УДК 004.05

# Методы синхронизации агентов при верификации систем в терминах мультиагентных сред

А. А. Губа
*Киевский национальный университет имени Т. Шевченко, Украина*

В статье рассматриваются методы синхронизации поведений агентов при верификации систем в терминах мультиагентных сред. Описаны сценарный и событийный подходы при декомпозиции MSC-сценариев и диаграмм последовательности событий UML opt-областей на базовые протоколы. Предложен, более оптимальный, алгоритм для генерации SDL.

***Ключевые слова:*** *верификация, декомпозиция MSC-сценариев, генерация SDL.*

У статті розглядаються методи синхронізації поведінок агентів при верифікації систем у термінах мультиагентних середовищ. Описані сценарний і подійний підходи при декомпозиції MSC-сценаріїв і діаграм послідовності подій UML opt-областей на базові протоколи. Запропоновано більш оптимальний алгоритм для генерації SDL.

***Ключові слова:*** *верифікація, декомпозиція MSC-сценаріїв, генерація SDL..*

In the paper we consider the methods for synchronization behavior of the agents in verification of systems in terms of multi-agents environment. This paper also describes scenario and event approaches for decomposition MSC-scenarios and diagrams of the events sequences UML opt-regions on the basic protocols. More optimal algorithm to generate the SDL is proposed.

***Key words:*** *verification, decomposition of MSC-scenarios, SDL generation.*

## 1. Введение

Методы тестирования не обеспечивают исчерпывающего анализа всех возможных вариантов поведений систем, тем самым, не могут обнаружить все случаи нарушений свойств, какими должны обладать системы, которые разрабатываются. Для систем со сложной моделью поведения становится невозможным обойтись без автоматизации проверки правильности. Под системой понимается несколько компонент (объектов, агентов), которые взаимодействуют между собой и с окружающей их средой (среда также может задаваться в виде отдельного агента).

Часто требования к системам подаются в виде сценариев. Это могут быть либо MSC-сценарии, либо диаграммы последовательности событий UML (Sequence Diagrams). Для проведения верификации заданных требований в системе VRS [2] необходимо осуществить их формализацию или построить систему базовых протоколов, которые используются как входные данные для трассового генератора (одна из компонент VRS). С помощью трассового генератора мы получаем трассы или последовательности базовых протоколов, отражающих поведение системы. В результате мы получим либо трассы с удовлетворительным вердиктом о том, что покрыты все протоколы и требования, либо трассы, которые приводят к тупиковым состояниям системы и вердикт о непокрытых требованиях. В любом случае данные трассы должны быть согласованы со сценарием. Данное согласование состоит в том, что

последовательности событий (цепочки событий) на диаграмме должны совпадать с последовательностями событий на трассах. При проектировании любой сложной системы архитекторы используют встроенные выражения: ссылки, циклы, операторы альтернатив и другие. На данный момент методов, которые бы верифицировали сценарии, нет, поэтому в статье рассматривается одна из задач, возникающих при декомпозиции сценариев на базовые протоколы: декомпозиция встроенных выражений (inline expressions). В качестве таковых мы используем MSC операторы alt, opt. Если MSC оператор opt с условием, проблемы с синхронизацией не возникает, так как само условие уже есть синхронизацией. Безусловные opt-операторы – пример недетерминизма в динамических системах. В этом случае, при отсутствии синхронизации агентов, которые принимают участие в этой области при декомпозиции сценария, как было реализовано в предыдущих работах [4], получаем абсолютно независимые участки трассы, что может противоречить исходной диаграмме. Это ведет к некорректности всей трассы, а также к ложным тупиковым состояниям.

