Сервисно-ориентированная система моделирования, обработки и анализа данных лидарного зондирования атмосферы

И. В. Бойченко[1], В. Н. Маричев[2], И. Ю. Турчановский[1]
[1]*Институт вычислительных технологий СО РАН*
e-mail: saturn900@gmail.com, tur@hcei.tsc.ru
[2]*Институт оптики атмосферы СО РАН*
e-mail: marichev@iao.ru

Рассматривается подход к построению программных систем (ПС) нового поколения в области лидарного зондирования атмосферы. Сервисно-ориентированная архитектура предлагается как основная парадигма построения системы, обеспечивающая такие свойства, как открытость, кросплатформенность и гибкость. Описывается подход к организации сервиса для построения графов решения задачи (графов управления). Рассматриваются базовые принципы и технологии построения ключевых сервисов: "конструктор" и "хранилище". В заключении делается вывод о возможности применения предложенного подхода к построению перспективных программных систем для других областей науных исследований.

Ключевые слова: программное обеспечение научных исследований, сервисно-ориентированная архитектура, графы управления, сервис хранения данных

1. Введение

Дистанционное зондирование атмосферы с помощью лидаров осуществляется с 60-х годов XX-го века [1–3]. Данный метод исследований сопряжен с обработкой пространственно распределенных числовых данных, получаемых в ходе сеансов зондирования. Очевидно, что эффективная обработка массивов экспериментальных данных невозможна без применения вычислительной машинерии и соответствующего математического и программного обеспечения.

Разнообразие параметров зондирующих установок, методов зондирования, а также большое множество математических методов и алгоритмов, применяемых для обработки данных, порождают высокую сложность программных систем (ПС).

Опыт разработки ряда ПС [4–8] показывает, что изменчивость предметной области приводит к интенсивному потоку изменений, вносимых в программный код. Фактически, ПС эволюционирует вместе с получением новых знаний о предмете исследования. Соответственно, программное обеспечение становится способом представления и закрепления знаний.

При таком жизненном цикле вполне закономерно, что разработка программ, как правило, осуществляется силами самих исследователей. И получаемое в результате программное обеспечение является уникальным артефактом, ценным, как с научной точки зрения, так и с точки зрения затраченного труда.

В связи с этим актуальными являются следующие вопросы:

1. как организовать верификацию полученных результатов: сопоставление с результатами других исследователей, с ранее известными моделями и данными, например, полученными другими методами зондирования?

2. как организовать повторное использование полученных результатов для получения новых знаний и построения новых моделей?

Под результатми здесь понимаются не только численные результаты расчетов, но и сам программнный код, а также наборы входных параметров, позволяющие при необходимости воспроизвести расчет.

Таким образом, жизненный цикл научного программного обеспечения не должен заканчиваться только лишь получением удовлетворительных с исследовательской точки зрения результатов расчета, а должен продолжаться или в виде верифицирующего эталона и/или в виде активного компонента в новых исследованиях.

При этом, получаемые результаты, должны быть доступны всему научному сообществу, именно в том виде, в каком их получил исследователь.

Программные системы, обеспечивающие решение указанных выше задач мы предлагаем относить к системам нового поколения, в противовес системам, ориентированным только на получение численного результата расчета.

Исходя из сформулированных выше задач, и учитывая современный уровень развития информационно-коммуникационных технологий, нами был предложен подход к построению программных систем нового поколения в области моделирования, обработки и анализа данных ЛЗА.

Идеологически близкий подход был рассмотрен в работе [9] с теоретической точки зрения. В нашей работе предлагаются конкретные технологические решения, рассматриваемых в [9] вопросов, в частности, управление вариантами решения задач. Предложенный нами формат графа управления позволяет применять программные модули, написанные на разных языках программирования, что особенно актуально при включении в состав системы модулей, разработанных ранее, без учета интерфейса системы.

