УДК 004.428

ПРОГРАММНЫЙ ИНТЕРФЕЙС ДЛЯ ГИБРИДИЗАЦИИ ИНТЕЛЛЕКТУАЛЬНЫХ НЕЧЕТКИХ И НЕЙРО-НЕЧЕТКИХ МОДЕЛЕЙ

[1]Сеньков А.В., [2]Забурдаев А.Г., [1]Зюзиков Д.А., [1]Зюзикова Е.А.

[1]Филиал ФГБОУ ВО НИУ «МЭИ», Смоленск, e-mail: a.v.senkov@mail.ru, hardrain11@yandex.ru, lena.filonowa@yandex.ru;

[2]ООО «Инновационный центр ИТ», Смоленск, e-mail: azaburdaev@init-center.ru

В статье предложен программный интерфейс для гибридизации интеллектуальных нечетких и нейро-нечетких моделей, включающий: паттерн проектирования интеллектуальных моделей, состоящий из паттерна интеллектуальной модели, паттерна для реализации гибридных моделей и паттерна для работы с числами различной природы; паттерна структуры базы данных, поддерживающего паттерн проектирования интеллектуальных моделей. Приведен пример системы, реализованной с применением обозначенных паттернов. Реализованная система обеспечила вариативность использования различных моделей, а также предоставила возможность изменить набор и последовательность вызова моделей без участия программиста. Предлагаемый программный интерфейс, во-первых, является универсальным и обеспечивает программную реализацию различных видов интеллектуальных моделей. Во-вторых, он обеспечивает все основные виды гибридизации интеллектуальных моделей. В-третьих, интерфейс позволяет формировать интеллектуальные системы высокой степени адаптивности, в том числе системы, строящиеся пользователем без участия программиста. В-четвертых, такой интерфейс обеспечивает широкие возможности по обмену передовым опытом, под которым понимается возможность не только передачи фактов базы знаний, но и моделей или правил базы знаний.

Ключевые слова: программный интерфейс, гибридизация нечетких и нейро-нечетких моделей, передача базы правил между системами

SOFTWARE INTERFACE FOR HYBRIDIZATION OF INTELLIGENT FUZZY AND NEURO-FUZZY MODELS

[1]Senkov A.V., [2]Zaburdaev A.G., [1]Zyuzikov D.A., [1]Zyuzikova E.A.

[1]Smolensk branch of Federal Autonomous Educational Institution of Higher Education Moscow Power Engineering Institute (National Research University), Smolensk, e-mail: a.v.senkov@mail.ru, hardrain11@yandex.ru, lena.filonowa@yandex.ru;

[2]LLC «Innovation-Center IT», Smolensk, e-mail: azaburdaev@init-center.ru

The program interface for hybridization of intelligent fuzzy and neuro-fuzzy models is proposed. It includes: the intelligent models design pattern, consisting of a pattern of the intellectual model, a pattern for implementing hybrid models and a pattern for working with numbers of different nature; a pattern of a database structure that supports the intelligent model design pattern. An example of a system implemented using the designated patterns is given. The implemented system ensured the variability of the use of various models, and provided the opportunity to change the set and sequence of calling models without the participation of the programmer. The proposed software interface, firstly, is universal and provides software implementation of various types of intelligent models. Secondly, it provides all the main types of hybridization of intelligent models. Thirdly, the interface allows to build intelligent systems with a high degree of adaptability, including systems that are built without the participation of the programmer. Fourthly, this interface provides ample opportunities for the exchange of best practices, which is understood as the ability not only to transmit the facts of the knowledge base, but also models or knowledge base rules.

Keywords: software interface, hybridization of fuzzy and neuro-fuzzy models, transfer of the rules base between systems

В настоящее время разработано значительное количество интеллектуальных моделей и систем, решающих как отдельные задачи, так и комплексы задач, характеризующиеся неточностью и неопределенностью системных и внешних параметров, например [1–3]. Как правило, такие работы направлены на решение частных случаев каких-либо задач. Однако окружающий мир характеризуется высокой степенью изменчивости, в результате чего модели и программные средства, ещё вчера эффективно выполнявшие поставленные перед ними задачи, устаревают и не подходят для решения новых трактовок таких задач или решения задач в новых условиях. Таким образом, очевидно, современные интеллектуальные системы должны обеспечивать возможность гибкой адаптации, перестройки собственной структуры под изменяющиеся условия.

