Гурьянов В.И. (Чебоксары)

МОДЕЛИРОВАНИЕ СЛОЖНОГО ПОВЕДЕНИЯ ОБЩЕСТВЕННЫХ И ЭКОНОМИЧЕСКИХ СИСТЕМ

В дискретно-событийных имитационных моделях разнообразие поведения модели достигается за счет использования условного выбора в коде программы. Однако если необходимо моделировать сложное поведение, как, например, процессы развития систем, то соответствующий подход вызывает определенные затруднения. Это связано с тем, что в процессе развития, система проходит ряд стадий, которые могут кардинально изменить процессы, протекающие в системе. В данной статье предложено одно из решений данной проблемы, основанное на использовании паттерна *State*.

Для описания объектных моделей систем удобно использовать специальный профиль UML, предназначенный для объектно-ориентированного имитационного моделирования [1]. В этом формальном языке вводится двойственная семантика – предметная и вычислительная. В докладе [2] рассматривается субпрофиль для моделирования активных систем; дальнейшее изложение построено на этом языке. Языки реализации - Smalltalk и C++; примеры кода приведены на C++.

Модельная задача «Грузовые перевозки». Пусть в пункте Shop производится погрузка товаров в грузовой автомобиль. Товары доставляются в пункт Home, где автомобиль разгружается. Процесс движения автомобиля рассматривать не будем. Объект Shop находится в двух состояниях: «Погрузка» и «Простой». Объект Home также находится в двух состояниях – «Простой» и «Разгрузка». Вся система в целом находится в одном из двух состояниях: ⟨«Погрузка», «Простой»⟩ или ⟨«Простой», «Разгрузка»⟩.

На Рис.1 приведена диаграмма классов для объектной модели. *Исследователь* инициирует моделируемый процесс, посылая сообщение со стереотипом *Exist*, которое последовательно передается всем объектам модели по экземплярам агрегации.

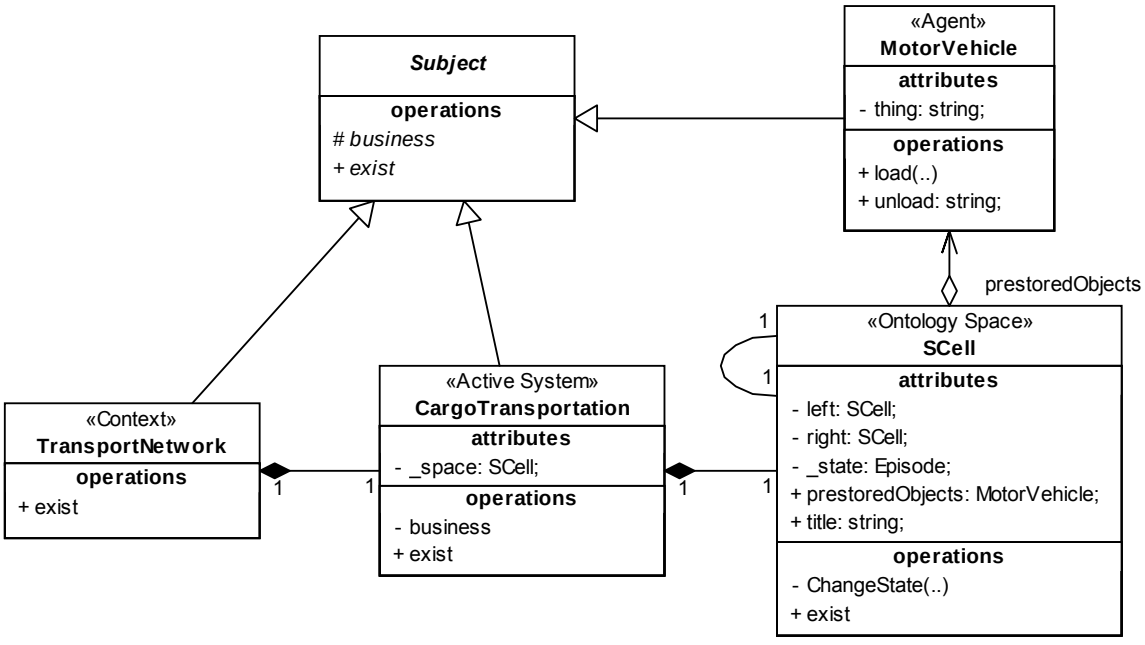

Рис.1. Диаграмма классов для модели «Грузовые перевозки»

В *профиле* каждой программной сущности назначается предметная семантика, для этого используется механизм помеченных значений с меткой Concept. Далее предполагается, где это возможно, что концепты совпадают с именами классов. Класс

TransportNetwork моделирует контекст модели; он определяет системную динамику – свойства и методы класса CargoTransportation. Экземпляр класса CargoTransportation моделирует изучаемую систему. Экземпляр класса MotorVehicle моделирует грузовой автомобиль. В композицию класса CargoTransportation входит динамический список из двух элементов класса SCell; экземпляры этого класса моделируют Shop и Home. Указатель на экземпляр класса MotorVehicle хранится в поле prestoredObjects. Процессы, инкапсулированные в объектах классов TransportNetwork, CargoTransportation, SCell и MotorVehicle, считаются параллельными. Синхронизация между ячейками Shop, Home и объектом класса MotorVehicle осуществляется путем обмена сообщениями.

