Колесов Юрий Борисович,
Сениченков Юрий Борисович,
Инихов Дмитрий Борисович

«ИХ ЕСТЬ У МЕНЯ!»

Моделирование и исследование сложных динамических систем в MvStudium

На протяжении нескольких последних лет мы неоднократно обращались к теме компьютерного моделирования на примере семейства графических оболочек MvStudim (www.mvstudium.com). Сейчас появилась новая версия пакета – MvStudium 4.1, и можно было бы вновь пойти по проверенному пути – рассказать об изменениях, а их много, и пояснить новые возможности на примерах[1]. Но нам кажется, что сейчас более важно поговорить о новых технологиях проектирования на основе моделирования.

Компьютерное моделирование, в основе которого лежит математическое моделирование, в свое время внесло революционные изменения в процесс исследования окружающего нас мира. Даже если под компьютерным моделированием понимать только процесс автоматического построения математической модели по ее заданному описанию на языке моделирования, обычно графическом, и проведение вычислительного эксперимента с оценкой погрешности результата, то этого уже будет достаточно, чтобы этот термин имел право на жизнь. Можно согласиться и с утверждением, что компьютер – всего лишь новый инструмент в математическом моделировании, породивший новую технологию исследования. Правда, появление этого нового инструмента рождало надежду, что интерес к моделированию возрастет. Видимо, это вечный самообман ученых любых областей (по-человечески понятный и простимый). Как всегда, новое стало повседневным, остались сугубо профессиональные проблемы создания и исследования моделей, не более привлекательные, чем любые другие научные проблемы.

Однако есть область, где компьютерное моделирование играет прагматичную роль – производство. Производство постоянно требует новых инструментов, например, новых технологий проектирования, в которые вовлечены большие коллективы. Компьютерное проектирование на основе модели-

...виртуальные объекты, становятся дешевле и удобнее материальных...

[1] См. краткое описание новой версии в конце статьи.

рования становится таким инструментом и товаром. Производство в буквальном смысле оценило преимущества работы с абстрактной моделью, по сравнению с материальным объектом – прототипом будущего реального устройства. Хоть это и кажется очевидным, виртуальные объекты становятся дешевле и удобнее материальных (при определенном уровне развития вычислительной техники).

На какой-то миг возникает необычная ситуация – руководители производства пытаются купить сулящие выгоду технологии в условиях практически полной неопределенности. Прецедентов нет, риски велики, времени на раздумья в условиях конкуренции нет, и вложенные деньги должны вернуться с прибылью. Лихорадочные времена пройдут, победители станут осмотрительными и скупыми, а ученые опять будут довольствоваться правительственными и благотворительными грантами. Производство прирастет новой отраслью, в данном случае «фабриками», выпускающими все новые и новые инструменты моделирования.

Компьютеры, встроенные процессоры стали частью бытовой техники. Сложность практически любого устройства возросла настолько, что старые технологии проектирования сдерживают производителя. Разработчики уникальных и дорогих устройств уже освоили проектирование на основе моделирования, теперь очередь за всеми остальными. Настает время простых, удобных, достаточно дешевых средств, автоматизирующих процесс построения компьютерных моделей.

...купить технологии..в условиях практически полной неопределенности...

Пример удачного вложения денег есть. Маленькая программа К. Молера, помещавшаяся на дискетку, стала всемирно признанной компьютерной системой MATLAB (MathWorks). Аналогичным примером, из близкой области служит программа LabView (National Instruments).

Успех фирмы MathWorks прост – первая промышленная программа моделирования широкого назначения Simulink представляет собой *библиотеку* компонентов, из которых можно собрать достаточно сложное техническое устройство. Сборка виртуального устройства очень похожа на сборку реального – берешь стандартные компоненты, соединяешь их, подсоединяешь осциллограф и настраиваешь параметры. Позже стало ясно, что это хорошо, но недостаточно. Модель надо воплотить в «железо». Для чрезвычайно распространенного сегодня типа устройств – встроенных систем управления – можно практически автоматизировать все цепочку от моделирования управляемого объекта совместно с системой управления до построения кода, реализующего устройство управления для специальных процессоров. Другим примером являются тренажеры различного типа, в основе которых лежат компьютерные модели. И здесь технология моделирования фактически становится технологией производства конечного продукта.

Существует два способа построения новых библиотечных компонентов.

