V. СИСТЕМА ОБРАЗОВАНИЯ РОССИИ КАК ОБЪЕКТ ИНФОРМАЦИОННОГО ПРОТИВОБОРСТВА

УДК 373.2.

© Титов А.А.
Titov A.

МОДЕЛЬ ВИРТУАЛЬНОЙ СРЕДЫ В СИСТЕМЕ ПЕРЕДАЧИ ЗНАНИЯ

MODEL OF THE VIRTUAL ENVIRONMENT IN THE TRANSMISSION OF KNOWLEDGE

Аннотация. *Статья посвящена проблеме управления передаваемым знаниям что, на наш взгляд, является достаточно важным при ведении информационного противоборства. В статье предложена математическая формулировка модели виртуальной среды, рассмотрены основные подходы к построению такой среды, определен набор элементарных операций, а также алгоритм передачи знания посредством рождения, модификации и гибели элементов в виртуальной среде.*

Annotation. *The article is devoted to the management of transferable knowledge that, in our opinion, is fairly important in the conduct of information warfare. In this paper, a mathematical formulation of the model of the virtual environment, the basic approaches to building an environment defined by a set of elementary operations, as well as the ability to transfer knowledge through birth, modification, and destruction of elements in the virtual environment.*

Ключевые слова. *Управление, знания, модель, виртуальная среда, программирование, алгоритм.*

Key words. *Management, knowledge model, the virtual environment, programming, algorithm.*

Изначально рассмотрим пример проектирования автомобиля. Итак, автомобиль должен содержать двигатель, подвеску, две передних фары, 4 колеса, и т.д. Автомобиль должен иметь возможность набирать и сбавлять скорость, совершать поворот и двигаться задним ходом. Известно, как взаимодействует двигатель и колёса, согласно каким законам движется распредвал и коленвал, а также как устроены дифференциалы.

Для создания эскиза необходимо описать все запчасти, из которых состоит автомобиль, а также то, каким образом эти запчасти взаимодействуют между собой. Кроме того, необходимо описать, что должен сделать пользователь, чтобы машина затормозила, или включился дальний свет фар. Результатом такой работы будет некоторый эскиз, который в ООП называется класс.

Класс – это способ описания сущности, определяющий состояние и поведение, зависящее от этого состояния, а также правила для взаимодействия с данной сущностью.

С точки зрения программирования класс можно рассматривать как набор данных (полей, атрибутов, членов класса) и функций для работы с ними (методов).

С точки зрения структуры программы класс является сложным типом данных.

В нашем случае класс будет отображать сущность – автомобиль. Атрибутами класса будут являться двигатель, подвеска, кузов, четыре колеса и т.д. Методами класса будет «открыть дверь», «нажать на педаль газа», а также «закачать порцию бензина из бензобака в двигатель». Первые два метода доступны для выполнения другим классам (в частности, классу «Водитель»). Последний описывает взаимодействия внутри класса и недоступен пользователю.

Создадим на заводе по такому эскизу машины, каждая из которых точно повторяет эскизные чертежи, все системы взаимодействуют именно так, как они были спроектированы. Но каждая машина уникальна. Они все имеют номер кузова и двигателя, но все эти номера разные, автомобили различаются цветом, а некоторые даже имеют литые вместо штампованных диски. Эти автомобили, по сути, являются объектами класса.

Объект (экземпляр) – это отдельный представитель класса, имеющий конкретное состояние и поведение, полностью определяемое классом. Объект имеет конкретные значения атрибутов и методы, работающие с этими значениями на основе правил, заданных в классе.

Титов Андрей Андреевич – соискатель, МИГиК, тел. (495)543-36-76.

Titov Andrey – applicant, MIGiK, tel. (495)543-36-76.

В данном примере, если класс – это некоторый абстрактный автомобиль из «мира идей», то объект – это конкретный автомобиль.

Когда пользователь садится за руль, он начинает взаимодействие с автомобилем. Обычно, взаимодействие происходит с помощью некоторого набора элементов: руль, педали, рычаг коробки переключения передач в автомобиле. Всегда существует некоторый ограниченный набор элементов управления, с которыми мы можем взаимодействовать.

