# §7 МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ НОВЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ



Куракин П.В.

## СПЕЦИАЛИЗИРОВАННЫЕ СИСТЕМЫ МАТЕМАТИЧЕСКИХ РАСЧЕТОВ НОВОГО ПОКОЛЕНИЯ

***Аннотация:*** *Во многих отраслях и государственных администрациях требуются программные средства для специализированных расчетов, аналогичные популярной системе MATLAB в совокупности с подсистемой графического редактирования задач Simulink, но в сочетании с некоторым хранилищем данных и описаний задач, при этом основанные на бесплатном программном обеспечении. Ключевой недостаток стандартной связки пакетов MATLAB + Simulink (помимо коммерческой цены) следующий: библиотека графических примитивов подсистемы Simulink на деле ограничена популярными и типичными инженерно - расчетными задачами.Требуется разработка соответствующей программной среды, опирающейся на свободно распространяемое программное обеспечение. Описываемая программная среда в целом укладывается в концепцию «клиент – сервер» и опирается на платформу Java и веб – технологии. Клиентская часть использует визуальный графический редактор, реализованный как браузерное приложение. Серверная часть передает данные (конфигурацию задачи) к пакету численных расчетов Octave и наоборот - результаты расчетов - в браузер. Данные передаются по сети в виде строки в формате JSON. Разработана оригинальная программная архитектура для специализированных систем математических расчетов на основе свободно распространяемого программного обеспечения. С учетом подсистемы хранения конфигураций задач (которая требует дальнейшего развития) данная архитектура становится основой для создания специализированных систем поддержки принятия решений во многих отраслях. Архитектура оставляет большое пространство возможностей для дальнейшего развития.*

***Ключевые слова:*** *математические, расчеты, Java, JavaScript, Octave, Python, системы, поддержки, принятия, решений*

***Abstract:*** *In many sectors and state administration agencies there is a great need in the software designed for specialized calculations like a popular MATLAB system combined with the Simulink graphical programming environment, but dealing with a particular database and task descriptions and not requiring any special payment. The main disadvantage of the MATLAB+Simulink package*



DOI: 10.7256/2305-6061.2016.1.17997

*(apart from its price) is that the Simulink library of graphical primitives is in fact limited by popular and typical engineering calculating tasks. The author of the article emphasizes the need to develop the kind of programming environment which would be based on the freely distributed software. The programming environment described by the author supports on the 'client-server' concept and bases on the Java platform and web-technologies. The client uses the visual graphics editor implemented as a browser application. The server transmits data (task configuration) to the Octave calculation package and vice versa, calculation results are transmitted to the browser. Data is transmitted online as the JSON line. The author creates the original programming architecture for specialized mathematical calculation systems based on the freely distributed software. Taking into account the subsystem that stores the task configuration (and needs to be developed further), this architecture becomes the basis for creating specialized systems of the decision making processes in many spheres. The architecture provides a wide range of opportunities for further development.*
***Keywords:*** *mathematical , calculations, Java, JavaScript, Octave, Python, systems, support, making, decision*

## Введение

Практика показывает [1], что в настоящее время все чаще оказываются востребованными специализированные системы математических расчетов, которые включают:

1. (расширяемые) хранилища специализированных данных пользователя (как правило, это аналитические службы и лица, принимающие решения в отраслевых и государственных организациях) связанных с соответствующей предметной областью;
2. (расширяемые) библиотеки вычислительных методов и математических моделей, характерных для соответствующей предметной области;
3. пользовательский интерфейс, представляющий собой либо включающий в себя как ключевой компонент визуальные средства создания и редактирования типовых расчетных задач, характерных для соответствующей предметной области;
4. вычислительная подсистема, позволяющая автоматически выполнять требуемые расчеты при наличии нужных входных данных;
5. программный механизм передачи данных (структурированного описания задачи, требующей тех или иных расчетов) от пользовательского интерфейса к вычислительной подсистеме (перед выполнением расчетов) и в обратном направлении (после выполнения расчетов).

Кратко и на качественном уровне проблему можно описать так: требуются программные средства, аналогичные популярной системе MATLAB [2] в совокупности с подсистемой графического редактирования задач Simulink [3], но в сочетании с некоторым хранилищем данных и описаний задач, при этом основанные на бесплатном (свободно распространяемом) программном обеспечении.

Ключевой недостаток стандартной связки пакетов MATLAB + Simulink (помимо того, что это весьма дорогостоящий коммерческий продукт, т.е. практически непригоден для



DOI: 10.7256/2305-6061.2016.1.17997

изготовления легальных «коробочных» программных решений для государственных заказчиков) следующий: библиотека графических примитивов подсистемы Simulink, при ее кажущемся обилии, на деле ограничена некоторыми популярными и типичными инженерно - расчетными задачами.

Все большему количеству организаций требуется доступный программный инструмент математических расчетов в самых разнообразных прикладных областях с развитым инструментарием визуального редактирования задач, причем эти задачи, как правило, требуют специфических графических примитивов и способов их объединения в целостные графические описания этих задач.

Далее в тесте, для удобства, наряду с термином «архитектура» используются термины «решение», «архитектурное решение» и «программная среда», поскольку описываемая архитектура и есть решение, объединяющее в единую схему и единую среду ряд программных модулей, как готовых (присутствующих на рынке свободного ПО), так и разработанных в ходе выполненных исследовательских работ.

