УДК 004.415.26

Ю.И. Рогозов, А.А. Дубровский, А.С. Свиридов

НОВЫЙ ПОДХОД РЕАЛИЗАЦИИ ИДЕИ «ПРОГРАММИРОВАНИЕ БЕЗ ПРОГРАММИСТА»

Введение

Развитие программного обеспечения информационных систем идет в направлении постоянного увеличения сложности создаваемого ПО. Требуются огромные трудозатраты для создания современных систем как в гражданских областях, так и в области специализированных систем, применяемых в военной и космической промышленности. Именно военная и космическая техника всегда была на переднем рубеже технического прогресса в нашей стране, и именно в этой области впервые встала проблема снижения трудозатрат и увеличения прозрачности процесса разработки систем управления.

Запуск первого спутника в 1957 г. потряс весь мир и застал американцев врасплох. Полет Юрия Гагарина в 1961 г. стал триумфом советской науки. Воплотились в жизнь и стали явью фантастические проекты национального героя СССР – легендарного конструктора Сергея Королева. Интеллектуальную поддержку его дерзких проектов обеспечивала «нервная система» ракеты – бортовая система управления, созданная Николаем Пилюгиным [1].

Чем больше усложнялись ракетно-космические комплексы и системы, тем сильнее была нужда в безупречном *взаимопонимании* между разработчиками. Отсутствие взаимопонимания болезненно сказывалось на ходе крупных ракетных проектов. Особенно остро проблема взаимопонимания проявила себя при создании космического комплекса «Энергия–Буран» [1].

В качестве ответа на эти вызовы совместными усилиями Российского космического агентства (Научно-производственный центр автоматики и приборостроения имени академика Н.А. Пилюгина, г. Москва) и Российской академии наук (Институт прикладной математики имени академика М.В. Келдыша, г. Москва) была начата разработка инструмента, обеспечивающего безукоризненное взаимопонимание между людьми, позволяющего описать структуру и логику программы или бизнес-процесса в понятной неспециалисту по программированию форме, и дальнейшим преобразованием такого описания в программный код.

Разработка получила название «Дракон» и реализовывала основную парадигму **«программирование без программиста»** или **«кто обладает знаниями, тот и должен их формализовать»**. «Дракон» развивался более пятнадцати лет и успешно применялся при создании целого ряда управляющих систем использовавшихся в ракетно-комических комплексах [1]: «Энергия–Буран», «Морской старт» (Sea Launch), «Протон-М» и др.

«Дракон» использует оригинальную графическую нотацию, основанную на модифицированных блок-схемах алгоритмов и ряде дополнительных правил их построения. Может использоваться как для описания бизнес-процессов, так и для описания алгоритмов работы программы, в таком случае он использует синтаксис определенного языка программирования, образуя подмножество исходного языка («Дракон-С», «Дракон-Паскаль», «Дракон-Оберон» и т.д.).

При использовании такого программно-ориентированного «Дракона» возможна трансляция «Дракон-алгоритма» в программный код на соответствующем языке программирования.

Анализ структуры «Дракон» и области его применения показывает ряд недостатков, особенно остро проявляющихся при использовании для разработки систем, ориентированных не на управление, а на накопление и обработку информации:

- привязка к языку программирования производится на стадии составления «Дракон-алгоритма» работы программного модуля;
- требуется знание языка программирования аналитиком, при создании «Дракон-алгоритма», что нарушает один их принципов «Дракон»;
- высокая трудоемкость создания систем, тесно взаимодействующих с БД, так как операции по работе с БД описываются в «Дракон-алгоритме» средствами языка программирования, снижая понятность схемы.

Принципы построения алгоритмического языка описания модулей

Выявленные недостатки языка ДРАКОН в области построения систем, ориентированных на создание информационных систем привели к необходимости создания нового инструмента. Данный инструмент должен реализовать основную идею, заложенную создателями «Дракон» – сокращение звеньев в цепочке аналитик – программа, и, за счет этого, снижение затрат при создании системы.

Предлагается новый алгоритмический язык описания структуры и логики программных модулей (АОМ), на входе которого будут знания о предметной области, на выходе – работоспособная информационная система. Основным действующим лицом в такой схеме будет системный аналитик, а степень участия программиста должна стремиться к нулю. Особенностью применения является ориентация на создание систем сбора, хранения и обработки информации в широком круге предметных областей и бизнес-процессов (БП).

