УДК 519.6

ПРЕПОСТПРОЦЕССОР ЛОГОС-ПРЕПОСТ. АРХИТЕКТУРА УРОВНЯ БИЗНЕС-ЛОГИКИ, ХРАНЕНИЕ, ИМПОРТ И ЭКСПОРТ ДАННЫХ

А. А. Анищенко, В. И. Дерюгин, В. Н. Дюпин, К. В. Иванов,
А. С. Санталов, Е. Е. Санталова
(РФЯЦ-ВНИИЭФ, г. Саров)

Дается описание архитектуры уровня бизнес-логики программного приложения ЛОГОС-Препост, входящего в состав программного комплекса имитационного моделирования ЛОГОС. Описано взаимодействие уровня бизнес-логики с уровнями представления и данных, освещены вопросы хранения, импорта и экспорта данных. Перечислены решения, делающие приложение гибким, настраиваемым и легко модифицируемым.

Ключевые слова: препостпроцессор, ЛОГОС, архитектура, бизнес-логика, импорт данных, экспорт данных.

Введение

Программное приложение ЛОГОС-Препост входит в состав многофункционального программного комплекса ЛОГОС для полномасштабного комплексного имитационного моделирования на суперЭВМ с массовым параллелизмом, созданного в РФЯЦ-ВНИИЭФ в рамках проекта "Развитие суперкомпьютеров и грид-технологий". Программный комплекс ЛОГОС состоит из отечественных пакетов программ, включающих в себя базовые модели и методы для имитационного моделирования на суперЭВМ широкого спектра физических процессов, характерных для практических трехмерных задач промышленных предприятий.

ЛОГОС-Препост обеспечивает программный комплекс современными средствами предварительной и постобработки. В данной статье описана архитектура уровня бизнес-логики данного приложения, а также освещены вопросы хранения, импорта и экспорта данных.

Модель взаимодействия данных в системе ЛОГОС-Препост

Концепция уровней (layers) — одна из распространенных моделей, используемых разработчиками программного обеспечения для разделения сложных систем на более простые части. При проектировании систем реализуется многоуровневая модель взаимодействия данных.

Наиболее часто применяется архитектура с тремя основными уровнями: 1) представление (presentation); 2) домен или предметная область, бизнес-логика (domain); 3) данные (data). Уровень представления охватывает все, что имеет отношение к общению пользователя с системой. Уровень данных обеспечивает хранение и доступ к обрабатываемым данным. Уровень бизнес-логики реализует основную функциональность системы и обеспечивает взаимодействие между данными и представлением.

Рассмотрим указанную модель применительно к системе ЛОГОС-Препост.

Уровень данных составляют основное хранилище данных в оперативной памяти системы, входные и выходные файлы, а также модули, обеспечивающие загрузку данных из файлов в хранилище и выгрузку обратно.

На уровне представления находятся такие компоненты, как модуль визуализации, графический интерфейс пользователя, макроязык.

Уровень бизнес-логики образуют модуль конфигурации системы, библиотеки операций и объектов бизнес-логики [1].

Архитектура уровня бизнес-логики

Уровень бизнес-логики оперирует двумя ключевыми понятиями:

- *объект* — набор данных, представляющий собой логически выделенную единицу с точки зрения предметной области;
- *операция* — выделенная с точки зрения предметной области законченная последовательность действий.

Объекты бизнес-логики представляют собой сущности, которые описывают модель. Такими сущностями в данном случае являются материалы, регионы, граничные условия и др. С помощью конфигурационных файлов в системе описываются типы объектов бизнес-логики, задается набор параметров, которым обладает объект данного типа. Также можно задать условия видимости и доступности этих параметров, ограничения на число определенных объектов. Можно ограничить набор возможных значений параметров, задать зависимость одного параметра от другого для того или иного объекта.

При работе в системе пользователь вызывает операции, которые осуществляют те или иные действия над объектами бизнес-логики. Примеры таких операций: добавление, модификация или удаление экземпляра объекта, модификация сетки модели и др. Операциями оформляются все законченные последовательности действий над данными и объектами бизнес-логики, влекущие за собой их изменение.

Всю метаинформацию о системе, используемую на уровне бизнес-логики, содержит модуль конфигурации ЛОГОС-Препост. Эта информация включает описания структуры хранилища, объектов, операций, моделей и многое другое.

