ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ПРИКЛАДНАЯ МАТЕМАТИКА

УДК 004.94

Е.В. Никульчев, С.В. Паяин, Е.В. Плужник

ДИНАМИЧЕСКОЕ УПРАВЛЕНИЕ ТРАФИКОМ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ В ОБЛАЧНОЙ ИНФРАСТРУКТУРЕ

Статья посвящена разработке алгоритма динамического управления трафиком сети, ориентированной на облачную инфраструктуру. Предложенный алгоритм реализован в виде программного модуля конфигурации маршрутизатора.

***Ключевые слова**: компьютерные сети, управление трафиком, облачные вычисления, динамические модели.*

Введение. Облачные вычисления (англ. cloud computing) основаны на использовании вычислительных ресурсов как интернет-сервиса. В облачной технологии процессорные мощности, оперативная память, дисковое пространство, специализированные контроллеры и программное обеспечение являются виртуальными и могут быть расширены или уменьшены. При этом наличие нескольких источников таких ресурсов позволяет повышать гибкость системы, а также снижает риск неработоспособности [1–3].

В настоящее время получают распространение «гибридные облака», которые представляют собой такое внедрение облачных вычислений, при котором часть системы размещается в публичном «облаке», на базе дата-центров облачного провайдера, а часть – в приватном «облаке», на собственных серверных мощностях. Гибридное облако не является самостоятельной технологией, а является интеграцией виртуализированных публичных и приватных облачных систем.

Например, такая интеграция эффективна при вынесении системы резервного копирования в публичное «облако». Другой вариант использования гибридного «облака» предполагает установку приложений на внутренних серверах компании с арендой дополнительных мощностей в «облаке» стороннего поставщика на случай повышения нагрузки.

Гибридные «облака» позволяют избежать проблем, связанных с потерей контроля над ключевыми данными: эти данные останутся во внутренней сети. Если данные и будут передаваться на обработку вовне, то только в таком виде, который не создает угроз для утечки конфиденциальной информации. Также гибридная модель позволит интегрировать публичные облачные сервисы от разных поставщиков.

Пример эффективного использования облачных сервисов – проведение он-лайн конференций с большим количеством подключений [4]. Главное преимущество использования облачных технологий в том, что эти технологии позволяют очень хорошо производить масштабирование, при котором не требуется резервировать оборудование (как серверное, так и обеспечивающее отказоустойчивость) на случай большого количества участников он-лайн мероприятий.

Таким образом, для современных приложений узким местом являются сети передачи данных, и одной из важных задач обеспечения функционирования систем – программное конфигурирование загруженных сетей, направленных на использования приложений в гибридной облачной инфраструктуре.

Постановка задачи. При перегрузке канала связи пакеты помещаются в очереди, в случае переполнения очереди пакеты отбрасываются, что приводит к замедлению скорости передачи пакетов протоколом TCP/IP и потере пакетов.

Чтобы добиться гарантии качества обслуживания, применяют QoS-архитектуру (Quality of

service), которая включает в себя поддержку качества на всех уровнях стека протоколов TCP/IP и во всех сетевых элементах. Но и при этом обеспечение гарантированного качества обслуживания все равно остается самым слабым местом процесса передачи информации от источника к приемнику, так как QoS-архитектура представляет собой систему разделения трафика на статические, заранее определяемые классы с фиксированными приоритетами, процентным соотношением ширины канала для каждого типа трафика.

Задача состоит в разработке программного модуля, динамического, устанавливающего приоритеты типам трафика в QoS-архитектуре.

Экспериментальная часть.Для имитации методов управления был создан имитационный экспериментальный стенд (рисунок 1). Основой для экспериментов являются данные с вузовских корпоративных сетей. На имитационном экспериментальном стенде корпоративная сеть заменена хостом-источником.

В качестве центрально маршрутизатора использован Cisco 2610, подключенный к внутренней сети через порт FastEthernet, а к внешней сети – через порт с последней расширенной версией операционной системы IOS advipservicesk версии 12.4, поддерживающей все методы QoS.

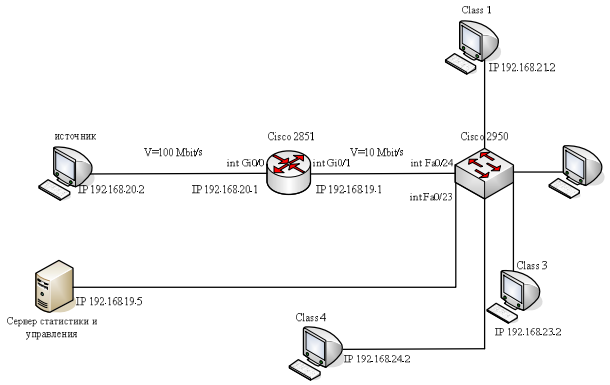


Рисунок 1 – Экспериментальный стенд

Используемая в IOS cisco 2851 технология Network-Based Application Recognition (NBAR) позволяет распознавать различные протоколы и приложения. На внешнем интерфейсе маршрутизатора запущен механизм nbar protocol-discovery, ведущий сбор данных входящего и исходящего трафиков по всем протоколам. Пример результата команды:

```
show ip nbar protocol-discovery:
http 168918343146 83872621920
secure-http 2201481628 449933737
smtp 1016344550 857620746
pop3 692671185 654180418
dns 442955841 519403663
rtp 405990792 382476819
```

В качестве маршрутизатора использован Cisco 2851, подключенный к внутренней сети и внешней сети через порты GigabitEthernet. На данном маршрутизаторе созданы сабинтерфейсы с IP адресами из различных подсетей для каждого класса трафика. Для реализации возможности работы с различными подсетями используется коммутатор Cisco Catalyst 2950, на котором создано необходимое количество Vlan-ов (Virtual Local Area Network).

Для моделирования трафика использовались динамические модели, построенные по экспериментальным данным [5].

Задача динамического управления загрузкой канала связи сформулирована следующим образом. Необходимо для каждого класса трафика ($s=\overline{1,n}$) построить закон управления $u^s(t)$ на интервале $[t_1;t_2]$ (горизонте прогноза), где модель динамики трафика имеет вид динамической системы:

$$\mathbf{x}^s(k+1)=A^s\mathbf{x}^s(k)+B^s u^s(k)+\Phi(\mathbf{x}^s(k)),$$

$$y^s=C^s{}_0(\mathbf{x}^s(k)),$$

при заданных критериях:

$$\sum_{k=t_1}^{t_2}\left(\hat{y}^s(k)-y^s(k)\right)^2\to\min,$$

$$\xi\ge y^s(k)>\eta,$$

где $\mathbf{x}^s\in\mathbf{R}^{n_s}$ – n_s-мерный вектор состояний системы s; $u^s\in U^s\subset\mathbf{R}^1$ – управление; k – дискретное время; A^s, B^s – матрицы; $\Phi(\mathbf{x}^s(k))$ – нелинейная функция, идентифицированная по методу [5]; $\hat{y}^s(k)$ – необходимая пропускная способность; $y^s(k)$ – пропускная способность канала s в условиях ограничений; ξ,η – заданные ограничения на ширину канала.

Алгоритм динамического управления. Разработан алгоритм управления, обеспечивающий повышение надежности передачи и обработки информации.

Укрупненно алгоритм состоит из следующих этапов.

1. Мониторинг трафика по типам.

2. Если загрузка канала по любому типу трафика приближается к 70 %, включается модуль динамического управления.

3. Если прогнозное значение (на горизонт прогноза) трафика старшего приоритета увеличивается, то наблюдается увеличение ширины канала для данного типа на прогнозное значение.

4. Если прогнозное значение (на горизонт прогноза) трафика старшего приоритета уменьшается, то наблюдается уменьшение ширины канала для данного типа на прогнозное значение.

5. Если происходит увеличение трафика i и i+1 приоритета, то наблюдается уменьшение трафика меньшего приоритета на величину прогноза, но не более чем на критическое значение.

Для трех типов трафика блок-схема приведена на рисунке 2.

