ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 59 за 2014 г.

**Зухба Р.Д., Куракин П.В., Малинецкий Г.Г., Махов С.А., Митин Н.А., Сорокин А.П., Торопыгина С.А.**

# Программно-математические комплексы систем поддержки принятия решений нового поколения

***Рекомендуемая форма библиографической ссылки:*** Программно-математические комплексы систем поддержки принятия решений нового поколения / Р.Д.Зухба [и др.] // Препринты ИПМ им. М.В.Келдыша. 2014. № 59. 32 с. URL: http://library.keldysh.ru/preprint.asp?id=2014-59

**О р д е н а  Л е н и н а**
**ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ**
**имени М.В.Келдыша**
**Р о с с и й с к о й  а к а д е м и и  н а у к**

**Р.Д. Зухба, П.В. Куракин, Г.Г. Малинецкий, С.А. Махов, Н.А. Митин, А.П. Сорокин, С.А. Торопыгина**

# Программно-математические комплексы систем поддержки принятия решений нового поколения

**Москва — 2014**

Р.Д. Зухба, П.В. Куракин, Г.Г. Малинецкий, С.А. Махов, Н.А. Митин,
А.П. Сорокин, С.А. Торопыгина

# Программно-математические комплексы систем поддержки принятия решений нового поколения

## АННОТАЦИЯ

В работе изложены принципы организации и функционирования программно-математических комплексов систем принятия решений нового поколения, разработанные авторами и реализованные в виде пилотной версии такого комплекса. Изложена мотивировка создания комплексов такого класса, описана ожидаемая область применения.

*Ключевые слова:* программные комплексы, трехслойная архитектура, открытый код, веб-технологии, слой веб-интерфейса, слой-посредник, вычислительный слой, конфигурация задачи, конфигурация решения, архив конфигураций.

R.D. Zukhba, P.V. Kurakin, G.G. Malinetckii, S.A. Makhov, N.A. Mitin,
A.P. Sorokin, S.A. Toropygina

# New generation software for decision support systems

## ABSTRACT

Organizational and functional principles of new generation of mathematical software developed and implemented by authors for decision support systems is described. The reason for such software and expected application area is also described.

*Key words:* software packages, three – layered architecture, open source, web technologies, web interface layer, intermediate layer, calculation layer, problem configuration, solution configuration, archive for configurations.

Работа выполнена при поддержке РФФИ (проекты 12-06-00402, 13-01-00617).

Содержание



## 1. Введение и постановка задачи

В настоящее время становится актуальным определенный класс задач математического моделирования, встречающихся в работе управленческих администраций различного типа и уровня (государственных – как федеральных, так и региональных; корпоративных; и т.п.). В качестве ключевой особенности этого класса задач можно указать следующий примерный список характеристик:

- задача характеризуется наличием достаточно большого набора некоторых объектов различной природы и их связей; в качестве объектов могут выступать промышленные предприятия, отдельные цеха и/или этапы технологической переработки исходных материалов и компонентов, этапы полета космического аппарата при реализации различных миссий,

отдельные этапы транспортных перевозок при решении логистических задач и т.п.;

- связи между объектами имеют преимущественно бинарный характер (связи типа «объект – объект»);
- объекты, рассматриваемые моделью, могут быть нескольких типов; также, нескольких типов могут быть и бинарные связи между объектами;
- как рассматриваемые объекты, так и связи между ними обладают наборами количественных характеристик (параметров); представители одного типа объектов (связей) имеют одинаковые наборы параметров;
- указанные параметры объектов и связей желательно было бы гибко и оперативно настраивать, включая, возможно, переформирование самих наборов;
- полное описание задачи моделирования представляет собой некоторое перечисление всех вовлеченных объектов и связей и задание (как полное, так и неполное) соответствующих параметров;
- целью математического моделирования является, как правило, вычисление тех параметров, значения которые остались незаданными на этапе формулировки задачи;
- крайне желательным представляется наличие некоторого графического интерфейса пользователя для быстрого и удобного создания и редактирования ранее созданных описаний конкретных вариантов задачи, представляющих интерес для указанных выше администраций;
- работа пользователя с системой моделирования представляет собой некоторый итеративный процесс, когда пользователь последовательно формирует описания задач, затем отправляет их на вычисление, потом может ознакомиться с полученным результатом расчетов;

- желательно также иметь некоторым образом систематизированное хранилище, как для описаний исходных задач, так и для описаний, найденных в результате расчетов решений.

Перечисленные характеристики можно выразить кратко: описание задач интересующего класса представляет собой *математический граф*, узлы которого есть некоторые объекты, а ребра есть те или иные связи между объектами; необходима среда, поддерживающая создание, редактирование, хранение описаний таких задач; алгоритмы и программы решения таких задач с широким привлечением различных методов численного анализа подразумеваются.

Учитывая эти требования, были сформулированы следующие принципы построения и работы программно-математического комплекса системы поддержки принятия решений, которые затем были реализованы в пилотной версии такого комплекса.

1. Необходим адекватный и соответствующий структурной сложности задач описанного выше типа *язык описания* каждой модельной задачи. Этот язык следует рассматривать как некоторый *текстовый формат* описания структурированных данных. Такой формат, с одной стороны, должен быть достаточно ёмким, чтобы быть в состоянии отображать сложность моделируемой задачи. С другой стороны, формат должен быть достаточно понятным, легким в изучении пользователем и допускающим простую технологию создания и модификации.

2. Создаваемое средство моделирования должно располагать достаточно современным графическим интерфейсом, позволяющим пользователю создавать, редактировать, сохранять в архиве (см. далее), а также загружать в пользовательский интерфейс из этого архива конфигурации исходных задач и конфигурации решений.

3. Создание и модификация (редактирование) конфигураций исходных задач в графическом интерфейсе пользователя, с одной стороны, и в текстовом

редакторе, с другой, должны быть в широком смысле взаимозаменяемыми и симметричными механизмами. Создание конфигураций в графическом интерфейсе более подходит для начинающего пользователя либо для первоначального наброска задачи и быстрого получения ее решения с целью предварительного («оценочного») ознакомления с ожидаемыми порядками величин.

Создание и модификацию конфигурации исходной задачи в текстовом редакторе можно рассматривать как способ работы для опытного («продвинутого») пользователя либо как детализацию задачи после оценочного этапа.

4. Созданные конфигурации должны храниться в некотором архиве с удобными механизмами поиска.

5. С точки зрения архитектуры, желательно, чтобы программный комплекс имел хорошую модульность, и, чтобы функции графического интерфейса и выполнения вычислений были разнесены.

6. С точки зрения реализации указанной архитектуры, система моделирования должна опираться на решения, минимизирующие стоимость разработки. Этого возможно добиться путем ставки на интенсивное использование всевозможных библиотек готового открытого кода («open source»), имеющихся на рынке, и их интеграцию.

В ходе проведения исследований были найдены пути реализации этих принципов, они описаны в следующем разделе.

