрацией того, как используя ABIS, простейшие свободно-распространяемые элементы Linux и Web-технологий можно быстро сознать базу данных с уникальными свойствами. Которую в силу простоты можно без труда модифицировать при изменении требований.

Отметим, что несколько тысяч записей в базе данных – это очень мало для ABIS-программ. В приложениях для АСУ ТП базы данных содержат сотни тысяч записей, а поток транзакций может составлять 10 тыс. в секунду. Это связано с тем, что в программе на языке ABIS данные могут размещаться в оперативной памяти в виде сложно структурированных небольших сегментов с ускорением доступа при помощи ключей и хеширования.

5. Пример аналитической программы на языке ABIS: КАЛЬКИБЕР

На ABIS можно создавать замкнутые среды сложной обработки данных без привлечения сторонних программ. Примером служит программа КАЛЬКИБЕР, предназначенная для оценки киберугроз и рисков, возникающих из-за уязвимостей программного обеспечения.

Научные основы программы описаны в [7].

Программа (см. рисунок. 3) содержит единственный файл на языке ABIS, который считывает входные данные на языке ABIS, производит вычисления и выдает результат тоже на языке ABIS. Подробности использования см. в [5].

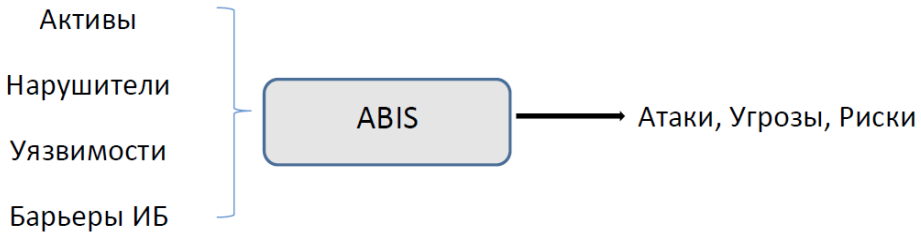

Рисунок 3. Схема работы КАЛЬБКИБЕР

Ниже описывается метаязык высокого уровня для моделирования киберугроз, рисков и др.

Входные данные содержат описание активов в виде кортежей сл. вида:

TG (<наименование требования>, I| A|IA, <наименование элемента ПО или агрегата ПО>, <код ТС| наименование подмножества ТС>).

Например: TG (ARH12, IA, U_ServSW, 30CRV20); содержит требование ARH12, чтобы все ПО агрегата U_ServSW было целостным и доступным на 30CRV20.

Входные данные о нарушителях описываются в виде кортежей вида:

SeT (<наименование группы нарушителей>, <наименование нарушителя>)

в файле intrud.bd.

*Пример: /*Все нарушители кроме, кто не знает паролей защиты*/*

*SeT (I_0, H0); /*Человек, имеющий физ. доступ к РС и права оператора*/*

*SeT (I_0, PC);/*Компьютер, подключенный непосредственно к ЛВС.*/*

*SeT (I_0, RPC);/*Компьютер, подключенный извне к ЛВС через смежные системы. Но паролей защиты и прав оператора не имеет.*/*

Входные данные о уязвимостях описываются в виде кортежей вида:

VLB (<наименование уязвимости>, I| A|IA, <наименование компоненты ПО>| <наименование агрегата ПО>,<наименование подмножества ТС>|<код ТС>,<наименование группы нарушителей>|<наименование нарушителя>);

Пример:

VLB (V_WEBserver, A,U_ServSW,H_serv, PC);

означает наличие уязвимости V_WEBserver, которая позволяет блокировать доступность ПО серверов через компьютер, подключенный непосредственно к ЛВС.

Входные данные о барьерах информационной безопасности описываются в виде кортежей вида:

BR (<наименование барьера>, <наименование компонента ПО>| <наименование агрегата ПО>,<наименование подмножества ТС>|<код ТС>,< наименование уязвимости>|<наименование группы уязвимостей>, <наименование группы нарушителей>|<наименование нарушителя>, Higth|Midle);

Пример:

BR (VIRT_LAN,IA, U_ServSW. H_serv, V_WEBserver, I_0, Higth);

барьер VIRT_LAN без участия человека защищает "Ц" и "Д" ПО серверов от атаки группы "Все нарушители кроме, кто не знает паролей защиты" через уязвимость V_WEBserver.