Одним из современных подходов к созданию гибких эффективных интеллектуальных систем с высокой степенью адаптивности является подход, основанный на гибридизации нейросетевых и нечетких моделей. В работах [4–6] рассмотрены следующие основные виды гибридизации таких моделей:

● гибридизация с функциональным замещением – в качестве доминирующей берется одна технология (модель), а от-

дельные ее компоненты замещаются компонентами других технологий (моделей);

● гибридизация с взаимодействием – технологии (модели) используются относительно независимо, и при этом, обмениваясь информацией, они выполняют различные задачи по достижению общей цели;

● полиморфная гибридизация – одна технология (модель) применяется для имитации и реализации функционирования другой.

Таким образом, современные интеллектуальные системы должны, помимо требования эффективного решения поставленной перед ними задачи, удовлетворять следующим дополнительным требованиям:

● обладать высокой степенью адаптивности под изменяющиеся внешние условия применения модели и изменяющуюся структуру и функции объекта, в рамках которого они используются;

● представлять широкие возможности для гибридизации нейросетевых и нечетких моделей.

Задача подобного масштаба может быть решена лишь путем выработки общих «правил игры» при программной реализации интеллектуальных моделей, а именно, разработки паттерна проектирования [7–9] интеллектуальных моделей, включающего программный интерфейс интеллектуальной модели и типовую структуру базы данных, обеспечивающую функционирование такой модели.

Паттерн проектирования интеллектуальных моделей

Интерфейс для реализации интеллектуальных моделей

Обобщенно интеллектуальная модель может быть представлена в виде кортежа:

$$R = <I, O>,$$

где I – вектор входов интеллектуальной модели; O – вектор выходов интеллектуальной модели.

Вектора входов и выходов интеллектуальных моделей определяют перечень общедоступных (*public*) свойств интерфейса интеллектуальной модели. Интерфейс интеллектуальной модели представлен на рис. 1.

Таким образом, все модели, построенные на базе интерфейса IIntelligentModel, будут иметь следуюшие общие методы.

● *GetInput* – метод, который получает входные параметры для модели. В качестве результата выполнения метода запрашивающая система получает совокупность параметров, которые могут использовать в качестве входов в модель. Эта совокупность составляет все параметры, которые требуется задать для расчета модели. К таким параметрам относятся входные параметры самой модели, а также все входные параметры связанных моделей, результаты выполнения которых используются в качестве входов для рассматриваемой модели.

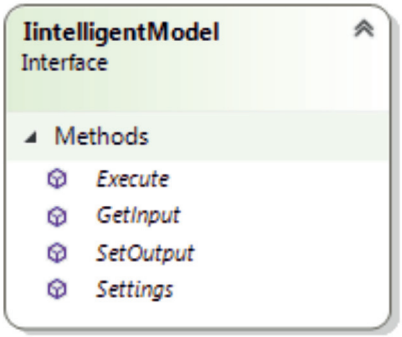

Рис. 1. Интерфейс интеллектуальной модели

● *SetOutput* – метод, который задает выходные параметры модели. В результате выполнения данного метода для модели задается совокупность параметров, которые должны быть рассчитаны и будут использованы в качестве выходов модели. Этот список выходных параметров в дальнейшем может использоваться в качестве входов для других моделей.

● *Execute* – метод, позволяющий рассчитать выходные параметры модели, используя заданные входные параметры.

● *Setting* – метод, позволяющий выполнить обучение или настройку моделей. Этот метод, как правило, должен вызывать отдельный пользовательский интерфейс настройки модели. Как правило, настройка выполняется в 3 этапа. Во-первых – выполняется построение структуры модели. Для этого на вход метода должен быть подан файл исходных данных, представленных в общепринятом формате (например, в формате графической нотации рисков, предложенной в [10], или нотации нечетких бизнес-процессов [11]. Исходные данные преобразуются в структуру модели в результате применения способа обучения или построения модели, реализованного программно. Во-вторых, факультативно, выполняется графическая адаптация модели (если сама модель предполагает такую адаптацию). Изображение модели приводится пользователем в удобочитаемый вид. В-третьих, при необходимости выполняется оперативное моделирование. Для этого пользователь подает на вход модели тестовые данные и сравнивает полученный результат с ожидаемым. При необходимости модель может быть доработана (обучена дополнительно или обучена заново).

