УДК 681.324

М.А. ВОЛК, канд. техн. наук, доц. ХНУРЭ (г. Харьков)

ПАРАДОКС МОМЕНТА МОДЕЛЬНОГО ВРЕМЕНИ В РАСПРЕДЕЛЕННЫХ ПОВЕДЕНЧЕСКИХ ИМИТАЦИОННЫХ МОДЕЛЯХ

В статье рассматриваются проблемы управления состоянием распределенной имитационной модели при зафиксированном модельном времени. В качестве формального аппарата описания распределенных имитационных моделей выбраны процессная алгебра и структурная реализации программного представления моделей. Предлагаются пути достижения однозначности состояния модели в неизменный момент модельного времени и в момент отката модельного времени. Библиогр.: 10 назв.

Ключевые слова: распределенные имитационные модели, процессная алгебра, откат модельного времени.

Постановка проблемы. Основной задачей поведенческого имитационного моделирования является правильное отображение порядка и временных отношений между изменениями в моделируемой системе на порядок выполнения событий в модели [1]. Управление временем в имитационных системах является наиболее интересной, сложной, наукоемкой проблемой, которая увеличивает свою сложность с применением распределенных моделей, то есть моделей, функционирующих на разных ресурсах и взаимодействующих между собой. В статье рассматриваются некоторые составляющие проблемы – управление состоянием модели в зафиксированный момент модельного времени и в момент отката модельного времени в прошлое.

Анализ литературы. Наиболее известными способами описания процессов в имитационных средах моделирования являются структурно-алгоритмический [2], на основе процессной алгебры [3 – 5], объектно-ориентированный, например, на основе стандарта HLA (High Level Architecture) [6, 7]. Все они обладают недостаточно эффективным формальным аппаратом описания распределенных имитационных моделей, систем моделирования и алгоритмов их взаимодействия. Некоторые модели, отражающие процесс функционирования распределенных имитационных моделей приведены в [8, 9]. Их общий недостаток заключается в высоком уровне абстракций, который не позволяет достаточно формализовать процесс построения моделирующей среды с целью создания распределенных моделей и автоматизации управления ими. В работе [10] приведен их анализ и предложено процессное представление состояний распределенных имитационных моделей с учетом специфики их программной реализации.

Цель статьи – развить формальный аппарат представления имитационных моделей на основе процессной алгебры [2, 3], который позволит представить вычислительные процессы, протекающие в имитационной системе моделирования, с естественной (программной) точки зрения. Рассмотреть парадокс момента модельного времени, который заключается в том, что в один и тот же момент модельного времени модель может иметь множество состояний, возникающее в результате последовательного или параллельного исполнения кода модели (совокупности частных моделей). Предложить пути достижения однозначности состояния модели в неизменный момент модельного времени и в момент отката модельного времени.

Транзакционные модели. Наличие у программной модели локальных переменных, которые не отражают напрямую состояние модели, приводит к обсуждению вопроса о необходимости их журнализации. Действительно, например, в подпрограмме активности в цикле идет расчет суммы ряда значений или опрос значений на множестве входных сигналов. Переменная цикла в этом случае указывает на номер итерации в цикле и, фактически, не отражается на состоянии модели. Для уменьшения времени выполнения такой активности переменную цикла можно не сохранять через интерфейс данных. В таком случае, с одной стороны, мы получаем выигрыш во времени работы активности, с другой стороны, теряем возможность восстановления работы модели в любой момент времени.

Отметим, что активность может реализовывать все операции с данными через интерфейс данных. В этом случае, модель может быть возвращена не только в одно из состояний, связанных с конкретным значением модельного времени, но и в любое состояние между изменениями модельного времени (то есть в любой момент исполнения программы, на котором произошел сбой или останов). На практике такое свойство необходимо редко (например, для обеспечения высокой надежности в реальном времени), а повышение времени исполнения активности делает такой подход нецелесообразным.

