21. Шейко В.М., Кушнаренко Н.М. Організація та методика науково-дослідницької діяльності [Текст]. 6-те вид.переробл і доповн. – К.:Знання, 2008. – 310с.
22. Хабутдінов Р.А. Концептуальна характеристика транспортної системи та її інтегративної властивості [Текст]/ Управління проектами, системний аналіз і логістика. Наук.журн. Гол.ред. Дмитриченко М.Ф. Вип.3. – К.: НТУ, 2006. – С.153-157.
23. Рахмангулов А.Н., Трофимов С.В., Корнилов С.Н. Управление транспортными системами. Теоретические основы [Текст].- Магнитогорск: МГТУ им. Г.И.Носова, 2001.-191с.
24. Прокофьева Т.А., Лопаткин О.М. Логистика транспортно-распределительных систем: Региональный аспект. [Текст]. – М.: РКонсульт, 2003. – 400с.
25. http://en.wikipedia.org/ - the free encyclopedia Wikipedia. – 21.12.2010.
26. http://www.geodiswilson.com/ - the site of company Geodis Wilson. - 21.12.2010.
27. Вельможин А.В. и др. Грузовые автомобильные перевозки [Текст]. – М.: Горячая линия-Телеком, 2006. – 560с.
28. ГОСТ 27.004-85. Надежность в технике. Системы технологические. Термины и определения [Электронный ресурс]. – Режим доступа: http://www.complexdoc.ru/ - 21.12.2010.
29. ГОСТ Р 50995.3.1-96. Технологическое обеспечение создания продукции. Технологическая подготовка производства [Электронный ресурс]. – Режим доступа: http://www.snip-info.ru/ - 21.12.2010.
30. Горев А.Э. Грузовые автомобильные перевозки [Текст]. – 2-е изд., стер. – М.: Изд.центр «Академия», 2004. – 288с.
31. ГОСТ 3.1109-82. Единая система технологической документации. Термины и определения основных понятий [Электронный ресурс]. Режим доступа: http://www.yondi.ru/ - 21.12.2010.

Пропонується близька до природної мови форма запису технічного завдання (ТЗ), що придатна для подальшої програмної інтерпретації. ТЗ представляється у вигляді модифікованого інформаційного графу Р-схеми

Ключові слова: технічне завдання, мова проектування, модель поведінки

Предлагается близкая к естественному языку форма записи технического задания (ТЗ), которая пригодна для дальнейшей программной интерпретации. ТЗ представляется в виде модифицированного информационного графа Р-схемы

Ключевые слова: техническое задание, язык проектирования, модель поведения

The language form for specification which is suitable for programming was observed. The specification is presented as a modified information graph of P-scheme

Keywords: specification, design language, behavior model

УДК 519.713

ЯЗЫК ПРОЕКТИРОВАНИЯ ТЕХНИЧЕСКОГО ЗАДАНИЯ СИСТЕМЫ УПРАВЛЕНИЯ

С.Л. Харченко
Старший преподаватель
Кафедра программного обеспечения ЭВМ
Харьковский национальный университет радиоэлектроники
пр. Ленина, 14, г. Харьков, Украина, 61166
Контактный тел.: 096-791-37-00
E-mail: khsln@yandex.ru

1. Введение

Современное производство немыслимо без разнообразных систем управления, которые функционируют на технологическом уровне управления производством и систем управления различными объектами. Одним из основных компонентов названных систем управления являются программы, в которых реализуется алгоритм управления, т.е. определяются реакции системы на внешние воздействия с учетом изменяющихся внутренних состояний самой системы, что в итоге можно назвать поведением системы управления.

Можно утверждать, что большинство систем управления в своем составе имеют подсистемы решающие однотипные задачи по эквивалентным алгоритмам. Поэтому применение компонентного подхода в проектировании, когда повторно используются уже созданные

и апробированные практикой компоненты системы управления, актуальная задача для проектировщика.

Так как задачи управления решаются для различных вычислительных и программных платформ то возникают проблемы с переносом уже имеющихся решений на другую платформу. Это, зачастую, требует затрат которые эквивалентны затратам на повторное проектирование. Поэтому задача переноса уже имеющихся решений с платформы на платформу одна из важных задач практики.

