2. Трубецков Д. И., Мчеллова Е. С., Красичков Л. В. Введение в теорию самоорганизации открытых систем. М.: Физматлит, 2005.

3. Храмченков М. Г. Элементы физико-химической механики природных пористых сред. Казань: Казан. Мат. общ-во, 2003.

4. Киссин И. Г. Флюиды в земной коре: геофизический и тектонический аспекты. М.: Наука, 2009.

5. Астанин С. А., Колобов А. В., Лобанов А. И. и др. Влияние пространственной гетерогенности среды на рост и инвазию опухоли. Анализ методами математического моделирования // Медицина в зеркале информатики. 2008. С. 188–223.

ДИНАМИЧЕСКОЕ КОНФИГУРИРОВАНИЕ ПАРАМЕТРОВ УНИВЕРСАЛЬНОГО ПРЕПРОЦЕССОРА С ПОМОЩЬЮ ЯЗЫКА XML SCHEMA ДЛЯ ПОДГОТОВКИ ИСХОДНЫХ ДАННЫХ ЗАДАЧ МОДЕЛИРОВАНИЯ НА СУПЕРКОМПЬЮТЕРАХ

А. Д. Черевань, А. Г. Надуев, Д. А. Кожаев

ООО «Центр компетенций и обучения», г. Саров

Введение. Постановка задачи

Современные комплексы математического моделирования включают в себя модули подготовки входных данных (препроцессор), модули, выполняющие расчет модели, в том числе в параллельном режиме с применением суперкомпьютеров, и модули визуализации и анализа полученных результатов. К модулям подготовки входных данных предъявляются разносторонние требования. Они должны обеспечивать прием данных в различных промышленных и отраслевых форматах, иметь возможность первичной проверки непротиворечивости данных, обладать возможностью эффективного задания или изменения параметров модели, начальных и граничных условий и графического представления данных. Немаловажной является также возможность быстрой адаптации как препроцессора, так и счетного модуля при модификации или создании новых расчетных методик. Более того, при наличии быстро развивающейся счетной методики, именно необходимость постоянной программной адаптации препроцессора и счетного модуля к существенным изменениям структуры данных на протяжении всего развития проекта является основным сдерживающим фактором. Подобная адаптация, как правило, приводит к существенной переработке всех частей комплекса, поскольку затрагивает базовую структуру данных, которая связывает все модули комплекса. В связи с этим возникла необходимость разработать систему динамического конфигурирования комплекса, которая позволила бы без значительных расходов на программную адаптацию иметь состояние комплекса, отвечающее последним требованиям по структуре данных.

Схема перехода к динамическому конфигурированию параметров

При традиционной схеме построения программного комплекса структура данных представлена двумя статически-динамическими подструктурами с фиксированными на этапе разработки объектами (вариант расчетной задачи и объекты проекта препроцессора). Полный цикл адаптации препроцессора под новые функциональные возможности решаемых задач занимает 2–3 месяца и включает в себя блокирующие разработку методики этапы (рис. 1,а), в то время как схема с дина-

мическим конфигурированием параметров позволяет как избежать затрат времени на изменение структур данных, так и исключить блокирующие стадии за счет использования универсального диалога работы с объектами (рис 1,б).