## 1. Качественное описание предложенного решения

Наша логика при проектировании решения описанной проблемы состояла в следующем. Если ставится задача максимально возможным образом опираться на свободно распространяемое ПО, это означает, что приложение заведомо будет модульным. Если использовать целые платформы им и готовые и полноценные программные комплексы (например, с самого начала была сделана ставка на бесплатный пакет математических расчетов Octave [13]), то решение будет распределенным, т.е. представлять собой совокупность параллельно работающих процессов (в рамках одной либо нескольких вычислительных машин). Это значит, что система, так или иначе, должна опираться на интенсивное взаимодействие процессов. Так или иначе, в том или ином виде, в той или иной «обертке», потребуется реализовать механизмы удаленного вызова процедур (концепция RPC [16]). Именно, потребуется реализовать такой удаленный вызов при запуске математических вычислений (средой Octave либо какой-то другой) на основе конфигурации задачи, созданной в среде графического визуального редактора.

С другой стороны, надо сразу принять решение, на основе какой платформы осуществлять разработку визуальных средств редактирования описаний задач в среде пользовательского интерфейса. Реализация графического интерфейса в среде веб - браузера (средствами стандартного браузерного языка программирования JavaScript) оказалась наиболее естественным решением, поскольку имеются хорошо зарекомендовавшие себя библиотеки разработки графических пользовательских интерфейсов на языке программирования JavaScript.

С учетом сказанного, был сделан выбор: все решение следует реализовать как веб - приложение. Далее, решение создавать веб - приложение предопредило и выбор Java как базовой платформы всей платформы, поскольку наиболее популярные, опробованные, а тем более бесплатные серверы веб - приложений (семейство Apache Tomcat) реализованы именно на Java.




DOI: 10.7256/2305-6061.2016.1.17997

## 2. Общее описание архитектуры

Описываемая программная среда в целом укладывается в концепцию «клиент – сервер» [4] и опирается на платформу Java и веб – технологии (Рис. 1).

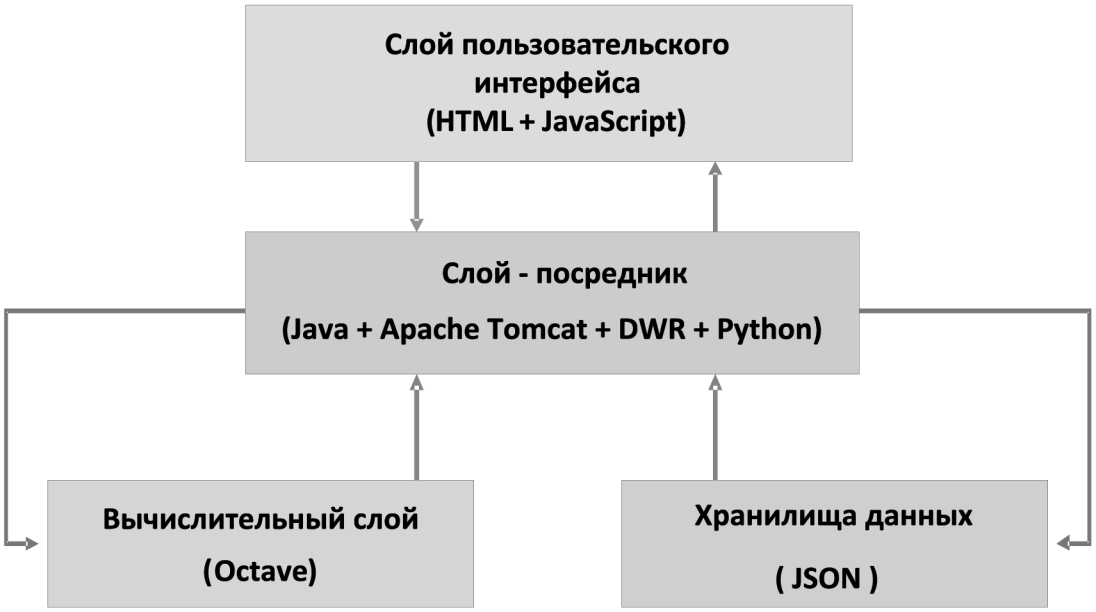


Рис. 1. Архитектура программной среды

С точки зрения количества «уровней» данную архитектуру можно с равным успехом характеризовать как трехуровневую, так и четырехуровневую. В некоторых системах классификации подсистемы «логики» и «данных» разносят на разные уровни. Автор предпочитает рассматривать эти подсистемы как находящиеся на одном уровне, тогда о всей архитектуре уместно говорить как о трехуровневой.

Ключевую роль для функционирования описанной архитектуры играет язык разметки данных JSON [5], который исторически возник как раз для браузерных задач (аббревиатура расшифровывается как JavaScript Object Notation, иногда этот язык называют «обезжиренный XML»). JSON – документ представляет собой объект типа *словарь*, т.е. перечень вхождений вида <ключ> : <значение>, через запятую. Значение может быть как атомарным типом (включая массив), либо также представлять собой объект (перечень включений). Каждый объект – перечень заключается в фигурные скобки.