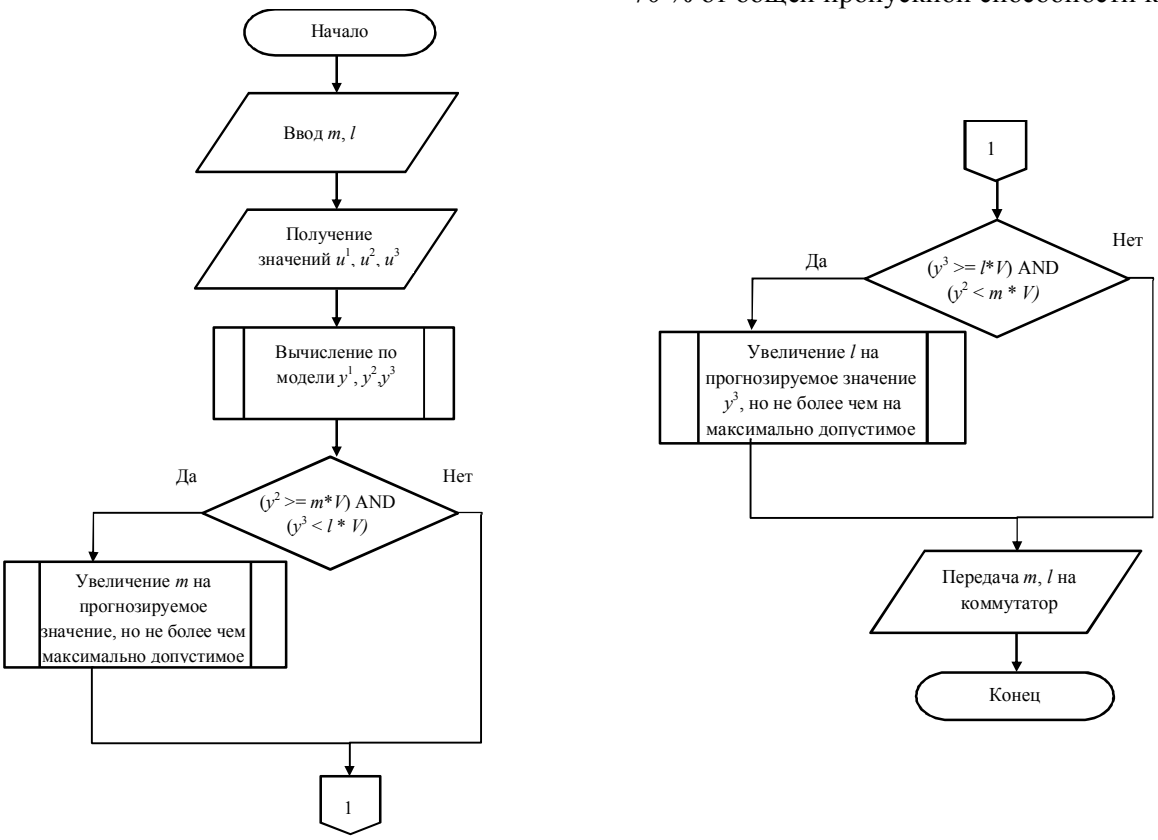


Рисунок 2 – Блок-схема алгоритма динамического разделения пропускной способностью

В приведённой блок-схеме использованы следующие обозначения: k – дискретное время; $u^1(k)$ – пропускная способность для первого типа трафика; $u^2(k)$ – пропускная способность для второго типа трафика; $u^3(k)$ – пропускная способность для третьего типа трафика; $y^2(k)$ – прогнозируемая пропускная способность для второго типа трафика; $y^3(k)$ – прогнозируемая пропускная способность для третьего типа трафика; V – общая пропускная способность канала; m – текущий коэффициент сжатия пропускной способности для второго типа трафика; l – текущий коэффициент сжатия пропускной способности для второго типа трафика.

В оперативном режиме происходит проверка прогнозируемого значения изменения пропускной способности трафика второго типа при условии передачи трафика третьего типа в следующий момент времени без потерь. Пропускная способность второго типа трафика увеличивается на установленный коэффициент макси

Первый тип трафика – служебный, значение пропускной способности не может быть изменено, второй тип – интернет трафик пользователей, третий тип – потоковое видео.

Алгоритм управления трафиком функционирует в случае, если загрузка канала превышает 70 % от общей пропускной способности канала.

сжатия для уменьшения потерь трафика третьего типа.

Далее происходит проверка прогнозируемого шага изменения пропускной способности при условии передачи трафика третьего типа на каждом шаге без потерь и не превышения коэффициента максимального сжатия пропускной способности для второго типа трафика. В результате устанавливаются параметры пропускных способностей для всех классов.

Алгоритм реализован в виде программного модуля программной конфигурации сети.

Заключение. Полученные результаты внедрения свидетельствуют об эффективности использования динамического управления в загруженных сетях, использующих гибридные облачные технологии. В частности, оно позволило функционировать без задержек системам реального времени: видеоконференции, передача образовательного мультимедиа контента в условиях большого количества подключений.

Работа выполнена при финансовой поддержке РФФИ (грант № 11-07-00772-а).

Библиографический список

1. *Облачные сервисы.* Взгляд из России / под ред. Е. Гребнева. – М.: CNews, 2011. – 282 с.

2. *Тейхриб А. П.* Состояние и перспективы передачи потоковых данных из веб-браузера в рамках организации облачного контакт-центра / А.П. Тейхриб // Научный журнал Кубанского государственного аграрного университета. 2012. № 8(82). – С. 449–465.

3. *Ковшов Е.Е.* «Облачные» вычисления при управлении инновациями и интеллектуальной собственностью промышленного предприятия / Е.Е. Ковшов, П.Н. Мартынов, О.В. Горяева, Е.Е. Чугреева // Вестник МГТУ Станкин. – 2012. – № 3. – С. 124–128.

4. *Комлева Н.В.* Информационно – образовательная среда современного вуза / Н.В. Комлева, В.В. Коновалов, Е.В. Никульчев // Экономика. Налоги. Право. 2011. – № 2. – С. 209–212.

5. *Nikulchev E.* Geometric Method of Reconstructing Evolution Equations from Experimental Data. Evolution Equations, Ed.: A.L. Claes. – New York: Nova Science Publishers, 2012. – P. 373–400.

УДК 621.391

А.В. Пруцков, Д.М. Цыбулько

ПРОБЛЕМНО-ОРИЕНТИРОВАННОЕ ОБЪЕКТНОЕ ПРОГРАММИРОВАНИЕ

Предложен подход к программированию, состоящий в записи программы как описания объектов некоторой предметной области и их взаимодействия. Данный подход назван проблемно-ориентированным объектным программированием. Для программирования используется минимальное число конструкций и команд. Предложенный подход позволяет быстро разрабатывать и отлаживать программы для решения задач в данной предметной области, оперативно вносить в них изменения.

Ключевые слова*: предметно-ориентированное программирование, объектное программирование.*

Введение. Сопровождение программного обеспечения является не менее важным и трудоемким процессом, чем разработка. Во многих предметных областях (например, бухгалтерский учет) требуется часто вносить изменения в программный код при каждом обновлении правил и взаимосвязей. Для этого требуется найти нужное место в программе, внести изменения и вновь провести отладку. Все эти действия затрудняют сопровождение программных продуктов и увеличивают время перехода на новое программное обеспечение.

В то же время внесение изменений можно упростить, если выделить в предметной области ключевые элементы, зависящие от изменений. Ключевые элементы необходимо вынести из программного кода и предоставить пользователю возможность редактировать их. При этом способ редактирования должен быть прост и понятен специалисту в данной предметной области, владеющему азами программирования.

Встроенные языки программирования некоторых программных систем (например, Microsoft Office) позволяют пользователям повысить их функциональность. Однако они сложны в освоении, поэтому используются пользователями редко или не в полной мере.

Целью работы является разработка способа программирования, использующего минимальное количество структур и инструкций, способного учитывать особенности предметной области и доступного для освоения специалистами в этих предметных областях.

Проблемно-ориентированное объектное программирование. Любую программу можно разделить на логически обособленные разделы, выполняющие определенные действия для решения локальной задачи. Разделы программы используют локальные и глобальные данные. Глобальные данные принимаются и передаются от одного раздела другому разделу. Локальные данные используются внутри раздела для выполнения внутренних действий.

Предлагается разделы программы описать как объекты – программные структуры, выполняющие определенные действия с глобальными данными и передающие управление в зависимости (или вне зависимости) от некоторых условий другим объектам. В зависимости от совершаемых действий каждый объект имеет свой тип.

Объект состоит из двух частей:

1) скрытой – реализующей действия и проверку условий передачи управления следующему объекту;

2) открытой – включающей следующие секции:

– поля – параметры, использующиеся при выполнении действий объекта; поля имеют значения по умолчанию;

– секцию инициализации, содержащую команды, выполняемые при получении объектом управления (например, получение глобальных данных);

– секцию условий, содержащую команды передачи глобальных данных и управления другим объектам в зависимости от условий, заданных для данного объекта.

Объекты обмениваются глобальными данными через особый объект (область) данных, который может быть организован как стек, очередь или память с прямым доступом (массив). Обращения к этому объекту со стороны других объектов осуществляются из их секции инициализации или секции условий.

Тогда вся программа сводится к совокупности объектов, передающих друг другу глобальные данные и управление.

Такая последовательная передача управления от одного объекта к другому повсеместно встречается в нашей жизни. Рассмотрим предметную область «Публикация статьи в рецензируемом издании». В этой предметной области существуют объекты: автор, секретарь, рецензент, корректор и печатник, каждый из которых выполняет определенные действия.

Упрощенный алгоритм публикации статьи включает следующие шаги.

1. Автор пишет статью и передает ее секретарю.

2. Секретарь получает статью, выбирает рецензента и передает ему статью.

3. Рецензент получает статью и дает отзыв.

Если отзыв отрицательный, то закончить алгоритм с отрицательным результатом.

Если отзыв положительный, но с замечаниями, то передать статью автору; автор исправляет статью и передает управление рецензенту (снова на шаг 3).

Если отзыв положительный без замечаний, то рецензент передает статью корректору.

4. Корректор получает статью и проверяет ее на ошибки.

Если найдены ошибки, то передать статью автору; автор исправляет статью и передает управление корректору (снова на шаг 4).