Выполнение методов load и unload зависит от того, в какой ячейке находится объект класса MotorVehicle и определяется в процедуре business метода exist класса CargoTransportation. Выбор поведения можно осуществить путем проверки значения поля title и условия prestoredObjects == NULL.

Модель хронотопа. Рассмотрим альтернативный способ описания активности модели, используя для создания композиции паттерн *State*. Композиция SCell показана на Рис.2, роли классов расписаны в соответствии с описанием паттерна в книге [3].

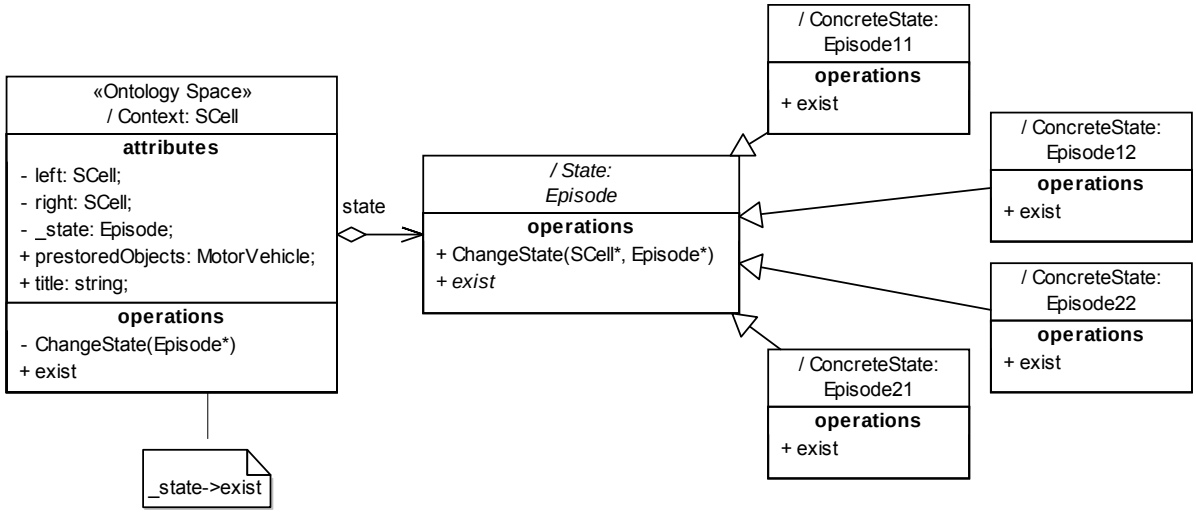

Рис.2. Паттерн State для модели хронотопа

Класс SCell, моделирующий ячейку пространства, выступает в роли контекста паттерна и имеет поле _state класса Episode. Класс Episode {Concept = Эпизод} объявлен другом класса SCell, что дает ему привилегированный доступ к приватным полям и процедурам SCell. Это позволяет манипулировать с объектами, входящими в композицию класса SCell, а также вызывать процедуру ChangeState(Episode*), которая изменяет поле _state. Подклассы Episode реализуют поведение в конкретных ячейках пространства – Episode11 и Episode12 в ячейке Shop, а Episode21 и Episode22 в ячейке Home. Каждая процедура exist подклассов Episode заканчивается переходом в следующее состояние. Например, для Episode11 соответствующий фрагмент кода будет иметь вид

```
Episode * tNext = new Episode12;
ChangeState(j, tNext);
```

, где j – указатель на объект класса SCell. Неустойчивые состояния системы моделируются условным выбором.

Чаще всего хорошим приближением является циклическая смена состояний, как в рассмотренном примере. Однако в некоторых задачах необходимо обеспечить уникальность эпизодов. В этом случае можно воспользоваться паттерном *Singleton*.

<<Research Analysis Model>> (см. [1]) не должна зависеть от конкретного языка программирования; различия проявляются только в *<<Research Design Model>>*. Рассмотренная выше программная конструкция была реализована как на C++, так и на Smalltalk [4]. Паттерн *State* по-разному реализуется на этих языках, однако диаграмма классов Рис.2 будет в обоих случаях одинаковой.