Вся схема может работать, только если существует возможность передавать данные из одного процесса операционной системы в другой, желательно в виде текстовой строки, формат которой легко поддается расшифровке в обоих процессах. Стандарт JSON, как



DOI: 10.7256/2305-6061.2016.1.17997

показал опыт разработки, лучше всего соответствует этой цели.

Как отмечено выше, самый естественный и полновесный способ связи различных процессов в любой операционной системе – стек сетевых протоколов TCP\IP. Но гораздо удобнее пользоваться этим стеком не непосредственно, а при помощи какой-то «настройки» - т.е., пользоваться протоколом уровня приложения, а не транспортного уровня [6]. Естественным ответом является протокол гипертекстовой передачи гипертекста HTTP [7] (термин гипертекст означает обычный текст в совокупности с управляющими символами разметки этого текста).

В целом, функционирование всей среды как системы математических расчетов основано на том, что протокол HTTP передает описание задачи в виде JSON – строки от пользовательского интерфейса (реализованного как браузерный графический редактор) к вычисляющему слою. После выполнения расчетов, данные снова преобразуются в JSON – строку, которая сохраняется в хранилище системы и может быть потом снова загружена в пользовательский интерфейс. Далее это процесс рассмотрен более детально.

### 3. Схема работы архитектуры

Интерфейс пользователя, как уже указано, размещается на веб – странице, поэтому изначально речь идет о стандартном веб – приложении с HTML – файлами, установленном на веб – сервере Apache Tomcat [8] (версия 6.0.33, работает на платформе Java, которая должна быть предустановленна на компьютере).

На стороне веб – браузера данные объектов JavaScript, созданные пользователем в графическом редакторе, конвертируются в строку в формате JSON (*сериализация* данных [9]). Это преобразование выполняется средствами бесплатной JavaScript – библиотеки Yahoo UI [10] (эта же библиотека использована для создания графического интерфейса пользователя). Эта строка посредством технологии DWR (см. далее) передается в вычислительную подсистему.

Архитектурное решение использует Java - библиотеку открытого кода DWR [11]. Из Рис. 1 видно, что библиотека DWR находится в центре описываемой архитектуры. Технология DWR (Direct Web Remoting) встраивает в исходное веб – приложение специальный *сервлет*. Обычный сервлет [12] представляет собой небольшой и относительно простой Java – класс, устанавливаемый (вместе с дополнительными служебными файлами) на веб – сервере. Класс сервлета умеет обрабатывать команды (запросы) протокола HTTP, поступающие к серверу со стороны веб - страницы. В случае технологии DWR класс сервлета генерируется *автоматически*.

Технология DWR работает так. На стороне веб – сервера пользователь этой технологии размещает (самостоятельно, в отличие от тех классов, которые автоматически создает DWR) один или несколько собственных Java – классов и регистрирует их в конфигурационных файлах DWR (на сервере), а также на веб – странице приложения при помощи




DOI: 10.7256/2305-6061.2016.1.17997

специальной директивы языка разметки HTML. При старте приложения (при обращении к веб – странице по сети по протоколу HTTP) в пространстве оперативной памяти браузера создаются JavaScript объекты – дублеры серверных Java – классов пользователя. Эти объекты имеют те же методы, что серверные классы. С практической точки зрения достаточно иметь всего один такой Java – объект на сервере, назовем его *диспетчером*.

На стороне веб – страницы пользователь может вызывать JavaScript - методы объекта - дублера, что приводит к вызову одноименных методов серверного Java – объекта. Все такие методы серверного класса возвращают строку, и эта строка *асинхронно* возвращается (через сеть, т.е. средствами протокола HTTP) в браузер.

Таким образом, технология DWR позволяет создать на стороне веб – страницы «пульт удаленного управления» серверного Java – объекта (диспетчера) и получать от него результирующие данные в форме текстовой строки. Вызываемые удаленно методы суть методы, обращающиеся к хранилищам данных, либо методы, вызывающие на исполнение вычислительную подсистему. Разумеется, при вызове вычислительной подсистемы диспетчер передает ей входные данные, полученные в виде JSON – строки от браузера.

В качестве вычислительной подсистемы используется пакет прикладных математических расчетов Octave [13], который номинируется как функционально полный и бесплатный аналог пакета MATLAB. На стороне пакета Octave используется еще одна библиотека открытого кода для конвертирования JSON – строки в объекты языка программирования Octave (*десериализация* данных), который совпадает с языком программирования MATLAB. Это позволяет обрабатывать полученные данные средствами языка программирования Octave\MATLAB. После проведения вычислений объекты языка программирования Octave снова конвертируются в JSON – строку (снова сериализация), которая сохраняется в файл. Такие файлы мы называем *конфигурациями*. Этот JSON – файл снова можно загрузить (по HTTP – запросу через указанное выше сервлетное приложение) в среду графического интерфейса в браузере.

Необходимо отметить, что среда Octave запускается на исполнение асинхронно; решение записывается в исходный файл JSON – конфигурации задачи. При этом, исходный файл переписывается; поддерживается идеология одна задача – один файл; обработка конфигурации означает изменение значений тех или иных полей либо появление определенных значений в тех полях, где были указаны неопределенные значения (“null” или “undefined”). Автоматической отправки решения клиенту не происходит; для загрузки решения в браузер требуется отправка отдельного запроса. В этом смысле решенная задача вообще никак выделена и не отличается от решенной – с точки зрения системы и то и другое есть один конфигурационный файл в формате JSON.