# 2. Организация пилотного программного комплекса

## *2.1. Язык описания задач*

В качестве требуемого формата структурированного описания задач был выбран популярный (и получающий все большее распространение в информационных технологиях) язык описания структурированных данных

JSON [1]. Фактически, именно выбор этого формата позволил одновременно решить проблему удобного текстового описания моделируемых задач (и их решений) и организовать модульную архитектуру системы моделирования на основе бесплатных *open source* – библиотек.

В дальнейшем описание модельной задачи в виде текстового документа в том или ином формате (не обязательно JSON) называется *конфигурацией задачи*. Также надо иметь в виду, что введенное таким образом понятие конфигурации далее применяется не только к описанию *исходной задачи*, но и к описанию *решения*, полученного как результат применения тех или иных вычислительных алгоритмов.

Формат разметки структурированных данных JSON представляет собой «упрощенный» XML (Extendable Markup Language). Он изначально был разработан для хранения и передачи данных, содержащихся на веб-страницах в виде объектов языка программирования сценариев веб-страниц JavaScript (JSON = JavaScript Object Notation). В настоящее время этот формат нашел широчайшее применение за пределами веб-технологий.

В частности, существуют свободно распространяемые библиотеки для преобразования данных MATLAB в формат JSON и обратно (одна из таких библиотек используется в программном комплексе). Наличие такой специальной и бесплатной MATLAB-библиотеки для обмена данными в формате JSON представляется особенно удачным. Фактически, именно этот факт позволил спроектировать и реализовать пилотную версию требуемого программного обеспечения, в которой осуществляется следующая простая и наглядная схема движения данных:

- описание модели, созданное в среде веб-браузера и изначально существующее в виде набора объектов языка JavaScript (встроенного в любой веб-браузер), *сериализуется* (конвертируется) в JSON – строку;
- JSON – строка *маршализируется* (транспортируется по сети) в вычисляющую среду (MATLAB); вычисляющая среда выполняет

*десериализацию* (операция обратного конвертирования) полученной строки в объекты собственного языка программирования и выполняет требуемые вычисления;

- полученное в среде MATLAB решение задачи (в виде набора объектов языка программирования среды MATLAB) аналогичным образом сериализуется в JSON – строку и маршализируется обратно в среду веб-браузера, где происходит еще одна *десериализация* из JSON – строки в объекты JavaScript и их графическое отображение средствами браузера.

## *2.2. Организация модульной архитектуры программного комплекса – общее описание*

Сформулированное выше требование модульности программно-математического комплекса было выполнено путем организации трехслойной архитектуры. Пилотная версия комплекса состоит из следующих трех уровней: 1) пользовательский интерфейс, 2) вычислительный слой, 3) слой-посредник (веб-сервер) (рис. 1).

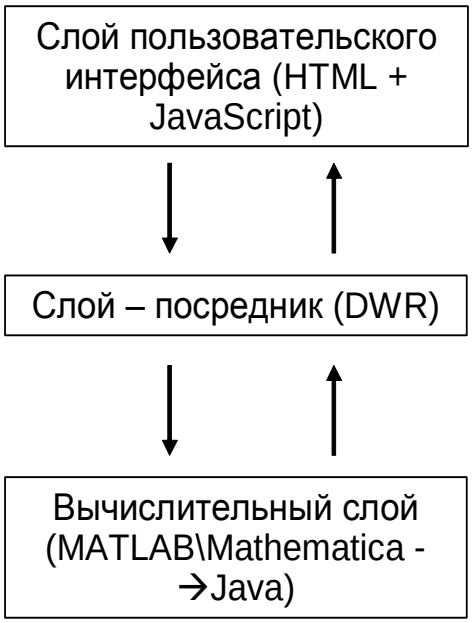


Рис. 1. Архитектура комплекса.

В текущей версии комплекса пользователь работает в веб-браузере Mozilla FireFox (для последующих версий планируется введение возможности работы с произвольным браузером: Internet Explorer, Opera, Google Chrome и т.п.).

Вычислительный блок опирается на готовые численные алгоритмы и исходный код программного пакета MATLAB, но реализует концепцию «отторжения» этих алгоритмов и кода от самого пакета. Это означает, что комплекс используется без непосредственного наличия в компьютерной системе установленного пакета MATLAB (см. детали ниже).

В пилотной версии предусмотрено решение только одной модельной задачи; соответственно, используются фиксированные вычислительные алгоритмы, реализованные на языке программирования пакета MATLAB. Принципиально возможно значительное расширение репертуара моделируемых задач; возможность выбора одной из нескольких модельных задач предполагается добавить в последующих версиях комплекса.

Особенность вычислительного слоя программного комплекса состоит в том, что программный код вычислительных алгоритмов, непосредственно задействованных в математической модели, изначально написан на языке программирования MATLAB, но используется, в конечном счете, в виде Java – классов. Компиляция проводится специальной утилитой пакета MATLAB (MATLAB Builder JA, которая входит в стандартный дистрибутив пакета MATLAB). Важно, что полученные после компиляции Java – классы можно использовать самостоятельно, без полномасштабной установки всего пакета MATLAB. Для этого необходимо вместе с классами предоставить их пользователю Java - архив `javabuilder.jar` (входящий в стандартный дистрибутив пакета MATLAB), а также установить в операционной системе пользователя среду MCR (Matlab Runtime Compiler) – текущая лицензия использования пакета MATLAB это позволяет.

В качестве «посредника» между пользовательским слоем и вычислительным слоем используется бесплатно распространяемый веб-сервер Tomcat совместно с (также бесплатной) библиотекой DWR (Direct Web Remoting), осуществляющий максимально простой способ обмена информацией между средами Java и JavaScript. Фактически, на основе библиотеки DWR создается небольшое серверное приложение, устанавливаемое на веб-сервере Tomcat. Это приложение, с одной стороны, по протоколу HTTP обменивается информацией с веб-страницей пользовательского интерфейса (т.е. со средой JavaScript), с другой стороны – может обмениваться информацией с вычислительным слоем в рамках среды Java.

Поскольку единственным ограничением на этот обмен с вычислительным слоем является только использование среды Java (то есть – Java классов), то в принципе выбор пакета MATLAB в качестве единственного возможного вычислительного слоя не является критическим. В дальнейшем планируется детальнее изучить возможности применения в вычислительном слое альтернативных платформ и библиотек численных методов (в частности, проект Octave, являющийся полностью свободным и бесплатным аналогом пакета MATLAB).

Последней ключевой составляющей технологией программного комплекса является формат описания структурированных данных JSON. Передача данных между уровнями архитектуры программного комплекса (изображенная стрелками на рис. 1) осуществляется путем передачи строк в формате JSON по протоколам TCP/IP и HTTP (который сам также использует TCP/IP). Именно поэтому программный комплекс работает как веб-приложение под управлением веб-сервера.

