УДК 519.72

МОДЕЛЬ АВТОНОМНОГО ПОВЕДЕНИЯ ИНТЕЛЛЕКТУАЛЬНОГО ПРОГРАММНОГО АГЕНТА В ИНФОРМАЦИОННОМ ПРОСТРАНСТВЕ

В.А. Филатов, ХНУРЭ, Харьков, Украина

This article is devoted to a problem of management the corporate system's information space. The management is carried out on the technology of the program agents. The approach to construction the program agent's model on a basis of frame structure is considered. The frame acts as universal extension system. For the decision of concrete tasks the functional modules-slots can be added in it.

Base structure for model of behavior of the program agents in information environment is the model based on the finite-state machine. The offered models can be used for development the systems for administration of information resources in the allocated computing systems.

Введение

Внедрение корпоративных информационных систем как основы для комплексной автоматизации деятельности предприятий, офисов и учреждений направлено на поддержку принятия управленческих решений. Особое место среди методов управления информацией в таких системах занимают технологии потоков работ.

Потоки работ в общем случае можно представить как координируемое выполнение множества задач, выполняемых различными обрабатывающими объектами. Задача определяет некоторую работу, которую необходимо выполнить, и она может быть специфицирована различными программными средствами. Выполнение множества задач с привлечением различных обрабатывающих объектов контролируется и управляется специализированной программной системой – системой управления потоками работ (СУПР) [1].

Данная статья посвящена разработке концепции СУПР на основе технологии программных агентов.

Цель исследования

Целью проводимых исследований является разработка моделей и методов управления информационным пространством корпоративных систем на основе технологии программных агентов.

Формальное определение модели программного агента

Вычислительная среда современной корпорации, учреждения, офиса, как правило, представляет собой сервер, сетевое оборудование, локальные компьютеры, оргтехнику различного назначения, прикладное, системное и специальное программное обеспечение. Все ресурсы вычислительной системы – от файлов пользователя, структур баз данных, прикладных программ до системных компонент – хранятся в файлах, а файлы – в поименованных

каталогах. Такая технология организации вычислительной системы является концептуальной и не зависит ни от класса решаемых пользователем задач, ни от типа взаимодействия сервер – локальный компьютер. Таким образом, вычислительная среда является инвариантной составляющей любой информационной системы и образует информационное пространство.

Информационное пространство можно представить в виде абстрактной алгебраической системы

$$\mathbf{P} = < \mathbf{O}, \mathbf{S}, \mathbf{\Omega} >, \qquad (1)$$

где O – объекты информационного пространства; S – отношения между объектами O; Ω – операции манипулирования объектами в пространстве P.

В качестве объектов модели (1) могут выступать компоненты современной вычислительной системы – файлы всех типов, каталоги, логические и физические диски, персональные компьютеры. Отношение S между объектами информационного пространства определяет конкретную конфигурацию вычислительной среды $\mathbf{p_i} = < \mathbf{o_i}, \mathbf{s_i}, \mathbf{q_i} >$, где $\mathbf{p_i} \in \mathbf{P}, \mathbf{o_i} \in \mathbf{O}, \mathbf{s_i} \in \mathbf{S}, \mathbf{q_i} \in \mathbf{\Omega}$, ориентированную на конкретного пользователя или пользователей $\mathbf{u_i} \in \mathbf{U}, \mathbf{i} = \mathbf{1, N}$, где N – количество пользователей.

В зависимости от целей, поставленных перед пользователем и вычислительной средой, все действия выполняются на основе операций.

Примером множества элементарных операций $\mathbf{q_i}$ могут быть {<run>,<copy__from__to>,…..<delete>}. Все операции над объектами в зависимости от поставленной цели инициирует пользователь $\mathbf{u_i} \in \mathbf{U}$ информационного пространства на основе формализованного плана действий $\mathbf{x^*_{i1}(t), x^*_{i2}(t), ... x^*_{in}(t)}$ или унитарного действия $\mathbf{x^*_1(t)}$. Схема взаимодействия пользователь – информационное пространство представлена на рис.1.