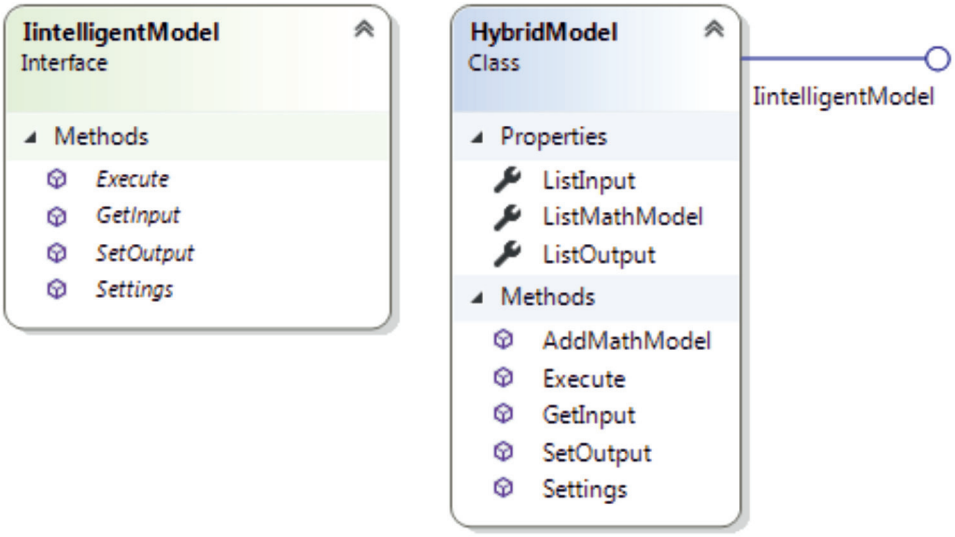

Рис. 2. Определение класса гибридной модели

Класс для реализации гибридных моделей

Гибридная модель должна также строиться на базе разработанного интерфейса для обеспечения совместимости с другими моделями.

Класс гибридных моделей можно определить следующим образом (рис. 2).

Класс *HybridModel* обладает следующими свойствами:

- *ListInput* – список входных параметров;
- *ListOutput* – список выходных параметров;
- *ListMathModel* – список математических моделей, используемых в гибридной модели в порядке их запуска.

Помимо уже рассмотренных методов интерфейса, в классе определен метод AddMathModel, который позволяет добавить в гибридную модель другую модель.

На базе представленного класса могут быть определены модели, использующие следующие типы гибридизации: с функциональным замещением, с взаимодействием и полиморфная гибридизация.

Паттерн для представления четких и нечетких чисел

Другим важным элементом определения интерфейса для гибридизации интеллектуальных моделей является паттерн представления четких и нечетких чисел. Различные модели зачастую могут оперировать одновременно как четкими, так и нечеткими числами. Для того, чтобы унифицировать обмен данными, необходимо разработать соответствующий паттерн проектирования.

Для этих целей разработан универсальный класс *NumberContrainer*, имеющий следующие свойства:

- *ValueTypeId* – определяет тип числа (четкое / нечеткое);
- *ValueString* – строка в которой хранится сериализованный объект – наследник абстрактного класса *BaseNumberValue*.
- *Number* – десериализованный объект – экземпляр наследника абстрактного класса *BaseNumberValue*.

Класс *NumberClear* описывает четкое число и содержит лишь значение четкого числа *Value*.

Класс *NumberFuzzy* описывает нечеткое число с функцией принадлежности трапецеидального или треугольного вида и содержит следующие поля, его описывающие:

- *A* – левое значение носителя;
- *B* – правое значение носителя;
- *Alpha* – величина левой границы нечеткого числа;
- *Beta* – величина правой границы нечеткого числа.

Подобным образом могут быть разработаны классы нечетких чисел, задающихся функциями принадлежности произвольного вида, как параметрических (гауссовы, Z-образные, S-образные, П-образные), а также кусочно-аналитическими функциями принадлежности и функциями принадлежности, заданными точечно (наборами точек).

