ВИЗУАЛЬНАЯ РАЗРАБОТКА ИМИТАЦИОННЫХ МОДЕЛЕЙ

Т. В. Девятков (Казань)

Имитационное моделирование – один из самых мощных инструментов анализа, которыми располагают люди, ответственные за разработку и функционирование сложных процессов и систем. В настоящее время создано и эксплуатируется большое число систем имитационного моделирования. Одной из таких систем, прошедшей испытание временем, но пользующейся большой популярностью, является GPSS World. Но к сожалению, несмотря на очевидные преимущества, ее использование затрудняется из-за отсутствия полноценного визуального интерфейса. Поэтому автором была начата работа по разработке визуальной графической среды моделирования VSM, основанной на мощном моделирующем ядре – языке GPSS World.

Визуализация в современном представлении несет в себе огромный потенциал развития средств ИМ, состоящий в упрощении самого процесса моделирования и предоставляет человеку возможность сосредоточиться не на знании точных описаний функций, переменных и операторов, а на более полном и объемлющем понимании сути модели. В связи с этим большая часть усилий пользователя будет уделяться именно моделированию, которое, вследствие введения визуализации, становится более качественным.

Основные задачи, решаемые VSM, можно представить в следующем виде.

1. Анализ внутренней структуры системы, с разбиением на однотипные элементы, отличающиеся лишь параметрами;
2. Собственно построение, графически, визуальной модели посредством представленных системой инструментов;
3. Разработку собственных или использование готовых шаблонов для каждого элемента системы. Каждому элементу должен быть поставлен в соответствие шаблон.
4. Генерация модели в соответствии с введенной структурой и с пользованием шаблонов. Генератор моделей представляет из себя открытый для пользователя текст программы, написанной на встроенном языке PascalScript. Это позволяет вносить собственные коррективы в работу генератора.
5. Запуск имитационной модели в «моделирующем ядре».

Вопросы способов представления результатов и методов их анализа не являются функцией VSM и исключены автором из рассмотрения.

Далее рассмотрим более детально строение внутренней структуры VSM.

Из каких частей состоит структура системы и на каких классах она основывается, представлено на диаграмме классов (рис. 1).

