УДК 004.9:504:519.6

Е.А. Евдин
Ин-т проблем математических машин и систем НАН Украины
(Украина, 03680, Киев, ул. Глушкова, 42,
тел. (044) 5261438, e-mail: yewgen@env.com.ua)

Технология интеграции математических моделей в системы поддержки принятия решений в сфере экологической безопасности на основе распределенных объектов-оберток

Разработана информационная технология интеграции вычислительных моделей на основе распределенного объекта-обертки (РОО), который распределяет задачи во время их выполнения между различными компонентами системы и обеспечивает логическую, визуальную и техническую интеграцию математической модели в систему поддержки принятия решений. Приведена классификация функциональных подходов к интеграции вычислительных моделей, влияющих на логическую структуру РОО, типы и структуры данных, разработанные для построения РОО. Описан пошаговый процесс интеграции новых моделей с использованием РОО, позволяющий минимизировать возможность появления ошибок интеграции, своевременно их обнаружить и исправить.

Розроблено інформаційну технологію інтеграції обчислювальних моделей на основі розподіленого об'єкта-обгортки (РОО), який розподіляє завдання під час їх виконання між різними компонентами системи і забезпечує логічну, візуальну і технічну інтеграцію математичної моделі в систему підтримки прийняття рішень. Наведено класифікацію функціональних підходів до інтеграції обчислювальних моделей, які впливають на логічну структуру РОО, типи та структури даних, розроблені для побудови РОО. Описано покроковий процес інтеграції нових моделей з використанням РОО, який дозволяє мінімізувати можливість появи помилок інтеграції, своєчасно їх виявити і виправити.

К л ю ч е в ы е с л о в а: интеграция моделей, система поддержки принятия решений, коммуникация.

В задачах экологической безопасности и рационального использования природных ресурсов все шире внедряются системы поддержки принятия решений (СППР), основанные на математических моделях динамики окружающей природной среды. Автоматизировав процесс запуска расчетов моделей, сбора и анализа результатов, можно создать СППР, пользователь которой, имеющий знания в предметной области и заинтересованный в получении практически значимых результатов, не обязан знать структуру, уравнения и

алгоритмы математических моделей, интегрированных в СППР. Задачей СППР является предоставление удобного пользовательского интерфейса для инициализации, управления ходом расчета модели, визуализации и последующей обработки результатов. Часть входных данных может быть настроена на определенный регион и быть недоступной пользователю.

Любая практическая задача находится на пересечении различных предметных областей. Например, такими являются задачи экологической безопасности с использованием моделей метеорологических, гидрологических, океанологических, гидрогеологических процессов, моделей переноса загрязнений в окружающей среде, в биологических экосистемах и пищевых цепях. Следовательно, современные СППР, содержащие множество вычислительных моделей, должны организовывать непрерывный (с точки зрения пользователя) поток информации и передачи управления от системы к модели и от одной модели к другой. Часто эти модели разработаны на различных языках программирования вне связи с разработкой СППР, без учета их совместимости с другими моделями [1].

Задача интеграции программных модулей, написанных в различных программных средах, возникает в самых разных приложениях [2—5]. Интеграция подразумевает несколько аспектов: семантический, методологический и технический [6]. Технический аспект интеграции может быть реализован с использованием коммерческих или открытых моделирующих фреймворков — программных систем для управления интегрированными приложениями. Некоторые фреймворки только определяют стандарты и интерфейсы, тогда как другие предоставляют решение в виде готового приложения [1].

Альтернативным подходом к интеграции моделей является внедрение в систему интерпретатора — специально разработанного проблемно-ориентированного языка программирования и сервисных средств написания и отладки программ на этом языке. При этом для интеграции внешних моделей требуется описание иерархии входных и выходных наборов данных модели и их представление в интерфейсе пользователя с помощью классов данного языка. Такой поход реализован для ряда СППР в области экологической безопасности (Rodos-Hydro, MOIRA, COSYMA) с использованием языка LIANA [7, 8].

Из моделирующих фреймворков можно упомянуть Open Modelling Interface (OpenMI) [9] как стандарт для динамического объединения моделей во время исполнения. OpenMI можно использовать во многих предметных областях, но в настоящее время он в основном применяется к гидрологическим моделям. Первая версия интерфейса позволяла объединять компоненты, разработанные различными пользователями, без их перекомпи-

ляции, но базировалась исключительно на механизме опрашивания данных (pull-based). Вторая версия стандарта [10] обладает расширенной функциональностью, что в целом облегчило возможность ее использования в СППР и в программах для калибровки, оптимизации и ассимиляции данных [5]. Например, в ней добавлены возможность активной передачи данных (push-based) и поддержка геопривязанной информации.

Часто используются и другие моделирующие фреймворки, например TIME [11] — для построения и запуска различных междисциплинарных моделей, MODCOM [12] — для моделирования роста культур в растениеводстве. В [13] предложен Common Modelling Protocol, сфокусированный на динамических и биофизических моделях, в котором динамические модели помещены в иерархии с общим интерфейсом. Фреймворк Kepler [14] используется для интеграции научных программ в открытых кодах, поддерживает дискретные события и динамическую или параллельную концепцию потока информации.

Интеграция моделей, в зависимости от фреймворка, требует полного переписывания модели или незначительных изменений, или позволяет оставить код модели нетронутым, используя для интеграции так называемый объект-обертку (ОО), т.е. программный компонент фреймворка, транслирующий запросы системы к интегрируемому приложению. Объект-обертка разрабатывается под конкретное интегрируемое (унаследованное) приложение. С одной стороны, ОО следует всем требованиям по стандарту и интерфейсу, которые предъявляются фреймворком к интегрируемым приложениям, а с другой, — управляет унаследованным приложением с использованием соответствующих технологий.

При интеграции моделей в СППР, а не в моделирующий фреймворк, функциональность обертки должна быть шире обычного транслятора. Например, ОО может обеспечить интерфейс пользователя, который в удобной для себя форме будет вводить входные данные. Кроме того, обертка модели должна обеспечить контроль за процессом расчета, выполнить прямое и обратное преобразование данных из типов данных системы в типы данных модели, а также реализовать возможность объединения моделей в вычислительные цепочки, т.е. обеспечить передачу данных от одной модели к другой.

В работах [15, 16] описана архитектура распределенной кросплатформенной СППР, основанной на математических моделях, которые интегрируются в систему в виде независимых программных компонент — плагинов. Продолжением этих исследований является разработка информационной технологии интеграции моделей в СППР на основе распределенного объекта-обертки (РОО) для создания и функционирования таких плагинов.

