Программирование как формирование структуры ситуаций-действий в модели интеллект-окружение

В. С. Щепин

CMA Small Systems AB (Швеция)

vs@cma.ru

Ключевые слова: действия, ситуации, цикл, программа, интеллект, окружение, взаимодействие, фрактальность, диалог.

Предлагается новая парадигма программирования, как диалогового процесса взаимодействия интеллекта с окружающей средой с целью формирования семантической структуры, устанавливающей взаимосвязи между действиями и ситуациями. Работа выполнена в рамках задачи, поставленной А.С.Нариньяни в публикации на сайте www.softmarket.ru "Национальная идея и российский путь в информационные технологии 21 века". Автор опирается на исследования российских ученых и уверен, что знания, накопленные российской наукой, могут и должны быть использованы для создания по-настоящему эффективных информационных технологий, способных обеспечить технологический отрыв, без которого невозможно обойти конкурентов, существующих и процветающих в этом мире в настоящее время.

Рассматривается модель интеллектуального программируемого устройства способного:
1) Выполнять некоторый набор элементарных действий по отношению к внешней среде.
2) Распознавать некоторый набор ситуаций, зависящих от состояния (положения) устройства во внешней среде.
3) Выбирать целесообразное действие в зависимости от ситуации.
4) Формировать древовидную структуру действий-ситуаций-действий с циклами в процессе взаимодействия с окружением и сохранять ее в своей памяти.

1.Введение

А.С.Нариньяни в своей публикации [1] обращает внимание на целесообразность и возможность использования проектов в области ИТ для возрождения российской национальной идеи.

Все усиливающийся кризис в этой области на Западе только подтверждает точку зрения Нариньяни и открывает новые возможности для России достойно войти в индустрию разработки программного обеспечения, предложив и реализовав идеи, способные осуществить прорыв в этой сфере деятельности.

Возможно, что причиной кризиса ИТ является не обыкновенная цикличность экономических процессов, а отсутствие прочных фундаментальных основ и хаос в методологии разработки программного обеспечения. В связи с этим, можно поставить под сомнение парадигму Объектно-Ориентированного Программирования, которая возникла вместе с WINDOWS и является основой идеологии WINDOWS, да и всего современного программного обеспечения. Автор убежден, что для этого могут быть эффективно использованы знания, накопленные в России в области диалоговых и интеллектуальных технологий.

В данном сообщении предлагается новая парадигмы программирования, основанная на модели взаимодействия Интеллекта с окружающей средой. Автор сознает дискуссионный характер предлагаемой концепции и считает основной задачей этого сообщения инициировать творческую деятельность широкого круга специалистов в направлении разработки более "разумного" и надежного компьютерного программного обеспечения, чем то, что мы сейчас имеем. По мнению автора, длительная монополия MICROSOFT привела к стагнации научную деятельность в области программирования и является одной из причин разочарования потребителей и кризиса в компьютерной сфере.

2.Модель интеллектуального устройства

Отправной точкой служит приведенная в [2] китайская паремия (пословица): "Откроешь дверь - увидишь горы". Один из смыслов этой фразы заключается в установлении связи между действием и видением. Увидеть мир возможно, совершая некоторые действия (открыть глаза, как минимум). Логично дополнить приведенную пословицу связью между видением и действием: "Увидишь горы - выберешь путь", и далее: "Тронешься в путь - увидишь новые горы".

Отталкиваясь от этого описания, определим требования к интеллектуальному программируемому устройству, которое должно быть способно:

1) Сканировать некоторый входной поток данных.
2) Выполнять некоторый набор элементарных действий по обработке элементов данных и записи их в выходной поток или в промежуточное хранилище.
3) Фиксировать некоторый набор ситуаций, зависящих от входных или промежуточных данных.
4) Обращаться к программисту за указанием, какое действие необходимо выполнить в возникшей ситуации, если это не было определено для этой ситуации ранее. Вместо выбора очередного действия программист может указать на выполнявшееся ранее действие, уже зафиксированное в программе - в этом случае возникает цикл.
5) Автоматически формировать программу - древовидную структуру, состоящую из последовательностей действий, ветвлений при возникновении ситуаций и циклов.
6) Наглядно демонстрировать программисту результат работы программы после очередного шага расширения дерева действий-ситуаций для принятия решение о продолжении расширения программы или об откате на один или несколько шагов назад.
7) Выполнять откат результатов действий, выполненных после одного или нескольких предыдущих шагов формирования программы.