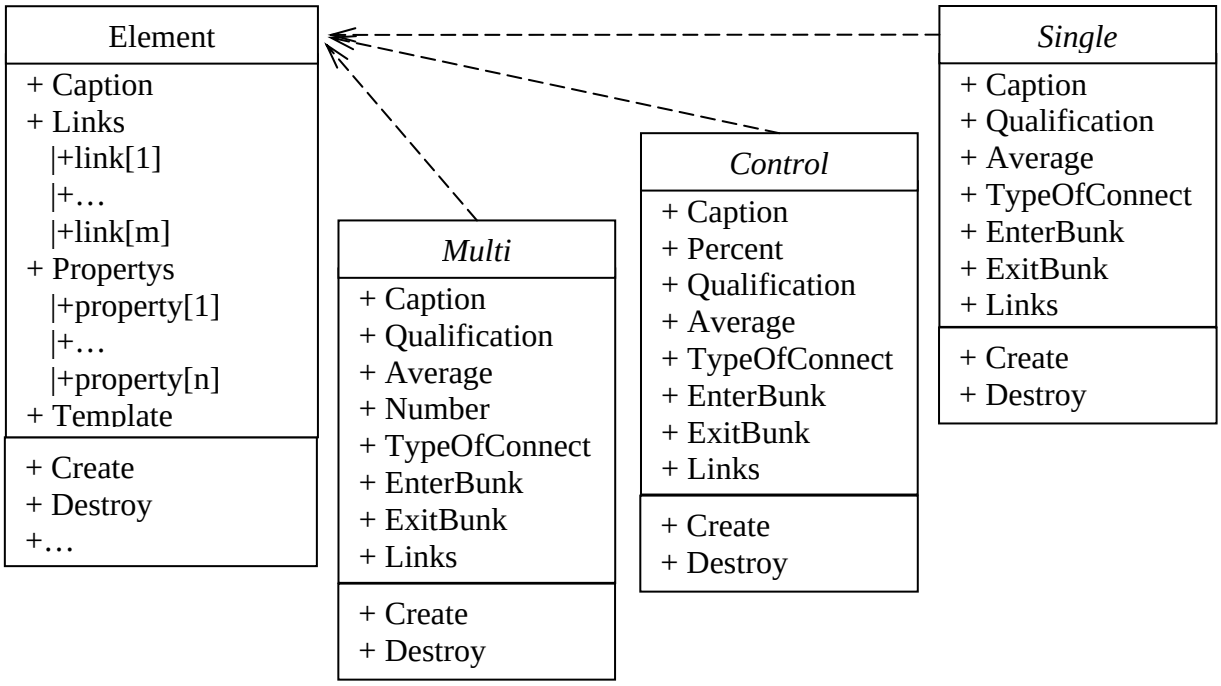

Рис. 1. Диаграмма классов

Здесь класс Element является родительским классом, который описывает основные свойства всех элементов, их интерфейс. Остальные перечисленные классы представлены как примеры каких-то дочерних классов, которые наследуют структуру по заданному интерфейсу. Вся структура системы построена на основе работы класса Element.

Отношение между этими представленными в примере классами можно представить в виде диаграммы ассоциации (рис. 2):

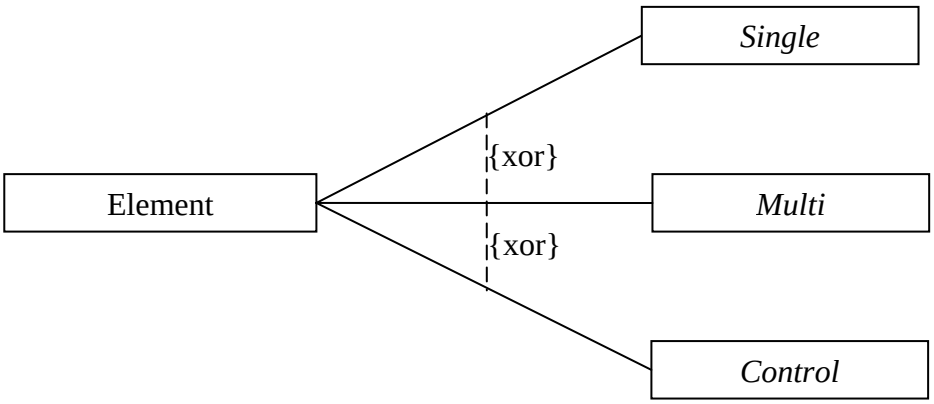

Рис. 2. Диаграмма исключающей ассоциации классов

Для унификации вложенных структур, экземпляр класса Element может представлять из себя элемент схемы всего лишь с одним параметром, указывающим на файл или область памяти, где хранится вложенная схема. Т. е. любой элемент в схеме может представлять из себя другую схему, и т.д. до нужной пользователю точности представ-

ления схемы. Ограничением на использование такой степени вложенности может выступать лишь здравый смысл и возможности компьютера.

Существует два основных, принципиально различных, пути обеспечения универсальности.

Первый путь заключается в выявлении и реализации только базовых, наиболее элементарных и общих для всех приложений функций, что перекладывает основную часть работы на пользователя. Такой подход обеспечивает в большинстве случаев максимальную эффективность использования функциональных возможностей системы, но предполагает высокую трудоемкость создания и модификации сложных моделей.

Второй путь заключается в реализации максимального количества различных функций, покрывающего весь диапазон возможных применений. В этом случае, как правило, действует следующая закономерность: чем выше функциональность системы, тем сложнее она в освоении и тем менее эффективно используются её возможности.

Для данной программы был использован средний путь, не относящийся ни к первому, ни ко второму напрямую, т. к. пользователь может сам выбирать – работать ли со встроенными шаблонами, количество которых неограниченно, или же самому писать шаблоны, спускаясь к нижнему и самому сложному уровню.