Для работы всей системы необходимо разместить где-то стартовые настройки всего приложения (где находятся хранилища данных, где находятся математические алгоритмы, где находится Python - код). Текстовый файл с такими настройками уместно расположить, например, в главном каталоге /bin веб - сервера.



DOI: 10.7256/2305-6061.2016.1.17997

### 4. Хранилища данных

Важно подчеркнуть, что формат JSON используется не только для передачи данных, описывающих конфигурацию решаемой задачи, но и для хранения всех сопутствующих данных. Из Рис. 1 видно, что нижний слой трехзвенной архитектуры программной среды состоит не только из вычислительной подсистемы, но из хранилища данных (текстовые файлы в формате JSON). Эти данные представляют собой как совокупность конфигураций задач, так и всевозможных сопутствующих технических данных из соответствующей предметной области, которые используются при моделировании. Например, когда описываемая архитектура применялась в задачах оценочного моделирования пилотируемой экспедиции на Луну, создавались и использовались хранилища данных по космическим средствам, бортовым системам, полетным операциям, и т.п.

Необходимо подчеркнуть, что практика разработки и эксплуатации программной среды показала, что использование традиционных СУБД, основанных на реляционной модели данных, здесь нецелесообразно. Реляционная модель представляется слишком жесткой для тех задач, для которых создавалась архитектура. Говоря кратко, в виде таблиц удобно представлять только очень ограниченные по типу массивы информации. В основном, эта модель данных подходит только для задач учета (продукции, персонала и т.п.).

Стандарт JSON (как «обезжиренный XML») соответствует древовидной (иерархической) организации данных. Практически, это соответствует понятию «объекта» в объектно – ориентированном программировании (ООП), что чаще всего и есть лучшая абстракция данных в задачах математического моделирования сложных систем. Ниже приведен пример небольшого JSON – описания объекта «Посадочная ступень лунного модуля при 2-пусковой схеме»:

```
{
   «description_»: «Posadochnaya stupen’ lunnogo korablya v
2-puskovoi skheme»,
   «type»: «Lander unit of Moon landing ship 2 start»,
   «name»: «LU_MLS2»,
   «dry_mass»: «5000.0»,
   «fuel_mass»: «10000.0»,
   «exhaust_velocity»: «3000»,
   «bs_list»: [«EngineSystem», «ComplementSystem»],
}
```




DOI: 10.7256/2305-6061.2016.1.17997

Как можно видеть, этот объект состоит из описания, указания типа данного космического модуля, его названия, сухой массы, массы топлива, требуемой скорости истечения топлива, и списка бортовых систем.

Хранилища можно организовывать произвольным образом. В принципе, это может быть просто один каталог в файловой системе, но разумнее сделать систему каталогов по тематическому признаку (отдельные каталоги).

## 5. Использование Python

Место языка программирования Python в описываемой программной среде следующее. Набор методов, которые (посредством технологии DWR) требуется удаленно вызывать на сервере, меняется в зависимости от конкретной реализации всей системы. Для этого приходится переписывать заново (и перекомпилировать) Java - класс диспетчера. Использование языка программирования Python позволило поступить более гибко; одновременно, заложена основа для дальнейшего развития архитектуры.

Гибкость достигается следующим образом. Класс диспетчера написан так, чтобы его перекомпиляция больше не требовалось («раз и навсегда»): набор методов этого классов фиксирован. Но это не отменяет возможности для расширения набора методов, доступных для удаленного вызова, вообще. Фактически, класс диспетчера имплементирует (реализует) всего один метод, который передает управление подсистеме центрального слоя (согласно Рис.1), реализованной на языке программирования Python.

Эта подсистема состоит просто из набора текстовых файлов с программным кодом на языке Python. Имя конкретной требуемой процедуры передается в среду Python как строка; набор параметров этой процедуры передается как массив строк. Как было отмечено выше, вызовы Python – процедур функционально сводятся к вызову вычислительной подсистемы (Octave) либо работе с хранилищем вспомогательных данных и конфигураций задач (Рис 2). Библиотека используемых вычислительных алгоритмов (файлы имеют стандартное для MATLAB \ Octave расширение ".m") рассматривается как часть хранилища данных.

Благодаря такой организации цепочки удаленных вызовов удалось достичь высокой гибкости и адаптивности архитектуры к совершенно различным предметным областям. Python – код можно оперативно обновлять, не затрагивая центральное ядро архитектуры (DWR - слой); компиляции этого кода не требуется – он вызывается на исполнение интерпретатором Jython.

Принципиально, описанная цепочка вызовов представляет собой одну из возможных реализации концепции удаленного вызова процедур RPC (Remote procedure call) [15]. Роль языка программирования Python в дальнейшем развитии архитектуры описана ниже, в разделе «Заключение о обсуждение результатов».