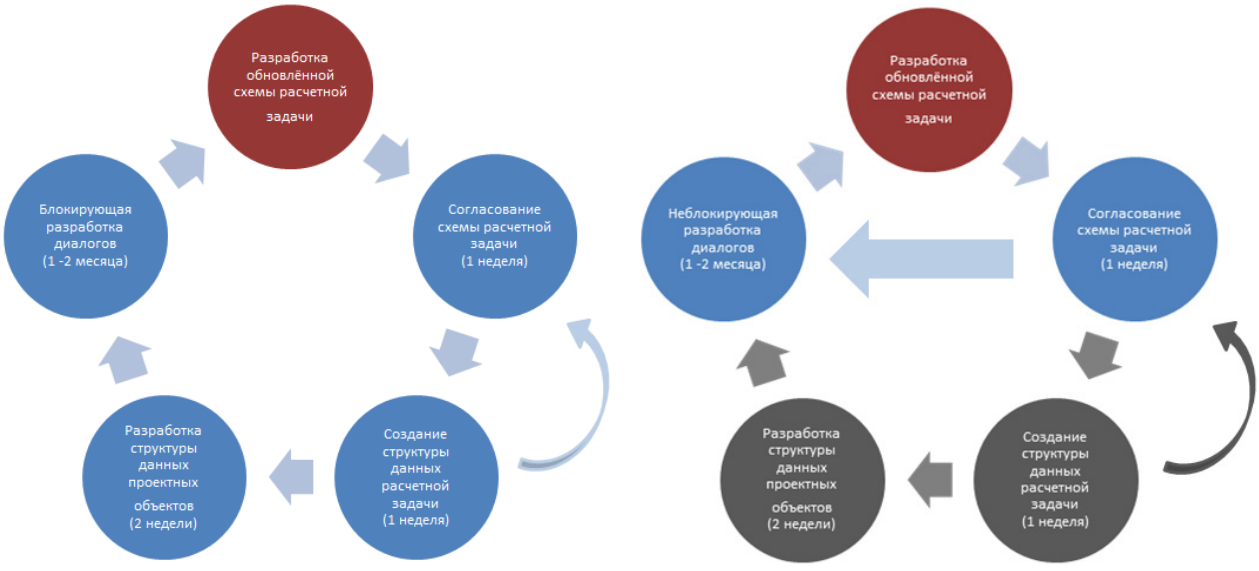

Рис. 1. Схема цикла разработки при использовании статическидинамической (а) и полностью динамической (б) структуры данных

Для перехода на полностью динамическую структуру требуется иметь возможность конфигурировать внешними средствами следующие параметры:

- набор объектов в терминах предметной области;
- список параметров объекта, требующихся для решения задачи моделирования;
- список параметров объекта, существенных для визуального восприятия его в пре- и постпроцессоре (название объекта, текущий режим работы и т. д.);
- список параметров, определяющих поведение элемента при его редактировании в препроцессоре (должен ли быть видим, может ли иметь видимые дочерние элементы и т. д.);
- взаимосвязь объектов различного типа – как отношения владения, так и ссылочные зависимости (отношения использования);
- тривиальные условия правильности схемы (допустимые диапазоны значений, количество элементов и т. д.);
- нетривиальные условия правильности схемы и алгоритмы проверки, подразумевающие анализ сочетания элементов и их содержимого.

Выбор XML Schema для динамического конфигурирования

В качестве языка динамического конфигурирования параметров был выбран язык XML Schema. XML Schema – язык описания структуры XML-документа, разработанный для создания программного обеспечения для обработки XML документов. В качестве основных преимуществ, учтенных при выборе этого языка, следует отметить:

- большой набор логических блоков
- («список», «выбор», «последовательность» и т. д.), с помощью которых легко описываются сколь угодно сложные структуры данных с глубокой вложенностью;

• наличие на рынке программных продуктов редакторов XML Schema, позволяющих легко, используя графический интерфейс, редактировать, трансформировать и отлаживать разрабатываемые структуры данных;

• наличие открытых программных библиотек, написанных на языке C++ и позволяющих существенно облегчить разработку синтаксического анализатора этого языка и встраивание его в комплекс Нимфа.

После детального анализа доступных библиотек обработки XML Schema был выбран продукт Apache Xerces C++ (вер. 3.1) как:

• содержащий быстрый SAX-парсер для обработки файлов в формате именно XML Schema (т. е. оперирующий понятиями и атрибутами стандарта);

• кроссплатформенный;

• предоставляющий возможности валидации полученного XML файла на соответствие исходной XML Schema.

Замечание. В реальности, как оказалось, все остальные рассмотренные парсеры также используют Xerces C++.

Основные логические блоки динамической структуры данных

Так как языком управления новой динамической структурой был выбран язык XML Schema, то основные блоки динамической структуры данных – базовые классы, разрабатываемые на языке C++, – должны соответствовать, а иногда и дополнять набор функциональных блоков языка XML Schema. На рис. 2 представлена UML диаграмма иерархии базовых классов предлагаемой динамической структуры данных.

