УДК 519.72

В.А. Филатов

ИССЛЕДОВАНИЕ КЛАССОВ ПРОГРАММНЫХ АГЕНТОВ В ИНТЕГРАЦИОННЫХ ТЕХНОЛОГИЯХ

Запропоновано модель поведінки програмних агентів у мультіагентному середовищі на основі теорії оптимального вибору варіантів і теорії кінцевих автоматів. Програмний агент виступає як абстрактна дискретна керуюча система, що на основі своїх дій і реакції на ці дії середовища, обирає стратегію поведінки. Досліджено класи програмних агентів на основі фреймової структури для рішення задач автономного управління інформаційними ресурсами розподілених регіональних інформаційних систем.

Введение.

На современном этапе развития компьютерно-ориентированных методов и средств управления информацией проблемы интеграции информационных ресурсов вычислительных систем занимают одно из центральных мест. Компьютерные технологии, особенно за последние десять лет, окончательно преодолели границы локального применения, а всемирная компьютерная сеть Internet открыла перед пользователями новые функциональные возможности. При этом интеграционный уровень информационных систем, а именно инструментальных средств, должен соответствовать требованиям пользователя.

Постановка проблемы исследования.

Рассматриваемые в статье вопросы построения классов программных агентов являются частью проводимых исследований, направленных на создание интеллектуальных систем интеграции и управления информационными ресурсами распределенных вычислительных систем.

Под интеграцией понимается коллективный доступ к корпоративным и региональным информационным ресурсам распределенных вычислительных систем и управление данными на основе интеллектуальной системы-посредника, которая позволит организовать доступ к разноформатной информации, содержащейся в сформировавшихся структурах – файлах данных и базах данных. В качестве системы-посредника может быть использована интеллектуальная система на основе технологии программных агентов [1].

Цель исследования.

Целью проводимых исследований является разработка эффективных моделей программных агентов и технологии интеграции информационных ресурсов распределенных вычислительных систем и распределенных баз данных.

Обоснование модели поведения программных агентов.

В контексте развития теории программных агентов как объектов, производящих целенаправленные действия в окружающей среде, центральной задачей становится исследование функциональной структуры и динамики деятельности агентов. В целях построения и обоснования архитектур программных агентов необходимы исследования общих принципов и внутренних механизмов деятельности, что предполагает разработку обобщенной схемы деятельности как сложной, автономной, самоорганизующейся системы.

В основу технологии поведения программных агентов может быть положена общая теория управления техническими системами [2]. Поведение может быть рассмотрено как частный случай общей теории оптимального управления объектами. В такой постановке программный агент выступает как некоторая абстрактная дискретная управляющая система, которая на основе своих действий и реакции на эти действия среды, выбирает оптимальную стратегию поведения.

Во многих реальных ситуациях выбор вариантов приходится осуществлять в условиях априорной неопределенности, когда по имеющимся данным нельзя заранее указать, какие из возможных вариантов следует выбирать, чтобы обеспечить достижение заданной цели. В этом случае достижение заданной цели возможно лишь на основе применения адаптивного подхода, смысл которого состоит в использовании текущей информации, получаемой в результате отдельных действий выбора, что позволяет компенсировать недостаток информации и реализовать оптимальную на классе систем стратегию управления.

Подход на основании теории оптимального выбора вариантов состоит в следующем – в каждый из последовательных моментов времени $t_n (n = 1,2...)$ необходимо выбирать вариант v_n из конечного множества возможных вариантов V. В результате произведенного выбора потери системы ξ_n представляют собой случайную величину – функцию элементарного исхода ω, и зависят от v_n и, возможно, от состояний системы. Реализуемая при этом последовательность вариантов $\{v_n\}$ должна быть такой, чтобы достигалась заданная цель, формулируемая в терминах предельных значений текущих средних потерь.

Наличие априорной неопределенности, состоящей в отсутствии точной информации о потерях системы и ее характеристиках, приводит к тому, что формирование последовательности вариантов $\{v_n\}$, обеспечивающей достижение целевого условия решаемой задачи, следует осуществлять в соответствии с адаптивным подходом. При этом выбор очередного варианта v_{n+1} производится на основе полученной к данному моменту времени

последовательности потерь $\xi_1, \xi_2, ..., \xi_n$, соответствующей реализованной последовательности вариантов $v_1, v_2, ..., v_n$. Это значит, что v_{n+1} является функцией от $v_1, v_2, ..., v_n$, $\xi_1, \xi_2, ..., \xi_n$, от момента времени n и от элементарного исхода ω. Таким образом:

$$v_{n+1} = T_n(v_1, v_2, ..., v_n; \xi_1, \xi_2, ..., \xi_n; \omega), \ n = 1,2,... \ , \qquad (1)$$

где ξ_n – в зависимости от задачи – либо скаляр, либо вектор.

