Описание систем при помощи X-машин

М.С. Соболев

Аннотация. В данной работе описывается способ построения систем из X-машин, позволяющий создавать масштабируемые системы, компоненты которых можно легко тестировать и использовать в дальнейшем.

Ключевые слова: X-машина, сложные системы, автоматы.

Существует множество подходов к созданию (описанию и реализации) сложных программных систем, однако пока нет оснований полагать, что какие-либо из них, за исключением формальных методов, ведут к созданию «правильных» систем. В последние десятилетия было выдвинуто множество аргументов как за, так и против формальных методов описания таких систем, однако в итоге стала ясной необходимость в формальных методах. В описаниях программных систем преобладают модели, основанные на типах данных, например, функциональные, реляционные (Z [1], VDM [2]) или аксиоматические (OBJ [3]). Несмотря на то, что эти модели значительно облегчили проектирование программного обеспечения, они недостаточны для описания динамического поведения систем. Кроме того, следует учесть, что процесс создания рабочей системы по прямому формальному описанию несет в себе определенные трудности. Другие формальные методы, такие как конечные автоматы [4] или сети Петри [5], позволяют подробно описать систему в динамике, однако плохо описывают изменения внутренних данных во время переходов между состояниями. Существуют методы, хорошо описывающие как динамическое поведение системы, так и процессы, происходящие с данными, например метод Statecharts [6], однако эти методы недостаточно формализованы. До сих пор не существует формального метода, удовлетворительно описывающего все стадии проектирования системы ПО: моделирование, верификацию и тестирование.

В данной работе описывается способ построения системы из X-машин, не требующий существенного изменения традиционного определения X-машины. Этот метод позволяет создавать масштабируемые системы, компоненты которых можно легко тестировать и использовать в дальнейшем.

X-машина – это модель позволяющая описывать системы на всех этих стадиях. Эта модель впервые была введена в [7] и расширена в [8]. X-машина напоминает конечный автомат, однако существуют два важных различия:

- каждой X-машине соответствует некий набор данных (содержание памяти),
- переходы зависят не только от входных данных, но являются функцией от входного значения и этого набора данных.

Эти отличия делают X-машины более содержательным и гибким методом описания систем, чем конечные автоматы. Как было сказано выше, большинство существующих методов описывают либо динамическое поведение системы, либо изменения данных. Существует интересный тип X-машин, называемый *поточные X-машины* (stream X-machines, SXM), которые позволяют описывать нетривиальные структуры данных типизированными упорядоченными списками в памяти. Поточные X-машины используют диаграммы для моделирования переходов, расширяя при этом возможности конечных автоматов. Пе-

реходы между состояниями определяются функциями, описанными формальным языком и моделирующими обработку данных, находящихся в памяти. Эти функции получают входные символы и данные из памяти и в процессе преобразования их в выход изменяют содержимое памяти.

Машина, основываясь на текущем состоянии и данных в памяти, читает очередной входной символ и определяет новое состояние, модифицирует данные в памяти и определяет выходной символ, который будет являться частью выходного потока. Приведем формальное описание Х-машины.

Х-машина определяется с помощью восьмерки:

Σ, Г - входной и выходной алфавиты, соответственно;

Q – конечный набор состояний;

М – память, набор (возможно бесконечный) типизированных символов;

Ф- конечный набор функций φ, отображающих входной символ и содержание памяти в выходной символ и новое содержание памяти φ: Σ×М →Г×М;

F – функция, отображающая состояние и функцию из Ф в новое состояние F: Q×Ф;

q0, m0 - начальное состояние и начальное содержание памяти, соответственно.

Работа Х-машины заключается в выполнении некоторого количества шагов (тактов), на каждом из которых происходит чтение символа из входной цепочки. Исходя из текущего состояния Х-машины выбирается функция из Ф, которая генерирует выходной символ и, возможно, неким образом модифицирует память.

Переходы между состояниями задаются функцией F, которую удобно представить графически в виде диаграммы переходов между состояниями. Представим себе ориентированный граф, в котором каждому состоянию соответствует вершина, а каждой функции из Ф – дуга. Тогда дуга, помеченная символом a, соединяет две вершины Q_1 и Q_2, если $Q_2 = F(Q_1, a)$.

Легко заметить, что определение Х-машины расширяет определение конечного автомата. Отметим, что Х-машины обладают большей общностью, чем конечные автоматы. Проиллюстрируем это, задав с помощью Х-машины все цепочки вида $a^n b^n c^n$. Как известно, это невозможно сделать при помощи автоматов с магазинной памятью или конечных автоматов.

