УДК 519.95

# РЕАЛИЗАЦИЯ МОДЕЛИ *SaaS* ДЛЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ, ИСПОЛЬЗУЮЩЕЙ ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ

**© О.В. Крючин, Д.О. Квашенкин, А.А. Арзамасцев**

*Ключевые слова*: информационная система; интеллектуальное ядро; искусственные нейронные сети; модель *SaaS*.

Описаны технология модели *SaaS* и ее применение для реализации доступа к информационной системе с интеллектуальным ядром, основанным на искусственных нейронных сетях. Показаны преимущества данной технологии и основные принципы эксплуатации системы, в которой она использована. Приведено описание протоколов передачи данных, используемых при реализации модели *SaaS*, и фрагментов программного кода на серверной и клиентской сторонах.

**Введение.** Как известно, приложения, реализованные согласно технологии *SaaS* (приложение как услуга), обеспечивают повсеместный и удобный сетевой доступ по требованию к общему пулу настраиваемых вычислительных ресурсов, которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами и/или обращениями к провайдеру.

Одним из главных преимуществ модели *SaaS* является значительное уменьшение расходов на инфраструктуру информационных технологий и возможность гибко реагировать на изменения вычислительных потребностей, используя свойства вычислительной эластичности приложения, реализованного согласно модели *SaaS* [1].

*Целью* данной работы является разработка информационной системы с использованием в качестве интеллектуального ядра моделей, построенных на основе *искусственных нейронных сетей* (ИНС), и реализация модели *SaaS* для взаимодействия с пользователями.

**Информационная система.** Предлагаемая в данной работе информационная система определяет решение задач, возникающих в различных предметных областях на основе накопленной базы знаний о рассматриваемой проблеме. Формирование базы знаний происходит при совместной работе пользователей и эксперта в исследуемой предметной области. При последовательном вводе информации пользователями (входные данные задачи) происходит ее оценка экспертом, при этом определяются соответствующие выходные состояния задачи. Таким образом, происходит накопление базы знаний о рассматриваемой задаче до необходимого объема. Затем происходит построение интеллектуальной модели задачи. Моделирование осуществляется с помощью построения и обучения искусственной нейронной сети на основе накопленных данных. Данный метод моделирования ранее был рассмотрен в работе [2]. По мере поступления новой информации в базу знаний полученная интеллектуальная модель задачи совершенствуется. При накоплении некоторого критического объема базы знаний модель способна самостоятельно определять решение задачи. Таким образом, роль эксперта уменьшается по мере формирования интеллектуального ядра системы. В результате функционирования предложенной технологии происходит построение экспертных систем, позволяющих решать конкретную задачу в заданной предметной области [3].

**Способ реализации технологии.** Предлагаемая технология реализуется путем разработки целостной интерактивной системы, состоящей из взаимосвязанных компонентов (архитектура представлена на рис. 1), позволяющих осуществлять построение нейросетевых систем. Для каждого компонента системы определен круг решаемых им задач. Система включает *три основных подсистемы*: нейросетевая подсистема (интеллектуальное ядро – компоненты *ANN-Builder* и *ANN-Executor*), пользовательская подсистема (компонент *User-Mediator*) и управляющая подсистема (компонент *Manager*) [3].

**Биологическая предпосылка выбора искусственных нейронных сетей в качестве интеллектуального ядра информационной системы.** Выбор аппарата ИНС в качестве интеллектуального ядра обусловлен способностью к обучению и генерализации (обобщению) накопленных знаний. Натренированная на ограниченном множестве данных сеть способна обобщать полученную информацию и показывать хорошие прогностические способности на данных, не использовавшихся в процессе обучения [2].

Предлагаемый здесь подход является *бионическим*, т. е. в определенной степени воспроизводит процессы формирования и использования интеллектуального ядра биологических объектов.