В первом случае новые компоненты можно строить только из заданного, предопределенного разработчиками пакета компонентов (как в Simulink). Из данных компонентов можно построить функциональную схему – новое устройство. Созданное новое устройство можно оформить как новый компонент. В результате появляются все новые и новые составные компоненты. В Simulink этот процесс не требует практически никаких усилий, если структура нового компонента уже разработана и надо только реализовать ее. Переносишь экземпляры библиотечных компонентов на чистый «лист бумаги», «рисуешь» связи и помещаешь в устройство-контейнер. Средств

создания совсем новых компонентов самим пользователем не предусмотрено. Новые компоненты создаются разработчиками по совсем другой технологии, обычно достаточно сложной с точки зрения рядового пользователя.

Во втором случае пользователь сам может создавать новые устройства-компоненты, а затем, как обычно, агрегировать их в более сложные компоненты. Технология создания новых компонентов (язык моделирования) должна быть проста, понятна и удобна пользователю. Возможности второго способа, несомненно, шире, чем в первом случае, но пользователь должен иметь навыки программирования на языках высокого уровня. И это может мешать распространению таких средств.

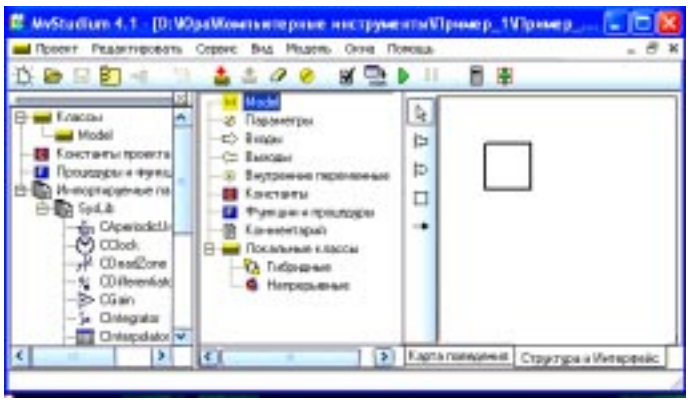


Рис. 1. Менеджер проекта

Казалось бы, что сдерживает применение такой технологии? Почему фирмы-производители тратят огромные средства на пропаганду новых средств проектирования, когда их казалось бы «должны отрывать с руками»? Мы, видимо, имеем дело с тем же эффектом, который имел место при создании систематизированных коллекций и библиотек программных реализаций численных методов в конце прошлого века. Не верил пользователь в то, что они правильно работают. Проверить самому чужие коды уже было невозможно, а верить на слово – страшно. Время сделало свое дело – библиотеки сегодня пишут профессионалы, и мало кто из пользователей пытается разобраться, как они построены. Компьютерное моделирование родилось совсем недавно, университеты еще не успели обучить достаточное число специалистов, не сомневающихся в правильности такого подхода (достаточно просто в процессе обучения не сомневаться, что это правильно).

В MvStudium поддерживается возможность как проектировать с помощью агрегирования, так и создавать новые блоки самому (рис. 1). На рисунке мы видим два окна – окно «Менеджер Проекта» и окно «Редактор Классов» (в данном случае класса Model). В окне «Менеджер Проекта» перечислены строительные блоки – классы проекта. *Класс* – это описание программного компонента, из которого можно автоматически строить экземпляры классов – почти идентичные программные коды, отличающиеся, например, значением параметров, что в нашем случае эквивалентно созданию однотипных устройств с разными характеристиками по единому для всех них описанию. В свою очередь, окно «Редактор Классов» делится на редакторы отдельных элементов класса.

Язык моделирования пакета объектно-ориентированный. Описание всего проекта представляет собой совокупность классов. В данном случае есть один собственный класс Model и набор уже готовых классов из библиотеки SysLib (она поставляется с пакетом). Библиотека Syslib содержит блоки, которые можно найти в пакете Simulink.

Это сделано для тех, кто знаком с пакетом Simulink. Имея такую библиотеку, они могут сразу приступать к делу и создавать новое устройство. Сейчас как раз открыто окно «Структура и Интерфейс» – «чистый лист», на котором уже нарисован образ будущего устройства. Внутрь этого квадрата (предварительно растянув его до нужных размеров) и следует перетаскивать экземпляры библиотечных классов из Менеджера проекта. Щелкнув правой кнопкой на изображении экземпляра использованного библиотечного класса, можно открыть диалог, позволяющий изменить значения параметров, заданных по умолчанию (рис. 2).