Идея сервиса “конструктор”, являющегося ключевым в нашей системе, коррелирует с идеями, высказанными в работе [10]. В частности, нами использована идея о том, что конкретный вариант выполняющейся программы должен собираться автоматически на основе ранее сконструированной вычислительной модели - графа решения задачи (или графа управления). Но в отличие от [10], на данный момент, мы не ставим целью создание универсального автоматического решателя задач, обладающего элементами искусственного интеллекта. Мы считаем, что исследователь предпочтет сохранить контроль над методами решения задачи, а система должна лишь избавить его от рутинных действий при модификации графа управления в ходе исследований.

Идея об обмене вычислительными моделями в исследовательском сообществе уже успешно реализуется, например, в системах Taverna и myExperiments [11, 12]. Эти системы ориентированы на биологические исследования. Мы предполагаем использование опыта этих проектов при создании нашей системы.

2. Концепция системы

Функциональная схема системы представлена на рисунке 1.

Наиболее очевидным требованием к системе, является то, что система должна быть многопользовательской и веб-ориентированной. В современных условиях исследователь

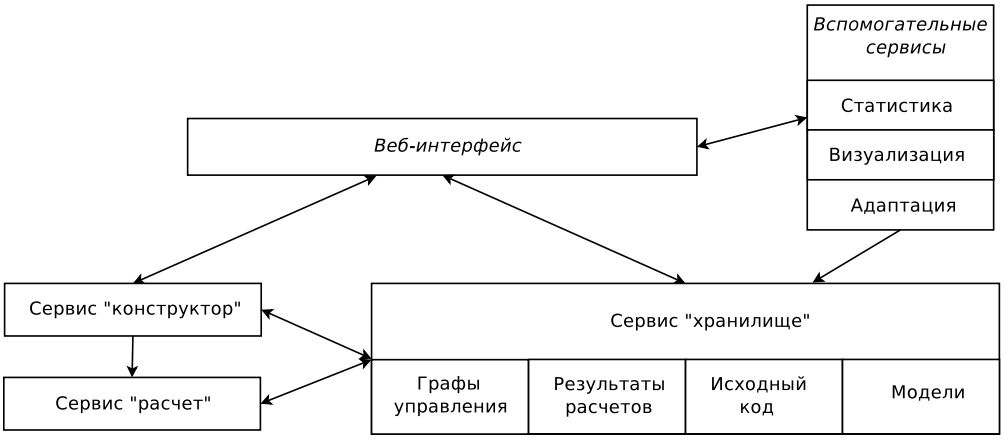

Рис. 1. Функциональная схема системы

должен иметь возможность работать с системой из любой точки мира, имея типовой набор программного обеспечения (веб-браузер и стек Интернет-протоколов).

Станции лидарного зондирования размещаются в различных точках Земли, соответственно, географически распределены и исследовательские группы. Использование сети Интернет для обмена данными и программным обеспечением является в данном случее оправданным решением.

Так как исследователи должны обмениваться результатами между собой в целях верификации и повторного использования, то система должна предоставлять соответствующие службы (сервисы) описания, хранения и поиска результатов работы исследователей. Назовем данный сервис “хранилищем”.

Для того, чтобы непосредственно использовать разработанное ранее программное обеспечение как составной компонент в новых расчетах, необходимо наличие сервиса, позволяющего задать граф управления расчетом, оперируя как типовыми, так и разработанными ранее компонентами. Данный сервис назовем “конструктором”.

Сервис “расчет” принимает граф управления в качестве входного параметра и строит программу расчета из обозначенных в графе управления модулей, а затем запускает и контролирует выполнение программы.

Перечислим также ряд вспомогательных сервисов:

1. сервис статистики и поствычислений;

2. сервис визуализации;

3. сервис адаптации;

Вспомогательные сервисы не являются частью результатов исследователя, но помогают обрабатывать данные на любом этапе без трудозатрат на разработку ПО.

Сервис адаптации предназначен для предоставления нестандартным модулям необходимого для них окружения. Так, для модуля, которому необходима другая операционная система, может быть создан отдельный виртуальный компьютер.

Далее, рассмотрим два основных сервиса системы: “конструктор” и “хранилище”.

3. Сервис “конструктор”

Как уже было сказано, основной проблемой является интенсивность изменений, порождаемых поисковой природой научных исследований, и, следовательно, необходимостью синтеза и апробации различных способов решения научных задач.

