УДК 519.713; 621.785.05

АРХИТЕКТУРА ПОСТРОЕНИЯ ПРОГРАММНЫХ КОМПЛЕКСОВ ДЛЯ МОДЕЛИРОВАНИЯ ТЕХНОЛОГИЧЕСКИХ СИСТЕМ НА ОСНОВЕ ДИНАМИЧЕСКИ ЗАМЕНЯЕМЫХ МОДУЛЕЙ

д-р техн. наук, проф. С.П. КУНДАС
(Международный государственный экологический университет им. А.Д. Сахарова, Минск),
Д.Г. ИВАНОВ, А.В. ЛЕМЗИКОВ, В.И. КОВАЛЕНКО
(Белорусский государственный университет информатики и радиоэлектроники, Минск)

Рассмотрены вопросы, связанные с разработкой структуры программного комплекса, основанного на динамически заменяемых модулях. Описаны достоинства и недостатки данного подхода, приведен пример подобной структуры для двух программных комплексов: для управления процессом закалки и для моделирования процесса плазменного напыления.

С развитием вычислительной техники и информационных технологий, разработчику программного обеспечения (ПО) необходимо работать со все более сложными системами, к которым можно отнести и программные комплексы для компьютерного моделирования технологических систем, реализующие многоуровневые трехмерные математические модели и алгоритмы с большим количеством входных и выходных данных [1].

В качестве примера рассмотрим программный комплекс для моделирования процессов термообработки, который должен реализовать следующие функции [1]:

- решение прямой задачи моделирования - нахождение выходных характеристик моделируемого объекта (напряженно-деформируемое состояние, распределение твердости и др.), исходя из входных параметров технологического процесса (температура, скорость охлаждения и др.);
- решение обратной задачи - определение входных параметров технологического процесса, при заданном наборе требуемых выходных параметров объекта моделирования;
- компьютерное управление сопрягаемым физическим объектом - технологической установкой закалки.

Приведенное перечисление функций системы выделяет ее составные части на самом высоком уровне абстракции. Для реализации системы необходима ее декомпозиция (например, объектно-ориентированная) на более низких уровнях. В том случае если система или подсистема является сложной, то в процессе объектно-ориентированной декомпозиции вероятно появление совокупности классов, обладающих большой связностью между собой и небольшой - со всей системой в целом [2]. Обычно эти классы составляют функционально-законченный модуль. Такие модули можно рассматривать как отдельные подсистемы со своей структурой, границы которых представлены в виде определенных архитектором ПО интерфейсов. В большинстве случаев возможно множество вариантов реализации той или иной подсистемы, каждый из которых обладает своими плюсами и минусами, поэтому важно предусмотреть механизм динамической замены подсистемы на ее аналог без перекомпиляции системы в целом. Это требование и способ его реализации тесно переплетается с выбранной платформой. Большинство операционных систем (ОС) поддерживает механизм динамической загрузки и отображения в адресное пространство библиотечных подпрограмм - DLL (Dynamic Link Library), что дает возможность реализации на его основе так называемых плагинов (от англ, plug-in - съемный, сменный). Плагины позволяют с успехом решать задачу комплектования комплекса модулями, отвечающими требованиям пользователя как с точки зрения функциональности, так и стоимости. Более того, как показывает практика, четкое определение и документирование интерфейсов плагинов позволяет развивать систему сторонним разработчикам, адаптировать ее для решения разнообразных задач. В рассматриваемом комплексе для моделирования процессов термообработки на роль реализации в виде плагинов идеально подходят следующие подсистемы:

- (mesher) мешер - модуль разбития геометрии детали на конечные элементы;
- модуль решения прямой задачи;
- модуль решения обратной задачи;
- модуль управления технологическим оборудованием.

Как видно, все модули, кроме мешера, соответствуют функциональной декомпозиции системы. Несмотря на то, что каждая из выделенных подсистем обладает, на первый взгляд, некоторой самостоятельностью, все они нуждаются в головной программе, служащей для осуществления взаимодействия, организации входных и интерпретации выходных данных плагинов, а также реализации части программного кода, не выполняющего непосредственно указанные задачи, но необходимого для организации взаимодействия с пользователем и базой данных.

Подключаемые модули реализуют различную функциональность. Головная программа производит их вызов посредством предоставляемых интерфейсов. В зависимости от используемого интерфейса головная программа помещает каждый модуль в соответствующий список и позволяет пользователю выбрать требуемый в каждом конкретном случае. С помощью диаграммы компонентов языка UML на рис, I показаны основные заменяемые модули:

- mesher.dll - модуль мешера;
- processor.dll - модуль решения прямой или обратной задачи;
- hardware.dll - модуль управления технологическим оборудованием.

