ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн "Сириус"

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 12
December
2015

Published since September 2010

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/pi.html, e-mail:prin@novtex.ru

УДК 004.89

В. В. Грибова, д-р техн. наук, зам. дир. по науч. работе, e-mail: gribova@iacp.dvo.ru,
А. С. Клещев, д-р физ.-мат. наук, проф., гл. науч. сотр., e-mail: kleschev@iacp.dvo.ru,
Д. А. Крылов, канд. техн. наук, вед. инженер-программист, e-mail: dmalkr@gmail.com,
Ф. М. Москаленко, канд. техн. наук, науч. сотр., e-mail: philipmm@iacp.dvo.ru,
В. А. Тимченко, канд. техн. наук, науч. сотр., e-mail: vadim@iacp.dvo.ru,
Е. А. Шалфеева, канд. техн. наук, доц., ст. науч. сотр., e-mail: shalf@iacp.dvo.ru,
Институт автоматики и процессов управления Дальневосточного отделения Российской академии наук, г. Владивосток

Базовая технология разработки интеллектуальных сервисов на облачной платформе IACPaaS. Часть 1. Разработка базы знаний и решателя задач

Данная работа открывает цикл из трех работ, посвященных описанию базовой технологии разработки интеллектуальных мультиагентных сервисов и их компонентов с использованием инструментальных средств облачной платформы IACPaaS. Технология направлена на снижение трудоемкости разработки и сопровождения интеллектуальных сервисов и предполагает участие (без посредников или дополнительного обучения) в этих процессах экспертов предметных областей, задачей которых является формирование и поддержание в актуальном состоянии информационных компонентов прикладных сервисов платформы на протяжении всего их жизненного цикла. Первая часть посвящена описанию концептуальной архитектуры платформы IACPaaS, технологии разработки базы знаний и решателя задач, сборки интеллектуального сервиса на их основе.

Ключевые слова: *интеллектуальные системы, мультиагентные системы, технология разработки интеллектуальных систем, иерархические орграфы, агентно-ориентированное программирование, облачные платформы, облачные сервисы*

Введение

Разработка и сопровождение интеллектуальных систем (ИС) или систем, основанных на знаниях (состоящих, в общем случае, из базы знаний, решателя задач и пользовательского интерфейса), являются сложными и трудоемкими процессами. Данные процессы имеют свои особенности, поэтому стандартная технология разработки программных систем не может быть напрямую спроецирована на разработку ИС. В первую очередь это связано с тем обстоятельством, что входящая в архитектуру ИС база знаний должна разрабатываться и сопровождаться экспертами предметной области и быть им понятной. Традиционный подход к разработке ИС, основанный на использовании оболочек, имеет ряд недостатков, которые подробно обсуждены в работах [1—6]. Отмеченные недостатки особенно остро проявляются при сопровождении ИС.

Один из современных широко используемых подходов к разработке программных систем основан на использовании концепции облачных вычислений. Такой подход, с одной стороны, обеспечивает доступность программных систем широкому кругу пользователей, с другой стороны, позволяет на протяжении всего жизненного цикла программной системы осуществлять ее сопровождение, поскольку все находящиеся в эксплуатации компоненты системы по-прежнему контролирует разработчик. К настоящему времени в рамках данного подхода для разработки ИС созданы редакторы информационного наполнения ИС, которые можно задействовать при создании новых ИС [7, 8]; разработаны и широко используются платформы [9—11], которые поддерживают отдельные этапы цикла разработки облачных программных систем.

Предложенная в работах [12, 13] открытая семантическая технология компонентного проектирования ИС (OSTIS) основана на идее модульного проектирования ИС из многократно используемых семантически совместимых компонентов, хранящихся в библиотеке. При этом не ограничивается

аппарат используемых средств разработки, языки программирования, операционные системы, платформы, поэтому для обеспечения совместимости компонентов разработчикам необходимо спроектировать и реализовать абстрактную унифицированную логико-семантическую модель проектируемой ИС и ее интерпретатор (либо усовершенствовать существующий в библиотеке). По мнению авторов, такой поход удобен для интеграции в едином информационном пространстве уже созданных ИС. Однако в этом случае сложно обеспечить высокоуровневую инструментальную поддержку всех этапов технологии разработки ИС, усложняется ее сопровождение, что противоречит наиболее важному требованию, снижаются и усложняются возможности повторного использования компонентов, снижается эффективность работы ИС вследствие издержек дополнительного интерпретатора модели.

Резюмируя представленные выше доводы, в настоящее время можно констатировать отсутствие принятой в качестве стандарта технологии, позволяющей разрабатывать жизнеспособные ИС. Попрежнему ощущается острая потребность в средствах разработки ИС, а также в повторном использовании компонентов ИС.

Для решения перечисленных выше проблемных вопросов в области разработки и сопровождения жизнеспособных ИС в работах [14, 15] предложена облачная платформа IACPaaS, поддерживающая следующие технологические принципы разработки, сопровождения и использования облачных ИС:

- базу знаний, решатель задач и пользовательский интерфейс разрабатывают раздельно;
- все информационные ресурсы (онтологии, знания, данные) имеют единое унифицированное декларативное представление;
- формирование и сопровождение знаний осуществляется экспертами предметной области на основе моделей онтологий;
- пользовательский интерфейс редакторов для эксперта генерируется по модели онтологии;
- метод решения задачи разбивается на подзадачи, где каждой подзадаче ставится в соответствие решающий ее агент (входящий в состав некоторого решателя задач);
- для доступа агентов к информационным ресурсам, имеющим унифицированное представление, предусмотрены программные интерфейсы;
- ИС предоставляется пользователю как облачный сервис.

Платформа IACPaaS поддерживает:

- ♦ базовую технологию разработки прикладных и специализированных инструментальных (интеллектуальных) сервисов с использованием базовых инструментальных сервисов платформы, поддерживающих эту технологию;
- ♦ множество специализированных технологий разработки прикладных и специализированных инструментальных (интеллектуальных) сервисов с использованием специализированных инструментальных сервисов платформы, поддерживающих эти технологии.

Специализированные технологии (по сравнению с базовой) и инструментальные средства их поддержки, с одной стороны, как правило, накладывают определенные ограничения (по области применения и/или классу решаемых задач) на разрабатываемые с их использованием сервисы, а с другой стороны, за счет учета специфики проблемной области и/или класса решаемых задач обеспечивают более высокоуровневую поддержку разработки последних.

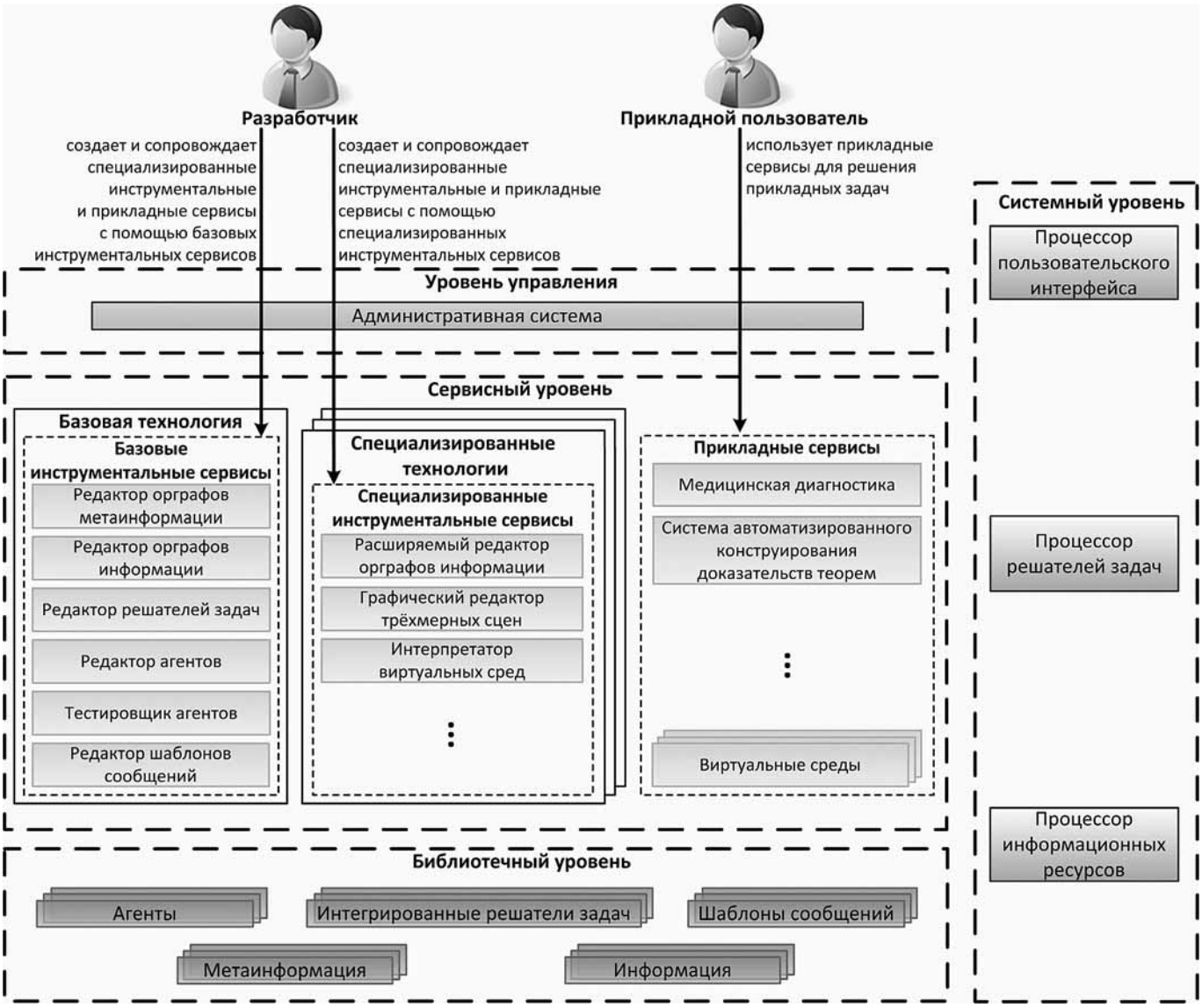
В данной и планируемых далее к публикации статьях описывается базовая технология разработки ИС как облачных мультиагентных сервисов.