3. Модель окружения

Что является окружающей средой для модели программирования интеллекта? Данные, которые должны быть обработаны, стандартные средства интерфейса, такие как «окна», устройства ввода-вывода информации, существующие программные объекты, с которыми можно взаимодействовать, как с «черными ящиками», посылая сообщения и получая ответные сообщения – одним словом, мир разнообразных компьютерных объектов, данный нам в информационном взаимодействии. Восприятие информации искусственным интеллектом базируется на идее «курсора» - понятия, хорошо известного программистам СУБД, в частности, определенного в языке PL/SQL СУБД ORACLE. «Курсор» - это аналог концентрации внимания на чем-либо, «умственного взора». В «курсор» считывается информация, необходимая на текущем шаге алгоритма решения задачи. На основании той информации, что находится в данный момент в «курсоре», принимается решение о дальнейших действиях, в результате которых и «курсор» на следующем шаге может заполниться совершенно другой информацией.

4. Взаимодействие интеллекта с окружением

Рассмотрим механизм взаимодействия интеллекта с окружением, который может быть применен к процессу составления компьютерных программ. Более общая модель взаимодействия с интеллекта с окружением была описана в [3].

4.1. Цель и действия для ее достижения

Существует цель, являющаяся стимулом для деятельности. Деятельность Интеллекта происходит в окружающей среде во взаимодействии с ней. Интеллект выбирает действие, приближающее к цели. Если не возникло ситуаций, требующих какой-то особой реакции, это действие повторяется до тех пор, пока цель не будет достигнута. Можно привести пример из области программирования: задача преобразования строки символов. Пример действия: прочитать символ входной строки, поместить его в выходную и перейти к следующему символу. Действие может

поместить символ и в некоторую промежуточную строку, а Интеллект может по ходу достижения цели читать символы из промежуточных строк.

4.2. Ситуации, как следствия действий

Набор ситуаций, которые могут возникнуть после выполнения действия, зависит от самого действия и от состояния среды, в которой действует интеллект. Интеллект способен воспринимать ситуации, то есть определять, какая из них действительно возникла. Пример ситуации: во входной строке встретился символ-разделитель. Ситуацией может быть и определенное слово (последовательность символов) во входной или промежуточной строке.

4.3. Действия как реакции на ситуации

Предположим, что Интеллект способен выбрать действие, которое целесообразно выполнить в ответ на возникшую ситуацию. Он выполняет это действие и запоминает выполненные действия и ситуацию. Если после выполнения действия не возникло ситуации, требующей особой реакции, интеллект возвращается к выполнению первоначально выбранного действия, приближающего к цели. Если после выполнения этого действия снова возникла прежняя ситуация, то уже известно, какое действие нужно выполнить. Если возникла другая ситуация, то интеллект снова должен принять решение о выборе целесообразного действия в ответ на новую ситуацию.

4.4.Структура ситуаций-действий

При запоминании действий и ситуаций, в памяти интеллекта формируется расширяемая структура дерева действий-ситуаций-действий с циклами. В точках возникновения ситуаций происходит ветвление дерева. Потенциально ситуация может возникнуть после любого действия. Для возникшей ситуации должно быть выбрано новое действие (расширение дерева) либо начато повторение уже имеющейся в программе последовательности действий (цикл). Интеллект должен принять решение, в какую точку дерева (к какому действию) он должен возвратиться для продолжения движения к цели, то есть выбрать начало цикла. Начало цикла должно принадлежать пути от корня дерева до конечной вершины возврата в цикл.

