1-я Российская мультиконференция по проблемам управления. Сборник докладов четвертой научной конференции "Управление и информационные технологии". СПГУ ЭТУ "ЛЭТИ". 2006, с. 138–144.

Групповое управление беспилотными летательными объектами на основе автоматного программирования

Ф.Н. Царев, А.А. Шалыто

Санкт-Петербургский государственный
университет информационных технологий, механики и оптики
Кафедра «Компьютерные технологии»
Россия, Санкт-Петербург, Кронверкский пр., 49

1. Цель настоящей работы

Рефлексная технология [1] исследовалась психологами [2], которые использовали отображение "вход-выход". Такому отображению соответствует модель автомата без памяти – комбинационная схема.

В результате прогресса в психологии возникло понимание того, что необходимо учитывать внутренние состояния системы [3]. В большинстве работ по искусственному интеллекту рефлексная модель с внутренними состояниями (реактивные агенты) рассматривалась как слишком простая для того, чтобы на ее основе можно было добиться значительных успехов, но исследования, описанные в работах [4, 5], "позволили поставить под сомнение это предположение" [1].

Использование такого подхода позволяет избежать создания сложной модели внешней среды – вместо нее предлагается использовать реактивную систему (систему, которая реагирует на внешние события), поведение которой описывается системой взаимодействующих конечных автоматов.

Важно отметить, что в *NASA* используется аналогичный подход [6] при создании программного обеспечения, входящего в проект *Mars Science Laboratory*.

В 1999 году было предложено автоматное программирование [7] и выполнены исследования по его применению при создании:

- агентов, осуществляющих логическое управление [7, 8];
- реактивных агентов [9];
- объектно-ориентированных реактивных агентов [10].

Для отработки технологии автоматного программирования в рамках *Движения за открытую проектную документацию* [11] выполнен ряд студенческих проектов. Примеры построения реактивных систем со сложным поведением описаны в работах [12-15], а примеры построения объектно-ориентированных реактивных агентов – в работах [16, 17]. В ходе выполнения этих проектов программирование агентов выполнялось вручную – без использования инструментальных средств.

Цель настоящей работы – описание технологии моделирования управления движением одного класса беспилотных летательных объектов на основе автоматного программирования. Эта технология базируется на применении инструментального средства *UniMod* [17, 18], предназначенного для поддержки технологии автоматного программирования, называемой также *SWITCH*-технологией [7].

2. Автоматное программирование

При автоматном программировании программы предлагается строить как автоматизированные системы, которые состоят из системы управления (система взаимодействующих конечных автоматов), объектов управления и контуров обратной связи между объектами и системой. При этом автоматы по входным воздействиям (события и входные переменные) выполняют переходы между состояниями, которые явно выделяются, и формируют выходные воздействия, соответствующие методам, реализуемым в объектах управления. Такой взгляд на программирование является естественным для решения задач управления разных уровней.

1-я Российская мультиконференция по проблемам управления. Сборник докладов четвертой научной конференции "Управление и информационные технологии". СПГУ ЭТУ "ЛЭТИ". 2006, с. 138–144.

В рамках автоматного программирования предлагается использовать только два типа диаграмм: схема связей автоматов с поставщиками событий и объектами управления (рис. 1), определяющая интерфейс автоматов, а также граф переходов (диаграмма состояний, рис. 2), который описывает поведение автомата.