2. Постановка задачи

Можно утверждать, что создание формального подхода к проектированию технического задания - актуальная задача практики.

Формальный подход позволяет ввести в процесс создания технического задания на систему управления форму записи, которая будет пригодна для машинного хранения, а в дальнейшем такая запись может быть интерпретирована в последовательность операторов, которые реализуют этот элемент описания.

Применение формальной формы записи, в описании ТЗ, позволяет использовать её при построении модели проектируемой системы управления. Такая модель визуализирует процессы и их взаимодействие в системе управления, что позволяет говорить о моделировании поведения системы при имеющихся ограничениях.

Сама модель поведения и ограничений представляется как модифицированный информационный граф Р-схемы, который строится на принципах последовательной декомпозиции компонент системы управления до неделимого компонента системы.

Наличие возможности хранения и машинной обработки элементов конструкций модели поведения и ограничений даёт возможность использовать конфигурационное управление проектом, что позволяет иметь множество вариантов исполнения. А хранимая формальная запись технического задания позволяет применить множество претрансляторов, для преобразования формальной записи в последовательности операторов на различных языках программирования и для различных вычислительных платформ.

3. Требования к языку проектирования модели поведения и ограничений программной компоненты

Сложность понимания естественных языков при решении задач искусственного интеллекта объясняется такими факторами как потребность в больших объёмах знаний, способностей и опыта при его использовании, а для программ, в которых реализуется понимание естественного языка, требуется представление этих знаний. Так как понимание естественного языка связано с описанием взаимосвязей в реальном мире, со структурой самого языка и понимания, что языковые конструкции, это продукт некоторого агента, а сами языковые действия целенаправленны.

Так как на текущий момент времени комплексное решение проблемы создания естественного языка проектирования для систем управления отсутствует, то воспользуемся формальными подходами. В этом случае язык проектирования модели поведения (ЯПМП) для системы управления состоит из нескольких элементов, которые объединяются в рамках системы проектирования:

- текстового описания;
- текстографической части, отображающей ветвление графа управляющего алгоритма, описания условий и операторов, отнесенных к элементам графа;
- правил записи условий и действий для графических элементов конструкции языка.

А саму систему проектирования ТЗ можно рассматривать как CASE систему, в которой реализован одновременный множественный доступ к процессу проектирования и реализовано разграничение доступа разработчиков к компонентам создаваемой модели системы и к уже ранее реализованным компонентам.

4. Основные элементы содержательного описания языка

Так как любой проект начинается с обозначения имени, указания назначения и т.п., то и разработчик должен повторить эти действия, относительно выполняемого проекта. При этом разработчик устанавливает «имя проекта» и его аббревиатуру для системы проектирования.

Так как любой проект может иметь варианты исполнения, то необходимо в CASE системе, а именно в системе хранения, учесть возможность появления нового варианта реализации уже существующей компоненты. При этом сам механизм конфигурационного управления проектом встраивается в систему проектирования и должен предоставлять возможность оперировать всеми вариантами исполнения проекта.

Первым элементом содержательного описания должно быть описание назначения проектируемой компоненты. Описание выполняется без привязки к языку общения.

Проектировщик, выполнив описание назначения создаваемой компоненты, должен определить и условия её применения, т.е. определить ограничения. Такое описание требований и ограничений позволяет определить возможность использования компоненты в конкретных условиях. Ограничениями могут выступать как внутренние состояния системы управления, так и внешние факторы, воздействующие на систему через её интерфейс.

Важной деталью любого проекта является описание интерфейса создаваемой программной компоненты. Эта работа выполняется на модели компоненты «черный ящик» и направлена на определение списка входных и выходных параметров, их типов, размерности, месторасположения в порту обмена информацией и их назначения. Поэтому в системе проектирования жестко определяется перечень элементов описания и их характеристики.

При описании элементов интерфейса компоненты необходимо учитывать, что они могут быть - входными (in), выходными (out), совмещать признаки входных и выходных параметров (in/out). Внутри компоненты могут быть и локальные переменные (loc). Первые три типа - это элементы порта ввода/вывода, для которого характерны точное указание на разряды в конкрет-

ном слове порта, а четвертый - это вспомогательный элемент, который находится в оперативной памяти контроллера. Типы и размерность всех элементов задаются исходя из потребностей и возможностей компилятора, который предполагается использовать, и ограничений системы хранения проекта.