Если ошибки не найдены, то корректор передает статью печатнику.

5. Печатник получает статью и печатает ее. Закончить алгоритм с положительным результатом.

Каждый объект выполняет определенные действия. На шагах 3 и 4 использовалась условная передача управления другому объекту, а на шагах 1, 2, 5 – безусловная передача. Этот же пример показывает, как с помощью объектов организуются ветвления и циклы.

Передача данных (текста статьи) между объектами происходит через область данных.

Очевидно, что структура типа объекта (его поля, виды условий) будет зависеть от предметной области, где он будет использоваться. Описание решения задачи в некоторой предметной области с помощью объектов такой структуры назовем проблемно-ориентированным объектным программированием (ПООП).

Общее описание объекта и команд. Объект имеет следующее описание.

```
Имя = Тип
{
// Поля
Параметр1: значение;
Параметр2: значение;
...
// Секция инициализации
init
  команды;
// Секция условий
ifусловие1
  команды;
ifусловие2
  команды;
...
}
```

Если объект передает управление безусловно, то в секции условий используется ключевое слово «ifnocondition» (обозначается C). Очевидно, что в этом случае в секции присутствует только данное ключевое слово.

В секциях инициализации и условий объекта применяются два типа команд:

1) передачи управления:

– передача управления объекту Имя:

```
call Имя;
```

– завершение программы:

```
end;
```

2) получения и передачи глобальных данных:

– получение данных из области данных и помещение их поле Параметр1:

```
pop Параметр1;
```

– передача данных в область данных из поля Параметр2:

```
push Параметр2;
```

– получение данных из подобласти Область1 в поле Параметр1:

```
Параметр1 = Область1;
```

– запись данных из поля Параметр2 в подобласти Область2:

```
Область2 = Параметр2.
```

Другие структуры и команды в ПООП не используются.

Объектная программа состоит из последовательности описаний объектов и команды call Имя, запускающей выполнение программы.

Порядок проблемно-ориентированного объектного программирования. Для начала ПООП необходим подготовительный этап. Проектировщик изучает предметную область, выделяет типы объектов, их поля, условия и их действия. Программист реализует интерпретатор (или транслятор), выполняющий инструкции объектной программы.

Пользователь осуществляет ПООП, описывая объекты, передачу данных и управления между ними в секциях инициализации и условий. Пользователь не может менять структуру объекта.

Практическое применение проблемно-ориентированного объектного программирования. Авторами статьи было разработано Интернет-приложение перевода и проверки знаний правил образования количественных числительных естественных языков [1] и зарегистрировано в РОСПАТЕНТе (свидетельство о государственной регистрации программы для ЭВМ № 2012661379, Российская Федерация). В основе Интернет-приложения лежит метод обработки количественных числительных через промежуточный этап – модель числительных [2]. Пример формулировки задания, используемого при проверке знаний: «Запишите числительное "триста пять" на испанском языке».

Проверка знаний заключается в выдаче вопросов и в итоговой выдаче правильных ответов и результатов проверки. Однако для более объективной проверки необходимо составить ее сценарий. Сценарий проверки состоит из последовательности вопросов. Переход от одного вопроса к другому происходит в зависимости от правильного или неправильного ответа обучаемого. Также обучаемый получает пояснения и результат проверки. Необходимо, чтобы преподаватель имел возможность составлять сценарий проверки знаний. Используем для решения этой задачи ПООП.

После анализа предметной области «Проверка знаний правил образования количественных числительных естественных языков» были выделены следующие типы объектов.

Тип объекта Question (обозначается Q). Объект динамически генерирует вопрос по заданным параметрам, ответ к нему, выдает его пользователю, ожидает ответа, сравнивает его с правильным и результаты помещает в область данных.

Поля объекта:

RangeLow, RangeHi – нижняя и верхняя границы диапазона, из которого будет случайно выбрано число для генерации задания;

Limit – предельное значение количество вызовов объекта;

Out, In – направления перевода числительного.

Условия объекта:

iflimit (обозначается L) – выполнить команды, если количество вызовов объекта превышает значение Limit;

ifyes (Y) – выполнить команды, если обучаемый правильно ответил на вопрос;

ifno (N) – выполнить команды, если обучаемый неправильно ответил на вопрос.

Тип объекта Tip (T). Объект выдает подсказку и ожидает нажатия клавиши обучаемым после прочтения.

Поля объекта:

Text – текст подсказки.

Условие объекта:

ifnocondition.

Тип объекта Result (R). Объект извлекает из области данных результаты ответов на вопросы, анализирует их и выдает их обучаемому.

Поля объекта: нет.

Условие объекта:

ifnocondition.

Пусть необходимо реализовать следующий сценарий проверки знаний.

1. Проверить знания обучаемого единиц русского языка: «Запишите числительное 4 на русском языке». Если обучаемый правильно выполнит задание, то перейти к следующему шагу. Если обучаемый ответит неправильно, то выдать пояснение и повторить вопрос с другим числом. Если обучаемый 3 раза ответит неправильно, то закончить сценарий.

2–3. Повторить те же действия для десятков и сотен русского языка.

Взаимодействие объектов для реализации данного сценария проверки знаний представлено на рисунке 1.

Приведем описание некоторых объектов, использующихся в этом сценарии.

```
Q1 = Question
{
  RangeLow: 1;
  RangeHi: 9;
  Limit: 3;
  Out: Число;
  In: Русский;
  iflimit
    call R1;
  ifyes
```

```
    call Q10;
  ifno
    call T1;
}

T100 = Tip
{
  Text: "Текст 100 подсказки";
  ifnocondition
    call Q100;
}

R1 = Result
{
  ifnocondition
    end;
}
```

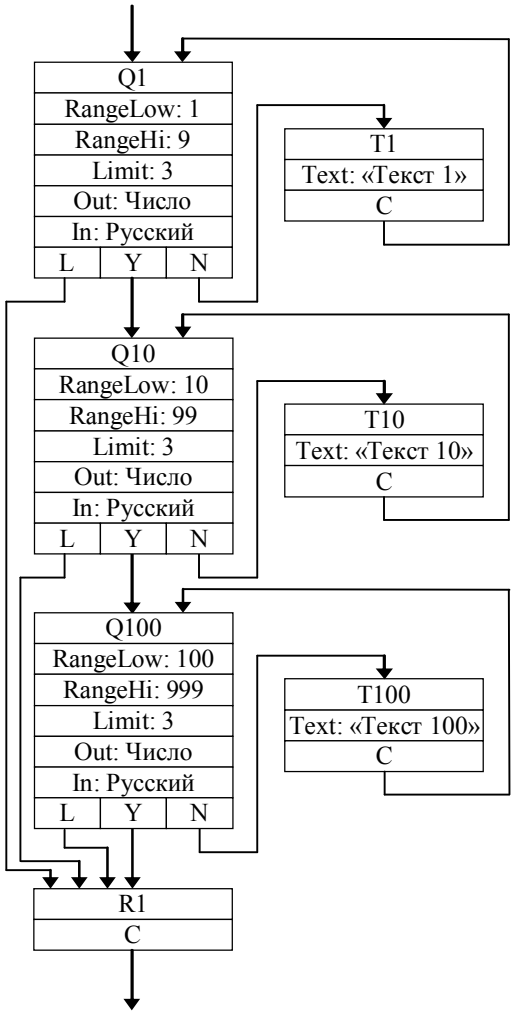


Рисунок 1 – Сценарий проверки знаний

При использовании ПООП преподаватель может самостоятельно изменить параметры вопросов и порядок их выдачи без привлечения программиста. Например, преподаватель без труда может изменить сценарий так, чтобы результат проверки знаний выдавался не в конце, а после ответа на каждый вопрос путем создания новых объектов и изменения передачи управления между объектами.

Из данного примера ПООП можно сделать следующие выводы.

1. ПООП использует простой язык и поэтому может быть легко освоено пользователем, знающим основы программирования.

2. ПООП позволяет легко разработать программу, быстро ее отладить и внести в нее изменения.

3. Объекты могут использоваться только в программах решения задач (проблем) в данной предметной области, потому предложенный подход называется проблемно-ориентированным.

Аналогичные объекты могут быть спроектированы также для проверки знаний морфологии естественных языков [3–4], а также других дисциплин.

Самоподобие объектов проблемно-ориентированного объектного программирования. Объект выполняет действие. Это действие можно описать как последовательную передачу управления между объектами нижнего уровня. Действия объектов нижнего уровня в свою очередь также можно описать через объекты и т. д. (рисунок 2).

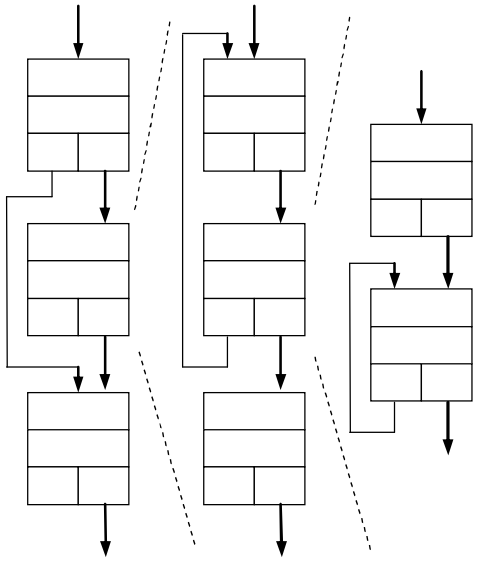


Рисунок 2 – Самоподобие объектов

Например, объект типа Question можно описать с помощью двух действий:

1) задать вопрос;

2) обработать ответ.