В общих чертах процесс накопления знаний и их использование биологическими объектами выглядит следующим образом. При рождении биологического объекта имеет место некоторая априорная натренированность его интеллектуального ядра, в качестве которого выступает обученность нейронной системы. Такая предварительная натренированность ассоциируется с системой врожденных безусловных рефлексов. В процессе жизни биологический объект получает информа-

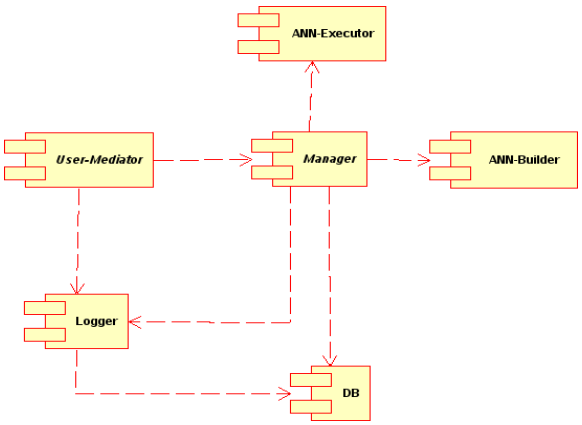


**Рис. 1.** Архитектура информационной системы

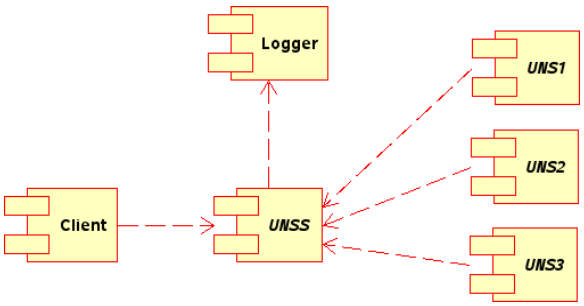


**Рис. 2.** Схема взаимодействия подсистемы с другими компонентами

цию из окружающего мира и с помощью эксперта, в качестве которого обычно выступает его мать, осуществляет ее классификацию и «привязку» к ранее накопленной информации. Начиная с некоторого уровня обученности, такой объект сам может анализировать и классифицировать поступающую информацию, создавать новую систему связей внутри интеллектуального ядра, а также использовать накопленную информацию при принятии решений, прогнозировании и т. д. [2].

**Интеллектуальное ядро.** Интеллектуальное ядро информационной системы (*ANNBuilder*) компоненты представляет собой набор компонентов построения ИНС-модели (*UNS*), размещаемых на вычислительных узлах кластерной системы (при наличии вычислительного кластера) или на вычислительных машинах и сервера нейросетевых симуляторов (*UNSS*), размещаемого на мастер-узле кластерной системы или на нейросетевом сервере. Назначение подсистемы состоит в построении искусственной нейронной сети, обученной на множестве совокупностей входных параметров объекта с соответствующими выходными состояниями, способной классифицировать вновь поступающую информацию, а также обладающей способностью доучиваться. Результатом работы данной подсистемы является интеллектуальная модель объекта, по ходу работы системы принимающая соответствующие состояния. Модель может находиться в трех состояниях – построения, классификации и доучивания [4, 5].

В процессе построения интеллектуальной модели объекта происходит выявление взаимосвязей и зависимостей между входными параметрами объекта и соответствующими выходными состояниями. Построенная модель определяет выходное состояние объекта по вновь поступающим входным данным. На основе определенного выходного состояния происходит классификация поступившей информации [3].

**Пользовательская подсистема.** Данная подсистема (*UserInteractor*), реализующая модель *SaaS*, состоит из набора *PHP*-скриптов, взаимодействующих с пользователем, и компонента, взаимодействующего с управляющей подсистемой. Назначение подсистемы – накопление, хранение и предоставление информации об объекте, а также обеспечение интерфейса для конечного пользователя.