1. Облачная платформа IACPaaS

Облачная платформа IACPaaS представляет собой программно-информационный интернет-комплекс, который предназначен для поддержки разработки, управления и удаленного использования прикладных и инструментальных мультиагентных облачных сервисов (прежде всего интеллектуальных) и их компонентов. Комплекс основан на технологии облачных вычислений и обеспечивает удаленный доступ конечным пользователям к ИС, а разработчикам и управляющим — к средствам создания ИС и управления ими. Концептуальная четырехуровневая архитектура платформы IACPaaS представлена на рисунке.

1. *Системный уровень* (уровень Виртуальной машины). Виртуальная машина платформы IACPaaS состоит из процессора информационных ресурсов (ПИР), процессора решателей задач (ПРЗ) и процессора пользовательского интерфейса (ППИ). Каждый из перечисленных процессоров предназначен для поддержки соответствующих компонентов ИС. На данном уровне обеспечиваются доступ к Фонду единиц хранения платформы IACPaaS (средствами ПИР), запуск и работа сервисов (поддерживаемые ПРЗ), а также взаимодействие сервисов с пользователями (посредством ППИ).

2. *Библиотечный уровень* (уровень Фонда). Фонд платформы IACPaaS представляет собой совокупность единиц хранения различных видов и уровней общности (содержательно это хранимые данные и знаний, их онтологии, метаонтологии, решатели задач, агенты, шаблоны сообщений, модель абстрактного пользовательского интерфейса и т. п.). Представлением в Фонде любой единицы хранения является *информационный ресурс*. Структурно Фонд разделен на предметные области, а те, в свою очередь, на разделы. Раздел содержит единицы хранения, среди которых присутствуют повторно используемые при разработке и сопровождении прикладных, а также базовых и специализированных инструментальных (интеллектуальных) сервисов компоненты. К их числу относятся решатели задач, агенты, шаблоны сообщений, посредством которых агенты взаимодействуют между собой, а также другие единицы



Концептуальная четырехуровневая архитектура платформы IACPaaS

хранения (базы знаний и данных, онтологии и т. п.), которые могут быть двух типов — представляющие метаинформацию и информацию.

3. *Сервисный уровень.* Данный уровень — совокупность сервисов, каждый из которых представлен множеством взаимодействующих агентов, обрабатывающих информационные ресурсы Фонда. Выделяют прикладные сервисы, создаваемые разработчиками для решения прикладных задач пользователей, а также базовые и специализированные инструментальные сервисы, обеспечивающие поддержку соответствующих технологий разработки сервисов и необходимые для развития Фонда платформы.

4. *Уровень управления.* Данный уровень представлен базовым инструментальным сервисом платформы *Административная система*, которая предназначена для обеспечения контролируемого доступа к функциональным возможностям платформы IACPaaS и управления правами доступа на использование прикладных и инструментальных сервисов, разработанных на базе данной платформы.

Базовая технология разработки прикладных и специализированных инструментальных облачных мультиагентных сервисов и их компонентов в общем случае включает следующие процессы: сборка интеллектуального сервиса из компонентов; разработка информационных ресурсов (обрабатываемых интеллектуальным сервисом); разработка решателя задач интеллектуального сервиса и его связывание с формальными параметрами и интерфейсом интеллектуального сервиса; разработка агентов решателя задач; разработка шаблонов сообщений; разработка интерфейса интеллектуального сервиса.

2. Сборка интеллектуального сервиса из компонентов

Под *интеллектуальным сервисом* облачной платформы IACPaaS будем понимать интеллектуальную интернет-систему, решающую прикладные задачи, относящиеся к некоторой предметной области, и доступную для использования через административную систему платформы IACPaaS. Интеллектуальный сервис есть пара: $service = \langle app, \langle\langle in_1, ..., in_k\rangle, \langle out_1, ..., out_p\rangle\rangle\rangle$, $k \geqslant 0$, $p \geqslant 0$, где:

- *app* — *интегрированный* с формальными параметрами и пользовательским интерфейсом *решатель задач* (его устройство описано в разд. 4);
- $\langle in_1, ..., in_k\rangle$ — возможно пустой кортеж информационных ресурсов Фонда (баз знаний, данных), доступных решателю задач только для чтения, — *входных* информационных ресурсов;
- $\langle out_1, ..., out_p\rangle$ — возможно пустой кортеж (длины *p*) информационных ресурсов Фонда, доступных решателю задач как для чтения, так и для модификации (вплоть до полного формирования всего содержимого), — *выходных* информационных ресурсов[1].

Разделение информационных и программных компонентов сервиса преследует следующие цели.

1. Независимая разработка решателей задач и информационных ресурсов соответствующими специалистами (базы знаний, например, формируют эксперты в соответствующих предметных областях).

2. Компоненты обоих типов хранятся в Фонде и являются повторно используемыми (один решатель задач может быть связан с разными информационными ресурсами и наоборот).

В обоих случаях совместимость обеспечивается на уровне формальных параметров решателей задач. Таким образом, для того чтобы разработать сервис, в Фонде, во-первых, должен присутствовать подходящий решатель задач — *app**, а во-вторых, если $\langle in_1, ..., in_k\rangle \cup \langle out_1, ..., out_p\rangle \neq \varnothing$, то для каждого формального параметра этого решателя задач должен присутствовать хотя бы один информационный ресурс, принадлежащий области допустимых значений этого формального параметра, — фактический параметр.

Формальным параметром решателя является информационный ресурс, представляющий метаинформацию (онтологию), а *фактический параметр* — один из информационных ресурсов, представляющих информацию, который сформирован по этой метаинформации.

Если для некоторого формального параметра выбранного интегрированного решателя задач в Фонде отсутствует фактический параметр, то соответствующий информационный ресурс необходимо разработать по технологии, описанной в подразд. 3.2. Если нужный интегрированный решатель задач отсутствует в Фонде, то его необходимо разработать по технологии, описанной в разд. 4.

Сборка интеллектуального сервиса *service** выполняется следующим образом.

1. В Фонде выбирается подходящий для решения задачи интегрированный решатель задач — *app**.

2. Если $\langle in_1, ..., in_k\rangle \cup \langle out_1, ..., out_p\rangle \neq \varnothing$, то для каждого формального параметра решателя задач *app** в Фонде выбирается подходящий для решения задачи фактический параметр, принадлежащий области допустимых значений этого формального параметра. Таким образом устанавливается соответствие между формальными и фактическими параметрами.

3. Если $\langle in_1, ..., in_k\rangle \cup \langle out_1, ..., out_p\rangle \neq \varnothing$, то средствами Административной системы выполняются связывание интегрированного решателя задач *app** с выбранными фактическими параметрами, поставленными в соответствие его формальным параметрам, и создание на их основе сервиса *service**. Если же $\langle in_1, ..., in_k\rangle \cup \langle out_1, ..., out_p\rangle = \varnothing$ (т. е. решение сервисом *service** задачи не подразумевает обработку информационных ресурсов, за исключением, возможно, собственных информационных ресурсов *app**), то средствами Административной системы выполняется создание сервиса *service** только на основе интегрированного решателя задач *app**.

