Методы проектирования пространственно распределенной многоагентной системы управления в рамках концепции интернета вещей

А. Е. Коньков (МГУ им. М. В. Ломоносова, Москва)

В предлагаемой работе исследуется проблема синтеза пространственно распределенных систем управления в контексте существующего уровня развития систем связи и вычислительных устройств, обосновывается преимущество агентного подхода к систезу пространственно распределенных систем управления. Отдельное внимание уделяется некоторым аспектам онтологического обеспечения и проектирования распределенных многоагентнных систем управления.

Ключевые слова: система управления, агентный подход, интернет вещей.

1. Введение

Глобальная экономика входит в эпоху Интернета вещей и массового межмашинного взаимодействия.

Интернет вещей (IoT, Internet of Things) становится следующим скачком развития, сравнимым с изобретением парового двигателя или индустриализацией электричества.

Ожидается, что в ближайшие годы количество машинных подключений будет увеличиваться на 25% в год, а всего к 2021 году на планете будет 28 миллиардов подключенных устройств. Из них всего 13 миллиардов придется на привычные пользовательские гаджеты: смартфоны, планшеты, лэптопы и ПК — в то время как 15 миллиардов будут составлять пользовательские и промышленные устройства: разного рода датчики, терминалы для продаж, автомобили, табло, индикаторы и т. д.

Возникает проблема выбора подхода к синтезу пространственно распределенной системы управления такими устройствами.

2. Агентный подход

В задачах автоматизации сложных, многообъектных и пространственно распределенных систем управления всё большее применение находят Многоагентные системы управления (МАС), относящихся к классу интеллектуальных систем. В основе МАС лежит взаимодействие Интеллектуальных Агентов (ИА) — сущностей, получающих информацию через систему сенсоров о состоянии управляемых ими процессов и влияющих на них через систему актуаторов, при этом их реакция рациональна в том смысле, что их действия содействуют достижению определенной цели.

Стоит отметить важные для IoT свойства МАС:

- Автономность: агенты, хотя бы частично, независимы;
- Ограниченность представления: ни у одного из агентов нет представления о всей системе;
- Децентрализация: нет агентов, управляющих всей системой.

Интернет вещей представляется идеальной средой для размещения МАС. Существующие объектно-ориентированные языки программирования, компиляторы и интерпретаторы программ в машинный код, программаторы чипов, стандартизированные протоколы передачи данных, развитые линии проводной и беспроводной связи полностью удовлетворяют любой возможной архитектонике МАС.

Агентный подход в самом ближайшем будущем станет основным методом синтеза систем управления, основанных на устройствах интернета вещей, а создание общей теории МАС является актуальной задачей.

3. Самоорганизация

МАС представляет из себя систему, состоящую из интеллектуальных агентов (элементов системы) и связей между ними.

Определение. Под системой понимается объект, обладающий следующими свойствами:

- он является совокупностью нескольких (не менее двух) других элементов и связей (между элементами) системы;
- (свойство внутренней вложенности) элементам системы может быть совокупностью нескольких (не менее двух) других элементов системы и нескольких связей системы; такой элемент системы

называется сложный; элемент системы, не являющимся сложный, называется атомарным (элементом нулевого уровня иерархии системы), к таким элементам не может применяться декомпозиция; несколько атомарных элементов, имеющих некоторый общий признак, образуют элемент первого уровня иерархии системы; несколько элементов первого уровня иерархии, имеющих некоторый общий признак, образуют элемент второго уровня иерархии системы и так далее;

- (свойство внутренней необоснованности) и сама система, и каждый ее сложный элемент обязательно имеет хотя бы одно новое свойство, которого не было ни у одного из составляющих элементов; такое свойство называется внутренне необоснованным;
- (свойство внешнего обоснования) каждое внутренне необоснованное свойство системы [сложного элемента] обязательно основано на наличии некоторых связей системы [сложного элемента] хотя бы с одним внешним по отношению к системе [к сложному элементу] внешнему воздействию; такой внешнее воздействие можно назвать обосновывающим данное свойство системы [сложного элемента] и устраивающим данную систему [сложный элемент].

Из перечисленных выше свойств следует, что каждый сложный элемент системы сам является системой, называемой подсистемой системы или уровнем иерархии. Отметим, что внешние воздействия, устраивающие сложный элемент системы, могут сами быть элементами системы.

Такое определение системы дает возможность сделать следующее утверждение о принципе самоорганизации.

Утверждение 1. *Система организуется (устраивается) обязательно с участием внешних устраивающих воздействий; никакого чисто внутреннего самоустроения (автоэмергентности) системы из какой-либо совокупности элементов, которые могут стать порождающими элементами этой системы, не существует.*

Иное понимание принципа самоорганизации приведет к противоречию, например к парадоксу Рассела, когда система (сложный элемент) устраивает сама себя (автоэмергентна).