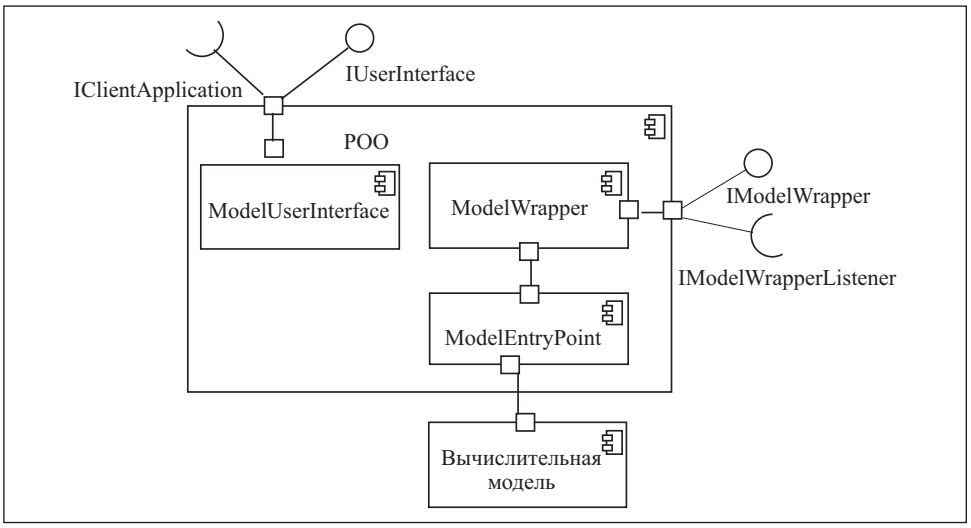

Рис. 1. Структура РОО модели

Распределенный ОО модели — это программный объект коммуникации СППР и модели, распределяемый во время выполнения между различными компонентами системы и обеспечивающий логическую, визуальную и техническую интеграцию математической модели в СППР.

Логическая интеграция обеспечивается посредством прямого и обратного преобразования типов данных системы и модели, приема запросов и команд как от системы, так и от модели и отправки ответов на них.

Визуальная интеграция происходит при предоставлении пользователю удобного интерфейса для внесения данных в модель, верификации входных данных на этапе ввода, предоставлении рекомендаций и помощи пользователю относительно введенных значений, визуализации полученных результатов.

Техническая интеграция означает обеспечение потоков данных между РОО и собственно моделью и зависит от реализации модели в виде конкретного программного объекта.

Для обеспечения каждого из перечисленных типов интеграции в обертке модели предусмотрен соответствующий независимый компонент. Для каждого из этих компонентов существует отдельная точка доступа из системы. Поэтому компоненты распределяются между частями системы, выполняемыми в различных процессах. Рассмотрим взаимодействие компонентов РОО с системой и между собой (рис. 1).

Основным и обязательным компонентом РОО модели является ее оболочка (ModelWrapper). Это единственный компонент, для которого види-

мыми являются унифицированные типы данных системы (датаитемов) и данные в формате модели. Кроме того, он реагирует на запросы системы, имплементируя интерфейс IModelWrapper, необходимый классу Task системы. Этот интерфейс содержит основные команды управления расчетом модели со стороны системы: инициализация, запуск расчета, остановка расчета, получение результатов в виде дерева датаитемов. Для обратной связи классу ModelWrapper требуется интерфейс IModelWrapperListener, предоставляемый классом Task.

Вторым компонентом РОО модели является класс ModelEntryPoint, обеспечивающий техническую часть работы по интеграции внешней модели. Этот компонент реализует вызов функций модели каким-либо способом. Способ вызова, в первую очередь, зависит от реализации модели в виде программного объекта (автономная программа, динамически подключаемая библиотека, веб-сервис, программа, загруженная на кластер, и др.). При совместной разработке, когда разработчикам обертки модели доступны ее исходные коды, модель может быть имплементирована в виде программного объекта, наиболее удобного для его дальнейшего интегрирования с системой. При этом необходимо разработать программные средства для верификации модели вне системы во избежание накопления ошибок интеграции.

Выделение ModelEntryPoint из класса ModelWrapper сделано для создания нового процесса, обеспечивающего работу точки доступа модели, которая выполняет операции с моделью в отдельном от системы процессе. Это необходимо для защиты системы от критических ошибок модели (утечка памяти, неперехваченные исключения, непредусмотренное прерывание исполнения и др.). При этом работа модели не влияет на состояние процесса основной системы.

В общем случае ввод входных данных вычислительных моделей осуществляется через графический интерфейс пользователя. В некоторых случаях разрабатывается искусственный язык создания интерфейса. Интерпретатор такого языка строит интерфейс пользователя, создавая по метаданным (название, тип, размерность и др.) подходящий элемент интерфейса для ввода данных. Такие языки дают возможность группировать объекты ввода данных и размещать их на основной панели в определенном порядке.

Если выбран путь создания языка интерфейса, то ОО модели будет содержать файл-программу на этом языке. Такой подход имеет преимущество в скорости разработки интерфейса, но ограничивает его функциональность заранее определенными типами входных параметров. При интегрировании большого количества внешних приложений может встретиться

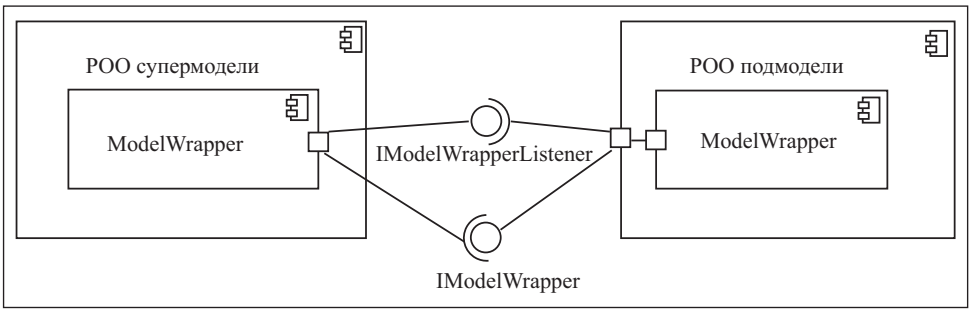

Рис. 2. Структурная схема РОО супермодели и ее взаимодействия с подмоделью

модель с непредусмотренным типом параметров или с требованиями по созданию более сложного интерфейса (например, ввод геопривязанной информации, валидация параметров или запрос к базе данных). В этом случае возникает необходимость доработки языка интерфейса и редактирования кода системы, обеспечивающего интерпретацию языка интерфейса. При этом целесообразен вынос всей логики по созданию интерфейса в ОО модели, что снимает большую часть ограничений по функциональности интерфейса пользователя модели.