5. Графические нотации языка

Если взять за основу языка схему с распределенной памятью (Р-схема), т.е. когда используется информационный граф, у которого вершины это операнды операторов, а дуги отражают пересылку значений от выходов до входов. То в этом случае Р-схема (α) представляется как тройка ($\mathbf{G}_\alpha$, $\mathbf{J}_\alpha$, Ω_α), где $\mathbf{G}_\alpha$ - управляющий граф, Ω_α - множество интерпретаций приписывающих смысл символам функций и константам, а $\mathbf{J}_\alpha$ - информационный граф. Вершинами графа $\mathbf{J}_\alpha$ являются операнды операторов из $\mathbf{G}_\alpha$, а дуги соединяют выходы операторов с входами таким образом, что для любых операндов **a**, **c** и **d** выполняются следующие свойства:

- если **a** и **c** - несовместимые выходы и **(a, d)** - дуга графа $\mathbf{J}_\alpha$, то в $\mathbf{J}_\alpha$ нет дуги **(c, d)**;

- если **a** и **c** - связанные выходы и **(d, a)** - дуги графа $\mathbf{J}_\alpha$ то **(d, c)** является дугой в $\mathbf{J}_\alpha$.

В этом случае, если рассматривать цепочку выполнения **(старт, S_1, S_2, S_3, S_4, стоп)**, то $\mathbf{S}_1$ поместит на вход $\mathbf{S}_2$ значение $\mathbf{c}_1$, а на вход оператора **стоп** - значение $\mathbf{c}_2$. Выполнение $\mathbf{S}_2$ заменит значение, хранящееся на входе оператора **стоп**, на значение $\mathbf{f}(\mathbf{c}_i)$, которое и будет результатом выполнения всей схемы.

В Р-схемах элементами памяти являются входы операторов. Поэтому состояние памяти в Р-схеме сопоставляется с текущим значением входа операторов, и всякий **S** оператор, при своём выполнении, берет текущее значение со своих входов и может изменить текущее значение только у тех видов **b**, для которых **(a, b)** - дуга графа $\mathbf{J}_a$, где **a** - выход оператора **S**. Если **a** - обязательный выход, то выход **b**, в качестве нового текущего значения, получит значение, выработанное оператором **S** по выходу **a**, если же **a** - необязательный выход, то вычисленное оператором **S** по выходу, **a** значение используется лишь для модификации текущего значения входа **b**.

Тогда путь $\mathbf{P} = (\mathbf{S}_1, \ldots, \mathbf{S}_r)$ по Р-схеме можно называть маршрутом пары **(c, d)**. Если **c** - выход оператора $\mathbf{S}_1$, **d** - выход оператора $\mathbf{S}_r$ и не соединен дугой с выходом **d** в $\mathbf{J}\alpha$ ни один из выходов оператора $\mathbf{S}_1$, отличных от **c** и ни один из обязательных выходов внутренних операторов пути **P**.

На основании вышеизложенного можно выполнить построение языка моделирования, но возникают проблемы с удобством восприятия и построения конструкций. В нашем случае наиболее удобной формой восприятия и построения будет конструкция, у которой вершины отражают пересылку значений от выходов до входов, а дуги это операнды операторов.

Приняв это и введя в Р-схему новые элементы и идентификаторы, которые позволяют однозначно идентифицировать и позиционировать узлы и рёбра информационного графа, можно утверждать то, что получена модифицированная форма записи информационного графа Р-схемы, у которого текстовые и конструкционные элементы графа несут смысловую нагрузку.

Теперь рассмотрим элементы информационного графа.

Узел информационного графа

Узел графа выполняет объединяющую или расщепляющую функцию, указывает вид процесса, и какие ребра графа участвуют в процессе. Предполагается, что количество входящих и исходящих рёбер из узла не ограничивается.

Теперь рассмотрим этот элемент графа более подробно. Каждый узел имеет идентификатор - уникальный номер, который состоит из трех элементов разделенных точкой - номер проекта, номер компоненты и номер узла в компоненте. Так как узел, для наглядности, может быть повторен в диаграмме, то для таких элементов вводится дополнительный идентификатор, который указывает количество его повторений. Он должен быть записан через дефис к основному идентификатору.