DOI: 10.7256/2305-6061.2016.1.17997

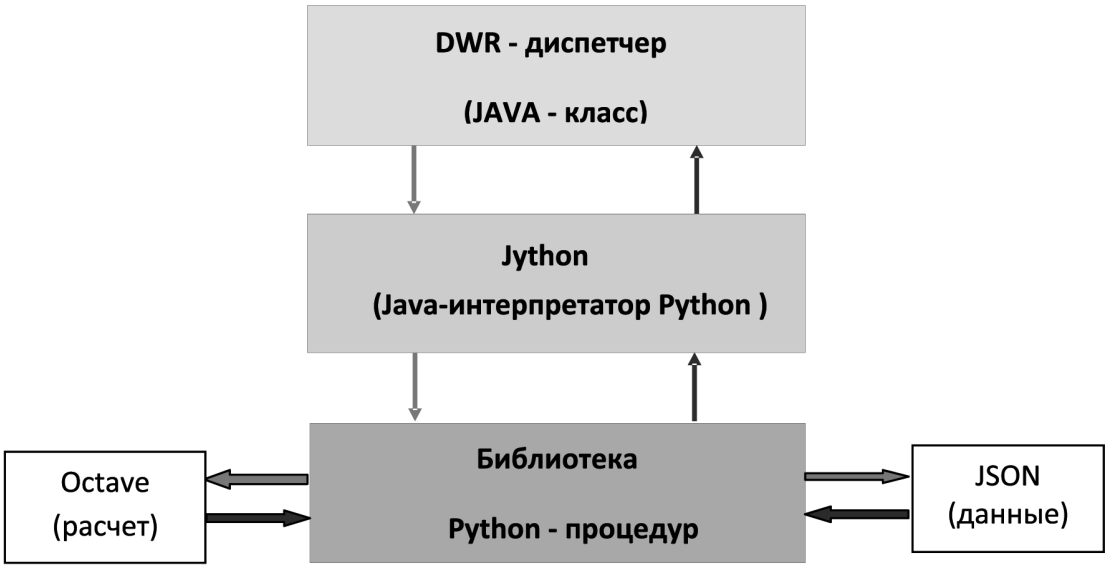


Рис.2. Детализация среднего и нижнего слоев архитектуры

## 6. Интерфейс пользователя и графический редактор

Как отмечено во Введении, вся описываемая программная среда разрабатывалась, по сути, для того чтобы создать инструментарий, позволяющий специалисту создавать средствами *визуального проектирования* описания задач в его предметной области. Поэтому графический редактор – функционально самая важная часть архитектуры.

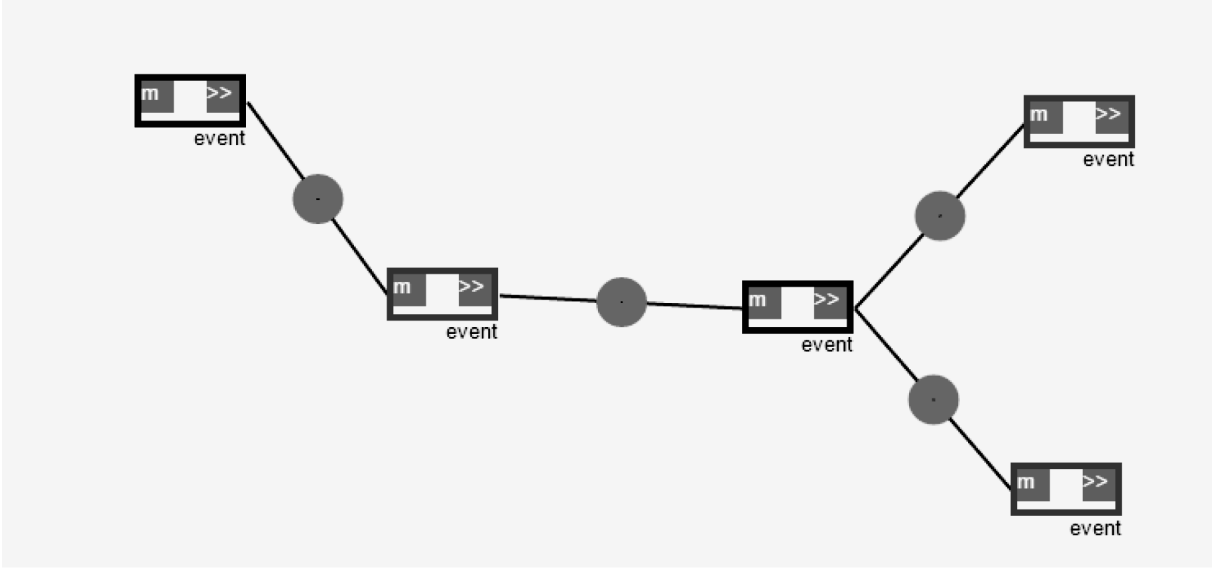


Рис. 3. Представление типовой конфигурации задачи в среде браузерного графического редактора


DOI: 10.7256/2305-6061.2016.1.17997

На Рис. 3 приведен пример графического представления типичной конфигурации в системе оценочного моделирования космический миссий. Конфигурация создается пользователем из некоторого набора графических примитивов. В данной реализации программной среды использовались всего два примитива – «событие» (или, что то же – «полетная точка») и «операция». Событию соответствует графический примитив «прямоугольник», операция создается как отрезок с кружочком, соединяющий два события. В другой реализации среды, созданной для моделирования производственно – экономических задач, прямоугольникам соответствуют промышленные предприятия, а отрезкам – поставки продукции между предприятиями.