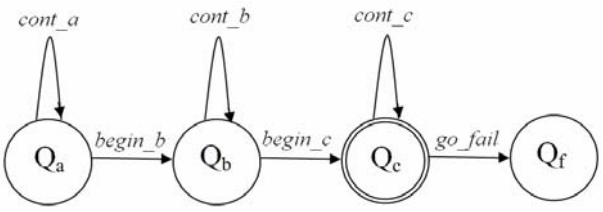

Рис.1 Диаграмма переходов между состояниями

На Рис. 1 представлена диаграмма переходов между состояниями Х-машины (таким образом, задана функция F). Будем считать, что данная машина допускает цепочку входных символов, если в результате работы Х-машины все входные символы цепочки будут считаны и машина при этом будет находиться в состоянии Q_c. По аналогии с конечными автоматами назовем его конечным состоянием.

Зададим входной алфавит, память, а также начальное состояние Х-машины. Выходной алфавит у данной машины пуст.

Σ={a,b,c}

M={n_b, n_c}

$q_0 = Q_a$ m_0={0,0}

Наконец, требуется задать функции перехода из Ф, которые будут модифицировать память, основываясь на состоянии Х-машины и входном символе:

cont_a('a', {n_b, n_c}) : $n_b' = n_b+1$; $n_c' = n_c+1$

begin_b('b', {n_b, n_c}) : $n_b' = n_b-1$

cont_b('b', {n_b, n_c}) : $n_b' = n_b-1$

begin_c('c', {n_b=0, n_c}) : $n_c' = n_c-1$

cont_c('c', { n_b=0, n_c>1 }): $n_c' = n_c-1$

go_fail(x, { n_b=0, n_c=0 }):.

Данной Х-машиной допускаются только цепочки вида $a^n b^n c^n$; таким образом Х-машина позволяет задавать цепочки, которые невозможно задать при помощи конечных автоматов.

Система связанных поточных Х-машин (communicating stream X-machines, CSXM) с n компонентами определяется следующим образом: $CSXM_n = ((XMC_i)_{i=1,...n}, CM, CM_0))$, где XMC_i – i-я Х-машина в системе, CM – матрица n*n, называемая *матрицей связи* (communication matrix), CM_0 - начальная матрица связи.

В дальнейшем мы будем называть компоненты системы связанных потоковых Х-машин

XMC, а не X-машинами, так как определение XMC отличается от определения отдельной X-машины.

В описываемой модели (Рис.2) связь XMC осуществляется через матрицу связи. Элементы матрицы содержат «сообщения» от одной XMC к другой. Элемент (i,j) содержит сообщение от XMC_i к XMC_j, то есть XMC_i осуществляет чтение только из i-го столбца и осуществляет запись только в i-ю строку.

Элемент матрицы может содержать значение λ, которое обозначает отсутствие сообщения, а элементы (i,i) остаются пустыми.

Сообщения могут иметь любой тип, определенный в памяти всех XMC. Для того, чтобы XMC_i могла осуществлять отправку или прием сообщений, у нее должен быть определен входной порт IP_i и выходной порт OP_i, которые содержат индивидуальные для каждой XMC элементы. Типы этих элементов, IN_i и OUT_i соответственно, определяются как наборы значений из памяти M_i или как значение λ.

Для того чтобы XMC могли использовать IP и OP порты, существует специальный тип состояний и функций, называемых *коммуникационными состояниями* и *коммуникационными функциями* соответственно. Таким образом, в каждой XMC существуют:

- функции обработки, которые влияют на содержимое IP и OP портов, но не изменят матрицу связи,
- коммуникационные функции, которые считывают элемент матрицы в IP порт или записывают содержимое OP порта в соответствующий элемент матрицы.

Коммуникационные функции вызываются только из коммуникационных состояний, принимают пустой символ ε в качестве входного и генерируют этот же символ в качестве выходного значения, не модифицируя при этом память. Коммуникационные функции могут осуществлять запись в матрицу СМ только в том случае, если в соответствующем элементе матрицы содержится значение λ. Если коммуникационная функция считывает значение из матрицы СМ, то на место этого значения записывается λ. Если коммуникационную функцию невозможно применить в данный момент, то она ожидает момента, когда это можно будет сделать.

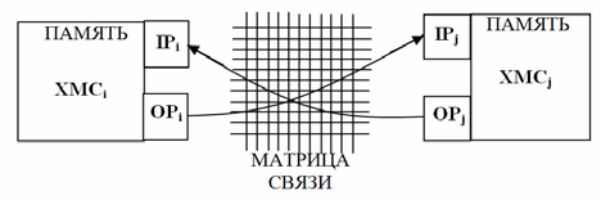

Рис. 2. Обмен данными между XMC

Рассмотрим пример комбинации отправителя и получателя сообщений. Отправитель генерирует объекты, а получатель получает пару объектов и производит над ними некоторую операцию. Две XMC, вместе с их ***состояниями обработки, коммуникационными состояниями*** и функциями, представлены на Рис.3.