При старте системы производится разбор конфигурационных файлов, которые в текстовом виде содержат всю необходимую информацию. Информация размещается в оперативной памяти и доступна для использования другими компонентами системы посредством специализированного API.

Модуль конфигурации позволяет обеспечить подключение к системе плагинов (динамически загружаемых библиотек). Например, в виде плагинов оформлены операции бизнес-логики. С помощью описания в текстовом конфигурационном файле производится регистрация операции в системе. При этом задаются ее имя, список параметров, указывается функция, экспортируемая из библиотеки *.dll, которая реализует логику операции. После разбора конфигурационных файлов производится загрузка исполняемых кодов операций в контроллер операций, и они становятся доступны для ядра системы без перекомпиляции исходного кода системы.

Взаимодействие уровня бизнес-логики с уровнем представления

Доступ к операциям пользователь может получить посредством компонентов ЛОГОС-Препост, находящихся на уровне представления: графического пользовательского интерфейса или макроязыка.

Все модификации данных, производимые благодаря пользовательскому интерфейсу, выполняются с помощью вызовов операций бизнес-логики. Вызов операций осуществляет контроллер операций.

В ядро ЛОГОС-Препост встроен интерпретатор Python [2], который с помощью специальных библиотек-переходников может взаимодействовать с контроллером операций.

Взаимодействие уровня бизнес-логики с уровнем данных. Импорт и экспорт данных

Уровень данных отвечает за хранение и предоставление данных системе. Выбор способа хранения данных играет важную роль в быстродействии и расширяемости системы.

Как отмечалось выше, на уровне данных системы ЛОГОС-Препост находится специальное хранилище данных счетной модели: расчетных сеток, наборов сеточных элементов, заданных на этих наборах начальных и граничных условий и т. д. Информация о структуре хранилища содержится в конфигурационных файлах и доступна на уровне бизнес-логики через API модуля конфигурации. Эта информация используется операциями бизнес-логики для доступа к данным хранилища.

На рисунке представлена схема взаимодействия уровня бизнес-логики с уровнем данных.

Препостпроцессор ЛОГОС-Препост разрабатывался параллельно с развитием программных пакетов имитационного моделирования, поэтому важно было обеспечить возможность расширения структуры хранилища в соответствии с