Определение 1. Будем называть активность, которая реализует работу с данными через интерфейс данных *нетранзакционной* активностью, активность, имеющую собственные внутренние нежурнализируемые переменные – *транзакционной активностью*.

Свойство транзакционности подчеркивает тот факт, что, с точки зрения системы моделирования, активность должна выполнить все свои операции от начала до конца, а фиксация состояния такой активности возможно до или после ее исполнения.

Модель, в которой все активности нетранзакционны, сама является нетранзакционной. Модель, в которой хотя бы одна из активностей транзакционна, является транзакционной.

При построении модели разработчик сам решает вопрос о реализации модели. В отношении свойства транзакционности можно выработать критерий

рациональности его применения. Так, если время выполнения активности превышает (или соизмеримо) суммарное время выполнения всех остальных активностей модели, а вероятность сбоя или отката активности велика, можно рекомендовать использование нетранзакционной активности. Если время выполнения активности незначительно по сравнению с другими активностями модели, лучшим решением будет использование транзакционной активности.

Отметим, что вполне допустимо наличие в одной имитационной модели активностей транзакционногого и нетранзакционного типа.

Парадокс момента модельного времени. В большинстве систем моделирования продвижением модельного времени занимается управляющая программа или отдельная подпрограмма частной модели, а поведение имитационной модели состоит в изменении данных модели в зафиксированный, неизменный момент времени t_k. Следовательно, в один и тот же момент модельного времени существуют два разных состояния данных модели: $dm^{t_k} \xrightarrow{A_i} dm'^{t_k}$. Если активность нетранзакционного типа, то таких состояний данных модели может быть несколько.

При реализации отката модельного времени в такой ситуации возникает закономерный вопрос о том, в какое из множества состояний модели, зафиксированных в журнале на момент времени t_k, необходимо вернуть модель.

Определение 2. Состояние модели в некоторый момент модельного времени t_k называется *однозначным*, если с этим моментом времени ассоциируется только одно значение dm^{t_k}.

Утверждение 1. Алгоритмы синхронизации должны осуществлять дамп памяти модели либо до, либо после выполнения всех активностей в момент модельного времени t_k для однозначного определения состояния модели в этот момент времени.

Доказательство. Рассмотрим множество возможных состояний модели в момент времени t_k при условии существования двух активностей, изменяющих состояние модели в этот момент времени:

$$\begin{cases} dm_1^{t_k} \xrightarrow{A_1} dm_1'^{t_k},\ A_1 \in A,\ dm_1 \in dm, \\ dm_2^{t_k} \xrightarrow{A_2} dm_2'^{t_k},\ A_2 \in A,\ dm_2 \in dm. \end{cases} \quad (1)$$

Возможны следующие стратегии осуществления дампа памяти:

1. Дамп памяти осуществляется перед выполнением каждой активности, реализуя, так называемое, префиксное действие:

$$\begin{cases} A_1 = A_{\text{дамп}}(T_k).A_1, \\ A_2 = A_{\text{дамп}}(T_k).A_2. \end{cases}$$

В этом случае, согласно (1), в журнал будет внесено два разных значения состояния модели: $dm_1^{t_k}$ и $dm_2^{t_k}$. В случае выполнения N активностей, таких возможных значений состояния модели будет соответственно N. В результате, однозначного соответствия состояния модели моменту времени не достигается.

2. Дамп памяти осуществляется после выполнения каждой активности, реализуя, так называемое, постфиксное действие:

$$\begin{cases} A_1 = A_1.A_{\text{дамп}}(T_k), \\ A_2 = A_2.A_{\text{дамп}}(T_k). \end{cases}$$

В этом случае, согласно (1), в журнал будет внесено два разных значения состояния модели: $dm_1'^{t_k}$ и $dm_2'^{t_k}$. В случае выполнения N активностей, таких возможных значений состояния модели будет соответственно N. В результате, однозначного соответствия состояния модели моменту времени не достигается.