Интерфейс – это набор методов класса, доступных для использования другими классами.

Очевидно, что интерфейсом класса будет являться набор всех его публичных методов в совокупности с набором публичных атрибутов. По сути, интерфейс специфицирует класс, чётко определяя все возможные действия над ним.

Хорошим примером интерфейса может служить приборная панель автомобиля, которая позволяет вызвать такие методы, как увеличение скорости, торможение, поворот, переключение передач, включение фар, и т.п, то есть все действия, которые может осуществить другой класс (в нашем случае – водитель) при взаимодействии с автомобилем.

Любые программные системы предназначены для моделирования реальных систем, поэтому очень важно в каких терминах следует описывать эти реальные системы. Описание в виде последовательности действий (процедурный подход к программированию) может оказаться довольно сложным. Объектно-ориентированный подход предлагает описывать системы в виде взаимодействия объектов.

Объект в ООП — это сущность, способная сохранять свое состояние (информацию) и обеспечивающая набор операций (поведение) для проверки и изменения этого состояния.

Объект в объектно-ориентированном программировании — это модель или абстракция реальной сущности в программной системе. Предмет моделирования при построении объекта в ООП может быть различным. Например, могут существовать следующие типы абстракции, используемые при построении объекта:

• абстракция понятия: объект — это модель какого-то понятия предметной области;

• абстракция действия: объект объединяет набор операций для выполнения какой-либо функции;

• абстракция виртуальной машины: объект объединяет операции, которые используются другими, более высокими уровнями абстракции;

• случайная абстракция: объект объединяет не связанные между собой операции.

Система — совокупность элементов, находящихся в определённых отношениях друг с другом и со средой [2,3].

Среда – совокупность объектов, изменение свойств которых влияет на систему, а также тех объектов, чьи свойства меняются под воздействием поведения системы. [2,3]

Таким образом, представим реальную систему с помощью программной модели ООП, а затем и математической, преимущественной для системы передачи данных.

```
TInterface=class(Tobject)  // Описываем класс интерфейсов: общих
                              свойств и методов для элементов и среды
    Description:integer; // блок определения общих свойств (положе
                              ния, идентификации, структуры, движения и др.)
    function methods:integer; // блок определения общих методов
                                  (движения)
    elements:Tobject; // блок определения общих классов
    Private
    Public
    End;
TElements=class(Tobject)    // Описываем класс элементов среды
    Description:Tdata_Elem; // блок определения свойств элементов
    function methods:Tdata_Elem; // блок определения методов элементов
    Procedure Create(A:Tdata_Elem, B:Tdata);// блок определения метода
                                                рождения элемента
    Procedure Destroy(A:Tdata_Elem, B:Tdata);// блок определения метода
                                                гибели элемента
    Procedure Modify(A:Tdata_Elem, B:Tdata);// блок определения метода
                                                модификации элемента
    Description: TInterface; // блок определения общих свойств
                                  с элементами, средой
    function methods: TInterface; // блок определения общих методов
                                      с элементами, средой
    Private
    Public
    End;
TEnvironment=class(Tobject)   //Описываем класс среды
    Description:Tdata_Env; // блок определения свойств среды
    function methods:Tdata_Env; // блок определения методов среды
    Procedure Create();// блок определения метода рождения среды
    Procedure Destroy();// блок определения метода гибели среды
    Procedure Modify();// блок определения метода модификации среды
    Elements:TElements; //блок определения элементов среды
    Interface:TInterface; //блок определения общих свойств и методов
                              среды с элементами
    Private
    Public
    End;
```

Например, согласно классическому подходу объектно-ориентированного программирования среду можно считать объектом, порожденным от своего прародителя (системы). Именно в таком понимании в среде можно выделить следующие основные ее компоненты: свойства среды, которые определяют ее размерность и различные описания; операции (методы) среды – набор процедур и функций, призванный для обеспечения жизнедеятельности среды в целом и ее составных элементов; элементы среды – сложные объекты со своими свойствами и методами, а также общие свойства и методы, присущие всем элементам среды и призванные для идентифи-

кации элементов в среде и порядком взаимодействия со средой и друг с другом.