С использованием такого паттерна обеспечивается независимость работы моделей как с четкими, так и с нечеткими числами. Диаграмма классов паттерна представлена на рис. 3.

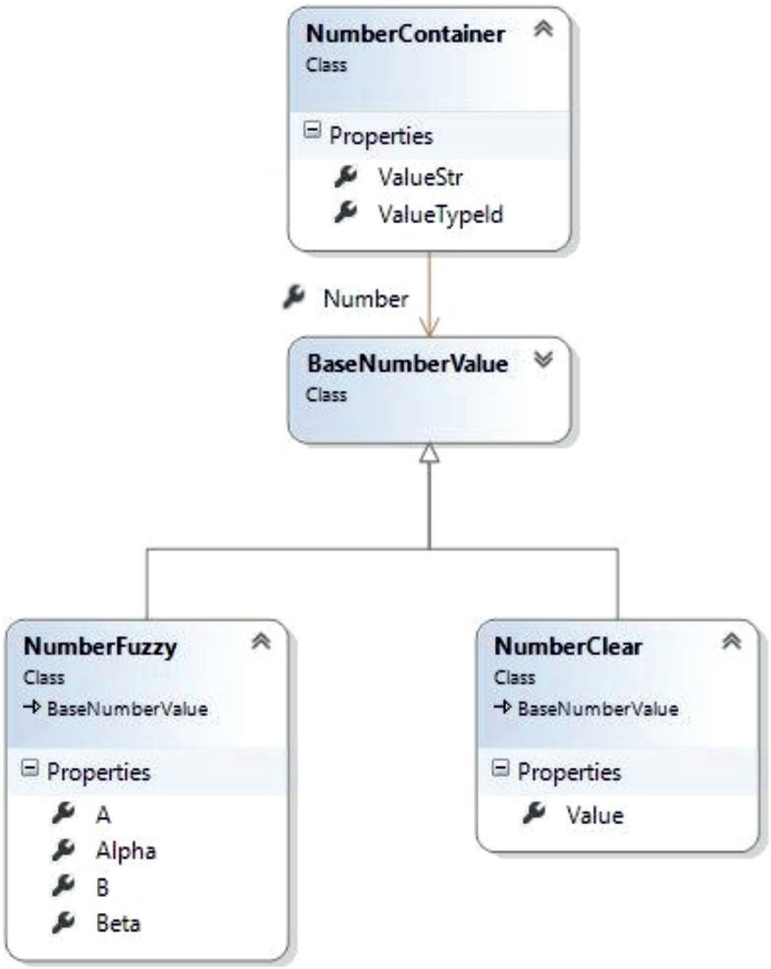

Рис. 3. Диаграмма классов паттерна для представления четких и нечетких чисел

Паттерн структуры базы данных для реализации интеллектуальных моделей

Предложенный паттерн проектирования интеллектуальных моделей должен быть поддержан паттерном структуры базы данных. Типовая структура базы данных для реализации интеллектуальных моделей приведена на рис. 4.

В представленной структуре таблица *MathModel* хранит информацию обо всех моделях в целом, а также их тип (например, нечеткое дерево отказов). Также в этой таблице хранится информация о том, какие модели используются в гибридной модели и последовательность их выполнения. В таблице *ElementMathModel* хранится информация обо всех элементах математических моделей (элементах, концептах, правилах, термах и т.д.). В таблице *ElementMathModelType* хранится расшифровка названий элементов моделей, а в таблице *MathType* – расшифровка названий самих моделей. В таблице *MathModelPatametr* хранится информация обо всех параметрах математической модели (тип параметра, единица измерения, тип значения). В таблице *ElementMathModelParametr* – информация о параметрах конкретных элементов моделей, а в таблице *ElementValueMathModelParametr* – о параметрах для конкретных значений (термов) элементов математических моделей. В таблице *MathModelLink* хранится информация обо всех связях между элементами моделей, а именно: от какого элемента идет связь, к какому элементу связь приходит, а также вес связи, если таковой имеется.

Подход к гибридизации моделей

Представленный выше программный интерфейс для гибридизации интеллектуальных нечетких и нейро-нечетких моделей обеспечивает реализацию всех перечисленных ранее видов гибридизации.