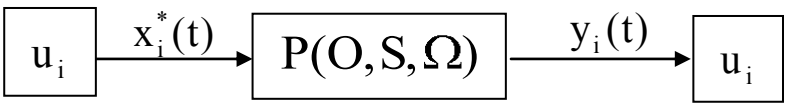

Рис.1. Схема взаимодействия пользователя с информационным пространством

В общем виде модель взаимодействия пользователя с информационным пространством можно представить как:

$$\mathbf{y_i(t) = p_i(x_i(t)),\ i \in I}, \qquad (2)$$

где $\mathbf{x_i(t)}$ – входное воздействие на информационное пространство со стороны пользователя $\mathbf{u_i} \in \mathbf{U}$; $\mathbf{y_i(t)}$ – реакция системы $\mathbf{P}$ на входное воздействие $\mathbf{x_i(t)}$, I – количество пользователей информационной системы P.

В общем случае $x_i(t)$ – элементарная задача, которую пользователь $u_i \in U$ решает при помощи информационной системы $P(O,S,\Omega)$. Формально задача $x_i(t)$ может быть определена как некоторое отображение $z_i : x_i \Rightarrow y_i$, для выполнения которой требуется операция $q_i, i \in I,$. Элементарная задача в терминах модели (1)-(2) может быть представлена как $z_i = (o_i, s_i, q_i, x_i(t), y_i(t))$.

Элементарные задачи могут быть объединены в функциональные задачи (ФЗ) путем объединения элементарных операций $q_i, i \in I,$ в последовательность взаимосвязанных операций алгоритма $a_k = \{q_1, q_i, ...q_n\}$, $a_k \in A$ $k \in K$, где A – множество алгоритмов решения функциональных задач, К – количество функциональных задач. Тогда функциональная задача может быть представлена в виде $z_к = (o_k, s_k, a_k, x_k(t), y_k(t))$.

В зависимости от структуры и взаимосвязи задач, решаемых пользователем, они могут объединяться в потоки задач. Потоком задач i – го пользователя будем называть такую их последовательность, для которой выполняется условие $y_{ij}(t) = p_{ij}(x_{ij}(t))$, $i \in I,$ $j \in K$, где К – количество задач в потоке, при ограничениях $x_{ij}(t) = y_{ij}(t), i \neq j,$. Схема линейного потока задач представлена на рис.2.

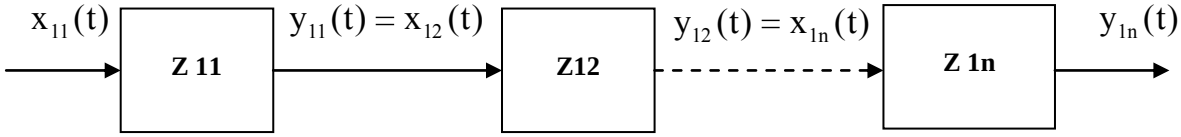

Рис.2. Линейный поток задач

Особенностью технологии потоков задач, представленной на рис.2, является то, что задачи могут образовывать как последовательный, так и произвольный последовательно-параллельный процесс их решения. Основным условием группировки задач в поток является согласование смежных задач по входу/выходу, при этом на промежуточных этапах под задачей может пониматься и действие пользователя, не требующее инструментальных средств поддержки вычислительной системы.

Рассмотрим частный случай потока задач, когда для решения каждой задачи z_{ij}, $i \in I,$ $j \in J$ существует инструментальный программный комплекс или программный модуль $m_j \in M$, где М – множество программных модулей, которые реализуют множество задач Z. При этом должно удовлетворяться одно из двух условий согласования задач в потоке. Первым условием является логическое завершение предыдущей задачи и

начало выполнения последующей задачи. Вторым – является условие согласования данных на промежуточных этапах решения. $y_{ij}(t) = x_{ij}(t), i \neq j,$ – выходные данные предыдущей задачи являются входными последующей. Тогда общее решение потока задач может быть представлено как последовательность решения задач z_{ij}, $i \in I$, $j \in K$.