Собранный таким образом сервис *service** может быть запущен пользователем через Административную систему.

В целях получения отладочной информации о процессе работы сервиса *service** виртуальная машина предоставляет возможность агентам, входящим в состав решателя задач *app**, вносить записи, содержащие информацию о выполняемых ими действиях и состоянии обрабатываемых ими объектов, в журнал *app**. Для просмотра, а также, при необходимости, для удаления журналов используется сервис платформы — Редактор решателей задач.

3. Разработка информационных ресурсов

3.1. Общие сведения

Все информационные ресурсы в Фонде платформы IACPaaS имеют единое унифицированное декларативное представление — в форме размеченного иерархического однородного бинарного орграфа с возможными петлями и циклами — посредством чего достигается универсальность их обработки.

В соответствии с общепринятым в настоящее время двухуровневым подходом к формированию информационных единиц [16—24] будем различать два типа информационных ресурсов по уровню абстракции представляемой ими информации: информационные ресурсы, представляющие метаинформацию (абстрактный уровень), и информационные ресурсы, представляющие информацию (конкретный уровень). Таким образом, орграф метаинформации описывает язык (структуру), в терминах которого формируется множество орграфов информации. Орграф метаинформации формируется на языке представления орграфов метаинформации, основан-

[1] Содержимое выходных информационных ресурсов формируется в результате работы сервиса. На момент запуска сервиса через Административную систему содержимое выходных информационных ресурсов может быть пустым (см. подразд. 3.1).

ном на орграфовой связной двухуровневой модели информационных ресурсов, к которой может быть сведена наиболее распространенная в настоящее время объектно-ориентированная двухуровневая модель информационных единиц [21—24].

Обрабатываемые интеллектуальным сервисом информационные ресурсы разрабатывают по технологии, описанной в подразд. 3.2.

3.2. Разработка информационных ресурсов, представляющих информацию

При разработке некоторого информационного ресурса, представляющего информацию — *ig**, информационный ресурс, представляющий его метаинформацию — *mg**, должен присутствовать в Фонде. В противном случае *mg** необходимо разработать по технологии, описанной в подразд. 3.3.

Разработка информационного ресурса *ig** состоит из двух этапов.

1. Создание с использованием Административной системы в разделе некоторой предметной области Фонда нового информационного ресурса *ig** с пустым содержимым (орграф представлен единственной корневой вершиной) с указанием в качестве метаинформации *mg** — информационного ресурса, представляющего метаинформацию и описывающего область допустимых значений (множество информационных ресурсов) некоторых формальных параметров интегрированных решателей задач.

2. Формирование пользователем содержимого информационного ресурса *ig** с использованием Редактора орграфов информации[2], в котором процесс редактирования *ig** управляется орграфом метаинформации *mg**.

В процессе работы Редактор орграфов информации формирует и поддерживает соответствие между дугами орграфов *ig** и *mg**. Формирование орграфа *ig** осуществляется пользователем "сверху вниз": от вершин, представляющих сложносоставные понятия, к атомарным, начиная с корневой вершины орграфа *ig**. Преимущество данного подхода над подходом к формированию орграфовой структуры "снизу вверх" [7] состоит в том, что первый является более естественным для человека: орграф информации "разворачивается" пользователем строго в соответствии со структурой (шаблоном), заданной в орграфе метаинформации. Во втором случае, напротив, пользователю необходимо четко представлять и постоянно держать в уме всю структуру (связи между вершинами) формируемого орграфа (от чего в первом случае он полностью избавлен).

Web-интерфейс Редактора орграфов информации отображает уже сформированную часть подграфа *ig** и набор интерфейсных элементов для выполнения возможных действий пользователя по редактированию значений полей активных вершин подграфа *ig** и порождению исходящих из них дуг, который формируется Редактором орграфов информации на основе вершин орграфа *mg**, соответствующих активным вершинам *ig**, и множества выходящих из них дуг в орграфе *mg** (если таковые имеются). Если порождаются дуги, исходящие из активной вершины *ig**, то их концы становятся активными вершинами. Редактирование *ig** возможно лишь в тех пределах, которые не нарушают соответствие между орграфами *ig** и *mg**.

Пользовательский интерфейс Редактора орграфов информации генерируется по соответствующему орграфу метаинформации. Это означает, что если последний описывает онтологию знаний или данных в некоторой предметной области, то эксперты предметной области могут создавать и сопровождать базы знаний или данных в привычной для них системе понятий (без посредников в лице системных аналитиков).

3.3. Разработка информационных ресурсов, представляющих метаинформацию

Разработка информационного ресурса *mg** также состоит из следующих двух этапов.

1. Создание с использованием Административной системы в разделе некоторой предметной области Фонда нового информационного ресурса *mg** с пустым содержимым (орграф представлен единственной корневой вершиной), с указанием в качестве метаинформации информационного ресурса, содержащего описание языка представления орграфов метаинформации (присутствующем в Фонде платформы IACPaaS априори).

2. Формирование содержимого информационного ресурса *mg** "сверху вниз" (начиная с корневой вершины орграфа) с использованием Редактора орграфов метаинформации в терминах языка представления орграфов метаинформации.

Пользователями Редактора орграфов метаинформации являются носители разного вида метаинформации, обычно это инженеры знаний и системные аналитики в различных сферах профессиональной деятельности. Модель процесса редактирования, положенная в основу Редактора орграфов метаинформации, во многом совпадает с той, что положена в основу Редактора орграфов информации. Имеющиеся отличия обусловлены лишь формализмом представления соответствующих орграфов, а также тем, что орграф метаинформации может иметь произвольный вид (ограничения накладываются лишь выразительными средствами языка представления орграфов метаинформации), в то время как ограничения на структуру орграфа информации накладываются видом орграфа ее метаинформации.

4. Разработка решателя задач интеллектуального сервиса

4.1. Общие сведения

Под *решателем задач* облачной платформы IACPaaS будем понимать компонент некоторого интеллектуального сервиса, обрабатывающий доступное ему содержимое Фонда и инкапсулирующий бизнес-логику решения задачи.

[2] Если содержимое информационного ресурса *ig** должно формироваться (по метаинформации *mg**) другим сервисом, у которого *ig** должен быть указан в качестве выходного фактического параметра, то данный этап пропускается.

Решатель задач представляет собой набор агентов платформы IACPaaS, взаимодействующих друг с другом посредством обмена сообщениями. Среди них, в общем случае, у двух агентов — *Корневого агента* и *Интерфейсного контроллера* — особые роли.

- *Корневой агент* — агент, которому процессор решателей задач платформы IACPaaS посылает инициализирующее сообщение при запуске сервиса, чем инициирует работу сервиса (т. е. среди множества агентов решателя задач данный агент всегда начинает работать первым).
- *Интерфейсный контроллер* — агент, взаимодействие которого с системным агентом платформы *Вид* обеспечивает сопряжение пользовательского интерфейса с решателем задач (в случае разработки сервиса с пользовательским интерфейсом). Агент Вид создает и посылает сообщения Интерфейсному контроллеру (и только ему) в ответ на события, генерируемые в пользовательском интерфейсе.

В процессе выполнения сервиса агенты, входящие в состав решателя задач, связываются друг с другом динамически начиная с *корневого агента* или *Интерфейсного контроллера* (при обработке событий, генерируемых в пользовательском интерфейсе). При этом агент *sender* может послать сообщения множеству агентов $\{receiver_1, ..., receiver_n\}$, где $receiver_i$ ($i = 1, ..., n$) может быть одним из следующих агентов:

- агент, от которого *sender* получил сообщение (за исключением случая посылки процессором решателей задач инициализирующего сообщения корневому агенту);
- агент, указанный в принимаемом агентом *sender* сообщении;
- агент, название или указатель на которого задается в коде агента *sender*, т. е. получатель однозначно определяется на этапе разработки агента *sender*.

Таким образом, если функциональные возможности разрабатываемого решателя задач уже реализованы в виде некоторого набора присутствующих в Фонде агентов, то необходимо разработать:

- корневой агент, в задачу которого должна входить подготовка входных данных, пересылка их в сообщении агентам-адресатам, а также, возможно, получение и обработка итогового результата и завершение работы сервиса;
- агенты-адаптеры, если требуется связать существующие в Фонде агенты, которые друг с другом напрямую не взаимодействуют.

Если присутствующие в Фонде агенты не обеспечивают программную поддержку необходимых функций или обеспечивают ее частично, то следует разработать соответствующие агенты по технологии, описанной во второй части данного цикла работ.