Формируя концепцию сервиса мы исходили из предположения, что научная задача, как правило, разбивается на несколько этапов решения, и на каждом этапе применяется тот или иной метод или алгоритм.

Сервис “конструктор” предназначен для выделения этапов (подзадач) решения задачи, задания зависимости между этапами и назначения конкретного численного метода решения подзадачи (этапа).

Сформированный граф решения задачи, является неотъемлемой частью результата исследований и может использоваться для сопоставления с другими результатами и моделями, а также включаться в состав новых графов.

В ходе научного поиска именно граф решения может наиболее интенсивно меняться.

Далее, рассмотрим принципы организации сервиса “конструктор”.

Этапы рассчета предлагается выделять на основе сущностей (физических величин), которыми оперирует специалист в предметной области. Таким образом, этап - это часть алгоритма, в которой полностью рассчитывается какая-либо величина, часто встречаемая в языке предметной области.

В общем случае может существовать большое количество вариантов декомпозиции исходной задачи на этапы, поэтому, при выделении этапов, мы предлагаем исходить из следующих соображений: с одной стороны, не следует выделять слишком много этапов, на которых рассчитываются малоинформативные сами по себе величины, с другой стороны, не следует выделять слишком крупные этапы, так как в этом случае будет создано мало точек контроля процесса расчета, и будут затруднены последующие модификация, верификация и повторное использование результатов.

На основе сделанного анализа можно построить пропозициональный («И/ИЛИ») граф процесса расчета некоторой физической задачи. Граф процесса расчета показывает последовательность этапов, приводящщую к решению исходной задачи.

На рисунках 2 и 3 приведены примеры графов решения прямой и обратной задач лидарного зондирования атмосферы [4].

Вершины P на рисунке 2 представляют собой этапы расчета прямой задачи.

P_1 - Расчет лидарных эхосигналов для выбранных длин волн;

P_2 - Расчет модели атмосферы;

P_3 - Расчет модели фонового излучения;

P_4 - Выбор параметров лидарной установки;

P_5 - Выбор длин волн для зондирования, определение параметров лазеров;

P_6 – Выбор техники зондирования (конечная вершина)

Вершины R - представляют точку выбора метода для рассчета этапа.

Вершины M - представляют конкретные численные методы и алгоритмы, которые могут быть применены для решения задачи.

Вершины S на рисунке 3 представляют собой этапы рассчета обратной задачи. Вершины M и R имеют такое же значение как на рисунке 2.

S_1 – Дифференцирование оптической толщи, расчет концентрации газа;

S_2 – Расчет опти ческой толщи;

S_3 – Расчет модели атмосферы;

