# АВТОМАТИЧЕСКОЕ ДООПРЕДЕЛЕНИЕ ИМИТАЦИОННОЙ МОДЕЛИ АГЕНТА

**А. И. Миков, Е. Б. Замятина (Пермь)**

## Введение

Объектом исследования в данной работе являются программные средства, которые позволяют оптимизировать процесс построения имитационной модели агента или совокупности агентов, представляющих собой мультиагентные системы (МАС) – кросс-платформенные распределённые интеллектуальные системы. Под агентами будем понимать автономные программы, которые могут самостоятельно реагировать на внешние события и выбирать соответствующие действия [1, 2, 3].

Известно, что в настоящее время мультиагентные системы получили широкое применение в таких областях, как системы телекоммуникации, поисковые системы в Internet, логистика, компьютерные игры, САПР, системы управления и контроля сложными процессами в медицине и промышленности, программы для электронной коммерции и т. д.[1–4].

Поскольку агенты применяются в самых различных областях, то понятие агента для каждого автора имеет свою смысловую нагрузку. В зависимости от среды обитания агента наделяют конкретным набором свойств. Существует большое количество типов агентов: автономные агенты, мобильные, интеллектуальные, социальные, персональные ассистенты и т. д.

При проектировании мультиагентных систем необходимо подробно исследовать возможности их функционирования в динамически изменяющейся среде, учитывая тот факт, что взаимодействие между агентами также неустойчиво и часто изменяется [3]. Одним из эффективных методов проектирования и исследования мультиагентных систем является имитационное моделирование [1, 2, 4, 5].

Моделирование мультиагентных систем является ресурсоёмким процессом и, следовательно, системы моделирования должны обладать такими программными средствами, которые позволяли бы оптимизировать этот процесс. Ниже предлагается рассмотреть программные средства системы имитации Triad.Net.

Система имитации Triad.Net [8–10] является распределённой версией системы автоматизированного проектирования и моделирования Triad [6,7], которая ранее разрабатывалась сотрудниками кафедры математического обеспечения вычислительных систем Пермского государственного университета и первоначально предназначалась для проектирования и исследования вычислительных систем. Гибкие и эффективные программные средства системы имитации Triad.Net позволяют применить её и для моделирования МАС. Системе имитации Triad.Net свойственны:

- гибкая реконфигурация структуры модели (схема взаимодействий агентов между собой и схема взаимодействия агента – для разных типов агентов – со средой может часто меняться, агент может изменить своё состояние, модифицировав частоту планирования того или иного события, связанного с его функционированием, или изменить среду, в которой он функционирует);
- интероперабельность;
- использование распределённых или параллельных программных средств для моделирования. Для повышения производительности распределённых систем необходимо наличие средств динамической загрузки и перераспределения вычислительных ресурсов;

- масштабируемость (по количеству агентов и по количеству вычислительных узлов, на которых выполняется моделирование);
- наличие подсистемы сбора информации о модели, которая позволяет получать произвольный набор интересующих пользователя характеристик модели, а не строго регламентированный набор, как это практикуется в большинстве систем имитации. Изменять набор интересующих исследователя характеристик можно в ходе моделирования. При этом перекомпиляции кода модели не потребуется.

Кроме того, система Triad.Net даёт возможность автоматически или полуавтоматически доопределить модель. Пользователю достаточно лишь частично описать модель, не указывая алгоритма поведения некоторых её компонентов (автоматическое доопределение). Если пользователь не может сделать предположений о внешних воздействиях среды на агента или о том, как преобразуются сигналы, которыми агент обменивается со средой или другим агентом, то используются языковые средства и программные механизмы полуавтоматического доопределения модели.

Кратко рассмотрим возможности описания имитационной модели в Triad.Net.

**Описание имитационной модели**

Имитационная модель в Triad представляет собой совокупность объектов, которые действуют по определённым сценариям и обмениваются информацией друг с другом.

Имитационная модель $\mu = \{Str, Rout, Mes\}$ представлена тремя слоями: слоем структур (*Str*), слоем рутин (*Rout*), слоем сообщений (*Mes*).