4.2. Связывание решателя задач с формальными параметрами и интерфейсом интеллектуального сервиса

Для того чтобы решатель задач можно было использовать на этапе сборки различных сервисов, в Фонде должен быть информационный ресурс особого вида, представляющий декларативную спецификацию решателя задач и его связей с другими компонентами, необходимыми для создания сервисов — спецификацию *интегрированного решателя задач*. В спецификации содержится информация о назначении и устройстве решателя задач, о его корневом агенте, а также задается связь решателя задач с формальными параметрами (если сервисы должны обрабатывать информационные ресурсы, т. е. иметь фактические параметры) и пользовательским интерфейсом (в случае разработки сервисов с пользовательским интерфейсом).

Интегрированный с формальными параметрами и пользовательским интерфейсом решатель задач может быть представлен кортежем следующего вида (необязательные компоненты здесь и далее заключены в скобки [и]): $app = \langle name, descr, a_{root}, [a_{ui}], [wiki\_page], \langle IN_1, ..., IN_k\rangle, \langle OUT_1, ..., OUT_p\rangle, \langle own_1, ..., own_q\rangle, \langle log_1, ..., log_s\rangle\rangle$, $k \geqslant 0$, $p \geqslant 0$, $q \geqslant 0$, $s \geqslant 0$, где

- *name* — название интегрированного решателя задач на естественном языке, отображаемое всем авторизованным пользователям платформы IACPaaS при просмотре содержимого Фонда через Административную систему;
- *descr* — содержательное описание назначения решателя задач на естественном языке, включающее описание решаемой задачи или класса задач, а также входных и выходных данных, и доступное всем авторизованным пользователям платформы IACPaaS при просмотре содержимого Фонда через Административную систему;
- a_{root} — корневой агент решателя задач;
- a_{ui} — агент Интерфейсный контроллер;
- *wiki_page* — стартовая wiki-страница решателя задач на web-сайте платформы — wiki-страница, на которую выполняется перенаправление при запуске сервиса через Административную систему;
- $\langle IN_1, ..., IN_k\rangle$ — возможно пустой кортеж входных формальных параметров решателя задач. Каждый IN_i ($i = 1, ..., k$) есть информационный ресурс Фонда, представляющий метаинформацию и определяющий множество информационных ресурсов, представляющих информацию, которому должен принадлежать фактический параметр in_i;
- $\langle OUT_1, ..., OUT_p\rangle$ — возможно пустой кортеж выходных формальных параметров решателя задач, каждый OUT_j ($j = 1, ..., p$) есть информационный ресурс Фонда, представляющий метаинформацию и определяющий множество информационных ресур-

сов, представляющих информацию, которому должен принадлежать фактический параметр out_j;

- $\langle own_1, ..., own_q \rangle$ — возможно пустой кортеж собственных информационных ресурсов решателя задач, представляющих информацию и доступных ему как для чтения, так и для модификации; собственные информационные ресурсы при этом используются для хранения собственных данных и настроек решателя задач, управляющих логикой его работы и т. п., а также являются разделяемыми для всех сервисов, в состав которых входит данный решатель задач;
- $\langle log_1, ..., log_s \rangle$ — возможно пустой кортеж информационных ресурсов, каждый из которых представляет собой журнал работы агентов решателя задач, формируемый при запуске некоторого сервиса, в состав которого входит данный решатель, и заполняемый вплоть до завершения его работы.

Помимо преимуществ, указанных в разд. 2 (сформулированных как цели разделения информационных и программных компонентов сервиса), такое устройство интегрированного решателя задач (подразумевающее выделение среди агентов решателя специализированного агента Интерфейсный контроллер) дает возможность отделить (и, соответственно, сделать эти процессы независимыми) разработку и сопровождение бизнес-логики решателя задач от разработки и сопровождения пользовательского интерфейса интеллектуального сервиса, а также привлечь к этим видам работ соответствующих независимых специалистов.

Для того чтобы сформировать декларативную спецификацию интегрированного решателя задач, необходимо, чтобы выполнялись перечисленные далее условия.

1. В Фонде должен присутствовать агент, являющийся корневым агентом решателя задач, — a_{root}.

2. В случае разработки сервиса с пользовательским интерфейсом — на web-сайте платформы IACPaaS должна быть создана стартовая wiki-страница для интегрируемого решателя задач, а в Фонде — присутствовать агент, являющийся его интерфейсным контроллером, — a_{ui} (данный агент может быть повторно используемым и, таким образом, входить в состав различных решателей задач).

3. Если $\langle IN_1, ..., IN_k \rangle \neq \varnothing$, то в Фонде должны присутствовать все информационные ресурсы IN_i ($i = 1, ..., k$).

4. Если $\langle OUT_1, ..., OUT_p \rangle \neq \varnothing$, то в Фонде должны присутствовать все информационные ресурсы OUT_j ($j = 1, ..., p$).

5. Если $\langle own_1, ..., own_q \rangle \neq \varnothing$, то в Фонде должны присутствовать все информационные ресурсы own_l ($l = 1, ..., q$).

Если некоторый информационный ресурс IN_i (OUT_j) отсутствует в Фонде, то его необходимо разработать по технологии, описанной в подразд. 3.3. Если некоторый информационный ресурс own_l отсутствует в Фонде, то его необходимо разработать по технологии, описанной в подразд. 3.2. Если агенты a_{root} и/или a_{ui} отсутствуют в Фонде, то их необходимо разработать по технологии, которая будет описана во второй статье данной серии, отражающей результаты работ по рассматриваемой проблематике.

Декларативная спецификация интегрированного решателя задач формируется в представленные далее два этапа.

1. Создание с использованием Административной системы в Фонде платформы IACPaaS нового информационного ресурса (с названием, совпадающим с названием интегрированного решателя задач, и пустым содержимым) по хранящейся в Фонде метаинформации *Структура решателей задач*, описывающей онтологию декларативных спецификаций интегрированных решателей задач платформы.

2. Формирование содержимого созданного информационного ресурса с использованием Редактора решателей задач, в котором процесс редактирования управляется метаинформацией Структура решателей задач. Специфицирование решателя задач состоит в задании:

- описания решателя задач (на естественном языке);
- названия стартовой wiki-страницы решателя задач на web-сайте платформы (если сервис должен поддерживать пользовательский интерфейс);
- значения логического признака, указывающего на необходимость журналирования работы агентов решателя задач;
- корневого агента решателя задач;
- агента Интерфейсный контроллер (если сервис должен поддерживать пользовательский интерфейс);
- входных и выходных формальных параметров, а также собственных информационных ресурсов решателя задач.

Корневой агент и агент Интерфейсный контроллер задают путем создания ссылок на информационные ресурсы в Фонде платформы IACPaaS, представляющие декларативные спецификации соответствующих агентов. Формальные параметры и собственные информационные ресурсы интегрируемого решателя задач также задают путем создания ссылок на соответствующие информационные ресурсы в Фонде платформы IACPaaS.

После того как декларативная спецификация интегрированного решателя задач сформирована, последний с разрешения администратора предметной области, для которой разработан решатель задач, средствами Административной системы должен быть переведен в режим публичного доступа. Находящийся в публичном доступе интегрированный решатель задач может быть использован на этапе сборки интеллектуальных сервисов.

Заключение

В работе рассмотрена концептуальная архитектура облачной платформы IACPaaS для разработки, управления и удаленного использования прикладных и инструментальных мультиагентных облачных сервисов и их компонентов. Описана технология разработки обрабатываемых интеллектуальным сервисом баз знаний и их онтологий, решателя задач интеллектуального сервиса и его связывания с онтологиями баз знаний и пользовательским интерфейсом сервиса.

Разработка интеллектуального сервиса состоит в его сборке на основе хранящихся в Фонде информационных ресурсов платформы IACPaaS повторно используемых баз знаний и интегрированного (на основе соответствующей декларативной спецификации) с онтологиями этих баз знаний и, возможно, с пользовательским интерфейсом решателя задач. Разработка решателя задач также сводится к его сборке из повторно используемых хранящихся в Фонде программных компонентов — агентов.

Базы знаний, обрабатываемые интеллектуальными сервисами, формируются по соответствующим онтологиям. Между этими двумя уровнями поддерживается четкое терминологическое разделение, которое предполагает формирование онтологий инженерами знаний через интерфейс, ориентированный на метаязык, а баз знаний — экспертами через сгенерированный по соответствующей онтологии интерфейс.

Работа выполнена при частичной финансовой поддержке РФФИ (проект 13-07-00024, проект 14-07-00299 и проект 15-07-03193) и программы ФНИ (Дальний Восток).