Отправитель получает входной поток и выполняет функции обработки *начать* и *продолжить,* которые записывают частично сконструированный объект в память. После этого объект записывается в выходной порт OP и машина переводится в передающее состояние функцией *закончить*. Затем функция *переслать* пересылает объект из выходного порта в матрицу связи.

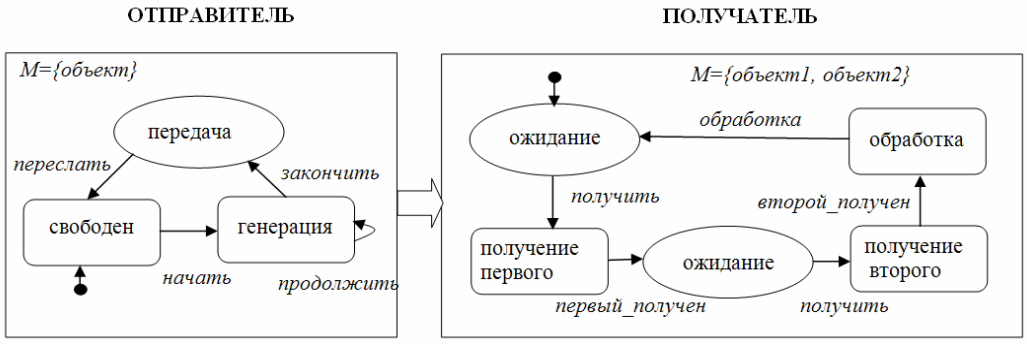

Рис. 3. Отправитель и получатель

Получатель, в свою очередь, находится в состоянии ожидания до тех пор, пока объект не появится в матрице связи. Когда это происходит, активируется коммуникационная функция, записывающая объект во входной порт получателя IP. После этого функция обработки перемещает объект в память получателя. Когда таким же образом считывается второе значение, для обоих значений вызывается функция *обработка.*

Можно дать формальное описание этим двум ХМС, однако такое описание будет выходить за рамки данной работы. Легко заметить, что описанные выше ХМС отличаются от одиночных X-машин, не предназначенных для коммуникаций между собой.

Вышеописанный подход к созданию систем связанных X-машин представляет возможность тестирования системы, обеспечивая, таким образом, ее корректность. Главным недостатком данного подхода является необходимость рассмотрения системы как единого целого, а не как набора независимых компонентов. Вследствие этого при создании большей системы тестирование требуется производить снова. Кроме того, ХМС не могут быть использованы в качестве одиночных X-машин или частей какой-либо другой системы.

Немаловажным обстоятельством является то, что семантика функций, изменяющих матрицу связи, накладывает ограничения на функционирование ХМС в асинхронном режиме. Например, ХМС-отправитель не может приступить к конструированию второго объекта до тех пор, пока первый объект не будет получен ХМС-получателем (элемент коммуникационной матрицы содержит значение, что заставляет отправителя оставаться в состоянии *transmitting*).

В данной работе будет описан альтернативный подход к созданию систем связанных X-машин, разработанный с учетом возможности их практического применения. Мы не будем вводить ХМС, как было сделано в описанном ранее подходе, но ограничимся стандартным определением X-машин, что позволяет использовать обыкновенные X-машины в сколь угодно больших системах.

Такой подход имеет ряд важных преимуществ для разработчика:

- не требуется полностью разрабатывать новую систему коммуникаций,
- можно повторно использовать части других систем,
- при разработке системы можно разделить создание отдельных X-машин и моделирование их коммуникации,
- возможно использование существующего инструментария для разработки X-машин.

Предлагаемая методология создания систем X-машин состоит из трех основных шагов:

- разработка моделей X-машин независимо от системы в целом либо использование существующих X-машин;
- определение способа коммуникации полученных моделей X-машин;
- расширение полученной системы, позволяющее ей оперировать большим количеством X-машин.

Далее все три шага будут проиллюстрированы простейшим примером.

Моделирование независимых X-машин

Проиллюстрируем практическое применение X-машин с помощью простого примера. Попробуем смоделировать с помощью X-машины процесс разработки программного обеспечения (Рис.4). Допустим, что вся необходимая работа разбита на простые части. Каждая часть работы (назовем ее заданием) проходит через несколько групп разработчиков, прежде чем ей присваивается статус выполненной.