Действие 2 «Обработать ответ» можно представить вновь двумя действиями:

2.1) принять ответ;

2.2) проверить правильность ответа и т. д.

Таким образом, объекты в ПООП самоподобны (как фракталы). Однако уровни объектов имеют пределы. Объекты самого нижнего уровня реализуются с помощью языка программирования высокого или низкого уровня. Макси-

мальный уровень объектов определяется возможностью обобщения объектов предметной области.

Отличия проблемно-ориентированного объектного программирования от существующих разработок. Объектно-ориентированное программирование (ООП) подразумевает использование классических структур (условных операторов, операторов циклов), помимо объектов («объектно-ориентированное программирование»). ПООП исключает использование классических структур («объектное программирование»). В ООП «характерной чертой объектов является объединение данных (декларативной информации) и алгоритмов их обработки (процедуральной информации)» [5]. Объекты в ПООП – это действия, логически обособленные разделы программы, принимающие и передающие глобальные данные и управление другим объектам.

В отличие от фреймов [6] объекты в ПООП не имеют связанных с атрибутами (полями) процедур. Поля объектов – это параметры выполняемого действия. Однако ПООП может использоваться для представления знаний, например, в продукционной модели.

«Предметно-ориентированный язык (Domain Specific Language, DSL) – это язык программирования (или визуального конструирования), созданный для использования в рамках конкретной предметной области» [7]. «Такие языки создаются для каждой конкретной задачи» [7]. ПО-ОП – это универсальный подход, в котором от предметной области зависят только используемые объекты. Предметно-ориентированные языки предназначены для упрощения труда программистов, а ПООП – для изменения конечными пользователями последовательности действий при решении задач в определенной предметной области.

В отличие от нисходящего проектирования алгоритмов самоподобие объектов – это их свойство, а не способ их проектирования. При ПООП пользователь не должен строить нисходящую цепочку алгоритмов. Эти действия выполняет проектировщик. Пользователь работает с объектами самого верхнего уровня.

Перечисленные разработки были учтены при создании ПООП.

Заключение. В работе получены следующие результаты.

1. Предложен новый подход к программированию, заключающийся в выделении в предметной области объектов и записи программы как описания объектов с последовательной передачей управления между ними.

2. Приведен синтаксис записи объекта, и описаны команды, использующиеся для передачи управления и данных.

3. Представлен пример использования ПО-ОП в предметной области «Проверка знаний правил образования количественных числительных естественных языков».

Для реализации ПООП необходимо провести анализ предметной области, выделить все объекты и определить их поля и условия, использующиеся при решении задач. Также необходимо разработать интерпретатор, выполняющий объектную программу.

Однако все эти временные и материальные затраты компенсируются быстрой разработкой и отладкой программ, возможностью оперативно вносить изменения в программу.

В дальнейшем планируется разработка визуального конструктора объектных программ (как на рисунке 1), а также реализация с помощью ПООП выбора алгоритма в зависимости от входных данных [8], а также алгоритмов адаптивной ускоренной маршрутизации в динамических корпоративных сетях [9–11] и алгоритмов помехоустойчивого кодирования [12]. Объекты ПООП могут также использоваться для описания отношений концептов в онтологии (например, [13]). Такой подход позволяет классифицировать отношения, а также ввести их параметры.

Библиографический список

1. *Пруцков А.В., Цыбулько Д.М.* Интернет-приложение метода обработки количественных числительных естественных языков // Вестник Рязанского государственного радиотехнического университета. – 2012. – № 41. – С. 70–74.

2. *Пруцков А.В.* Обработка числительных естественных языков с помощью формальных грамматик и нормальных алгоритмов Маркова // Вестник Рязанского государственного радиотехнического университета. – 2009. – № 28. – С. 49–55.

3. *Пруцков А.В.* Генерация и определения форм слов естественных языков на основе их последовательных преобразований // Вестник Рязанского государственного радиотехнического университета. – 2009. – № 27. – С. 51–58.

4. *Пруцков А.В.* Применение информационных ресурсов для автоматизации обучения и проверки знаний // Информационные ресурсы России. – 2005. – № 1. – С. 18–20.

5. *Программирование. Структурирование программ и данных*: учебник для студ. учреждений высш. проф. образования / Н.И. Парфилова, А.Н. Пылькин, Б.Г. Трусов. – М.: Издательский центр «Академия», 2010. – 240 с.

6. *Минский М.* Фреймы для представления знаний. – М.: Мир, 1979. – 151 с.

7. *Лядова Л.Н.* Многоуровневые модели и языки DSL как основа создания интеллектуальных CASE-

систем // Труды Междунар. науч.-техн. конф. «Интеллектуальные системы» (AIS'08) и «Интеллектуальные САПР» (CAD-2008). Науч. издание в 4-х томах. – 2008. – Т. 2. – С. 37–41.

8. *Наумова О.А., Ульянов М.В.* Классификация методов построения алгоритмических систем // Вычислительные технологии. – 2011. – Т. 16. – № 1. – С. 105–118.

9. *Перепелкин Д.А., Перепелкин А.И.* Разработка алгоритмов адаптивной маршрутизации в корпоративных вычислительных сетях // Вестник Рязанской государственной радиотехнической академии. – 2006. – № 19. – С. 114–116.

10. *Корячко В.П., Перепелкин Д.А., Перепелкин А.И.* Повышение эффективности функционирования корпоративных сетей при динамических изменениях в их структуре и нагрузках на линиях связи // Вестник Рязанского государственного радиотехнического университета. – 2010. – № 33. – С. 49–55.

11. *Перепелкин Д.А., Перепелкин А.И.* Алгоритм адаптивной ускоренной маршрутизации в условии динамически изменяющихся нагрузок на линиях связи в корпоративной сети // Информационные технологии. – 2011. – № 3. – С. 2–7.

12. *Зубарев Ю.Б., Овечкин Г.В.* Помехоустойчивое кодирование в цифровых системах передачи данных // Электросвязь. – 2008. – № 12. – С. 58–61.

13. *Страхова З.В., Цуканова Н.И.* Онтология учебно-методического комплекса // Вестник Рязанского государственного радиотехнического университета. – 2013. – № 43. – С. 85–89.

УДК 004.89

Р.Е. Медведев

ИЗОМОРФИЗМ В ОНТОЛОГИЧЕСКОЙ МОДЕЛИ ЗНАНИЙ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ДИСТАНЦИОННОГО ОБУЧЕНИЯ

В статье рассматривается отображение структурных элементов учебного материала в соответствующие структурные элементы тестовых задач на основе разработанной прикладной онтологии интеллектуальной системы дистанционного обучения. Доказывается теорема об изоморфизме задачной и понятийной составляющих прикладной онтологии.

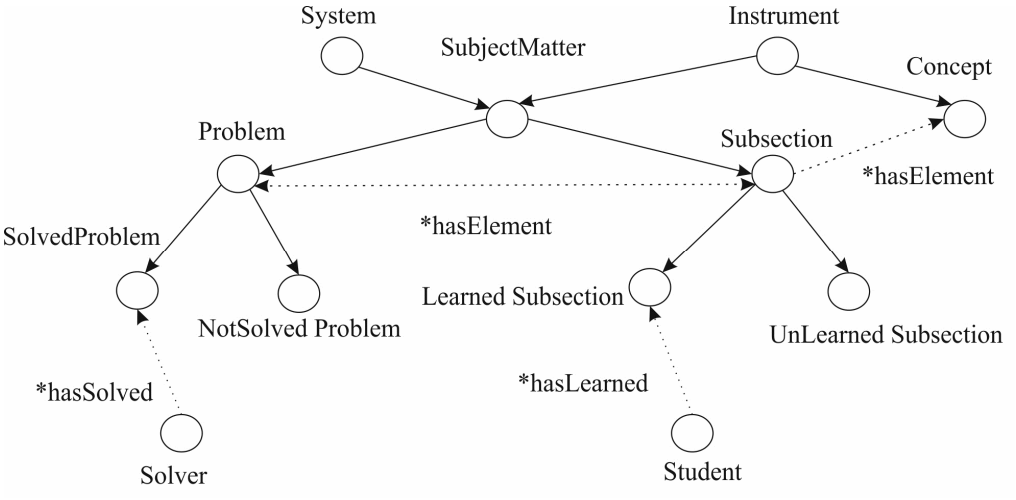
Ключевые слова: *дистанционное обучение, онтологическая модель, представление знаний, дескриптивная логика.*

Введение. Качество обучения является сложным многоплановым понятием [1]. Для решения нарастающих проблем, возникающих при разработке современных интеллектуальных систем дистанционного обучения, необходимо уделить особое внимание организации, структуризации, повторному использованию и интеграции разнородных знаний. Указанные требования можно выполнить, применив интеллектуальный анализ, который предлагают онтологии.