Обмен между веб-браузером и веб-сервером обеспечивает протокол HTTP, а между веб-сервером и вычислительным слоем – протокол TCP/IP. Механизм коммуникации браузера с сервером является стандартным, а

механизм коммуникации сервера с вычислительным слоем был разработан нами отдельно (детали см. в подразделе 2.3). Фактически, протокол TCP/IP (либо опирающийся на него HTTP) и формат JSON выступают в роли универсального (в высокой степени) средства сочленения различных информационных технологий.

Использование в качестве среды пользовательского интерфейса веб-браузера, а также реализация 2-го (посреднического) и 3-го (вычислительного) слоев архитектуры приложения на основе технологии Java делает программный комплекс, в принципе, платформенно-независимым. В частности, используемый в пилотной версии бесплатный веб-сервер Tomcat имеет реализации для различных операционных систем (включая семейство ОС Windows) и активно использует среду Java.

Для текущей версии испытания мультиплатформенности не проводились, работа комплекса описана в предположении использования ОС Windows XP.

В целом, описанная схема позволила кардинально сократить ресурсы, необходимые для разработки комплекса, в то же время предоставить развитый графический интерфейс пользователя и сопрячь его с богатыми возможностями вычислительных методов пакета прикладных математических вычислений MATLAB. Представление вычислительных алгоритмов в Java – виде и позволяет провести такое сопряжение за счет максимального использования современных веб-технологий и свободно распространяемых библиотек прикладного программирования.

### *2.3. Технология разработки графического интерфейса пользователя*

Несколько более подробно следует остановиться на разработке графического интерфейса пользователя, потому что традиционно именно графический интерфейс пользователя являлся самой трудоемкой либо одной из самых трудоемких составляющих в разработке stand-alone приложений. Для примера достаточно вспомнить историю коммерческого успеха линейки

операционных систем Windows компании Microsoft. Именно удобные, эргономичные пользовательские интерфейсы, созданные этой компанией, обеспечили ей занятие значительной доли на рынке информационных технологий. Большую роль в истории этого успеха имела также библиотека MFC (Microsoft Foundation Classes), включенная в состав среды разработки Visual C++. Значительную часть этой библиотеки составляют как раз базовые компоненты для разработки графических интерфейсов пользователя.

Можно сказать, что за последние годы JavaScript, как язык динамических сценариев для веб-приложений, совершил целую революцию в скорости и стоимости разработки графических интерфейсов в контексте веб-приложений. Несмотря на то, что этот язык считается весьма неоднозначным, с точки зрения устойчивости, переносимости и безопасности кода, де факто бурное развитие WWW начиная с 90-х гг. XX века сделало его практически обязательным стандартом для всех веб-браузеров любых производителей.

Важно, что в настоящее время уже недостаточно использовать «голый» JavaScript, реальная практика разработки активно опирается на расширения JavaScript в виде свободно распространяемых библиотек, написанных на самом JavaScript, но предоставляющих собственный API (Application programming Interface). Такие библиотеки решают много разных задач. В контексте описываемой разработки можно упомянуть следующие две задачи:

- повышение переносимости и кросс-платформенности кода; реальный «голый» JavaScript номинально – один и тот же язык в различных браузерах, фактически его реализации сильно различаются; библиотеки-надстройки избавляют разработчика от необходимости изучать многочисленные особенности исполнения конкретных методов JavaScript в различных браузерах: программист использует универсальный вызов, предоставляемый API, а полиформизм исполнения (конкретные реализации для разных браузеров) метода обеспечивает сама библиотека;

- повышение качества, эргономичности и профессионализма в эстетическом исполнении графических элементов интерфейса.

Описываемый программный комплекс также использует на стороне веб-браузера две популярные библиотеки – надстройки: Yahoo UI [3] и Raphael [4]. Первая библиотека предназначена для создания эргономичных элементов пользовательского интерфейса, исполненных в соответствии с современными требованиями дизайна; вторая – для работы с векторной графикой. Эти библиотеки также являются бесплатными и позволили кардинально сократить время и стоимости разработки при одновременном повышении качества исполнения по сравнению с использованием «голого» JavaScript.

### *2.4. Файловая структура программного комплекса*

Напомним, что мы имеем дело с веб-приложением, работающим под управлением веб-сервера. Поэтому корневой каталог `/<application>` приложения находится внутри каталога этого веб-сервера, в папке `<tomcat>/webapps`, т.е. там же, где должны находиться все веб-приложения, установленные на этом сервере.

Полное описание процесса установки и запуска программного комплекса можно найти в выпущенной документации на комплекс. Здесь же имеет смысл кратко описать наиболее важные для понимания работы приложения файлы и каталоги.

В целом, эта структура, с учетом некоторых добавлений, является типичной для веб-приложений.

- Файл `/index.html`. Это собственно веб-страница с пользовательским интерфейсом. Для ее открытия необходимо пользоваться веб-браузером (для пилотной версии обязателен Mozilla Firefox), по адресной строке `http://localhost:8080/<application>` открывается страница, как на рис. 2 (см. ниже на стр.17).

- Каталог `/WEB-INF`. Это непосредственное местонахождение посреднического веб-приложения, там расположены Java – классы, осуществляющие логику посреднического слоя; у пользователя нет необходимости явно обращаться к содержимому этого каталога.
- Каталог `/listener`. В этом каталоге находятся все исполняемые и вспомогательные файлы так называемого *слушающего процесса*. Напомним, смысл работы всего программного комплекса состоит в том, чтобы сочленить несколько готовых технологий, а именно – интерфейсные возможности HTML + JavaScript и вычислительные возможности MATLAB. Это сочленение выполняет веб-сервер Tomcat при помощи технологии, реализованной свободной библиотекой DWR. Веб-сервер – это посредник, который получает данные от веб-страницы интерфейса пользователя (в формате JSON), и передает эти данные вычислительным библиотекам пакета MATLAB. Эти библиотеки, как было указано выше, отторгнуты от самого пакета (весь пакет не нужен для работы комплекса), они работают как раз *в слушающем процессе*. Таким образом, вычислительный слой приложения расположен именно в этом каталоге. Слушающий процесс так назван потому, что слушает на отдельном программном порту команды, передаваемые по протоколу TCP / IP от посреднического слоя (расположенного в каталоге `/WEB-INF`) к вычислительному слою. Выполнять вычисления непосредственно в процессе веб-сервера нельзя, в соответствии как с общими принципами работы веб-серверов, так и с учетом вычислительной ресурсоемкости конкретно библиотек MATLAB. В дальнейшем, предполагается разработать так называемый *коннектор* – специальный компонент приложения для технологичного и потенциально многократного подключения к вычисляющему слою. Для запуска слушающего процесса нужно вызвать на исполнение командный файл `start_listener.bat`.