В обоих этих случаях формальное описание задачи представляет собой, в целом, математический граф, т.е. совокупность узлов и ребер. Узлы графа (прямоугольники) создаются по принципу “drag and drop” («захвати и урони»), как и в большинстве коммерческих средств визуального проектирования (например, упомянутый во Введении Simulink или Visio компании Microsoft). Чтобы создать ребро графа («операцию» или «поставку продукции») надо схватить мышью символ «>>» на графическом примитиве узла и перетащить его до следующего (целевого) узла – прямоугольника. Принципиально, узлы графа могут представлять объекты произвольной природы, а ребра графа – те или иные бинарные связи таких объектов.

Графические примитивы создаются при помощи открытой JavaScript – библиотеки Raphael [16]. В целом рынок открытых JavaScript – библиотек бурно развивается в последние годы, что также послужило, на начальном этапе разработки архитектуры, дополнительным стимулом построить искомую архитектуру специализированных систем математических расчетов на основе веб-технологий.

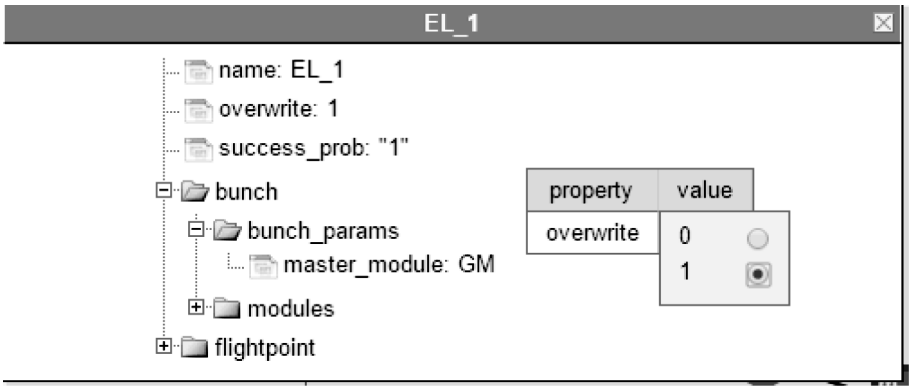


Рис. 4. Меню редактирования параметров объекта

Главная особенность графических примитивов, создаваемых в среде графического редактора – к ним можно привязывать наборы данных и редактировать их (Рис. 4). Как можно видеть из рисунка, набор данных, привязанных к отдельному примитиву (соответственно – объекту, который этот примитив представляет) имеет древовидную структуру; допустима произвольная глубина вложенности данных, иными словами – узлам и ребрам графа задачи можно сопоставлять объекты произвольной сложности.



DOI: 10.7256/2305-6061.2016.1.17997

В совокупности с механизмом построения графа, это составляет полный механизм создания и редактирования конфигураций задач. Как отмечено выше, созданную описанным способом конфигурацию можно средствами библиотеки Yahoo UI конвертировать в JSON – строку, а далее, на основе описанного выше механизма удаленного вызова соответствующей процедуры (реализованной на языке Python), сохранить как текстовый файл в хранилище конфигураций. Аналогичным путем, можно вызвать процедуру обработки ранее сохраненной конфигурации (расчет), а также загрузки ранее созданной конфигурации (неважно – обработанной или нет).

На данный момент реализован графический редактор, предполагающий совокупную структуру данных, описывающих задачу в виде графа. Принципиально, технологии Raphael и Yahoo UI позволяют разработать средство визуального проектирования для любых других сложных структур данных.

## 7. Пример решаемой задачи

Эта задача была реализована в демо-версии программной среды. В задаче производится расчет выпусков продуктов у «объектов» (предприятия космической отрасли) по объемам заказа, заданным конечным потребителем (государством), при следующих ограничениях на конфигурацию связей:

1. в конфигурационном графе нет замкнутых петель: нельзя, двигаясь по поставкам, придти снова в тот же объект;
2. нет поставки «самому себе»;
3. два любых объекта могут быть связаны только одной поставкой;
4. каждый объект производит только один продукт.

Помимо материальных потоков продукции в экономической системе – ракетно-космической отрасли – есть и денежные, которые текут в обратную сторону. То есть, за поставки назначаются цены, причем делают это субъекты – собственники предприятий-объектов (один субъект может владеть как одним, так и несколькими предприятиями). Цены не могут назначаться совсем произвольно: есть ограничения, связанные с максимумом доходности или, что то же, прибыльности. То есть отношение прибыли к расходу для данного субъекта не превышает некоторой заданной государством нормы. В остальном же субъекты стремятся к максимизации собственной прибыли.

Помимо объектов-фирм в системе имеется конечный потребитель (-ли). Под последним понимается либо Рынок, либо Государство, либо оба, если рассматривается два потребителя. Конечный потребитель формирует заказ на продукцию, создаваемую фирмами, в зависимости от типа, различается лишь способом этого формирования. Государство задает объемы заказа, согласно строго установленным планам (принцип нормативного планирования). Рынок же регулирует объемы в зависимости от цен на продукцию согласно функции спроса на тот или иной вид продукции. Обе ситуации могут быть описаны однотипным способом с помощью функции спроса, в случае Государства это постоянная функция объема от цены. В текущей версии это пока что не реализовано.