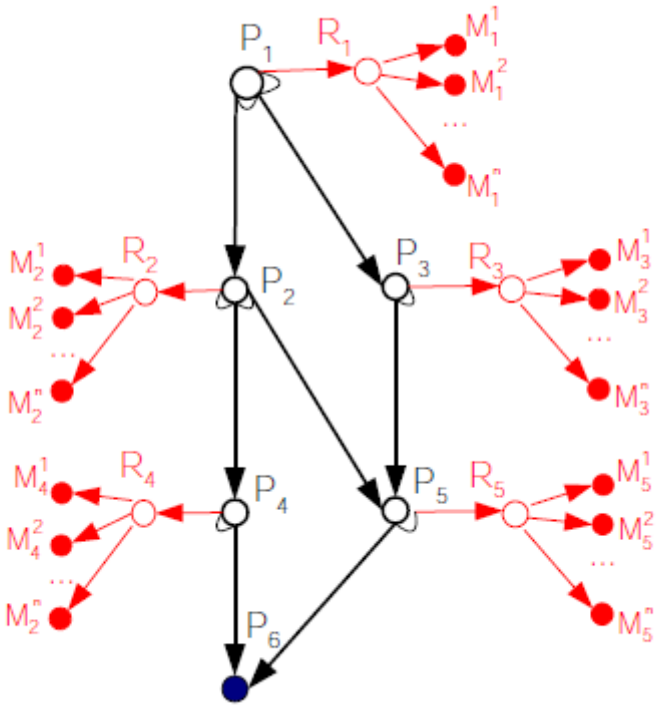

Рис. 2. Граф решения прямой задачи лидарного зондирования атмосферы

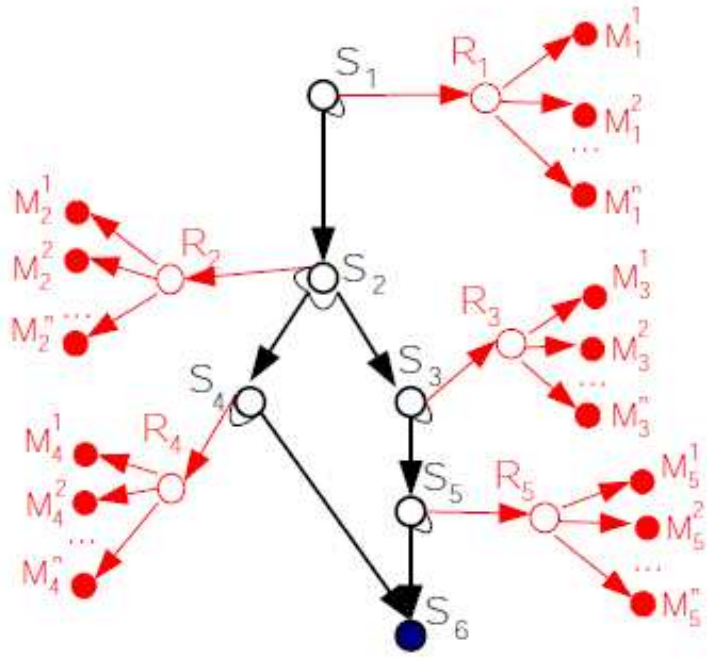

Рис. 3. Граф решения обратной задачи лидарного зондирования атмосферы

S_4 – Расчет фонового излучения;

S_5 – Сглаживание/сжатие сигнала;

S_6 – Выбор и загрузка сигналов лидарного зондирования (конечная вершина);

В ходе эволюции программного обеспечения разработчик (исследователь) варьирует параметры и вносит структурные изменения в общий алгоритм решения задачи.

Изменения можно сгруппировать следующим образом:

1. изменение состава и последовательности этапов решения задачи;
2. применение различных численных методов на конкретном этапе решения;
3. изменение состава и значений параметров численных методов;
4. применение инструментов постобработки, визуализации и архивирования (сохранения) выходных данных.

Процесс и технология разработки ПО для научных исследований должны обеспечивать простоту и эффективность внесения вышеописанных групп изменений.

Одним из способов эффективного управления изменениями является выделение слабосвязанных компонент, таким образом, чтобы изменения, вносимые в отдельный компонент, были бы изолированными, то есть не приводили бы к цепной реакции изменений в системе.

Реализация указанной выше концепции в виде ПО означает комбинацию программных блоков или модулей, каждый из которых отвечает за решение отдельного этапа задачи.

3.1. Представление графа управления в сервисе “конструктор”

Для описания графов решения задачи был выбран формат XML. Данный формат обеспечивает формальную проверку синтаксиса документа с помощью готовых компонент, обладает гибкостью с точки зрения последующего расширения словаря тегов и атрибутов, а также предоставляет возможности визуализации структуры файла в веббраузерах и редакторах, и трансформации документа в любые существующие форматы с помощью технологии XSLT.