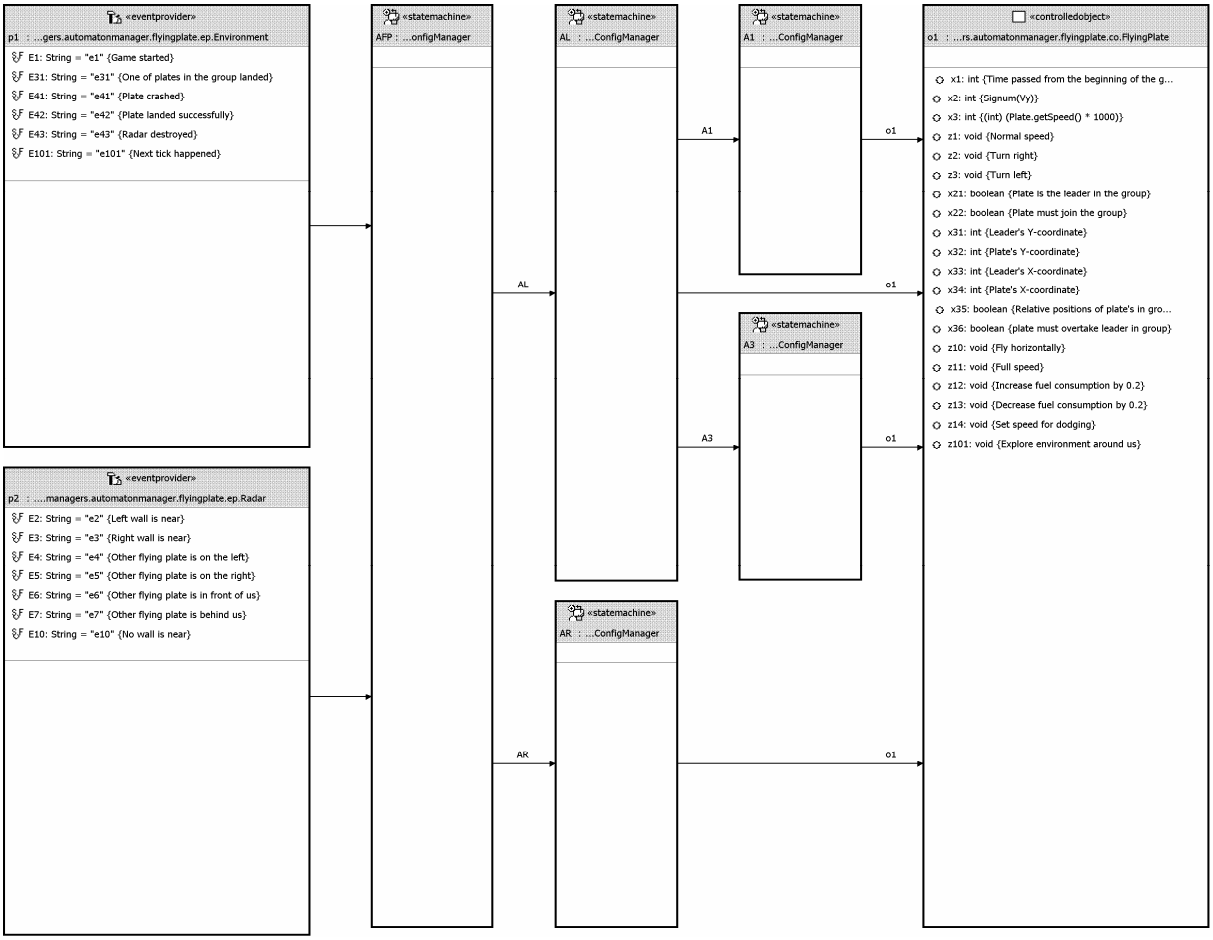

Рис. 1. Схема связей

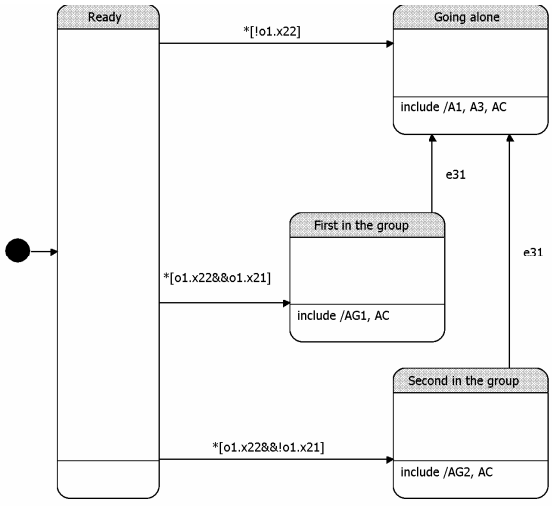

Рис. 2. Диаграмма переходов автомата *Режим полета*

1-я Российская мультиконференция по проблемам управления. Сборник докладов четвертой научной конференции "Управление и информационные технологии". СПГУ ЭТУ "ЛЭТИ". 2006, с. 138–144.