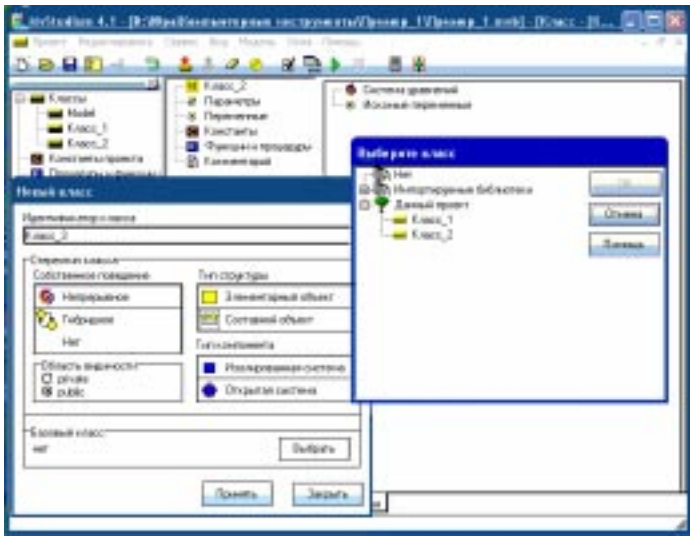


Рис. 2. Похоже на Simulink?

На вертикальной панели окна «Структура и Интерфейс» – инструменты рисования: «широкие» стрелки – образы входов и выходов, тонкая стрелка – «карандаш» для рисования связей. Еще один инструмент – прямоугольник – образ нового компонента, если его тоже придется разрабатывать самому. Таким образом, новый компонент можно разрабатывать «сверху-вниз»: сначала рисуем функциональную схему – на этом и завершится разработка, если все компоненты библиотечные, – а затем разрабатываем совсем новые блоки, если они предусмотрены, точно таким способом.

В любом объектно-ориентированном языке можно, используя механизм наследования, создавать новые классы, дополняя и модифицируя уже существующие (рис. 3).

Практически это делается так. В «Менеджере Проекта» правой кнопкой открываем диалог «Новый класс». Поле «Базовый класс» содержит команду «Выбрать». С помощью этой команды остается только указать базовый класс (рис. 4). В результате можно построить целое дерево, описывающее множество классов, связанных отношением наследования.

Если вы обратили внимание, на рис. 1 и 4 открыты два окна – окно «Менеджера Проекта» и окно «Редактора Классов», в первом случае речь идет о редактировании класса Model, а во втором – Класса_3. Но окна «Редактора Классов» отличаются. В классе Model у окна две закладки «Карта Поведения» и «Структура и Интерфейс», а у окна Редактора Класса_3 одна – «Уравнения». Ответ можно найти на рис. 3 в окне диалога «Новый класс». Там содержится новое понятие – стереотип класса. Сложность класса и сложность порождаемых блоков – экземпляров класса – может быть различна. Меняя стереотип класса, пользователь может менять его сложность, и, что сейчас для нас более важно, более простым классам, соответствуют более простые Редакторы Классов: элементов меньше – меньше кнопок и команд. Стереотип учитывает следующие особенности проектируемых классов: поведение, структуру, связи с другими компонентами.

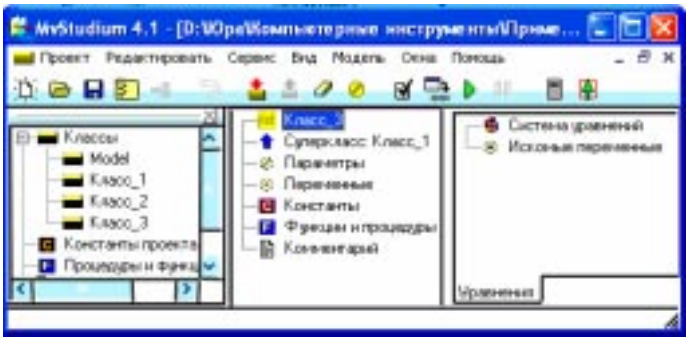
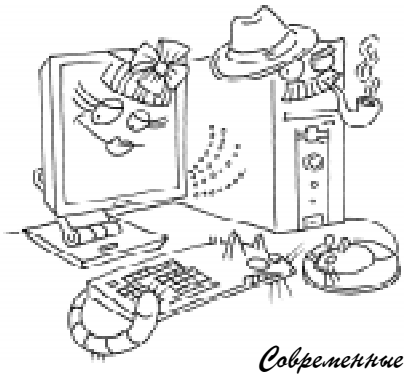


Рис. 3. Использование механизма наследования проектировании новых классов



Современные устройства демонстрируют очень сложное поведение...