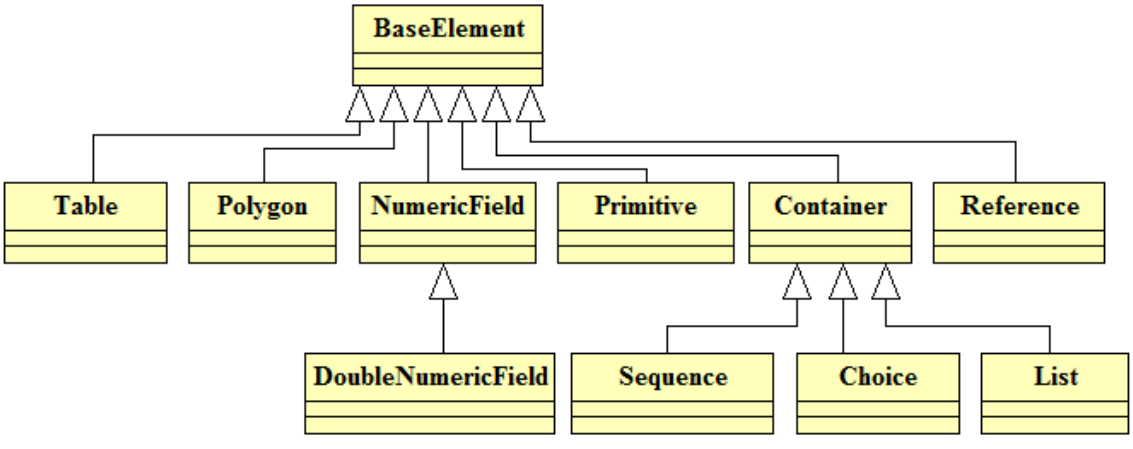

Рис. 2. UML диаграмма иерархии базовых классов динамической структуры данных

Кратко базовые классы динамической структуры описываются следующим образом:

• BaseElement – класс, описывающий общие атрибуты и методы всех базовых классов;

• Primitive – «простой» класс, не владеющий другими классами, а только атрибутами. Выступает в качестве «листьев» в древовидной структуре;

• Polygon – разновидность «простого» класса, введен в качестве расширения логических блоков описывающих структуру данных для указания специального типа объекта «полигон». Задает список объектов типа Point2D (точка в двухмерном пространстве);

• Table – разновидность «простого» класса, введен в качестве расширения логических блоков описывающих структуру данных для указания специального типа объекта «таблица». Задает список объектов типа TableRow (строку таблицы);

• NumericField – разновидность «простого» класса, введен в качестве расширения логических блоков описывающих структуру данных для указания специального типа объекта «цифровое поле». Задает общие атрибуты, описывающие двухмерный массив значений;

• DoubleNumericField – «цифровое поле» чисел с плавающей точкой;

• Container – «сложный» класс. Задает список объектов, которыми он владеет – при удалении объекта из этого списка приводит к его «разрушению»;

• Sequence – «сложный» класс, описывающий список-последовательность разнотипных объектов;

• Choice – «сложный» класс, описывающий список объектов-вариантов, из которых может быть выбран только один;

• List – «сложный» класс, описывающий список однотипных объектов;

• Reference – «сложный» класс, описывающий связь-ссылку на различные классы структуры. Удаление этой связи не приводит к «разрушению» объекта, на который была указана ссылка.

Дополнительно выделен ограниченный набор условно простых типов, используемых в xml-схеме и имеющих прямое отображение на классы C++, а именно:

• 2D и 3D точки;

• массив вещественных значений.

Ограничения, накладываемые на XML Schema

В реализованном подходе динамического конфигурирования параметров комплекса используются не все возможности стандарта XML Schema, в частности, нет поддержки групп атрибутов (**attributeGroup**), групп элементов (**group**), ссылок (**ref**).

