МОДЕЛЬНЫЙ СИНТЕЗ И МОДЕЛЬНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ КАК МЕТОД САПР ДЛЯ РЕАЛИЗАЦИИ ИМИТАЦИОННЫХ МОДЕЛЕЙ КРУПНОМАСШТАБНЫХ СИСТЕМ

Бродский Ю.И.

Федеральный исследовательский центр «Информатика и управление» РАН,
Россия, г. Москва, ул. Вавилова д.40
yury_brodsky@mail.ru

Аннотация: Предлагается новая парадигма программирования, с более высоким уровнем инкапсуляции, нежели в объектно-ориентированном подходе. Ее ключевые особенности – исключение императивного программирования и ориентированность на распределенные и параллельные вычисления. Предлагаемый подход применим для достаточно широкого круга задач, включая создание имитационных моделей сложных крупномасштабных систем.

Ключевые слова: крупномасштабные системы, высокопроизводительные вычисления, модельный синтез, модельно-ориентированное программирование, имитационное моделирование.

Введение

Исторически смена парадигм программирования сопровождалась укрупнением, агрегированием основного инструмента деятельности программиста. Начиналось все с машинной команды, затем, с появлением языков программирования высокого уровня – таким инструментом стал оператор языка, реализующий некое законченное действие, возможно, с помощью нескольких машинных команд. С

победой идей структурного программирования – на смену отдельным операторам и переменным пришли стандартные конструкции типа «цикл», «ветвление», подпрограммы-функции и структуры данных.

С появлением объектного анализа основной единицей конструирования стал объект, объединяющие некую структуру данных с набором необходимых для их обработки методов. Кроме того, с помощью механизма наследования можно строить иерархии классов объектов, развивающих, конкретизирующих и воплощающих некоторый набор базовых идей, заложенных в корневых классах такой иерархии. Данная парадигма программирования в настоящее время является господствующей и ее базовые понятия, такие как класс, объект, типизация, наследование, инкапсуляция, полиморфизм реализованы с некоторыми нюансами в большинстве современных императивных языков программирования, таких как C++, Java, C#, Delphi и многих других.

Отношение наследования для множества классов объектно-ориентированного языка программирования есть отношение частичного порядка. Классы, не имеющие предков, но обладающие потомками, называются по отношению к ним корневыми или базовыми. Классы, не имеющие потомков, называются листовыми.

Проектирование в объектно-ориентированной парадигме большой программной системы состоит в воплощении базовых понятий и представлений этой системы в базовые классы объектов и построении затем иерархии классов, развивающих, конкретизирующих и воплощающих эти идеи во множестве листовых классов, с помощью которых и будет строиться целевая программная система. Однако все сложности организации вычислительного процесса, состоящего в использовании разработанных и отлаженных методов листовых классов, лежит на разработчике системы: чтобы система что-то делала – необходимо организовать вызов нужных методов нужных объектов в нужной последовательности. Формализации этого процесса не существует – объектно-ориентированное программирование поэтому остается искусством, а не технологией.

1 О сложностях программной реализации моделей крупномасштабных систем

Почему так трудно программировать модель сложной системы?

Потому что она и в самом деле очень сложная!

Попробуем перечислить основные сложности:

1. Сложная структура системы и сложные связи между ее компонентами;
2. Сложная организация данных;
3. Сложное поведение компонент системы:

а) сложная логика поведения;

б) сложная функциональность действий.

Здесь мы выделили несколько типов сложности, а в модели сложной системы они еще все тесно переплетены между собой. Психологи считают, что средний человек уверенно способен оперировать не более чем 4 – 6 объектами одновременно. Если объектов становится больше, человек начинает ошибаться. Быть может, природная одаренность в данной области или же опыт и тренировка способны повысить этот порог с 4 – 6 даже до 15 – 20 объектов. Беда лишь в том, что в сложных крупномасштабных системах приходится оперировать сотнями и тысячами объектов.

2 Модельный синтез и модельно-ориентированное программирование

Можно предложить новый подход к программированию систем, имеющих выраженную многокомпонентную организацию, где уместен синтез целого из составляющих его частей, природа которых, а также способы взаимодействия между собой, известны заранее.

Модельно-ориентированная парадигма программирования предлагает снова увеличить степень агрегирования основной единицы программирования. Предлагается конструировать программную систему из моделей-компонент – сущностей, помимо характеристик и отдельных умений, обладающих собственным поведением, т.е. способных во всех ситуациях, которые могут им встретиться, давать стандартные для них ответы на стандартные запросы внутренней и внешней среды.

В отличие от объекта объектного анализа, методы модели-компоненты не нужно (да и невозможно) вызывать извне. Не нужно заботиться и об организации функционирования модели-компоненты. Модель-компонента функционирует всегда (как всегда функционирует, например, операционная система компьютера) и всегда готова отреагировать заложенным в ее конструкцию способом на происходящие внутри и вне ее изменения, на которые способны реагировать ее методы-события. Поэтому, про внутреннее устройство однажды отлаженной модели-компоненты можно

полностью забыть, используя ее в дальнейших конструкциях как готовый функциональный блок – просто правильно коммутировать входы и выходы – и все будет работать.

Описания моделей-компонент, как математических объектов соответствующего рода структуры, декларативны. Также декларативны описания объединения моделей-компонент в модели-комплексы. Для таких описаний предлагается специальный декларативный язык ЯОКК (язык описания компонент и комплексов). Значительный удельный вес в этом языке имеют операторы коммутации.