Первый шаг состоит в моделировании двух частей системы: группы разработчиков и очереди (пула) из заданий.

Память X-машины для очереди заданий хранит последовательность заданий, которые необходимо передать в группу разработчиков. Функции активируются прибытием очередного задания или сигналом *задание_выполнено* в том случае, если рабочая группа освободилась.

Вторая X-машина моделирует рабочую группу, которая выполняет задания. В памяти X-машины для светофора хранится количество тактов (времени), прошедшего с получения последнего задания и продолжительность выполнения задания в тактах. Функции активируются

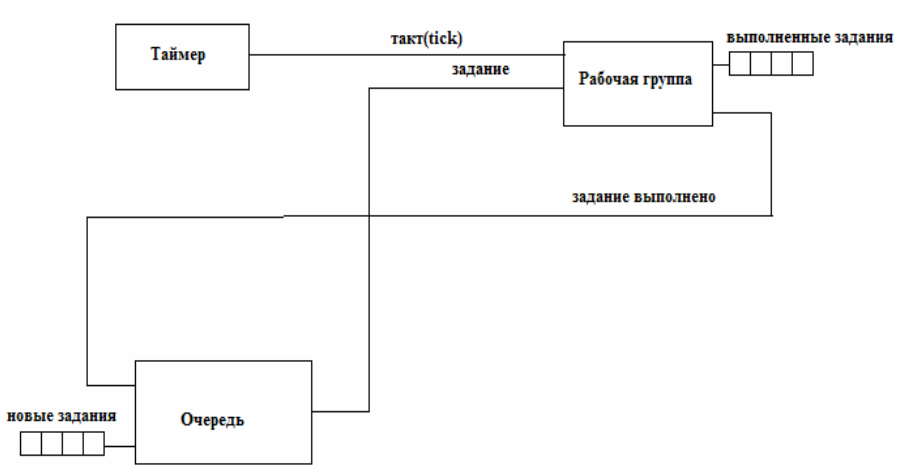

Рис. 4. Рабочая группа и очередь из заданий

входным сигналом, генерируемым таймером на каждом такте. Переходы между состояниями для моделей представлены в виде диаграмм на Рис. 5 и Рис.6. У группы есть два состояния: *занята* и *свободна*.

Построение системы коммуникаций

В нашем подходе мы заменили матрицу сразу несколькими входными потоками, ассоциированными с компонентами X-машины. Несмотря на то, что это кажется другим способом описания одного и того же подхода, данная замена позволяет создать существенно иную систему, в которой каждая из X-машин работает в асинхронном режиме. X-машины используют собственные входные потоки, но количество этих потоков может увеличиваться по мере роста системы.

Может быть реализован следующий механизм взаимодействия ранее определенных X-машин: когда состояние группы меняется на *свободна*, отправляется сообщение очереди, которая выдает группе очередное задание (в случае наличия заданий в очереди). Сообщение *задание_выполнено* отправляется очереди в случае отсутствия задания на очередном такте времени. В это же время в очередь могут вставать новые задания. Таким образом, сообщения от X-машины - «группы» посылаются на вход X-машины - «очереди».

В случае, если это указано в описании, функции могут принимать входные данные из коммуникационных потоков вместо стандартного входа и посылать выходные символы на вход другой X-машины.

Расширение системы

Разработка более общих систем из готовых «строительных блоков» требует добавления некоторых новых частей в систему. Например, система с несколькими светофорами требует отдельной X-машины для задания порядка их переключения.

В рамках XMC-подхода разработку системы придется начать с самого начала, переопределив XMC. В нашем подходе такое обобщение потребует лишь модификации коммуникаци-