Приведем пример описания одного из расчетных модулей, применяемых в ходе решения задачи восстановления отношения аэрозольного рассеяния в атмосфере Земли на основе отраженных лидарных сигналов.

```
<module name="Correcting LIDAR signals">
  <title> Correcting LIDAR signals </title>
  <exec_name> rad.bin.RunnerCorrect </exec_name>
  <sufix> correct </sufix>
  <exec_path> rad.bin </exec_path>
  <parameters>
    <par name="tau_slip" type="float">
      0.00000004
    </par>
  </parameters>
  <outFile>
        <name> signal*.txt  </name>
        <cols>
```

```
            <col num="1">
                <name> altitude </name>
              <unit> km </unit>
            </col>
            <col num="2">
              <name> signal </name>
              <unit> photons </unit>
            </col>
          </cols>
        </outFile>
        <graphics type="gnuplot">
          <script>
            set autoscale xy
            set style data lines
            set logscale x
            plot "signal.txt" using 2:1
            replot "signal.correct.txt" using 2:1
            # pause -1 "Press Ok or Cancel to continue ..."
            # pause 15
          </script>
        </graphics>
    </module>
```

Тег module описывает один этап (подзадачу расчета) и документ xml может содержать один и более таких тегов.

Тег module, в свою очередь, содержит следующие теги:

title – задает словесное описание этапа вычислений, в данном случае это коррекция сигнала на предмет «слипания фотонов»,

exec_name – определяет название исполняемого модуля, в данном случае это название класса Java – rad.bin.RunnerCorrect, который реализует алгоритм коррекции сигнала,

exec_path – относительный путь до исполняемого модуля, используется если в exec_name не задан абсолютный путь parameters – тег раздела содержащего описания параметров par par – тег описания параметра расчета, характеризуется атрибутом name – имя параметра, и type – задающий тип параметра.

outFile – тег описания выходного файла, содержит название файла и теги описания столбцов (предполагается, что выходной файл будет иметь матричную структуру)

graphics – тег описания сценария для визуализации результатов расчета, фактически содержит сценарий (скрипт), например, интерпретируемый пакетом GnuPlot [14].

В системе предусмотрено два типа расчетных модулей – Java-модули, реализующие стандартный интерфейс, и не-Java модули, написанные на других языках программирования и собранные в виде отдельных исполняемых модулей.

В системе предусмотрен специальный класс-оболочка UniversalExecRunner, который обеспечивает запуск стороннего (не-Java) модуля в виде отдельного процесса с перехватом стандартных потоков ввода/вывода.

Параметры, описанные в XML — передаются исполняемому процессу либо через стандартный ввод, либо через текстовые файлы (имеется опыт использования модулей

написанных на языке Fortran и подключенных к системе в виде отдельно запускаемых программ).

Таким образом, сервис обеспечивает использование как модулей, изначально ориентированных на интерфейсы системы, так и модулей, которые разрабатывались вне системы, и которые могут использовать любые другие технологии программирования. Главным требованием для подключения внешнего модуля является наличие файлового или консольного интерфейса.

В предельном случае, если модуль использует специфическое окружение, то для его включения в систему предусмотрено использование сервиса виртуализации, такого как XEN [16].

3.2. Возможности, возникающие при представлении алгоритма решения задачи в виде графа

Предложенный метод декомпозиции позволяет управлять сложностью разрабатываемой программной системы за счет выделения подсистем и категоризации модулей. Имея наглядное представление состава этапов и структуры зависимостей между ними исследователю легче управлять элементами решения задачи.

При этом применение описанного выше подхода возникает ряд ценных возможностей для проведения исследований. По крайней мере видятся следующие возможности:

1. синтез алгоритма решения задачи по заданному критерию (например, определения наиболее эффективного метода решения подзадачи в зависимости от размерности входных параметров);

2. организация направленного или ненаправленного поиска оптимальных решений за счет проведения массовых расчетов с перебором параметров;

3. возможность смешанного применения символьных и численных рассчетов на различных этапах решения задачи;

4. распараллеливание решения задачи, за счет выделения наиболее трудоемких этапов расчетов для решения на высокопроизводительных вычислительных системах.

4. Сервис “хранилище”

Сервис предназначен для выполнения следующих функций:

1. доступ к локальным и удаленным базам и архивам данных (метеомодели, спектроскопические данные и модели), а также к информационным сервисам (например сервисы метеоданных);

2. хранение результатов расчетов в виде графов решения, входных, промежуточных и выходных данных, а также версий исходного и/или бинарного кода;

Сервис “хранилище” должен обеспечить сохранение, поиск и извлечение всех возможных результатов исследований, а также всех ресурсов, необходимых для проведения расчетов.