**2. Язык базовых протоколов**

В качестве языка формального описания моделей рассматривается язык базовых протоколов (БП). Базовый протокол – выражение вида $\forall x\left(\alpha \to \langle u \rangle \beta\right)$ [1,6], где x – список (типизированных) параметров, α і β – формулы базового логического языка, u – процесс протокола (конечное поведение композиции нескольких агентов и сред MSC диаграмм). Формула α называется предусловием, а формула β – постусловием базового протокола. Базовый протокол может рассматриваться как формула темпоральной логики, которая выражает тот факт, если состояние системы удовлетворяет условие α, то процесс u может быть инициированный, и после его завершения состояние системы будет удовлетворять условие β.

В качестве базового языка используется язык многосортного исчисления предикатов первого порядка. Формула базового языка может содержать переменные и константы следующих простых типов: числовые (целый (int) и действительный (real)) и символьные (тип Bool, перечислимый, произвольный символьный (symb)). Допустимы массивы элементов простых типов с целыми и перечислимыми индексами, функциональные типы (функции от аргументов простых типов, которые возвращают значения простого типа), списки элементов простых типов. В качестве переменных, которые меняют свое значение в процессе функционирования системы, выступают атрибуты и атрибутные выражения. Атрибутными выражениями являются операторы доступа к элементу массива (array) по индексам ($\arg(mas, i)$), функции доступа к спискам (lists) ($get\_from\_head(l)$, $get\_from\_tail(l)$, $empty(l)$), атрибутные выражения функциональных типов ($a(x, y, z)$).

Предусловие базового протокола содержит формулу базового языка, в постусловии используются присваивания, операторы обновления списков, а также формула базового языка. Левыми частями присваивания могут выступать атрибуты простых типов и атрибутные выражения, кроме функций доступа к

спискам. Присваивания и операторы обновления списков рассматриваются как равенства, которые связывают старые и новые значения атрибутов простых типов и атрибутных выражений. Предусловие и постусловие также содержит функцию состояния агента $state(t)$, для каждого агента t определенного в environment description.

БП описывают поведение каждого из агентов, их взаимодействие между собой и средой. Каждый из агентов имеет состояние и набор атрибутов, которые описывают его свойства.

Преимущество этого формализма в простоте и эффективности использования методов верификации - язык использует базовый набор структур данных и операций над ними, и при этом обладает достаточной выразительной мощностью для описания систем, специфицированных в таких инженерных языках, как MSC [3], SDL [5], UML [7].

В данной статье рассмотрим также класс базовых протоколов, который является частным случаем конкретных базовых протоколов из [2]. Этот класс можно определить как класс асинхронных императивных базовых протоколов с одним ключевым агентом. Базовые протоколы указанного класса будем называть К-протоколами [8]. Асинхронность понимается в том смысле, что обмен информацией осуществляется по буферизованным каналам. Императивность означает, что каждый протокол эквивалентен некоторой императивной программе, а один ключевой агент полностью определяет функционирование базового протокола.

Далее мы конвертируем либо сценарии, либо описание локальных действий (приход сигнала) в базовые и К-протоколы и рассматриваем сложности конвертации встроенных выражений.

### 3. Сценарный подход при декомпозиции сценариев в базовые протоколы

Сценарный подход широко используется при моделировании систем. Одним из стандартизированных и наиболее используемых языков является язык диаграмм взаимодействия (Message Sequence Charts, MSC) – язык описания поведения системы в виде последовательности событий. События могут относиться к отдельным компонентам системы, к взаимодействию между компонентами системы или к взаимодействию между системой и ее окружением. Основное назначение диаграмм взаимодействия - описание допустимых последовательностей событий в системе. Основным понятием диаграммы взаимодействия является трасса. Для каждого агента на диаграмме есть отдельная вертикальная ось. На этой оси отмечаются события, которые имеют отношение к данному агенту. Трасса – это последовательность всех событий от первого взаимодействия в системе между агентами и до последнего. Считается, что все агенты существуют одновременно, и последовательности событий агентов развиваются параллельно.