Для рассматриваемого класса задач, где пользователь является "активным элементом" инициализации их выполнения, а форматы данных на промежуточных этапах согласования совпадают или последовательность их решения строго определена, может быть сформулирована и решена задача автономного управления потоком задач на основе технологии программных агентов.

Для формального представления задачи автономного управления информационными ресурсами вычислительной системы введем следующие определения.

Мультиагентное пространство определим как систему

$$G =< Z, M, S_z, G, \Omega_z >, \qquad (3)$$

где Z – множество задач информационного пространства P; M – множество программных модулей, реализации решений задач; S_z – множество отношений между задачами; G – множество программных агентов, реализующих функцию автономного управления и контроля выполнения задач в соответствии с S_z; Ω_z – множество операций по управлению Z, M, G в мультиагентном пространстве.

В соответствии с моделью (3) дадим определения основным компонентам мультиагентной системы:

Программный агент – программный модуль с элементами автономной интеллектуальной системы, осуществляющий целенаправленные действия в информационном пространстве по сценарию пользователя.

Мультиагентная система – система автономного администрирования и управления информационным пространством вычислительной системы на основе программных агентов.

Автономное администрирование и управление – выполнение последовательности операций в информационном пространстве над объектами на основе сценария.

Разработка модели программного агента

Одной из известных структур представления данных и знаний, удовлетворяющей концептуальной модели мультиагентного пространства (3), является фрейм [2].

Для реализации технологии взаимодействия программного агента с объектами вычислительной среды в качестве модели программного агента также использована фреймовая структура.

Рассмотрим модель программного агента, представленную в виде фрейма. В общем случае такая модель может быть записана в виде:

$$FR < R_1, C_{11,} C_{12,..} C_{1m,} >,... < R_2, C_{21}, C_{22,} ... C_{2m,} >, < R_{k1}, C_{km} > , \qquad (4)$$

где FR – имя фрейма (агента); пара <Ri,Ci> – i -й слот фрейма; Ri, – имя слота, Ci – значение слота.

Схема взаимодействия объектов мультиагентного пространства на основе модели "фрейм-программный агент" приведена на рис.3.

