УДК 004.94:004.8

ПРОБЛЕМЫ СОЗДАНИЯ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПРОГРАММИРОВАНИЯ ЛОГИКИ ПОВЕДЕНИЯ АГЕНТОВ В МУЛЬТИАГЕНТНЫХ СИСТЕМАХ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДВУХСТОРОННИХ БОЕВЫХ ДЕЙСТВИЙ

К. В. Иванов, М. В. Галкин*, А. И. Сайфуллин, Р. Н. Сайфуллина, Д. В. Девятых
(ФГУП "РФЯЦ-ВНИИЭФ", г. Саров Нижегородской области)

Рассмотрены ключевые аспекты создания инструментальных средств программирования логики агентов в мультиагентных системах и освещены проблемы, возникающие при их разработке. Дан краткий обзор существующих инструментальных средств программирования логики агентов, в котором отмечены как зарубежные, так и отечественные программные продукты.

Основной акцент сделан на имитационном моделировании двухсторонних боевых действий, ориентированном на специалистов предметной области. Для этого выделены три типа агентов: *тактическая единица*, *тактическая группа* и *объединение*. Показана структура их связи и взаимодействия, сформулированы особенности каждого типа агентов. Для всех типов агентов предложено решение, определяющее, каким должно быть наиболее подходящее инструментальное средство для программирования логики их поведения, которое было бы удобно для применения специалистами предметной области.

Ключевые слова: имитационное моделирование, агент, мультиагентная система, программирование, логика поведения, система управления.

Введение

В современной литературе встречается множество определений понятия *мультиагентная система* (МАС). Однако в контексте проблем, затрагиваемых данной статьей, целесообразно обратиться к определению Дж. Люгера, в котором под МАС понимается вычислительная программа, решатели которой расположены в некоторой среде и каждый из них способен к гибким, автономным и социально-организованным действиям в направлении предопределенной цели [1]. В данной статье рассматривается подкласс МАС — МАС имитационного моделирования (МСИМ) двухсторонних боевых действий (ДБД).

Термин *агент* происходит от латинского глагола "agere", что означает "действовать", "двигать", "править", "управлять". В энциклопедическом словаре Ф. А. Брокгауза и И. А. Ефрона дано следующее определение: "агент — деятель, лицо, действующее по поручению или полномочию другого" [2]. Это определение правильно выражает суть агента МСИМ, который может функционировать автономно от имени своего владельца (человека-оператора) и решать разнообразные задачи по обработке информации.[1] Для успешной работы агент должен обладать достаточными интеллектуальными способностями, иметь возможность взаимодействия с владельцем для получения заданий и передачи результатов, должен ориентироваться в среде своего существования и принимать необходимые решения.

*Галкин Максим Владимирович, научный сотрудник, *e-mail:* MVGalkin@vniief.ru

[1] Исходя из данного определения агент МСИМ, будучи программой, воспринимается как одушевленный предмет, наделенный способностями человека.

Две базовые характеристики — *автономность* и *целенаправленность* — позволяют отличать агента от других программных и аппаратных объектов (модулей, подпрограмм, процедур и т. п.). Автономность агента означает его способность взаимодействовать со средой без прямого участия других агентов, для чего он должен уметь контролировать свое внутреннее состояние и выполняемые действия. Целенаправленность означает, что соответствующий агенту программный код выполняется не автоматически, а по решению самого агента в зависимости от ситуации и согласно поставленным перед ним целям.

Наличие целесообразности поведения требует, чтобы агент обладал свойством *реактивности*. Такой уровень интеллекта соответствует рефлекторному поведению животного. Если же агент обладает знаниями о среде, собственных целях и способах их достижения, то такой агент может быть назван разумным (*когнитивным*). Таким образом, может быть проведена граница между интеллектуальными и неинтеллектуальными агентами.

Еще одна характеристика агента — *коммуникабельность* — предполагает необходимость решения своих задач совместно с другими агентами и наличие развитых протоколов коммуникации.

Обзор существующих инструментальных средств программирования логики агентов

Перечень систем, ориентированных на решение задач, связанных с программированием логики агентов, довольно обширен. Наиболее известные проекты — AgentBuilder [3], Ascape [4], Bee-gent [5], CABLE [6], Decaf [7], FIPA-OS [8], Grasshopper [9], Gypsy [10], JADE [11], JASON [12], JATLite [13], JAFMAS [14], MAML [15], ProcessLink [16], Swarm [17], ZEUS [18].

Среди отечественных разработок большую известность получили инструментальные средства группы интеллектуальных систем Санкт-Петербургского института информатики и автоматизации РАН, возглавляемой В. И. Городецким [19]. Концептуальной основой развиваемой коллективом Городецкого технологии разработки МАС является выделение *общности* реализуемых функций и представление их в виде классов и структур данных, являющихся составной частью программного инструментария. Реализуемая в этом подходе идея состоит в выделении наибольшего количества классов и структур данных, повторно используемых в агентных приложениях. В системе MAS DK [19] создана конструкция *типовой агент* (Generic Agent), являющаяся базой для последующей специализации классов и структур данных, клонирования экземпляров агентов и индивидуальной доработки специфических компонентов. Эти процессы поддерживаются соответствующим набором редакторов.