Интерфейс пользователя реализуется компонентом ModelUserInterface, который имплементирует интерфейс IModelUserInterface, содержащий методы получения дерева входных данных, возврата обновленных входных данных, а также методы получения объекта — панели для встраивания в графическое окно пользователя. Входные данные могут быть введены во время предыдущего запуска или в виде параметров по умолчанию. Компоненту ModelUserInterface предоставляется интерфейс для обратной связи с системой (назовем его IClientApplication).

ModelUserInterface имеет доступ только к поддереву входных данных, которое может содержать также данные с двойной входной и выходной направленностью. При необходимости через предоставляемый интерфейс IClientApplication может запросить выходные данные текущей модели, что целесообразно в случае работы со сложными (интерактивными) моделями. Следует заметить, что при изменении входных данных все выходные данные, видимые пользователю, автоматически удаляются. Это необходимо для обеспечения целостности данных. В сложных моделях удаляются только результаты, имеющие непосредственное отношение к измененным параметрам.

Кроме обычных РОО моделей существуют также супермодели, содержащие внутри другие интегрированные модели. В основном это композитные модели, реализующие дополнительную логику над обычными моделями, которые невидимы для системы, так как весь их жизненный цикл

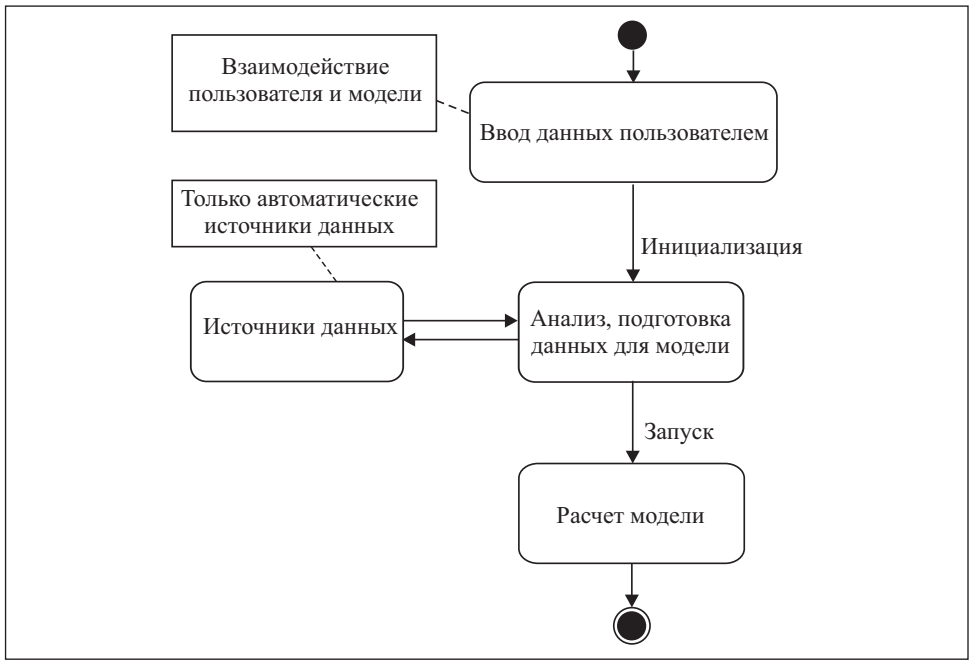

Рис. 3. Схема состояний простой модели

происходит внутри супермодели. В ModelWrapper супермодели реализуется интерфейс IModelWrapperListener, что позволяет обрабатывать результаты расчета подмоделей. Важно заметить, что подмодели являются не частью супермодели, а самостоятельными, независимо от супермодели интегрированными, компонентами, которые используются супермоделью так, как использовались бы системой при симуляции и расчете. На рис. 2 приведена схема компонентов супермодели и ее взаимодействия с подмоделью, где для упрощения из всех компонентов РОО указаны только оболочки модели (ModelWrapper).

Организация ввода (вывода) данных в математические модели. В системе вычислительная модель является «черным ящиком», преобразующим входные параметры в выходные значения. При этом входные параметры должны поступать в модель в определенном формате, иметь определенную структуру, размерность, единицы измерения. Задача преобразования данных, поступающих от пользовательского интерфейса, и баз данных, возложена на РОО модели. Этот процесс, в общем случае включающий интерполяцию и геопространственный анализ, может оказаться сложным и наукоемким. Аналогичная ситуация наблюдается с преобразованием результатов расчета моделей в удобочитаемый для поль-

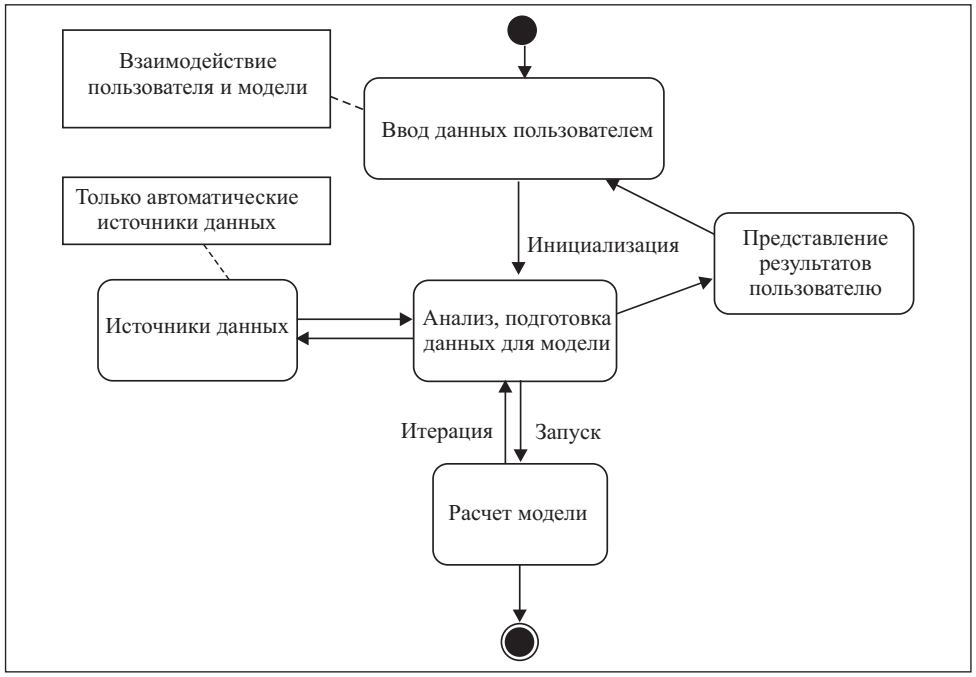

Рис. 4. Схема состояний сложной (интерактивной) модели

зователя формат или в формат, соответствующий схеме базы данных, в которой необходимо сохранить результат. Предполагается, что модель на этапе расчета не совершает никаких действий, приводящих к невозможности продолжить расчет без вмешательства пользователя. Например, модель ожидает ввода некоторой информации в консоль. Все подобные обращения должны быть проведены через обертку модели на этапе интегрирования.