Основными принципами, которые закладываются в АОМ, являются:

- качество элементов описания структуры и логики работы программного модуля (используется набор типовых шаблонов, причем ориентированных на тесное взаимодействие с БД);
- качество выразительного средства (используются модифицированные блок-схемы алгоритмов МБА, с добавлением специализированных элементов);
- отказ от трансляции алгоритма в текст программы на языке программирования и компиляции текста в исполняемый код (непосредственная передача описания модуля в исполняющую среду);
- возможность адаптации ИС к изменениям предметной области;
- высокая универсальность для решения задач накопления и обработки данных в ИС различных предметных областей, за счет использования оригинальной формы представления данных со статической структурой [2].

Проанализируем соответствие и различия, предлагаемого подхода и структуры использования языка «Дракон» в разрезе стадий разработки, представленных в табл. 1.

Таблица 1

Сравнение «Дракон» и АОМ

<table>
<tr><th></th><th></th><th>ДРАКОН</th><th>АОМ</th></tr>
<tr><td rowspan="3">Описание БП</td><td>Средство</td><td>«Дракон»</td><td>Любые методологии</td></tr>
<tr><td>Предмет описания</td><td>БП предметной области</td><td>БП предметной области</td></tr>
<tr><td>Элемент</td><td>Функции БП</td><td>Функции БП</td></tr>
<tr><td rowspan="3">Программный модуль</td><td>Средство</td><td>«Дракон-С», «Дракон-ПАСКАЛЬ» и др.</td><td>АОМ</td></tr>
<tr><td>Предмет описания</td><td>Структура и логика работы программного модуля</td><td>Структура и логика работы программного модуля</td></tr>
<tr><td>Элемент</td><td>Код на языке программирования</td><td>Шаблоны программ</td></tr>
<tr><td rowspan="3">Программный код</td><td>Средство</td><td>«С», «Паскаль» и др.</td><td rowspan="4">Исполняющая среда</td></tr>
<tr><td>Предмет описания</td><td>Код программы</td></tr>
<tr><td>Элемент</td><td>Операторы языка программирования</td></tr>
<tr><td>Исполняе-мый код</td><td></td><td>Исполняемый код программы</td></tr>
</table>

При описании бизнес-процессов предметной области могут быть использованы любые известные методологии, относящиеся как к методам структурного, так и объектно-ориентированного (SADT, UML, ARIS) анализа и проектирования [3, 4]. Элементами, из которых строится модель на данном уровне, являются отдельные шаги бизнес-процесса и объекты предметной области.

Переход на этап описания структуры программных модулей производится после выделения атомарных функций бизнес-процесса и синтеза модульной структуры ИС [5]. Каждый модуль реализует одну или несколько функций бизнес-процесса, объединенных в логическую последовательность. Данная последовательность записывается в виде алгоритма на АОМ. Элементы АОМ, в отличие от «Дракон», содержат не код на языке программирования, а являются прототипами шаблонов, с указанием необходимых параметров. Шаблон представляет собой набор заранее определенных типовых операций, которые применяются при создании ИС. Шаблоны описывают операции, связанные с манипуляцией данными в БД, например, получение значений атрибутов указанной сущности, удаление и модификация данных в БД. Кроме того, шаблоны описывают структуру взаимодействия программного модуля с пользователем через графический интерфейс ИС.

При переходе на уровень программной реализации предлагается переводить алгоритм, записанный в виде совокупности элементов АОМ, не в текст на языке программирования, а передавать на выполнение в исполняющую среду. При вызове соответствующего программного модуля в момент исполнения происходит интерпретация элементов алгоритма в программные элементы (интерфейсные окна и компоненты, функции обработки данных и взаимодействия с БД). Для решения

задачи унификации программных элементов ИС, т.е. выделения шаблонов АОМ необходимо преодолеть сложность связанную с особенностями предметной области. Каждая область имеет свой набор данных со своей уникальной структурой. Проблема унификации традиционно решатся на основании двух подходов:

1) использование низкоуровневых элементов унификации, фактически близких к языкам программирования (в том числе и в подмножествах «Дракон» - «Дракон-С», и т.п.) [1];
2) специализации шаблонов по предметным областям (подход в ERP-системах) [6].

