СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2009. – № 2(56).** – 77–86

*СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ*

УДК 681.3

# ОБ ОСОБЕННОСТЯХ ПРЕОБРАЗОВАНИЯ UML-ДИАГРАММ ДЕЯТЕЛЬНОСТИ В СЕТИ ПЕТРИ

А.А. ВОЕВОДА♠, И.В. ЗИМАЕВ♣

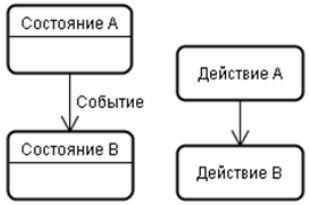
Рассмотрены базовые элементы UML-диаграмм деятельности (activity diagram) в аспекте их отражения на элементарную базу сетей Петри. Предложены соглашения для построения диаграмм деятельности, предназначенных для перевода в сеть Петри. Описаны некоторые дополнительные по отношению к диаграммам возможности раскрашенных и временных сетей Петри. Приведен пример построения сети Петри по диаграмме деятельности согласно рассмотренным правилам.

## ВВЕДЕНИЕ

Для упрощенного представления различных систем широко применяются графические схемы как абстрактные модели, опускающие детали и отражающие общую структуру системы или ее поведение. Использование моделей помогает правильно спроектировать систему и выявить возможные противоречия посредством имитационного эксперимента. Примером визуального представления алгоритмов и процессов может служить стандарт выполнения блок-схем (ГОСТ 19.701-90) или соответствующие ему UML-диаграммы деятельности (activity diagram).

Диаграмма деятельности представляет собой граф, вершины которого обозначают действия, а дуги – переходы от одного действия к другому. Важно отметить, что язык UML обладает недостаточно формальным синтаксисом, а средств для проверки непротиворечивости поведенческих диаграмм не имеет вовсе. Поэтому в качестве метода имитационного моделирования поведенческих диаграмм зачастую применяются сети Петри. И хотя способы преобразования UML-диаграмм в сеть Петри рассмотрены в ряде работ [1, 2], отсутствие формальности синтаксиса самого языка UML весьма затрудняет указанные преобразования. Кроме того, ряд широко применяемых в настоящее время программных элементов (таких как семафоры, критические секции, дорожки, счетчики, исторические состояния) в специальной литературе не отражен.

♠ Профессор кафедры автоматики, д-р техн. наук
♣ Аспирант кафедры автоматики




*а* *б*

*Рис. 1.* различие диаграмм состояний (*а*) и действий (*б*)

Для построения сети Петри обычно [1, 2] используется диаграмма деятельности.

Отличие диаграммы деятельности от другой распространенной поведенческой диаграммы – состояний – показана на рис. 1.

В первом случае элементом диаграммы является состояние, а во втором – состояние действия (действие). На дуге диаграммы состояний обычно указывается событие, приводящее к смене состояний, чего нет на диаграмме деятельности, где переход от одного действия к другому полагается мгновенным.

## 1. БАЗОВЫЕ ПРЕОБРАЗОВАНИЯ

Рассмотрим в качестве примера минимальную конструкцию диаграммы деятельности, представляющую собой одно действие (рис. 2, *а*).

К базовым элементам диаграммы деятельности можно отнести действия и дуги между ними, а к базовым элементам сетей Петри – позиции и переходы. Согласно некоторым [2] источникам, действию на диаграмме соответствует переход в сети Петри (рис. 2, *б*).

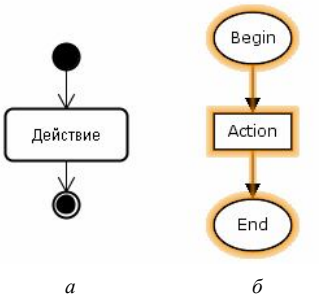
В подтверждение данной трактовки приводятся [2, с. 60] следующие аргументы.

1. Переходы в сетях Петри изначально предназначены для представления событий, а позиции отражают условия для возможности их срабатывания.