Функцию T_n называют правилом выбора варианта v_{n+1}. Эта функция может быть как детерминированной, так и случайной (рандомизированной). Последовательность $\{T_n\}$ правил выбора определяет стратегию выбора вариантов или стратегию управления.

Детерминированные стратегии выбора составляют предмет изучения теории поведения детерминированных автоматов в случайных средах. Эти стратегии допускают возможность простой реализации с помощью детерминированных конечных автоматов. Они в основном ориентированы на задачи с бинарными потерями, хотя, как отмечалось выше, могут применяться и в других случаях. Кроме того, для них характерно обеспечение приближенно оптимального поведения, близость которого к оптимальному возрастает с увеличением глубины памяти автомата, что влечет за собой уменьшение скорости достижения решения и увеличивает сложность (число состояний) соответствующего автомата. Это же свойственно и стохастическим автоматам с постоянной структурой, которые реализуют рандомизированные стратегии выбора.

Отметим также, что в условиях полной информации оптимальная стратегия всегда принадлежит классу детерминированных стратегий:

$$v_{n+1} = T_n(\omega), \ n = 1,2,... \ . \qquad (2)$$

Наличие априорной неопределенности приводит к необходимости использовать более сложные рандомизированные стратегии. В теории поведения автоматов таким стратегиям соответствуют стохастические автоматы с переменной структурой.

Задача адаптивного выбора вариантов является одной из важнейших задач теории адаптивных систем, предмет изучения которой составляют разнообразные адаптивные алгоритмы, позволяющие оптимизировать функционирование систем в условиях априорной неопределенности. Суть этих алгоритмов заключается в том, что они указывают, как следует распоряжаться текущей информацией и в результате ее обработки воздействовать на работу системы, изменяя режим, или вариант ее функционирования с тем, чтобы обеспечить достижение заданной цели. В задачах адаптивного выбора вариантов такой текущей информацией являются реализации потерь, получаемых в результате выбора конкретных вариантов.

Разработка и исследование классов программных агентов.

В результате проведенного анализа методов адаптивного выбора вариантов при оптимизации систем с дискретным временем, предлагается следующая классификация классов-моделей системы «информационная среда − программный агент».

Класс А.

Тип состояния информационной системы:	*полная информация*
Функция потерь:	$\xi_n = \{1,0\}$ бинарная
Стратегия поведения:	$v_{n+1} = T_n(\omega)$
Модель поведения:	автоматная модель поведения, модель типа «автомат-строка»
Тип автомата:	детерминированный, стохастический с постоянной структурой
Тип программного агента:	*рефлексивный, автономный*

Класс С.

Тип состояния информационной системы:	*априорная неопределенность*
Функция потерь:	$\xi_n = \{1,0\}$ бинарная
Стратегия поведения:	$v_{n+1} = T_n(v_1, v_2, ..., v_n; \xi_1, \xi_2, ..., \xi_n; \omega)$
Модель поведения:	автоматная модель поведения
Тип автомата:	детерминированный, стохастический с переменной структурой
Тип программного агента:	*автономный*

Класс Е.

Тип состояния информационной системы:	*априорная неопределенность*
Функция потерь:	$\xi_n = \{1,0\}$ бинарная
Стратегия поведения:	$p_{n+1} = R_n(v_1, v_2, ..., v_n; p_1, p_2, ..., p_n; \xi_1, \xi_2, ..., \xi_n)$
Модель поведения:	автоматная модель поведения

Тип автомата: детерминированный, стохастический с переменной структурой
Тип программного агента: *интеллектуальный*

Рассмотрим логические модели и модели поведения программных агентов классов А и С.

Рефлексивный программный агент класса А.

Определение 1.

Рефлексивный агент – программный агент, выполняющий по заданию пользователя унитарное действие с объектом в информационном пространстве вычислительной системы при ограничениях на выполняемое действие в виде продукции типа IF <условие> – THEN <действие>.