В зависимости от организации ввода-вывода данных в вычислительную модель и, соответственно, функционального подхода к интегрированию модели в систему существует три типа моделей.

1. П р о с т ы е м о д е л и (рис. 3). В таких моделях сначала выполняется инициализация входных параметров, затем вызов одной или нескольких функций, обеспечивающих расчет модели. После завершения расчета результаты вводятся в РОО модели и далее транслируются в систему. При этом расчет может выполняться пошагово, с запросом результатов после расчета определенного числа шагов. Основным свойством простых моделей является отсутствие зависимости управления расчетом от полученных результатов, т.е. РОО модели сама однозначно определяет по входным значениям порядок вызова методов и передаваемые параметры.

2. И т е р а ц и о н н ы е м о д е л и. В таких моделях управление порядком выполнения расчета с определенного момента зависит от результатов расчета модели. Следовательно, РОО модели не может определить только по входным данным порядок выполнения и значения передаваемых в модель параметров. Но анализ результатов, запросы в базы данных (при необходимости) и дальнейший порядок функционирования модели выполняется РОО модели автоматически без вмешательства пользователя системы. В отличие от простой модели в схеме состояний итерационной модели предусмотрена возможность перехода из блока расчета в блок анализа и подготовки данных.

3. С л о ж н ы е и л и и н т е р а к т и в н ы е м о д е л и (рис. 4). В этих моделях ход выполнения расчета с определенного момента также зависит от результатов модели, но при этом ожидаются дополнительные входные данные от пользователя. Обертка модели получает результаты определенного этапа расчета модели, и система представляет их пользователю, который на основе полученных данных вводит дополнительные входные параметры. После их обработки РОО вызывает остальные функции модели до момента окончания работы модели или повторной необходимости принять решение в автоматическом или интерактивном режиме.

В зависимости от типа модели изменяется логическая структура разрабатываемого РОО. При этом для каждого из упомянутых типов модели можно разработать шаблон обертки, содержащий основную логику управления различными компонентами РОО.

Поскольку расчетные модели описывают процессы одного природного явления, часто в качестве некоторых входных данных используются результаты других (предшествующих) моделей. Поэтому модели объединяются в расчетные цепочки, в которых некоторые модели (нижестоящие) зависят от предыдущих (вышестоящих) моделей. Одной из задач системы поддержки принятия решений является организация цепочек моделей и обеспечение потоков данных от вышестоящей модели к нижестоящей. Поскольку модели проектируются в основном без учета применения их в цепочке, основная работа по обеспечению передачи данных выполняется в обертке нижестоящей модели.

Существует два функциональных подхода к построению вычислительной цепочки.

1. В о с х о д я щ и е в ы з о в ы. Модель, находящаяся внизу цепочки, отправляет запрос в вышестоящую модель для получения необходимого параметра. Вышестоящая модель выполняет расчет и возвращает результат в нижнюю модель. В случае необходимости вышестоящая модель вызывает интерфейс пользователя для ввода параметров или отправляет

запрос другой вышестоящей модели. Для оптимизации времени исполнения результаты расчетов могут быть кэшированы, чтобы при повторном обращении были получены уже рассчитанные результаты. Такой подход используется в стандарте интеграции OpenMI [9].

2. Н и с х о д я щ и е в ы з о в ы. Для расчета нижестоящей модели необходимо выполнить расчет всех вышестоящих, от которых модель зависит прямо или косвенно. После этого ОО модели преобразует результаты предыдущей модели по аналогии с остальными входными данными, поступающими из базы данных или интерфейса пользователя. Преимуществами такого подхода являются бóльшая автономность моделей, которые работают только с уже полученными входными данными из различных источников, а также возможность использовать полученные ранее результаты. Недостатки нисходящих вызовов—избыточность и излишний объем памяти, необходимый для хранения результатов вышестоящих моделей, которые не будут использованы в нижестоящих.

Оба способа организации цепочек соизмеримы по времени исполнения и объему используемой памяти, если при запросе от нижестоящей модели полностью рассчитываются все возможные результаты вышестоящей модели и при дальнейших обращениях возвращаются уже рассчитанные результаты. Если в случае восходящих вызовов модель и ее обертка возвращают только запрашиваемый результат, то, возможно, вышестоящую модель придется запускать несколько раз при запросах разнородных величин. Например, такая ситуация возможна, если от одной модели зависят несколько моделей, рассчитывающих несоизмеримые величины.

Выбор способа организации вычислительных цепочек так же, как и тип модели влияет на логическую структуру РОО. Но для цепочек на этапе проектирования можно выбрать один, наиболее удобный, способ организации и разрабатывать систему и шаблоны РОО только исходя из выбранного варианта.

Типы данных для работы с моделями. Рассмотрим классы системы, с помощью которых реализуется управление моделями и цепочками моделей в рамках объектно-ориентированного подхода. Модели представлены в системе с помощью класса Task (задание), который содержит входные и выходные данные, метаданные (название, состояние, тип), а также ссылку на ОО модели. Каждое задание соответствует одной модели, определяющей его тип. При этом для моделей можно создать неограниченное число заданий, отличающихся одно от другого входными и выходными значениями. Для заданий определены операции управления расчетом модели (инициализация, запуск, остановка) через ОО.

Для работы с данными модели (входными и выходными) разработан унифицированный тип данных — структура классов датаитемов (Data-

Item) [15], в которых хранятся собственно численные данные, метаданные (размерность, единицы измерения, субстанция) и отношения между ними. Для датаитемов реализованы процедуры хранения и передачи данных, созданы различные визуализаторы, позволяющие отображать данные в графическом интерфейсе пользователя. Данные из всех возможных источников унифицируются и направляются в те или иные компоненты системы в зависимости от необходимых действий.