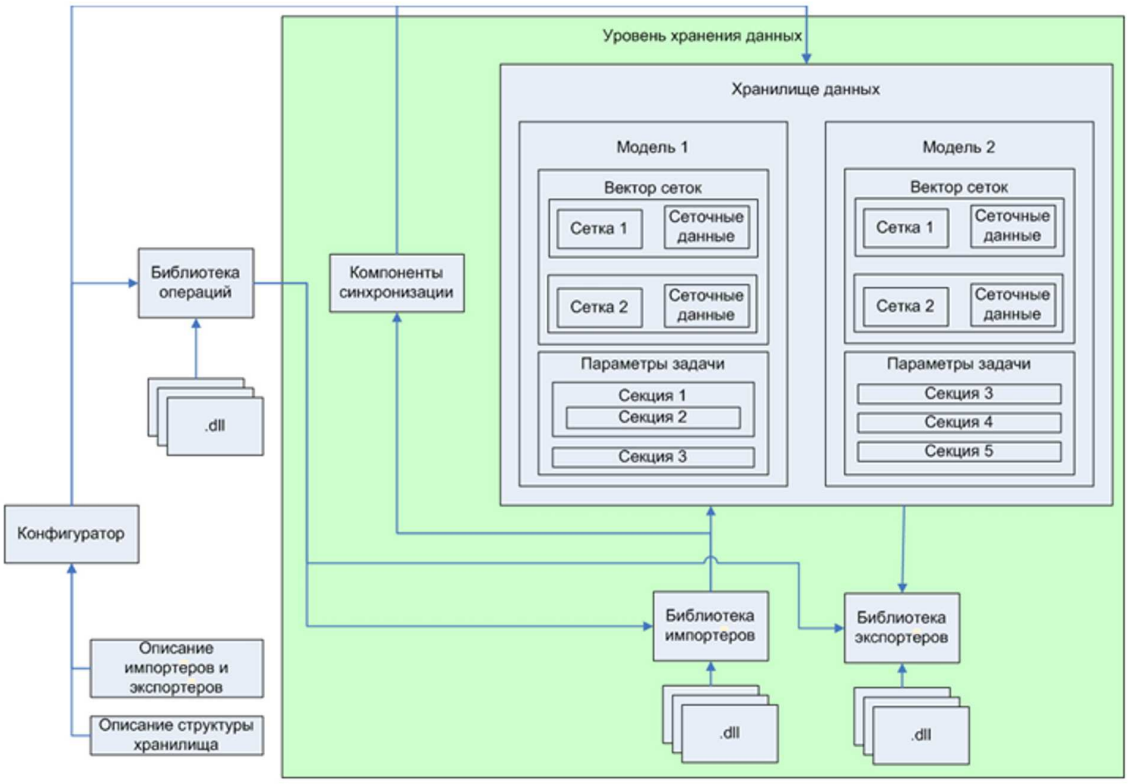

Схема взаимодействия уровня бизнес-логики с уровнем данных

изменяющимися требованиями [3]. Кроме того, необходимо было поддержать иерархические структуры данных и использование библиотеки VTK [4].

Исходя из этого, в качестве базовой структуры хранилища была взята структура DataSet из библиотеки VTK, представляющая собой *таблицу таблиц* (т. е. значением ячейки таблицы может быть другая таблица). На основе ее реализации был создан класс с дополнительными возможностями для удобного и унифицированного доступа к данным.

Отличительной особенностью универсального хранилища ЛОГОС-Препост является возможность его настройки при помощи конфигурационных файлов без перекомпиляции исходного кода. Конфигурационные файлы позволяют описать структуру и типы данных, которые будут располагаться в хранилище.

Каждая задача, моделируемая в ЛОГОС-Препост, представлена в памяти блоком данных, который поделен на несколько разделов. Основными из них являются *вектор сеток* и параметры задачи — данные, связанные с сетками, но не относящиеся к какому-либо их конкретному экземпляру.

Каждый экземпляр в векторе сеток хранит непосредственно сетку в формате VTK и данные, которые с ней связаны: наборы узлов, граней, ячеек и т. д. Эти данные делятся на два раздела: с фиксированной и произвольной структурой. Данные с произвольной структурой описываются в конфигурационном файле секцией `MeshData` (сеточные данные).

Раздел параметров задачи имеет свободную структуру, которая описывается в конфигурационном файле секцией `CommonData` (общие данные).

Для упрощения связи с подсистемой визуализации сетка в хранилище записывается в формате VTK, который обеспечивает ячеечно-узловое представление. Это потребовало от программ-импортеров конвертирования загружаемых файлов из исходного гране-реберного представления в ячеечно-узловое. Для обеспечения возможности обратного преобразования разработана секция для хранения данных гране-реберного формата `Facet Notion`. Она предназначена для дополнения структуры `vtkUnstructuredGrid` с це-

лью обеспечения хранения информации о двусторонних связях ячеек и граней.

Конфигуратор позволяет определять иерархию пользовательских секций хранилища в конфигурационных файлах системы ЛОГОС-Препост. Это можно сделать, задавая в описании пользовательской секции хранилища другую секцию в качестве параметра. При разборе конфигурационных файлов конфигуратором формируется метаинформация об иерархии пользовательских секций, в соответствии с которой в дальнейшем происходит работа с этими секциями.

Подсистема импорта и экспорта обеспечивает возможность использования форматов коммерческих программных пакетов имитационного моделирования, таких как Ansys [5], Nastran [6], LS-PrePost, Abaqus [7], Star-CD и др. С помощью конфигурационных файлов подсистема настраивается на каждую модель и обеспечивает тем самым работу с необходимыми форматами файлов.

Расширение набора программ — импортеров и экспортеров осуществляется также с помощью конфигурационных файлов. Для этого необходимо поместить код процедуры, реализующей операцию импорта или экспорта, в динамически загружаемую библиотеку и обеспечить ее экспорт. Затем нужно зарегистрировать новый импортер/экспортер с помощью конфигурационного файла системы и подключить его в конфигурационном файле модели. После этого новым импортером/экспортером можно будет пользоваться без перекомпиляции исходного кода системы.