Список литературы

1. **Грибова В. В., Клещев А. С., Шалфеева Е. А.** Управление интеллектуальными системами // Известия РАН. Теории и системы управления. 2010. № 6. С. 122—137.

2. **Грибова В. В., Клещев А. С.** Технология разработки интеллектуальных сервисов, ориентированных на декларативные предметные базы знаний. Часть 1. Информационные ресурсы // Информационные технологии. 2013. № 9. С. 7—11.

3. **Грибова В. В., Клещев А. С.** Технология разработки интеллектуальных сервисов, ориентированных на декларативные предметные базы знаний. Часть 2. Решатель задач. Пользовательский интерфейс // Информационные технологии. 2013. № 10. С. 10—14.

4. **Musen M.** Technology for building intelligent systems: from psychology to engineering // Nebraska Symposium on Motivation / Ed. by B. Shuart, W. Spaulding, and J. Poland. Lincoln, Nebraska: U Nebraska P. 2009. Vol. 52. P. 145—184.

5. **Sonar R. M.** An Enterprise Intelligent System Development and Solution Framework // International Journal of Applied Science, Engineering and Technology. 2007. Vol. 4, N. 1. P. 34—39.

6. **Schalkoff R. J.** Intelligent Systems: Principles, Paradigms and Pragmatics // Sudbury, Mass. Jones and Bartlett Publishers, 2011. 758 p.

7. **The Protégé** Ontology Editor and Knowledge Acquisition System [Электронный ресурс]. URL: http://protege.stanford.edu/ (дата обращения 18.02.2015).

8. **Орлов В. А., Клещев А. С.** Компьютерные банки знаний. Универсальный подход к решению проблемы редактирования информации // Информационные технологии. 2006. № 5. С. 25—31.

9. **TopQuadrant** [Электронный ресурс]. URL: http://www.topquadrant.com/ (дата обращения 18.02.2015).

10. **Sanderson D.** Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure. Sebastopol, California: O'Reilly Media, 2009. 394 p.

11. **Орлов В. А., Клещев А. С.** Компьютерные банки знаний. Многоцелевой банк знаний // Информационные технологии. 2006. № 2. С. 2—8.

12. **Голенков В. В., Гулякина Н. А.** Проект открытой семантической технологии компонентного проектирования интеллектуальных систем. Часть 1: Принципы создания // Онтология проектирования. 2014. № 1 (11). С. 42—64.

13. **Голенков В. В., Гулякина Н. А.** Проект открытой семантической технологии компонентного проектирования интеллектуальных систем. Часть 2: Унифицированные модели проектирования // Онтология проектирования. 2014. № 4 (14). С. 34—53.

14. **Грибова В. В., Клещев А. С., Крылов Д. А.** и др. Проект IACPaaS. Комплекс для интеллектуальных систем на основе облачных вычислений // Искусственный интеллект и принятие решений. 2011. № 1. С. 27—35.

15. **Gribova V. V., Kleschev A. S., Krylov D. A.** et al. A software platform for the development of intelligent multi-agent internet-services // Proceedings of the Distributed Intelligent Systems and Technologies Workshop (DIST'2013). 1—4 July 2013. St. Petersburg, Russia. 2013. P. 29—36.

16. **Handbook** of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations / Ed. by G. Rozenberg. World Scientific Publishing Co. Pte. Ltd., 1997. 572 p.

17. **Handbook** on Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications / Ed. by H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg. Languages and Tools. World Scientific, 1999. 720 p.

18. **Agrawal A., Vizhanyo A., Kalmar Z.** et al. Reusable Idioms and Patterns in Graph Transformation Languages // Electronic Notes in Theoretical Computer Science. 2005. N. 127 (1). P. 181—192.

19. **Klempien-Hinrichs R., Kreowski H.-J., Kuske S.** Rule-Based Transformation of Graphs and the Product Type // In Transformation of knowledge, information and data: Theory and Applications. Information Science Publishing. 2005. P. 29—51.

20. **Atkinson C., Kuhne T.** Model-driven development: A metamodeling foundation // IEEE Software. September, 2003. Vol. 20, Issue 5. P. 36—41.

21. **Kleppe A., Warmer S., Bast W.** MDA Explained. The Model Driven Architecture: Practice and Promise. Addison-Wesley, 2003. 192 p.

22. **Varro D., Pataricza A.** VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML // Journal of Software and Systems Modeling. 2003. Vol. 2 (3). P. 187—210.

23. **Muller P.-A., Fleurey F., Jezequel J.-M.** Weaving Executability into Object-Oriented Metalanguages // ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica. 2005. P. 264—278.

24. **MDA** web site [Электронный ресурс]. URL: http://www.omg.org/mda (дата обращения 18.02.2015).

V. V. Gribova, Research Deputy Director, e-mail: gribova@iacp.dvo.ru, **A. S. Kleschev,** Leading Researcher, e-mail: kleschev@iacp.dvo.ru, **D. A. Krylov,** Leading Software-Engineer, e-mail: dmalkr@gmail.com, **Ph. M. Moskalenko,** Researcher, e-mail: philipmm@iacp.dvo.ru, **V. A. Timchenko**, Researcher, e-mail: vadim@iacp.dvo.ru, **E. A. Shalfeyeva,** Senior Researcher, e-mail: shalf@iacp.dvo.ru, Institute for Automation and Control Processes, Far Eastern Branch of the Russian Academy of Sciences, Vladivostok

A Base Technology for Development of Intelligent Services with the Use of IACPaaS Cloud Platform. Part 1. A Development of Knowledge Base and Problem Solver

The paper opens a series of three works which describe a base technology for development of intelligent multi-agent cloud services and their components with the use of system tools of IACPaaS platform. The technology is put to reduce the labour-intensiveness of development and support for intelligent services and proposes involvement of domain experts (without mediators or additional training) in these processes, whose task is to create and maintain information components of platforms' applied services in actual state during their lifecycle. This first part presents a description of conceptual architecture of IACPaaS platform (system, fund, services, and management); a definition of intelligent service and a description of its assembly with the use of software and program components; a technology for development of an information resource (ontology or knowledge/data base); a definition of a problem solver and a technology for its assembly.

Keywords: *intelligent system, multi-agent system, intelligent software development technology, hierarchical semantic network, agent-oriented programming, cloud platform, cloud service*