Модель рефлексивного программного агента может быть представлена в виде:

$$\mathrm{Slot} = \langle \mathrm{U}, \mathrm{D}, \mathrm{dom}, \mathrm{r}, \theta, \Omega \rangle, \tag{3}$$

где U – множество имен атрибутов, D – множество доменов, $\mathbf{dom}$ – отображение $\mathrm{U} \Rightarrow \mathrm{D}$, Ω – множество операций, r – модель-кортеж одного слота, θ – множество, определяющее начальные условия и признаки выполнения действий в структуре задания.

Модель кортежа r будет иметь вид:

$$\mathrm{r} = \mathrm{R}, \mathrm{V}, \tag{4}$$

где R – состояние кортежа r, V – ограничения целостности, $\mathrm{U} = \{\mathrm{OBG}, \mathrm{ACT}, \mathrm{CON}, \mathrm{STA}\}$.

Логическое описание рефлексивного агента.

Рефлексивный программный агент представляет собой один слот-кортеж, при этом он может быть связан с одним объектом информационного пространства и содержит в своем арсенале только одно действие, определяемое спецификацией оператора [ACT]. Выполнение заданного действия может быть обусловлено ограничениями, задаваемыми в опции [CON]. Результат выполнения действия отражается в одном единственном кортеже-состоянии.

Модель поведения рефлексивного агента.

Поведение рефлексивного программного агента относится к примитивному типу поведения типа «условие-действие». При выполнении условия, заданного форматом атрибута [CON], выполняется действие, определенное спецификацией [ACT]. Результат выполнения задания отражается в единственном состоянии кортежа, признак выполнения 1 – успешное завершение действия, 0 – нет, и сохраняется в значении атрибута [STA]. Если условие задано для многократного повторения действий, то в состоянии слота-кортежа будет отображен результат последнего выполнения.

Пример.

Задание для рефлексивного программного агента:

Ежедневно в 18-00 копировать файл Itog.txt, находящийся в каталоге C\PR в каталог C\ARXIV. Структура рефлексивного программного агента приведена в табл. 1.

Таблица 1

Структура рефлексивного программного агента

OBG: *C\PR\Itog.txt*	**CON:** *Time=18-00*	**ACT:** *Copy from C\PR to C\ARXIV*	**STA:**
Itog.txt	18-00	Copy from C\PR to C\ARXIV	1

При анализе действий рефлексивного программного агента можно судить только о том, прошла ли корректно последняя операция по копированию файла в 18-00 или нет.

Рефлексивный программный агент класса А с памятью.

Определение 2.

Рефлексивный агент – программный агент, выполняющий по заданию пользователя унитарное действие с объектом в информационном пространстве вычислительной системы при ограничениях на выполняемое действие в виде продукции типа IF <условие> – THEN <действие>. В отличие от простого рефлексивного программного агента может сохранять результаты выполненных действий.

Модель рефлексивного программного агента с памятью может быть представлена:

$$\mathrm{Slot} = \langle \mathrm{U}, \mathrm{D}, \mathrm{dom}, \mathrm{r}, \theta, \Omega \rangle, \tag{5}$$

где U – множество имен атрибутов, D – множество доменов, $\mathbf{dom}$ – отображение $\mathrm{U} \Rightarrow \mathrm{D}$, Ω – множество операций, r – модель-кортеж одного слота, Θ – множество, определяющее начальные условия и признаки выполнения действий в структуре задания.

Кортеж r в модели (3.19) будет иметь вид:

$$r = R_j, \Omega, V \quad , \tag{6}$$

где R_j – множество состояний кортежа r, V – множество ограничений целостности, Ω – множество операций, заданных на R_j, $U = \{OBG, ACT, CON, STA\}$.

Логическое описание рефлексивного агента с памятью.

Рефлексивный программный агент с памятью представляет собой один слот-кортеж, при этом он может быть связан с одним объектом информационного пространства и содержит в своем арсенале только одно действие, определяемое спецификацией оператора [ACT]. Выполнение заданного действия может быть обусловлено ограничениями, задаваемыми в опции [CON]. Результат выполнения действия отражается в кортежах-состояниях R_{ij} слота задания.

Модель поведения рефлексивного агента с памятью.