Отметим, что подавляющее большинство из приведенных выше систем ориентированы на разработку МАС "с нуля" и по сути представляют собой платформы для ее создания разработчиком-программистом. Целью же авторов данной статьи является определение принципов и подходов к созданию инструментария, позволяющего не разработчику, а конечному пользователю МСИМ ДБД производить наполнение системы агентами, обладающими требуемой логикой поведения. При этом пользователь не должен отвлекаться на решение сопутствующих общих проблем, связанных с созданием МАС (разработка пользовательских интерфейсов, средств коммуникаций агентов и т. п.), и может не являться программистом высокой квалификации. Исходя из этого, указанные системы целесообразно рассматривать с точки зрения языков и семантик, применяемых для программирования логики агентов.

Так, например, в системе JASON реализована операционная семантика, формально определенная для языка AgentSpeak(L) [20]. Данный язык определяет агента через спецификации множества *базовых атомарных убеждений* и множества *планов*.

Начальное множество убеждений является набором базовых атомарных убеждений, которые являются обычными предикатами первого порядка.

Обобщенная синтаксическая конструкция плана выглядит следующим образом:

Метка
Переключающее событие : Контекст
<- Тело плана .

План инициируется *переключающим событием*. Событие может быть внутренним, когда достигнута определенная подцель, или внешним,

порождаемым в результате модификации убеждений после восприятия состояния внешней среды.

За именем переключающего события следует *контекст*. Контекст должен быть последовательностью таких текущих убеждений агента, для которых данный план применим.

Тело плана есть последовательность *базовых действий*, реализующих предназначение плана, которые агент способен выполнить в среде, а также *целей*, которые агент достигает (или проверяет), когда план выбран для исполнения.

В качестве базовых действий могут выступать:
- математические операции;
- вызовы внутренних функций JASON;
- действия, направленные на объекты среды функционирования МАС;
- коммуникационные акты.

Базовые действия также определяются как предикаты первого порядка.

В языке AgentSpeak(L) определяются два типа целей: достижимые и тестовые. Достижимая цель устанавливается тогда, когда агент стремится достичь определенного *состояния мира*, в котором связанный предикат является истинным. Практически для этого инициируется выполнение вложенных планов. Тестовая цель возвращает результат сравнения связанного предиката с одним из убеждений агента; в случае несовпадения такая цель отвергается.

Программы агентов интерпретируются циклически [20]. В каждом цикле обновляется список событий, порождаемых либо восприятием из внешней среды, либо выполнением подцелей, определенных в теле плана. Далее интерпретатор сверяет выбранное событие с переключающими событиями в заголовках планов, определяя таким образом множество релевантных планов. Проверяя содержимое контекстов этих планов, интерпретатор определяет множество применимых планов и выбирает из него один. Когда все формулы в теле плана оказываются выполненными, цель считается достигнутой и план удаляется из очереди планов. Цикл работы интерпретатора повторяется.

Ниже приведен пример программы на языке AgentSpeak(L) в которой вычисляется факториал числа 5:

```
fact(0,1).
@calc
+fact(X,Y): X < 5
<- +fact(X+1, (X+1)*Y).
@print
+fact(X,Y): X == 5
<- .print("fact 5 == ", Y).
```

Программа содержит два разных плана, которые активизируются при возникновении одного и того же убеждения `fact(X,Y)`, но в разных контекстах: `X < 5` и `X == 5`. Благодаря наличию начального убеждения `fact(0,1).` и первому плану у агента будут последовательно формироваться следующие убеждения:

```
fact(1,1).
fact(2,2).
fact(3,6).
fact(4,24).
fact(5,120).
```

При возникновении последнего убеждения первый план уже не активизируется, зато активизируется второй план, который выведет результат вычисления факториала.

Типы агентов в МСИМ ДБД

Рассмотренный выше язык AgentSpeak(L), безусловно, является мощным средством программирования логики агентов. В отличие от языков программирования общего назначения (C++, Java, Python и т. д.) он имеет высокую степень специализации применительно к рассматриваемой предметной области. Однако достаточная универсальность языка AgentSpeak(L) в рамках этой предметной области делает его довольно сложным в практическом использовании.

Для того чтобы определить требования к инструментарию программирования логики агентов МСИМ ДБД, который был бы удобен для специалистов данной предметной области, попытаемся систематизировать типы применяемых агентов и определить характерные решаемые ими задачи. Рассмотрим наиболее общий случай, когда необходимо моделировать действия каждой отдельно взятой тактической единицы в связке с многоуровневой системой управления (СУ) силами.