References

1. **Gribova V. V., Kleschev A. S., Shalfeyeva E. A.** Upravlenie intellektual'nymi sistemami. *Izvestija RAN. Teorii i Sistemy Upravlenija*, 2010, no. 6, pp. 122—137 (in Russian).
2. **Gribova V. V., Kleschev A. S.** Tekhnologiya razrabotki intellektual'nykh servisov, orientirovannykh na deklarativnye predmetnye bazy znanii. Chast' 1. Informatsionnye resursy. *Informacionnye Tekhnologii*, 2013, no. 9, pp. 7—11 (in Russian).
3. **Gribova V. V., Kleschev A. S.** Tekhnologiya razrabotki intellektual'nykh servisov, orientirovannykh na deklarativnye predmetnye bazy znanii. Chast' 2. Reshatel' zadach. Pol'zovatel'skii interfeis. *Informacionnye Tekhnologii*, 2013, no. 10, pp. 10—14 (in Russian).
4. **Musen M.** Technology for building intelligent systems: from psychology to engineering, *Nebraska Symposium on Motivation* / Ed. by B. Shuart, W. Spaulding, and J. Poland. Lincoln, Nebraska, U Nebraska P, 2009, vol. 52, pp. 145—184.
5. **Sonar M. R**. An Enterprise Intelligent System Development and Solution Framework. *International Journal of Applied Science, Engineering and Technology*, 2007, vol. 4, no. 1, pp. 34—39.
6. **Schalkoff R. J.** *Intelligent Systems: Principles, Paradigms and Pragmatic*. Sudbury, Mass. Jones and Bartlett Publishers, 2011, 758 p.
7. **The Proteg** Ontology Editor and Knowledge Acquisition System, available at: http://protege.stanford.edu/
8. **Orlov V. A., Kleschev A. S.** Komp'yuternye banki znanii. Universal'nyi podkhod k resheniyu problemy redaktirovaniya informatsii. *Informacionnye Tehnologii*, 2006, no. 5, pp. 25—31 (in Russian).
9. **TopQuadrant,** available at: http://www.topquadrant.com/
10. **Sanderson D.** *Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure*. Sebastopol, California, O'Reilly Media, 2009, 394 p.
11. **Orlov V. A., Kleschev A. S.** Komp'yuternye banki znanii. Mnogotselevoi bank znanii. *Informacionnye Tehnologii*, 2006, no. 2, pp. 2—8 (in Russian).
12. **Golenkov V. V., Gulyakina N. A**. Proekt otkrytoi semanticheskoi tekhnologii komponentnogo proektirovaniya intellektual'nykh sistem. Chast' 1: Printsipy sozdaniya. *Ontologiya Proektirovaniya*, 2014, no. 1 (11), pp. 42—64 (in Russian).
13. **Golenkov V. V., Gulyakina N. A.** Proekt otkrytoi semanticheskoi tekhnologii komponentnogo proektirovaniya intellektual'nykh sistem. Chast' 2: Unifitsirovannye modeli proektirovaniya. *Ontologiya Proektirovaniya*, 2014, no. 4 (14), pp. 34—53 (in Russian).
14. **Gribova V. V., Kleschev A. S., Krylov D. A., Moskalenko Ph. M., Smagin S. V., Timchenko V. A., Tyutyunnik M. B., Shalfeyeva E. A.** Proekt IACPaaS. Kompleks dlya intellektual'nykh sistem na osnove oblachnykh vychislenii. *Iskusstvennyj Intellekt i Prinjatie Reshenij*, 2011, no. 1, pp. 27—35 (in Russian).
15. **Gribova V. V., Kleschev A. S., Krylov D. A., Moskalenko Ph. M., Timchenko V. A., Shalfeyeva E. A., Goldstein M. L.** A software platform for the development of intelligent multi-agent internet-services. *Proc. of the Distributed Intelligent Systems and Technologies Workshop (DIST'2013),* 2013, St. Petersburg, Russia. 2013, pp. 29—36.
16. ***Handbook*** *of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations* / Ed. by G. Rozenberg. World Scientific Publishing Co. Pte. Ltd., 1997, 572 p.
17. ***Handbook*** *of Graph Grammars and Computing by Graph Transformation, Vol. 2: Applications, Languages and Tools* / Ed. by H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg. World Scientific, 1999, 720 p.
18. **Agrawal A., Vizhanyo A., Kalmar Z., Shi F., Narayanan A., Karsai G.** Reusable Idioms and Patterns in Graph Transformation Languages, *Electronic Notes in Theoretical Computer Science*, 2005, no. 127 (1), pp. 181—192.
19. **Klempien-Hinrichs R., Kreowski H.-J., Kuske S.** Rule-Based Transformation of Graphs and the Product Type, *Transformation of knowledge, information and data: Theory and Applications.* Information Science Publishing, 2005, pp. 29—51.
20. **Atkinson C., Kuhne T.** Model-driven development: A metamodeling foundation, *IEEE Software*, September, 2003, vol. 20, issue 5, pp. 36—41.
21. **Kleppe A., Warmer S., Bast W.** *MDA Explained. The Model Driven Architecture: Practice and Promise,* Addison-Wesley, 2003, 192 p.
22. **Varro D., Pataricza A.** VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML, *Journal of Software and Systems Modeling,* 2003, vol. 2 (3), pp. 187—210.
23. **Muller P.-A., Fleurey F., Jezequel J.-M.** Weaving Executability into Object-Oriented Metalanguages, *ACM/IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica*, 2005, pp. 264—278.
24. **MDA** web site, available at: http://www.omg.org/mda

УДК 004.8

А. Е. Ермаков, канд. техн. наук, вед. специалист, e-mail: ermakov@rco.ru,
ЗАО "Крибрум", ООО "ЭР СИ О", г. Москва

Метаязык описания грамматики в синтаксических анализаторах естественного языка на основе объектно-ориентированного языка программирования

Раскрыты ключевые особенности синтаксического анализатора текста на естественном языке, более десяти лет развиваемого автором. Описание реализации синтаксического парсера публикуется впервые, однако основное внимание автор хотел акцентировать на новом разработанном метаязыке описания правил формальной грамматики. Назначение метаязыка — описывать правила декларативными средствами языка C++: логическими выражениями и специально реализованными процедурами, тогда как поддержка необходимой процедурной составляющей реализуется интерпретатором метаязыка в ходе синтаксического разбора текста.

Ключевые слова: *компьютерный анализ текста на естественном языке, синтаксический анализатор текста, метаязык описания формальной грамматики, дерево синтаксических составляющих, сеть синтактико-семантических отношений*

Введение

Начиная с 2001 г. автор занимается разработкой программных компонентов для коммерческих систем лингвистического анализа текста на естественном языке, ядром которых выступает синтаксическй анализатор. Зарубежные разработки, осуществляющие разбор русского языка, автору неизвестны, поэтому речь пойдет только о российских достижениях. Сегодня в России существуют три синтаксических анализатора текстов на русском языке, используемых в коммерческих продуктах: анализаторы компаний ЭР СИ О (*http*://www.rco.ru), Диктум [2] и ABBYY [3] (перечислены в порядке их появления на рынке). Две другие известные разработки созданы в академических коллективах: синтаксический анализатор ИППИ РАН для системы автоматического перевода ЭТАП-3 [4] и анализатор группы АОТ (*http://www.aot.ru/docs/synan.html*). В разработке анализаторов компании ЭР СИ О и Диктума автор принимал непосредственное участие, с работой трех других знаком по результатам общения с их разработчиками и по отзывам клиентов, проводивших независимое тестирование анализаторов для их последующего приобретения. Анализатор компании ЭР СИ О имеет производительность около 100 Мбайт текста в час, описанный в работе [2] анализатор компании Диктум способен обработать около 5 Мбайт текста в час, анализаторы ABBYY, ЭТАП-3 и АОТ имеют скорость работы на два порядка ниже, чем у анализатора компании ЭР СИ О.

Таким образом, статья посвящена описанию реализации самого быстрого анализатора русского текста, принципиально завершенного к 2005 г., но до сих пор нигде не описанного, а также достижению последнего года работы автора — метаязыку описания правил естественного языка над C++, введение которого позволило устранить ключевой недостаток парсера ЭР СИ О — сложность описания правил грамматики естественного языка, которые раньше писались непосредственно на языке C++.

Относительно качества синтаксического анализа текста стоит отметить следующее. Полнота и точность разбора определяются не столько подходом к построению парсера, скольку скрупулезностью описания правил грамматики (что определяет полноту) и системой правил выбора наилучшего варианта разбора при омонимии (что определяет точность).

В компании Диктум к 2011 г. был разработан синтаксический анализатор, точность работы которого была невысока — на соревнованиях синтаксических парсеров в рамках конференции Диалог-2012 [1] он занял шестое место, вследствие чего был доработан до состояния, описанного в работе [2], обеспечив значительно более высокое качество разбора, чем исходный анализатор, и по внутренним тестам разработчиков Диктума (независимое сравнение парсеров на конференциях Диалог больше не проводили) сравнялся с анализатором ЭТАП-3, занявшим второе место. Тем не менее дальнейшее повышение качества на практике обеспечить не удавалось вследствие

сложности задания ограничений на недопустимые варианты разбора с точки зрения сочетаемости составляющих в модели [5], в результате чего был осуществлен переход на описываемую в настоящей статье модель синтаксиса и синтаксический парсер, показавшие в компании ЭР СИ О более высокую точность — 95 % по результатам тестирования разработчиками (в соревнованиях парсеров на конференции Диалог-2012 по организационным причинам парсер ЭР СИ О не участвовал). Такая точность сравнима с точностью победителя соревнования по качеству — анализатора ABBYY, хотя достигнутая в ЭР СИ О полнота разбора (процент правильно устанавливаемых синтаксических отношений) несколько уступает полноте разбора других анализаторов и составляет около 90 %. Низкая полнота разбора связана с отсутствием в анализаторе правил для множества синтаксических конструкций, относительно редко встречающихся в русскоязычных текстах, вследствие указанного выше недостатка — трудоемкости описания правил непосредственно на языке C++. Именно внедрение предложенного в статье метаязыка должно позволить описать более полную систему правил и повысить качество работы наиболее быстрого из созданных в России анализаторов текста.

Модель синтаксиса

Окончательной целью синтаксического анализа является построение структуры связей между словами предложения, которая может быть представлена как сеть направленных и типизированных синтактико-семантических отношений [5]. При этом форма внутреннего представления синтаксической структуры фразы, с которой непосредственно работает синтаксический анализатор, может быть различной, аналогично формам синтаксической разметки предложения лингвистами. Описываемый анализатор использует в своем внутреннем представлении модель деревьев синтаксических составляющих [6, 7].