Другой тип ограничений – зарезервированные комплексом ключевые слова, реакция на которые определена и которые не следует использовать в иных целях. К таким словам относятся атрибуты конфигурирования команд меню, управления видимостью, атрибуты названия, идентификатора, перечня функций объекта и некоторые другие.

Также не поддерживается смешение на одном уровне произвольного количества разнотипных элементов, т. е. не может быть в одной последовательности один элемент типа A и от двух до пяти элементов типа **B**. Это ограничение условно, и для описания примера используется последовательность из элемента типа **A** и списка элементов типа **B** (рис. 3).

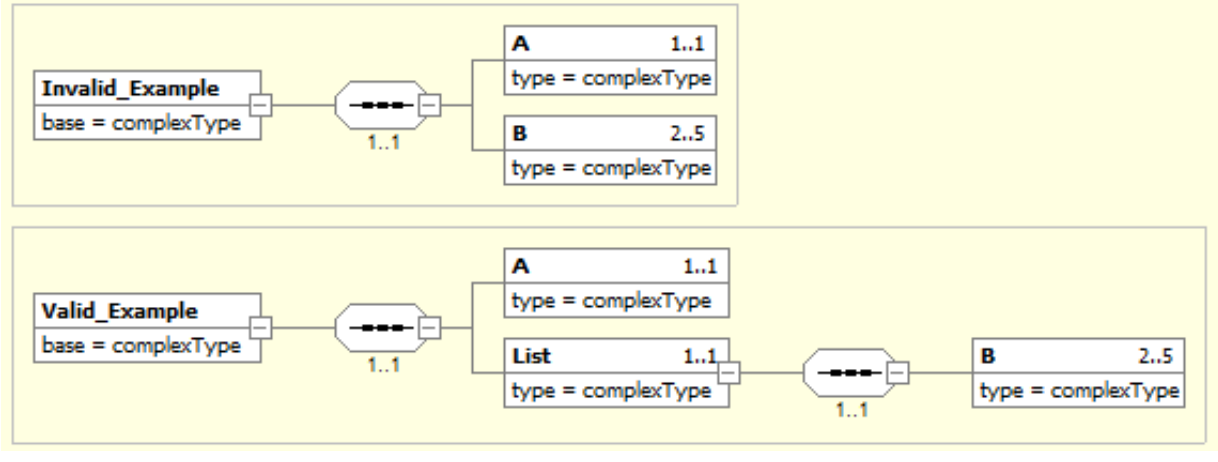

Рис. 3. Неподдерживаемое (вверху) и поддерживаемое (внизу) описание сложного элемента

Нужно отметить, что несоблюдение указанных ограничений не приведет к полной неработоспособности комплекса – неподдерживаемые конструкции и атрибуты игнорируются синтаксическим анализатором с выводом соответствующей диагностики.

Поддержка в препроцессоре

Препроцессор предоставляет следующие основные возможности по обработке проекта:

- дерево объектов для визуализации иерархии объектов;
- 2D-редактор с возможностями редактирования предопределенных типов объектов;
- 3D-вьювер для просмотра предопределенных типов объектов.
- универсальный диалог редактирования элемента с возможностями:
  - добавления, удаления вложенных элементов;
  - изменения доступности атрибутов/вложенных элементов;
  - инициализации ссылок на внешние элементы;
  - выбор альтернативного значения элементов;
  - редактирование значения атрибутов;
  - изменение табличных данных (для элементов типа «таблица»);
  - вызов внешних функций как для отдельных атрибутов, так и для элемента в целом.

Набор внешних функций ограничен и поставляется вместе с препроцессором, в дальнейшем планируется развитие механизма подготовки процессирующих функций (например, через подключаемую библиотеку и файл описания). Поставляемый функции могут быть применены для воздействия как на один только атрибут, так и на весь элемент в целом. К первым относятся, например, функции получения имени файла, ко вторым – загрузка элемента (таблицы) из файла промышленного стандарта (например, **csv** для таблиц свойств, **grid** или **surfer** для геологический поверхностей).