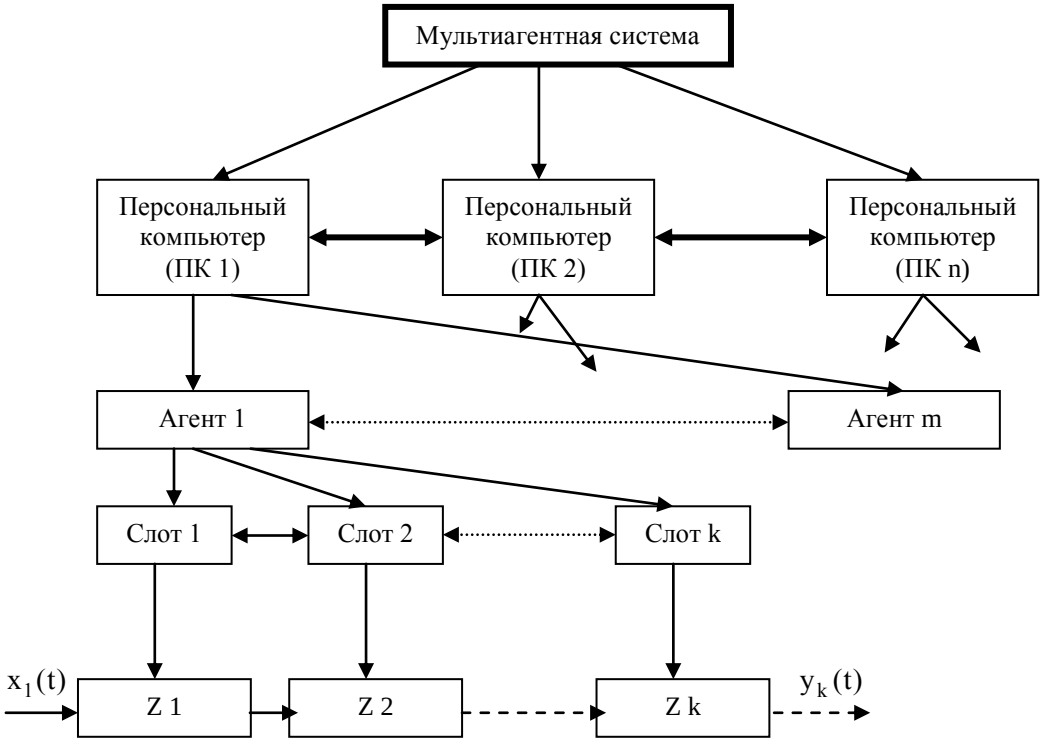

Рис.3. Схема взаимодействия задач и программных агентов в мультиагентном пространстве

Слот в модели (4) является логической конструкцией для реализации конкретных заданий фрейму - программному агенту. Слоты из фрейма можно удалять, добавлять, изменять функциональное назначение слота-задания. Перспективным направлением развития концепции "фрейм - программный агент", "слот - задание" является создание базы данных функциональных задач в виде типовых слотов. Однако в виде (4) классическое представление слота в виде <имя>,<значение> не может в полной мере отражать требования концептуальной модели (3).

Модифицируем структуру слота и приведем его к виду

$$\mathbf{Slot} = \langle \mathbf{Y}, \mathbf{D}, \mathbf{dom}, \mathbf{r_i}, \Theta, \Sigma \rangle , \qquad (5)$$

где Y – множество имен атрибутов, D – множество доменов, *dom* – отображение $Y \Rightarrow D$, r_i – модель-кортеж i-го задания агента, Σ – множество операций над отношениями.

$$r_i = \{\{R\}_i, \Omega_{ij}, V_i\}, \qquad (6)$$

где R_i – множество состояний кортежа r_i, V_i – множество ограничений целостности, Ω_{ij} – множество операций заданных на R_i, Θ – множество, определяющее начальные условия и признаки выполнения действий в структуре задания.

В соответствии с основными требованиями, предъявляемыми к свойствам программных агентов – автономностью функционирования и способностью выполнять целесообразные действия, рассмотрим один подходов к построению модели поведения программного агента. В качестве базовой структуры для построения модели поведения программных агентов в информационной среде может быть рассмотрена модель на основе конечного автомата достаточно подробно исследованная М.Л. Цетлиным [3] и представленная на рис.4.

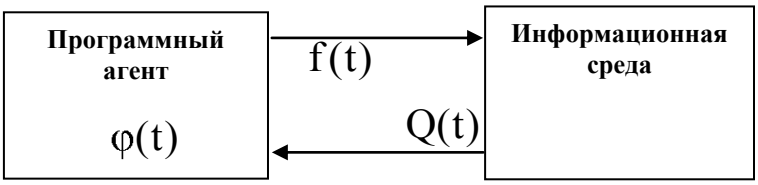

Рис.4. Схема взаимодействия программного агента с внешней средой

В предлагаемой модели "фрейм - программный агент", n-слотов образуют конечное число внутренних состояний агента $\varphi_i (i = 1,2,...n)$. Каждый слот решает одну задачу в информационной среде и обладает одним определенным действием $f_i (l = 1,2,...n)$, т.е. одному состоянию агента соответствует одно действие. На каждое состояние среда отвечает ответным сигналом $Q(t)$. Тогда поведение программного агента может быть задано уравнениями вида:

$$\varphi(t+1) = \Phi[\varphi(t), Q(t+1)], \qquad (7)$$

$$f(t) = F[\varphi(t)]. \qquad (8)$$

Уравнение (7) определяет смену внутренних состояний программного агента под воздействием входной переменной $Q(t)$, а уравнение (8) – зависимость выходного сигналов действий агента от его внутреннего состояния [4].