Составляющая соответствует цепочке следующих подряд слов предложения и знаков препинания, она имеет грамматические атрибуты, соответствующие своему вершинному слову — корню дерева составляющей. Исходные слова и знаки препинания являются терминальными составляющими, нетерминальная составляющая покрывает две или более дочерних составляющих (терминальных или нетерминальных), при этом определяя направленные связи между дочерними составляющими, типы и атрибуты связей, а также вершинную дочернюю составляющую, каковых может быть несколько, если составляющая покрывает однородные члены. Знание вершинной дочерней составляющей необходимо для установления синтактико-семантических связей между исходными словами, что осуществляется в ходе спуска по дереву составляющих, соответствующему наилучшему варианту разбора предложения. Так, если в составляющей — глагольной группе *VP = Verb*(*создать*) -> *NP(новые алгоритм и программу*) именная группа *NP* подчинена глаголу *Verb* связью типа "управление винительным без предлога", а в составляющей *NP = Adjective*(*новые*) <- *NP*(*алгоритм и программу*) вершинной дочерней составляющей является *NP = Noun*(*алгоритм*) <-> *Noun*(*программу*), которая, в свою очередь, имеет две вершинные составляющие *Noun*(*алгоритм*) и *Noun*(*программу*), то в ходе спуска по дереву и связывания вершинных составляющих будут установлены две связи типа "управление винительным без предлога" от *создать* к *алгоритм* и к *программа.*

Здесь и далее *NP* и *VP* обозначают нетерминальные составляющие, соответствующие именной и глагольной группам, а *Noun* и *Adjective* — терминальные составляющие, соответствующие отдельным словам — имени существительному и прилагательному соответственно.

Составляющая в общем случае имеет несколько наборов грамматических атрибутов, каждый из которых характеризует свой омоформ — то слово естественного языка (ЕЯ), грамматической формой которого может быть употребленная в тексте словоформа вершинного слова составляющей. Исключение — составляющие однородных членов предложения, грамматические атрибуты которых формируются аналитически на основе атрибутов однородных членов (так, грамматическое число у составляющей однородных *NP = NP* <-> *NP* или *VP = VP* <-> *VP* всегда будет множественным). Если составляющая нетерминальная, то ей соответствует набор тех омоформов вершинного слова составляющей, которые удовлетворили условиям правил грамматики, задействованных при связывании всех дочерних составляющих в структуру данной, — в ходе подъема вверх по дереву составляющих омонимия уменьшается.

Помимо грамматических атрибутов, у каждого омоформа составляющая имеет общие структурные атрибуты, которые описывают структуру дерева ее дочерних составляющих. Например, *NP*-составляющие (именные группы) могут иметь следующие структурные атрибуты: `CONST _ ADJ _ RIGHT` — прилагательное справа, `CONST _ GEN _ RIGHT` — несогласованнное определение-генитив справа, `CONST _ PARTICIPLE _ RIGHT` — причастный оборот справа и др. Так, если составляющая s_t имеет грамматической атрибут части речи `POS _ NOUN`, а следующая за ней составляющая s_{t+1} имеет грамматические атрибуты `POS _ NOUN` и `CASE _ GEN` (родительный падеж — генитив), то они не могут входить в составляющую типа *NP = NP* -> -> *NP*gen (именная группа с вершинным существительным в генитиве), если составляющая s_t имеет хотя бы один из трех вышеописанных структурных атрибутов — соответствующее ограничение `CONST _ ADJ _ RIGHT | CONST _ GEN _ RIGHT | CONST _ PARTICIPLE _ RIGHT` записывается в правиле.

Для *VP*-составляющих (глагольных групп) важнейшими являются структурные атрибуты `CONST _ CONTROL _ SUBJECT`, `CONST _ CONTROL _ GEN`, `CONST _ CONTROL _ DAT`, `CONST _ CONTROL _ ACC`, `CONST _ CONTROL _ INSTR`, которые означают, что у вершинного глагола заполнена валентность именительного, родительного, дательного, винительного

и творительного падежей без предлога. Знание того, какие валентности *VP*-составляющей уже заполнены другими *NP*-составляющими, позволяет включить ограничения, не допускающие возникновения составляющих *VP = VP -> NP* с многократным заполнением одной валентности.

Синтаксический парсер на C++

Синтаксический парсер обеспечивает применение правил формальной грамматики к последовательности составляющих предложения и построение всех возможных разборов — деревьев составляющих — с выбором наилучшего, без повторного построения возникающих в ходе парсинга поддеревьев. Базовый принцип работы описываемого парсера был почерпнут из работы [8].

На вход парсера поступает последовательность указателей на структуры `SConstituent`, описывающие терминальные составляющие предложения:

```
struct SConstituent {
   long long lConstAttrs; /*структурные атрибуты – 64-битовая маска*/
   vector<SGrammarForm> vGrammar; /*грамматические атрибуты омоформов*/
   vector<SConstituent*> vpChild; /*указатели на дочерние составляющие*/
   vector<SRelationInfo> vRelInfo; /*связи дочерних составляющих*/
};
```

Структура `SGrammarForm` описывает отдельный омоформ в составе поля:

```
vector<SGrammarForm> vGrammar:
struct SGrammarForm {
   long long lGrammar; /*грамматические атрибуты – 64-битовая маска*/
   SGrammarForm* pModifier; /*аналитический модификатор омоформа*/
   SGrammarForm* ppChildG[MAX _ RULE _ LENGTH]; /*омоформы дочерних составляющих*/
   };
```

Здесь `SGrammarForm* pModifier` указывает на другой омоформ — аналитический модификатор грамматической формы данного омоформа (предлог для *PP*, частица или служебный глагол для *VP*), значение которого устанавливается в результате работы определенных правил, например *PP = Preposition <- <- NP*, *VerbAux <- Verb*, где *PP* — именная группа с предлогом, *Preposition* — предлог, *VerbAux* — служебный глагол. В массиве `SGrammarForm* ppChildG[MAX _ RULE _ LENGTH]` сохраняются указатели на омоформы дочерних составляющих, удовлетворившие условиям сформировавшего ее правила, чтобы в финале снять омонимию, спускаясь по дереву разбора и оставляя в дочерних составляющих только те омоформы, которые задействовались в сработавших правилах.

Структура `SRelationInfo` описывает синтактико-семантическую связь отдельной парой дочерних составляющих в составе поля `vector<SRelationInfo> vRelInfo`:

```
struct SRelationInfo {
   SConstituent *pSrc, *pDst; /*указатели на связанные составляющие*/
   int nRelationType; /*тип связи*/
   long long lCase; /*семантический падеж связи*/
   SGrammarForm* pModifier; /*аналогично pModifier в составе SGrammarForm*/
};
```

В составе одной составляющей могут быть установлены связи между несколькими парами дочерних, например, для разбора непроективных конструкций вида "*мозги народу пытаются морочить либералы*" необходимо одно сложное правило бесконтекстной грамматики, формирующее одну составляющую с четырьмя парами связей от вершинной дочерней составляющей *пытаются*.

Результатом разбора является дерево составляющих `SConstituent`, в котором каждый узел хранит указатели `vector<SConstituent*> vpChild` на дочерние узлы.

Все правила грамматики парсера являются C++-классами — наследниками базового класса `CSyntRule` — и реализуют его виртуальные методы `Init()`, `_ S()`, `G1()`, ... `G5()`, `Concord()`, `Result()`, `Postproc()`. Содержательные особенности реализации этих методов подробно рассмотрены далее. Парсер обращается

к каждому из правил, вызывая его методы через указатель на базовый класс `CSyntRule*`, ничего не зная о правиле, кроме длины цепочки обрабатываемых им составляющих `m_nRuleLen`.

Базовый синтаксический парсер работает по следующей схеме.

Двигаясь справа налево по цепочке составляющих, парсер на каждом шаге применяет к каждому фрагменту цепочки каждое правило-реализацию базового класса `CSyntRule`, вызывая его метод `int CSyntRule::Execute(const SConstituent **ppS, SConstituent &NewConstituent)`, с учетом известной для каждого правила длины `m_nRuleLen` покрываемой им цепочки составляющих, не давая правилу выйти за границы предложения. Каждому правилу доступен только переданный ему фрагмент, на который указывают `m_nRuleLen` указателей от `ppS` до `ppS + m_nRuleLen-1` — формальная грамматика является контекстно-свободной. Правило срабатывает, если все его методы `_S()`, `G1()`, ..., `G5()` и `Concord()` возвращают `true` — тогда `CSyntRule::Execute` вызывает метод `CSyntRule::Result()`, который создает единственную результирующую составляющую `NewConstituent`. В новой составляющей `NewConstituent` сработавшим правилом будут заполнены структуры с ее атрибутами, указатели на дочерние составляющие `vpChild`, указатели на все удовлетворившие правилу комбинации омоформов `ppChildG` для последующего снятия омонимии.