Подсистема поддерживает *три уровня доступа* – пользователя, оператора (эксперта в предметной области) и администратора. Каждому уровню соответствует определенный набор полномочий и функциональных возможностей. Уровню администратора соответствуют полномочия управления пользователями. Уровень оператора наделен возможностями управления объектом. Пользовательский уровень дает возможность вносить информацию о параметрах объекта и получать результат по внесенным данным.

Конкретная задача, рассматриваемая в заданной предметной области, называется в разрабатываемой системе *объектом*. Объект системы определяется набором совокупностей входных параметров с соответствующими выходными состояниями. Создание объекта доступно для пользователя с полномочиями оператора. В результате создания определяются такие его характеристики, как:

– количество входных параметров;

– входные параметры, по которым впоследствии будет проводиться анализ;

– предполагаемые выходные состояния.

Накопление информации об объекте может осуществляться двумя способами.

*Первый способ* осуществляется с участием пользователя и оператора. При этом пользователь, зарегистрированный в системе, вносит параметрические данные объекта, которые сохраняются в базе данных. Затем оператор, анализируя совокупность входных данных, введенных ранее пользователем, определяет соответствующее выходное состояние объекта. Таким образом, полученные знания накапливаются в базе данных, формируя базу знаний об объекте. Информация, хранящаяся в базе знаний, образует классы, определенные в соответствии с выходными состояниями объекта.

*Второй способ* происходит без участия пользователя. Здесь оператор, уже имея набор совокупностей входных данных об объекте, может загружать его в систему. Набор может быть уже проанализированным или впоследствии оценен экспертом [3].

Как уже было сказано, для взаимодействия информационной системы с пользователями используется технология построения приложений согласно модели *SaaS*. Среди особенностей этой технологии можно выделить следующие:

– универсальный доступ по сети, в результате которого услуги доступны потребителям по сети передачи данных вне зависимости от используемого терминального устройства (т. е. пользователь может использовать персональных компьютер, ноутбук, нетбук, смартфон или любое другое устройство);

– объединение ресурсов в единый пул для динамического перераспределения мощностей между потребителями в условиях постоянного изменения спроса на мощности (при этом потребители контролируют только основные параметры услуги);

– эластичность, услуги могут быть предоставлены, расширены, сужены в любой момент времени, без дополнительных издержек на взаимодействие с поставщиком, как правило, в автоматическом режиме;

– учет потребления, потребленные ресурсы автоматически исчисляются на определенном уровне абстракции, и на основе этих данных оценивается объем предоставленных потребителям услуг.

**Эксплуатация информационной системы и ее основные отличительные особенности.** Основное преимущество модели *SaaS* состоит в отсутствии затрат, связанных с установкой, обновлением и поддержкой работоспособности оборудования и работающего на нем программного обеспечения.

Отличительными особенностями предлагаемой информационной системы являются:

– информационная система приспособлена для удаленного использования;

– одна информационная система используется несколькими клиентами;

– оплата взимается либо в виде ежемесячной абонентской платы, либо на основе объема операций;

– техническая поддержка информационной системы включена в оплату;

– модернизация и обновление информационной системы происходит оперативно и прозрачно для клиентов.

Таким образом, заказчики платят не за владение программным обеспечением как таковым, а за его аренду (т. е. за его использование через веб-интерфейс). Следовательно, в отличие от классической схемы лицензирования ПО заказчик несет сравнительно небольшие периодические затраты, и ему не требуется инвестировать значительные средства в приобретение ПО и аппаратной платформы для его развертывания, а затем поддерживать его работоспособность. Схема периодической оплаты предполагает, что если необходимость в программном обеспечении временно отсутствует, то заказчик может приостановить его использование и заморозить выплаты разработчику [1, 6].

Данный способ реализации также позволяет эффективно бороться с нелицензионным использованием информационной системы, поскольку программное обеспечение как таковое не попадает к конечным заказчикам.