Следующие элементы могут присутствовать или отсутствовать при описании узла графа исходя из сложившихся требований и условий.

Так как функционирование системы управления предполагает возникновение и завершение параллельных процессов, а моменты возникновения и завершения этих событий ассоциируются с узлами графа, то возникает необходимость в их управлении. Поэтому, с узлом графа может быть ассоциированы следующие события:

- указание на тип завершения параллельных событий (тип «и»/«или»), которые взаимодействуют с указанным узлом графа;

- указание на одновременный запуск процессов исходящих из одного узла графа (событие типа «и»);

- выбор одного из ребер исходящих из узла графа;

- безусловный переход от одного ребра графа к другому ребру графа.

Кроме этого с узлом графа можно связывать оператор, с помощью которого будут анализироваться условия, которые записанные над рёбрами исходящими из узла графа. Это может быть оператор «case» или оператор «if».

Программным способом достаточно сложно выбрать оптимальную конструкцию наиболее соответствующую возникшей ситуации, поэтому этот выбор можно предоставить человеку, выделив поле в котором можно указать предпочтение в выборе оператора. В общем случае решение этой проблемы возлагается на программу анализа, которая реализует ситуацию через систему вложенных операторов «if». Следовательно, вершина графа может быть представлена следующим образом, см. рис. 1.

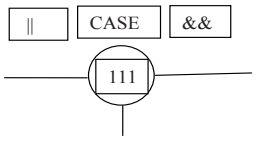

Рис. 1. Пример оформления узла графа

Теперь уточним элементы приведенных конструкций. Уникальный идентификатор узла графа записы-

вается в прямоугольник, вписанный в окружность, что соответствует узлу графа на диаграмме. Слева вверху, в прямоугольнике записывается условие, которое указывает, как должно быть реализовано завершение параллельных процессов приходящих в узел графа, а системный аналитик должен обеспечить такое событие введением перечня соответствующих инструкций (действий). Данное указание может быть выполнено, только если ранее были запущены параллельные процессы. В противном случае применение прямоугольника над узлом в левой части запрещено, так как этому событию отсутствует событие, которое указывает на запуск параллельных процессов, а это прямоугольник над узлом графа с указанием на запуск параллельных процессов (см. рис. 1, запись справа вверху над узлом графа).

Непосредственно над узлом графа программного компонента можно указать, какой оператор обработки условия более предпочтителен для данной ситуации. Это условный оператор 'case' или условный оператор 'if'. В случае отсутствия указания используется конструкция "if", в которой при числе ветвлений более трех применяется принцип вложенных друг в друга операторов условия. Все вышеизложенное взаимосвязано с условиями, которые записаны над ребрами графа, которые выходят из узла.

Ребра информационного графа

Ребра информационного графа могут быть представлены рядом элементов. Основной характеристикой ребра является то, что ребро исходит из узла графа и входит в узел графа, причем номера узлов должны быть разные. Ребро графа идентифицируется уникальной учетной записью, структура какой совпадает с учетной записью узла графа. Теперь рассмотрим их поочередно.

Горизонтальная прерывистая.

Ребро графа изображенное горизонтальной прерывистой линией используется исключительно для операций объявления. Над ребром графа отсутствуют какие-либо надписи, так как это «ребро выполняется» безусловно, и оно должно быть расположено в вершине графа (общего или локального) или до ребер несущих элементы действия. Под ребром записывается конструкция, которая определяет вид (входная или выходная), тип, размерность, «смещение» (это номер слова в буфере или в оперативной памяти), «маску» (определяет номер битов в слове) и имя переменной. Каждая запись отделяется от предыдущей записи разделителями, которые определены в языке описания как перевод строки или «точка с запятой», см. рис. 2.