В качестве *тактической единицы* будем рассматривать элементарную с точки зрения моделируемой СУ сущность. Такими сущностями могут являться корабль, самолет, рота (взвод, отделение — в зависимости от того, какая требуется степень детализации) солдат. *Такти-*

ческой группой будем называть несколько однотипных тактических единиц, объединенных под единым командованием для решения общей тактической задачи. Тактические группы и тактические единицы могут входить в состав *объединений*, которые представляют собой формирования, создаваемые для решения задач оперативно-тактического, оперативного и оперативно-стратегического уровней. Одни объединения могут входить в состав других.

Таким образом, можно выделить три типа агентов, между которыми выстроена иерархия с точки зрения СУ (рис. 1): тактические единицы, тактические группы, объединения.

Охарактеризуем задачи, которые решают агенты указанных типов в процессе моделирования.

Тактическая единица. Действуя самостоятельно или в составе тактической группы, решает тактические задачи, поставленные перед ней вышестоящим командованием. В качестве командира могут выступать как агенты типов *тактическая группа* и *объединение*, так и человек (оператор).

Роль человека в командовании тактической единицей определяется режимом моделирования. При автоматическом режиме он задает только исходные данные моделирования (сценарии). Если же применяется автоматизированный режим, который допускает интерактивное вмешательство человека в процесс моделирования, то человек, помимо задания исходных данных, может также отдавать приказы на выполнение различных тактических задач.

Рис. 1. Иерархия типов агентов МСИМ ДБД

Тактическая единица должна уметь решать ставящиеся перед ней тактические задачи, действуя автономно (принимая самостоятельно решения в установленных рамках, планируя и осуществляя действия, меняющие окружающую среду), целенаправленно и коммуникабельно (см. Введение). Поскольку автономность подразумевает способность агента взаимодействовать со средой, тактическая единица должна обладать сенсорами, с помощью которых она сможет получать информацию о состоянии среды. Тактическая единица должна обладать способностью восприятия и анализа информации, поставляемой сенсорами, в процессе выполнения тактической задачи. Целенаправленность подразумевает, что информация, полученная агентом от сенсоров или других тактических единиц, используется для анализа текущей ситуации и выбора способа действия для достижения цели, заданной в рамках выполняемой тактической задачи. Коммуникабельность подразумевает способность агента решать свои задачи совместно с другими агентами, например, получая приказы от агента типа *тактическая группа* или *объединение*.

Тактическая группа. Представляет собой объединение нескольких тактических единиц. Исходя из того, что в рассматриваемом случае каждая тактическая единица представлена отдельным агентом, т. е. сущностью, имеющей собственную логику поведения, тактическую группу целесообразно рассматривать в виде агента, формирующего и уточняющего индивидуальные задачи для тактических единиц. При этом агент действует в соответствии с приказами, получаемыми от вышестоящего командования в иерархии СУ агентами, а также использует информацию о среде, получаемую от собственных сенсоров, сенсоров подчиненных тактических единиц либо поставляемую другими агентами.

Объединение. Представляет собой формирование, состоящее из тактических единиц, тактических групп и других объединений. Поэтому, как и в случае тактических групп, объединения целесообразно представлять в виде агентов, формирующих и уточняющих задачи для подчиненных агентов. Логика поведения агента типа *объединение* в некотором смысле дублирует логику поведения агента типа *тактическая груп-*

па — в том и другом случае производится анализ информации, поступающей от подчиненных тактических единиц, средств и СУ, и на основе этого анализа формируются задачи для подчиненных объектов. Различаются лишь масштабы собираемых данных и подчиненных сил.

Таким образом, видно, что агенты всех выделенных типов призваны решать задачи, поставляемые вышестоящим командованием в иерархии СУ агентами. В процессе решения этих задач агенты типов *объединение* и *тактическая группа* осуществляют управление подчиненными агентами, а агенты типа *тактическая единица* изменяют собственное состояние, задавая параметры своего передвижения в пространстве, а также параметры имеющихся в их распоряжении средств обнаружения, поражения и связи.

Программирование логики агентов типа *тактическая единица*

Поскольку назначение агента типа *тактическая единица* состоит в решении тактических задач, поставленных перед ним вышестоящим командованием, будем считать, что логика поведения агента определена в том случае, если заданы алгоритмы решения всех возможных (в рамках конкретного модельного эксперимента) тактических задач. Как уже было отмечено, алгоритм решения тактической задачи для агента данного типа должен определять порядок перемещения тактической единицы в пространстве и управления ее системами в интересах достижения поставленных в задаче целей. При этом необходимо обеспечить анализ информации о текущей обстановке и реакцию на ее изменение в процессе решения задачи.