Предположим, что входная переменная $Q(t)$ в общем случае может принимать три значения "0", "1" и "неопределенность". Значение Q=1 будет соответствовать успешному завершению действия программного агента в момент времени $t = t^*$ на действие агента $f(t^*)$. Значение Q=0 – неудачному действию и действию, находящемуся в режиме обслуживания или выполнения будет соответствовать состояние Q ="неопределенность".

Переменная $\varphi(t)$ может принимать n различных значений $\varphi_1, \varphi_2, ..., \varphi_n$. Эти значения будем называть состоянием программного агента; n – его

информационная емкость [5]. Значения переменной $f(t)$ будем называть действиями программного агента. Тогда можно утверждать следующее: в момент времени *t* программный агент A_g находится в *j* -м состоянии $j = 1,2,..n$, если $\varphi(t) = \varphi_j$. Действие f_j называется действием, соответствующим состоянию φ_j, $F(\varphi_j) = f_j$. Уравнение (7) описывает зависимость действий программного агента от его состояний, уравнение (8) – изменения его состояний под воздействием входной переменной. Функция перехода программного агента (7) может быть задана различными способами. Наиболее эффективным из них является задание функции $\Phi[\varphi(t), Q(t+1)]$ системой матриц. Каждому значению Q_1 соответствует матрица состояний $\left\|\alpha_{ij}(Q_1)\right\|, (i, j = 1,2,\ldots,n)$, которая определяет смену состояний программного агента под воздействием сигнала Q_1.

Пример

Проиллюстрировать модель поведения программного агента в информационной среде можно на примере работы одной из подсистем региональной системы реагирования на чрезвычайные ситуации. В режиме действия по сценарию оперативный дежурный выполняет последовательный набор операций с различным программным обеспечением. Если результат действия положителен, он переходит к следующей задаче, если нет – расширяет область поиска и повторяет запрос.

Последовательность выполняемых действий приведена на рис.5.

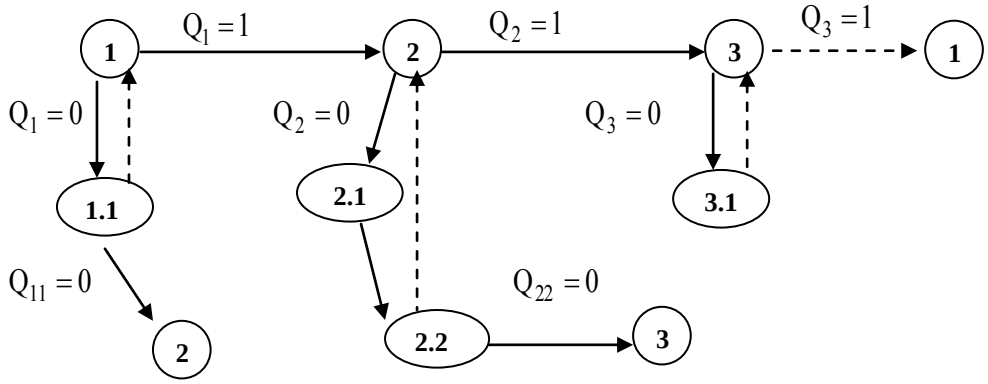

Рис.5. Последовательность выполнения действий оперативным дежурным

Рассмотрим пример логической модели программного агента, который реализует автономное выполнение задач, представленных на рис.5. Краткое описание задач:

- задача φ_1 по определению количества свободных бригад скорой помощи представляет собой запрос на выборку на языке SQL к базе данных D:\Base\med_gorod.dbf. Если запрос выполнен успешно, и расчетное

значение удовлетворяет заданному ограничению, перейти к задаче φ_2. В случае невыполнения условия задачи, перейти к φ_{11};

- подзадача φ_{11} представляет собой SQL-запрос на добавление к базе данных D:\Base\med_gorod.dbf файла D:\Base\med_oblast.dbf. После успешного завершения – вернуться к задаче φ_1, в противном случае – перейти к задаче φ_2;

Последующие действия оперативного дежурного согласно сценарию аналогичны описанным выше задачам и направлены на определение количества мест для эвакуации пострадавших и оповещение местных органов самоуправления.