Таким образом, информационная система обладает следующими особенностями:

– доступ к системе предоставляется удаленно по сетевым каналам и, как правило, через веб-интерфейс, кроме того, могут использоваться тонкие клиенты и терминальный доступ;

– система развертывается в центре обработки данных в виде единого программного ядра, с которым работают все заказчики;

– система предоставляется на условиях уплаты периодических арендных платежей;

– обслуживание и обновление компонентов системы выполняется централизованно на стороне сервера и происходит незаметно для клиентов;

– стоимость технической поддержки включается в арендную плату.

Контракт на аренду информационной системы включает в себя не только плату за использование, но и оплату всех затрат, связанных с поддержкой его работоспособности, обновлением и защитой данных. В том случае, если задачи заказчика не требуют адаптации системы, то первоначальный платеж отсутствует. Данное обстоятельство является важнейшим преимуществом предлагаемой модели над классическим лицензированием программного обеспечения, которое требует существенных начальных инвестиций на его закупку. Периодические арендные платежи можно сравнить со стоимостью технической поддержки – обычно они жестко прописываются в договоре и потому являются предсказуемыми. Тем самым обеспечивается защита инвестиций заказчика в используемый программный продукт [1].

**Механизм взаимодействия компонентов целостной системы.** Все вышеописанные компоненты являются взаимосвязанными частями единой целостной системы. Каждому этапу соответствует реализация определенных задач.

*Первый этап* – накопление информации. На этом этапе происходит формирование и накопление базы знаний объекта.

*Второй этап* – построение модели. При получении определенного объема знаний объекта данные из базы импортируются в модуль работы с ИНС. На данном этапе происходит построение интеллектуальной модели объекта.

*Третий этап* – классификация. На данном этапе построенная интеллектуальная модель способна самостоятельно классифицировать поступающую в базу данных информацию.

**Факторы, способствующие продвижению информационной системы.** Ключевым фактором, объясняющим экономическую целесообразность предлагаемой информационной системы, является «эффект масштаба». Кроме того, использование единого программного ядра позволяет планировать вычислительные мощности и уменьшает проблему пиковых нагрузок для отдельных заказчиков. Все это позволяет существенно снизить стоимость обслуживания ПО.

Другим ключевым фактором является уровень обслуживания информационной системы. Также можно предложить уровень обслуживания и поддержки информационной системы в работоспособном состоянии, недоступный для внутренних ИТ-отделов компаний [1, 6].

Кроме того, можно выделить несколько других факторов, стимулирующих использование программного обеспечения:

– отсутствие необходимости установки ПО на рабочих местах пользователей – доступ к информационной системе осуществляется через обычный браузер;

– радикальное сокращение затрат на развертывание системы в организации (расходы на аренду помещения, организацию хранилища данных, оплату труда сотрудников и т. д.);

– сокращение затрат на техническую поддержку и обновление развернутых систем (вплоть до их полного отсутствия);

– быстрота внедрения, обусловленная отсутствием затрат времени на развертывание системы;

– понятный интерфейс – большинство пользователей уже привыкли к использованию веб-сервисов;

– мультиплатформенность;

– возможность получить более высокий уровень обслуживания ПО.

**Протокол передачи данных.** Протокол передачи данных между компонентами информационной системы, названный *CrVSISP*-2 (*CreoVector Simulation Information System Protocol*), предназначен для передачи строки символов в кодировке *UTF-8*. Он является развитием протокола *CrVSISP*, предложенного компанией ООО «КреоВектор» для передачи данных между нейросетевым симулятором и сервером.

Для того чтобы отправить строку в кодировке *UTF-8*, необходимо разделить ее на пакеты для передачи, каждый из которых представляет собой последовательность байт длиною 255, где 254 байт являются частью передаваемой строки, а последний байт зарезервирован и является управляющим.