Моделирование атак состоит в поиске уязвимостей, которыми могут воспользоваться нарушители для того, чтобы добиться TG (…)=0 и LOGF (…) = 0.

Сначала производится поиск гипотетических атак на компоненты ПО и подсчитывается число барьеров, которые им препятствуют. Результат выводится в файл attacks в виде конструкций:

Attack (<наименование нарушителя>, I| A|IA, <наименование компоненты ПО>,<код ТС>,(<наименование уязвимости>,<число барьеров "Higth" > , <общее число барьеров >;1.00

Пример: Attack (H0,A,IZ,30PCA10,V_json,1,1);1.00

Атака нарушителя H0 на целостность компоненты IZ PC 30PCA10 через уязвимость V_json, которой препятствуют один барьер "Higth".

В файл collisions вводятся подробности о действиях барьеров в виде конструкций:

Collision (Attack (…,_,_), Obstacle (<наименование барьера>, I| A|IA, Higth|Midle));1.00

Пример: Collision (Attack (H0,A,IZ,30PCA10,V_json,_,_), Obstacle (JSON_sign,A,Higth));1.00

Далее производится оценка достаточности барьеров для того, чтобы блокировать атаку. Для этого служат константы, установленные в файле настроек моделирования set_up.bd:

Const (H_MIN, <Минимальное число барьеров Height для нейтрализации атаки>);

Const (L_MIN, <*Минимальное число барьеров для нейтрализации атаки*>);

Пример:

Const (H_MIN, 1);

Const (L_MIN, 2);

Для приведенного примера атаки имеется один барьер "Height", что является достаточным.

В файле set_up.bd могут содержаться ограничения на рассматриваемых нарушителей, ПО, ТС, уязвимостей и барьеров в виде конструкций вида:

Intruder (<наименование нарушителя>);

SOFT (<наименование компоненты ПО>|);

Hardware (<код ТС>);

Vuln (<наименование уязвимости>);

Отсутствие Intruder означает рассмотрение всех нарушителей при моделировании. Если есть одна или более, то моделирование производится только для указанных нарушителей.

Отсутствие SOFT означает моделирование атак на все компоненты ПО. Если есть одна или более, то моделирование атак производится только для указанных компонент ПО.

Отсутствие Hardware означает моделирование атак на все ТС. Если есть одна или более, то моделирование атак производится только для указанных ТС.

Отсутствие Vuln означает использование всех уязвимостей при моделировании. Если есть одна или более, то моделирование производится только для указанных уязвимостей.

В результате моделирования в выходной файл risk всегда выводится конструкция:

risk (<значение H_MIN>,<значение *L_MIN*> ,<величина риска>);1.00

и копия конструкций с ограничениями из set_up.bd

Значение величины риска "0" означает отсутствие угроз.

Если барьеров не достаточно, то возникают угрозы нарушения ПФ, информация по которым выводится в файл risk в виде конструкций:

threat (<наименование нарушителя>,<наименование ПФ>,<величина ущерба от нарушения ПФ>);1.00

trace (Attack (…),<наименование ПФ>, TG (…));1.00

В случаях наличия угроз величина риска вычисляется как максимум из величин ущербов нарушений ПФ.

Пример:

……

trace (Attack (H0,IA,ABIS,30CRV20,V_libxkbcommon,0,1),PPA,
TG (PPA22,IA,U_PPA_serv,30CRV20));1.00

TG (PPA22,IA,U_PPA_serv,30CRV20));1.00

.....

threat (RPC,MON_RCONT,100);1.00

....

threat (PC,PPA,1);1.00

.....

risk (1,2,100);1.00

По архитектуре КАЛЬКИБЕР примитивен. А по функционалу и сложности алгоритма обработки информации совсем не прост: он имитирует рассуждения специалиста по информационной безопасности. Такую программу способен создать студент, аспирант, научный работник с элементарными навыками программирования. Это позволяет рекомендовать язык ABIS для создания прототипов, отработки алгоритмов, создания уникальных программ, требующих частой модификации, адаптации под специфические требования.