В соответствии с (5) и (6) проектируется модель программного агента в виде отношения. Первые два атрибута соответствуют матрице переходов $Q_l(1)$ и $Q_l(0)$. В зависимости от результата действия осуществляется переход на новый слот и на новую задачу. Каждый слот имеет универсальную структуру и содержит пять атрибутов [6]. Первый атрибут KL задает уникальное имя слота или его ключ. Второй атрибут OBG определяет объект действия. Третий атрибут ACT определяет вид действия над объектом. Четвертый атрибут COND определяет условие выполнения действия. Следующий атрибут STATE показывает состояние выполнения действия: 1 – выполнено успешно, 0 – не выполнено, * – состояние "неопределенность". Описанная модель программного агента приведена в табл. 1.

Таблица 1

Логическая модель программного агента

$Q_l(1)$	$Q_l(0)$	TASK	KL	OBJ	ACT	COND	STATE
2	4	φ_1	1	D:\Base\med_gorod.dbf	Select from med_gorod…where [quant] = 40	40	
3	5	φ_{11}	2	D:\Base\med_oblast.dbf	Insert into med_gorod … from med_oblast		
1	7	φ_2	3	D:\Base\hotel_gorod.dbf	Select from hotel_gorod…where [quant] = 300	300	
1	2	φ_{21}	4	D:\Base\hotel_distr.dbf	Insert into hotel_gorod … from hotel_distr		
6	6	φ_{22}	5	D:\Base\hotel_oblast.dbf	Insert into hotel_gorod … from hotel_oblast		
3	3	φ_3	6	D:\Mail\admin_distr.mdb	Sent_massage		
3	1	φ_{31}	7	D:\Mobil\admin_distr.mdb	Sent_mobile_mass		

Выводы

В статье рассмотрены вопросы построения моделей информационного пространства и программных агентов для решения комплекса задач автономного управления информационными ресурсами в распределенных вычислительных системах. Предложен оригинальный подход к построению модели программного агента на основе фреймовой структуры. Фрейм

рассмотрен в виде универсальной расширяемой системы, в которую могут добавляться функциональные модули – слоты для решения конкретных задач.

В качестве базовой структуры для построения модели поведения программных агентов в информационной среде рассмотрена модель на основе конечного автомата. Предложенные модели могут использоваться для разработки мультиагентных систем администрирования информационных ресурсов распределенных вычислительных систем. В статье приводится пример, иллюстрирующий предлагаемый подход.

Литература

1. Пономаренко Л.А. Агентные технологии в задачах поиска информации и принятия решений / Л.А. Пономаренко, В.А. Филатов, Е.Е. Цыбульник // УСиМ: Управляющие системы и машины. – 2003. – №1. – С. 36–41.
2. Филатов В.А. Модель интеллектуального агента как абстрактный тип данных / В.А. Филатов, Л.Э. Чалая // Вестник Херсонского государственного технического университета. – 2003. – № 2(18). – С. 216–219.
3. Цетлин М. Л. О поведении конечных автоматов в случайных средах.– Автоматика и телемеханика XXII, № 10, – 1961.
4. Варшавский В.И. Коллективное поведение автоматов – М.: Наука. – 1973. – 408с.
5. Пономаренко Л. А. Програмні агентні технології в адмініструванні баз даних / Л. А. Пономаренко, В. О. Філатов. // Вісник Київського торговельно-економічного університету. – 2001. – №3. – С. 68–73.
6. Филатов В.А. Модель поведения автономного агента на основе теории автоматов / В.А. Филатов // Вестник Херсонского государственного технического университета. – Херсон: ХГТУ. – 2004. – №1(19). – С. 108 – 111.

Филатов Валентин Александрович
Харьковский национальный университет радиоэлектроники, кафедра искусственного интеллекта, доцент, канд. техн. наук Filatov_val@ukr.net