Существует множество определений понятия “онтология”. Под онтологией будем понимать спецификацию концептуализации предметной области [2, 3], где в качестве концептуализации выступает описание множества объектов и связей между ними. Если рассматривать онтологическую составляющую обучения, т.е. содержательную часть этого процесса, можно выделить факторы, непосредственно влияющие на это качество. Так, например, можно оценивать способности студента к наукам выбранного вида, его психологический тип, соматическую составляющую, а также скорость адаптации к новым задачам. Однако перечисленные факторы весьма сложно формализовать с достаточной степенью точности. По крайней мере, все они так или иначе будут «размыты» или нечетки. Несмотря на сказанное, существует более формальная составляющая, представляющая собой множество средств обучения: учебно-методические материалы, пособия, лабораторные устройства, стенды, тестовые задачи. Без существенной потери качества эти средства обучения можно, еще более упростив подход, свести к двум основным подмножествам: учебные материалы и тестовые задачи.

Цель работы – доказать возможность отображения структурных элементов учебного материала в соответствующие структурные элементы тестовых задач.

Теоретическая часть. Пример прикладной онтологии, в рамках которой можно было бы сформулировать ключевую фразу задачного подхода: “Для решения задачи студент должен усвоить все понятия, которые необходимы при ее решении”, приведен на рисунке. Основные формулы концептуального раздела дескриптивной логики TBox, адекватные цели получения



Семантическая сеть фрагмента прикладной онтологии

абстрактной части базы знаний “Обучение” приведены в таблице 1.

Таблица 1 – Формулы прикладной онтологии обучения

Понятие / Концепт	Формула
System	= ∀hasElement.Element
Causa	= ∃hasConsequence.Consequence
Instrument	= ∃hasActor.Person ∩ ∩ ∃isApplied ¯.Problem
Solver	= ∃isSolved.Problem ∩ Student
SolvedProblem	= ∃isSolved ¯ .Problem ∪ ∪ ∃isSolved.Solver
NotSolvedProblem	= ∃hasSolver. ⊥
Problem	= ∃isSolved ¯ .Problem ∪ ∪ ∃isSolved.Solver
SubjectMatter	⊆ System ∩ Instrument
SubjectMatter	= ∀hasElement. Subsection
Subsection	⊆ SubjectMatter
Problem	⊆ SubjectMatter
Subsection	= ∀hasElement.Problem ∩ ∩ ∀hasElement.Concept
LearnedSubsection	= Subsection ∩ ∩ (∃hasLearned¯. Subsection ∪ ∪ ∃hasLearned. Student)
UnLearnedSubsection	= Subsection ∩ ∃hasLearned. ⊥
Concept	⊆ Instrument
Student	⊆ Person
Process	= Dynamic
Learning	⊆ Process ∩ ∩ ∃hasConsequence. Solved-Problem
LearnTheConcept	= ∃hasLearn.Concept
ProblemConcepts	= Concept ∩ ∩ ∀isApplied ¯.SolvedProblem
LearnProblemConcepts	= Learning ∩ ∩∀ hasLearn .ProblemConcepts

При проектировании онтологической модели процесса обучения обе парадигмы (учебные материалы и тестовые задачи) имеют схожую структуру подразделов и опираются на базовые множества лежащих в их основе понятий. Исходя из изложенного, несложно сформулировать следующий постулат: «Чем точнее соответствие множества подразделов, понятий учебного материала и множества тестовых задач, тем более пригодной для обучения является прикладная онтология, опирающаяся на эти множества». Иначе говоря, от точности соответствия друг другу элементов этих множеств зависит качество процесса обучения.

С точки зрения математики имеет смысл рассматривать различные отображения структурных элементов учебного материала в соответствующие структурные элементы тестовых задач. К таким отображениям можно отнести, например, гомоморфизм, подразумевающий полное одностороннее отображение элементов одного множества в другое. Наиболее же качественным из отображений следует признать изоморфизм, предполагающий взаимно однозначное соответствие элементов базовых множеств.

Для математически строгого описания изоморфизма онтологий будем рассматривать множество правильно построенных формул (ППФ) этих онтологий как множество упорядоченных пар (C,F), где C – множество концептов, F – множество формул, стоящих в правой части таблицы 1. Таким образом, с помощью универсума $C \times F$ задается отношение дескрипции $R_d \subseteq C \times F$.

Далее рассмотрим только правые части F этого отношения как множество, представляющее собой проекцию на второй аргумент отношения. В этом случае выражения ППФ в части F можно представить как результат конечного

множества операций композиции элементов $\{c_1, c_2, ..., c_j, ..., c_k\}$ из множества C. Эти выражения получаются из имен концептов c_j с помощью специальных знаков дескриптивной логики: $\{\subseteq, =, \exists, \forall, \cup, \cap, \cdot, ^-\}$. Приведем далее семантику таких операций композиции Ω, учитывая замкнутость этих операций на множестве F.

Таблица 2 – Операции композиции ППФ

№ п/п	Обозначение операции из Ω	Семантика (результат операции)
1	$\omega_{\cap}(F_j, F_i)$	$F_j \cap F_i$
2	$\omega_{\cup}(F_j, F_i)$	$F_j \cup F_i$
3	$\omega_{\exists Rj}(c_i)$	$\exists R_j \cdot c_i$
4	$\omega_{\forall Rj}(c_i)$	$\forall R_j \cdot c_i$
5	$\omega_{\forall Rj-}(c_i)$	$\forall R_j^- \cdot c_i$
6	$\omega_{\exists Rj-}(c_i)$	$\exists R_j^- \cdot c_i$
7	$\omega_{:}(c_i)$	$: c_i$

Как следует из таблицы 2, в ней приведены лишь те композиционные операции, которые использованы в приведенной ранее прикладной онтологии.

Заметим также, что в ППФ можно выделить два отношения $\{\subseteq, =\}$, которые связывают левую и правую части онтологических определений.

В сделанных обозначениях на множестве ППФ онтологии можно задать алгебраическую систему A:

$$A = \langle F, \Omega, \{\subseteq, =\} \rangle. \quad (1)$$

В алгебраической системе A можно выделять подалгебры, замкнутые на своих множествах-носителях. Например, определим две подалгебры G и D для алгебры $\langle F, \Omega \rangle : G = (F_g, \Omega_g)$ и $D = (F_d, \Omega_d)$. В них справедливы соотношения:

$$F_g \subseteq F,\ \Omega_g \subseteq \Omega,\ F_d \subseteq F, \Omega_d \subseteq \Omega.$$

Предположим также, что эти подалгебры имеют один и тот же тип.

Алгебры G и D являются алгебрами одинакового типа. Алгебры G и D являются *гомоморфными* [4], так как существует отображение $\Gamma : M_g \to M_d$, удовлетворяющее условию:

$$\Gamma(\omega_g(f_{g1}, f_{g2})) = \omega_d(\Gamma(f_{g1}), \Gamma(f_{g2})), \quad (2)$$

где $f_{g1}, f_{g2} \subseteq F_g$. Независимо от того, выполнена ли сначала операция композиции в G и затем произведено отображение Γ, либо сначала произведено отображение Γ, а затем в D выполнена соответствующая операция композиции ω_d, результат будет одинаков.

Алгебры G и D являются *изоморфными* [5, 6] или *однозначно гомоморфными*, так как существует обратное отображение $\Gamma^{-1} : F_d \to F_g$, удовлетворяющее условию:

$$\omega_g(\Gamma^{-1}(f_{d1}), \Gamma^{-1}(f_{d2})) = \Gamma^{-1}(\omega_d(f_{d1}, f_{d2})), \quad (3)$$

где $f_{d1}, f_{d2} \subseteq F_d$.

Кроме того, потребуем для всех элементов множеств-носителей из G и D соответствие в алгебраических системах всех элементов множества-носителя каждого определяемого концепта отношениям $\{\subseteq, =\}$. То есть, если концепт из G соединен в итоговое описание отношением $\subseteq$, то и соответствующий ему концепт из D также должен быть описан с помощью отношения $\subseteq$. Если же при описании концепта из G было задействовано отношение $=$, то соответствующий концепт из D также должен быть описан с помощью равенства.

Равенство (3) приводится к равенству (2) следующим образом.

1. Заменим в условии (2) левые части на правые:

$$\omega_d(\Gamma(f_{g1}), \Gamma(f_{g2})) = \Gamma(\omega_g(f_{g1}, f_{g2})).$$

2. Применим Γ^{-1} к обеим частям получившегося равенства:

$$\Gamma^{-1}(\omega_d(\Gamma(f_{g1}), \Gamma(f_{g2}))) = \Gamma^{-1}(\Gamma(\omega_g(f_{f1}, f_{f2}))) = \omega_g(fg_{g1}, fg_{g2}).$$

3. Так как $\Gamma(f_{gi}) = f_{di}$, $f_{di} \in F_d$, тогда $f_{gi} = \Gamma^{-1}(f_{di})$, следовательно,

$$\omega_g(\Gamma^{-1}(f_{d1}), \Gamma^{-1}(f_{d2})) = \Gamma^{-1}(\omega_d(f_{d1}, f_{d2})).$$

Мощности основных множеств изоморфных алгебр G и D равны, т.е. изоморфизм алгебр G и D называется *изоморфизмом на себя* или *автоморфизмом*.