3. Инструментальное средство *UniMod*

Открытое инструментальное средство *UniMod* [18, 19] базируется на трех открытых компонентах: унифицированный язык моделирования *UML*, *SWITCH*-технология и среда разработки *Eclipse* [20]. Это средство является встраиваемым модулем (плагином) для указанной среды разработки.

UniMod позволяет строить и редактировать схемы связей и диаграммы состояний, обеспечивать проверку формальной корректности этих диаграмм, проводить отладку диаграмм в графическом режиме и т. д.

После построения диаграмм и автоматической проверки их корректности, по ним строится их *XML*-описание. Далее вручную пишутся следующие фрагменты программы на языке *Java*: для поставщиков событий – их объявления, инициализация и преобразование системных событий в автоматные, а для объектов управления – методы, реализующие входные переменные и выходные воздействия.

Для реализации программ могут использоваться интерпретационный и компиляционный подходы. В первом случае *XML*-описание интерпретируется, вызывая методы скомпилированных *Java*-классов, написанных вручную. Во втором – по *XML*-описанию с помощью шаблонов *Velocity* [21] создается код программы на языке *Java*, реализующий систему автоматов. Этот код компилируется совместно с фрагментами программы, написанными вручную. В обоих случаях в процессе работы программы ведется протокол в терминах автоматов.

4. Описание беспилотных летательных объектов

Беспилотные объекты в этой задаче называются летающими тарелками [22], каждая из которых представляет собой дискообразное "летающее крыло" заданного радиуса. Она оснащена реактивным двигателем с топливным баком заданной емкости, аэродинамическими рулями, позволяющими поворачивать тарелку относительно ее текущего направления движения на угол, не превышающий 25°.

На летательном объекте размещен компьютер, который управляет расходом топлива (тягой двигателя) и положением аэродинамических рулей для обеспечения поворотов. Летающая тарелка может передвигаться со скоростями не ниже заданного предела. Тарелка, скорость которой упала ниже предела, не может держаться в воздухе. Если при этом ее топливный бак пуст, то она нормально завершает движение, если же в топливном баке осталось топливо, то тарелка завершает движение аварийно.

Тарелка движется в соответствии со вторым законом Ньютона. Ее движение определяется двумя силами: сопротивлением воздуха $\boldsymbol{F}$ и тягой двигателя $\boldsymbol{T}$. Сопротивление воздуха определяется по формуле $F = c_1 + c_2 v^2$, где $\boldsymbol{v}$ – скорость тарелки, а коэффициенты $\boldsymbol{c_1}$ и $\boldsymbol{c_2}$ определяются ее аэродинамическими характеристиками. Тяга двигателя определяется по формуле $T = c_4 q$, где q – расход топлива. Расход топлива контролируется компьютером и может изменяться в заданных пределах. Константа $\boldsymbol{c_4}$ определяется характеристиками двигателя.

Тарелки группируются в две команды по N. Команды «сражаются» между собой. В дальнейшем будем называть каждое такое сражение «игрой». В начале игры летательные объекты первой команды располагаются случайным образом в 25 метрах от линии старта в левой половине коридора, а второй команды – симметрично им в правой половине. Ширина коридора задана, его длина бесконечна. Для каждого объекта заданы начальная скорость и направление движения. В простейшем случае все скорости одинаковы, а направления – строго вперед. После команды «Старт» все объекты движутся по коридору с целью максимально удалиться от линии старта.

Объекты могут сталкиваться друг с другом, так как имеют возможность изменять расход топлива и поворачивать. Столкновения при этом предполагаются абсолютно упругими. Если относительная скорость столкновения превышает заданную величину, то оба объекта счита-

ются выбывшими из игры. Объекты также выбывают из игры при выходе за правую или левую границу коридора.

Между объектами существует аэродинамическое взаимодействие. В некоторых областях сопротивление воздуха увеличивается на 50%, в то время как в других – уменьшается на 50%. Разумеется, в первом случае объектам требуется больше топлива, чем во втором. Находясь в зоне повышенного сопротивления воздуха, объекты стремятся ее покинуть.