6. Пример применения языка ABIS в связке с другой программой через сетевые каналы

Этот пример иллюстрирует применение языка ABIS в диалоговой программе edit. Программа на языке ABIS взаимодействует с библиотекой Motif X11 через промежуточную программу, входящую в Оператор.

Назначение программы – создание, редактирование, адаптация эскизов мнемосхем из готовых блоков AutoCAD. Структура программы представлена на рисунке.

Рисунок 3. Структура программы edit

Ключевой особенностью программы является адаптация. Дело в том, что эскизы мнемосхем для Оператор удобнее всего создавать на AutoCAD, его клонах ZVCAD, NanoCAD и других мощных инженерных пакетах программ инженерной графики, поддерживающих экспорт в формат dxf. Соревноваться с ними по сервису для разработчика практически невозможно. Но результат работы, файл в фомате dxf, в силу того, что указанные пакеты ориентированы на рисование, может содержать невидимые элементы, которые могут негативно влиять на использование эскизов в АСУ ТП АЭС и даже представлять опасность. Поэтому, сердцевиной программы на языке ABIS в составе edit является экспертная система, которая читает, разбирает и очищает файлы эскизов от посторонних включений. Можно условно высказать мнение, что она «понимает» замыслы людей, «читая» инженерную графику (электронные чертежи). Если ей что-то не нравится, она сообщает об этом человеку в понятной для него форме. (Способность объяснять свои выводы людям считается в классическом понимании ИИ обязательной чертой.). Конечно «интеллект» программы узко специализирован: она понимает только эскизы мнемосхем. Но результатом адаптации является то, что эскизы, подвергшиеся «интеллектуальной» обработке, становятся доверенными.

При старте программы запускаются два вычислительных процесса. Затем они устанавливают два канала по протоколу TCP/IP и начинают работать как единое целое. На рис 4. Представлен скриншот программы.

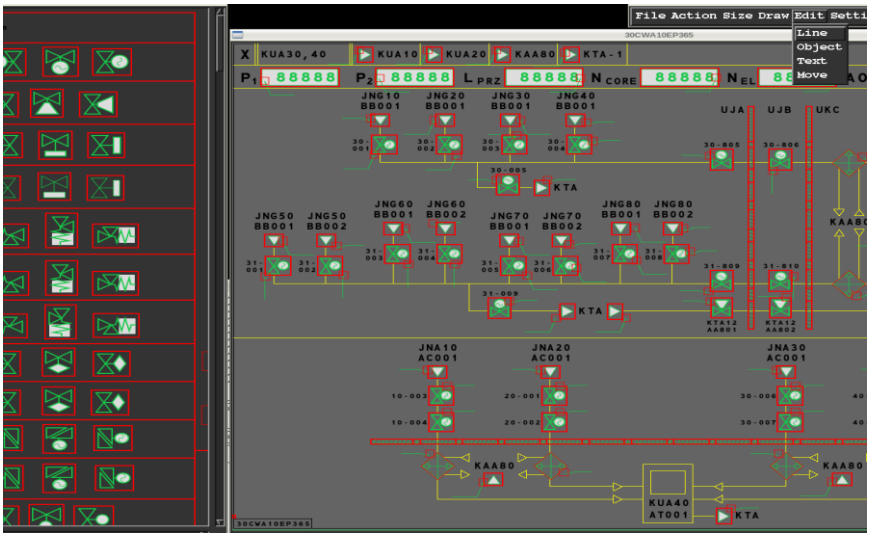

Рисунок 4. Скриншот программы edit

По организации вычислений часть программы edit на языке ABIS отличается от тех, что были описаны выше. Программы, представленные в разделах 3,4, выполняются в простом пакетном режиме: загрузился, считал данные, произвел вычисления, выдал результат и выгрузился.

В edit программа на языке ABIS работает в циклическом режиме: загрузился, получил команду оператора, выполнил нужные вычисления и встал в ожидание новых команд оператора до тех пор, пока он не прикажет прекратить работу.

По внутренней организации программа на языке ABIS в edit использует стиль объектно-ориентированного программирования, который считается наиболее эффективным для диалоговых программ. Имеются классы, к которым привязаны методы. Имеется иерархия классов. Ход вычислительного процесса организован через посылки сообщений между экземплярами классов.