Задача преобразования композиции параллельных процессов *<<Research Analysis Model>>* в квазипараллельный процесс для *<<Research Design Model>>* может быть решена одним из двух способов – опираясь на концепцию событийно-ориентированного или процессно-ориентированного моделирования. Например, в нашем случае, можно определить фиктивное свойство isEnabled в классе MotorVehicle и, перед передачей объекта в смежную ячейку, устанавливать isEnabled = false, что позволяет исходную активность разбить на совокупность независимых деятельностей. Можно показать, что между процессами имеет место отношение *наблюдаемой конгруэнтности* по Р.Милнеру. Т.е. всякое допустимое алгебраическое преобразование процесса *<<Research Design Model>>* может быть выполнено и с процессом *<<Research Analysis Model>>*, верно и обратное. Тем самым обосновывается возможность изучения имитационной модели на *<<Research Design Model>>*. В этом же рабочем потоке определяется модельное время, которое позволяет синхронизировать состояния объектов.

Предметная семантика. Предметная семантика рассмотренных конструкций определяется понятием *хронотоп*, понятие, которое нашло широкое применение в естественных и, особенно, в гуманитарных науках. Под хронотопом мы понимаем последовательность эпизодов, связанных с одной и той же пространственной областью. В нашем случае существуют хронотопы Home, Shop и системы в целом.

Основанием для подобной интерпретации будет следующее. Существенной особенностью хронотопа является тесная связь пространства и времени. Особенно четко это видно на описанной выше модели. В пространстве кинотеатра возможен процесс «Просмотр», в пространстве кафе возможен процесс «Прием пищи». В пространстве пустыни возможен процесс «Идти» и невозможен процесс «Пить воду». Набор объектов манипулирования определяет структуру ячейки пространства. Последовательность действий эпизода определяет временную структуру. Программная конструкция поддерживает единство пространства и времени путем согласования пространственной и временной структуры, причем объекты манипулирования выступают в качестве посредников этого согласования. С одной стороны, поскольку метод exist определен в классе SCell, можно создавать модели с разнородным темпом времени в разных ячейках пространства. Тем самым появляется возможность моделировать системы, разные части которой имеют разные темпы развития, что характерно для больших систем. С другой стороны, поскольку эпизоды хронотопа связаны с одной ячейкой пространства, а ячейки пространства все имеют класс SCell, имеет место преемственность – дискретный аналог непрерывности пространства и времени.

Выбор механизма продвижения модельного времени в значительной степени предопределен моделью времени, которая задается в *<<Research Analysis Model>>*. Проиллюстрируем это на материале книги [5, С.97-101]. Ф.Варелла предложил шкалу актов восприятия для человека, которая состоит из интервалов продолжительностью 0.1, 1 и 10 с. События 0.1 с. он называет моментными когнитивными актами; в нашем случае это будет соответствовать выполнению отдельных операторов методов со стереотипом *Exist*. Акт восприятия 1 с. характеризуется целостностью, соответствует коротким и законченным движениям и моделируется обычным квантом дискретно-событийного времени системы, а механизм продвижения модельного времени имеет список событий, составленный из выделенных точек этих движений. Мгновенный снимок состояния системы после выполнения процедур методов со стереотипом *Exist*, можно назвать *кадром* состояния системы. Акт восприятия продолжительностью 10 с. уже предполагает осознание прошлого и будущего. Для моделирования времени может быть введена

некоторая переменная, общая для всех ячеек пространства, и которой может быть приписан концепт темпорального характера (тем самым переменная становится онтологической сущностью) – «если было выполнено действие *a*, то возможно действие *b*». Механизм продвижения модельного времени должен определяться этой переменной, т.е. особым событием продвижения времени становится *кадр*. Это переходная форма времени между дискретно-событийным временем и «дискретно-кадровым» временем эпизода.

Модель хронотопа предполагает более протяженные интервалы, в течении которых выполняются некоторые законченные действия (как например, погрузка автомобиля) и эти действия уже привязаны к конкретному месту. Механизм продвижения модельного времени должен в качестве особых событий использовать некоторые выделенные кадры. Время становится двухуровневым – «быстрым», дискретно-событийным между двумя кадрами и «медленным», дискретно-кадровым в эпизоде. В хронотопе, на третьем уровне, время можно определить как «историческое»: особые события – это все или некоторые финальные кадры эпизодов. Таким образом, модель хронотопа – это модель квазифрактального дискретно-событийного времени. Причем в этой модели именно «историческое» время является определяющим. Хронотоп системы не однозначен - в зависимости от целей исследования в качестве эпизода можно выбирать действия разного временного масштаба. Заметим, что в контексте модели (класс TransportNetwork) время остается по-прежнему дискретно-событийным. В работе [2] для формального описания программной симуляции было предложено использовать темпоральную логику CTL*, которая хорошо описывает время на каждом из трех уровней хронотопа. Формальный аппарат для «исторического» времени может быть построен также на более простом варианте логики времени Г.Х. фон Вригта.