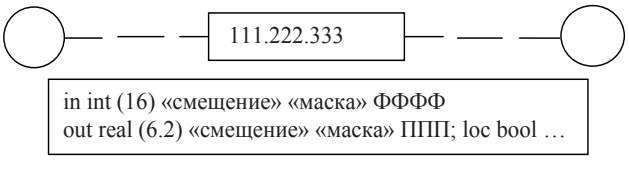

Рис. 2. Пример оформления ребра объявляющего вид, тип, размерность имени переменной

Нумерация ребра графа позволяет жестко идентифицировать место ребра в графе проекта.

Горизонтальная сплошная.

Над дугой записываются условия разрешающие прохождение по дуге, а под дугой – операторы или перечень операторов, которые необходимо выполнить при реализации этого ребра графа.

К элементу «условие», которое записывается над ребром графа, в прямоугольник, необходимо отнести следующие варианты:

- отсутствие записи – безусловное прохождение по ребру графа;
- «звездочка» - требуется обязательно «пройти» по ребру графа;
- любая форма записи условия – прохождение по ребру графа выполнить только при выполнении указанного условия.

Особо надо подчеркнуть, что конструкции модифицированного информационного графа Р-схемы содержащие условия должны иметь возможность, в случае если все перечисленные условия не выполнились, или повторно проверить условия или иметь возможность напрямую выйти из конструкции проверки условия. В случае ситуации с повторной проверкой условия необходимо предусмотреть возможность дальнейшего продолжения работы управляющей программы после n-го повторения цикла или после истечения интервала времени. Это требование должно выполняться обязательно, при создании модели поведения компоненты.

Условие, записываемое над ребром, может иметь вид - пустая строка, «звёздочка», простая форма условия или условие которое состоит из множества условий объединенных логическим выражением.

Первая конструкция, безусловное выполнение, используется в ситуации реализации последовательности выполнения операторов. Конструкция «звёздочка» используется в конструкциях реализующих условие, когда указывается альтернатива выполнения условия, а само условие может быть как «простым» так и «составным».

Операторы, которые записываются под ребром, могут иметь разнообразные формы. Это может быть операция присвоения, во всех её проявлениях, вызова процедуры, вызова функции и вызова компоненты (процедуру и функцию тоже можно рассматривать как компоненту) и.т.п. Количество элементов под ребром не ограничивается, но вводятся следующие правила – разделитель элементов в строке должен быть «;» или используется специальный символ «перевод строки - enter». Отличие разделителей в том, что нажатие клавиши «перевод строки» закрывает входной поток символов на текущей строке и выполняет его открытие на следующей строке. А при использовании разделителя «;» входной поток на текущей строке не закрывается.

Возможны варианты использования первой и второй формы записи при указании действия под ребром графа.

В этом случае ребро графа может иметь вид, см. рис. 3.

Следует помнить, что «условие» и «действе» жестко ассоциированы с ребром графа. Если количество элементов в действии более чем одно, то системе проектирования необходимо создать список из последовательно исполняемых элементов и привязать его к ребру графа.

Горизонтальная двойная.

Следующим графическим элементом есть ребро графа, которое изображено двойной чертой. Такой элемент используется для указания циклических структур графа, когда «последующая вершина связывается с предыдущей вершиной». Вариант замыкания на себя не рассматривается, так как ребро графа связано с двумя вершинами имеющих различные уникальные идентификаторы.

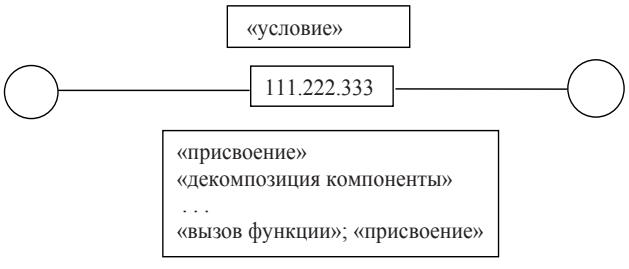

Рис. 3. Пример оформления действия, под ребром модифицированного Р-графа

В этом случае над ребром графа записывается условие, которое определяет выход из циклической структуры, т.е. определяется условие, при котором должно наступить событие исполнения «следующего элемента графа».

А под таким ребром должно записываться действие, которое приводит или способствует выполнению условия, которое записано над ребром графа, см. рис. 4.