Объект может продолжать игру или завершить ее. Игра продолжается до тех пор, пока ее не завершил хотя бы один объект. После того, как ее закончит и этот объект, игра завершается. При подведении итогов игры учитываются только объекты, которые нормально ее завершили. Результатом команды является максимальное из расстояний, на которое удалился от линии старта один из ее объектов, нормально завершивший игру (скорость меньше заданного предела при пустом топливном баке).

5. Групповое управление движением

В ходе настоящей работы было сделано предположение, что победной будет стратегия, при которой в команде одна половина объектов движется вперед и уклоняется от границ коридора и рядом находящихся объектов, а вторая половина объектов разбивается на пары. При этом объекты в парах движутся, поддерживая взаимное расположение для использования аэродинамического взаимодействия. Эта стратегия почти всегда оказывается победной по сравнению со стратегией, реализованной для противника: один объект летит вперед, а другие – пытаются сбить летательные объекты другой команды.

При моделировании поставщики событий используются для реализации «органов чувств» летательных объектов, конечные автоматы – для описания поведения этих объектов, а объекты управления – для влияния на окружающий мир.

В данной версии программы поведение всех объектов «нашей» команды описывается одной и той же системой взаимодействующих автоматов. Например, автомат *Состояние объекта* имеет три состояния: «Летит», «Нормальное завершение», «Аварийное завершение». В первое состояние этого автомата вложены автомат *Режим полета* (рис. 2) и автомат *Радар*. В состояние «Полет в одиночку» автомата *Режим полета* вложены автомат *Уклонение от границ коридора и объектов справа и слева* и автомат *Уклонение от объектов спереди и сзади*. В состояние «Полет первым в паре» автомата *Режим полета* вложен автомат *Полет первым в паре*, а в состояние «Полет вторым в паре» автомата *Режим полета* вложен автомат *Полет вторым в паре*. Также в каждое из указанных выше трех состояний автомата *Режим полета* вложен автомат *Прием и обработка сообщений от других объектов*.

6. Моделирование

Моделирование системы выполнено на основе инструментального средства *UniMod* по технологии, описанной в разд. 3. При этом разработка программ выполняется не традиционным путем, а в значительной мере с помощью их проектирования на основе моделей. Таким образом, можно утверждать, что проектирование выполняется в рамках *MDA* (*Model Driven Architecture*) [23], что позволило выполнить моделирование платформенно-независимо.

7. Почему целесообразно использовать автоматное программирование?

Как отмечалось в разд. 3, UML-диаграммы, создаваемые программистом, при использовании компиляционного подхода преобразуются в текст программы на языке *Java*. На рис. 3 показано соотношение объемов исходного кода, написанного вручную и автоматически сгенерированного по *UniMod*-моделям.

Рис. 3. Соотношение объемов исходного кода, сгенерированного автоматически и написанного вручную

Таким образом, можно заключить, что основная функциональность системы управления реализована на основе автоматов. Размер текста программы, сгенерированного автоматически, более чем в четыре раза превосходит размер текста, написанного вручную. Поэтому существенно повышается надежность программы, особенно в связи с тем, что при построении диаграмм автоматически проверяется их корректность, а корректность функций, вызываемых из автоматов можно установить с помощью доказательств на основе пред- и постусловий.

8. Заключение

Обычно считается, что научная новизна работы в этой области должна состоять в изобретении некоторой «хорошей» стратегии. Авторы считают, что новизна настоящей работы состоит в формализации процесса программной реализации выбранной стратегии. При этом стратегии, как соперника, так и «нашей» команды могут быть изменены, а качество программного обеспечения обычно не снижается.

Простая автоматная модель, описываемая в настоящей статье, позволяет обеспечить достаточно сложное поведение системы, а инструментальное средство *UniMod* его эффективно реализовать. Обратим внимание, что при построении диаграмм выполняется проверка корректности их построения. При участии авторов проводятся работы по верификации автоматных программ на основе метода *Model checking* [24].