Нами предлагается подход к унификации программных элементов (шаблонов) на основании унификации структуры данных, хранящихся в БД – статической структуры данных [2]. Использование такой структуры позволяет на физическом уровне единообразно работать с данными, имеющими различную структуру на логическом уровне. Такая унификация структуры данных позволяет выделить небольшой и полный набор шаблонов, достаточный для построения ИС в любой предметной области без узкой специализации и без низкоуровневого описания.

Структура ИС при таком построении включает несколько составляющих (рис. 1):

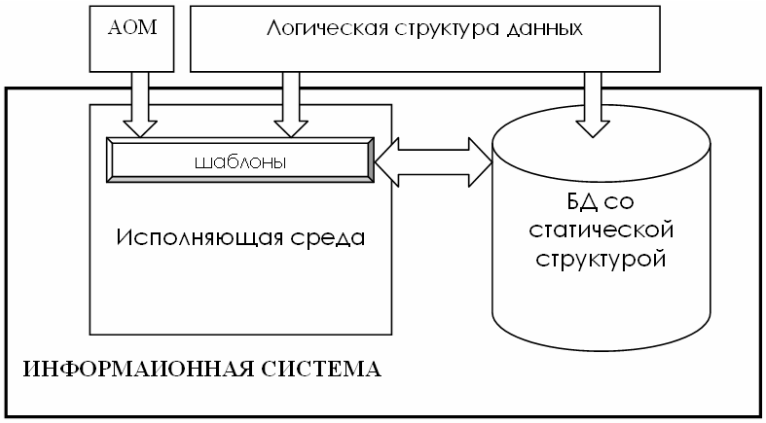


Рис. 1. Структура ИС при использовании АОМ

Реализация алгоритмического языка описания модулей

Алгоритмический язык описания модулей основан на следующих составляющих:

- декларативная часть, описывающая внешнее взаимодействие модуля и структуру данных, необходимых для работы модуля;
- алгоритм работы модуля, записанный в виде специализированных МБА с использование шаблонов.

Рассмотрим подробнее декларативное описание модуля, которое состоит:

- из описания интерфейсов модуля описываются способы взаимодействия с этим модулем, исходя из того что, модуль это подмножество $m_i=\{f_1,f_2...f_j\}$ множества $M = \{m_1,m_2..m_i\}$ всех функций ИС, где f_i атомарная функция системы, можно сказать что интерфейсами являются пересечения подмножеств m_i множества M. Принимаемые и передаваемые данные, например, имеется модуль, который может добавлять и изменять данные по человеку, такой модуль будет иметь 2 интерфейса – «Добавить» без каких-либо параметров и «Изменить» с параметром идентификатор человека;

- описания логической структуры данных, с которыми будет работать модуль. Сущности $e_i \in E = \{e_1, e_2..e_n\}$ и их атрибуты, которые будут участвовать в работе модуля;
- описания интерфейсных компонентов модуля – детальное описание компонентов отображения и редактирования информации с указанием атрибутов сущностей и результатов запросов, с которыми будет работать этот компонент, является ли обязательным для заполнения этот компонент, ограничения по внесению информации;
- перечня действий, которые пользователь может выполнить над модулем (реакция системы события, являющиеся действием пользователя).

Алгоритм работы модуля составляется на основе специальных икон [1] – графических элементов, отражающих отдельную программную конструкцию или заранее определенный шаблон (табл. 2).

В табл. 2 приведены некоторые иконы алгоритмического языка описания модулей и их символьное обозначение. Владимир Паронджанов доказал [1], что графический синтаксис языка может представлять собой графическое логическое исчисление («исчисление икон»). Исследования по реализации такого исчисления для АОМ еще ведутся, перечень икон и их символьное представление в настоящее время не являются полностью сформированным. Указанное обстоятельство не препятствует успешному практическому применению АОМ в соответствии с разработанной методикой.

Исходными данными для предлагаемого метода является:

- структура сущностей ИС;
- структура модулей ИС;
- набор функциональных элементов языка.