Теперь конструктивное доказательство изоморфизма двух локальных фрагментов онтологий может рассматриваться как полное структурное совпадение соответствующих ППФ с точностью до определения соответствия концептов из таблицы 1.

В нашем случае проверяется соответствие ППФ для следующих пар концептов: (SubjectMstter, Concept), (Problem, Subsection), (solvedProblem, LearnedSubsection), (UnsolvedProblem, UnlearnedSubsection), (Solver, Student), которое является очевидным из графического представления (см. рисунок), если удалить из онтологии дугу (Instrument, SubMatter). Это достигается несложным удалением выражения 8 таблицы 1:

$$\text{SubjectMatter} \subseteq \text{System} \cap \text{Instrument}.$$

Поскольку это выражение является лишь фрагментом интерфейсной части прикладной онтологии с общей онтологией и далее в алгоритмах нигде не будет задействовано, удаление выражения 8 можно считать правомерным. В то же время можно ввести и другое, более корректное, исправление в прикладную онтологию, например, добавив выражение:

$$\text{Problem} \subseteq \text{System} .$$

Таким образом, доказана теорема об изоморфизме задачной и понятийной составляющих прикладной онтологии “Образование”.

Заключение. Значение доказанной теоремы заключается не только в том, что изоморфизм прикладных онтологий открывает путь к их совместному применению как пары («Материал», «Тест») в обучающих системах, но и дает возможность использовать аналогичное доказательство при автоматическом подборе материала в раздел онтологии ABox.

Библиографический список

1. "Education and Computing", v.1, 1995. // Hebenstreit J. Computers in education – The next step. – p. 37–43.

2. *Gruber T.R.* A translation approach to portable ontologies // Knowledge Acquisition, 1993, V. 5(2), P. 199–220.

3. *Гладун А.Я., Рогушина Ю.В.* Онтологии в корпоративных системах // Корпоративные системы. 2006. № 1. – С. 41–47.

4. *Zilles S.N.* Types, Algebras, and modelling. - Concept. Modelling: Perspective of Artificial Intelligence, Databases, and Programming Languages. – N.Y. e.a., 1984. – pp. 441–450.

5. *Кузнецов О.П., Адельсон-Вельский Г.М.* Дискретная математика для инженера. – 2-е изд., перераб. и доп. – М.: Энергоатомиздат, 1988. – 480 с.

6. Логический подход к искусственному интеллекту: от классической логики к логическому программированию: пер. с франц./ А. Тейз, П. Грибомон, Ж. Луи и др. – М.: Мир, 1990. – 432 с.

УДК 615.47

А.Н. Варнавский, А.В. Антоненко

ОПРЕДЕЛЕНИЕ ВОДИТЕЛЯ РИТМА СЕРДЦА ОПЕРАТОРА ЧЕЛОВЕКО-МАШИННЫХ СИСТЕМ С ПОМОЩЬЮ НЕЧЕТКОЙ ЛОГИКИ

Описана возможность определения водителя ритма сердца оператора с помощью совместного использования нелинейных интегральных преобразований и нечеткой логики, позволяющей осуществить анализ в режиме реального времени. Предложена система нечеткого логического вывода, определены функции принадлежности и правила нечетких продукций. Показано, что отношение значения степени истинности условия правила-победителя к значению степени истинности условия любого другого правила равно как минимум 3.

Ключевые слова: *водитель ритма, электрокардиосигнал, система нечеткого логического вывода, функции принадлежности, правила нечетких продукций, степень принадлежности высказываний.*

Введение. В процессе эксплуатации человеко-машинных систем на транспорте или в промышленности возникает необходимость контроля состояния человека-оператора для решения таких задач, как поддержание работоспособности, оптимального психофизиологического состояния оператора, оценка уровня психологического воздействия на него, степени адаптации к случайным или постоянно действующим нагрузкам. Эти задачи направлены на повышение безопасности и снижение рисков работы таких систем и предотвращение возникновения аварий и катастроф по причине человеческого фактора.

Одним из способов информативной оценки состояния оператора является анализ сердечного ритма, осуществляющийся на основе обработки электрокардиосигнала (ЭКС). В ходе такого анализа необходимо осуществлять обнаружение различных приступов и преждевременных внеочередных сокращений сердца, являющихся экстрасистолическими. В последнем случае необходимость их обнаружения даже у здоровых лиц обусловлена тем, что, во-первых, появление экстрасистолических комплексов свидетельствует о психоэмоциональных перегрузках, во вторых, экстрасистолические интервалы должны быть

исключены из массива длительностей кардиоциклов при расчете параметров вариабельности сердечного ритма.

Для распознавания форм экстрасистол и других нарушений и приступов необходимо определять источник возбуждения, или так называемый водитель ритма. Данная задача применительно к анализу состояния оператора и повышения безопасности человеко-машинных систем должна осуществляться в режиме реального времени.

В настоящее время автоматизированное определение водителя ритма фактически осуществляется только для синусового и желудочкового ритма в режиме реального времени. Остальные виды либо не выделяют, либо выделяют не в режиме реального времени.

Цель работы — разработка способа определения водителя ритма сердца оператора человеко-машинных систем с помощью нечеткой логики в режиме реального времени.

Виды водителей ритма сердца. Для определения водителя ритма необходимо проанализировать взаимосвязь возбуждений предсердий и желудочков в каждом кардиоцикле, то есть определить, какой отдел сердца возбуждается первым.

В норме электрический импульс, возникающий в синоаурикулярном узле, распространяется по предсердиям сверху вниз. На ЭКС во II отведении фиксируются положительные зубцы Р. Возбуждение предсердий при этом всегда предшествует возбуждению желудочков, поэтому положительные зубцы P_{II} регистрируются перед каждым комплексом QRS. В большинстве случаев в каждом отведении они имеют одинаковую форму и обычно располагаются на одинаковом расстоянии от комплекса QRS.

При отсутствии этих признаков диагностируются различные варианты несинусового ритма. К ним относятся предсердные ритмы, ритмы из АВ-соединения, желудочковые ритмы.

В тех случаях когда источник возбуждения располагается в нижних отделах предсердий (например, в области коронарного синуса), электрический импульс по предсердиям распространяется в обратном направлении (снизу вверх) и на ЭКС во II отведении регистрируются отрицательные зубцы Р, которые предшествуют комплексам QRS. Поскольку движение волны возбуждения по желудочкам не нарушено, регистрируются обычные неизменённые (узкие) комплексы QRS.

Если водитель ритма локализуется в АВ-соединении, то возбуждение желудочков происходит обычным путем — сверху вниз, а предсердий — ретроградно, снизу вверх. Поэтому на ЭКС регистрируются нормальные неизмененные комплексы QRS и отрицательные зубцы Р. При этом, если эктопический импульс одновременно достигает предсердий и желудочков, зубец Р наслаивается на комплекс QRS и не виден на ЭКС. Если эктопический импульс вначале достигает желудочков и только потом предсердий, отрицательный зубец Р располагается после комплекса QRS.

Если источником возбуждения является проводящая система желудочков (ножки и ветви пучка Гиса или волокна Пуркинье), то речь идет о желудочковом ритме. Возбуждение проводится по желудочкам необычным путем: оно сначала охватывает тот желудочек, в котором находится эктопический водитель ритма, и только потом медленно достигает противоположного желудочка. Вследствие этого комплексы QRS расширены и деформированы. Возбуждение не проводится на миокард предсердий, поэтому отсутствует постоянная закономерная связь комплексов QRS с зубцами Р: желудочки возбуждаются в своем медленном ритме, а предсердия в своем обычном ритме, источником которого продолжает оставаться синоаурикулярный узел [1].

В таблицу 1 сведены признаки водителей ритма сердца для ЭКС второго отведения.

Таблица 1 – Признаки водителей ритма сердца

Водитель ритма	Р зубцы	QRS-комплексы
Синоаурикулярный узел	Положительные с постоянной одинаковой формой, предшествуют *QRS*-комплексам	Обычные неизменные
Предсердия	Отрицательные, предшествуют *QRS*-комплексам	Обычные неизменные
АВ-соединение	Отсутствуют либо отрицательные после *QRS*-комплексов. Отсутствие закономерной связи в положении с *QRS*-комплексами	Обычные неизменные
Желудочки	Отсутствие закономерной связи в положении с *QRS*-комплексами	Расширены и деформированы

Для определения источника возбуждения необходимо выделить зубцы ЭКС, после чего, применяя логические правила к опорным точкам положения зубцов, можно осуществить классификацию водителя ритма [2]. Логические правила могут быть представлены в виде нейронной сети [3].

Однако из-за того, что при желудочковом ритме и ритме из АВ-соединения отсутствует связь между наличием и положением зубцов Р и QRS-комплексов, применение логических правил может приводить к ошибкам классификации в режиме реального времени. В таких случаях оправданно использование нечеткой логики.