Применительно к рассматриваемой предметной области упомянутые выше алгоритмы, как правило, строго регламентированы боевыми уставами. Например, алгоритмы движения подводных лодок и использования ими вооружения в процессе выполнения поставленных задач четко описаны и легко представимы в виде блок-схем. То же самое касается алгоритмов поведения подводных лодок в ситуациях, которые могут возникнуть в процессе выполнения поставленных задач при изменении боевой обстановки. Введем термин *поведенческие модели*, объединяющий алгоритмы поведения агента в рамках решения конкретных тактических задач и алгоритмы реакций на изменение обстановки.

Учитывая вышесказанное, для программирования поведенческих моделей удобным представляется инструмент, который бы позволил оператору формировать алгоритмы в виде графической схемы, основными элементами которой являются:

- *процессоры* — реализуют микропрограммы управления тактической единицей (движение с заданными параметрами и т. п.);
- *функции* — обеспечивают расчет входных параметров для процессоров (параметров движения и т. п.);
- *связи* — определяют условия и порядок передачи управления между процессорами и функциями.

Собранная по такому принципу схема может выглядеть, как на рис. 2. На данной схеме используются следующие обозначения элементов: F — функция; H — начало схемы; K — конец схемы.

Для полного описания логики поведения агента типа *тактическая единица*, кроме алгоритмов его поведения в рамках решения конкретных тактических задач и реакций, необходимо задать правила обработки информации об изменении окружающей обстановки, поступающей от сенсоров, и активизации того или иного алгоритма решения задачи или реакции. Назовем эти правила *логикой СУ*. В процессе работы СУ можно выделить следующие этапы:

1. Обработка информации, поступающей от сенсоров, и распознавание событий, имеющих семантическое значение в рамках описания логики действий агента (получение приказа, обнаружение цели и т. п.).
2. Анализ информации о произошедших событиях, текущем состоянии агента и окружающей среды, выбор актуальной поведенческой модели.
3. Активизация актуальной поведенческой модели.

Алгоритмы этапов 1 и 2 могут быть довольно сложными для описания в виде графических схем и требовать для большего удобства таких языковых конструкций, как циклы, ветвления, множественный выбор и т. п. Поэтому в качестве инструмента программирования логики СУ целесообразно рассматривать язык общего назначения (например, Python), расширенный библиотеками проблемно-ориентированной направ-