2. Маркер сети Петри может быть остановлен только в позиции, но не на переходе. Если бы действия на диаграмме соответствовали позициям, такая ситуация могла бы быть истолкована как приостановка действия (которое может быть и атомарным).

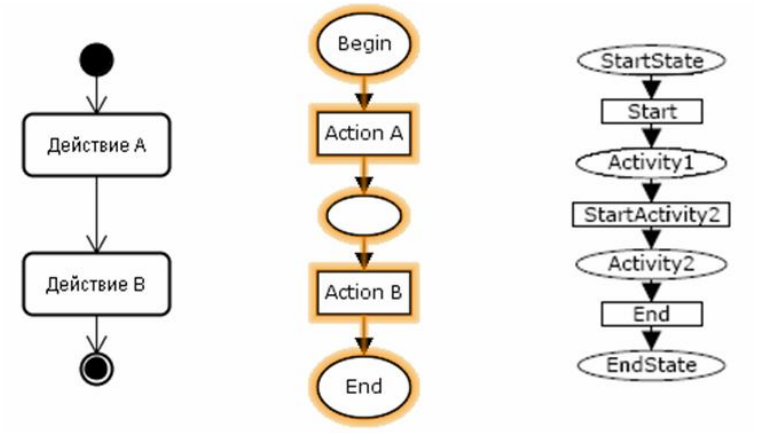
Преобразование последовательности действий осуществляется аналогичным образом (рис. 3 *а*, и *б*).

Как и прежде, каждому действию на UML-диаграмме соответствует переход в сети Петри. Предлагаемый способ не совпадает с действиями, приводимыми в ряде источников, в частности в [1] приводится отраженное на рис. 3, *в*.




*а* *б*

*Рис. 2.* Преобразование одного действия

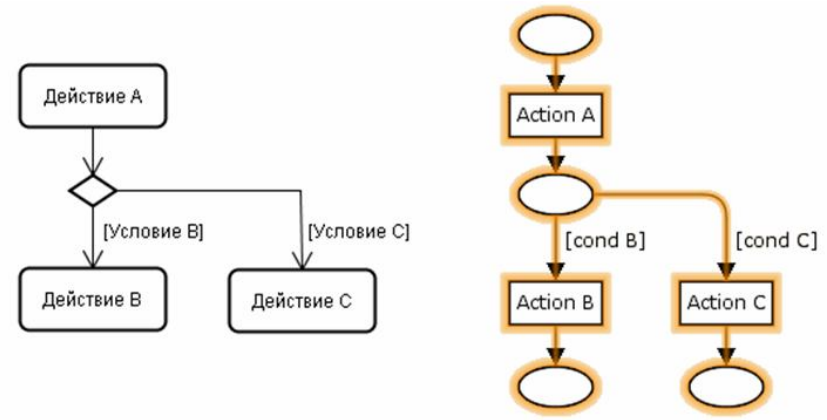



*а* *б* *в*

*Рис. 3*. Преобразование последовательности действий

В стремлении к формально строгому и однозначному синтаксису UML-диаграмм и правил их интерпретации в сеть Петри видится правильным исключать всякую неоднозначность, в частности использовать для обозначения действий только переходы сети Петри, а позициями задавать только условия (наличие ресурсов) и никогда не обозначать ими действия.

Рассмотрим пример преобразования конструкции if-else (рис. 4).




*а* *б*

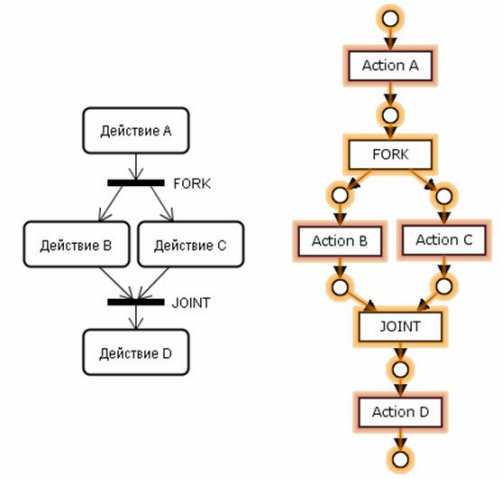
*Рис. 4*. Преобразование конструкции ветвления (*а*) в сеть Петри (*б*)