Связующее звено между уровнями бизнес-логики и данных в ЛОГОС-Препост составляют компоненты синхронизации данных. Уровень бизнес-логики позволяет настраивать каждую модель на использование или неиспользование тех или иных компонентов: синхронизации данных при выполнении операций импорта-экспорта и модификации наборов данных, операций по формированию гране-реберного представления и формата хранения наборов граней, операций-триггеров, срабатывающих при изменении данных объектов.

Компонент синхронизации данных при выполнении операций импорта-экспорта приводит блоки параметров — граничные условия, регионы, подобласти и др. — к виду, который требуют загруженные данные.

Операции по формированию гране-реберного представления и формата хранения наборов граней конвертируют наборы граней во внутреннюю структуру для экономии памяти во время работы системы.

Операции-триггеры, срабатывающие при изменении данных объектов, предназначены для синхронизации параметров между собой и поддержания хранилища в актуальном целостном состоянии.

Программные пакеты имитационного моделирования принимают параметры задачи в виде файла, записанного согласно синтаксису YAML [8]. Но в зависимости от счетной методики существуют различия в форматировании сохраняемых данных. Использовать для каждого пакета моделирования свой импортер/экспортер параметров задачи накладно, поэтому разработан импортер/экспортер, который "не привязан" к определенной структуре хранилища и позволяет сохранять данные с различными вариантами форматирования. В дальнейшем этот импортер/экспортер можно будет использовать при добавлении новых счетных методик.

Реализованная универсальная настраиваемая система импорта и экспорта параметров задачи позволяет осуществлять импорт и экспорт данных из пользовательских секций. При необходимости с помощью конфигурационных файлов можно настроить формат вывода секций и параметров: схему конвертации имен параметров и секций, игнорирование (или нет) служебных секций, метод записи перечислений, формат записи секций (вертикальный или горизонтальный), условие записи параметра или секции и др.

Заключение

Архитектурные решения, принятые на уровне бизнес-логики ЛОГОС-Препост, позволили:

1) перенести значительную часть реализации функционирования системы из кода на C++ в конфигурационные файлы. Это дает возможность самому пользователю осуществлять настройки системы без перекомпиляции кода и, в конечном итоге, увеличивает гибкость и надежность системы;
2) реализовать концепцию плагинов, что актуально с точки зрения расширения функциональности системы, в том числе с привлечением сторонних разработчиков;

3) реализовать динамически расширяемый интерфейс на уровне макрокоманд.

Опыт эксплуатации ЛОГОС-Препост в РФЯЦ-ВНИИЭФ и на других предприятиях показал эффективность описанных выше решений. Так, например, подключить новую счетную методику в случае уже имеющихся подходящих импортеров/экспортеров можно без программирования на C++. При отсутствии нужных импортеров/экспортеров достаточно их запрограммировать и разместить модули в отдельных dll-файлах без внесения изменений в уже отлаженные и оттестированные коды других компонентов.

Такие возможности минимизируют не только трудозатраты на развитие и сопровождение, но и вероятность внесения в программный код новых ошибок.

Список литературы

1. *Иванов К. В., Анищенко А. А.* ЛОГОС-Препост. Архитектурные решения на уровне бизнес-логики // XIV Межд. конф. "Супервычисления и математическое моделирование". Тезисы. Саров, 1—5 октября 2012 г. Саров: РФЯЦ-ВНИИЭФ, 2013. С. 329—333.
2. Python. http://www.pyton.org/
3. *Дерюгин В. И., Дюпин В. Н., Иванов К. В., Санталов А. С.* ЛОГОС-Препост. Форматы и структуры данных // XIV Межд. конф. "Супервычисления и математическое моделирование". Тезисы. Саров, 1—5 октября 2012 г. Саров: РФЯЦ-ВНИИЭФ, 2013. С. 222—225.
4. VTK — The Visualization Toolkit. http://www.vtk.org/
5. ANSYS — Simulation Driven Product Development. http://www.ansys.com.
6. MSC Nastran — Industry Leading Multidisciplinary FEA — MSC Software. http://www.mscsoftware.com/product/msc-nastran.
7. Abaqus Overview — Dassault Systemes. http://www.3ds.com/products-services/simulia/portfolio/abaqus/overview.
8. The Official YAML Web Site. http://yaml.org/

Статья поступила в редакцию 06.06.13.