3. Дамп памяти осуществляется до выполнения всех активностей модели в момент времени t_k. Введем два дополнительных состояния модели dm^{t_k} и dm'^{t_k}, соответствующие состояниям до и после выполнения всех активностей. Тогда при условии префиксного действия

$$A_{\text{дамп}}(T_k).(A_1 \mid A_2),$$

согласно (1), возможны следующие трассы поведения модели ($dm^{t_k} \to dm_1^{t_k} \to dm_2^{t_k} \to dm'^{t_k}$) или $dm^{t_k} \to dm_2^{t_k} \to dm_1^{t_k} \to dm'^{t_k}$.

Независимо от трассы, будет сохранено только одно состояние модели – dm^{t_k}.

Увеличение количества активностей приведет к увеличению количества трасс, но не повлияет на сохраненное состояние. Следовательно, в данном случае однозначное соответствие достигнуто.

4. Дамп памяти осуществляется после выполнения всех активностей модели в момент времени t_k. Тогда при условии постфиксного действия

$$(A_1 \mid A_2).A_{\text{дамп}}(T_k),$$

согласно (1), возможны следующие трассы поведения модели ($dm^{t_k} \to dm_1^{t_k} \to dm_2^{t_k} \to dm'^{t_k}$) или $dm^{t_k} \to dm_2^{t_k} \to dm_1^{t_k} \to dm'^{t_k}$.

Независимо от трассы, будет сохранено только одно состояние модели – dm'^{t_k}. Увеличение количества активностей приведет к увеличению количества

трасс, но не повлияет на сохраненное состояние. Следовательно, в данном случае однозначное соответствие достигнуто.

5. Выполнение дампа памяти до и после выполнения активностей

$$A_{\text{дамп}}(T_k).(A_1 \mid A_2).A_{\text{дамп}}(T_k)$$

приводит к существованию двух значений состояния модели в один и тот же момент времени: dm^{t_k} и dm'^{t_k}. Таким же образом, выполнение дампа памяти до и после выполнения активностей приводит к увеличению количества сохраненных состояний в один и тот же момент модельного времени.

Следствие 1. Для реализации механизмов, позволяющих выполнять неоднократные дампы памяти в один и тот же момент времени, порядок выполнения дампов не позволяет решить задачу однозначного состояния модели в фиксированный момент времени.

Следствие 2. При наличии двух и более дампов памяти в некоторый момент модельного времени t_k не существует однозначности в определении состояния модели в этот момент модельного времени.

Согласно введенных активностей сохранения данных модели существует еще одна активность – $A_{\text{фикс}}(dm_i, t_k)$, фиксирующая изменение части данных модели. Во время работы поведенческой активности таких изменений может быть несколько, в связи с чем ее реализация, в общем случае, гораздо сложнее дампа памяти и может потребовать наличия некоторого атрибута, который будет упорядочивать записи в порядке их совершения. Если мы имеем дело с последовательной моделью (или последовательной частной моделью) эта упорядоченность возникает сама собой. Для того, чтобы модель, которая прошла трассу изменений состояния

$$dm_i \xrightarrow{A_{\text{фикс}}(dm_i', t_k)} dm_i' \xrightarrow{A_{\text{фикс}}(dm_i'', t_k)} dm_i'' \quad \ldots \quad \xrightarrow{A_{\text{фикс}}(dm_i^{'\ldots'}, t_k)} dm_i^{'\ldots'},$$

вернуть в первоначальное состояние, необходимо выполнить упорядоченную обратную последовательность активностей:

$$dm_i^{'\ldots'} \xrightarrow{A'_{\text{фикс}}(dm_i^{'\ldots'}, t_k)} \ldots \quad dm_i'' \xrightarrow{A'_{\text{фикс}}(dm_i'', t_k)} dm_i' \xrightarrow{A'_{\text{фикс}}(dm_i', t_k)} dm_i.$$

Реализация такой трассы возможна в случае, если поведенческие активности подмодели выполняются последовательно или данные разных частных моделей не пересекаются. В случае параллельного (распределенного) выполнения частных моделей или, если данные разных частных моделей пересекаются, требуются более сложные алгоритмы управления данными, требующие дополнительной служебной информации.