5.Пример программы как структуры действий-ситуаций-действий

В качестве примера рассмотрим обработку строки символов. Окружением является строка символов, поступающая с некоторого устройства ввода информации. Предположим, что после обработки строки это устройство автоматически заменяет эту строку на новую. Цель состоит в формировании из фрагментов входной строки некоторой структуры в памяти компьютера или выходной строки в другом формате. Курсором будет промежуточная строка символов, использующаяся для сравнения с некоторыми фиксированными значениями, предусмотренными в списке возможных ситуаций (например, символы-разделители). Действия – это передвижение указателя по входной строке, считывание в курсор некоторой последовательности символов, копирование последовательности символов из входной строки в выходную и т.п.

Подобным образом можно строить программы:

- Преобразование строки в древовидную структуру, в частности распознавание XML – сообщений.
- Преобразование форматов текстовых файлов.
- Распознавание входных строк языков программирования.

6.Задача автоматического программирования

Концептуальное отличие предлагаемого подхода к программированию от традиционного легко понять по постановке задачи автоматического программирования.

6.1. Традиционная постановка задачи автоматического программирования

Формулировка из [4]: "Автоматическое Программирование: Дана спецификация, построить систему, реализующую эту спецификацию. Доказать, что реализация соответствует спецификации. Сделать это лучше бригады программистов."

Однако эта постановка вызывает много вопросов, в основном из-за неясности, что собой представляет спецификация, существует ли формальный язык для написания спецификаций, и поскольку очевидно, что в спецификации должно быть достаточно информации для построения системы, то не произойдет ли просто смещение большинства проблем программирования на этап построения спецификации, который предшествует автоматическому программированию. Если же предположить, что спецификации должны составляться на естественном языке, то все сводится к задаче понимания компьютером естественного языка плюс к задаче собственно программирования. А для этого надо сначала построить искусственный разум, а потом научить его понимать спецификации и программировать, что звучит пока что не вполне реально.

6.2. Автоматическое построение программы поведения интеллектуального устройства

Для предложенной выше модели задача автоматического программирования более определенна: «Даны возможные наборы исходных данных (тест-примеры), дан набор возможных действий, определена цель, описаны критерии достижения цели (приближения к цели в результате действий), описаны ситуации, которые могут возникнуть при выполнении действий, и критерии целесообразного выбора действий в ответ на возникающие ситуации. Построить программу достижения цели - древовидно-циклическую структуру действий-ситуаций-действий.»

Выбор действий может осуществляться методом проб и ошибок путем поиска с возвратами в дереве принимаемых решений. Существует риск большого количества вариантов перебора для сложных задач, что может не позволить найти решение за разумное время.

6.3. Построение программы в процессе диалога компьютера и программиста

Существует возможность человеко-машинного взаимодействия в процессе диалогового (полуавтоматического) программирования, когда выбор действий на каждом шаге формирования программы делает программист на основании своего опыта и интуиции. Компьютер при этом выполняет действия уже сформированной части программы, проверяет ситуации, организует последовательную обработку всех исходных тест-примеров и позволяет человеку контролировать процесс исполнения готовой части программы. При этом важно, что программист не пишет произвольные языковые конструкции, а только принимает решения о выборе действий, что гарантирует единообразие структуры, и программу сможет понять любой другой программист.

7.Программа - новое определение

Программа - управляющая структура, устанавливающая связь между ситуациями и действиями, которая формируется в памяти компьютера в процессе диалога с программистом или непосредственно при взаимодействии с окружением.

Определение основано на понятиях, описывающих элементарные акты взаимодействия интеллекта с окружением, в результате которого строится программа. При этом формируются традиционные конструкции языков программирования:

- Последовательности действий
- Ветвления (проверки ситуаций)
- Циклы
- Иерархию программ-подпрограмм
- Параллельные процессы.

Действие может быть элементарным, а может быть сложным и описываться целой программой. Ситуация также может быть сложной и описываться программой. Таким образом, в сложной

программе выполняется условие "часть подобна целому", и можно говорить о программе, как о фрактальном объекте. Возможности поэтапной (step-wise) разработки программ сохраняются в плане детализации действий и ситуаций (сверху-вниз или снизу-вверх – декомпозиция или интеграция), и к этому добавляется еще возможность поэтапного наращивания дерева ситуаций-действий при увеличении числа обрабатываемых ситуаций.