С другой стороны, любую среду можно представить в виде композиции множеств свойств, операций (методов), множества самих элементов и множества ячеек памяти, хранящих информацию, благодаря которой среда способна развиваться

$$Env = \{S, D, El, AI\},$$

где $S = \{s_1,...,s_n\}, n \in N, |S| = n$ – множество свойств среды *Env*;

$D = \{d_1,...,d_k\}, k \in N, |D| = k$ – множество разрешенных операций в среде Env;

$El = \{el_1,...,el_m\}, m \in N, |El| = m$ – множество элементов существующих в среде Env, причем

$El_i = \{S_i^{'}, D_i^{'}, S_{0i}, D_{0i}, AI\}, i \in N$, т. е. для каждого элемента из множества *El* характерны *i* набор значений свойств и операций из множеств *S'*, *D'*, а также *i* набор общих свойств и операций S_0, D_0 для каждого элемента El_i, последнее определяет его идентификацию в среде и взаимодействие со средой.

Множество *AI* –определим, как множество, хранящее полученные знания в результате самообучения элементов и среды. Для упрощения задачи будем считать, что *AI* =0.

Будем понимать, что однотипные элементы (элементы одной и той же природы и структуры) идентифицируются в среде и взаимодействуют с ней одинаково, т.е. по одному и тому же шаблону, структура которого задана с помощью множеств S_0, D_0.

Так, например, среди необходимых операций для существования и развития среды можно выделить следующие операции $D = \{d_1, d_2, d_3\}$ [4]:

- рождения *i* элемента $d_1(S_{0i}, D_{0i})$;
- модификации *i* элемента $d_2(S_{0i}, D_{0i}, S_i, D_i)$;
- гибели *i* элемента $d_3(S_{0i}, D_{0i})$.

В общем виде описать преобразование среды можно с помощью теории автоматов [1]. В начальный момент времени среда находится в состоянии $Z_0(S, D, El, AI)$. В конечный момент времени среда находится в состоянии $Z_n(S, D, El, AI)$. Будем считать, что процесс переходов среды из одного состояния в другое вызван операциями (методами) множества *D* среды.

На 1-м шаге под действием операции из множества $D(S_{0i}, D_{0i}, S_i, D_i)$ среда переходит из состояния Z_0 в состояние Z_1, то есть $Z_1=Z_1\{Z_0(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$. Здесь целевая функция равна $Func_1\{Z_0(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$.

На 2-м шаге под действием операции из множества $D(S_{0i}, D_{0i}, S_i, D_i)$ среда переходит из состояния Z_1 в состояние Z_2, то есть $Z_2=Z_2\{Z_0(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$. Здесь целевая функция равна $Func_2\{Z_1(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$.

На 3-м шаге под действием операции из множества $D(S_{0i}, D_{0i}, S_i, D_i)$ среда переходит из состояния Z_2 в состояние Z_3, то есть $Z_3=Z_3\{Z_2(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$. Здесь целевая функция равна $Func_3\{Z_2(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$ и т. д.

На последнем *n*-м шаге под действием операции из множества $D(S_{0i}, D_{0i}, S_i, D_i)$ среда переходит из состояния Z_{n-1} в состояние Z_n, то есть $Z_n=Z_n\{Z_{n-1}(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$. Таким образом, целевая функция равна $Func_n\{Z_{n-1}(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$.

В ходе преобразования среды, на каком-либо шаге возможен исход, при котором среда может погибнуть. Определим критерий гибели среды, т.е. при каких условиях для *n*-го шага под действием операции из множества $D(S_{0i}, D_{0i}, S_i, D_i)$ будет выполнено равенства $Z_0=Z_0\{Z_n(S, D, El, AI), D(S_{0i}, D_{0i}, S_i, D_i)\}$, т.е. среда перейдет в начальное состояние Z_0. Постулатами гибели среды $Z_n(S, D, El, AI)$ в общем смысле может стать разрушение ее свойств *S*, поскольку именно они претерпевают изменения в ходе ее преобразования и рождения элементов *El*.

Так, например среда может погибнуть в случае необратимого деградирования ее свойств, т.е. $\lim_{k \to n} S_k = 0$, а также в случае ее информационной перегрузки, т.е. $\lim_{k \to n}(S_k - \bigcup_{j=0}^{count(El)} S_{0j}) \neq 0$.