Взаимодействие между агентами (а также между агентом и окружением системы) осуществляется только с помощью обмена сообщениями. Сообщение моделирует взаимодействие (обмен информацией) между двумя агентами системы или между агентом и окружением системы. С точки зрения системы, взаимодействие между двумя агентами разбивается на два связанных события:

посылка сообщения одним агентом и прием сообщения другим агентом. Сообщения, которые приходят от среды, моделируются одним событием приема сообщения, а события, которые отправляются в среду, моделируются одним событием посылки сообщения. Сообщение имеет имя. Дополнительно, язык диаграмм взаимодействия позволяет описывать передачу информации в сообщении. К сообщению может быть привязан список параметров. Каждый параметр моделирует передачу конкретной информации от одного объекта к другому. MSC-диаграммы позволяют создавать сложные описания поведения системы с помощью специальных операторов. В MSC используются следующие типы операторов: alt - альтернативный оператор, loop - цикл, opt - опциональная область, reference – ссылка, exc – выполнение, par - параллельный оператор.

Конвертация сценариев – это представление всех событий (входящих, исходящих, локальных) диаграммы в форме базовых протоколов. Рассмотрим подробнее, какие сложности возникают при конвертации диаграммы во множество базовых протоколов, когда opt – общая область для нескольких агентов системы. Мы будем рассматривать работу системы в терминах текстового представления MSC-диаграмм [3]. Проанализируем два случая:

- первое событие в opt-области – отправка сообщения одним агентом другому:

all: opt begin
A: out mes1 to B;
B: in mes1 from A;
opt end;

- первое событие в opt-области – ссылка на другую диаграмму

all: opt begin
all: reference D2;
opt end;

Примечание. Вместо all, может быть перечень агентов, которые принимают участие в opt-области.

Оба агента находятся в диаграмме D. Для определения точной позиции агента в сценарии введем два атрибута типа list. Первый list_of_scenarios – список диаграмм, в которые погружен агент; в голове списка – текущая диаграмма и второй list_of_labels – список позиций агента на диаграмме. Control Flow для агента P – конъюнкция равенства головы списка list_of_scenarios с идентификатором диаграммы, и равенства головы списка list_of_labels с номером позиции в этой диаграмме:

$$\begin{aligned} get\_from\_head(P.list\_of\_scenarios) = D \,\&\, \\ get\_from\_head(P.list\_of\_labels) = number \end{aligned} \qquad (3.1)$$

Отправка сообщения mes1 агентом A и прием его агентом B моделируется двумя kbp(key basic one agent protocol или K-протоколы [8]). K-протоколы имеют семантику SDL: пред- и постусловие ограничены одним агентом, запрещена общая память, сообщения интерпретируются, разрешен только один входящий сигнал для ключевого агента.

Первый протокол (здесь ключевой агент A):

$$\begin{pmatrix}(get\_from\_head(A.list\_of\_scenarios)=D_K)\,\&\\(get\_from\_head(A.list\_of\_labels)=i)\end{pmatrix}\rightarrow$$
$$remove\_from\_head(A.list\_of\_labels);\qquad(3.2)$$
$$add\_to\_head(A.list\_of\_labels,i+1);$$
$$add\_to\_tail(B.queue,mes1)$$

Второй (здесь ключевой агент B):

$$\begin{pmatrix}(get\_from\_head(B.list\_of\_scenarios)=D_K)\,\&\\(get\_from\_head(B.list\_of\_labels)=j)\,\&\\(get\_from\_head(B.queue)=mes1)\end{pmatrix}\rightarrow\qquad(3.3)$$
$$remove\_from\_head(B.list\_of\_labels);$$
$$add\_to\_head(B.list\_of\_labels,j+1);$$
$$remove\_from\_head(B.queue)$$

Пустой MSC-сценарий в терминах БП (здесь ключевой агент любой):

$$\begin{pmatrix}(get\_from\_head(A.list\_of\_scenarios)=D_K)\,\&\\(get\_from\_head(A.list\_of\_labels)=i)\,\&\\(get\_from\_head(B.list\_of\_scenarios)=D_K)\,\&\\(get\_from\_head(B.list\_of\_labels)=j)\end{pmatrix}\rightarrow$$
$$remove\_from\_head(B.list\_of\_labels);$$
$$add\_to\_head(B.list\_of\_labels,j+1);\qquad(3.4)$$
$$remove\_from\_head(A.list\_of\_labels);$$
$$add\_to\_head(A.list\_of\_labels,i+1)$$

Данный подход обеспечивает корректность исходящих трасс и интерливинг агентов при коммуникации сообщениями.