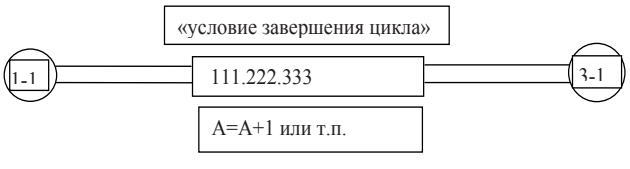

Рис. 4. Пример оформления ребра графа определяющего циклическую структуру

Вертикальная черта.

Данный элемент графа применяется как средство улучшения визуализации и не несет иных смысловых нагрузок. Как пример, можно рассмотреть вершину графа, из которого исходят два ребра. В традиционном отображении графа это вызовет неудобство при чтении условий и действий, которые отнесены к ребрам графа. В случае расщепления вершин графа это неудобство исчезает. Следует помнить, что ребра графа в первом и втором случаях имеют уникальные идентификаторы и несу информацию об одной вершине графа, из которой они выходят.

6. Правила работы с текстовыми нотациями информационного графа

Исходя из того, что разрабатываемый язык относится к интерпретируемым языкам, т.е. к языкам у которых каждая строка символов, перед исполнением, интерпретируется в последовательность операторов для выбранной платформы и языка программирования.

Таким интерпретатором, в общем случае, является претранслятор, который последовательность символов языка преобразует в последовательность команд языка для транслятора, исполняемого на вычислительной платформе.

Так как в среду проектирования включён контроль механизма использования имен переменных, то первым шагом проектирования программной компоненты должно быть объявление переменных, которые используются в программной компоненте и выполнено их описание.

Сама система проектирования оперирует следующими видами переменных – входной (in), выходной (out), реализующий действе вход/выход (in/out) и локальный (loc). Первые три вида переменных, наиболее вероятно, должны быть элементами «порта» обмена информацией системы управления с внешними устройствами. Для этих переменных введены следующие атрибуты:

- уникальное имя;
- имя переменной для программной реализации;
- указание на тип элемента;
- указание на место слова в порту ввода/вывода;
- маска (для определения занимаемых разрядов в слове порта ввода/вывода).

Все эти атрибуты имени должны быть объявлены на этапе описания имен интерфейса программной компоненты и должны быть внесены в базу данных проекта. Кроме этих данных необходимо хранить ещё один атрибут – вид переменной для данной компоненты.

С помощью атрибута вид можно проконтролировать была ли переменная сформирована ранее в проекте, можно ли с переменной выполнять указанную операцию. Например, переменная, имеющая вид «in», не должна участвовать в операции присвоения ей нового значения, так как это противоречит логике.

При рассмотрении локальной переменной необходимо помнить, что эта переменная не должна находиться в порту ввода/вывода, а должна находиться в оперативной памяти системы управления. При описании этой переменной используются все ранее объявленные поля. Единственным исключением может быть то, что поле «указание на место слова в порту ввода/вывода» будет пустое или там будет указатель слово в ОЗУ (смещение).

В графической части языка данное объявление должно иметь следующий вид, см. рис. 5.

Следует помнить, что «маска» может задаваться как в восьмеричном, так и шестнадцатеричном виде, тогда в первом случае перед маской должен находиться символ «B». Если используется переменная вида «loc», то смещение может присутствовать, а это в случае работы с буфером в ОЗУ или иметь код «0» в противном случае.

Резюмируя требования к записи декларативной записи имени переменных можно констатировать, что в строке символов следуют через разделитель, а это один или несколько пробелов, следующие элементы:

- имя переменной;
- вид переменной для данной компоненты;
- тип переменной;
- номер слова в буфере или смещение в ОЗУ (поле обязательно должно быть заполнено, если смещение отсутствует, то записывается символ «0»);

- восьмеричная или шестнадцатеричная маска, которая определяет разряды в порту обмена информацией или в слове ОЗУ, которые занимает переменная.

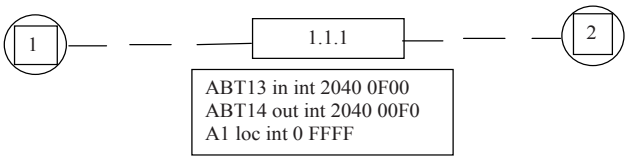

Рис. 5. Пример объявления переменных в графическом представлении поведения компоненты

Все требования могут динамически меняться исходя из требований претранслятора или требований разработчиков.