- Каталог `/listener/inputs`. Содержит конфигурационные файлы *задач* (в иной терминологии – «не обсчитанные» конфигурации). В этом каталоге сохраняются конфигурации задач, созданных пользователем при нажатии в интерфейсе кнопки *«сохранить задачу»* (см. рис. 2 и подраздел 3.2). Ключевая особенность системы моделирования состоит в том, что создавать либо изменять ранее созданные конфигурации можно не только посредством графического интерфейса, но и в любом текстовом редакторе, просто редактируя файлы, находящиеся в каталоге `/listener/inputs` (см. также подраздел 4.2).
- Каталог `/listener/outputs`. Содержит конфигурационные файлы *решений* (в иной терминологии – «обсчитанные» конфигурации). В этом каталоге сохраняются конфигурации обсчитанных задач после выполнения команд «расчет сохр.» и «расчет тек.» (см. рис. 2 и раздел 3).
- Каталог `/logs`. В этот каталог при работе приложения сохраняются логии (отчеты) компонентов программного комплекса за весь период их работы. Эти отчеты могут понадобиться при появлении рекламаций на программный продукт: в случае неправильной работы приложения разработчики запросят лог-файл за последние сутки. Пользователю необходимо знать, что приложение в процессе своей работы создает *два* лог-файла на каждые сутки. Первый лог-файл производится слушающим процессом и имеет шаблон названия `listener_yyyy.mm.dd.log`, второй производится процессом веб-сервера и имеет шаблон `dispatcher_yyyy.mm.dd.log`, где `yyyy.mm.dd` – шаблон текущей даты.
- Каталог `/matlab`. Этот каталог содержит файлы с программным кодом на языке программирования пакета MATLAB, т.н. m-функции. Это те же самые функции, которые используются программным комплексом для расчетов, но в комплексе они используются в виде Java – классов,

скомпилированных утилитой пакета MATLAB (эти классы расположены в каталоге `/listener/classes`). Но, точно так же, как пользователь может самостоятельно редактировать (включая создание «с нуля») конфигурационные файлы задач, он может самостоятельно выполнять вычисления над конфигурациями, но в этом случае ему понадобится полный дистрибутив пакета MATLAB (напомним, в самом комплексе произведено «отторжение» вычислительных библиотек этого пакета от самого программного продукта). Для выполнения вычислений нужно вызвать m-функции из каталога `/Matlab`, передавая им в качестве аргументов пути к соответствующим конфигурационным файлам (см. подраздел 4.2).

# 3. Графический интерфейс пользователя и работа с ним

## *3.1. Общие положения*

На данный момент создано две модификации программного комплекса для двух задач моделирования, соответственно. В дальнейшем планируется добиться большей гибкости в части возможности комплекса работать с разными моделями в рамках одной архитектуры.

Первая модификация предназначается для экономических задач. Поэтому объектами – узлами графа являются промышленные предприятия, а ребрами – поставки продукции от предприятия – производителя предприятию – потребителю (то есть, граф является ориентированным).

Вторая модификация комплекса предназначена для комплексного моделирования различных космических экспедиций. Здесь объектами – узлами графа являются *этапы полета* либо *вектора состояний* космического аппарата, а ребрами, соответственно – *переходные режимы* (между этапами) либо *полетные операции* (производящие преобразование из одного вектора состояния КА в другое).

Графические интерфейсы двух модификаций имеют различия, но общие принципы их работы остаются едиными. Как уже было указано, интерфейс работает в веб-браузере. Его общий вид приведен на рис. 2 (на примере первой модификации).

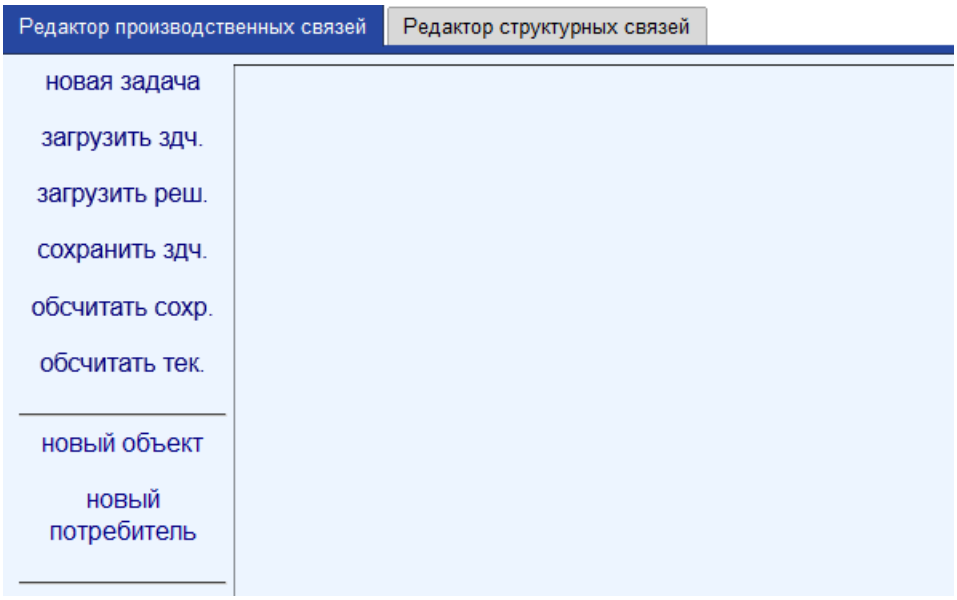


Рис. 2. Общий вид графического интерфейса пользователя.

### *3.2. Графический интерфейс и его использование*

Организацию графического интерфейса пользователя поясним на примере первой из указанных выше модификаций пилотной версии программно-математического комплекса.

В этой модификации графический интерфейс состоит из двух закладок: «Редактор производственных связей» и «Редактор структурных связей»

На рис. 3 изображена закладка редактора производственных связей с загруженной конфигурацией. Видно, что она состоит из двух полей: командное меню и поле холста.

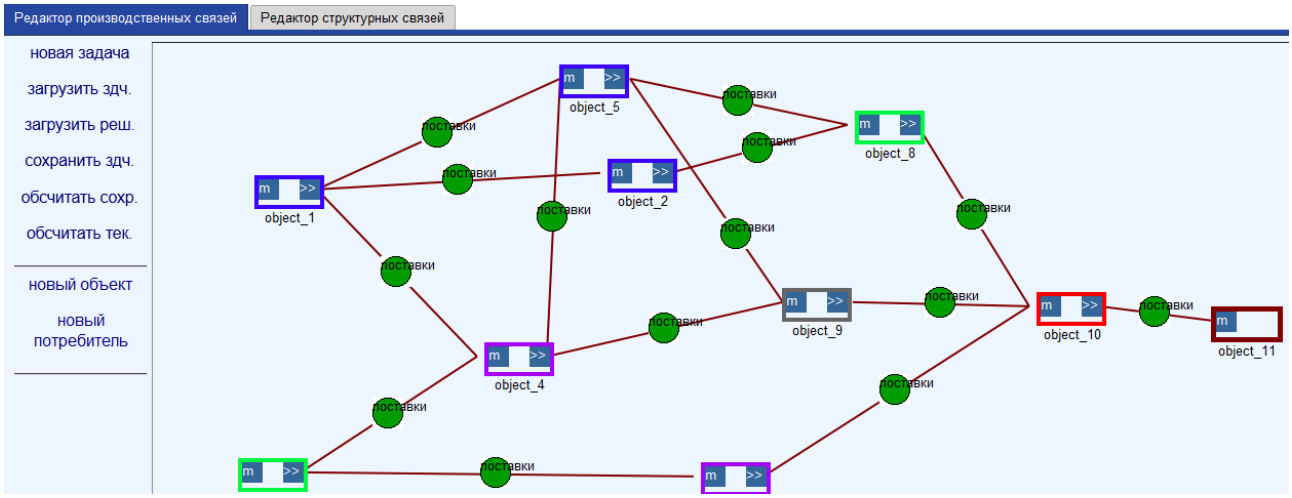


Рис. 3. Закладка редактора производственных связей; поле командного меню.