Объектно-ориентированный стиль для программы на языке ABIS позволил создать компактный исходный код, но этот код не прост для понимания человеком. И не очень эффективен.

Для главного применения языка ABIS в Оператор используется простой стиль организации управления вычислительным процессом в виде вложенных циклов. Хотя исходный код намного больше, но для понимания человеком он гораздо удобнее. Для скорости выполнения – тоже.

Заключение

Язык ABIS можно применять для создания «гибридных» программ, выполняющих и рутинные алгоритмы обработки разнотипных данных и алгоритмы, имитирующие рассуждения людей разной специальности и разной сложности. Преимущество состоит в том, что многое можно реализовать в единой языковой среде. Не все, но основную логику и вспомогательное окружение. Там, где требуется, через сетевые каналы языковая среда комплексируется с внешними программами.

Языковая среда чрезвычайно компактна, проста и эффективна по скорости выполнения программ. Авторы считают ее хорошим выбором для «точечной» автоматизации и интеллектуализации, где нужно заменять «средне интеллектуальный» труд людей «в меру умными» роботами.

Литература

1. Интеграционная платформа для АСУ ТП - Система Оператор, 2017. https://www.ipu.ru/science/applied-research/software/integration-platform.
2. *Полетыкин А.Г., Менгазетдинов Н.Э., Жарко Е.Ф., Промыслов В.Г., Бывайков М.Е., Степанов В.Н., Байбулатов А.А., Семенков К.В., Акафьев К.В.* Интеграционная платформа для АСУ ТП – Система Оператор / Труды 16-й Международной конференции «Управление развитием крупномасштабных систем» (MLSD’2023, Москва). М.: ИПУ РАН, 2023. С. 144-148. https://www.ipu.ru/sites/default/files/publications/75919/73481-75919.pdf.
3. *Бывайков М.Е.* Язык ABIS. Описание языка [Электронный ресурс]: монография. Электрон. текстовые и граф. дан. (0,6 Мб).– М.: ИПУ РАН, 2013.– 1 электрон. опт. диск (CD-R).– Систем. требования: IBM PC, Internet Explorer, Acrobat reader 3.0 и выше. М.: ИПУ РАН, 2013. – 87 с https://www.ipu.ru/sites/default/files/publications/20375/6010-20375.pdf
4. 4.https://www.ipu.ru/science/applied-research/software/abis/abis.ova.

5. Компилятор-интерпретатор языка ABIS.Инструкция по установке и применению. М.: ИПУ РАН, 2024 https://www.ipu.ru/science/applied-research/software/abis/abis_2024.pdf.
6. *Менгазетдинов Н.Э., Полетыкин А.Г., Промыслов В.Г., Зуенкова И.Н., Бывайков М.Е., Прокофьев В.Н., Коган И.Р., Коршунов А.С., Фельдман М.Е., Кольцов В.А.* Комплекс работ по созданию первой управляющей системы верхнего блочного уровня АСУ ТП для АЭС «БУШЕР» на основе отечественных информационных технологий М.: ИПУ РАН, 2013. – 95 с. https://www.ipu.ru/sites/default/files/publications/20368/6007-20368.pdf.
7. *Промыслов В.Г.*, *Полетыкин А.Г.* Digitally Controlled Assets Subjected to Cyberattacks: Definitions and Cyberproof Criteria Based on the Analysis of Explicit and Hidden Functions / Proceedings of the 7th IFAC Conference on Manufacturing Modelling, Management, and Control (MIM`2013, Saint Petersburg). Saint Petersburg: International Federation of Automatic Control, 2013. С. http://www.ifac-papersonline.net/Detailed/60161.html.

гл.н.с., д.т.н. Полетыкин Алексей Григорьевич (Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В.А. Трапезникова)
Телефон8 (903) 130-60-92
poletik@ipu.ru, poletik@inbox.ru/

с.н.с., к.т.н. Бывайков Михаил Евгеньевич (Федеральное государственное бюджетное учреждение науки Институт проблем управления им. В.А. Трапезникова)
Телефон: 8 (916) 578-72-81
lab31.5@mail.ru