Таким образом, фирмы-объекты связываются между собой и с конечным потреби-




DOI: 10.7256/2305-6061.2016.1.17997

телем поставками продукции, при этом, в общем случае может получиться ситуация, когда не вся продукция какого-либо объекта потреблена другими фирмами и конечным потребителем, то есть могут возникнуть излишки продукции. Наличие таких остатков означает несбалансированность производства и потребления внутри данной экономической системы, чего следует избегать. Таким образом, возникает задача сбалансировать производство и потребление за счет подбора объемов выпуска продукции для каждой фирмы. Иначе говоря, задача внутриотраслевого баланса. По сравнению с межотраслевым балансом, где каждая отрасль производит уникальную продукцию, здесь ситуация в общем случае не такова: несколько фирм могут производить одно и то же. Это приводит к невозможности однозначно решить данную задачу, к тому же создает трудности программистского характера по представлению множества решений. Учитывая сказанное, было принято решение упростить задачу так, чтобы она имела единственное решение, то есть свести ее к межотраслевому балансу, введя ограничения на конфигурацию связей-поставок.

В данной постановке задачи конечным потребителем является Государство, оно задает заказ на объемы того или иного вида продукции, по этим заказам и, зная коэффициенты прямых затрат, можно рассчитать объемы выпусков всех фирм-объектов.

Обозначим через $N$ – количество фирм-объектов, также пронумеруем их каким-либо образом; далее, пусть $X_i$ – выпуск $i$-го объекта, то есть количество продукции, выпущенной объектом с номером $i$; $X_{ij}$ – объем поставки от $i$-го объекта $j$-ому объекту, то есть количество продукции, поставленной от $i$-го объекта $j$-ому объекту; $a_{ij}$ – коэффициент прямых затрат продукции $i$-го объекта в выпуске $j$-ого объекта; $Y_i$ – заказ на продукцию $i$-го объекта со стороны конечного потребителя. Если какая-то поставка отсутствует, то соответствующий ей объем принимается равным нулю, аналогично, если заказ на продукцию данного объекта отсутствует, то считается равным нулю. Принимая всё сказанное, имеем следующие уравнения:

$X_i = \sum_{j=1}^{N} X_{ij} + Y_i$ – уравнения баланса потоков продукции,

$X_{ij} = a_{ij} X_j$ – определения коэффициентов прямых затрат,

для всех $i, j$ от 1 до $N$.

Получается система линейных уравнений относительно переменных $X_i, X_{ij}$, которые однозначно разрешаются, если заданы остальные величины ($Y_i$, $a_{ij}$).

В итоге, данная (демонстрационная) реализация программной среды использует следующие входные и выходные данные (они входят в JSON - конфигурации задачи и решения):

- Параметры, задаваемые пользователем:
  - «постоянные издержки», «технологические издержки» - для каждого объекта-предприятия (квадратик);



DOI: 10.7256/2305-6061.2016.1.17997

- «коэффициент прямых затрат» для поставки (отрезок с кружочком);
- «заказ» - для финальной поставки (конечному потребителю).

• Рассчитываемые параметры:
- «вектор выпуска» - для каждого объекта – предприятия;
- «Объем» - для каждой поставки, включая финальную поставку конечному потребителю.



**Заключение**

Описанная архитектура отвечает функциональным требованиям, заявленным во Введении. Можно повторно привести ключевое предложение из этого раздела: «Кратко и на качественном уровне проблему можно описать так: требуются программные средства, аналогичные популярной системе MATLAB [2] в совокупности с подсистемой графического редактирования задач Simulink [3], но в сочетании с некоторым хранилищем данных и описаний задач, при этом основанные на бесплатном (свободно распространяемом) программном обеспечении».

В целом, такая программная среда подходит для средств поддержки принятия решений (СППР [17]), предполагающих математические расчеты на основе некоторой, потенциально произвольно сложной и достаточно большой по размеру, структуры данных из соответствующей предметной области. Конкретно, данная архитектура была применена для задач моделирования производственно – экономических задач в космической отрасли и задач сценарного моделирования сложных космических экспедиций [1].

Результаты, описанные в статье, были ранее частично опубликованы в ИПМ им. М.В. Келдыша РАН [1], а также доложены на двух международных конференциях ([20, 21]).

***Библиография :***


1. Р. Д. Зухба, П. В. Куракин, Г. Г. Малинецкий, С. А. Махов, Н. А. Митин, А. П. Сорокин. "Программно – математические комплексы систем поддержки принятия решений нового поколения". – Препринт Института прикладной математики им. М. В. Келдыша РАН № 59, 2014 г. – 33 с.
2. МАТЛАБ (статья в Wikipedia) https://ru.wikipedia.org/wiki/MATLAB.
3. Simulink (статья в Wikipedia) https://en.wikipedia.org/wiki/Simulink.
4. Архитектура клиент – сервер (статья в Wikipedia) https://ru.wikipedia.org/wiki/Клиент-сервер.
5. Формат описания структурированных данных JSON (статья в Wikipedia): http://ru.wikipedia.org/w/index.php?title=JSON.
6. Сетевая модель ISO (статья в Wikipedia) https://ru.wikipedia.org/wiki/Сетевая_модель_OSI.
7. HTTP (статья в Wikipedia) https://ru.wikipedia.org/wiki/HTTP.
8. Apache Tomcat (статья в Wikipedia) https://ru.wikipedia.org/wiki/Apache_Tomcat.
9. Сериализация (статья в Wikipedia) https://ru.wikipedia.org/wiki/Сериализация.