Выделение зубцов ЭКС. Анализ ЭКС следует начинать с предварительной обработки сигнала, включающей устранение высокочастотных шумов, дрейфа изолинии [4]. Далее для определения водителя ритма необходимо осуществить выделение элементов ЭКС и опорных точек в каждом кардиоцикле. Эту задачу можно решить с использованием нелинейных интегральных преобразований [3], результатом которых будут являться сигналы:

$y^{(P)}$ – сигнал наличия положительного зубца Р;

$y^{(-P)}$ – сигнал наличия отрицательного зубца Р;

$y^{(QRS)}$ – сигнал наличия обычного QRS-комплекса;

$y^{(QRS*)}$ – сигнал наличия деформированного QRS-комплекса.

Данные сигналы принимают максимальные значения в области соответствующего элемента.

Опорные точки в каждом кардиоцикле можно выделить на ТР-сегменте [5]. В этом случае формируется логический сигнал s_{OT}, который принимает значение 1 в области ТР-сегмента и значение 0 в остальных случаях.

Система нечеткого логического вывода для определения вида водителя ритма сердца. Предлагается система нечеткого логического вывода (СНЛВ) для определения вида водителя ритма сердца, которая обладает рядом преимуществ по сравнению с ее реализациями в виде схемы булевой логики [2] и рассмотренного в [3] нейросетевого подхода. Данные преимущества заключаются в том, что СНЛВ ввиду оперирования нечеткими понятиями учитывает неоднозначность поступающих исходных данных, в частности информации о наличии и положении зубцов. Достоверность классификации определяется показателем степени принадлежности лингвистической переменной (ЛП) [6] к тому или иному лингвистическому высказыванию (ЛВ).

В качестве исходных данных СНЛВ использует полученную ранее совокупность признаков видов водителя ритма сердца на основе анализа поведения выходных сигналов наличия зубцов $y^{(-P)}$, $y^{(P)}$, $y^{(QRS)}$, $y^{(QRS*)}$. Причем на вход СНЛВ подаются значения максимумов сигналов наличия зубцов $y^{(-P)}$, $y^{(P)}$, $y^{(QRS)}$, $y^{(QRS*)}$, достигнутых в течение кардиоцикла, в моменты времени, соответствующие окончанию каждого кардиоцикла, т.е. в моменты s_{OT} =1. Также учитываются моменты времени появления максимумов сигналов $y^{(-P)}$, $y^{(P)}$, $y^{(QRS)}$ относительно друг друга.

Общий вид системы нечеткого логического вывода [7] представлен на рисунке 1. Нужно отметить, что в рассматриваемой системе отсутствует блок дефаззификатора, так как выход системы представлен лингвистической переменной.

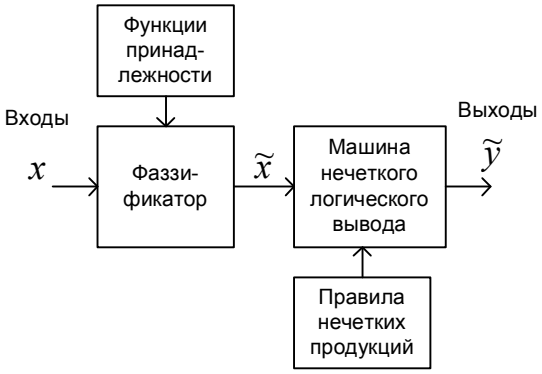


Рисунок 1 – Общий вид системы нечеткого логического вывода

Для задания структуры СНЛВ необходимо определить: входные и выходные ЛП; функции принадлежности (ФП) для каждого из ЛВ, значения которых могут принимать ЛП; правила нечетких продукций.

В рассматриваемом случае СНЛВ имеет шесть входных и одну выходную ЛП. Входные ЛП отражают выявленные ранее признаки водителя ритма. Так, ЛП1 «Отрицательный зубец Р», ЛП2 «Положительный зубец Р», ЛП3 «QRS-комплекс», ЛП4 «Деформированный QRS-комплекс» могут принимать значения ЛВ «Нет» и «Да» в соответствии с их ФП. Сигналы $y^{(-P)}$, $y^{(P)}$, $y^{(QRS)}$, $y^{(QRS*)}$ нормируются на этапе анализа, поэтому четкие значения переменных ЛП1 – ЛП4 изменяются в диапазоне [0, 1]. На рисунке 2 приведен пример ФП для ЛП2. Интервал значений [a, b] задает интервал неопределенности.

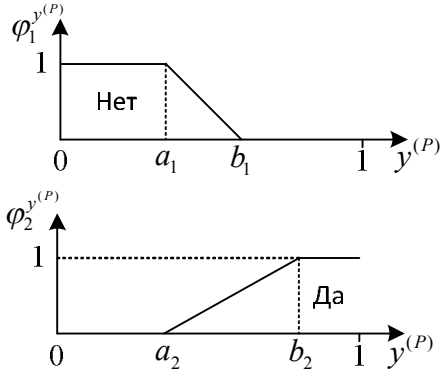


Рисунок 2 – Функции принадлежности лингвистических высказываний «Да» и «Нет» для ЛП2

В данном случае (см. рисунок 2) используются z-образная $\phi_1^{y^{(P)}}$ и s-образная $\phi_2^{y^{(P)}}$ функции принадлежности вида

$$\phi_1^{y^{(P)}}(x)=\begin{cases}1, \text{если } x \le a_1;\\ \dfrac{b_1-x}{b_1-a_1}, \text{если } a_1 < x \le b_1;\\ 0 \text{ иначе}\end{cases}$$

и

$$\phi_2^{y^{(P)}}(x)=\begin{cases}0, \text{если } x \le a_2;\\ \dfrac{x-a_2}{b_2-a_2}, \text{если } a_2 < x \le b_2;\\ 1 \text{ иначе.}\end{cases}$$

Две дополнительные входные ЛП отражают время появления максимумов сигналов $y^{(-P)}$, $y^{(P)}$ относительно момента появления максимума сигнала $y^{(QRS)}$. Так, ЛП5 «Момент зубца Р» и ЛП6 «Момент зубца -Р» принимают значения двух ЛВ «Зубец предшествовал QRS-комплексу» и «Зубец следовал за QRS-комплексом».

Выходная ЛП «Водитель ритма» может принимать значения четырех ЛВ: «Синусовый ритм», «Предсердный ритм», «Ритм из АВ-соединения» и «Желудочковый ритм». В качестве выходного значения принимается ЛВ, имеющее наибольшее значение степени принадлежности. В связи с этим отсутствует необходимость этапа дефаззификации и определения ФП для ЛВ выходной ЛП.

Правила нечетких продукций СНЛВ для определения вида водителя ритма сердца основываются на данных [1], представленных в табличном виде (таблица 1). В данном случае правила нечетких продукций запишутся следующим образом.

Правило 1: Если Положительный зубец Р = Да **И** (Отрицательный зубец Р = Нет **ИЛИ** Отрицательный зубец Р = Да **И** Момент зубца – Р = Зубец следовал за QRS-комплексом) **И** QRS-комплекс = Да **И** Деформированный QRS-комплекс = Нет **И** Момент зубца Р = Зубец предшествовал QRS-комплексу, **ТО** Водитель ритма = Синусовый ритм.

Правило 2: Если (Положительный зубец Р = Нет **ИЛИ** Положительный зубец Р = Да **И** Момент зубца Р = Зубец следовал за QRS-комплексом) **И** Отрицательный зубец Р = Да **И** QRS-комплекс = Да **И** Деформированный QRS-комплекс = Нет **И** Момент зубца -Р = Зубец предшествовал QRS-комплексу, **ТО** Водитель ритма = Предсердный ритм.

Правило 3: Если (Положительный зубец Р = Нет **ИЛИ** Положительный зубец Р = Да **И** Момент зубца Р = Зубец следовал за QRS-комплексом) **И** (Отрицательный зубец Р = Нет **ИЛИ** Отрицательный зубец Р = Да **И** Момент зубца -Р = Зубец следовал за QRS-комплексом) **И** QRS-комплекс = Да **И** Деформированный QRS-комплекс = Нет, **ТО** Водитель ритма = = Ритм из АВ-соединения.

Правило 4: Если QRS-комплекс = Нет **И** Деформированный QRS-комплекс = Да, **ТО** Водитель ритма = Желудочковый ритм.

В общем виде алгоритм определения водителя ритма на основе предложенной системы нечеткого логического вывода состоит из следующих шагов.

1. Определение четких значений входных сигналов, представляющих собой интегральные преобразования ЭКС.

2. Определение значений степеней принадлежности входных ЛП к ЛВ путем вычисления функций принадлежностей $\phi_1^{y^{(P)}}$ и $\phi_2^{y^{(P)}}$ с соответствующими параметрами (таблица 2).

3. Вычисление степени истинности условия каждого правила нечеткого вывода. При этом условие правила представляется многоуровневой комбинацией подусловий. На первом уровне подусловия объединяются по правилу **И** в следующем виде:

Подусловие 1 **И** Подусловие 2 **И** … (1)

На втором уровне подусловия объединяются по правилу **ИЛИ**:

Подусловие 1 **ИЛИ** Подусловие 2 **ИЛИ** … (2)

На третьем уровне подусловия объединяются по правилу **И** в виде (1). Как правило, оказывается достаточным трех уровней комбинации подусловий. Полученные два типа объединения подусловий (1) и (2) можно соответственно описать выражениями

$$\acute{o}=\min(x_1,x_2,...) \text{ и } \acute{o}=\max(x_1,x_2,...),$$

где $x_1,x_2,...$ – список входных аргументов, представляющих собой значения степени истинности соответствующих подусловий; $\acute{o}$ – степень истинности объединения подусловий.