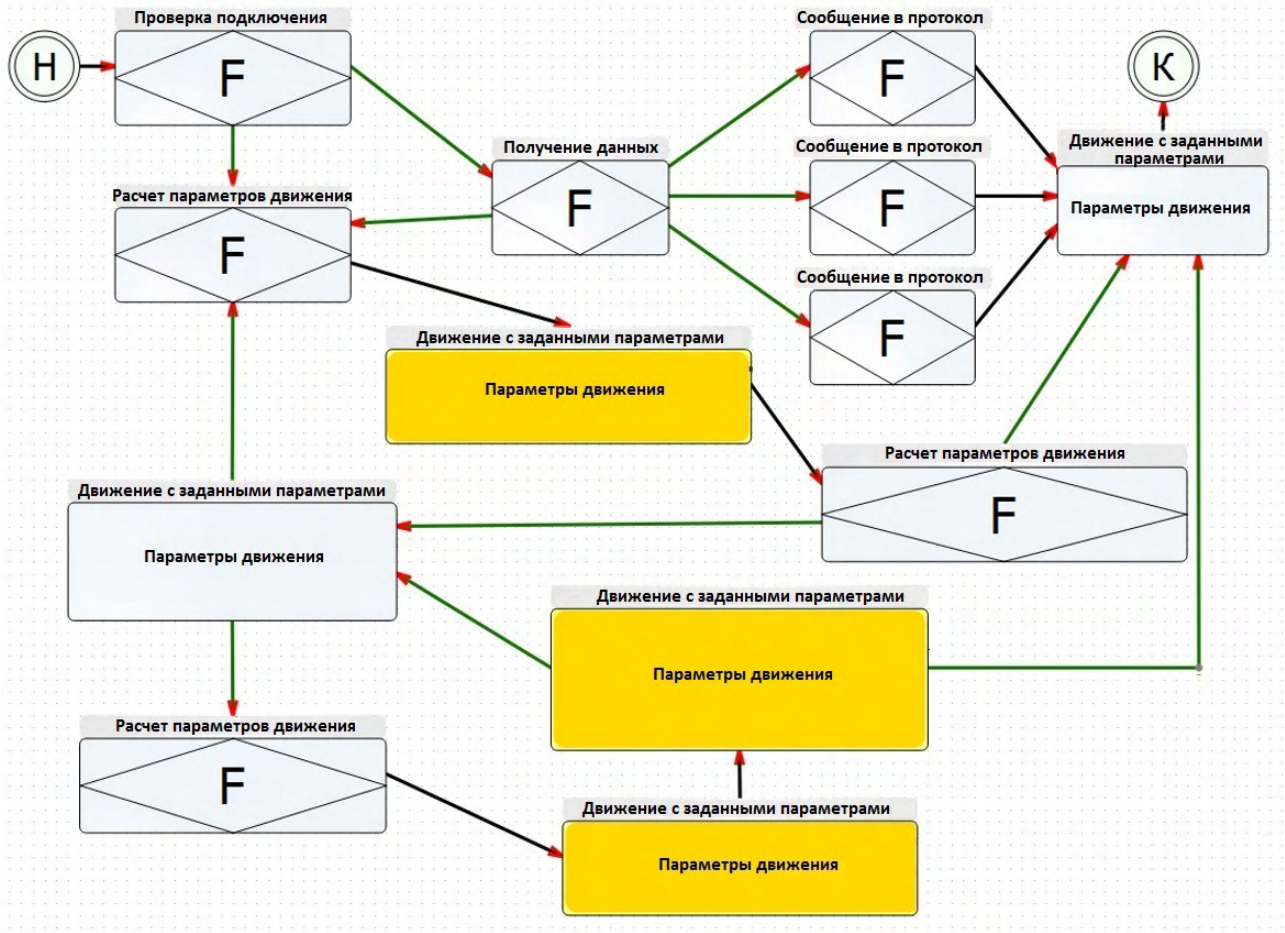

Рис. 2. Пример схемы алгоритма поведения агента типа *тактическая единица*

ленности. В данном случае эти библиотеки могут содержать следующие функции:

- опрос локальных сенсоров;
- отправка сообщения по каналу радиосвязи;
- получение сообщений по каналу радиосвязи;
- получение текущей активной поведенческой модели;
- активизация поведенческой модели;
- расчет расстояния до точки на карте;
- получение модельного времени

и т. п.

Очевидно, что поведенческие модели дополняют логику СУ и вместе с ней определяют полную логику поведения агента типа *тактическая единица*. Тем не менее, принимая во внимание указанные особенности алгоритмизации, авторам представляется целесообразным выделение для программирования поведенческих моделей и логики СУ отдельных интерфейсных модулей, обладающих вышеприведенными свойствами.

Программирование логики агентов типа *тактическая группа*

Введенные понятия поведенческой модели и логики СУ можно применить и к агентам типа *тактическая группа*.

Задачей агентов данного типа является формирование и уточнение индивидуальных задач для подчиненных тактических единиц исходя из получаемых приказов и информации о среде. Учитывая это, в качестве поведенческих моделей агентов типа *тактическая группа* будем рассматривать алгоритмы назначения частных задач подчиненным тактическим единицам и расчета входных параметров для этих задач, а в качестве логики СУ — алгоритмы обработки приказов и информации, получаемой от подчиненных тактических единиц, активизации поведенческих моделей.

При таком подходе выводы о желаемом инструменте программирования логики СУ агентов типа *тактическая группа* могут быть ана-

логичны выводам, сделанным для агентов типа *тактическая единица*. То есть здесь также целесообразно применение языка общего назначения, расширенного библиотеками проблемно-ориентированной направленности. Состав функций этих библиотек должен быть аналогичен приведенному для агентов типа *тактическая единица*, но дополнен функциями, позволяющими обрабатывать информацию о расположении, свойствах и текущем состоянии подчиненных тактических единиц. Эти функции должны обеспечивать получение:

- списка подчиненных тактических единиц;
- информации о расположении подчиненной тактической единицы;
- текущей задачи подчиненной тактической единицы;
- текущего состояния подчиненной тактической единицы

и т. п.

В процессе выполнения поведенческих моделей агентов данного типа в основном производятся различного рода вычисления, подразумевающие оценку текущей обстановки и расчеты значений входных параметров подчиненных тактических единиц. Поэтому для программирования их логики также целесообразно применение языка общего назначения с расширениями, обеспечивающими доступ к геоинформационным данным, информации о тактико-технических характеристиках и текущем состоянии подчиненных тактических единиц.

Программирование логики агентов типа *объединение*

Как уже было отмечено, задачи агента типа *объединение* схожи с задачами агента типа *тактическая группа*. Но если последний решает задачи тактического уровня и при принятии решения оценивает обстановку в рамках нескольких подчиненных тактических единиц, то агент типа *объединение* решает задачи оперативно-тактического, оперативного и оперативно-стратегического уровней и при принятии решений оценивает обстановку в рамках нескольких тактических групп.