### *Закладка редактора производственных связей: поле командного меню*

Командное меню содержит в виде подсвеченных гиперссылок следующие команды:

- «новая задача»;
- «загрузить задачу»;
- «загрузить решение»;
- «сохранить задачу»;
- «обсчитать сохраненную задачу»;
- «обсчитать текущую задачу»;
- «новый объект»;
- «новый потребитель».

На рис. 4 изображено поле командного меню, после того как была выбрана команда «новый объект». После создания объектов или потребителей («объект» может как потреблять продукцию, так и производить, потребитель – только потреблять) пользователь мышкой перетаскивает их с поля командного меню на поле холста. «Захватив» мышкой значок «>>» в левой части прямоугольной иконки, символизирующей «объект», можно соединить два

«объекта» экземпляром бинарной связи, если перетащить «>>» от поставщика к потребителю.


Рис. 4. Создан новый «объект».

При клике по гиперссылке *«новая задача»* поле холста графического редактора очищается от всех объектов и связей. При этом делается автоматическая проверка на предмет сохранения текущей конфигурации.

При клике по гиперссылке *«загрузить задачу»* можно загрузить ранее созданную конфигурацию задачи. При этом в поле холста отобразится соответствующая система объектов и связей.

При клике по гиперссылке *«сохранить задачу»* сохраняется текущая конфигурация. Это же относится к случаю, если текущая конфигурация (отображенная в поле холста) уже представляла собой загруженное решение, а также была решением, дополненным последующим редактированием. Во всех этих случаях конфигурация сохраняется в хранилище описаний задач.

При клике по гиперссылке *«обсчитать текущую задачу»* начинается расчет по текущей конфигурации. В текущей версии программного комплекса

предполагается, что решается только одна определенная задача моделирования. В дальнейшем (см. раздел 5) предполагается наделить пользователя возможностью задавать схему решения задачи, т.е. конфигурацию последовательности применяемых к конфигурации задачи вычислительных алгоритмов.

Аналогично для клика по гиперссылке *«обсчитать сохраненную задачу»*.

### *Закладка редактора производственных связей: поле холста*

На поле холста закладки графического редактора производственных связей отрисовывается конфигурация модели – совокупность графических элементов, изображающих экземпляры сущностей «объект», «поставка» и «потребитель», связанных между собой в ориентированный граф.

Каждый экземпляр «объекта» и связи между объектами располагает интерфейсом для редактирования его атрибутов. В текущей реализации (в обеих модификациях) набор атрибутов является фиксированным. В дальнейшем предполагается разработать механизм гибкого редактирования самим пользователем наборов атрибутов для каждого типа «объектов» (узлов графа), также для ребер графа (бинарных связей между объектами).

При клике на имени объекта левой кнопкой мыши выскакивает окошко с заголовком «Свойства объекта “имя объекта”» (рис. 5).

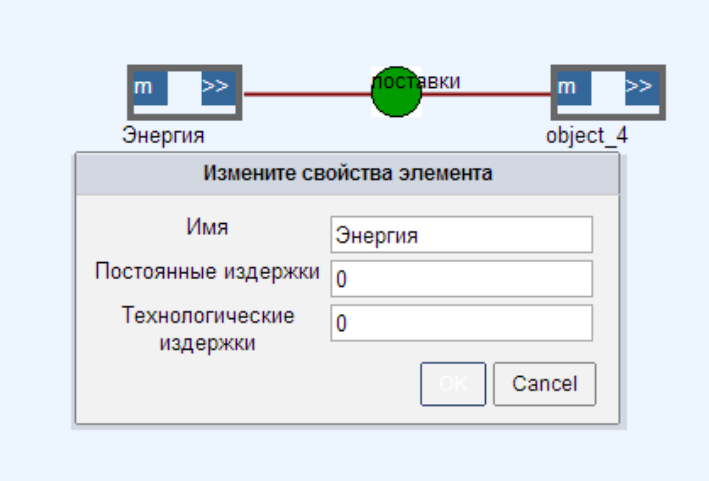


Рис. 5. Редактирование атрибутов объекта.

Аналогично для экземпляра сущности «связь», т.е. для ребра графа (рис. 6). Подробнее об атрибутах в подразделе 4.1.

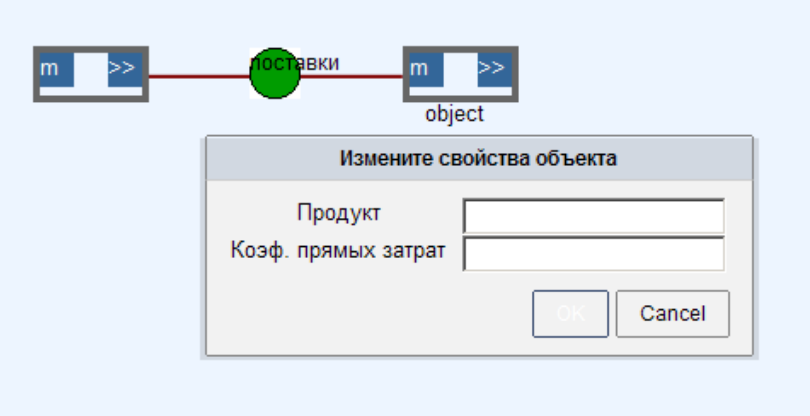


Рис. 6. Редактирование атрибутов экземпляра связи (ребра).

***Закладка графического редактора структурно-организационных связей***

Необходимость этой части интерфейса связана со спецификой конкретной экономической задачи, под которую изначально разрабатывался комплекс. Эта часть пользовательского интерфейса и предоставляемые ей графические возможности не носят универсального характера, но демонстрируют возможности технологии в целом.

По аналогии с первой закладкой интерфейс состоит из командного меню и поля холста. Меню включает в себя две кнопки: «новый субъект» и «обновить».

На поле холста отрисовывается структура организационных связей объектов, то есть их принадлежность тому или иному (экономико-юридическому) субъекту. Субъект изображается графическим элементом в виде вертикальной линии, объект, как и на предыдущей вкладке – в виде прямоугольника, принадлежность объекта субъекту изображается геометрическим расположением прямоугольников около соответствующей линии. Объекты, созданные на закладке графического редактора

производственных связей, автоматически появляются на закладке редактора структурно-организационных связей, после чего их можно перетаскивать к линиям создаваемых «субъектов» (рис. 7).