ПОВЕДЕНИЕ

Современные устройства демонстрируют очень сложное поведение. Поведение можно описать, прежде всего, как множество режимов или состояний устройства. Самолет может совершать полет под управлением летчика или автопилота. Оба режима включают взлет, полет по заданному курсу, посадку. Эти состояния, (количество состояний и последовательность их смены) одинаковы для всех членов экипажа, но в каждом состоянии активность каждого члена экипажа различна. Для описания структуры поведения лучше всего подходит ориентированный граф, у которого узлами являются состояния, а дуги указывают на возможные новые состояния после окончания текущего режима, состояния и условия перехода в эти состояния. В первом приближении для всего экипажа такой граф будет одним и тем же. Если у нас будет возможность приписывать одинаковым узлам различные активности (виды деятельности) членов экипажа, мы опишем все различные поведения всех членов экипажа. Таким образом, у нас появится граф для командира и граф для стюардессы, своеобразная инструкция всему экипажу на время полета. Исключением является случай графа с одним узлом и одной активностью (ясно, что это чисто теоретическая придумка, потому что в реальной жизни даже ленивый человек имеет два состояния – ест и спит). Для таких объектов можно забыть о структуре поведения, то есть о графе. Нужно описать только активность. В MvStudium под активностью в данном случае понимают поведение классической динамической системы, которую описывают с помощью систем обыкновенных дифференциальных уравнений («непрерывное поведение»). Если

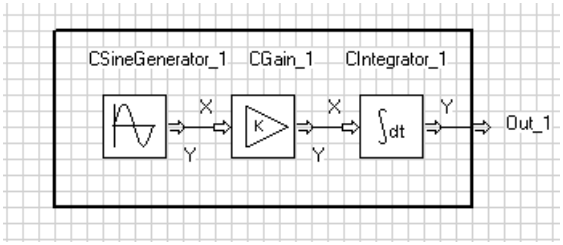


Рис. 4. Новый Класс 3 наследует свойства класса 1

для описания поведения нужен граф, – мы говорим о гибридных системах, то есть математических моделях, обобщающих классическую динамическую систему. Для такой модели каждая активность представляет собой поведение классической динамической системы. Такая система описывается последовательностью классических динамических систем. Граф системы порождает различные последовательности систем дифференциальных уравнений, в зависимости от событий, происходящих в объекте («гибридное поведение»).

СТРУКТУРА

Проектируемый компонент может иметь внутреннюю структуру (составной объект) или не иметь ее (элементарный объект).

СВЯЗИ

Компонент может быть связан с внешним миром («открытая система») или содержать всю необходимую информацию в себе самом и с внешним миром не общаться («изолированная система»).

Совсем скоро будет также учитываться время жизни компонента – экземпляра класса. Компонент можно будет создавать и уничтожать по ходу выполнения модели.

Теперь можно перейти к созданию компонентов с различными стереотипами.

Приложение

КРАТКОЕ ОПИСАНИЕ НОВОЙ ВЕРСИИ MVSTUDIUM

MvStudium [1–3] – это семейство графических оболочек для построения *сложных динамических систем* и их исследования. Сложными считаются системы, чьи математические модели трудно построить сразу в виде, пригодном для их численного иссле-

дования. Их строят автоматически, используя *языки моделирования*. Основной математической моделью MvStudium является *гибридная система*. Гибридные системы рассматриваются как обобщение (классических) динамических систем [5]. Под гибридной системой понимается множество последовательностей алгебро-дифференциальных уравнений, порождаемых *гибридным автоматом*. Элементы любой последовательности, то есть последовательно решаемые системы на заданных временных промежутках с согласованными начальными условиями могут иметь различные наборы искомых переменных и различные размерности.

В графическом языке моделирования Model Vision Language (MVL) сложная динамическая система строится из блоков с *входами-выходами* и/или *контактами,* возможно имеющих свою внутреннюю структуру (функциональную схему) и свое собственное *поведение*, заданное *иерархической картой поведения*. *Стереотип* блока определяется сочетанием следующих характеристик:

- Собственное поведение может быть представлено динамической системой с непрерывным временем, либо гибридной системой, которая описывается картой поведения (B-Chart). Карта поведения или гибридный автомат – это направленный граф, каждому узлу которого приписана динамическая система или гибридный автомат (локальное поведение, *Активность*), а дугам – условия смены поведения и последовательности мгновенных действий, сопровождающих смену поведения.
- Блок может быть изолированным или иметь связи: входы, выходы и контакты . Блоки с входами-выходами называются ориентированными, блоки с контактами – неориентированными, блоки с входами-выходами и контактами – блоками со связями.
- Блоки могут иметь внутреннюю структуру (функциональную схему) – в этом случае они называются составными, – или не иметь ее, то есть быть элементарными.
- Время жизни экземпляра блока может совпадать со временем моделирования, либо экземпляры блоков могут создаваться и уничтожаться по ходу выполнения моделирования.
- Блок может одновременно иметь собственную карту поведения и иметь внутреннюю структуру, заданную функциональной схемой.