Классы датаитемов реализуют композитный шаблон программирования, объединяющий объекты в древовидную структуру и позволяющий системе обращаться к отдельным объектам и группе объектов единообразно. Таким образом, рассматриваемая система классов представляет собой иерархическую структуру, листьями которой являются датаитемы с конкретными численными данными, а промежуточными вершинами — комплексные датаитемы. Входные и выходные данные организованы в древовидную структуру, корень которой принадлежит объекту-заданию. Каждый элемент дерева данных однозначно определяется относительно корня дерева датаитемов.

На рис. 5 приведена UML диаграмма классов датаитемов. Если существующих классов недостаточно для описания предметной области, можно добавлять новые типы датаитемов, не меняя существующую структуру. Каждый датаитем реализует интерфейс IDataItem, определяющий операции, поддерживаемые всеми датаитемами. Абстрактный класс AbstractDataItem содержит поля метаинформации, а также реализует общие функции интерфейса, которые, при необходимости, могут быть переопределены в его потомках. Комплексный датаитем (ComplexDataItem) предназначен для организации датаитемов в иерархическую структуру, содержит ссылки на дочерние элементы и не используется для хранения численных данных. Однако он может иметь другие общие для всех потомков характеристики, такие как описание, субстанция, единицы измерения. Остальные классы предназначены для хранения различной информации: скаляра, вектора, геопривязанной сетки, графика и др.

Особого внимания заслуживает датаитем-ссылка (LinkDataItem), который в качестве значения содержит ссылку на другой датаитем и переопределяет иерархические операции и операции возврата значения на соответствующие функции с датаитемом-значением.

Для объединения моделей в цепочки используется нисходящий поток вызовов: сначала полностью рассчитываются верхние модели, а потом нижние. При объединении моделей в цепочки установление соответствия выходов предыдущей модели входам следующей модели заменяется операциями с датаитемами. Нижестоящая модель в качестве некоторых вход-

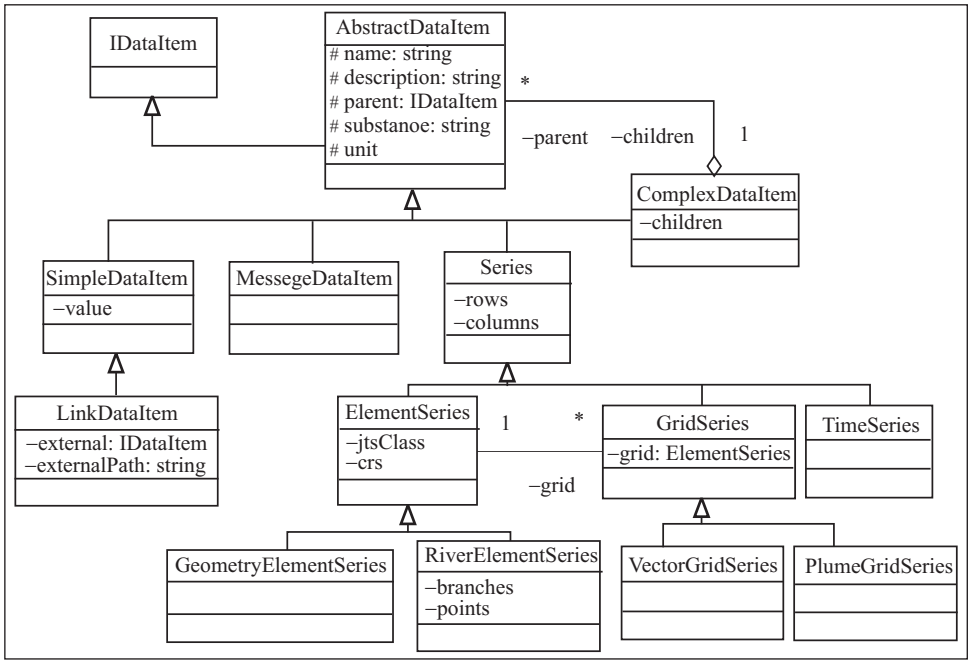

Рис. 5. UML диаграмма классов датаитемов

ных данных имеет датаитемы-ссылки, которые хранят пути к значениям в дереве датаитемов заданий, соответствующих вышестоящим моделям. После указания пользователем системы связей между заданиями эти ссылки становятся указателем на один из датаитемов предшествующей модели. Для нижней модели обращение к датаитему вышестоящей модели не отличается от манипуляций с локальным объектом унифицированного типа данных.

Задания объединяются в проекты. В общем случае можно объединять различное число заданий произвольного типа. Кроме того, предусматривается создание проекта на основе зарегистрированной цепочки моделей, что позволяет установить связи между заданиями в автоматическом порядке. При работе с созданным проектом пользователь может перенаправить поток данных на модели из любых проектов произвольным, но физически и логически верным способом.

Для сохранения достоверности данных задание рассматривается в системе как единица хранения информации. При изменении входных параметров результаты расчета удаляются автоматически. Задание может принадлежать только одному проекту, чтобы избежать одновременного редактирования заданий из нескольких проектов (рис. 6).

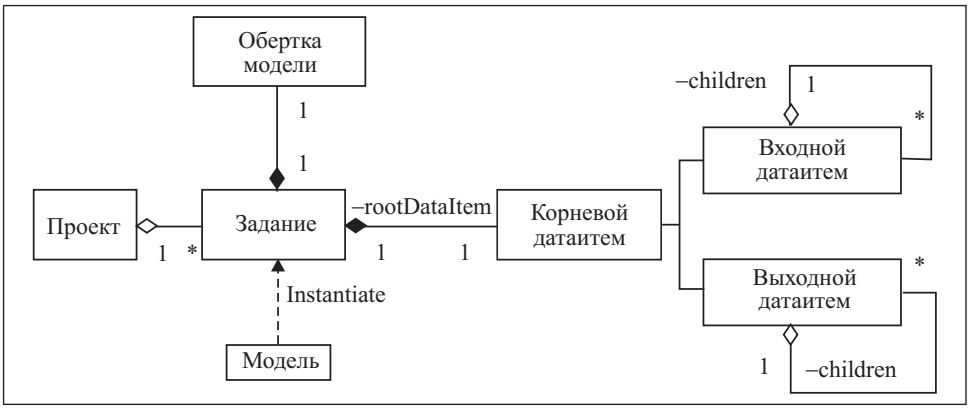

Рис. 6. Диаграмма классов системы, отвечающих за работу с моделями

Жизненный цикл (ЖЦ) интеграции математических моделей. Процедуру интеграции новой модели в готовую СППР можно рассматривать как отдельный проект по разработке программного обеспечения. Действительно, должны быть определены задачи и цели интегрирования, желаемая функциональность, создан программный продукт (РОО модели, модифицированный код расчетной модели), протестировано качество интеграции. Затем модель в виде плагина должна быть внедрена в систему. При этом процедура интеграции имеет свои особенности, на основе которых можно предложить модель ЖЦ интеграции математических моделей, которая адаптирована к условиям задачи и должна помочь разработчикам, интегрирующим модель, справиться с задачей в короткие сроки.