Покажем применимость виртуальной среды на примере автомобиля для передачи данных. Превратим исходное описание в компьютерный файл *F*, который надо передать из прошлого в будущее в рамках виртуальной среды совершенствующихся автомобилей. Представим множество байт файла в виде $F=\{b_1,..,b_n\}$, где $n=|F|$ – мощность множества *F*. Элементы данного множества (байты) $\{b_1,..,b_n\}$ могут принимать одно из значений подмножества $C=\{c_0,..,c_{255}\}$ (таблица ASCII), причем $|C|=256$.

В первом случае для передачи соответствующего файла за *n*-шагов будем использовать операцию рождения *i*-го элемента $d_1(S_{0i}, D_{0i})$ в данной среде, причем $Func(S_{0i}, D_{0i})=b_i$, а для приема операцию гибели *i* элемента $d_3(S_{0i}, D_{0i})$, причем $b_i=Func(S_{0i}, D_{0i})$. При этом в первом случае будем использовать алгоритм *A*. В результате трансформаций среды под управлением файла вида $F=\{b_1,..,b_n\}$ фактически будет *передано состояние среды* $Z_n(S, D, El, AI)$, т.е.

$$\begin{cases} X=A(F,Z_0,S,D,El,AI), X=Z_n; \\ F=A(X,Z_0,S,D,El,AI), n=|F|. \end{cases}$$

Во втором случае для передачи соответствующего

файла за *n*-шагов будем использовать операцию рождения *i*-го элемента $d_1(S_{0i}, D_{0i})$ в данной среде, причем $El_i = \{S_i^{'}, D_i^{'}, S_{0i}, D_{0i}, AI\}, i \in N$, $Func(S_i^{'}, D_i^{'}) = b_i$, а для приема операцию гибели *i*-го элемента $d_3(S_{0i}, D_{0i})$, причем $b_i = Func(S_i^{'}, D_i^{'})$. При этом в первом случае будем использовать алгоритм *A*. В результате трансформаций среды под управлением файла вида $F = \{b_1,..,b_n\}$ фактически будет *передана последовательность операций,* вектор строка операций из множества *D*, $x = \overrightarrow{d_{i1},..,d_{in}}$ т.е.

$$\begin{cases} X=A(F,Z_0,S,D,El,AI), X = \overrightarrow{d_{i1},...,d_{in}}; \\ F = A(X,Z_0,S,D,El,AI), n = |F|. \end{cases}$$

Полученные данные X - представим в виде файла $X = \{b_1,..,b_n\}$, где $n = |X|$ – мощность множества X. Элементы данного множества (байты) $\{b_1,..,b_n\}$ могут принимать одно из значений подмножества $C = \{c_0,..,c_{255}\}$.

В первом случае все автомобили можно рассматривать, как некоторое «общество», в котором знание всеобъемлюще зафиксировано в различных информационных пространствах и для его получения достаточно лишь «подключиться» к соответствующему пространству.

Во втором случае автомобили можно рассматривать, как носитель знания, который из поколения в поколение передает знания с помощью операций самомодификации.

В целом, конечно, и само человечество является образцом среды для передачи знания.

Литература

1. Беллман Р. Динамическое программирование. Москва.-1960.

2. Большой Российский энциклопедический словарь.-М.:БРЭ.-2003.

3. Берталанфи Л. Фон. История и статус общей теории систем// системные исследования.-М.:Наука,1973.

4. СП. Расторгуев. Информационная война. Проблемы и модели. М.: «Гелиос АРВ», 2006.

Материал поступил в редакцию 12. 08. 2013 г.

Отпечатано в издательско-полиграфическом центре ЗАО «ПСТМ»
Подписано в печать 10.11.13. Формат 60x90/8. Усл. печ. л.9,5. Тираж 600 экз. Заказ 18/13.
Адрес редакции: 141092, Московская область, г. Юбилейный, а/я 4. Тел./факс: (095) 543-36-76 Тел. 543-36-76, 543-36-77
E-mail: pva@cpsias.msk.ru, Site: www.cpsias.ru