Устраивающей всю МАС внешнюю систему будем называть общей онтологией МАС. Ни МАС, ни один ее элемент, не могут устраивающим воздействием на общую онтологию МАС, таким воздействием могут являться только начальные условия, которые задаются разработчиком системы.

Введем обозначения:
A_i^j — i агент, j-го уровня (иерархии);
A_i^0 — i агент нулевого уровня.
O_i^j — онтология A_i^j.
S_i^j — внешнее воздействие на A_i^j.
D_i^j — действие A_i^j.

Отображение $\alpha : O_i^j \to A_i^j$ называется инициализацией (созданием) A_i^j. Отображение $\gamma : S_i^{j+1} \to O_i^j$, где S_i^{j+1} — внешнее воздействий на A_{i+1}, называется устраивающим воздействием A_i^{j+1} на A_i^j.

Рассмотрим самоорганизацию множества A_i^j агентов, где $i = 1, 2, \ldots, n$, $A_k^{j+1} \supseteq A_i^j$, $k > 0$. В общем случае онтологию элементарного агента удобно формализовать в виде следующей модели:
Пусть $R_i^{j+1} \subset O_i^{j+1}$ — некоторый организующий параметр A_i^{j+1}, например ресурсное или временное ограничение.

Формализация A_i^j:
$S_i^j(t+1) = \lambda(S_i^j(t), D_i^j(t))$;
$D_i^j(t) = \beta(S_i^j(t)), \quad \lambda : S \times D \to S$, $\delta : S \to D$.
$f : O_i^j \to F_i^j$ — ресурс; $g : D_i^j \to G_i^j$ — затраты;

Таким образом задачу самоорганизации A_i^j агентов под утверждающим воздействием A_k^{j+1} агента, где $i = 1, 2, \ldots, n$, $j > 0$, $k > 0$ можно свести к задаче оптимального распределения: найти минимум (максимум) суммы:
$\sum\limits_{i=1}^{n} g_i(x_i)$ при ограничениях $\sum\limits_{i=1}^{n} f_i(x_i) \leqslant R, x_i \in X_i, i = 1, 2, \ldots, n$
где X_i — конечные множества; $f_i(x_i) \geqslant 0$; $g_i(x_i) \geqslant 0$ и $R > 0$.
В таком виде могут быть записаны различные задачи, сводящиеся к распределению заданного ресурса R между n потребителями.

В задаче на максимум функции $f_i(x_i)$ характеризуют ресурс, а $g_i(x_i)$ — эффективность его использования. При $f_i(x_i) = p_i x_i$ и $g_i(x_i) = c_i x_i$ получается задача оптимальной загрузки транспортного средства предметами, веса которых равны p_i, стоимости — c_i, а количества — $x_i = 0, 1, 2, \ldots$; если при этом $x_i \in \{0, 1\}$, то получается известная задача о ранце.

В задаче на минимум функции $f_i(x_i)$ — ресурс, а $g_i(x_i)$ — затраты. Например, $f_i(x_i)$ — ресурс времени, а $g_i(x_i)$ — затраты на осуществление некоторого проекта.

Методы решения подобной задачи с использованием алгоритма Немхаузера–Ульмана описаны в [3].

4. Прототипизация агентов

Другой важной задачей является синтез агентов, находящихся на одном уровне иерархии. Задачу синтеза большого числа таких агентов с однотипным поведением упрощает использование прототипно-ориентированных языков программирования, например JavaScript.

В JavaScript есть встроенное наследование между объектами при помощи специального свойства __proto__. Объекты в JavaScript можно организовать в цепочки так, чтобы свойство, не найденное в одном объекте, искалось бы в другом. Пусть $agentgroup1 := A_1^{j+1}$, $agent3 := A_3^j$, $A_3^j \in A_1^{j+1}$.

Пример скрипта:

```
var agentgroup1 = {
property1: true
};
var agent3 = {
property2: false,
property3: false
};
agent3.__proto__ = agentgroup1;
alert( agent3.property1 ); // false, наследовано из agentgroup1
```

При установке свойства agent3.__proto__ = agentgroup1 говорят, что объект agentgroup1 будет «прототипом» agent3.

Свойство __proto__ удобно использовать и для изменения множества действий агента при самоорганизации.

Список литературы

[1] Коньков А. Е. Метод оптимизации взаимодействия многоагентной системы управления с реляционной базой данных // Материалы Международного молодежного научного форума «ЛОМОНОСОВ-2016».

[2] Захаров В. К. Номология. Воспроизведение и обновление человеческого бытия. — М.: Onebook.ru, 2016.

[3] Струченков В. И. Новые алгоритмы оптимального распределения ресурса // ПДМ. — 2010. — № 4 (10).

[4] Современный учебник JavaScript. ООП в прототипном стиле. Прототип объекта. — [Эл. ресурс] URL: https://learn.javascript.ru/prototype