Модель ЖЦ интеграции является вариантом итерационной модели [17], который состоит в разделении процесса на последовательность этапов или итераций. На каждом из этапов создается работоспособный прототип, последовательно приближающийся по функциональности к требуемому продукту. Итерационный процесс позволяет локализовать и исправлять ошибки интеграции последовательно, по мере углубления интеграции системы и модели. Основную работу, которую необходимо провести на каждом этапе, можно сформулировать заранее. Описанные ниже этапы интеграции имеют некоторую избыточную, отбрасываемую на последнем этапе функциональность, связанную с поддержкой предопределенных наборов входных данных. Однако именно такая очередность шагов дает возможность последовательно увеличивать глубину интеграции, получая на каждом шаге готовый к использованию системой прототип модели. Это позволяет легко выявлять характерные для каждого этапа ошибки интеграции.

Исходной точкой для начала процесса интеграции моделей является наличие работоспособной верифицированной версии кода расчетной мо-

дели (например, в виде автономного программного продукта, читающего входные и формирующего выходные файлы), что позволит тестировать правильность интеграции модели посредством сравнения результатов интегрированной модели и исходной программы.

На первом этапе выполняется интеграция модели на низком уровне, а именно техническая организация взаимодействия система — модель, т.е. происходит создание РОО модели (реализуются два компонента — ModelWrapper и ModelEntryPoint) на основе разработанных шаблонов, определение способа технической интеграции расчетного кода и РОО модели, модификация (при необходимости) кода расчетной модели для обеспечения этой интеграции и реализация связи система — модель для подтверждения правильности выбранного решения. Рекомендуется реализовать функциональность запуска расчета модели на основе предопределенных входных данных с формированием выходных данных в изначальном виде. Это позволит протестировать правильность интеграции посредством сравнения с результатами исходной версии модели.

На втором этапе в систему интегрируются выходные данные модели. Определяются исходящие данные, их структура, типы и формы представления пользователю, реализуется преобразование исходящих данных модели в типы данных системы (в унифицированный тип данных) и формируется дерево результатов. Как и на первом этапе, модель запускается на основе предопределенных входных данных, но результат отображается средствами системы. Это является дополнительной информацией для анализа и верификации результатов расчета авторами модели, что особенно важно, если модель все еще находится в разработке.

На третьем этапе происходит интегрирование входных данных модели. Определяются все необходимые входные данные (при различных вариантах наборов входных данных — все ветки входных данных), их типы и структура. Отдельной задачей является определение источников входных данных, например интерфейс пользователя, данные из базы данных или из предшествующей модели. Формируется дерево входящих данных и реализуется преобразование этих данных в формат модели. Дополнительно реализуется логика управления расчетом модели в зависимости от входных данных и промежуточных результатов. При этом входные данные все еще могут формироваться из набора предопределенных данных, однако уже в компоненте ModelWrapper они записываются в формате унифицированного типа данных, который позже будет считываться в компоненте ModelEntryPoint. Таким образом, целью третьего этапа является получение законченной функциональности компонента ModelEntryPoint.

На четвертом этапе реализуется функциональность наполнения входных данных: создается интерфейс пользователя, базы данных и запросы к

ним, выполняется предварительная обработка входных данных, если в этом есть необходимость. Результатом четвертого этапа должен стать полнофункциональный прототип интегрированной модели, который может подвергаться расширенному тестированию. Заметим, что четвертый этап является достаточно трудоемким, его можно выполнять параллельно с предыдущими этапами. Единственной общей точкой является разработка дерева входных данных (и всех его веток), которую необходимо один раз выполнить в начале третьего или четвертого этапа.

На пятом этапе по результатам тестирования прототипа вносятся изменения в любую часть РОО модели (добавление нового результата, новых входных параметров, постобработка данных и др.). Этот этап повторяется до тех пор, пока не будет принято решение о внедрении интегрированной модели в систему.

Применение разработанной технологии. Описанная технология интеграции математических моделей в СППР успешно использована в Европейской СППР при радиационных авариях РОДОС [16, 18]. Эта система включает разработанные (более чем в 20 европейских институтах) математические модели, базы данных, геоинформационную подсистему для прогнозирования и оценки последствий возможных радиационных аварий. РОДОС использует информацию систем станционного радиологического мониторинга, оперативного метеорологического прогноза (или сценария развития гидрометеорологической ситуации) для расчета доз внутреннего и внешнего облучения персонала и населения в зоне аварии, а также планирования неотложных и долгосрочных контрмер. Система РОДОС используется в Украине, Германии, Нидерландах, Швейцарии и других странах в центрах аварийного реагирования и регулирующих органах как система прогнозирования, а также для повышения аварийной готовности при формировании планов аварийного реагирования.

Разработка системы РОДОС началась в 90-х годах в среде UNIX. В начале 2000-х создана ее мультиплатформенная версия JRODOS [16, 18, 19]. Для интеграции математических моделей в JRODOS были разработаны шаблоны РОО модели и ее составляющих для каждого из типов моделей (простых, итерационных и сложных). Большинство моделей имеют графический интерфейс в виде набора взаимосвязанных панелей с типичными графическими компонентами для ввода данных (строка ввода, выпадающий список, переключатель и др.). Кроме того, несколько моделей требуют взаимодействия пользователя с геоинформационной подсистемой РОДОС для определения контрольных точек, выделения подобластей расчетной области и установления соответствия подобластей различным классам, которые определяются непосредственно в панелях пользовательс-

кого интерфейса, например классу, определяющему набор контрмер и параметров применения, классу, определяющему характер местности.