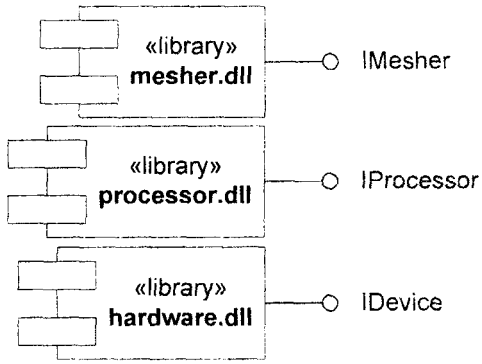

Рис. 1. Основные заменяемые модули

Рассмотрим особенности названных подсистем и их интерфейсов.

Мешер - сложная подсистема, оперирующая большими массивами данных, требующая больших вычислительных ресурсов. При решении задач моделирования возможно использование различных видов конечных элементов - в каждом плагине может быть реализован свой алгоритм, генерирующий тот или иной вид разбиения. Плагины мешера могут иметь возможность параллельных вычислений на мультипроцессорных системах. В настоящее время алгоритмы разбития на конечные элементы быстро развиваются, поэтому динамическая замена этого модуля особенно актуальна. Определение унифицированного интерфейса мешера не представляет особой сложности. Исходные данные могут быть представлены в формате STL (описание поверхности детали в виде совокупности треугольников). Результат работы - в виде множества полученных узлов, их номеров, номеров конечных элементов и перечисления узлов для каждого элемента. Так как в процессе работы модуль требует большого количества вычислительных ресурсов и его работа будет осуществляться в отдельном потоке, важно предусмотреть функцию интерфейса мешера, которая возвращала бы текущий статус вычислений.

Модуль решения прямой задачи наиболее эффективно реализуется на основе метода конечных элементов. Характеризуется большим количеством входных и выходных данных. Кроме того, в зависимости от математической модели наборы входных и выходных данных у каждого плагина могут отличаться. Это делает проблематичным реализацию плагина этой подсистемы с точки зрения обеспечения унифицированного интерфейса. Различие данных на границе плагина делает невозможным подготовку входных и интерпретацию выходных данных со стороны головной программы. Очевидным решением этой проблемы является перенос части функции подготовки и интерпретации данных на функциональность плагина. Плагин должен обладать метаинформацией о своих входных данных, четко определяющей ее формат и назначение. Задачей головной программы в таком случае является построение диалоговых форм пользователя в соответствии с требуемыми данными. Перенос части функциональности интерпретации выходных данных может выглядеть следующим образом: плагину передается динамическая информация о подсистеме визуализации (ссылка на интерфейсный объект), используя которую ведется передача головной программе информации-результата. Так же как и в случае мешера, необходима функция, возвращающая текущий статус расчета.

Модуль решения обратной задачи представляет собой обученную нейронную сеть, на вход которой подается информация о требуемом наборе выходных параметров детали, а на выходе формируется информация о совокупности входных параметров технологического процесса. Так как эти наборы данных могут отличаться по своей структуре, в организации унифицированного интерфейса возникают проблемы, аналогичные модулю решения прямой задачи, а значит и интерфейс должен организовываться схожим образом. Плагины этой подсистемы могут отличаться структурой используемой нейронной сети.

Модуль управления технологическим оборудованием - реализует в себе зависимую от конкретной технологической установки и устройств сопряжения часть кода, обеспечивая тем самым некоторое абстрактное представление аппаратуры для программных средств. В интерфейсе необходимо преду-

смотреть функции, которые позволили бы в любой момент времени знать состояние технологической установки (датчики), а также непосредственно воздействовать на нее с помощью активных механизмов (устройств управления).

Реализованная таким образом декомпозиция позволяет упростить понимание архитектуры проектируемой сложной системы, а также разбить ее на отдельно реализуемые функциональные модули. Все это дает возможность анализа взаимодействия между модулями еще на этапе проектирования архитектуры, выявить места, требующие оптимизации ввиду больших потоков информации, а также досконально понять назначение и функции каждого блока.

Каждый модуль выполняется в виде отдельной динамически подключаемой библиотеки, что позволяет формировать функциональность всего комплекса в целом путем простой замены этих библиотек.

Описанную архитектуру наиболее удобно реализовать используя понятие интерфейса, которое поддерживается современными объектно-ориентированными языками (C#, Java). Используя наследование интерфейсов (рис. 2), архитектор определяет взаимосвязи типа «общее - частное» для всей системы в целом.

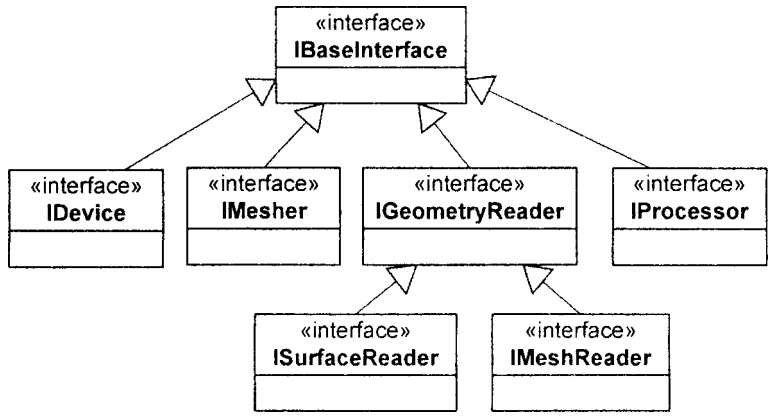

Рис. 2. Отношения наследования интерфейсов заменяемых модулей

Использована диаграмма классов языка UML (см. рис. 2). Выделены следующие интерфейсы заменяемых модулей системы:

- IBaselnterface - базовый интерфейс модулей. Определяет функциональность заменяемого модуля в целом;
- IDevice - интерфейс, определяющий набор функций, необходимых для управления устройством (технологической установкой);
- IMesher - интерфейс, определяющий функции, необходимые для реализации в модулях мешера;
- IGeometryReader (ISurfaceReader, IMeshReader) - интерфейсы классов чтения геометрии (поверхностной и объемной);
- IProcessor - интерфейс модуля решения обратной задачи.

Каждому интерфейсу необязательно соответствует заменяемый модуль. Так, например, интерфейсы чтения геометрии могут быть реализованы в модуле мешера.

Однако такой способ требует создания дополнительного программного обеспечения для работы с модулями, предоставления им различных услуг, связанных с интерфейсом пользователя, доступом к базам данных, оборудованию и т.п. Кроме того, возникает необходимость в разработке значительного количества интерфейсов, которые будут реализовываться в модулях. Тем не менее полученный программный комплекс будет обладать большой универсальностью с точки зрения последующей доработки, расширения и распространения.

Как развитие описанного подхода можно предложить провести декомпозицию сразу нескольких программных комплексов, моделирующих различные технологические процессы. В частности, кроме упомянутого выше комплекса для управления процессом закалки можно разбить на модули программный комплекс для моделирования процессов плазменного нанесения покрытий PLASMA [3]. В нем для каждой реализованной модели можно выделить следующие модули:

- модуль ввода информации;
- модуль доступа к базе данных;
- модуль математических вычислений;
- модуль отображения результатов.

Аналогичную декомпозицию можно провести для любого программного комплекса, что позволяет предложить универсальную структуру, позволяющую реализовать широкий набор моделей и средств работы с ними (рис. 3), организовать совместную работу различных математических моделей в единой программной оболочке.

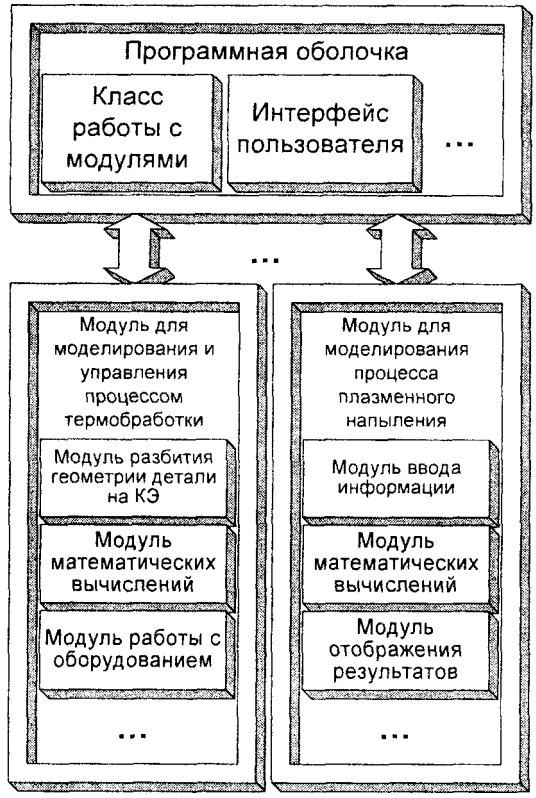

Рис. 3. Взаимодействие программной оболочки с различными подключаемыми модулями

Программная оболочка осуществляет поиск и подключение установленных программных модулей и предоставляет им средства для обеспечения интерфейса пользователя. Программные модули в свою очередь осуществляют подключение требуемых им библиотек с соответствующими функциями и организуют взаимодействие между ними.

Таким образом, приведенные технологии позволяют создать программные комплексы, которые легко адаптируются под аналогичные задачи, так как их основные модули создаются в виде отдельных библиотек, замена которых другими, выполняющими требуемые функции с помощью иных алгоритмов, но обладающих тем же интерфейсом, не требует перекомпиляции программного комплекса в целом.

ЛИТЕРАТУРА

1. Кундас С.П., Кашко Т.А. Компьютерное моделирование технологических систем: Учеб. пособие. - Мн.: БГУИР, 2002. 4.1. - 240 с.
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. - 2-е изд.: Пер с англ. - М.: Бином, 2000. - 560 с.
3. Program facilities for integrated simulation of coating plasma spraying / S. Kundas, V. Gurevich, S. Levashkevich, I. Smurov, M. Ignatiev // Plasma Physics and Plasma Technology: Proc. of III Intern. Conf. - Minsk, 2000. - V. 2. - P. 612 - 615.