Наиболее сложной и одновременно наименее универсальной является поддержка графических 2D/3D объектов. И 3D вьювер, и 2D редактор предназначены только для обработки заранее известных им сущностей. Следовательно, требуется взаимодействие между разработчиком методики и разработчиками соответствующих плагинов, в ходе которого должен быть зафиксирован перечень именованных атрибутов, достаточных для задания требуемых характеристик объекта.

Сопряжение препроцессора и счетного модуля

Описанный подход использует единый файл и для описания проекта в препроцессоре, и для передачи параметров задачи в счетный модуль. При этом и счетный модуль, и препроцессор используют единую библиотеку для обработки входных данных, что гарантирует синхронность развития указанных компонент.

Очевидно, что в использование единого файла требует передачи счетному модулю некоторой избыточной для него информации, что, однако, компенсируется возможностью вернуться к редактированию расчетного файла в препроцессоре. Нужно отметить, что такая потребность регулярно возникает, и в предыдущих версиях комплекса требовалось принимать дополнительные организационные усилия для сохранности файла проекта.

Заключение

Система динамической конфигурации на базе языка XML Schema 1.0 успешно применяется в последних версиях программного комплекса Нимфа, предназначенного для решения задач фильтрации на суперкомпьютерах. Описанный подход позволяет получить следующие преимущества:

- задание и проверка целостности структуры файлов варианта расчетной задачи и проекта препроцессора комплекса;
- задание (и изменение) структуры данных препроцессора комплекса без перекомпиляции исходных кодов;
- конфигурирование графического интерфейса препроцессора комплекса;

• экономия программных ресурсов на разработке и обслуживании сервисных функций структуры данных;

• непрерывное наличие работоспособного комплекса с поддержкой актуальной структуры данных;

• использование удобных разработчику методики готовых профессиональных инструментов для развития XML схемы.

В то же время, данный подход в его текущем виде ориентирован, в основном, на разработчиков счетных методик, а не на конечного потребителя – использование специализированных диалогов, учитывающих расширенную информацию о природе данных, удобнее для пользователя, чем работа с универсальным диалогом. В этой связи именно развитие средств конфигурирования пользовательского интерфейса видится первоочередной и наиболее востребованной задачей при развитии описанного подхода.

Литература

1. Сайт «XML Schema. Рекомендация W3C». [Электронный ресурс]. Режим доступа: http://www.w3.org/TR/xmlschema-0.

2. Сайт программного продукта «The Apache Xerces Project». [Электронный ресурс]. Режим доступа: http://xerces.apache.org/index.html.

КОМПОНЕНТЫ ТЕХНИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПРОЕКТИРОВАНИЯ СИСТЕМ НА КРИСТАЛЛЕ КОМПАНИИ ALTERA CORPORATION

Д. В. Чернов

Вятский государственный университет, г. Киров

Основой цифровой системы в различных областях человеческой деятельности, как правило, является микропроцессор. Это или универсальный процессор, или цифровой сигнальный процессор DSP (Digital Signal Processor). Идея интеграции устройств различного назначения в единой системе привела к появлению микроконтроллеров. В отличие от микропроцессора, серийно выпускаемые микроконтроллеры обладают большим набором периферийных устройств.

Дальнейшим развитием идеи интеграции различных устройств в одной системе стало понятие «система на кристалле» SoC (system-on-chip). Идея SoC – объединить в одном кристалле как можно большее количество элементов цифровой системы. В итоге такие системы приобретают характеристики, присущие как микроконтроллерам, так и процессорам. Увеличение логической емкости ПЛИС привело к тому, что они становятся полноправными конкурентам при реализации SoC. В программной реализации система на кристалле получила аббревиатуру SoPC (система на программируемом кристалле).

Основная идея создания SoPC состояла в том, чтобы дать разработчику полный набор стандартных средств, имеющихся у DSP и микроконтроллеров, плюс дополнительный объем кристалла и необходимый сервис для реализации своих схемных решений. Ядром всей системы может являться встраиваемый RISC-процессор Nios II, архитектура которого приведена на рис. 1.