Но протоколы перестают быть kbp, теряется семантика SDL и отсутствует интерливинг в случае пустой MSC-диаграммы, что является недостатками данного подхода.

Разделим протокол (3.4) на два, как это было сделано при передачи и приеме сообщения:

Первый протокол (ключевой агент A):

$$\begin{pmatrix}(get\_from\_head(A.list\_of\_scenarios)=D_K)\,\&\\(get\_from\_head(A.list\_of\_labels)=i)\end{pmatrix}\rightarrow$$
$$remove\_from\_head(A.list\_of\_labels);\qquad(3.5)$$
$$add\_to\_head(A.list\_of\_labels,i+1)$$

Второй протокол (ключевой агент B):

$$\begin{pmatrix}(get\_from\_head(B.list\_of\_scenarios)=D_K)\ \&\\ (get\_from\_head(B.list\_of\_labels)=j)\end{pmatrix}\rightarrow$$
$$remove\_from\_head(B.list\_of\_labels);\qquad(3.6)$$
$$add\_to\_head(B.list\_of\_labels, j+1)$$

После протокола (3.2) может выполниться протокол (3.6), и протокол (3.3) уже принять и обработать сообщение не сможет. Таким образом, возникает проблема синхронизации в opt-областях MSC-диаграмм.

**4. Событийный подход при декомпозиции сценариев в базовые протоколы**

Событийный (event-driving) подход используется в проектах управляемых событиями. Агенты реагируют только на входящее сообщение, изменяя свои атрибуты, и посылают исходящие сигналы другим агентам или среде. Проблема несоответствия трасс сценариям остается: агент, который опционально принимает сообщение, может быть не готов его принять за счет обхода действий opt-области. Поэтому, необходимо «сообщить» другому агенту, когда первый заходит или проходит опциональную область.

**5. Алгоритм синхронизации «извне»**

В системе имеется N агентов. OptController – дополнительно погруженный агент в среду, необходимый для осуществления синхронизации.

Шаг 1. Для каждой opt-области завести атрибут opt_id в каждом из агентов, который принимают участие в этой области и глобальный атрибут. Это идентификаторы opt-области.

Шаг 2. Начальное значение каждого из них Unknown.

$$\forall i=\overline{1..N}: P_i.opt\_id := Unknown\ \&\ opt\_id := Unknown\qquad(5.1)$$

Шаг 3. Для кождого из агентов $P_i$, которые входят в opt-область должны присутствовать протоколы:

$$\begin{pmatrix}(\text{opt\_id}=\text{Unknown})\ \&\\ (get\_from\_head(P_i.list\_of\_labels)=k_i)\end{pmatrix}\rightarrow$$
$$\text{opt\_id} := \text{true};$$
A) $\text{P}_\text{i}.\text{opt\_id} := \text{true};\qquad(5.2)$
$$remove\_from\_head(P_i.list\_of\_labels);$$
$$add\_to\_head(P_i.list\_of\_labels, k_i+1)$$

$$\begin{array}{l}\left(\begin{array}{l}(\text{opt\_id} = \text{Unknown})\,\&\\ (get\_from\_head(P_i.list\_of\_labels) = k_i)\end{array}\right) \rightarrow \\ \text{opt\_id} := \text{false};\\ \text{Б)}\ \text{P}_\text{i}.\text{opt\_id} := false;\\ remove\_from\_head(P_i.list\_of\_labels);\\ add\_to\_head(P_i.list\_of\_labels, q_i)\end{array} \quad (5.3)$$

где $q_i$ - позиция агента на диаграмме после opt-области.