Пусть в момент модельного времени t_k произошло L запусков активности $A_{\text{фикс}}(dm_i, t_k)$. Введем множество меток $M = \{1, \ldots, \infty\}$ и активность менеджера памяти $A_{getm}(1)$, которая выдает очередное неубывающее значение $m \in M$ при ее вызове (применение активности $A_{getm}(0)$ приведет к получению текущего значения m):

$$dm_i \xrightarrow{A_{getm}(1).A_{\text{фикс}}(dm_i', t_k)} dm_i' \cup m \,. \qquad (2)$$

Новое значение m сохраняется вместе с новым значением состояния модели и становиться атрибутом измененных данных модели. Очередное значение m может вычисляться по разным законам, например:

1. Неубывающая последовательность: $m_{next} > m_{next\text{-}1}$. Достоинства: все записи упорядочены, наименьшее время определения метки. Недостатки: сложность алгоритмов поиска нужной записи, особенно если необходим просмотр изменений только одной структуры данных; сложность реализации дампа памяти в прошлом.

2. Неубывающая последовательность в пределах одного момента модельного времени: $m_{next}^{t_k} > m_{next\text{-}1}^{t_k}$, $m_1^{t_k} = 1$.

3. Неубывающая последовательность, описанная в пункте 2, формируется для каждой распределенной частной модели в случае децентрализованного менеджера памяти. Образованное тем самым множество последовательностей для организации синхронизации может потребовать введения дополнительных атрибутов, фиксирующих продвижение поведенческих свойств подмодели при пересекающихся данных модели.

Утверждение 2. Наличие атрибута последовательности изменения данных позволяет вернуть модель в любое из прошедших состояний в текущий момент модельного времени.

Доказательство. Применение префиксной активности, предшествующей сохранению изменения подмножества данных (2), приводит к формированию множества записей типа

$$\{\{dm_i, 1\}, \{dm_i', 2\}, \ldots, \{dm_i^{'\ldots'}, m\}\}.$$

Однократное применение активности $A'_{\text{фикс}}(dm_i, t_k)$ приводит к выбору из этого множества записи с номером, на единицу меньшим текущего значения m, то есть модель перейдет в предыдущее состояние. Применение активности $A'_{\text{фикс}}(dm_i, t_k)$ возможно $m - 1$ раз, после чего модель перейдет в состояние dm_i, пройдя все состояния с метками от m до 1.

Следствие 1. Наличие упорядоченного множества изменений данных модели dm_i за счет применения метки из множества M, позволяет вернуть модель в прошлое, непосредственно используя номер метки $1 < m_{old} < m$, минуя промежуточные состояния данных, находящихся между m_{old} и m.

Аналогично дампу памяти, для достижения однозначности состояния модели в момент модельного времени t_k необходимо выбрать из множества изменений данных модели одно, соответствующее одной метке из множества M. Как правило, это минимальное или максимальное значение метки, связанное с t_k. Таким образом, аналогично дампу памяти, при откате модельного времени, осуществляется выборка данных dm_i, сохраненных до либо после выполнения всех активностей модели. Следует отметить, что

последнее правило действует вне зависимости от того факта, является ли модель транзакционной или нет.

Выводы. В данной работе получил развитие формальный аппарат представления программных распределенных имитационных моделей на основе процессной алгебры. Рассмотрены базисные активности журнализации с точки зрения достижения однозначности состояния модели в фиксированный момент модельного времени.

Практическая значимость работы заключается в возможной унификации программных имитационных моделей, упрощении и ускорении процесса их построения на основе простого интерфейса, основанного на управлении данными модели, возможности автоматизации, на базе менеджера памяти, алгоритмов управления модельным временем, в частности оптимистических и консервативных алгоритмов синхронизации распределенных имитационных моделей.