DOI: 10.7256/2305-6061.2016.1.17997

10. Yahoo! UI Library (статья в Wikipedia) https://ru.wikipedia.org/wiki/Yahoo!_UI_Library.
11. DWR (статья в Wikipedia) https://ru.wikipedia.org/wiki/DWR.
12. Сервлет (статья в Wikipedia) https://ru.wikipedia.org/wiki/Сервлет_(Java).
13. GNU Octave (статья в Wikipedia) https://ru.wikipedia.org/wiki/GNU_Octave.
14. Jython (статья в Wikipedia) https://ru.wikipedia.org/wiki/Jython.
15. Удалённый вызов процедур (статья в Wikipedia) https://ru.wikipedia.org/wiki/Удалённый_вызов_процедур.
16. Официальный веб-сайт проекта Raphael http://raphaeljs.com/.
17. Система поддержки принятия решений (статья в Wikipedia) https://ru.wikipedia.org/wiki/Система_поддержки_принятия_решений.
18. Java (статья в Wikipedia) https://ru.wikipedia.org/wiki/Java.
19. Программирование и научные вычисления на языке Python https://ru.wikiversity.org/wiki/Программирование_и_научные_вычисления_на_языке_Python
20. Куракин П.В., Малинецкий Г.Г., Митин Н.А., Махов С.А. «MATLAB – Based Software for Decision Support Systems». Proceedings of International Conference on Computer Technologies in Physical and Engineering Applications (ICCTPEA 2014). СПб.: IEEE Catalog number CFP14BDA-USB, 2014. Russia, Saint-Petersburg, June 30 — July 4, 2014. С. 93.
21. Куракин П. В., Малинецкий Г. Г., Митин Н. А., Махов С. А., Барыкина М. Н., Зухба Р. Д. «Программно-математические комплексы поддержки принятия решений в космической отрасли». Управление развитием крупномасштабных систем (MLSD'2015): Восьмая международная конференция, 29 сент.-1 окт. 2015 г, ИПУ им. В. А. Трапезникова РАН. С. 12.

***References:***

1. R. D. Zukhba, P. V. Kurakin, G. G. Malinetskii, S. A. Makhov, N. A. Mitin, A. P. Sorokin. "Programmno – matematicheskie kompleksy sistem podderzhki prinyatiya reshenii novogo pokoleniya". – Preprint Instituta prikladnoi matematiki im. M. V. Keldysha RAN № 59, 2014 g. – 33 s.
2. MATLAB (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/MATLAB.
3. Simulink (stat'ya v Wikipedia) https://en.wikipedia.org/wiki/Simulink.
4. Arkhitektura klient – server (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Klient-server.
5. Format opisaniya strukturirovannykh dannykh JSON (stat'ya v Wikipedia): http://ru.wikipedia.org/w/index.php?title=JSON.
6. Setevaya model' ISO (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Setevaya_model'_OSI.
7. HTTP (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/HTTP.
8. Apache Tomcat (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Apache_Tomcat.
9. Serializatsiya (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Serializatsiya.
10. Yahoo! UI Library (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Yahoo!_UI_Library.
11. DWR (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/DWR.
12. Servlet (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Servlet_(Java).



DOI: 10.7256/2305-6061.2016.1.17997

13. GNU Octave (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/GNU_Octave.
14. Jython (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Jython.
15. Udalennyi vyzov protsedur (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Udalennyi_vyzov_protsedur.
16. Ofitsial'nyi veb-sait proekta Raphael http://raphaeljs.com/.
17. Sistema podderzhki prinyatiya reshenii (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Sistema_podderzhki_prinyatiya_reshenii.
18. Java (stat'ya v Wikipedia) https://ru.wikipedia.org/wiki/Java.
19. Programmirovanie i nauchnye vychisleniya na yazyke Python https://ru.wikiversity.org/wiki/Programmirovanie_i_nauchnye_vychisleniya_na_yazyke_Python
20. Kurakin P.V., Malinetskii G.G., Mitin N.A., Makhov S.A. «MATLAB – Based Software for Decision Support Systems». Proceedings of International Conference on Computer Technologies in Physical and Engineering Applications (ICCTPEA 2014). SPb.: IEEE Catalog number CFP14BDA-USB, 2014. Russia, Saint-Petersburg, June 30 — July 4, 2014. S. 93.
21. Kurakin P. V., Malinetskii G. G., Mitin N. A., Makhov S. A., Barykina M. N., Zukhba R. D. «Programmno-matematicheskie kompleksy podderzhki prinyatiya reshenii v kosmicheskoi otrasli». Upravlenie razvitiem krupnomasshtabnykh sistem (MLSD'2015): Vos'maya mezhdunarodnaya konferentsiya, 29 sent.-1 okt. 2015 g, IPU im. V. A. Trapeznikova RAN. S. 12.




DOI: 10.7256/2305-6061.2016.1.17997