Следующим важным элементом, который определяет ветвление прохождения по дереву графа, является запись условия над ребром графа. Такая запись выполняется в прямоугольное поле над ребрами, которые имеют вид одинарной или двойной линии.

Вариант безусловного исполнения не будет подвергнут анализу, так как выполнение заявленных действий наступает всегда. Наиболее интересен вариант записи, когда формируются условия прохождения по ребру графа, рассмотрим его.

Так как было заявлено, что используется язык проектирования, работающий на принципах интерпретации, то можно предположить, что любая строка символов должна анализироваться по заранее установленным правилам.

Исходя из того, что простая логическая конструкция должна оперировать всеми условиями типа – больше, больше или рано, равно, меньше, меньше или равно и не равно. Эти условия записываются соответствующими знаками, которые входят в «алфавит» CASE системы. Сами знаки, в таком выражении являются разделителями имен, это в случае, когда не используются штатный разделитель – пробел. Возможно и дополнительное использование пробела в логическом выражении, тогда условие будет иметь вид, см. рис. 6.

Может возникнуть более сложная ситуация когда условие является составным. В этом случае используются круглые скобки, как указание на группировку условия, см. рис. 7. В этом случае сама круглая скобка (или фигурная) выступает как разделитель, так и средством группирования условия.

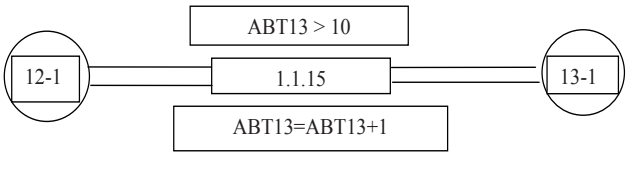

Рис. 6. Пример записи простого условия выхода из циклической структуры графа

В «составных» условиях круглые скобки имеют более низкий приоритет, чем условие, которое записано как указание на взаимоотношение кодов завершения структур реализованных в скобках. Само условие носит характер разделителя и эквивалентно пробелу.

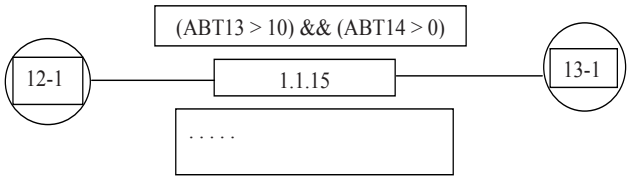

Рис. 7. Пример записи составного условия «прохождения» по ребру графа

Следует помнить, что в условии может находиться конструкция, которая определяет перечисление вариантов исполнения и которая должна быть завершена только после того как будет исчерпан список параметров заданных в условии, см. рис.8. Такая ситуация возникает при реализации управляющих элементов циклических структур. Аналогом такого условия является условие оператора «for».

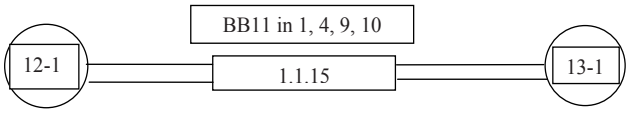

Рис. 8. Пример записи условия цикла основанного на перечислении

В приведенном примере условия ключевым словом является “in”. Оно отделяет имя переменной от списка значений и где признак перечисления символ “,” является разделителем в списке и имеет все права разделителя – «пробел». Следовательно, в этом варианте записи условия разделителями могут быть как пробелы, так и специальный символ “,” или ключевое слово “in”. Данный вариант записи условия применим только для использования с управляющими элементами циклических структур графа.

Подводя итог по описанию условия прохождения по ребру графа можно сказать, что основные конструкции формы записи приведены. Из них можно строить любые комбинации условий, которые запрещают или разрешают «прохождение» по ребру графа.

Следующим шагом будет определение правил записи, находящегося под ребром графа, оператора. Это возможно только для двух видов ребер графа, которые представлены при записи поведения - одинарной сплошной или двойной сплошной. Рассмотрим каждый случай.

В случае, когда применяются управляющие элементы цикличных структур графа (двойная сплошная линия) под ребром записывается оператор, который выполняет функцию счетчика или оператор аналогичного назначения, который формирует условие выхода из цикла, см. рис. 9.