С повышением уровня управления возрастает сложность формализации действий агента, осуществляющего управление. Если для тактического уровня алгоритмы управления и действий в большинстве случаев четко определены боевыми уставами, то в оперативном искусстве определения таких алгоритмов отсутствуют. Основной целью оперативных маневров является создание превосходства над противником на главном направлении в операции за счет изменения состава группировок, оперативного построения сил, использования резервов, переноса усилий на другое операционное направление. Таким образом, можно сказать, что критерием достижения основной цели оперативных маневров является обеспечение превосходства над противником на главном направлении, причем этот критерий не меняется от одной задачи к другой в отличие от случая с тактическими маневрами, в которых критерии достижения основной цели определяются индивидуально по каждой задаче. В процессе моделирования агенту типа *объединение* необходимо непрерывно производить анализ всей имеющейся информации о своих силах и силах противника с целью проверки выполнения условия достижения основной цели — превосходства над противником на главном направлении и, если обнаружено, что это условие не выполнено, обеспечить его выполнение вышеперечисленными способами.

Однако основной проблемой при создании инструмента программирования логики поведения агента типа *объединение* остается отсутствие возможности четкой формализации действий при принятии решения и определения детерминированных алгоритмов. В качестве решения данной проблемы может быть рассмотрено применение математического аппарата нечеткой логики и построенных на его основе нейронных сетей (НС).

Задачу управления силами можно разделить на две основные подзадачи:

1) анализ текущего соотношения сил на главном направлении;
2) принятие компенсирующих мер для обеспечения выгодного для себя соотношения сил на главном направлении исходя из имеющихся сил и средств.

Первая подзадача может быть решена посредством простого сравнения сил по выделенным критериям либо с применением более сложной методики оценки боевых потенциалов.

Для решения второй подзадачи целесообразно применение экспертной системы (ЭС), дополненной НС. ЭС позволит выработать решение на основе базы данных решающих правил. Если ЭС столкнется с ситуацией, которую эксперты не

рассмотрели заранее, НС может отнести непредвиденную ситуацию к одной из рассмотренных ранее. В том случае, если этого сделать не удалось, НС, исходя из имеющихся сил и средств, а также доступной информации о противнике, определит наиболее выгодную с точки зрения решаемой оперативной задачи конфигурацию сил, при которой будет обеспечено превосходство над противником на главном направлении. В этом случае будет определено новое правило.

В то время как ЭС используют правила импликации и логический вывод, НС имеют способность к обучению. Эта совокупность качеств делает НС и ЭС достойными претендентами на формирование гибридной интеллектуальной системы [21].

Таким образом, технология программирования логики поведения агента будет сводиться к процессам заполнения базы данных решающих правил и обучения НС.

Для ЭС должны быть определены сущности предметной области (цели, способы решения, возможные варианты событий и др.) и измеряемые характеристики (вероятности наступления событий, коэффициенты значимости целей, приоритетность решений и др.). При описании правил целесообразно использовать следующие базовые понятия: *объект*, *показатель* (признак) и *процедура сравнения*. В данном случае объектами являются агенты типов *объединение*, *тактическая единица*, *тактическая группа*. В качестве показателей могут быть использованы пространственно-временные характеристики, физические свойства объектов (степень поражения объекта, количество боеприпасов и т. д.). Процедура сравнения должна учитывать причинно-следственные связи между объектами, степень влияния одних объектов на другие.

Основными типами алгоритмов обучения НС [22] являются обучение с учителем, обучение с подкреплением, обучение без учителя. В рассматриваемом случае целесообразно применение алгоритмов обучения с подкреплением. Данный подход позволит решить проблемы отсутствия данных о крупных боевых столкновениях, ограниченности доступа к результатам и эффективности учений потенциального противника. Обучение же с учителем принципиально невозможно без размеченных наборов данных, отвечающих требованиям репрезентативности. Анализ литературных источников подтверждает высокий уровень интеллектуальных способностей агентов, обучаемых с подкреплением (системы AlphaGo [23], DeepMind [24], OpenAI Gym [25]). И хотя агенты, обученные с помощью перечисленных программных продуктов, не использовались для детального моделирования ДБД, принципы, на которых строились модели принятия решений, могут быть реализованы и для решения актуальных задач имитационного моделирования с использованием для машинного обучения библиотек с открытым исходным кодом и без лицензионных ограничений.

В ходе обучения с подкреплением программный агент совершает наблюдения и предпринимает действия внутри среды моделирования, возвращающей награды и штрафы. Цель агента — максимизация долгосрочных наград и сокращение штрафов. Это довольно абстрактная установка, конкретная реализация которой может достигаться с использованием таких приемов машинного обучения, как градиенты политики и глубокие Q-сети. Ниже приведен распространенный вариант алгоритма REINFORCE (*REward Increment = non-Negative Factor* $\times$ *Offset Reinforcement* $\times$ *Characteristic Eligibility*) [26]:

1. Запуск моделирования, при котором для политики в форме НС-модели вычисляются (но не применяются) градиенты, определяющие изменение вероятности совершения выбранного действия.
2. После прогона нескольких эпизодов вычисление оценок каждого действия, например, с использованием алгоритма дисконтной ставки.
3. Перемножение вектора градиента и оценки действия для изменения вероятности совершения действия.
4. Применение результирующих векторов-градиентов для изменения весовых коэффициентов НС-модели с использованием алгоритмов оптимизации (в простейшем случае градиентного спуска).