Условия для принятия решения в сетях Петри задаются как свойства соответствующих переходов. Позиция, предшествующая ветвлению, имеет два выхода, и маркер позиции перейдет через тот переход, условию которого удовлетворяет его значение. Стоит отметить потенциальную возможность ошибок, предупреждение которых целесообразно рассматривать как одно из формальных правил.

1. Если для переходов ветвления не заданы условия их срабатывания, возникает неопределенность, в какую ветку перейдет маркер.

2. В случае если ни одно из условий не является истинным, переход становится недоступным – маркер остается в своей позиции.

Список базовых конструкций диаграмм деятельности завершают операции FORK и JOINT (рис. 5).




*а* *б*

*Рис. 5.* разделение и слияние потоков действий на диаграммах (*а*) и сетях Петри (*б*)

Стоит отдельно отметить физический смысл маркера сетей Петри, при моделировании работы программ в частности. Маркер – это исполняемая в данный момент инструкция. Наличие двух и более маркеров в различных «ветках» сети позволяет некоторым действиям протекать параллельно, что хорошо описывает случай многопоточного приложения. Напрашивается вывод, согласно которому в модели однопоточного приложения всегда должен существовать один маркер, что не вполне верно. Наличие нескольких маркеров во входных для некоторого перехода позициях часто необходимо для запуска перехода. Разделение цепочки действий на потоки происходит при по-

мощи конструкции FORK – обычного перехода, имеющего одну входную и несколько выходных позиций. По правилам сетей Петри при срабатывании переход изымает маркеры из входных позиций и помещает по одному маркеру во все выходные позиции (даже если их количество больше, чем входных). Операция JOINT обладает обратным действием, уменьшающим число маркеров, соединяющим выполнение нескольких потоков в один. Данные конструкции особенно важны при моделировании многопоточных систем.

## 2. ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙ СЕТЕЙ ПЕТРИ

Сети Петри обладают рядом полезных возможностей, отсутствующих в диаграммах UML, поэтому видится целесообразным отразить некоторые из возможностей сетей Петри в диаграммы как дополнительные типы элементов.

Модель раскрашенных сетей Петри, в частности ее реализация в популярном пакете CPN Tools, предполагает наличие следующих базовых элементов:

1) позиций – возможных мест пребывания маркера. Для каждой позиции задаются тип, название и начальная маркировка;

2) переходов, а обозначающих действия. Для каждого из переходов существует возможность задать условие срабатывания, длительность срабатывания и подпрограмму работы (небольшой скрипт на языке ML);

3) дуг, б соединяющих позиции и переходы. Дуге сопоставляется заранее определенная переменная некого типа.

Понятие типа позиции и перехода родственно понятию типов данных в современных языках программирования.

При проектировании систем реального времени важны не только последовательность выполнения действий, но и их временные задержки. В системах смешанного типа могут присутствовать отдельные элементы, к быстродействию которых предъявляются жесткие требования. В процессе моделирования с использованием сетей Петри возникает потребность учета задержек на этих критических ко времени участках. Поскольку индикатором осуществления операций в сетях Петри является маркер, предусмотрена возможность снабдить его внутренним временным счетчиком, увеличиваемым на каждом переходе. А для конкретного перехода возможно задать условное время его срабатывания (информацию о котором необходимо получить, основываясь на изучении реальной системы).