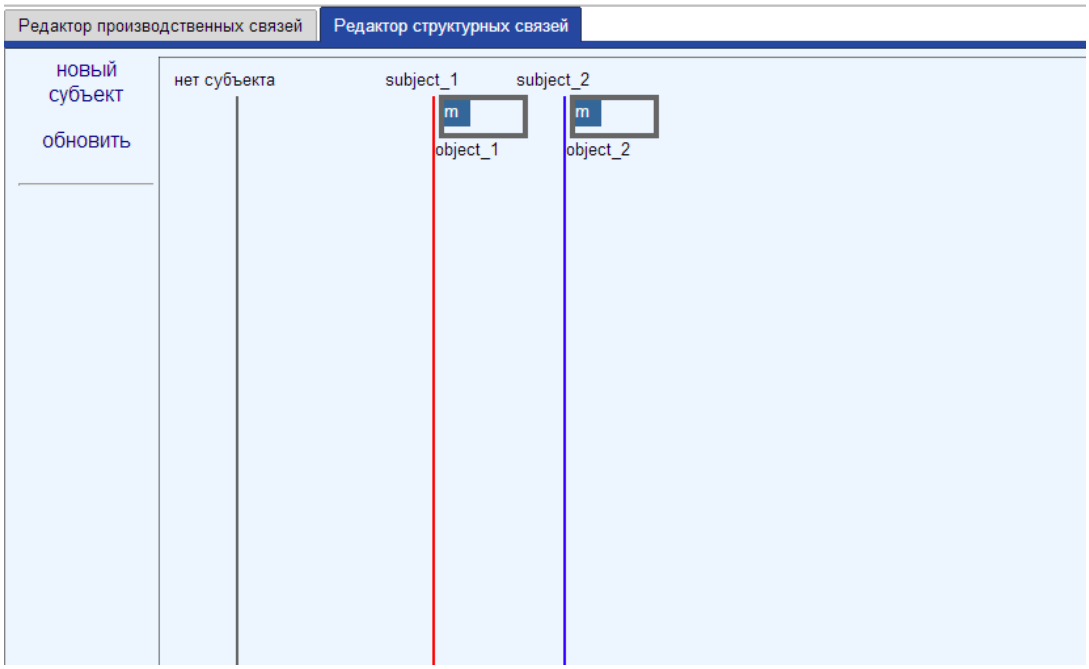


Рис. 7. Закладка редактора структурно-организационных связей.

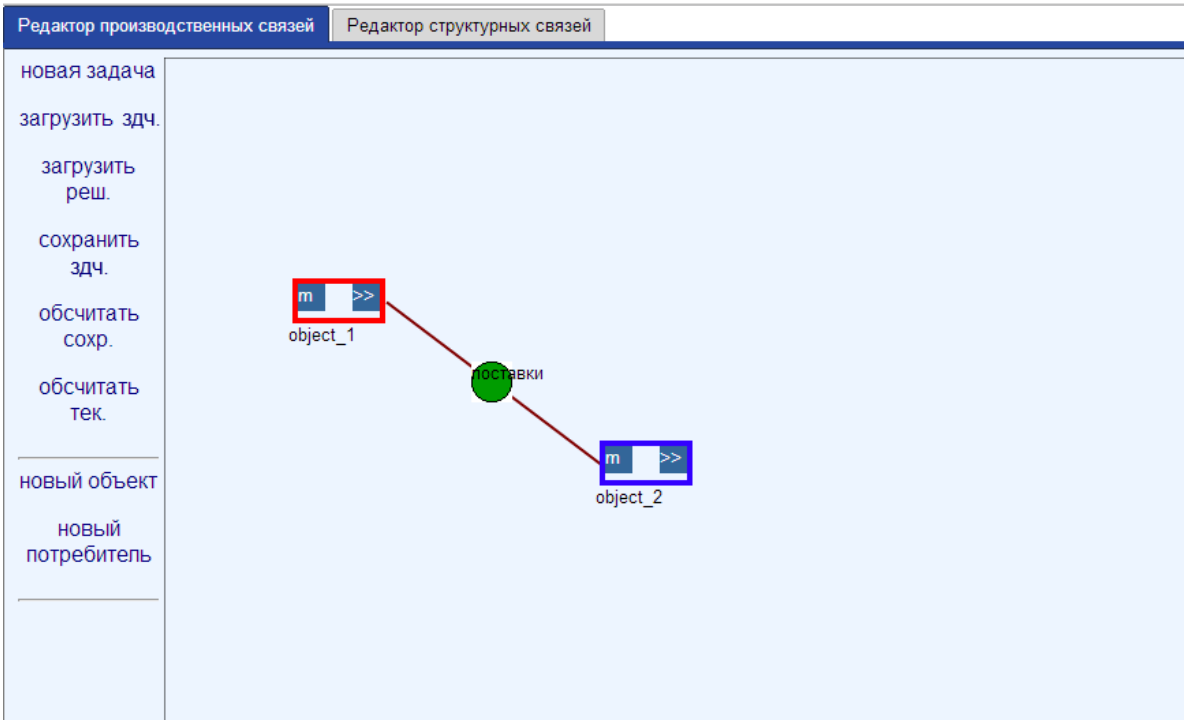


Рис. 8. Объекты подсвечиваются цветом своего субъекта.

После того как пользователь назначит субъекту «владеющий» им субъект, цвет прямоугольного графического элемента этого объекта на закладке редактора производственных связей меняется на цвет субъекта (рис. 8).

# 4. ФОРМАТ КОНФИГУРАЦИОННЫХ ФАЙЛОВ И ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ

## *4.1. Атрибуты объектов конфигурации*

Формат конфигурационных файлов и организацию вычислений поясним на примере второй модификации пилотной версии программно-вычислительного комплекса (узлы графа – этапы полета / вектора состояний КА, ребра графа – переходные режимы / полетные операции).

Конфигурация представляет собой, как указано выше, некоторое перечисление объектов, составляющих описание задачи, и бинарных связей между ними. Как объекты, так и связи обладают наборами атрибутов.

В момент создания экземпляров сущностей (т.е. объектов, соответствующих узлам и ребрам графа описания задачи) они уже наделены некоторыми атрибутами, другие атрибуты еще предстоит вычислить – в этом и состоит задача моделирования. В связи с этим атрибуты подразделяются на 3 класса:

- *автоматические*; их значения определяются самим процессом средствами интерфейсного графического редактора; на холсте названия таких атрибутов выделяются зеленым цветом;
- *редактируемые*; их значения можно явно задавать в соответствующих всплывающих меню; на окошке меню названия таких атрибутов выделяются синим цветом;
- *вычисляемые*; их значения приобретают определенные значения после выполнения вычислений; на холсте названия таких атрибутов выделяются красным цветом.