Шаг 4. Для каждого из агентов $P_i$ по последнему действию в opt-области должен присутствовать протокол:

$$\begin{array}{l}\left((get\_from\_head(P_i.list\_of\_labels) = q_i - 1)\right) \rightarrow \\ \text{P}_\text{i}.\text{opt\_id} := exitopt;\\ remove\_from\_head(P_i.list\_of\_labels);\\ add\_to\_head(P_i.list\_of\_labels, q_i)\end{array} \quad (5.4)$$

Шаг 5. Для агента OptController должен присутствовать протокол:

$$\begin{array}{l}\forall i(\text{P}_\text{i}.\text{opt\_id} = exitopt) \rightarrow \\ \text{A)}\ \text{opt\_id} := Unknown;\\ \text{P}_\text{i}.\text{opt\_id} := \text{Unknown}\end{array} \quad (5.5)$$

$$\begin{array}{l}\forall i(\text{P}_\text{i}.\text{opt\_id} = false) \rightarrow \\ \text{Б)}\ \text{opt\_id} := Unknown;\\ \text{P}_\text{i}.\text{opt\_id} := \text{Unknown}\end{array} \quad (5.6)$$

Шаг 6. Для каждого из агентов $P_i$ по первому действию после opt-области должен присутствовать протокол, предусловие которого имеет вид:

$$\left((get\_from\_head(P_i.list\_of\_labels) = q_i)\,\&\,opt\_id = Unknown\right) \quad (5.7)$$

**6. Выводы по результатам и направления дальнейших исследований**

В данной работе рассмотрены проблемы соответствия трасс и диаграмм, сценарный и событийный подходы при разложении MSC-сценариев и диаграмм последовательности событий UML opt-областей на базовые протоколы. Представленный алгоритм обеспечивает корректность трасс, уменьшает сложность генерации SDL-кода, эффективно обеспечивает интерливинг агентов. Алгоритм вводит синхронизацию для агентов, которые участвуют в безусловных областях диаграмм, что является новизной в области верификации требований, которые представлены в виде диаграмм MSC и UML. Таким образом, при декомпозиции сценария получаем абсолютно корректные участки трассы, которые не противоречат исходной диаграмме и не ведут к ошибочным тупиковым состояниям. Данный алгоритм уже успешно внедрен в систему верификации требований VRS. В дальнейшем планируется обеспечить оптимизацию алгоритма для opt-областей высокой вложенности.

ЛИТЕРАТУРА

1. Letichevsky, J.Kapitonova, A.Letichevsky Jr., V.Volkov, S.Baranov, V.Kotlyarov, T.Weigert. Basic Protocols, Message Sequence Charts, and the Verification of requirements Specifications. Computer Networks, 2005, 47. – P.662-675.
2. Letichevsky, J.Kapitonova, A.Letichevsky Jr., V.Volkov, S.Baranov, V.Kotlyarov, T.Weigert. System Specification with Basic Protocols, Cybernetics and System Analyses, 4, 2005.
3. International Telecommunications Union. Recommendation Z.120 Annex B. Formal semantics of Message Sequence Charts, 1998.
4. A.A. Letichevsky, J.V. Kapitonova, V.P.Kotlyarov, A.A. Letichevsky Jr., N.S. Nikitchenko, V.A. Volkov, and T. Weigert. Insertion modeling in distributed system design, Проблеми програмування. 2008. №4.
5. ITU-T Recommendation Z.100 Specification and Description Language (SDL), ITU – 2002.
6. С.В. Потиенко. Методы прямого и обратного моделирования систем, заданных базовыми протоколами, Проблеми програмування. 2008. №4.
7. ITU-T Recommendation Z.109 – SDL combined with UML. – ITU, 1999.
8. А.А. Летичевский. Об одном классе базовых протоколов, Проблеми програмування. 2005. №4

Надійшла у першій редакції 13.02.2011, в останній - 19.04.2011.