Слой структур предназначен для описания моделируемых объектов и связей между ними, слой рутин представляет собой набор алгоритмов поведения моделируемых объектов, а слой сообщений даёт возможность описывать сообщения сложной структуры. Сообщения простой структуры, которые можно интерпретировать как сигналы, посылаемые одним моделируемым объектом другому, описывают в слое рутин. Моделируемые объекты очень часто имеют иерархическую структуру. Имитационная модель также является иерархической. Каждый из уровней можно описать как граф с полюсами $P = \{U,V,W\}$, где $V$ – множество вершин графа, каждая вершина представляет собой моделируемый объект, который находится на конкретном уровне иерархии. $W$ – это набор дуг, связывающих вершины графа. Они отображают связи между объектами, $U$ – это набор внешних полюсов. Внутренние полюса используют для передачи сообщений на одном уровне иерархии. Их разделяют на входные $In(V)$ и выходные $Out(V)$. Набор внешних полюсов служит для передачи информации объектам, находящимся на более высоком или более низком уровнях иерархии.

Рутина представлена множеством событий ($E$), множеством состояний $Q$, частично упорядоченным множеством моментов времени. Каждое состояние определяется набором значений локальных переменных (множество *Var*) каждой конкретной рутины. Система имитации Triad.Net является параллельной дискретно-событийной системой имитации (PDES – Parallel Discrete Event Simulation), события планируют друг друга. Множество этих событий может быть представлено в виде графа запланированных событий, каждая вершина которого $e_i \in E$. Каждое событие рутины, кроме входного, имеет уникальное имя. Входное событие рутины выполняет обработку сообщений, полученных объектом.

Как уже было отмечено ранее, одной из отличительных особенностей системы моделирования Triad является возможность выполнения операций над моделью: добавление вершины графа $P$, удаление вершины, добавление дуги или ребра, удаление дуги

или ребра и наложения рутины на вершину графа *P*. При этом перетрансляции кода не происходит.

При моделировании мультиагентных систем с помощью системы моделирования Triad.Net целесообразно представлять агенты вершинами графа *P*, а для описания их поведения использовать языковые и программные средства слоя рутин. Поведение агента изменяют, используя операцию наложения рутины на вершину. Если агенты обмениваются сигналами сложной структуры, то их описание и преобразование этих сигналов описывают слоем сообщений. Следует отметить, что каждый слой модели можно исследовать отдельно.

**Автоматическое доопределение агентов**

Автоматическое доопределение имитационной модели агента или совокупности агентов в Triad.Net выполняет подсистема автоматического доопределения модели.

Подсистема автоматического доопределения является экспертным компонентом Triad.Net, который выполняет операции наложения рутин на терминальные вершины имитационной модели по правилам, хранящимся в базе знаний.

Подсистема автоматического доопределения включает следующие компоненты:

- анализатор модели;
- базу данных экземпляров рутин;
- базу знаний экземпляров рутин;
- базу правил;
- механизм вывода;
- преобразователь модели.

Анализатор модели выполняет грамматический разбор частично описанной модели (обозначим её $\mu^*$) и определяет множество терминальных вершин $V^t$, сценарий поведения которых пользователь при описании модели не указал. На анализатор модели возлагается также обязанность извлечь из модели информацию, необходимую для поиска релевантного экземпляра рутины в базе данных рутин. Такой информацией, в частности, является семантический тип вершины.

*Семантический тип* – специальное понятие, которое вводится для того, чтобы сгруппировать несколько объектов по некоторому смысловому, структурному или поведенческому типу. Так для обозначения множества специализированных поисковых агентов, которые настроены на поиск музыкальных файлов, можно задать семантический тип finder_music, для обозначения поискового агента книг–тип finder_book.

Существуют специальные языковые средства языка Triad для определения семантического типа модели (вершины, рутины или другого объекта): <имя_объекта> ==><имя типа>. Например:

```
type finder_music;      (*определение семантического типа *)
                        (*для специализированного поискового агента*)
                        (* музыкальных файлов*)
type finder_book;       (*определение семантического типа *)
                        (*для специализированного поискового агента книг *)
                                ……
node finder_music F1…
                        (*вершина модели получила ссылку на *)
                        (*семантический    тип  finder_music *)
node F2; F2 ==> finder_book;
                        (*вершина модели получила ссылку на *)
                        (*семантический    тип  finder_book *)
```

Преобразователь модели находит в базе данных релевантные экземпляры рутин, опираясь на информацию, полученную от анализатора модели и базы знаний экземпляров рутин, и выполняет операцию наложения рутины на терминальную вершину.