На рис. 9 видно, что вычисляемые атрибуты этапа полета `stage_2` *«высота орбиты»*, *«долгота восходящего угла»*, *«наклон орбиты к плоскости экватора»*, *«фокальный параметр орбиты»* имеют значения `null`, что соответствует тому, что вычисления еще не проведены. На рис. 10 этот же объект этапа полета имеет определенные значения перечисленных атрибутов, т.к. в данном случае загружена конфигурация решения.

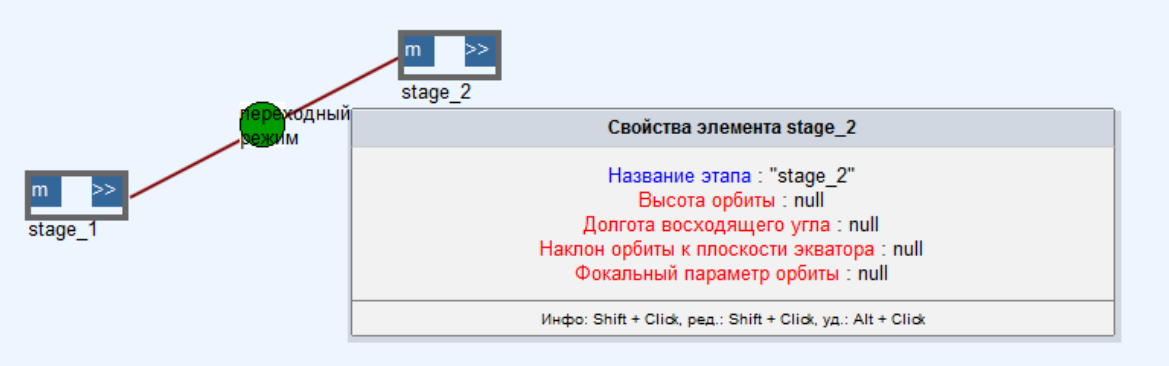


Рис. 9. Атрибуты «этапа полета» до проведения вычислений.

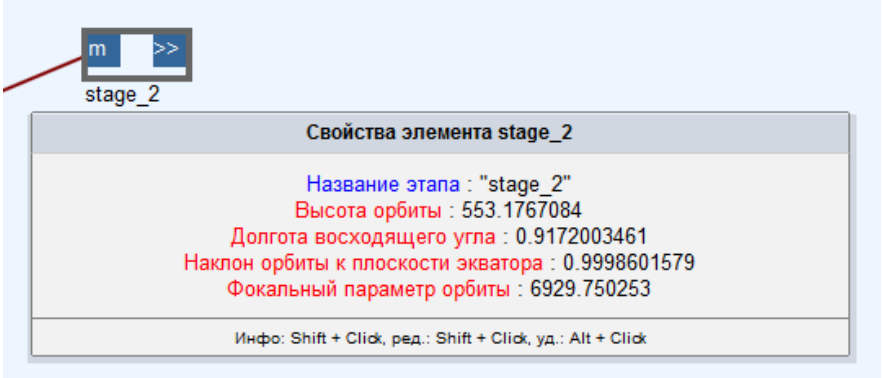


Рис. 10. Атрибуты «этапа полета» после вычислений.

На рис. 11 можно видеть названия и значения автоматических атрибутов *«начальный этап»* и *«конечный этап»* экземпляра переходного режима `operation_1`, а также редактируемых атрибутов *«t1»*, *«t2»*, *«t3»*, *«t4»*.

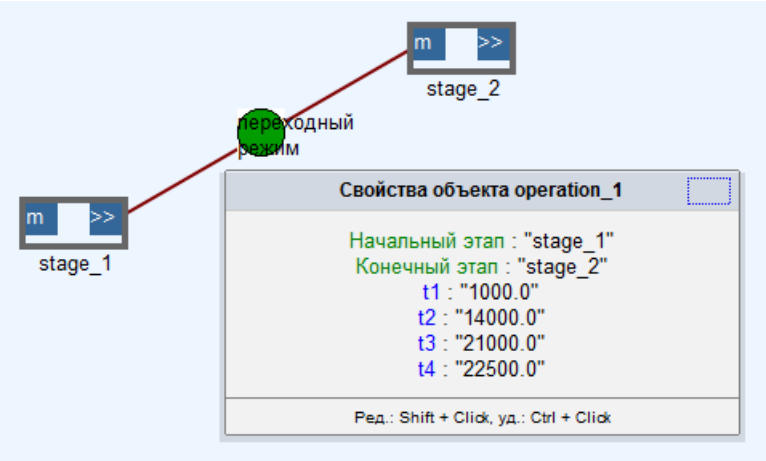


Рис. 11. Атрибуты экземпляра «переходного режима».

### *4.2. Общий вид и пример конфигурации описания задачи и решения*

Текстовая конфигурация в формате JSON представляет собой перечень произвольных объектов, понимаемых, в свою очередь, как перечень атрибутов:

```
{
    …
    object_i: {
        attribute_1 : value_1,
        attribute_2 : value_2,
        …
        attribute_N : value_N
    },
    …
}
```

Значение атрибута может быть массивом (вектором), например:

```
phoneNumbers : [ "812 123-1234", "916 123-4567"]
```

Ниже приведен пример рабочей конфигурации, созданной в процессе разработки и испытаний для второй модификации пилотной версии комплекса.

```
{"problem":{
    "all_objects": {
        "stage_1": {
            "name": "stage_1",
```

```
            "inputs": null,
            "outputs": {
                "stage_2": "operation_1",
                "stage_3": "operation_2"
            },
            "orbit_height": 200,
            "omega": 1,
            "i": 1,
            "p": 6578
        },
        "stage_2": {
            "name": "stage_2",
            "inputs": {
                "stage_1": "operation_1"
            },
            "outputs": null,
            "orbit_height": 553.1767084,
            "omega": 0.9172003461,
            "i": 0.9998601579,
            "p": 6929.750253
        },
        "stage_3": {
            "name": "stage_3",
            "inputs": {
                "stage_1": "operation_2"
            },
            "outputs": null,
            "orbit_height": 571.90933,
            "omega": 0.9174478291,
            "i": 0.9999384285,
            "p": 6943.382063
        }
    },
    "all_supplies": {
        "operation_1": {
            "origin": "stage_1",
            "target": "stage_2",
            "t1": "1000.0",
            "t2": "14000.0",
            "t3": "21000.0",
            "t4": "22500.0"
        },
        "operation_2": {
            "origin": "stage_1",
```

```
            "target": "stage_3",
            "t1": "1000.0",
            "t2": "14000.0",
            "t3": "21000.0",
            "t4": "23000.0"
        }
    }
}}
```

Конфигурация решения отличается от конфигурации задачи попросту тем, что вычисляемые атрибуты имеют определенные значения вместо `null`.