В дальнейшем планируется реализовать известные алгоритмы управления модельным временем в терминах представленного процессного описания.

Список литературы: **1.** *Окольнишников В.В.* Представление времени в имитационном моделировании / *В.В. Окольнишников* // Вычислительные технологии. – Сибирское отделение РАН. – 2005. – Т. 10. – № 5. – С. 57-77. **2.** *Максимей И.В.* Имитационное моделирование на ЭВМ. / *И.В. Максимей*. – М.: Радио и связь, 1988. – 222 с. **3**. *Хоар Ч.* Взаимодействующие последовательные процессы. Пер. с англ. / *Ч. Хоар*. – М.: Мир, 1989. – 264 с. **4.** A Calculus for Communicating Systems, LNCS92, 1980. – 171 p. **5.** The Space and Motion of Communicating Agents, to appear, Cambridge University Press, 2009. – 200 p. **6.** IEEE STD 1278.1-1995. IEEE Standard for Distributed Interactive Simulation – Application Protocols. – N.Y.: Institute of Electrical and Electronics Engineers, Inc., 1995. **7.** *Allen R.* Formal modeling and analysis of the HLA component integration standard / *R. Allen, D. Garlan, J. Ivers* // Proc. of the 6th ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering. – 1998. – P. 70-79. **8.** *Вознесенская Т.В.* Математическая модель для анализа производительности распределенных систем имитационного моделирования / *Т.В. Вознесенская* // Искуственный интеллект (Донецк). – 2002. – № 2. – С. 74-78. **9**. *Миков А.И.* Программные средства оптимизации распределенного имитационного эксперимента / *А.И. Миков, Е.Б. Замятина, А.А. Козлов* // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды Всероссийской суперкомпьютерной конференции (21-26 сентября 2009 г., г. Новороссийск). – М.: Изд-во МГУ, 2009. – 524 с. **10**. *Волк М.А.* Процессное представление состояний распределенных имитационных моделей с учетом специфики их программной реализации / *М.А. Волк* // Вісник Національного технічного університету "Харківський політехнічний інститут". Тематичний випуск: Інформатика і моделювання. – Харків: НТУ "ХПІ". – 2009. – № 13. – С. 23-33.

Статья представлена д.т.н. проф. каф. ЭВМ ХНУРЭ Удовенко С.Г.

УДК 681.324

Парадокс моменту модельного часу в розподілених поведінкових імітаційних моделях / Волк М.О. // Вісник НТУ "ХПІ". Тематичний випуск: Інформатика і моделювання. – Харків: НТУ "ХПІ". – 2010. – № 21. – С. 30 – 37.

У статті розглядаються проблеми керування станом розподіленої імітаційної моделі при зафіксованому модельному часі. Як формальний апарат опису розподілених імітаційних моделей обрані процесна алгебра і структурна реалізації програмного представлення моделей.

Пропонуються шляхи досягнення однозначності стану моделі в незмінний момент модельного часу й у момент відкоту модельного часу. Бібліогр.: 10 назв.

Ключові слова: розподілені імітаційні моделі, процесна алгебра, відкіт модельного часу.

UDC 681.324

The paradox of the model time in distributed behavioral simulation model / Volk M.A. // Herald of the National Technical University "KhPI". Subject issue: Information Science and Modelling. – Kharkov: NTU "KhPI". – 2010. – №. 21. – P. 30 – 37.

The problems of controlling the state of distributed simulation model for a fixed time model. As the formal apparatus of the description of distributed simulation models of selected process-algebra and the structural implementation of the program representation model. The ways to achieve the uniqueness of the state of the model in a constant time model time and time rollback modeling time. Refs: 10 titles.

Keywords: distributed simulation models, process algebra, rollback model time

Поступила в редакцию 15.04.2010