У исследователя должна быть возможность воспроизвести любой расчет, в любой версии графа решения. Можно сказать, что в этом качестве сервис выполняет функции системы управления версиями, с тем отличием, что автоматически версионируется не только код, но также и входные, выходные и промежуточные результаты расчетов.

В исследованиях атмосферы входные и выходные данные в общем случае являются многомерными. Для организации хранения многомерных данных нами использованы разработки UCAR [13], включающие форматы хранения данных netCDF и HDF, язык запросов к многомерным структурам, а также библиотеки для различных языков программирования и готовое программное обеспечение.

Для хранения результатов исследований в виде графов решения и их параметров предполагается использовать документоориентированные системы хранения, типа MongoDB [15].

Интерфейс сервиса "хранилище" организован в соответствии с принципами REST.

5. Заключение

Так как разработка программного обеспечения в различных научных областях состоит из типового набора действий, то предлагаемый подход может иметь универсальное применение.

В настоящее время сервисно-ориентированная система находится в стадии разработки. Отдельные элементы сервиса "конструктор" были разработаны в рамках системы RAD [8] и используются в Институте оптики атомсферы СО РАН и ряде других организаций.

Поэтапное развертывание сервисно-ориентированной системы планируется осуществлять на мощностях Томского филиала Института вычислительных технологий в течении ближайшего года.

Список литературы

[1] Лазерный контроль атмосферы /под ред. Хинкли Э.Д., Зуева В.Е.– М.:Мир, 1979. – 416 с.

[2] Захаров В.М., Костко О.К. Метеорологическая лазерная локация – Л.: Гидрометеоиздат, 1977. -203с.

[3] Межерис Р. Лазерное дистанционное зондирование – М.: Мир, 1987. – 550с.

[4] Бойченко И.В., Катаев М.Ю. Организация процесса исследований в системе ODRIS // Сб. трудов «Автоматизированные системы управления экспериментом», ТУСУР, г. Томск, 1999, Т 3, С. 51–56

[5] Зуев В.В., Катаев М.Ю. Информационная система для обработки, анализа и хранения стратосферных оптических измерений / Зуев В.В., Катаев М.Ю., Маричев В.Н., Мицель А.А., Бойченко И.В. // Оптика атмосферы и океана, 1999, Т 12, № 5 С. 453–457

[6] Бойченко И.В., Катаев М.Ю., Маричев В.Н. Информационная система для анализа данных лидарного зондирования озона. // Гидрология и метеорология, 2001, № 12 с. 96–105

[7] Kataev M.Yu., Maksyutov S., Boichenko I.V. Software 7S for simulating of the solar reflected radiance transfer in atmosphere // Proc. SPIE Vol. 6160, Twelfth Joint International Symposium on Atmospheric and Ocean Optics/Atmospheric Physics; G. A. Zherebtsov, G. G. Matvienko; 2006

[8] Бойченко И.В., Катаев М.Ю. Программная система моделирования отраженного от поверхности Земли солнечного излучения // Доклады Томского государственного университета систем управления и радиоэлектроники. - 2009. - 1(19), часть 1. - С. 88-95.

[9] Горбунов-Посадов М.М. Расширяемые программы. — М.: Полиптих, 1999. — 336С.

[10] Тыугу Э.Х. Концептуальное программирование – М.:Наука, 1984. – 255С.

[11] De Roure, D.; Goble, C. Software Design for Empowering Scientists // Software, IEEE, vol.26, no.1, pp.88,95, Jan.-Feb. 2009

[12] Roure, D.D.; Goble, C. et al. The Evolution of myExperiment e-Science (e-Science) // 2010 IEEE Sixth International Conference on, pp.153,160, 7-10 Dec. 2010

[13] University corporation for atmospheric research - Режим доступа: https://www2.ucar.edu/ - свободный

[14] Проект GnuPlot - Режим доступа: http://www.gnuplot.info/ - свободный

[15] Проект MongoDB - Режим доступа: http://www.mongodb.org/ - свободный

[16] Проект XEN - Режим доступа: http://www.xenproject.org/ - свободный