То есть если строка в кодировке *UTF-8* содержит в себе 1000 байт, то ее необходимо разделить на 4 пакета:

1) 254 байта, с управляющим байтом, имеющим значение 0*x2B* («+»), итого 255 байт;

2) 254 байта, с управляющим байтом, имеющим значение 0*x2B* («+»), итого 255 байт;

3) 254 байта, с управляющим байтом, имеющим значений 0*x2B* («+»), итого 255 байт;

4) вся оставшаяся часть строки, т. е. остаток от деления на 254 (1000 mod 254 = 238). Таким образом, этот пакет содержит 238 байт информации, и его необходимо дополнить до 254 байтом 0*x7C* («|»); т. к. это последний пакет, то в конце ставится управляющий байт 0*x2D* («–»).

Таким образом:

– пакет представляется собой последовательность байт длиною 255 элементов;

– первые 254 байта пакета представляют собой данные передаваемой строки;

– последний байт является управляющим и может принимать следующие значения: 0*x2B* – будет еще один пакет (этот пакет не последний); 0*x2D* – это последний пакет сообщения (больше данные передаваться не будут);

– если последняя передаваемая часть сообщения не меньше 254 байт, то необходимо заполнить недостающие байты значением 0*x7C*;

– в последнем передаваемом пакете управляющий символ (последний) имеет значение 0*x2D*.

Алгоритм приема сообщения:

1) получение сообщения;

2) считывание всех принятых значений за исключением элементов со значением 0*x7C* (если до этого уже было сообщение, то полученные значения добавляются к имеющимся);

3) если последний 255-й байт равен 0*x2B*, то происходит возврат к пункту 1;

4) если последний 255-й байт равен 0*x2D*, то происходит сохранение считанной строки – это и есть полученное сообщение.

Если рассмотреть на примере отправленного выше сообщения, то:

1) получение 1-го пакета:

2) аналогично принимаются следующие 2 пакета;

3) принимается последний пакет,

4) в результате получается полная строка.

Таким образом, получен массив, содержащий 1000 байт, и осталось лишь правильно его интерпретировать (т. е. использовать как строку в правильной кодировке).

Ниже приводится фрагмент исходного кода интерфейсной части, работающей с сокетами.

```
<?php
class Connection{
    var $socketIdentificator = -1;
    var $connectingFlag = false;
    var $serverFlag;
    var $address;
    var $port;
    var $socketBufferSize = 255;
    var $buf;
    var $error = 0;

    function Connection($address, $port){
       $this->address = $address;
       $this->port = $port;
       $this->connect();
    }

    function connect(){
        /*
        if ($this->serverFlag)//проверяем является
ли подключение клиентским
        {*/
//            socket_close($this->socketIdentificator);
            $this->connectingFlag = false;
            $this->socketIdentificator = sock-
et_create(AF_INET, SOCK_STREAM, SOL_TCP);//создаем
новый сокет
            if($this->socketIdentificator < 0)
            {
                $this->error = "Error creat socket";
                throw new Exception("Error create
socket", 1);
            }
            if($this->socketIdentificator === false){
                $errorcode = socket_last_error();
                $errormsg = sock-
et_strerror($errorcode);
                throw new Exception("Couldn't create
socket: [$errorcode] $errormsg", 1);
            }
            //создаем новое подключение
 //           print "11".$this->connectingFlag;
            if (!socket_connect($this-
>socketIdentificator, $this->address, $this->port))
            {
                $this->error = "cannot connect";
                return false;
            }
            else
            {
                $this->connectingFlag = true;
                return true;
            }
        }
        else
        {
            String bufS = "подключение к клиенту со
стороны сервера невозможно";
            debug(bufS);
        }
    }


/**
* Строка в кодировке utf-8 преобразуется в массив
байт и подается на вход этой функции
* Отправка сообщения на вход подается массив байт
* функция strlen показывает не количество символов,
а количество байт
*/
    function sendMessage($message){
         $str = $message;
        $buf = "";
        for($i = 0; $i<=strlen($str); $i+= $this-
>socketBufferSize -1){
           for($j = 0; $j < $this->socketBufferSize -1;
$j++){
                if($i+$j >= strlen($str)){
```

```
                    $buf .= '|';
                } else{
                    $buf .= $str{$i+$j};//{} - воз-
вращение байта на определенной позиции строки
                }
            }
           if(($i+$this->socketBufferSize - 1) >=
strlen($str)){
               $buf{$this->socketBufferSize -1} = '-';
            }else{
               $buf{$this->socketBufferSize -1} = '+';
            }
            if($buf[0] != "|"){
                socket_send($this-
>socketIdentificator, $buf, $this->socketBufferSize, 0);
            }
            $buf = "";
        }
    }
/***
* Чтение сообщения
* следующее обращение означает чтение байта на опре-
деленной позиции - $str{2} в данном случае 3-го бай-
та начиная с нуля
*/
    function recvMessage(){
        $stringBuf = "";
        $exit = false;
        while(!$exit){
            $buf = "";
            $bytesRead = socket_recv($this-
>socketIdentificator, $buf, $this->socketBufferSize,
MSG_WAITALL);
            if(false) //($bytesRead <= 0)
            {
                //если не удалось прочитать часть
сообщения
                $bufS = "error read message";
                $exit = true;
                socket_close($this-
>socketIdentificator);
                $this->connectingFlag = false;
            }
            else{
               for ($i = 0; $i<strlen($buf)-1; $i++)
                {
                    if ($buf{$i} == '|')
                    {
                        break;
                    }
                    if(!(ord($buf{$i}) == 0)){
                        $stringBuf.=$buf{$i};
                    }
                }
                //если последний символ равен '-',
то сообщение последнее выходим из цикла while
                //если последний символ равен '+',
то сообщение не последнее продолжаем считывать
                if ($buf[strlen($buf)-1] == '+')
                {
                    $exit = false;
                }
                else
                {
                    $exit = true;
                }
            }
        }
        return $stringBuff;
    }

    function close(){
    }
}
```