Возможна другая программная конструкция, также использующая паттерн *State*. В этом случае в качестве контекста паттерна выбирается класс CargoTransportation, для которого вводится поле _state. Это поле – массив из двух элементов класса Episode. Объекты класса SCell хранятся в поле _manipulatesObjects класса Episode и передаются от одного эпизода к другому (вариант паттерна *Memento*). Однако для этой модели времени не удалось определить предметную семантику.

Некоторые примеры. Приведем еще несколько примеров моделей хронотопов, разбив их по категориям (М. М. Бахтин). Эти модели показывают то, как уже было сказано выше, что в хронотопах акцент делается на «историческом» времени.

Хронотоп неравномерного времени. Для каждой пространственной ячейки один эпизод, но в качестве аргумента конструктор класса получает некоторое целое число $H_i = kH_{i\text{-}1}$, где $H_{i\text{-}1}$ – текущее значение H ($H_0 = 1$), а $k \neq 0$ – некоторый целый коэффициент. Число $H>0$ определяет количество повторений процесса эпизода для одного сообщения <<*Exist*>>; если $H<0$, то некоторые сообщения <<*Exist*>> игнорируются.

Хронотоп циклического времени. Один эпизод многократно повторяется, объект хранения имеет собственную линейную историю. Выход из цикла будет иметь место только тогда, когда будет выполнено некоторое условие. Эта модель интересна тем, что требуется формальное доказательство того, что данное условие будет выполнено когда-либо в будущем или, напротив, никогда не будет выполнено.

Хронотоп дороги. Ячейка пространства моделирует средство передвижения – каюту, купе, автомобиль. Перемещение моделируется специальным сообщением, изменяющим состояние системы и источником которого является агент. Эта модель иллюстрирует то, что модель пространства зависит от выбора системы отсчета.

Хронотоп когерентного времени. Пусть есть две пространственные ячейки и два эпизода для каждой из них (e_{11}, e_{12} и e_{21}, e_{22}). Смена эпизодов выполняется по следующему правилу: если для первой ячейки e_{11}, то для второй ячейки следующим эпизодом будет e_{21} и наоборот, если для второй ячейки e_{21}, то для первой ячейки следующим эпизодом будет e_{11}. То же верно для пары (e_{12}, e_{22}). Эта модель интересна тем, что выбор эпизода зависит

от состояния смежных ячеек и выборы согласованы между собой. Пример - когерентность экономических процессов в двух соседних странах.

Хронотоп исторического времени. Рассмотрим процесс в одной пространственной ячейке, такой, что эпизод e_0 сменяется эпизодом e_1или e_2, а затем опять e_0. Выбор альтернативы зависит от предыстории: $e_1 \Rightarrow e_0\mathbf{T}e_2$ и $e_2 \Rightarrow e_0\mathbf{T}e_1$ ($a\mathbf{T}b$ - «*a* и затем *b*»). Модель интересна тем, что демонстрирует сильную зависимость от истории. Пример - динамика политической жизни в двухпартийной политической системе.

Выводы. Имитационное моделирование процессов развития сложных систем требует новых подходов к моделированию времени. Типичный пример развивающейся системы – конфликт в общественной системе, - предполагает, по меньшей мере, три фазы развития: зарождение конфликта, развитие конфликта и разрешение конфликта. Такое разнообразие поведений трудно описать условными выборами в коде программы. Однако главное заключается в другом – такая модель времени не дает адекватного описания процессов изменений системы. Удачным подходом может оказаться модель времени, рассматриваемая в теории хронотопов. В представленном докладе эта модель реализована на основе паттерна *State* и позволяет описывать существенно разнородное поведение систем, как во временном, так и в пространственном срезе.

Литература

1. **Гурьянов В.И.** Специальный UML-профиль для моделирования сложных систем // Информационные технологии моделирования и управления. – Воронеж: Изд-во «Научная книга», 2010. – № 3(62). – С. 356-362. (см. http://econf.rae.ru/article/5385)
2. **Гурьянов В.И.** Логический подход в методологии имитационного моделирования активных систем // ИММОД-2011: труды конференции. – Том 1. – Санкт-Петербург, 2011. – С. 129-133 (см. http://immod.gpss.ru/)
3. **Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.** Приемы объектно-ориентирован-ного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.
4. **Alpert S., Brown K., Woolf B.** The Design Patterns Smalltalk Companion. 1st Edition. - Addison-Wesley Professional, 1998. – 464 p.
5. **Алюшин А.Л., Князева Е.Н.** Темпомиры: Скорость восприятия и шкалы времени. – М.: Издательство ЛКИ, 2008. – 240 с.
#