Одной из проблем данного подхода является оценка действий ввиду задержки в получении наград от среды. Совершив ряд действий и достигнув определенного результата, агент должен оценить, какие совокупности действий приводили к достижению результата, а какие мешали (очевидно, что конечный результат определяется не последним из действий, а всей их совокупностью). Одним из распространенных подходов решения проблемы назначения доверительных ко-

эффициентов является использование дисконтной ставки, при которой оценка N совокупности действий определяется по следующей формуле:

$$N = \sum_i n_i r^i, \quad i = 0, \ldots, M-1, \quad 0 \le r \le 1,$$

где M — количество совершенных агентом действий; n_i — награда за отдельное действие; r — значение дисконтной ставки. Чем выше значение дисконтной ставки, тем более значимыми будут предыдущие действия.

Заключение

В результате проведенного анализа подходов к построению инструментальных средств программирования логики поведения агентов в МСИМ ДБД была выполнена систематизация типов агентов и выстроена иерархия с точки зрения СУ. Установлено, что с повышением уровня иерархии возрастает степень абстрактности применяемых алгоритмов. Для каждого типа агентов предложено решение, определяющее наиболее подходящее инструментальное средство для программирования логики его поведения:

- для агентов типа *тактическая единица* — конструктор на основе графических схем;
- для агентов типа *тактическая группа* — язык общего назначения, расширенный библиотеками проблемно-ориентированной направленности;
- для агентов типа *объединение* — заполнение базы данных решающих правил и обучение НС.

Список литературы

1. *Люгер Д. Ф.* Искусственный интеллект: стратегии и методы решения сложных проблем: Пер. с англ. М.: Вильямс, 2003. *Lyuger D. F.* Iskusstvenny intellekt: strategii i metody resheniya slozhnykh problem: Per. s angl. M.: Villyams, 2003.
2. Энциклопедический словарь Ф. А. Брокгауза и И. А. Ефрона. С.-Пб.: Полрадис, 1993. Entsiklopedicheskiy slovar F. A. Brokgauza i I. A. Efrona. S.-Pb.: Polradis, 1993.
3. AgentBuilder: An Integrated Toolkit for Constructing Intelligent Software Agents. Alma, USA: Reticular Systems, 1999. http://www.agentbuilder.com.
4. *Brookings R. S.* Ascape: An Agent Based Modeling Framework in Java. USA, Washington: Brookings Center on Social and Economic Dynamics, 2000. http://www.brook.edu/es/dynamics/models/ascape/.
5. Bee-gent Multi-Agent Framework. Japan, Takamatsu: Toshiba Corporation Systems and Software Research Laboratories, 2000. http://www2.toshiba.co.jp/beegent/index.htm.
6. *Dee C., Millington P., Walls B., Ward T.* CABLE: A multi-agent architecture to support military command and control // Proc. PAAM 2000. Manchester, UK: Practical Application Company, 2000. P. 322—330.
7. *Graham J., Windley V., McHugh D., McGeary F., Cleaver D., Decker K.* Tools for developing and monitoring in distributed multi-agent systems // Workshop on Agents in Industry. Barcelona, Spain. June 2000. http://www.eecis.udel.edu/∼decaf/.
8. *Poslad S. J., Buckle S. J., Hadingham R.* The FIPA-OS agent platform: Open source for open standards // PAAM 2000. Manchester, UK. April 2000. http://fipa-os.sourceforge.net.
9. Grasshopper, Release 2.2. Basics and Concepts (Revision 1.0). Berlin: IKV++ GmbH, 2001. https://www.softwareresearch.net/fileadmin/src/docs/teaching/SS04/ST/BasicsAndConcepts2.2.dpf.
10. *Jazayeri M., Lugmayr W.* Gypsy: A component-based mobile agent system // 8th Euromicro Workshop on Parallel and Distributed Processing (PDP2000). Rhodos, Greece, 2000. http://www.infosys.tuwien.ac.at/Staff/lux/Gypsy.
11. *Bellifemine F., Poggi A., Rimassa G.* JADE — A FIPA-compliant agent framework // PAAM 2000. Manchester, UK. April 2000. http://sharon.cselt.it/projects/jade.
12. *Bordini R. H., Hubner J. F., Wooldridge M.* Programming Multi-Agent Systems in AgentSpeak with JASON. Chichester: Jonh Wiley & Sons, 2007.
13. JATLite. http://java.stanford.edu/java-agent/html.
14. JAFMAS. http://www.ececs.uc.edu/∼abaker/JAFMAS.