В отличие от объектно-ориентированного программирования, в рамках данной модели нет оснований говорить о данных, хранимых внутри описаний действий и ситуаций. Данные - это входной поток, единый для всех действий, который может состоять из данных, поступающих извне, данных, поступающих из внутреннего хранилища (памяти) системы и данных, сформированных предшествующими действиями, которые либо были записаны в долговременную память, либо остаются в текущем "курсоре" (кратковременной памяти Интеллекта), как временные. Такая точка зрения на данные сближает концепцию программирования для СУБД и обычного программирования. "Курсор", по-видимому, должен быть фрактальным, например, бесконечно расширяемым деревом. Описания действий и ситуаций могут включать в себя константы или иметь параметры.

Программа может запускать параллельное выполнение действий и затем контролировать завершение их выполнения либо прерывать их выполнение, если это необходимо. При этом ситуации, предусмотренные в программе, проверяются всегда после завершения действий. При параллельных действиях в программе могут возникать параллельно выполняемые ветви.

Программа может быть сформирована в компьютерной памяти фрактальной структуры в виде бесконечно расширяемого дерева [3].

8.Проблемы WINDOWS-программирования

WINDOWS придерживается материалистического подхода, обеспечивая средства для создания программных объектов и взаимодействия с ними. При этом сама ОС WINDOWS претендует на роль среды, где эти объекты существуют.

Однако большинство теоретических работ предшествующей эпохи рассматривали программирование с идеалистических позиций. Достаточно назвать следующие имена: Ю.И.Янов, А.П.Ершов, В.Ф.Турчин, С.С.Лавров, Д.А.Поспелов. Программа рассматривалась, как идея (способ) решения задачи или достижения цели. В книгах [5] и [6] можно найти немало идей такого рода.

В настоящее время программное обеспечение превратилось в отрасль промышленности с многомиллиардным оборотом. Но все еще не существует точного языка описания изделий, выпускаемых этой промышленностью, который был бы аналогом чертежей в машиностроении. Для описания программного обеспечения используется фактически некоторые аналоги естественных языков, позволяющие строить иерархические описания от общего к частному в виде классов и объектов. Однако при этом в действительности наследуется неточность и неоднозначность естественного языка. Созданные программные объекты имеют тенденцию вести себя непредсказуемо, и на практике программирование превращается в "черную магию". Разработки ПО часто хаотичны и выполняются без зафиксированных на бумаге спецификаций. Менеджер делит работу между программистами, которые взаимодействуют напрямую между собой и с WINDOWS, которая абсолютно непознаваема для менеджеров, так же впрочем, как и результаты работы программистов. Программист демонстрирует нечто и говорит: "Это работает" - но, к сожалению, работает не всегда и не везде. Тестирование помогает мало. Все доводится в процессе сдачи системы клиенту, и потом еще долго устраняются возникающие время от времени ошибки.

Хотя программы в основном страдают от ошибок самих программистов, но и претензии WINDOWS на роль среды, обеспечивающей функционирование произвольного количества объектов не очень обоснованы, потому что, строго говоря, не может не существовать ограничений на количество объектов и процессов. Эти ограничения зависят от конкретных параметров

конкретных компьютеров и на практике проявляются в непредсказуемые моменты в виде зависаний или резкого снижения производительности. Склонные к максимализму российские программисты обычно вводят излишний параллелизм и используют по максимуму заявленные возможности WINDOWS, что ухудшает надежность их программ, так как на практике WINDOWS оказывается вовсе не всемогущей.

9.Заключение

Человек не всегда понимает, как будет интерпретировать компьютер то, что он написал на языке программирования. Если же перейти на семантический уровень структуры действий–ситуаций-действий и в процессе программирования поэтапно формировать поведение интеллектуального устройства в некоторой среде, то это может привести к эффективному использованию представлений обычного здравого смысла при разработке и поддержке программ.