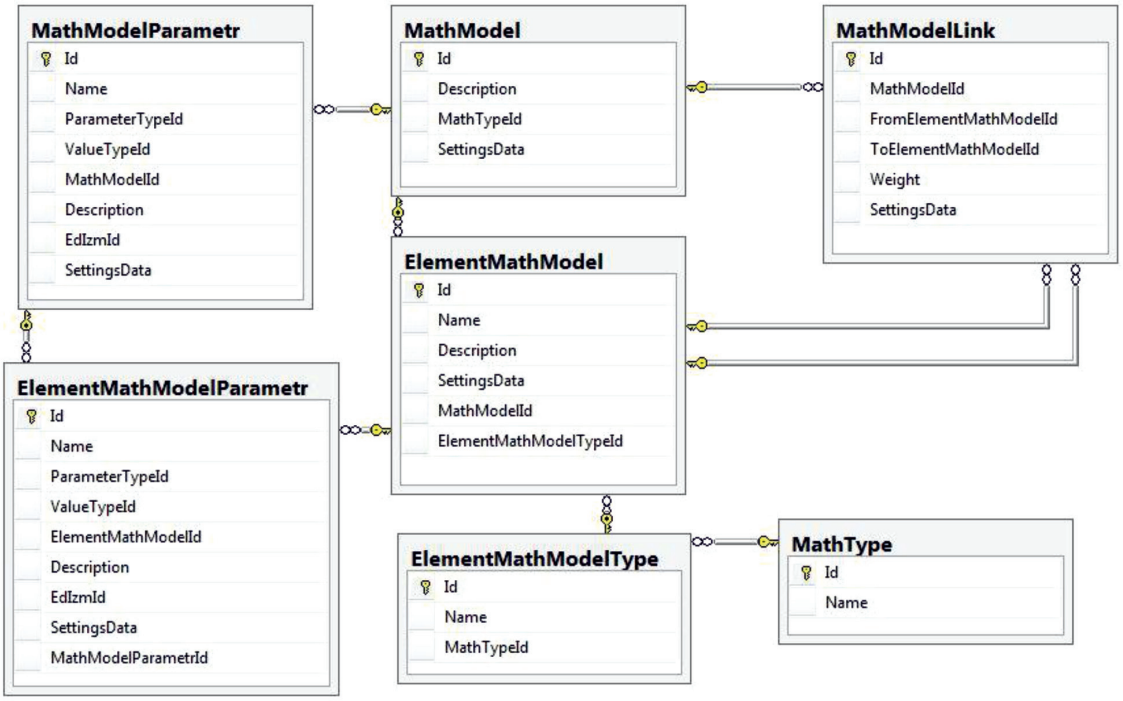

Рис. 4. Паттерн структуры базы данных для реализации интеллектуальных моделей

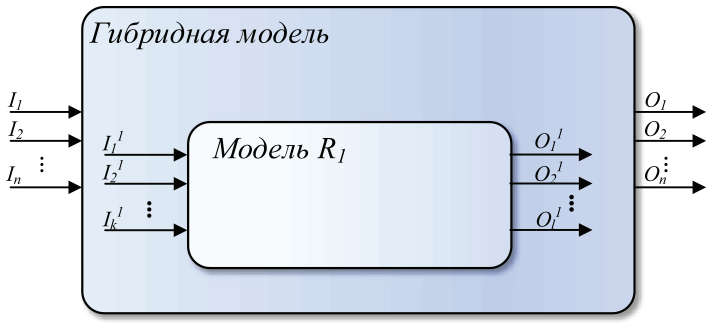

Рис. 5. Пример реализации гибридной модели с функциональным замещением

Гибридизация с функциональным замещением реализуется за счет построения новых моделей с использованием в своей структуре старых. Пример такой гибридизации приведен на рис. 5.

Гибридизация с взаимодействием реализуется за счет формирования комплексной модели, связывающей входы одних моделей с выходами других. Сама комплексная модель строится по образу и подобию базовых моделей, что позволяет выстраивать многоуровневые иерархические структуры моделей (рис. 6).

Полиморфная гибридизация, так же как и гибридизация с функциональным замещением, реализуется за счет построения новых моделей за счет изменения алгоритмов функционирования старых моделей.