15. *Raphael M. J., Deloach S. A.* A knowledge base for knowledge-based multiagent system construction // National Aerospase and Electronics Conf. (NAECON). Dayton, OH. October 10—20, 2000. http://www.cis.ksu.edu/~sdeloach/publications/Conference/arams-naecon.pdf.
16. *Kouadri G., Brezillon P.* A generic framework for context-based distributed authorizations // Proc. 4th Int. Conf. on Modeling and Using Context. Stanford, California (USA), June 23—25, 2003. N.-Y.: Springer-Verlag, 2003. P. 326—333.
17. Reference Guide for Swarm 2.1.1. Swarm Development Group. http://www.santafe.edu/projects/swarm/swarmdocs/set/book930.html.
18. *Collis J., Ndumu D.* The ZEUS Agent Bilding Toolkit: ZEUS Technical Manual. Release 1.0. London: Intelligent Systems Research Group, BT Labs, 1999. http://193.113.209.147/projects/agents/index.htm.
19. *Городецкий В. И., Карсаев О. В., Хотенко И. В., Хабалов А. В.* MAS DK: инструментарий для разработки многоагентных систем и примеры приложений // Тр. Межд. конгресса "Искусственный интеллект в XXI веке" (ICAI 2001). 3—8 сентября 2001 г. М.: Физматлит, 2001. С. 249—262.
Gorodetskiy V. I., Karsev O. V., Khotenko I. V., Khabalov A. V. MAS DK: instrumentariy dlya razrabotki mnogoagentnykh system i primery prilozheniy // Tr. Mezhd. kongressa "Iskusstvenny intellekt v XXI veke" (ICAI 2001). 3—8 sentyabrya 2001 g. M.: Fizmatlit, 2001. S. 249—262.
20. *Bordini R. H., Hubner J. F., Wooldridge M.* Programming Multi-Agent Systems in AgentSpeak Using JASON. New Jersey: Wiley, 2007.
21. *Алдошина А. Н.* Экспертная система на основе нейросетевых технологий для мониторинга и диагностики корпоративной локальной сети // Молодой ученый. 2016. № 18 (122). С. 37.
Aldoshina A. N. Ekspertnaya sistema na osnove neyrosetevykh tekhnologiy dlya monitoringa i diagnostiki korporativnoy lokalnoy seti // Molodoy uchyeny. 2016. № 18 (122). S. 37.
22. *Hastie T., Tibshirani R., Friedman J.* Elements of Statistical Learning. N.-Y.: Springer-Verlag, 2009.
23. *Silver D., Huang A., Maddison C. J., Guez A., Sifre L., Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M., Dieleman S., Grewe D., Nham J., Kalchbrenner N., Sutskever I., Lillicrap T., Leach M., Kavukcuoglu K., Graepel T., Hassabis D.* Mastering the game of Go with deep neural networks and tree search // Nature. 2016. Vol. 529. P. 484—489. doi:10.1038/nature16961.
24. *Higgins I., Pal A., Rusu A., Matthey L., Burgess C., Pritzel A., Botvinick M., Blundell Ch., Lerchner A.* DARLA: Improving zero-shot transfer in reinforcement learning // Proc. 34th Int. Conf. on Machine Learning (PMLR), 2017. Vol. 70. P. 1480—1490. https://arxiv.org/abs/1707.08475.
25. *Brockman G., Cheung V., Pettersson L., Schneider J., Schulman J., Tang J., Zaremba W.* OpenAI Gym. https://arxiv.org/pdf/1606.01540.pdf.
26. *Douglass M. J. J.* Book review: Hands-on machine learning with Scikit-Learn, Keras, and Tensorflow, 2nd edition by Aurelien Geron // Physical and Engineering Sciences in Medicine. 2020. Vol. 43. P. 1135—1136.

Статья поступила в редакцию 11.02.20.

PROBLEMS IN TOOL DEVELOPMENT FOR PROGRAMMING THE AGENT BEHAVIOR LOGIC IN MULTIAGENT SYSTEMS FOR THE SIMULATION OF TWO-SIDE MILITARY OPERATIONS / K. V. Ivanov, M. V. Galkin*, A. I. Sayfullin, R. N. Sayfullina, D. V. Devyatykh (FSUE "RFNC-VNIIEF", Sarov, N.Novgorod region)

The paper considers the key aspects of developing tools for programming the agent behavior logic in multiagent systems and discusses the tool development problems. A brief review of existing agent logic programming tools, including both foreign and domestic software products, is given.

The paper is focused on the simulation of two-side military operations for experts in this subject area. For the simulation purposes, three types of agents are identified: a tactical unit, a tactical group, and a command (large unit). The structure of their communication and interactions and specific features of each agent type are described. For each agent type, a solution is proposed to identify the most appropriate tool for programming the agent behavior logic, which is easy to use by experts in the subject area.

Keywords: simulation, an agent, a multiagent system, programming, behavior logic, management system.

*Galkin Maksim Vladimirovich , scientist,
e-mail: MVGalkin@vniief.ru