При наложении рутины на вершину преобразователь модели использует соответствующие правила. Например, наложение рутины на вершину может быть выполнено, если выполнены условия специализации, конфигурации и декомпозиции. Условие специализации считается выполненным, если семантические типы доопределяемой вершины и экземпляра рутины из базы данных совпадают. Однако выполнения этого условия недостаточно для адекватного определения поведения агента. Необходимо выполнение условия конфигурации, то есть требуется совпадение количества входов/выходов терминальной вершины $In(V^t)/Out(V^t)$ и рутины $In(G_r)/ Out(G_r)$. Это условие может быть ослаблено, часть входов и выходов вершины может оставаться висячими, а соответствующие сообщения останутся необработанными. Условия декомпозиции определяют правила соединения вершин в представляющем графе $G^t$ для терминальной вершины $V^t$. Граф $G^t$ определяет вершины-предшественницы и вершины-потомки доопределяемой терминальной вершины и должен быть изоморфен графу $G_r$, который хранится в базе знаний экземпляров рутин. Если условие декомпозиции не учитывать, то доопределение модели будет менее точным. Информация в базе знаний рутин пополняется пользователем во время имит ационных экспериментов (оператор ***read***).

Правила, которые используются в подсист еме автоматического доопределения имитационной модели, дают возможность принять соответствующее решение и в том случае, если какое-либо из условий не выполняется (например, семантический тип терминальной вершины может быть опущен, а условия конфигурации и декомпозиции выполняются).

**Выводы**

Задача моделирования многоагентных систем им еет ряд сложных аспектов, одним из которых является трудность полного (детального) описания поведения агента. Эта трудность имеет объективный характер, связанный с природой самих агентов. К тому же получение подробного описания как раз и может быть задачей имитационного моделирования, т.е. моделирование предназначено не только для получения конкретных численных результатов, но и для уточнения самой модели. Налицо определенная рекурсия.

Для решения поставленной задачи необходим м еханизм изменения модели в процессе имитационного моделирования, ее доопределения или переопределения. Такой механизм должен быть реализован в виде интеллектуальной подсистемы системы имитационного моделирования.

Подсистема автоматического доопределен ия частично описанной модели, представленная в работе, позволяет избавить исследователей от рутинной работы, оптимизировать процесс разработки имитационной модели исследуемых объектов, и, в частности, имитационных моделей мультиагентных систем, а также использовать разработанные ранее компоненты. В работе отражены также и другие программные средства системы имитации Triad.Net (гибкая реконфигурация модели, сбор информации о модели по нерегламентированному запросу, распределённый и масштабируемый алгоритм имитации), позволяющие эффективно моделировать не только мультиагентные системы, но и объекты, которые относятся к другим предметным областям [6,7,8,10].

**Литература**

1. **Uhrmacher A. M.** Simulation for Agent-Oriented Software Engineering/www.thesimguy.com/GC/papers/WMC02/G067_UHRMACHER.pdf.
2. **Борщёв А**. От системной динамики и традиционного ИМ – к практическим агентным моделям: причины, технология, инструменты/www.gpss.ru.
3. **Wooldridge M., Jennings N. R.** Intelligent agents: Theory and Practice//Knowledge Engeneering Review. – 1995. – Jun. Vol. 10. – No. 2. – P. 115–152.
4. **Logan B. and Theodoropolous G. A** Distributed Simulation of Multi-Agented Systems// Proceedings of IEEE. –2001. – February. Vol. 89. No 2. – P. 174–185.
5. **Gasser L., Kakugava K.** MACE3J: Fast Flexible Distributed Simulation of Large, Large-Grain Multi-Agent Systems/www.isrl.uiuc.edu/~gasser/papers/mace3j-aamas02-pap.pdf.
6. **Mikov A..I.** Simulation and Design of Hardware and Software with Triad//Proc.2nd Intl.Conf. on Electronic Hardware Description Languages, Las Vegas, USA, 1995. – P. 15–20.
7. **Mikov A. I.** Formal Method for Design of Dynamic Objects and Its Implementation in CAD Systems//Gero J.S. and F.Sudweeks F.(eds), Advances in Formal Design Methods for CAD, Preprints of the IFIP WG 5.2 Workshop on Formal Design Methods for Computer-Aided Design. – Mexico, Mexico, 1995. – P.105–127.
8. **Миков А. И., Замятина Е. Б., Фатыхов А. Х.** Система оперирования распределёнными имитационными моделями сетей телекоммуникаций//Труды Первой Всероссийской научной конференции «Методы и средства обработки информации». – М.: Изд-во МГУ, 2003. – С. 437–443.
9. **Миков А. И., Замятина Е. Б.** Система имитационного моделирования с XML интерфейсом//Сб. трудов международной школы-семинара «Современные проблемы механики и прикладной математики». Воронеж, 2004. – Ч. 1. – Т. 1. С. 244–247.
10. **Миков А. И., Замятина Е. Б.** Масштабируемые программные модели телекоммуникационных систем//Труды Международной научно-технической конференции «Новые методологии проектирования изделий микроэлектроники». Владимир, декабрь 2004.