На рис. 6 изображена простейшая сеть Петри, демонстрирующая измерение времени выполнения некого действия, для чего объявляется специальный тип данных int timed. Начальное значение маркера в позиции P1 задано как 2, а его временной счетчик сброшен в ноль. В зависимости от значения маркера происходит действие либо T1, либо T2. Каждое из них обладает своим време-

нем выполнения, заданным как @+3 и @+5 соответственно. После перехода маркера в позицию P2 счетчик времени маркера будет увеличен на значение, имитирующее время прохождение маркера через переход. В моделях систем реального времени может существовать множество переходов с заданной задержкой, очередность прохождения которых определяется сложным набором условий, на выходе такой схемы значение временного счетчика можно соотнести с существующими требованиями по быстродействию. Этот подход можно развивать дальше, например предусматривать возможность определения наибольшего времени прохождения маркера с запоминанием последовательности срабатывания переходов.

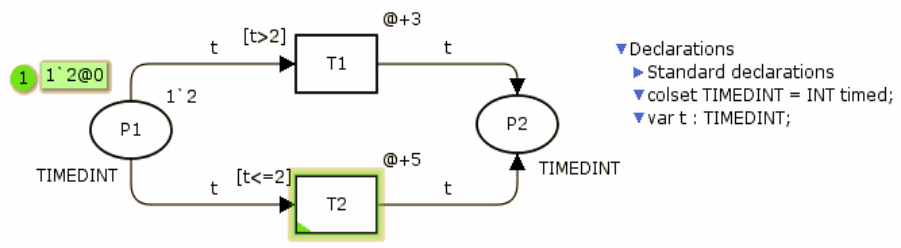



*Рис. 6.* Сеть Петри с измерением времени

Большим функциональным потенциалом для моделирования систем обладает возможность задавать для каждого перехода сети Петри подпрограмму на языке ML (рис.7).

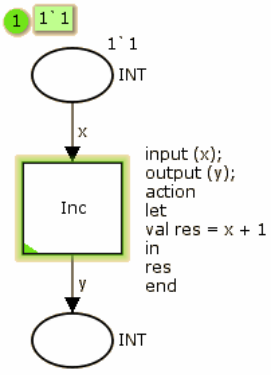



*Рис. 7.* Блок инкремента

В подпрограмме описываются набор входных и выходных данных перехода, а также выполняемые им действия. Предоставлять возможность описать на языке ML суть каждого действия модели, разумеется, было бы избыточными, однако данный механизм весьма полезен для реализации ряда типовых переходов.

На рис.7 представлен простой пример – блок инкремента, т. е. переход, увеличивающий значение проходящего через него маркера на единицу.

Набор подобных типовых звеньев, реализующих базовые логические и арифметические операции, может получить распространение в некоторых предметных областях, например моделях цифровой электроники, схемах принятия решения, основанных на элементах нечеткой логики.

## 3. ЛОГИКА ПЕРЕХОДА ОТ ДИАГРАММЫ ДЕЯТЕЛЬНОСТИ К СЕТИ ПЕТРИ

Основным элементом диаграммы является действие, при этом никак не формализовано, насколько составным это действие может быть. Рассмотренное ранее правило преобразования последовательности действий на диаграмме к совокупности переходов сети Петри правомерно лишь в случае простых действий. Чтобы увидеть неоднозначность преобразования сложных действий, рассмотрим в качестве примера диаграмму для следующей задачи. Имеется квадратное поле 3x3, каждая клетка которого условно содержит некоторое количество ресурсов. На одной из клеток изначально находится игрок, способный перемещаться за один ход на одну клетку в любую сторону (рис. 8).

Цикл работы данной симуляции состоит из следующих шагов:

1) выбор смежной клетки, которая еще не была посещена. Если таковых не осталось, игра заканчивается, иначе переход к п. 2;

2) перемещение в выбранную клетку;

3) ресурсы, сопоставленные с этой клеткой, становящиеся достоянием игрока, т. е. прибавляемые к некоторому счетчику, изначально нулевому. Однако на перемещение затрачивается некоторое фиксированное количество ресурсов. Если суммарное накопление ресурсов стало отрицательным, игрок не может продолжать движение и игра останавливается.