С использованием описанной методики в систему интегрированы три модели атмосферного переноса (ATSTEP, RIMPUFF, DIPCOT), модель ранних контрмер (EmerSIM), модели расчета краткосрочных и долгосрочных доз внутреннего и внешнего облучения различными способами, блок гидрологических моделей (модель смыва, переноса по речным сетям, морская модель), модели поздних контрмер и мониторинга для населенных пунктов (ERMIN, IAMM), модель поздних контрмер в сельском хозяйстве (AgriCP) и др. По запросам Международной группы пользователей системы РОДОС разработаны супермодели, объединяющие несколько уже интегрированных моделей для специальных задач комплексной оценки ситуации в автоматическом режиме, например супермодель Emergency, состоящая из моделей атмосферного переноса, ранних контрмер и оценки краткосрочных и долгосрочных доз облучения от различных источников, в том числе и от продуктов питания.

Разработанные шаблоны применены также для создания системы прогнозирования зон затопления при паводках на р. Днепр в районе г. Киева [20]. Система передана для оперативного моделирования в отдел гидрологических прогнозов Украинского гидрометеоцентра.

Выводы

Результаты проведенного анализа технологий разработки СППР, основанных на интеграции математических моделей экологических и других процессов свидетельствуют о том, что, несмотря на активное развитие моделирующих фреймворков и стандартов интеграции программных модулей, написанных в различных программных средах, актуальным остается создание новых гибких современных информационных технологий интеграции моделей в СППР. Разработанная информационная технология интеграции вычислительных моделей на основе РОО, обеспечивая важнейшие аспекты интеграции, позволяет превратить технологический фреймворк по управлению интегрированными моделями в СППР в сфере экологической безопасности с удобным интерфейсом для инициализации, управления ходом расчета модели, визуализации и постобработки результатов. Пользователи такой системы не обязаны быть знакомыми с деталями математических моделей и вычислительных алгоритмов.

Информационная технология РОО внедрена в Европейскую СППР по реагированию на радиационные аварии РОДОС и СППР по поддержке прогнозирования затоплений при экстремальных паводках.

СПИСОК ЛИТЕРАТУРЫ

1. *Jagers B.* Linking Data, Models and Tools: an Overview//Intern. Congress on Environmental Modelling and Software. Fifth Biennial Meeting. — Intern. Environmental Modelling and Software Society. Ottawa, Canada, July 2010. — P. 1150—1157.
2. *Лаврищева Е.М.* Сборочное программирование. Теория и практика//Кибернетика и системный анализ. — 2009. — № 6. — С. 3—12.
3. *Литвинов В.В., Казимир В.В., Гавсиевич И.Б.* Распределенная система имитационного моделирования на основе архитектуры CORBA //Математические машины и системы. — 2000. — № 2, 3. — С.76—87.
4. *Дорошенко А.Ю., Котюк М.В., Ніколаєв С.С.* Програмна платформа для наукових досліджень // Проблеми програмування. — 2007. — № 4. — С 49—59.
5. *Knapen R., Janssen S., Roosenschoon O. et al.* Evaluating OpenMI as a model integration platform across disciplines // Environmental Modelling & Software. — 2013. — Vol. 39. — P. 274—282.
6. *Rizzoli A.E., Donatelli M., Athanasiadis J.N. et al.* Semantic links in integrated modelling frameworks // Mathematics and Computers in Simulation. — 2008. — Vol. 78. — P. 412—423.
7. *Гофман Д.С.* Застосування програмно-інструментальної системи LIANA для інтеграції прикладних задач, ГІС і баз даних у системи підтримки прийняття рішень, засновані на моделях // Математичні машини і системи. — 1998. — № 1. — С. 75 — 88.
8. *Hofman D., Krause P., Kralisch S., Flügel W.* LIANA Model Integration System—architecture, user interface design and application in MOIRA DSS// Advances in geosciences. — 2005. — No 4. — P. 9—16.
9. *Moore R.V., Tindall C.I.* An overview of the open modelling interface and environment (the OpenMI)//Environmental Science and Policy. — 2005. — Vol. 8, Issue 3. — P. 279—286.
10. *Donchyts G., Hummel S., Vanecek S. et al.* OpenMI 2.0 - What's new? // Intern. Congress on Environmental Modelling and Software. Fifth Biennial Meeting. — Intern. Environmental Modelling and Software Society. Ottawa, Canada, July 2010. — P. 1177—1184.
11. *Rahman J.M., Perraud S.P., Hotham H. et al.* Evolution of TIME. Eds. A. Zerger and R. Argen. — Intern.Congress on Modelling and Simulation (MODSIM 2005). — Modelling and Simulation Society of Australia and New Zealand, December, 2005. — P. 697—703.
12. *Hillyer C., Bolte J., van Evert F., Lamaker A.* The ModCom modular simulation system// European Journal of Agronomy. 2003. — Vol. 18, Issues 3—4. — P. 333—343.
13. *Moore A.D., Holzworth D.P., Herrmann N.I. et al.* The common modelling protocol: a hierarchical framework for simulation of agricultural and environmental systems// Agricultural Systems. — 2007. — Vol. 95, Issues 1—3. — P. 37—48.
14. *Altintas I., Berkley C., Jaeger E. et al.* Kepler: an Extensible System for Design and Execution of Scientific Workflows // Proc. of the 16 Intern. Conf. on Scientific and Statistical Database Management (SSDBM 2004). — IEEE Computer Society Washington, DC, USA. — 2004. — P. 423—424.
15. *Євдін Є.О.* Розробка архітектури кросплатформних розподілених систем підтримки прийняття рішень, основаних на математичних моделях // Математичні машини і системи. — 2011. — № 1. — С. 72 — 81.
16. *Євдін Є.О., Железняк М.Й., Трибушний Д.М.* Розробка кросплатформеної версії системи підтримки прийняття рішень при радіаційних аваріях JRODOS //Там же. — 2012. — Т. 1. — № 1. — С. 45—59.
17. *Литвинов В.В., Голуб С.В., Григор'єв К.М., Жигульська В.Ю.* Обєктно-орієнтовне моделювання при проектуванні вбудованих систем і систем реального часу. Навч. посібник «Системний аналіз та проектування комп'ютерних інформаційних систем». — Черкаси: Вид. від. ЧНУ ім. Б. Хмельницького, 2011. — 376 с.