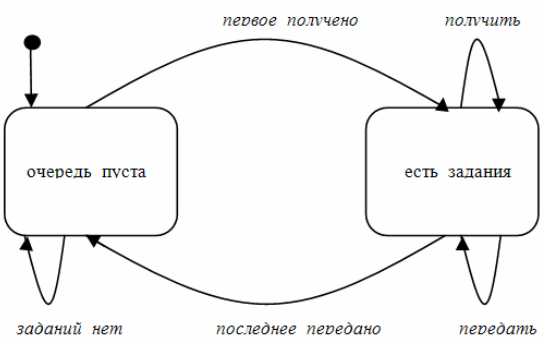

Рис. 5. Диаграмма переходов между состояниями для очереди

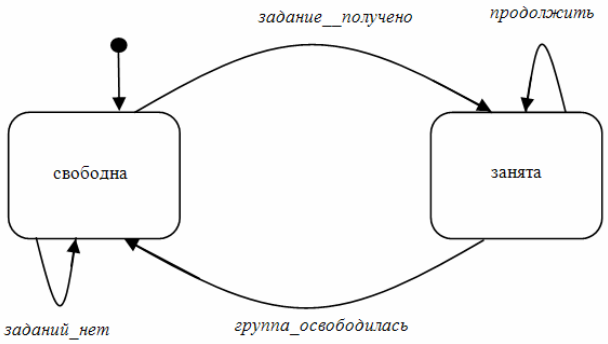

Рис. 6. Диаграмма переходов между состояниями для группы разработчиков

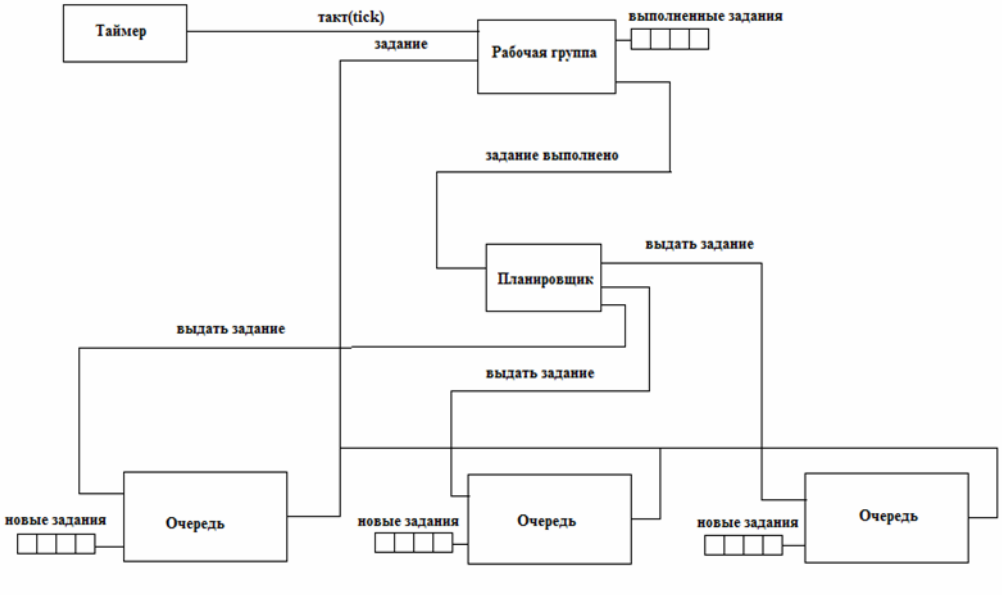

Рис. 7. Схема с тремя очередями

онной части и добавления одной X-машины – «планировщика».

Рассмотрим пример с рабочей группой, которая обрабатывает задания из трех очередей. Для этого дополнительно понадобится планировщик, который будет пересылать сигнал *задание_выполнено* одной из очередей. Планировщик может быть написан разработчиком или взят из какой-либо уже существующей системы, так как сущность планировщика не зависит от типа других X-машин (т.е. «очереди» и «группы»).

Схема системы с тремя очередями представлена на Рис.7. Как видно, нам не понадобилось что-либо изменять в моделях очереди и группы, созданных в самом начале. Однако требуется создать несколько экземпляров «группы», каждый со своим начальным состоянием и памятью.

Литература

1. M. Spivey, The Z Notation: A Reference Manual, Prentice-Hall, Englewood Cliffs, NJ, 1989.
2. C.B. Jones, Systematic Software Development using VDM, Second ed., Prentice-Hall, Englewood Cliffs, NJ, 1990.
3. K. Futatsugi, J. Goguen, J.-P. Jouannaud, J. Meseguer, Principles of OBJ2, Proceedings of the 12th ACM Symposium on Principles of Programming Languages, 1985.
4. Ахо А., Ульман Дж. Теория синтаксического анализа, перевода и компиляции. М.: Мир, 1978. - Т. 1,2
5. Питерсон Дж. Теория сетей Петри и моделирование систем. – М.: Мир, 1984.
6. D. Harel, Statecharts:a visual approach to complex systems, Science of Computer Programming 8 (3) (1987) 231–274
7. S. Eilenberg, Automata Machines and Languages, vol. A, Academic Press, New York, 1974.
8. M. Holcombe, X-machines as a basis for dynamic system specification, Software Engineering Journal vol. 3 (2) (1988) 69–76.

Соболев Михаил Сергеевич. Аспирант Московского физико-технического института. Окончил магистратуру МФТИ в 2007 году. Имеет 2 публикации. Область научных интересов: компьютерная лингвистика, теория автоматов.
E-mail: sobolev@competentum.ru.