Поведение рефлексивного программного агента с памятью относится к примитивному типу поведения типа «условие-действие». При выполнении условия, заданного форматом атрибута [CON], выполняется действие, определенное спецификацией [ACT]. Результат выполнения задания отражается на множестве состояниях-кортежах, признак выполнения 1 – успешное завершение действия, 0 – нет, и сохраняется в значении атрибута [STA].

Пример.

Задание для рефлексивного программного агента:

Каждый день в 18-00 копировать файл Itog.txt, находящийся в каталоге C\PR в каталог C\ARXIV. Структура рефлексивного программного агента приведена в табл. 2.

Таблица 2

Структура рефлексивного программного агента

OBG: *C\PR\Itog.txt*	**CON:** *Date Time=18-00*	**ACT:** *Copy from C\PR to C\ARXIV*	**STA:**
Itog.txt	21.08.04 18-00	Copy from C\PR to C\ARXIV	1
Itog.txt	22.08.04 18-00	Copy from C\PR to C\ARXIV	1
Itog.txt	23.08.04 18-00	Copy from C\PR to C\ARXIV	1

При анализе действий рефлексивного программного агента с памятью могут применяться операции, свойственные языку манипулирования данными SQL (SLECT, INSERT, DELETE …), что может значительно повысить эффективность оперативного анализа действий агента.

Автономный программный агент класса С.

Определение 3.

Автономный агент – программный агент, выполняющий по заданию пользователя линейную последовательность действий с объектами в информационном пространстве вычислительной системы при ограничениях на выполняемое действие, определяемых форматом атрибута [CON].

Модель рефлексивного программного агента с памятью может быть представлена в виде:

$$Slot = \langle U, D, dom, r_i, \theta, \Omega \rangle \ , \tag{7}$$

Где U – множество имен атрибутов, D – множество доменов, dom – отображение $U \Rightarrow D$, Ω – множество операций, r_i – модель-кортеж одного слота, Θ – множество, определяющее начальные условия и признаки выполнения действий в структуре задания.

Кортеж в модели представлен соотношением:

$$r_i = R_{ij}, \Omega_i, V_i \quad , \tag{8}$$

где R_{ij} – множество состояний кортежа r_i, V_i – множество ограничений целостности, $\Omega_i \subset \Omega$ – множество операций заданных на R_{ij}, $U = \{OBG, ACT, CON, STA\}$, $i = 1,2,3..., N$, $j = 1,2,3,..., K$, N – количество слотов-заданий, К – количество состояний i-го кортежа-задания.

Логическое описание автономного агента.

Автономный программный агент представляет собой набор слотов-кортежей. Каждый слот-кортеж может быть связан со своим объектом информационного пространства и может выполнять с ним действие, заданное спецификацией оператора [ACT]. Выполнение действия обусловлено ограничениями, задаваемыми в опции [CON]. Результат выполнения действия отражается в кортежах-состояниях R_{ij} слота-задания.

Модель поведения автономного агента.

Алгоритм поведения автономного программного состоит в следующем: задание, сформулированное в первом слоте-кортеже, выполняется k_1 количество раз, после чего управление передается на второй слот-кортеж.

Задача считается полностью выполненной тогда, когда действие r_n слота-кортежа выполнится k_n раз. Условие выполнения действия задается форматом атрибута [CON], а действие – спецификацией [ACT]. Результат выполнения задания отражается на множестве состояниях-кортежах, признак выполнения 1 – успешное завершение действия, 0 – нет, и сохраняется в значении атрибута [STA].

При анализе действий автономного программного агента, как и в случае с рефлексивным агентом с памятью, могут применяться операции языка манипулирования данными – SLECT, INSERT, DELETE и др. Отличие состоит в том, что обрабатываться могут данные межоперационных состояний, хранящихся в кортежах-состояниях отношений $r_i, i = 1,2,..., N$.

Автономный программный агент класса С с целесообразной моделью поведения.

Определение 4.

Автономный агент с целесообразным поведением – программный агент, выполняющий по заданию пользователя произвольную последовательность действий с объектами в информационном пространстве вычислительной системы при ограничениях на выполняемое действие, определяемых форматом атрибута [CON].

Модель автономного программного агента с целесообразным поведением аналогична модели (7), (8).

Логическое описание автономного агента с целесообразной моделью поведения.

Автономный программный агент рассматриваемого класса представляет собой набор слотов-кортежей. Каждый слот-кортеж может быть связан со своим объектом информационного пространства и может выполнять с ним действие, заданное спецификацией оператора [ACT]. Выполнение действия обусловлено ограничениями, задаваемыми в опции [CON]. Результат выполнения действия отражается в кортежах-состояниях R_{ij} слота задания.