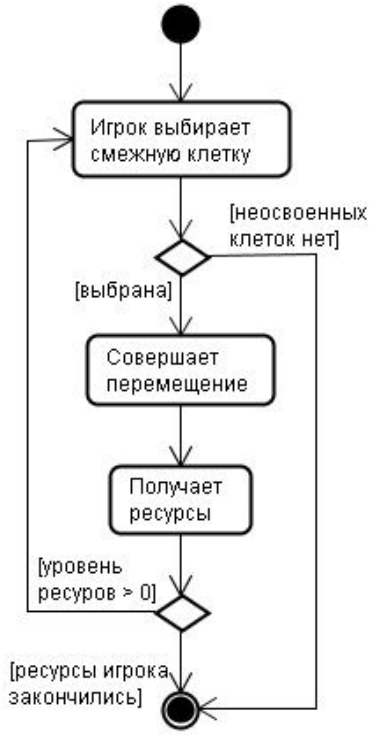



*Рис. 8.* Диаграмма UML

Для имитационного моделирования по приведенной диаграмме нам необходимо построить сеть Петри в соответствии с рассмотренными ранее правилами. Обратим внимание на первое действие: «игрок выбирает смежную клетку». Это сложное действие, в первую очередь по своей сути, нежели количеству вложенных действий. Предлагается раскладывать сложные действия по следующему принципу, отраженному на рис. 9.

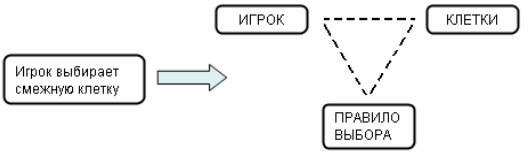



*Рис. 9.* Принцип треугольника сложного действия

Сначала выделяются субъект и объект действия. Если на объект действия накладываются условные ограничения (не всегда может быть доступен, например, канал передачи данных, свободный блок оперативной памяти и т. п.), его необходимо отделить от субъекта действия, так как в сети Петри он будет обеспечивать второе условие для срабатывания перехода действия. Субъект и объект действия в общем случае связаны правилом, в соответствии с которым это действие происходит. Это 3-я вершиной логического треугольника сложного действия.

Отдельно стоит отметить возможность задавать тип позиций и переходов, поскольку это первый шаг на пути интеграции аппарата сетей Петри с языками программирования. Для рассматриваемой модели объявляются следующие типы данных:

colset POSITION = product INT * INT
colset PRICE = INT
colset CELL = PRODUCT POSITION * PRICE
colset PLAYER = product POSITION * PRICE

Тип POSITION обозначает пару целых чисел, предназначенных для обозначения номера строки и столбца клетки, тип PRICE – целое значение количества ресурсов клетки, тип CELL обозначает клетку и включает в себя позицию и цену, тип PLAYER введен для обозначения игрока и хранит в себе его текущие позицию и количество собранных ресурсов.

Объявляемые дуговые переменные:

var playerBeforeStep, playerAfterStep, player : PLAYER
var choiseCell : CELL

Построим сеть Петри на основе приведенных рассуждений (рис.10). Как можно видеть на рисунке, субъект и объект первого действия представляют позиции CurPos и Cells соответственно.

В curPos заданы начальный маркер, обозначающий игрока в позиции (1;1), и обнуленный счетчик накопленных ресурсов. В позиции Cells определено восемь маркеров, соответствующих прочим клеткам игрового квадрата, каждая клетка представлена своими координатами и предопределенным уровнем ресурсов. Переход STEP обозначает действие: выбор смежной клетки и получения ее ресурсов. Для данного перехода задано следующее условие срабатывания:

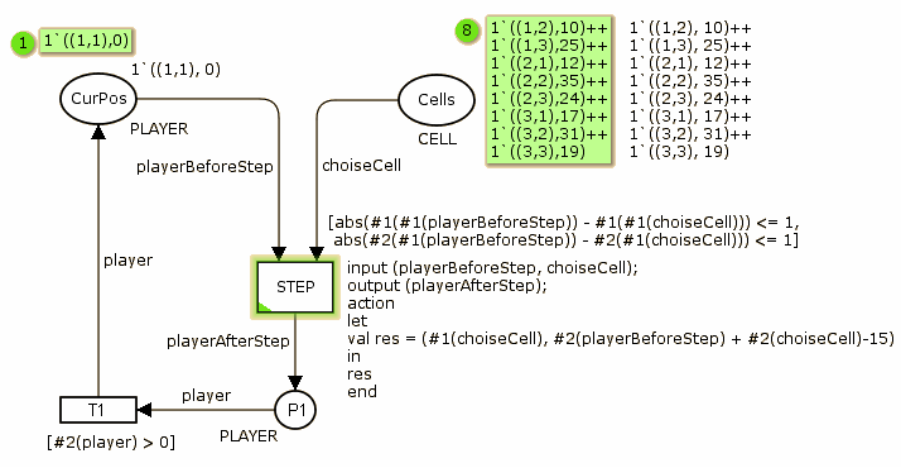



*Рис. 10.* Сеть Петри для задачи сбора ресурсов

[abs(#1(#1(playerBeforeStep)) - #1(#1(choiceCell))) <= 1
abs(#2(#1(playerBeforeStep)) - #2(#1(choiceCell))) <= 1]

Конструкция #1(x) означает «взять первый элемент в x». Таким образом, условие осуществляет проверку: следующая клетка, выбранная для перемещения, смежная с текущей позицией игрока. Благодаря этому условию игрок перемещается в случайном порядке, за один ход преодолевая один квадрат. После каждого хода количество маркеров в позиции Cells уменьшается на единицу, поскольку переход STEP их изымает. Это соответствует тому, что уже посещенные клетки недоступны для перемещения.

Переход STEP содержит следующую подпрограмму:

input(playerBeforeStep, choiseCell)
output(playerAfterStep)
action let
val res = (#1(choiseCell), #2(playerBeforeStep) + #2(choiseCell)-15)
in res  end

Здесь описаны входные (игрок и доступная клетка) и выходные параметры (игрок, совершивший ход). Алгоритм подпрограммы прост: в маркер игрока записывается позиция выбранной клетки, а к его счетчику ресурсов добавляется количество ресурсов этой клетки и вычитается 15 (число, взятое за цену перемещений).

Стоит отметить, что не каждая из возможных последовательность выбора следующей клетки при оговоренных ограничениях приведет к посещению всех клеток. Симуляция сети Петри позволяет построить пространство возможных состояний и увидеть, какие ходы оптимальны, какая стратегия выбора более выигрышная.

## ЗАКЛЮЧЕНИЕ

Базовые элементы раскрашенных сетей Петри уже давно применяются для верификации моделей программных систем, в том числе многопоточных. Временное расширение сетей бесценно при проектировании систем реального времени. Возможности встроенного в CPN Tools языка ML дополнительно расширяют средства моделирования. В первую очередь это касается реализации типовых блоков моделей, что призвано повысить уровень абстракций. Кроме того, применение сложных цветовых множеств фактически стирает границу между сетью Петри и программной реализацией модели, что в будущем может привести к их полной интеграции. Однако сложность сетей Петри растет пропорционально возможностям моделирования, и все более актуальным становится вопрос автоматического построения сети Петри на основе UML-диаграмм. Для решения этой сложной задачи необходима выработка ряда формальных соглашений по построению исходных UML-диаграмм, а также полноценного набора правил однозначных преобразований диаграмм и сетей Петри.

В данной работе предложено отнести к таким соглашениям четкое соотнесение переходов сетей Петри с действиями и, исключая прочие толкования, сложные действия разложить по принципу треугольника: субъект действия–объект действия–правила действия, расширить набор элементов UML-диаграмм типовыми математическими звеньями.

[1] *Коротиков С.В.* Применение сетей Петри в разработке программного обеспечения центров дистанционного контроля и управления: дис. … канд. техн. наук. – Новосибирск: НГТУ, 2007.

[2] *Питерсон Дж.* Теория сетей Петри и моделирование систем / пер. с англ. − М.: Мир, 1984.