**Реализация протокола на серверной стороне.** Приведенные ниже фрагменты программного кода, написанные на языке C++, используются для передачи данных между компонентами системы, а также для взаимодействия с интерфейсной частью.

```
/**
*   Сокетное подключение
*/
class Connection
{
public:
    /**
    *   Конструктор
    */
    Connection();
    /**
    *   Деструктор
    *   При наличии подключения оно завершается
    */
    ~Connection();
    /**
    *   Установка повторного подключения
    *   В случае уже имеющегося соединения происхо-
дит переподключение
    */
    void connect();
    /**
    *   Отключение от сервера
    */
    void disconnect();
    /**
    *   Проверка наличия сооединения
    *   @return - флаг наличия подключения
    */
    bool isConnect() const;
    /**
    *   Отправка сообщения
    *   @param message - отправляемое сообщение
    */
    void  sendMessage(const  Message  message,  bool
isComplex = true);
    /**
    *   Возвращение сообщения
    *   @return Message - возвращаемое сообщение
    */
    Message  recvMessage(String  lastSimbol  =  EMP-
TY_STRING);
    /**
    *   Проверка наличия нового сообщения
    *   @return bool - флаг нового сообщения
    */
    bool hasMessage() const;
    /**
    *   Возвращение идентификатора
    *   @return int - идентификатор
    */
    int identificator() const;
};

/**
*   Клиентское подключение к серверу
*/
class ClientConnection : public Connection
{
public:
    /**
    *   Конструктор
    *   Устанавливает подключение к серверу
    *   @param idAdress - ip-адрес сервера
    *   @param port - порт, прослушиваемый сервером
    */
    ClientConnection
        (
            String ipAdress = LOCAL_HOST_TEXT,
            const Port port = DEFAULT_PORT
        );
    /**
    *   Деструктор
    *   При наличии подключения оно завершается
    */
    ~ClientConnection();
    /**
    *   Установка повторного подключения к серверу
    *   В случае уже имеющегося соединения происхо-
дит переподключение
    *   По умолчанию параметры сервера текущие
    *   @param idAdress - ip-адрес сервера
    *   @param port - порт, прослушиваемый сервером
    */
    void connect
        (
           const String ipAdress = LOCAL_HOST_TEXT,
          const Port port = DEFAULT_PORT
       );
};

/**
*  Гнездо подключений
*/
class ConnectionsNest
{
public:
    /**
```

```
    *   Конструктор
    *   @param port - прослушиваемый порт
    */
    ConnectionsNest(Port port = DEFAULT_PORT);
    /**
    * Конструктор
    * @param port - прослушиваемый порт
    * @param error - ошибка создания прослушиваемо-
го порта
    */
    ConnectionsNest(Port port, bool &error);
    /**
    *   Деструктор
    *   Завершение всех подключений
    */
    ~ConnectionsNest();
    /**
    *   Начало прослушивания
    *   @return bool - флаг определяющий поставлен
ли сокет на прослушку
    *   true - постановка на прослушку удалась
    *   false - постановка на прослушку не удалась
    */
    bool listen();
    /**
    *   Возвращение нового подключения
    *   @return Connection * подключение
    */
    Connection * connection();
};
```