Таким образом, вычисление степени истинности условия n-го правила нечеткого вывода сводится к обобщенному выражению

$$y_n=\min_i \max_j \min_k (a_{ijk}),$$

где a_{ijk} – значения степени истинности подусловий.

4. Определение правила с максимальной степенью истинности условия, которое в свою очередь делает вывод о водителе ритма. При этом считаем выявление водителя ритма надежным в том случае, если значение степени истинности условия правила-победителя как минимум

в 3 раза больше, чем значение степени истинности условия любого другого правила. В противном случае можно считать выявление водителя ритма не однозначным.

Таблица 2 – Параметры функций принадлежности анализируемых ЛП

Лингвистические переменные	Лингвистические высказывания	Функция принадлежности	Параметры функции	
			a	*b*
Отрицательный зубец Р	«Да»	S-образная	0,2	0,7
	«Нет»	Z-образная	0,2	0,7
Положительный зубец Р	«Да»	S-образная	0,2	0,7
	«Нет»	Z-образная	0,2	0,7
QRS-комплекс	«Да»	S-образная	0,6	0,9
	«Нет»	Z-образная	0,6	0,9
Деформированный QRS-комплекс	«Да»	S-образная	0,6	0,9
	«Нет»	Z-образная	0,6	0,9
Момент зубца Р	«Зубец предшествовал QRS-комплексу»	Z-образная	-0,5	0,1
	«Зубец следовал за QRS-комплексом»	S-образная	-0,1	0,5
Момент зубца -Р	«Зубец предшествовал QRS-комплексу»	Z-образная	-0,5	0,1
	«Зубец следовал за QRS-комплексом»	S-образная	-0,1	0,5

Пример определения водителя ритма. На рисунке 3 представлен пример электрокардиосигнала (рисунок 3, а) и его преобразований для выделения положительного Р-зубца (рисунок 3, б), отрицательного Р-зубца (рисунок 3, в), QRS-комплекса (рисунок 3, г) и деформированного QRS-комплекса (рисунок 3, д). Для выделенного пунктирными линиями кардиоцикла численные значения анализируемых переменных после нормирования будут иметь величины, приведенные в таблице 3. Нечеткие логические операции И/ИЛИ определяются как функции min/max соответственно.

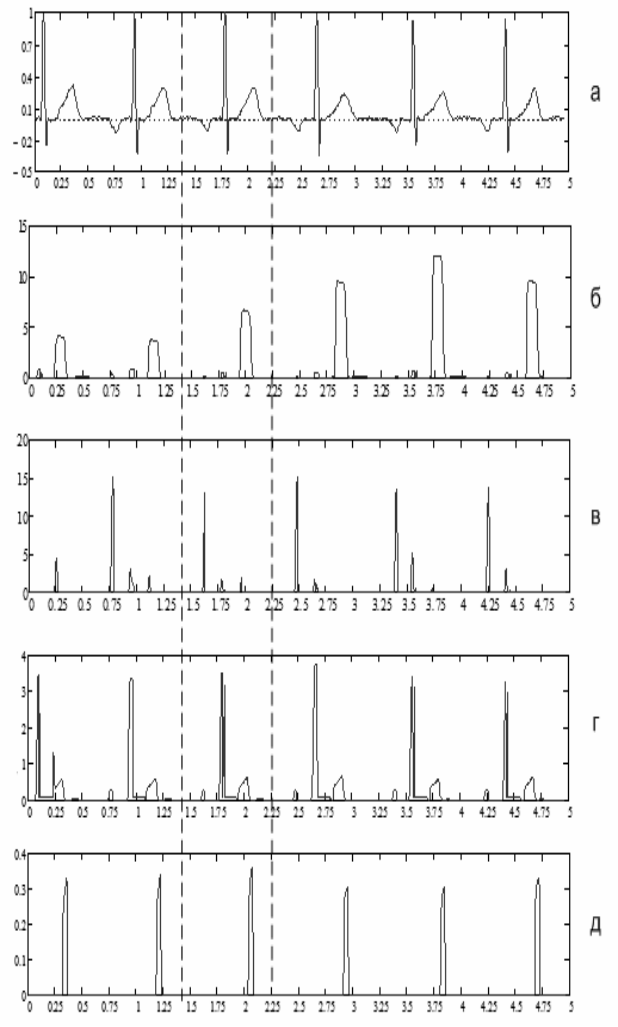


Рисунок 3 – Электрокардиосигнал (а) и результаты его преобразования для выделения положительного Р-зубца (б), отрицательного Р-зубца (в), QRS-комплекса (г) и деформированного QRS-комплекса (д)

Таблица 3 – Значения анализируемых переменных для рассматриваемого кардиоцикла (см. рисунок 3)

Лингвистические переменные	Нормированные значения сигналов преобразования	Степень принадлежности к высказыванию	
		«Да»	«Нет»
Отрицательный зубец Р	0,87	0,9	0,1
Положительный зубец Р	0,58	0,76	0,24
QRS-комплекс	0,87	0,9	0,1
Деформированный QRS-комплекс	0,035	0	1

Окончание таблицы 3

Лингвистические переменные	Нормированные значения сигналов преобразования	Степень принадлежности к высказыванию	
		«Да»	«Нет»
	Время появления относительно QRS-комплекса, с	«Зубец предшествовал QRS-комплексу»	«Зубец следовал за QRS-комплексом»
Момент зубца P	0,25	0	0,58
Момент зубца -P	-0,15	0,42	0

Подставляя значения степеней принадлежности высказываний ЛП в выражения правил нечетких продукций, получаем значения истинности условий:

Правило 1: min(0,76; max(0,1; min(0,9; 1)); 0,9; 1; 0) = 0;

Правило 2: min(max(0,24; min(0,76; 0,58)); 0,9; 0,9; 1; 0,42) = 0,42;

Правило 3: min(max(0,24; min(0,76; 0,58)); max(0,1; min(0,9; 0)); 0,9; 1) = 0,1;

Правило 4: min(0,1; 0) = 0.

Среди полученных значений максимальной является истинность условия второго правила, которое превышает значение истинности других правил минимум в 4,2 раза, из чего заключается, что водителем ритма сердца, ЭКС которого представлен на рисунке 3, а, являются предсердия.

Заключение. В работе описана возможность определения водителя ритма сердца оператора в режиме реального времени с помощью совместного использования нелинейных интегральных преобразований и нечеткой логики. Предложенная система нечеткого логического вывода использует z-образные и s-образные функции принадлежности и 4 правила нечетких продукций. Показано, что отношение значения степени истинности условия правила-победителя к значению степени истинности условия любого другого правила равно как минимум 3.

Достоинством использования нечеткой логики является учет неоднозначности поступающих исходных данных, в частности результатов нелинейных преобразований и соответствующей им информации о наличии и положении зубцов, особенно в случае ритма из АВ-соединения и желудочкового ритма, когда отсутствует закономерная связь в положении зубцов P с QRS-комплексами.

Реализация рассмотренного способа в блоке контроля состояния оператора человеко-машинных систем в промышленности и на транспорте позволит более достоверно определять формы экстрасистол и других нарушений и приступов, благодаря чему можно оценивать психоэмоциональные перегрузки оператора, оперативно реагировать на изменение его состояния, тем самым снижая риски работы систем от «человеческого фактора».

Работа проводилась при финансовой поддержке Министерства образования и науки РФ в рамках ФЦП «Научные и научно-педагогические кадры инновационной России» (госконтракт № 14.740.11.1108).

Библиографический список

1. *Мурашко В.В, Струтынский А.В.* Электрокардиография: учеб. пособие. – М.: ООО «МЕДпресс», 1998. – 313 с.

2. *Варнавский А.Н., Михеев А.А.* Автоматическое определение водителя ритма сердца оператора в человеко-машинной системе //Проектирование и технология электронных средств. 2009. № 2. – С. 27–31.

3. *Варнавский А.Н., Мусолин А.К.* Нейросетевой модуль контроля состояния оператора для снижения рисков работы автоматизированных производств // Вестник РГРТУ. 2010. №12. – С. 36–43.

4. *Мельник О.В., Михеев А.А.* Обработка и анализ электрокардиосигнала в режиме реального времени // Биотехносфера. 2009. № 4. – С. 17–20.

5. *Варнавский А.Н., Мельник О.В, Михеев А.А.* Метод выделения опорной точки в каждом кардиоцикле //Биомедицинские технологии и радиоэлектроника. 2005. № 1–2. – С. 36–39.

6. *Леоненков А.В.* Нечеткое моделирование в среде MATLAB. – СПб.: БХВ-Петербург, 2005. – 736 с.

7. *Штовба С.Д.* Проектирование нечетких систем средствами MATLAB. – М.: Горячая линия – Телеком, 2007. – 288 с.