Декларативные описания модели-компоненты, вполне определяют ее устройство и логику функционирования, при этом полностью абстрагируясь от содержательной стороны, как действий модели-компоненты, осуществляемых ее методами-элементами, так и причин этих действий, выявляемых методами-событиями.

Модельный синтез и модельно-ориентированное программирование, как методы описания, синтеза и программной реализации имитационных моделей сложных многокомпонентных систем, развивались в отделе Имитационных систем ВЦ АН СССР и затем ВЦ РАН, с конца 80-х гг. Основные их идеи изложены в работах [1-3], а сами термины впервые введены в работе [2].

В качестве основной единицы построения программной системы предлагается модель-компонента – более сложная и агрегированная конструкция, нежели объект объектного анализа. Главное ее отличие от объекта – обладание собственным поведением, в том смысле в каком обладает поведением, например, операционная система компьютера – способностью стандартным, заранее заданным способом отвечать на известный заранее набор внешних и внутренних запросов. При этом оказывается, что способ организации поведения в указанном смысле (а, следовательно, и организации вычислений на компьютере), также может быть стандартным – одинаковым для любой модели-компоненты, сколь бы сложной она ни была.

Предлагаемый подход минималистичен. По существу, в нем нет других понятий, кроме понятия модели-компоненты. Модели-компоненты могут объединяться в модели-комплексы, но модель-комплекс также формально принадлежит семейству моделей-компонент, следовательно, может быть включена в модель-комплекс следующего уровня в качестве компоненты и т.д. Любая модель выполняется по одним и тем же стандартным правилам, т.е., может быть выполнена однажды написанной и отлаженной программой выполнения моделей-компонент. Кроме того, эта программа выполнения моделей-компонент такова, что большинство ее содержательных вычислений допускает распараллеливание, притом, чем сложнее модель – тем больше у нее может быть параллельных процессов.

Вся реализация сложной программной системы распадается, во-первых, на ряд декларативных описаний моделей-компонент и моделей-комплексов объединяющих модели-компоненты, и во-вторых, на программирование некоторых функциональных зависимостей, которые являются функциями в математическом а не только программистском смысле (т.е. однозначны и не имеют состояний и побочных эффектов), и, следовательно могут быть запрограммированы в функциональной парадигме.

Это приводит к тому, что программная реализация даже сверхсложной фрактально устроенной имитационной модели обходится без императивного программирования – самого сложного на любой стадии, начиная с проектирования, до реализации и последующей отладки. Кроме того, про устройство однажды отлаженной компоненты можно навсегда забыть после завершения отладки. Во всех моделях-комплексах, куда она входит, она будет вести себя ровно так, как она умеет себя вести – это всегда обеспечивается автоматически программой выполнения моделей и поэтому включение компоненты в любой комплекс есть лишь вопрос ее правильной коммутации. Все происходит примерно так же, как при включении готовой микросхемы в микросборку и размещении затем этой микросборки на печатной плате. Если при этом на каждом уровне включения коммутация разумна – микросхема будет правильно функционировать в составе гораздо более сложного устройства.

Программный комплекс при этом подходе видится как комплекс моделей-компонент, которые сами могут состоять из компонент более низкого уровня. Программирование состоит в описании устройства и поведения компонент (по сути – в описании соответствующего рода структуры) и в описании построения комплексов из компонент. Для подобных описаний предлагается декларативный язык ЯОКК, подробно описанный в [2]. Описатели ЯОКК компилируются не в машинный код, а в базу данных модели (что снимает вопрос о качестве компиляции – остается лишь вопрос ее правильности). Первым полностью законченным воплощением идей модельного синтеза и модельно-ориентированного программирования можно считать систему MISS [1].

Заключение

Концепции модельного синтеза и модельно-ориентированного программирования позволяют формально описать на языке ЯОКК имеющиеся знания об «атомах» сложной многокомпонентной системы и их связях между собой в комплексы, которые становятся атомами следующего уровня; автоматически по этим описаниям построить синтез модели сложной системы, путем компиляции описателей ЯОКК в базу данных модели; далее остается запрограммировать методы модели (ориентируясь на функциональную парадигму) и заполнить в базе данных начальные значения ее характеристик. После этого модель готова к имитационным экспериментам. Из проекта программной реализации имитационной модели сложной системы удается полностью исключить императивное программирование. Универсальная программа выполнения моделей ориентирована на высокопроизводительные и распределенные вычисления.

Предложенная концепция моделирования сложных систем была воплощена в серии имитационных моделей, реализованных под влиянием модельно-ориентированной парадигмы программирования. Например, моделировались некоторые эпизоды СОИ (стратегическая оборонная инициатива) Рейгана, модель функционирования и взаимодействия нескольких стран. Кроме того, были созданы инструментальная система имитационного моделирования: система MISS [1], и макет пиринговой среды распределенного имитационного моделирования [2], полностью основанные на концепциях модельного синтеза и модельно-ориентированного программирования.

Литература

1. *Бродский Ю.И., Лебедев В.Ю.* Инструментальная система имитации MISS.–М.: ВЦ АН СССР, 1991.–180с.
2. *Бродский Ю.И.* Модельный синтез и модельно-ориентированное программирование. – М.: ВЦ РАН, 2013. – 142 с.
3. *Бродский Ю.И.* Роды структур Н. Бурбаки в задаче синтеза имитационных моделей сложных систем и модельно-ориентированное программирование //Журнал Вычислительной математики и математической физики, 2015, Т. 55, № 1, С. 153–164.