Здесь первый путь представлен на уровне создания шаблонов (Create templates), т. е. написание собственноручно кодов на GPSS. А второй путь – на остальных пунктах использования программы.

Отдельно должен быть рассмотрен и генератор моделей. Для его функционирования в «систему визуального моделирования» был встроен язык PascalScript, который позволяет более гибко управлять работой генератора. Для написания кода на этом языке достаточно лишь поверхностно знать язык Pascal, который предельно прост в использовании.

В программе взаимодействие элементов рассматривается в информационном аспекте их коммуникации, т. е. взаимодействующие объекты обмениваются между собой некоторой информацией. При этом информация принимает форму законченных сообщений. Тогда работу с программой можно представить следующей диаграммой, отображающей весь процесс работы в порядке следования во времени (рис. 3).

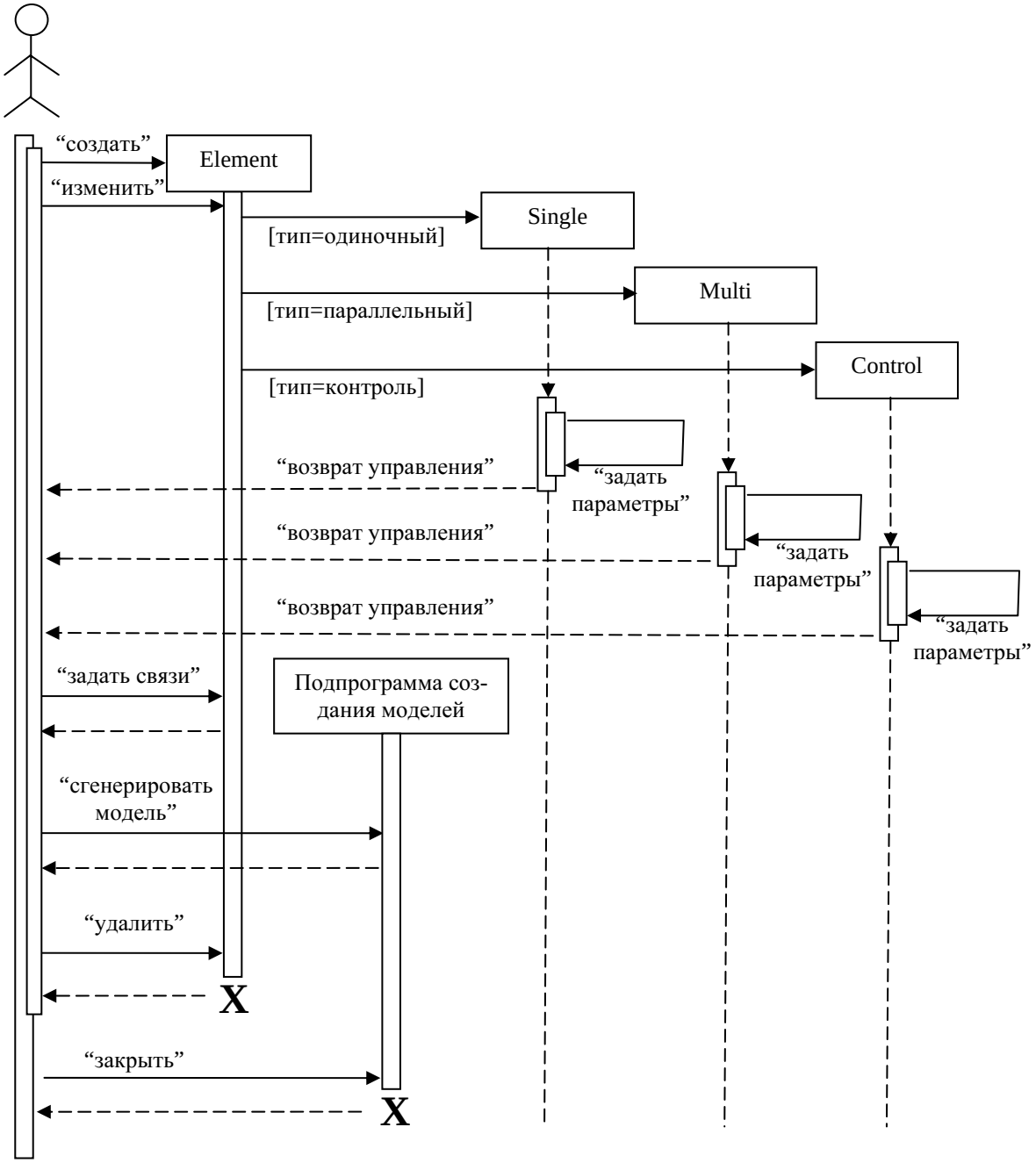

Рис. 3. Временная диаграмма

На рис. 4–6 показана работа с системой на конкретном примере моделирования конвейерной сборки компьютеров. На схеме видны три линии, два генератора запросов и склад готовой продукции.

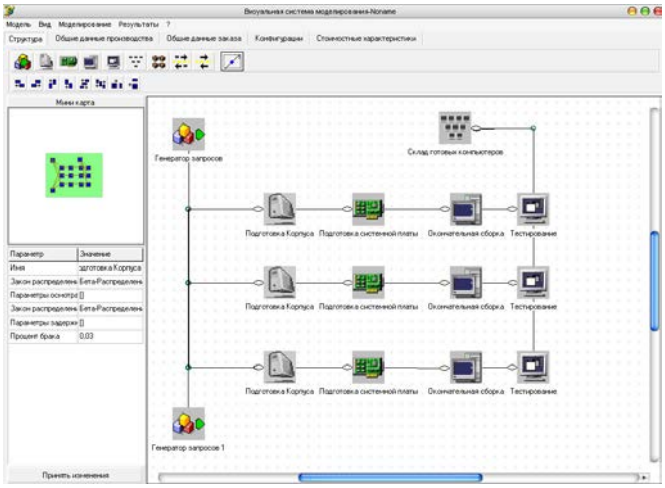

Рис. 4. Построение схемы для модели

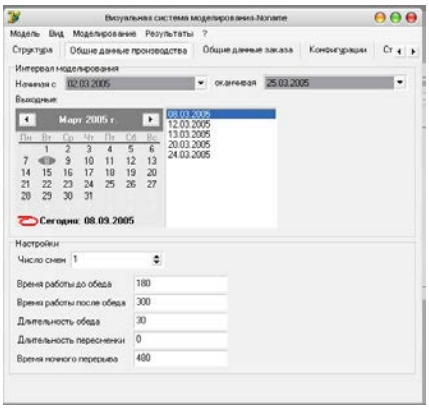

Рис. 5. Задание входных данных производства

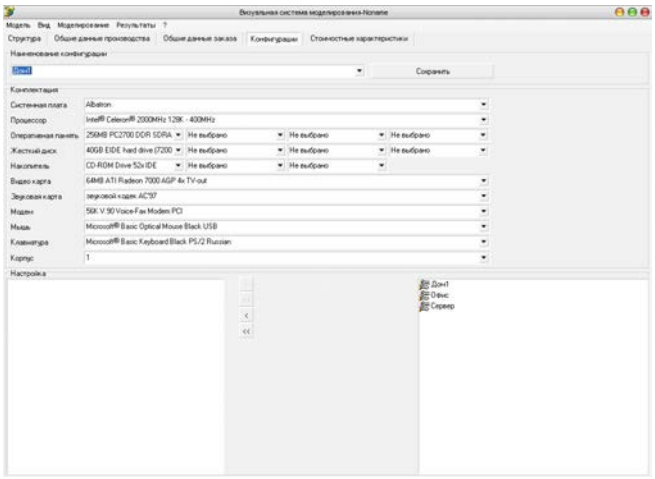

Рис. 6. Создание типовых конфигураций для создания заказов

Одним из главных направлений усовершенствования предполагается введение развитой системы представления результатов в графическом виде. А именно: использование анимации, показывающей динамику процесса моделирования; динамических графиков, которые показывают наглядно изменение значений во времени; представление моделей в 3D.