18. *Ievdin Ie., Trybushnyi D., Zheleznyak M., Raskob W.* RODOS reengineering: aims and implementation details // Radioprotection — 2010. — Vol. 45, No 5. — P. 181—189.
19. *Raskob W., Trybushnyi D., Ievdin Ie., Zheleznyak M.* JRODOS: Platform for improved long term countermeasures modeling and management // Radioprotection. — 2011. — Vol. 46, No 6. — P. 731—736.
20. *Коломієць П.С., Євдін Є.О., Дзюба Н.М. та ін.* Система прогнозування та картографування зон затоплень при повенях на основі чисельного розв'язку двовимірних рівнянь мілкої води // Сб. трудов конференции «Моделирование 2012», 16—18 мая 2012. — Киев: ИПМЭ им. Г.Е. Пухова НАН Украины, 2012. — С. 224—227.

E.A. Ievdin

TECHNOLOGY OF INTEGRATING MATHEMATICAL MODELS INTO THE DECISION SUPPORT SYSTEMS IN THE SPHERE OF ENVIRONMENT SAFETY BASED ON THE DISTRIBUTIED WRAPPER OBJECT

New information technology based on the distributed wrapper object (DWO) of integration of computational models is developed. DWO is communicational object between DSS and model, which is distributed at runtime between the different components of the system and provides a logical, visual and technical integration of mathematical models into the DSS. Data types developed for model integration are shown. Models are classified based on input/output requirements, which affects logical structure of the DWO. For each model type separate software template can be developed to facilitate model integration. Two types of model chain approaches are shown: pull and push-driven, which affects logical structure of DWO manager. Step by step process of integrating new models using DWO is described, which minimizes the emergence of errors and permits finding and correcting them in time.

K e y w o r d s : model integration, decision support system, communication.

REFERENCES

1. *Jagers B.* Linking Data, Models and Tools: an Overview//Intern. Congress on Environmental Modelling and Software. Fifth Biennial Meeting. — Intern. Environmental Modelling and Software Society. Ottawa, Canada, July 2010. — P. 1150—1157.
2. *Lavrischeva K.* Assembly programming. Theory and practice // Cybernetics and Systems Analysis. — 2009. — No 6. — P. 3—12 (in Russian).
3. *Litvinov V., Kazimir V., Gavsievich I.* Distributed simulation system based on the CORBA architecture // Mathematical Machines and Systems. — 2000. — No 2, 3. — P. 76—87 (in Russian).
4. *Doroshenko A., Kotiuk M., Nikolaev S.* Software platform for scientific research // Problems in Programming. — No 4. — P. 49—59 (in Russian).
5. *Knapen R. et al.* Evaluating OpenMI as a model integration platform across disciplines // Environmental Modelling & Software. — 2013. — Vol. 39. — P. 274—282 .
6. *Rizzoli A. E. et al*. Semantic links in integrated modelling frameworks // Mathematics and Computers in Simulation. — 2008. — Vol. 78. — P. 412—423.
7. *Hofman D*. Application of the software system LIANA for integrating applications, GIS and databases in a model based decision support system // Mathematical Machines and Systems. — 1998. — No 1. — P. 75—88.
8. *Hofman D., Krause P., Kralisch S., Flügel W.* LIANA Model Integration System — architecture, user interface design and application in MOIRA DSS // Advances in Geosciences. — 2005. — No 4. — P. 9—16.

9. *Moore R.V., Tindall C.I.* An overview of the open modelling interface and environment (the OpenMI) // Environmental Science and Policy. — 2005. — Vol. 8, Issue 3. — P. 279—286.
10. *Donchyts G., Hummel S., Vanecek S. et al.* OpenMI 2.0 - What's new? // Intern. Congress on Environmental Modelling and Software. Fifth Biennial Meeting. — Intern. Environmental Modelling and Software Society. Ottawa, Canada, July 2010. — P. 1177—1184.
11. *Rahman J.M., Perraud S.P., Hotham H. et al.* Evolution of TIME. Eds. A. Zerger and R. Argen. — Intern.Congress on Modelling and Simulation (MODSIM 2005). — Modelling and Simulation Society of Australia and New Zealand, December, 2005. — P. 697—703.
12. *Hillyer C. Bolte J., van Evert F., Lamaker A. et al.* The ModCom modular simulation system // European Journal of Agronomy. 2003. — Vol. 18, Issues 3—4. — p. 333—343.
13. *Moore A.D., Holzworth D.P., Herrmann N.I. et al.* The common modelling protocol: a hierarchical framework for simulation of agricultural and environmental systems // Agricultural Systems. — 2007. — Vol. 95, Issues 1—3. — P. 37—48.
14. *Altintas I., Berkley C., Jaeger E. et al.* Kepler: an Extensible System for Design and Execution of Scientific Workflows // Proc. of the 16 Intern. Conf. on Scientific and Statistical Database Management (SSDBM 2004). — IEEE Computer Society Washington, DC, USA. — 2004. — P. 423—424.
15. *Ievdin Ie.* Development of architecture of the cross-platform distributed decision support systems based on mathematical models // Mathematical Machines and Systems. — 2011. — No 1. — P. 72—81 (in Russian).
16. *Ievdin Ie., Zheleznyak M., Trybushnyi D.* Development of the cross-platform version of the decision support system for radiation accidents JRODOS // Ibid. — 2012. — No 1. — P. 45—59 (in Russian).
17. *Litvinov V. et al.* Object-oriented modeling in the design of embedded and real-time systems // System Analysis and Design of Computer Information Systems. — Cherkasy: Bohdan Khmelnytsky National University at Cherkasy, 2011. — 376 p. (in Russian).
18. *Ievdin Ie., Trybushnyi D., Zheleznyak M., Raskob W.* RODOS reengineering: aims and implementation details // Radioprotection — 2010. — Vol. 45, No 5. — P. 181 — 189. (in Russian).
19. *Raskob W., Trybushnyi D., Ievdin Ie., Zheleznyak M.* JRODOS: Platform for improved long term countermeasures modeling and management // Radioprotection. — 2011. — Vol. 46, No 6. — P. 731 — 736. (in Russian).
20. *Kolomiets P. et al.* Forecasting and mapping flooding floods areas system based on numerical solution of two-dimensional shallow water equations // Proc. of the Int. Conf. Modelling-2012— Kiev, Ukraine, 2012. — P. 224—227 (in Russian).

Поступила 29.04.14;
после доработки 15.09.14

ЕВДИН Евгений Александрович, науч. сотр. отдела моделирования окружающей среды Ин-та проблем математических машин и систем НАН Украины. В 2008 г. окончил Киевский национальный университет им. Тараса Шевченко. Область научных исследований — системы поддержки принятия решений, геоинформационые системы.