MVL – объектно-ориентированный язык. Каждому блоку (компоненту модели) соответствует свой класс. Класс содержит описание блока. Классы могут образовывать деревья с помощью механизма наследования. Совокупность собственных классов, классов, импортируемых из библиотек и других проектов, вместе с обязательным для всех проектов классом Model образуют Проект, соответствующий понятию пакета (UML). Экземпляры классов могут использоваться для построения функциональных схем и выступать в качестве активностей в картах поведения. Любой класс проекта может иметь свои константы, параметры и переменные, алгоритмические функции и процедуры, а также локальные классы – непрерывные (динамические системы) и гибридные (гибридные системы). Класс Model считается глобальным, и его единственный экземпляр становится в конце концов исполняемым модулем. В локальных классах видны переменные и функции непосредственно охватывающих их классов. Переменные классов могут иметь скалярные типы, свойственные традиционным алгоритмическим языкам, могут быть векторами и матрицами с фиксированными или динамически меняющимися размерами, сигналами, записями, динамическими переменными, создаваемыми и уничтожаемыми с помощью операторов new и delete.

С точки зрения объектно-ориентированного моделирования, MvStudium поддерживает понятие *Активного Динамического Объекта* [4], свойства и поведение которого *непрерывно* меняются в независимом времени на протяжении всего времени жизни объекта под воздействием возникающих внутренних или внешних событий. Программные часы, моделирующие течение локального или глобального времени и упорядочивающие все значения переменных и все события, являются неотъемлемой частью

любого активного динамического объекта.

Программная компонента, обеспечивающая создание моделей в MvStudium, называется Редактором моделей. Созданные модели автоматически переводятся из графического представления в текстовое (MVL представление), а затем на язык Delphi и компилируются. Окончательный код может работать под управлением оболочки (контролируемая оболочкой визуальная модель обычно используется при отладке), независимого приложения (независимой визуальной модели) и DLL.

Отладка, исследование, активный вычислительный эксперимент проводятся на *Виртуальном Испытательном Стенде* – второй программной компоненте MvStudium. Виртуальный испытательный стенд содержит:

– средства отладки (планировщик точек приостановки и калькулятор, работающий как с переменными модели, так и с виртуальными переменными, не описанными в модели, но представленными в виде выражений над существующими переменными);

– редакторы двумерной и трехмерной (на основе OpenGL) анимации, средства построения графиков, гистограмм, фазовых диаграмм;

– средства оптимизации с библиотекой численных методов для поиска локальных и глобальных экстремумов;

– библиотеку численных методов для решения алгебраических, дифференциальных, алгебро-дифференциальных уравнений. Метод для решения уравнений с запаздыванием предопределен разработчиками.

MvStudium (www.mvstudim.com) широко используется в учебном процессе, научной работе и при разработке промышленных систем.

Литературра

1. *Колесов Ю.Б., Сениченков Ю.Б.* Моделирование систем. Динамические и гибридные системы. СПб.: БХВ, 2006. 224 с.

2. *Колесов Ю.Б., Сениченков Ю.Б.* Моделирование систем. Объектно-ориентированный подход. СПб.: БХВ, 2006. 192 с.

3. *Колесов Ю.Б., Сениченков Ю.Б.* Моделирование систем. Практикум по компьютерному моделированию. СПб.: БХВ, 2007. 352 с.

(Трехтомное издание «Моделирующие системы» представляе ттттт собой справочник и дополнено двумя монографиями. Все это можно найти на сайте www.mvstudium.com

4. *Колесов Ю.Б., Сениченков Ю.Б.* Объектно-ориентированное моделирование сложных динамических систем. СПб.: Изд-во Политехнического университета, 2004. 239 с.

5. *Колесов Ю.Б., Сениченков Ю.Б.* Моделирование систем. Численное моделирование гибридных систем. СПб.: Изд-во Политехнического университета, 2004. 206 с.

Колесов Юрий Борисович,
доктор технических наук,
профессор кафедры РВКС
факультета технической
кибернетики СПбГПУ,

Сениченков Юрий Борисович,
доктор технических наук,
профессор кафедры РВКС
факультета технической
кибернетики СПбГПУ,

Инихов Дмитрий Борисович,
зам. директора фирмы «MvSoft».