Описанный проект и его документация опубликованы на сайте http://is.ifmo.ru в разделе *UniMod-проекты*.

Источники

1. Рассел С., Норвиг П. Искусственный интеллект. Современный подход. М.: Вильямс. 2006.
2. Skinner B.F. *Science and Human behavior.* London: Macmillan. 1953.
3. Putnam H. Mind and machines /*Dimensions of Mind.* London: Macmillan. 1960, pp.138–164.
4. Rosenschein S.J. Formal Theories of Knowledge in AI and Robostics *New Generation Computing.* 1985. 3(4), pp.345–357.
5. Brooks R.A. A Robust Layered Control System for a Mobil Robot *IEEE Journal of Robotics and Automation.* 1986. 2, pp.14–23.
6. Regan P., Hamilton S., NASA's Mission Reliable. *Computer*. 2004. January, pp. 59–68.
7. Шалыто А.А. Технология автоматного программирования /Труды первой Всероссийской научной конференции "Методы и средства обработки информации" М.: МГУ. 2003, с.528–535 http://is.ifmo.ru/works/tech_aut_prog
8. Шалыто А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. СПб.: Наука, 1998.

9. Наумов Л.А., Шалыто А.А. Методы объектно-ориентированной реализации реактивных агентов на основе конечных автоматов // Искусственный интеллект. 2004. №4, с. 756–762. http://is.ifmo.ru/works/_aut_oop.pdf
10. Туккель Н.И., Шалыто А.А. SWITCH-технология — автоматный подход к созданию программного обеспечения «реактивных» систем // Программирование. 2001. № 5, с. 45–62. http://is.ifmo.ru/works/switch/1/
11. Шалыто А. А. Новая инициатива в программировании. Движение за открытую проектную документацию // Мир ПК. 2003. №9, с. 52–56. http://is.ifmo.ru/works/open_doc/
12. Туккель Н.И., Шалыто А.А. Система управления дизель-генератором (фрагмент). Программирование с явным выделением состояний. Программная документация http://is.ifmo.ru/projects/dg/
13. Шалыто А. А., Туккель Н. И. Танки и автоматы //BYTE/Россия. 2003. № 2, с. 69–73. http://is.ifmo.ru/works/tanks_new/
14. Шалыто А. А., Кузнецов Д. В. Система управления танком для игры «Robocode». http://is.ifmo.ru/works/tanks_new
15. Ярцев Б.М., Шалыто А.А. Разработка программного обеспечения роботов *Lego Mindstorms* на основе автоматного подхода (Проект *Isenguard*) http://is.ifmo.ru/projects/lego/
16. Naumov L., Korneev G., Shalyto A. Methods of Object-Oriented Reactive Agents Implementation on the Basis of Finite Automata. *Proceedings of 2005 International Conference Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering.* KIMAS-05. Boston: IEEE Boston Section. 2005, pp. 460–465. http://is.ifmo.ru/articles_en/_kimas05-1.pdf
17. Yartsev B., Korneev G., Shalyto A., Kotov V. Automata-Based Programming of the Reactive Multi-Agent Control Systems. *Proceedings of 2005 International Conference Integration of Knowledge Intensive Multi-Agent Systems: Modeling, Exploration and Engineering.* KIMAS-05. Boston: IEEE Boston Section. 2005, pp. 449–453. http://is.ifmo.ru/articles_en/_kimas05-2.pdf
18. Гуров В., Мазин М., Нарвский А., Шалыто А. UML. SWITCH-технология. Eclipse //Информационно-управляющие системы. 2004, № 6, с.12–17. http://is.ifmo.ru/works/uml-switch-eclipse/
19. *UniMod* project. http://unimod.sourceforge.net
20. *Eclipse* project. http://eclipse.org
21. *Velocity* project. http://jakarta.apache.org/velocity/
22. VI Открытая Всесибирская олимпиада по программированию им. И.В.Поттосина http://olimpic.nsu.ru/widesiberia/archive/wso6/2005/rus/index.shtml
23. Model Driven Architecture. http://www.omg.org/mda/
24. Clarke E. M., Grumberg O., Peled D. A., Model checking. The MIT Press. 2000.