Модель поведения автономного агента с целесообразной моделью поведения.

Модель автономного программного агента с целесообразным поведением значительно сложней всех рассмотренных ранее моделей.

Программный агент с целесообразным поведением рассматривается как некоторый объект, способный в каждый момент времени $t = 0,1,2,...$ воспринимать конечное число сигналов $s \in (s_1, s_2, ...s_n)$ и изменять в зависимости от них свое действие $f \in (f_1, f_2, ..., f_n)$. Каждое f_i действие имеет конечное число внутренних состояний $\varphi \in (\varphi_1, \varphi_2, ..., \varphi_m)$.

Программный агент находится в информационной среде, и действия f агента вызывают ответные реакции s среды C. Эти реакции, в свою очередь, являются для агента входными сигналами для принятия решения о дальнейших действиях [3].

Ограничимся рассмотрением простейшего случая, когда все возможные реакции среды $s \in (s_1, s_2, ...s_n)$ воспринимаются агентом, как относящиеся к одному из двух классов – классу благоприятных реакций (выигрыш, $s = 1$) и классу реакций неблагоприятных (проигрыш, $s = 0$).

Целесообразность поведения программного агента в информационной среде заключается в увеличении числа благоприятных реакций и уменьшении числа реакций неблагоприятных.

Поведение программного агента задается уравнением $f(t) = F(\varphi(t))$, указывающим зависимость действия $f(t)$ автомата в момент t от его состояния $\varphi(t)$ и матрицей $\|a_{ij}(s)\|, i, j = 1,2,..., m$. Так как рассматриваются программные агенты, воспринимающие лишь два состояния $s = 0$ и $s = 1$, то достаточно задать две такие матрицы $\|a_{ij}(0)\|$ и $\|a_{ij}(1)\|$. Таким образом, поведение программного агента может быть задано уравнениями:

$$\varphi(t+1) = \Phi(\varphi(t), s(t+1)) + \varsigma(t), \tag{11}$$

$$f(t) = F(\varphi(t)) + \vartheta(t), \tag{12}$$

где $\varsigma(t)$ и $\vartheta(t)$ – неконтролируемые возмущения по входу и выходу.

Уравнение (12) описывает зависимость действий от его состояний, а уравнение (11) – изменения его состояний под воздействием входной переменной $s(t)$. Матрица состояний детерминированного программного агента является простой: каждая ее строка при любом фиксированном значении s содержит один элемент, равный единице, а остальные элементы равны нулю [4]. Переходы состояний детерминированного программного агента определяются следующим образом: если в момент t автомат находится в состоянии φ_i то в момент $t+1$ он перейдет в такое состояние φ_i, для которого $a_{ij}(s(t+1)) = 0$.

С целью разработки эффективной логической модели программного агента рассмотрим подробно структуру матриц $\|a_{ij}(0)\|$ и $\|a_{ij}(1)\|$, а также логическую модель фрейма-агента. Анализ двух матриц переходов и структуры модели программного агента показывает, что в случае применения обобщенной интегрированной структуры фрейма - программного агента можно создать одну обобщенную модель.

Такая функциональная логическая модель фрейма программного агента будет включать атрибуты: ID – ключевой атрибут слота-кортежа; S1 – атрибут, значениями которого являются указатели слота-кортежа, на который передается управление, если реакция после выполнения данного действия положительна; S0 – атрибут, значениями которого являются указатели слота-кортежа, на который передается управление, если реакция после выполнения данного действия отрицательна; атрибуты фиксированного набора – OBG, CON, ACT, PROC, STA.

Таким образом, модифицированная схема слота в структуре фрейма программного агента с целесообразным поведением имеет на два служебных атрибута больше, чем обычная схема. Рассмотрим на примере модель поведения автономного программного агента такого класса. Программный агент должен выполнить три взаимосвязанных действия f_1, f_2, f_3.