**Заключение.** Таким образом, реализована информационная система с интеллектуальным ядром, базирующимся на технологии искусственных нейронных сетей. Для передачи данных между компонентами системы использован протокол *CrVSISP*-2, а для взаимодействия с пользователями – модель *SaaS* вычислений.

## ЛИТЕРАТУРА


1. *Martin R.J., Hoover N.* Guide To Cloud Computing // Information Week. 2008. URL: http://www.informationweek.com/news/services/hosted_apps/208700713. Загл. с экрана.
2. *Арзамасцев А.А., Зенкова Н.А.* Использование аппарата искусственных нейронных сетей для идентификации свойств личности в учебном процессе // Открытое образование. 2004. № 4. С. 61-64.
3. *Арзамасцев А.А., Зенкова Н.А., Крючин О.В., Квашенкин Д.О., Неудахин А.В.* Автоматизированная технология и программно-технологический комплекс для построения экспертных систем с интеллектуальным ядром, основанным на нейросетевых моделях, поддержкой распределенного ввода данных и параллельных вычислений // Вестник Тамбовского университета. Серия Естественные и технические науки. Тамбов, 2012. Т. 17. Вып. 3. С. 948-978.
4. *Крючин О.В., Арзамасцев А.А., Королев А.Н., Горбачев С.И., Семенов Н.О.* Универсальный симулятор, базирующийся на технологии искусственных нейронных сетей, способный работать на параллельных машинах // Вестник Тамбовского университета. Серия Естественные и технические науки. Тамбов, 2008. Т. 13. Вып. 5. С. 372-375.
5. *Крючин О.В.* Программный комплекс для моделирования объектов социально-экономического назначения с использованием искусственных нейронных сетей на кластерных вычислительных системах // Психолого-педагогический журнал Гаудеамус. Актуальные проблемы информатики и информационных технологий: материалы 14 междунар. науч.-практ. конф. Тамбов, 2010. № 2 (16). С. 534.
6. *Терехов И.* Не рано ли Cloud Computing в массы? // Компьютерра. 2009. 19 окт.


Поступила в редакцию 8 апреля 2013 г.

Kryuchin O.V., Kvashenkin D.O., Arzamastsev A.A. REALIZATION OF MODEL *SaaS* FOR INFORMATION SYSTEM USING ARTIFICIAL NEURON NETWORKS

The SaaS model technology and its usage for the realization of the access to the information system with the intellectual core based on artificial neural networks are described. The technology benefits and main systems ideas use are presented. The passing data protocols which are used for the SaaS model implementation and the source fragments of the server and client parts are given.

*Key words*: information system; intellectual core; artificial neural networks; SaaS model.