Для построения модели программного модуля в виде АОМ необходимо выполнение следующих шагов:

1) определение набора интерфейсов взаимодействия системы с модулем на основании модели ИС, полученной в результате синтеза модульной структуры [5];

2) определение сущностей, с которыми будет производиться работа в данном модуле на основании модели бизнес-процессов и логической модели данных;

3) определение элементов управления/ввода/отображения обрабатывающихся в модуле данных на основании модели бизнес-процессов и логической модели данных;

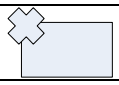
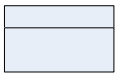
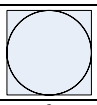
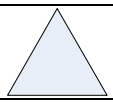
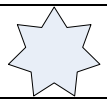
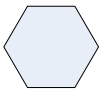
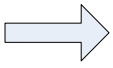
4) определения перечня действий и событий модуля на основании модели бизнес-процессов;

5) описание алгоритмов работы действий с помощью икон АОМ на основании модели бизнес-процессов предметной области.

В настоящее время построение АОМ-описания модуля выполняется на основании знаний аналитика с использованием указанных входных данных. Полученное описание передается программисту, а точнее – оператору по настройке исполняемой среды, который, внося в нее схему, фактически «программирует», а точнее – настраивает ИС под конкретный бизнес-процесс любой предметной области. Разработан визуальный редактор [7], предназначенный для работы с АОМ-схемами, который позволяет максимально сократить разрыв между постановкой задачи и готовой ИС.

Таблица 2

Иконы АОМ

Название	Обозначение	Описание	Икона
Добавление или изменение данных сущности	I[E{a1,a2..an},Id]	I – обозначение операции E – сущность, которая добавляется/изменяется a1, a2..an – атрибуты сущности, которые будут добавлены Id – идентификатор сущности	
Удаление	D[E]	D – обозначение операции E – сущность, которая будет удалена	
Получение единичного значение из БД	S[E{a},Id]	S – обозначение операции E – сущность, атрибут которой необходимо выбрать a – атрибут Id – идентификатор сущности	
Сохранение	Com	Сохранение данных в базу	
Установка параметра компонента	CSP[c,p,v]	CSP – обозначение операции c – имя компонента p – имя параметра v – значение параметра	
Вызов интерфейса	Call[I,[p1,p2..pi]]	Call - обозначение операции I – имя интерфейса p1, p2…pi – параметры интерфеса	
Вариант	V[p]	V – обозначение операции p – рассматриваемый параметр	
Имя ветки	B[p,v]	B – обозначение операции p – рассматриваемый параметр v – возможный вариант	
Цикл «ДЛЯ»	For[cp,i,j]	For – обозначение операции cp – переменная цикла i – начало цикла j – конец цикла	
Переменная	Var+name	Name – имя переменной	
Получение нового идентификатора	GNI[E]	GNI – обозначение операции E – тип сущности	
Сохранение в переменую	SIV[VarName,Val]	SIV – обозначение операции VarName – имя переменной Val – значение	
Загрузка данных для компонентов	LoadData	LoadData – обозначение операции	

Практическая реализация

На базе изложенной концепции и многолетнего опыта разработки информационных систем профессиональным коллективом научно-производственного предприятия "Deimand" был создан пакет программного обеспечения для проектирования и моделирования информационно-управляющих систем – "ПРИМИУС" [7] (Проектирование И Моделирование Информационно-Управляющих Систем). Данное средство реализует исполняющую среду ИС и предназначено для разработки и поддержки информационных систем различной направленности, сложности и масштаба, который дополнительно не требует привлечения услуг высококвалифицированных специалистов и затрат на обучение штатных программистов. Основной упор при создании был сделан на простоту построения пользовательских форм с помощью визуального редактора и упрощение работы с базами данных. Для возможности реализации логических алгоритмов был разработан собственный язык программирования.

Заключение

Алгоритмический язык описания программных модулей, рассмотренный в данной статье, формулирует новый подход к реализации принципа «программирование без программиста», расширяет его применение на новую область и обладает рядом достоинств:

- использование АОМ позволяет создавать ИС, ориентированные на сбор, хранение и обработку информации без использования труда высококвалифицированных специалистов;
- аналитику для построения схемы работы модуля не требуется знать языки программирования;
- АОМ не зависит от языка программирования, так как нет привязки к синтаксису языка программирования на этапе создания схемы работы модуля;
- нет привязки к особенностям предметной области, так как данные хранятся и обрабатываются в унифицированной статической структуре БД;
- привязка к способу реализации ИС происходит на уровне исполняющей среды, минуя дополнительную трансляцию в текст на языке программирования и компиляцию в исполняемый код.