Алгоритм поведения автономного программного агента с целесообразным поведением состоит в следующем: задание (выполнение действия f_1), сформулированное в первом слоте-кортеже, выполняется до тех пор, пока не изменится реакция информационной среды на это действие $1 \to 0$. При таком изменении параметра $s(t)$ управление передается на второй слот-кортеж. Действие f_2 сменится на действие f_3, тоже только при изменении $s(t) = 1 \to 0$. Как только реакция на действие отрицательна, тип действия изменяется; при положительной реакции объект остается в предыдущем состоянии [5]. Компактно в терминах разработанной модели автономного программного агента с целесообразным поведением логическая схема выглядит следующим образом (табл. 3.)

Таблица 3

Логическая модель программного агента с целесообразным поведением

ID	S1	S0	OBG	CON	ACT	STA
Slot 1	1	2				
Slot 2	2	3				
Slot 3	3	1				

Выводы.

В статье рассмотрены вопросы построения моделей автономного поведения программного агента для решения задач управления информационными ресурсами вычислительной системы. Предложена и теоретически обоснована классификация программных агентов, проведено исследование этих классов, что позволило обосновать модели программного агента в терминах теории коллективного поведения автоматов, построена автоматная модель программного агента и исследованы поведенческие аспекты классов программных агентов.

Рразработаны и исследованы классы программных агентов: рефлексивный агент, автономный и интеллектуальный программный агент. Рассмотрены модели поведения программного агента для решения различных задач управления информационными ресурсами вычислительной системы, реализация программного агента в виде фреймовой структуры открывает широкие перспективы создания инструментария работы с программными агентами.

ЛИТЕРАТУРА:

1. Пономаренко Л.А., Филатов В.А., Цыбульник Е.Е. Агентные технологии в задачах поиска информации и принятия решений // Международный научный журнал "Управляющие системы и машины". № 1 – 2003. – С. 36–41.

2. Назин А.В., Позняк А.С. Адаптивный выбор вариантов: рекуррентные алгоритмы. М.: Наука, 1986. 288 с.

3. Цетлин М. Л. Исследования по теории автоматов и моделированию биологических систем.– М.: "Наука", 1969. – 316с.

4. Варшавский В.И. Коллективное поведение автоматов. – М.: "Наука", 1973. – 408с.

5. Филатов В.А. Модель автономного поведения интеллектуального программного агента в информационном пространстве // Сборник научных трудов ДНГУ. Днепропетровск: НГУ, 2004. № 19. том 2 – С. 127–135.

Филатов Валентин Александрович

Кандидат технических наук, доцент кафедры Искусственного интеллекта Харьковского национального университета радиоэлектроники Filatov_val@ukr.net

Научные интересы: базы данных и знаний, агентные технологии, мультиагентные системы, извлечение знаний из данных.

ИССЛЕДОВАНИЕ КЛАССОВ ПРОГРАММНЫХ АГЕНТОВ В ИНТЕГРАЦИОННЫХ ТЕХНОЛОГИЯХ

В.А. Филатов

В статье предложена классификация программных агентов, проведенное исследование позволило обосновать формальную модель программного агента в терминах теории коллективного поведения автоматов. Исследованы классы программных агентов: рефлексивный агент, автономный и интеллектуальный программный агент. Рассмотрены модели поведения программного агента для решения различных задач управления информационными ресурсами вычислительной системы. Реализация логической модели программного агента в виде фреймовой структуры открывает широкие перспективы создания инструментария работы с программными агентами.

RESEARCH PROGRAM AGENTS' CLASSES IN THE INTEGRATIONAL TECHNOLOGIES

V. Filatov

In given paper the classification of the program agents is offered. The carried out research has allowed to prove the formal model of the program agent in the terms of the collective behavior theory the automatic devices. The classes of the program agents are investigated: the reflexive agent, independent and intellectual program agent. The models of program agent's behavior for the decision of various tasks of information resources' management of the computing system are considered. The realization of logic model of the program agent as frame structure opens wide opportunities of work toolkit creation with the program agents.

ДОСЛІДЖЕННЯ КЛАСІВ ПРОГРАМНИХ АГЕНТІВ У ІНТЕГРАЦІЙНИХ ТЕХНОЛОГІЯХ

В.О. Філатов

Запропоновано модель поведінки програмних агентів у мультіагентному середовищі на основі теорії оптимального вибору варіантів і теорії кінцевих автоматів. Програмний агент виступає як абстрактна дискретна керуюча система, що на основі своїх дій і реакції на ці дії середовища, обирає стратегію поведінки. Досліджено класи програмних агентів на основі фреймової структури для рішення задач автономного управління інформаційними ресурсами розподілених регіональних інформаційних систем.