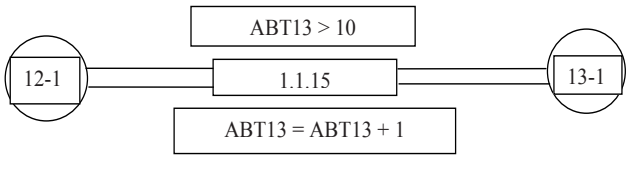

Рис. 9. Пример записи условия цикла основанного и формирования счетчика цикла

Исключение может вызывать только ситуация, когда используется перечисление, так как в этом случае запись под ребром отсутствует, см. рис. 8.

Если рассматривать реализацию управляющего элемента циклической структуры графа, то необходимо помнить о следующей последовательности действий - выполняется проверка условия или подстановка, если условие выполнено или список элементов исчерпан – то необходимо выйти из тела цикла. В противном случае выполняются последовательно все действия, которые объявлены в управляющей циклической структуре графа. Если в теле встретились команды безусловного завершения итерации или всего цикла, то они исполняются наравне со всеми действиями описания модели поведения графа.

Теперь рассмотрим правила записи операторов, которые располагаются под ребром графа, при этом ребро графа изображено одинарной линией. Следует помнить, что конструкции, которые расположены под ребром графа, рассматриваются как потоки символов, которые могут быть разделены специальными знаками. Так, символ «;» и «enter» являются стандартными разделителями потоков символов. В свою очередь, каждый поток символов может быть разделен на элементы при помощи пробела или его эквивалента в системе. Количество элементов, которые могут быть записаны под ребром графа, не ограничено.

Разобравшись с формой записи действия под ребром графа, следует уделить внимание и на саму запись действия. Так действием может быть любая форма присвоения, вызов процедуры или функции, специальный и т.п., которые записаны по правилам формирования строки символов shell UNIX.

Данное требование имеет довольно размытые границы, так как основные требования к синтаксису и семантике операторов реализуются в претрансляторе, который используется в конкретном проекте или могут быть определены для некоторой проектной организации как ОСТ. Следовательно, синтаксис и семантика операторов поведения компоненты определяется отраслевыми стандартами или соглашениями, которые приняты на момент начала проектирования модели поведения и соответствующего претранслятора.

Такой подход предполагает многообразие форм записи, что может помешать повторному использованию ранее спроектированных моделей поведения компонент и претрансляторов. Поэтому рекомендуется использовать синтаксис и семантику, которая реализована в оболочке bash ОС UNIX/LINUX. Реализация этого требования позволит выполнять проверку функциональности модели поведения в среде shell, т.е. на ранних этапах проектирования, как компонент, так и самой модели поведения и ограничений.

7. Выводы

В статье излагаются базовые основы создания формального языка для проектирования модели поведения программной компоненты системы управления, которая строится на основе модифицированного информационного графа Р-схемы.

Сама модель поведения программной компоненты входит как часть в документацию технического задания на проектирование системы.

Форма записи модели поведения программной системы пригодна для программной обработки, что позволяет применить множество претрансляторов, которые преобразуют формальную запись модели поведения в программную реализацию поведенческого алгоритма для различных программных и аппаратных платформ. Такой подход позволяет решить проблему переноса программной реализации системы управления на различные вычислительные и программные платформы.

Использование такого подхода может кардинально изменить технологию создания программного обеспечения систем управления.

Литература

1. Джордж Ф. Люггер Искусственный интеллект: стратегии и методы решения сложных проблем, 4-е издание. Пер. с англ.- М. : Изд. дом «Вильямс», 2003. – 864с.
2. Вельбицкий И.В. Технология программирования. – К.: Техніка, 1984 -279с.
3. Касьянов В.Н., Евстегнеев В.А. Графы в программировании: обработка, визуализация и применение. – СПб.: БХВ-Петербург, 2003. -1104с.
4. Харченко С.Л. Генерація компонент програмного обеспечениясистем управления по формальному описанию их моделей поведения и ограничений. – Восточно-европейский журнал передових технологий ; №2/3(44), 2010. –С. 33-35.