Апробация программного интерфейса для гибридизации интеллектуальных нечетких и нейро-нечетких моделей

Представленный программный интерфейс был апробирован при реализации системы поддержки принятия решений для прелиминарного управления эксплуатационными рисками вычислительного кластера. В рамках указанного проекта были успешно реализованы следующие модели:

1) нечеткое дерево отказов;
2) нечеткая байесовская сеть;
3) нечеткая когнитивная карта;
4) нечеткая продукционная модель;
5) нейро-нечеткий классификатор.

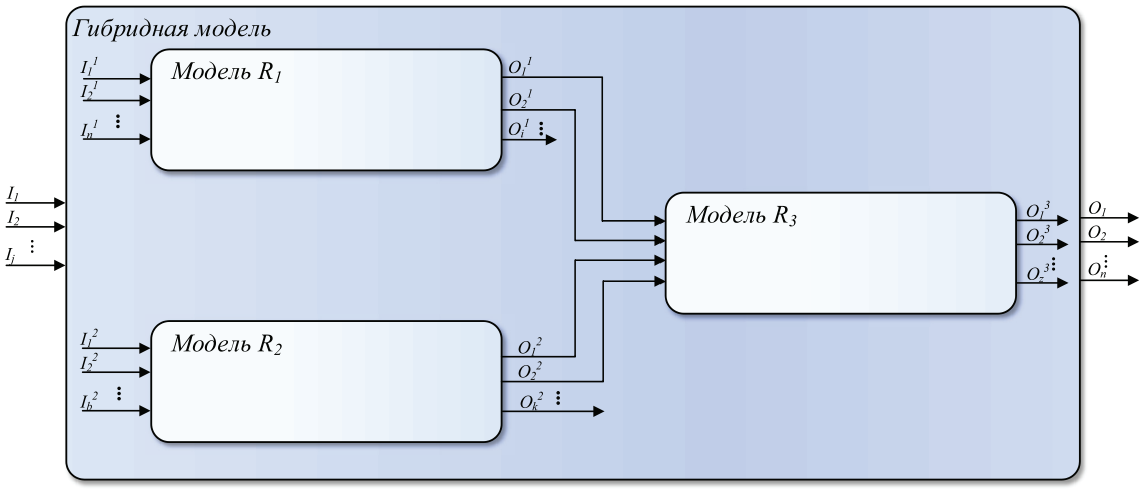

Рис. 6. Пример реализации гибридной модели с взаимодействием

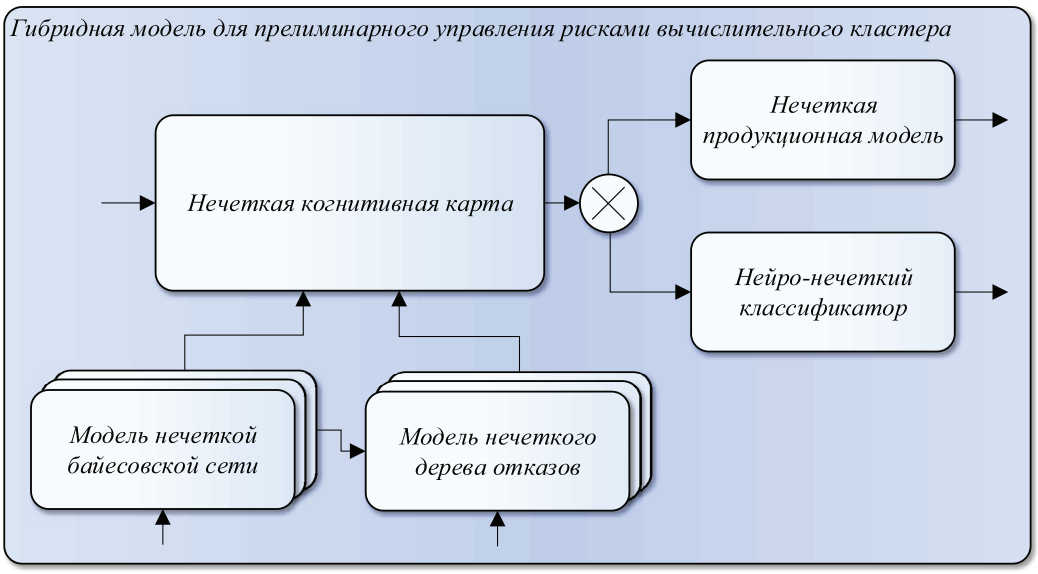

Рис. 7. Диаграмма классов паттерна для представления четких и нечетких чисел

Гибридная модель была построена по принципу гибридизации с функциональным замещением и взаимодействием. Общая схема построенной гибридной модели представлена на рис. 7.