Как отмечено выше, формат JSON очень удобен для визуального ознакомления и редактирования в любом текстовом редакторе. Принципиально, опытный пользователь может работать вообще без графического интерфейса, если у него есть возможность передать файл конфигурации в вычисляющую среду. Текущая версия программного комплекса предоставляет такую возможность: m-файлы системы MATLAB поставляются вместе с веб-приложением (см. также подраздел 2.3).

### *4.3 Организация вычислений*

Проведение вычислений и цикл работы пользователя организованы следующим образом.

1. Пользователь создает начальную конфигурацию задачи моделирования в среде графического интерфейса.
2. Пользователь сохраняет конфигурацию (команда *«сохранить задачу»* в командном меню главной закладки графического интерфейса).
3. Пользователь возвращается (потенциально – многократно) к редактированию ранее созданных конфигураций по команде «загрузить решение».
4. Вновь созданную или загруженную из архива конфигурацию задачи (либо ранее сохраненную без загрузки) можно отправить на вычисление

по командам *«обсчитать текущую задачу» («обсчитать сохраненную»)*. При этом автоматической загрузки конфигурации не последует.

5. По команде *«загрузить решение»* в браузер загружается конфигурация из архива решений и сразу отрисовывается на поле холста закладки графического редактора. При этом загруженную конфигурацию можно снова рассматривать как описание задачи: ее можно редактировать, в том числе удалять объекты и добавлять новые. Эту новую конфигурацию можно снова сохранить как задачу, – и она будет сохранена в архиве задач.

## 5. НАПРАВЛЕНИЯ ДАЛЬНЕЙШЕГО РАЗВИТИЯ ПРОГРАММНО-МАТЕМАТИЧЕСКОГО КОМПЛЕКСА

Можно сформулировать требования к расширениям и дополнениям функциональных возможностей комплекса, достигнутых на данном этапе работы. С точки зрения функциональных возможностей, законченная архитектура комплекса должна обеспечивать следующие возможности:

- необходимо отдельно конфигурировать как совокупность объектов и их бинарных связей (граф задачи), так и существо вычислений, которые следует проводить на основе этой совокупности; сейчас же, собственно математическая модель, включающая совокупность численных алгоритмов, применяемых к конфигурации задачи, – жестко заложена в программный код вычислительного слоя; необходимо иметь возможность подробно описывать, какую именно задачу моделирования хочет решать пользователь: задачу Коши, задачу оптимального управления, обратную задачу того или иного рода и т.п.;
- формат конфигурации расчетных задач должен обладать большей степенью гибкости: опытный пользователь должен иметь возможность в некоторых пределах расширять набор сущностей (какого рода «объекты»

могут быть узлами графа, какого рода бинарные связи могут быть между этими «объектами»), наборы атрибутов этих сущностей и критериев моделирования; сейчас эти наборы жестко заданы для каждой модификации пилотной версии комплекса;

- данные вычислений могут представлять собой пакет файлов с данными, а не просто конфигурацию решения, где вычисляемым атрибутам присвоены определенные значения вместо `null` (см. подраздел 4.2).

С точки зрения программной реализации комплекса представляют интерес следующие направления его развития:

- использование альтернатив пакету MATLAB как основы вычислительного слоя; как отмечалось выше, текущая комплектация комплекса не требует установки всего пакета, но требует установки некоторых его компонентов, на свободное распространение которого компания MathWorks предоставляет лицензию; однако, хотелось бы быть полностью независимым от любых коммерческих пакетов, даже с указанными оговорками; в качестве варианта нами рассматривается полностью свободный пакет вычислительной математики Octave; единственным существенным требованием для кандидата является возможность интегрирования в среду Java;
- необходима разработка полноценного компонента коннектора, осуществляющего надежное и технологичное подключение к вычисляющему слою, в том числе для нескольких пользователей одновременно; в текущей версии подключение реализовано «минималистским» способом (см. также подраздел 2.3);
- не исключено нахождение программных решений, делающих взаимодействие слоя графического интерфейса пользователя и вычисляющего слоя более динамичным; на данный момент технологию работы комплекса с точки зрения пользователя можно охарактеризовать формулой *«опиши задачу – сохрани – проведи расчет – загрузи*

*решение»*; представляется возможность более полного использования технологий создания динамичных веб-интерфейсов AJAX [2] (Asynchronous JavaScript + XML), DWR – только часть имеющегося спектра инструментов.

## 6. Инновационная ценность разработки

С нашей точки зрения, предложенная организация программного комплекса математического моделирования имеет следующие инновационные признаки и характеристики.

1. Традиционное понимание систем поддержки принятия решений (см., например, [8]) подразумевало реализацию таких систем в виде stand-alone приложений. Наш комплекс, как подробно описано в предыдущих разделах, организован как многозвенная архитектура, фактически, – сочленение нескольких приложений (точнее говоря, процессов операционной системы). Это решение имеет как преимущества, так и недостатки. В числе предположительных недостатков можно указать недостаточную динамичность и интерактивность таких систем по сравнению со stand-alone приложениями – что, в принципе, поддается компенсации (см. раздел 5).

К числу несомненных достоинств веб-архитектуры следует отнести радикальное упрощение и удешевление разработки комплексов такого типа: соединение нескольких модулей через протоколы TCP/IP (HTTP), как отмечено выше, представляет собой оригинальный способ сочленения разнородных технологий и имеющихся на рынке свободных решений.

2. Использование среды Java принципиально делает системы подобного рода платформенно независимыми.

3. Использование HTTP позволяет организовать не только локальный, но и удаленный доступ к системе, что наделяет систему характеристиками веб-

сервиса, которые могут быть эффективно использованы при организации работы сети когнитивных центров самого различного уровня [5, 6, 7].

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК


1. Формат описания структурированных данных JSON, статья в Wikipedia. URL: http://ru.wikipedia.org/w/index.php?title=JSON.
2. Технология AJAX, статья в Wikipedia. URL: http://ru.wikipedia.org/wiki/AJAX.
3. Официальный веб-сайт проекта Yahoo UI. URL: http://yuilibrary.com/.
4. Официальный веб-сайт проекта Raphael. URL: http://raphaeljs.com/.
5. Малинецкий Г.Г. Маненков С.К., Митин Н.А., Шишов В.В. Когнитивный вызов и информационные технологии. Часть 1 // Экономические стратегии, №7-8, 2011. С. 68-76.
6. Малинецкий Г.Г. Маненков С.К., Митин Н.А., Шишов В.В. Когнитивный вызов и информационные технологии. Часть 2 // Экономические стратегии, №9, 2011. С. 24-31.
7. Десятов И.В., Маненков С.К., Малинецкий Г.Г., Митин Н.А., Отоцкий П.Н., Ткачев В.Н., Шишов В.В. Когнитивные центры как информационные системы для стратегического прогнозирования. // Информационные технологии и вычислительные системы. №(1), 2011. С. 65-81.
8. *Борщев А.* Применение имитационного моделирования в России – состояние на 2007 г. // 3-я Всероссийская научно-практическая конференция по имитационному моделированию ИММОД, 17-19 октября 2007 года. – СПб. 2007. С. 11-16.