Практическое использование разработанной методики и ПРИМИУС в ряде проектов в области медицинского и социального обслуживания показало высокую эффективность предлагаемого подхода и его реализации. Во время реализации систем были оценены на практике преимущества от использования средства ПРИМИУС:

- понижение требований к квалификации программиста – действующие проекты были реализованы сотрудниками после кратких двухмесячных курсов обучения работе с ПРИМИУС;
- уменьшение времени на разработку проекта, а следовательно, и затрат – при использовании ПРИМИУС время на разработку системы сократилось в 2 раза в сравнении с проектами, реализующимися с использованием средств разработки (C#, Delphi);
- централизация прикладной логики и логики хранения данных, упрощение поддержки и сопровождения ИС.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Пароджанов В. Д.* Как улучшить работу ума: Алгоритмы без программистов – это очень просто! – М.: Дело, 2001. – 360 с.
2. *Почечуев Н.В., Рогозов Ю.И.* Построение базы данных со статической структурой // Материалы IX ВНК студентов и аспирантов «Техническая кибернетика, радиоэлектроника и системы управления». – Таганрог: Изд-во ТТИ ЮФУ, 2008. – С. 189.
3. *Трофимов С.А.* CASE-технологии: практическая работа в Rational Rose – М: Изд-во «БИНОМ», 2001. – 272 с.
4. *Марка Д.А., Мак Гоуэн К.* Методология структурного анализа и проектирования. – М.: МетаТехнология, 1993.
5. *Микита Р.М.* Методы реализации модульного принципа построения информационных систем // Известия ТРТУ. Специальный выпуск. Технические науки. – Таганрог: Изд-во ТРТУ, 2006. – №9. – С. 76.
6. *Д. О`Лири.* ERP-системы. Современное планирование и управление ресурсами предприятия. – М.: Вершина, 2004.
7. *Рогозов Ю.И., Бутенков С.А., Свиридов А.С., Горбань Н.С., Дубровский А.А., Друппов С.А., Жибулис Ю.А., Стукотий Л.Н.* Метод создания инструментальный средств разработки автоматизированных информационно-управляющих систем // Мехатроника, автоматизация, управление. – 2008. – №1 (82). – С. 52.

УДК 681.3.067: 007.52: 611.81

В.С. Поликарпов, Е.В. Поликарпова

НАЦИОНАЛЬНАЯ БЕЗОПАСНОСТЬ В КОНТЕКСТЕ ИНТЕЛЛЕКТУАЛЬНЫХ ВОЙН

В связи с особенностями новых интеллектуальных войн XXI столетия возникают проблемы обеспечения национальной безопасности России. Следует иметь в виду то существенное обстоятельство, что **в этом сверхсложном мире национальная безопасность является комплексной по своему характеру**. Поэтому в этом плане представляет немалый интерес исследования М.С. Алёшенкова комплексной безопасности на основе научных методов познания, предполагающих в первую очередь, систематизацию научных истин о пространствах, их объектах, субъектах, причинно-следственных механизмах развития угроз, их переменных характеристик, способах и средствах оптимального прогнозирования [1]. Здесь предлагается систематизировать любые расчетно-вероятностные пространства негативных факторов и дается следующее структурирование опасностей и угроз: 1) по объектам (субъектам) расчетного пространства той или иной системы; 2) по величине ущерба; 3) по степени вероятности наступления негативного события; 4) по масштабу проявления негативных факторов; 5) по причинам возникновения; 6) по характеру проявления; 7) по типу нанесенного (прогнозируемого) ущерба; 8) по виду интенсивности угроз. На основе данных понятий выписывается общая характеристика опасностей и угроз в следующих расчетных пространствах: 1) социальное; 2) экономическое; 3) культурно-образовательное; 4) экологическое; 5) техногенное; 6) энергоинформационное; 7) военное; 8) медицинское; 9) демографическое; 10) правовое; 11) геополитическое; 12) космическое. Возникает проблема обеспечения комплексной по своему существу национальной безопасности России в контексте интеллектуальных войн.

Прежде всего, **России необходимо найти новые возможности, которые помогут ей не только адаптироваться к вызовам и угрозам XXI века ради**