Разработанная система обеспечивает поддержку принятия решений для прелиминарного управления рисками с учетом всех трех основных аспектов рисков [12]: процессного, структурного и системного. Кроме того, система реализует все основные этапы управления рисками: идентификацию, анализ, оценивание и выработку управляющих решений.

Разработанная система предоставляет пользователю возможность вариативного использования различных моделей. Так, на конечном этапе управления могла использоваться либо нечеткая продукционная модель, либо нейро-нечеткий классификатор. Кроме того, система позволяет без участия программиста изменить набор и последовательность вызова моделей. В качестве неизменной или базовой (ввиду ряда функциональных ограничений) выбрана модель нечеткой когнитивной карты.

Таким образом, предложенный программный интерфейс, во-первых, является универсальным и обеспечивает программную реализацию различных видов интеллектуальных моделей, имеющих крайне различающиеся способы применения (и статические модели, и модели динамики). Во-вторых, он обеспечивает все

основные виды гибридизации интеллектуальных моделей. В-третьих, интерфейс позволяет формировать интеллектуальные системы высокой степени адаптивности, в том числе системы, строящиеся пользователем без участия программиста. В-четвертых, такой интерфейс обеспечивает широкие возможности по обмену передовым опытом, под которым понимается возможность не только передачи фактов базы знаний в терминах Д.А. Поспелова между интеллектуальными системами, но и передачи моделей (передачи правил базы знаний).

Работа выполнена при частичной финансовой поддержке РФФИ, проект № 16-37-60059, а также Совета по грантам Президента РФ в рамках научного проекта МК-6184.2016.8.

Список литературы

1. Денисенков М.А. Применение адаптивных нечетких ситуационных сетей для решения аналитических задач поддержки принятия решений // Системы компьютерной математики и их приложения. – 2013. – № 14. – С. 72–75.

2. Козлов П.Ю. Методы автоматизированного анализа коротких неструктурированных текстовых документов // Программные продукты и системы. – 2017. – № 1. – С. 100–105.

3. Захаров А.С. Темпоральный вывод с использованием нечетких байесовских сетей // Известия Смоленского государственного университета. – 2014. – № 1(25). – С. 417–439.

4. Аверкин А.Н., Прокопчина С.В. Мягкие вычисления и измерения // Интеллектуальные системы. – 1997. – Т. 2, Вып. 1–4. – С. 93–114.

5. Borisov V.V. Hybridization of Intellectual Technologies for Analytical Tasks of Decision-Making Support // Journal of Computer Engineering and Informatics Jan. – 2014. – Vol. 2, Iss. 1. – P. 148–156.

6. Комарцова Л.Г. Исследование нейросетевых и гибридных методов и технологий в интеллектуальных системах поддержки принятия решений: дис.… док. тех. наук: 05.13.11; [Место защиты: Калужский филиал Московского государственного технического университета им. Н.Э. Баумана]. – Калуга, 2003. – 436 с.

7. Гради Буч. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е издание. – 2007. – 720 с.

8. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – Спб.: Питер, 2016. – 366 с.

9. Тепляков С. Паттерны проектирования на платформе .NET. – Спб.: Питер, 2016. – 320 с.

10. Сеньков А.В. Графическая нотация для представления процесса управления комплексными рисками // Современные наукоемкие технологии. – 2016. – № 12–1. – С. 72–81.

11. Сорокин Е.В., Сеньков А.В., Марголин М.С. Формальный язык описания нечетких бизнес-процессов // Научный альманах. – 2016. – № 10–3(24). – С. 278–284.

12. Сеньков А.В. Управление рисками: интеллектуальные модели, методы, средства. – Смоленск: Универсум, 2016. – 284 с.