Программа не должна рассматриваться как текст, который должен понимать и выполнять компьютер. Предлагаемая модель программирования основывается на семантике, то есть на структуре действий, формируемой в процессе поведения и определяющей поведение компьютера. В каком-то смысле это возврат к своеобразному процедурному программированию, но программист не пишет текст программы, а указывает компьютеру, какие действия следует выполнять при возникновении различных ситуаций.

Если дать пользователю наборы действий и ситуаций для конкретных проблемных областей, то он сам сможет запрограммировать свои задачи по принципу: «Дать голодному "удочку", и пусть от ловит рыбу сам». В компьютер может быть встроена система полуавтоматического диалогового программирования с новыми возможностями автоматизации и более высокой надежностью.

10.Литература

1. А.С.Нариньяни, Ю.Н.Королев, Национальная идея и российский путь винформационные технологии 21 века, опубликовано на Интернет-сайте //www.softmarket.ru в июле 2002 года.

2. Т.С.Зевахина, Метафора мертвая и метафора живая: экспериментальный подход к паремиологии дунганского и китайского языков. //Труды международного семинара ДИАЛОГ 2002 «Компьютерная лингвистика и интеллектуальные технологии», т.1, стр. 154-162.

3. В.С.Щепин, Структурная модель прагматики диалога на основе фрактальной модели «интеллект – окружающая среда». //Труды международного семинара ДИАЛОГ 2002 «Компьютерная лингвистика и интеллектуальные технологии», т.1, стр. 619-625.

4. What Next? A Few Remaining Problems in Information Technology, Jim Gray, 1998 Turing Lecture, опубликовано на Интернет-сайте http://Research.Microsoft.com.

5. А.П. Ершов, Введение в теоретическое программирование, "Наука", Москва, 1977.

6. Д.А.Поспелов, В.Н.Пушкин, Мышление и Автоматы, "Советское Радио", Москва, 1972.

Computer program as an “actions – situations” structure in an Intelligence-Environment model

V. S. Shchepin

Key words: intelligence, environment, programming, action, vision, situation, tree structure, fractal memory, dialog**.**

The aim of this paper is to propose the paradigm of computer programming, which is different from the widely known object-oriented programming. This paradigm treats programming as intelligence–environment interaction driven process of “actions – situations – actions” semantic structure step-wise construction. It is an attempt to follow after publication of A.S.Narinyani and Yu.N.Korolev “The National Idea and Russian way into 21 century Information Technologies” that can be found on the Internet-site www.softmarket.ru. The author is convinced that knowledge legacy of the Russian scientists may be used for the development of truly effective information technologies capable to ensure a technological advantage, which is needed to beat prosperous competitors existing now in IT area.

A starting point is the nice Chinese proverb: "Open door – will see the mountains”. One of the possible meanings of this saying is the relationship between action and vision. You need to do something to see the world. When continue we may discover a relation between vision and action: "When see the mountains choose your way”. And further: "Keep on your way – will see another mountains”.

So, intelligence should be capable:

1) To perform a certain set of actions in relationship with an environment.
2) To recognize a certain set of situations depended on interaction between intelligence and environment.
3) To choose an action depending on situation.
4) To construct during its behavior in its memory a tree-like structure of actions-situations-actions and to reuse fragments of that structure depending on actions-situations history.

Based on above simple model new definition of computer program is proposed: "The program is an information structure needed to link situations and actions in order to establish goal-oriented behavior of a computer or another intelligent agent”. The program is created in computer memory during the dialog with the programmer or automatically during interaction between intelligence and environment.

The following traditional constructs of existing programming languages can be easily implemented in terms of actions-situations-actions structures:

- Sequences of actions
- Alternatives (after situation checking)
- Iterations
- Program modules hierarchy
- Parallel processes.

To represent tree-like program and data structures the fractal logical structure of computer memory is proposed. For prototyping author used a tree-structured double linked list indefinitely expandable at any node. The dialog mode of programming becomes principal. Instead of program text writing programmer must help computer to choose right actions in different situations arising in data processing. This approach has several advantages, mainly because of its simplicity and logical clarity. As a result programmer’s work will be more productive and reliable.