Будем называть шагом разбора S_t^r применение правила с номером r ($r = 1...R$, где R — число правил) к фрагменту последовательности составляющих F_t^r с позиции t ($t = 1...T$) до позиции `t + m_nRuleLen-1` правила r. Парсинг начинается с $t = T$, где T — длина последовательности составляющих верхнего уровня, указатели на которые хранятся в массиве `SConstituent **ppS`.

Если на очередном шаге S_t^r виртуальный метод `CSyntRule::Execute` правила r вернул 0, то парсер пытается применить правило $r + 1$ к позиции t. Если все правила вплоть до R для позиции t вернули 0, то парсер пытается применить правило $r = 1$ к позиции $t - 1$ и т. д., пока не сработает какое-то правило — `CSyntRule::Execute` вернет 1.

Если на очередном шаге S_t^r `CSyntRule::Execute` вернул 1, то цепочка указателей F_t^r заменяется указателем на новую покрывающую составляющую `NewConstituent` — цепочка дочерних составляющих сворачивается. Весь набор правил применятся сначала с той же позиции t (теперь этот фрагмент содержит новую составляющую `NewConstituent`) — проверяется правило $r = 1$ и т. д.

Информация о проведенных заменах составляющих сохраняется в так называемом протоколе разбора — логическом стеке, каждый элемент которого хранит указатель на указатель на новую вставленную составляющую `NewConstituent` в массиве `SConstituent **ppS`, номер составляющей в последовательности t и номер правила r, ее сформировавшего.

На последнем шаге S_1^R парсер завершает построение очередного варианта разбора, после чего проводится:

- сравнение текущего разбора и лучшего из ранее построенных по набору критериев, если текущий разбор оказывается лучше, то полученное дерево составляющих сохраняется взамен старого;
- откат состояния разбора — из протокола выбирается указатель на последнюю построенную покрывающую составляющую с соответствующими ей t и r, на основании чего восстанавливается предыдущее состояние разбора — на свое место в массиве `SConstituent **ppS` вставляются сохраненные в покрывающей составляющей указатели на дочерние составляющие, сама покрывающая и указатель на нее удаляются, а парсер переходит к шагу S_t^{r+1}, если $r < R$, или S_{t-1}^1, если $r = R$ и $t > 1$.

Если стек протокола пуст и выполнен шаг S_1^R, разбор заканчивается.

Описанный парсер иллюстрирует базовый принцип построения всех деревьев разбора, допустимых в рамках используемого множества правил грамматики. Однако он не является оптимальным с точки зрения производительности, поскольку не учитывает того факта, что одно и то же дерево составляющих может быть собрано из своих поддеревьев в разном порядке — в описанной логике уникальность составляющей определяется только порядком ее сборки, и все одинаковые по структуре составляющие будут далее обрабатываться правилами как разные, порождая формально новые, а фактически эквивалентные деревья разбора. Поэтому на практике используется более сложная реализация парсера, которая хранит информацию о ранее обработанных соседях составляющих, уже возникавших в ходе разбора, во избежание повторного парсинга.

Метаязык описания правил грамматики на C++

Настройка правил формальной грамматики является наиболее ресурсоемкой задачей при создании синтаксического анализатора ЕЯ, и основными ее исполнителями являются лингвисты, а не программисты. Потому наиболее практически приемлемой кажется такая реализация, в которой правила описываются не на языке программирования (ЯП), а на неком формальном декларативном языке, как, например, реализация, выполненная в компании "Диктум" [1]. Однако подобные реализации имеют два описанных ниже недостатка.

- Чтобы использовать правила при разборе, необходим интерпретатор этих правил, в простейшем случае — непосредственно из их исходного текста, а в наилучшем — из промежуточного бинарного представления, сформированного специальной программой — компилятором с исходного языка, и хранящего данные о правилах в специальных структурах для быстрого обращения к ним в ходе разбора. Так или иначе, производительность оказывается более низкой, чем при написании правил непосредственно на том ЯП, на котором написан синтаксический парсер.

• Среди порождаемых конструкций ЕЯ находятся исключения, которые не удается описать средствами декларативного языка формальной грамматики. В то же время возможности разумного расширения выразительных средств декларативного языка не безграничны и многие ситуации оказывается проще обрабатывать процедурно средствами ЯП, не расширяя всякий раз формализм декларативного языка.

Правильный подход к написанию правил на универсальном ЯП позволяет избавиться от обоих недостатков. Язык программирования поддерживает все логические операторы и позволяет описывать любые логические выражения для проверки на возможную сочетаемость грамматических и структурных атрибутов составляющих. Чтобы избежать "человечески нечитаемых" описаний грамматики вследствие включения в правила процедурной логики (условные переходы, циклы, присваивания переменных) и допустить к написанию правил лингвиста, не знакомого с программированием, необходимо определить стандарт написания правил, т. е. декларативный метаязык над ЯП, и разработать его интерпретатор, который априорно будет быстрее, чем любой интерпретатор внешнего к ЯП декларативного языка. При этом сохраняется возможность использовать все средства процедурного программирования для написания нестандартных правил в исключительных случаях.

Разработанный метаязык построен над C++, его основные элементы перечислены далее.

• `s1`, `s2`, `s3`, `s4`, `s5` — порядковые обозначения пяти составляющих, к которым могут применяться операции в правилах. Для русского языка самое длинное правило, задействующее пять составляющих, необходимо для покрытия непроективной конструкции вида "*мозги народу пытаются морочить либералы*", поскольку описание соответствующей непроективной конфигурации синтаксических связей, устанавливаемых от четвертого слова к первому и второму, а также от пятого к третьему, должно проводиться в рамках единого правила. Теоретически в языке подобная конструкция может содержать и более длинную цепочку инфинитных глаголов, например "*мозги народу пытаются начать морочить либералы*", что соответствует правилу для шести составляющих, однако практически кажется разумным ограничиться пятью составляющими.

• `CONST(s,ValueMask)` — предикат сравнения структурных атрибутов составляющей `s` со значением выражения `ValueMask`, например: `CONST(s2, CONST _ ADJ _ RIGHT|CONST _ PP _ RIGHT)` имеет значение `true`, если у составляющей `s2` есть атрибут "включает прилагательное справа" или "включает PP справа".

• `GRAM(s,ValueMask)` — предикат сравнения грамматических атрибутов составляющей `s` со значением выражения `ValueMask`, например: `GRAM(s2,POS _ NOUN|POS _ PRONOUN)` имеет значение `true`, если у составляющей `s2` проверяемый правилом омоформ — это существительное или местоимение. Специальные предикаты `gPLURAL(s)`, `gSINGULAR(s)` принимают значение `true` в зависимости от грамматического числа омоформа.

• `gCASE(s1,s2)`, `gGENDER(s1,s2)`, `gPERSON(s1,s2)` — предикаты проверки согласования, которые возвращают `true` в случае совпадения значений грамматических атрибутов у сопоставляемых правилом омоформов из двух заданных составляющих `s1` и `s2`. Имя предиката определяет проверяемый на согласование атрибут, здесь указаны предикаты для проверки падежа, рода и лица.

• `VALENCE(s1,s2)` — предикат проверки управления, имеет значение `true`, если для пары сопоставляемых правилом омоформов из `s1` и `s2` второй омоформ удовлетворяет модели управления первого, т. е. имеет допустимый падеж и предлог.

• `SetConstituent(ValueMask)`, `SetGrammar (ValueMask)` — установка наборов значений `ValueMask` структурных и грамматических атрибутов составляющей, покрывающей цепочку дочерних составляющих, удовлетворяющих правилу. `SetGrammar(s)` — установка всех грамматических атрибутов покрывающей составляющей из обрабатываемого правилом омоформа дочерней составляющей `s`.

• `SetRelation(s1,s2,RelationType,Case,Modifier)` — установка направленной связи типа `RelationType` с семантическим падежом `Case` и модификатором `Modifier` (предлогом, союзом) между вершинными дочерними составляющими в составляющих `s1` и `s2`. Падеж и модификатор задаются только для связей определенных типов, обычно устанавливаемых на основе моделей управления предикатов [1].

Рассмотрим теперь структуру правила грамматики, которое представляет собой C++-класс, наследованный от базового `CSyntRule` и реализующий его виртуальные методы с использованием вышеописанных элементов метаязыка.

```
class CSR _ Adjective _ NP: public CSyntRule {
    virtual void Init(){ _ C(0, 1); _ R(s2, s1, REL _ ATTRIBUTE); }
    virtual bool _ S() { return!CONST(s2,CONST _ ADJ _ RIGHT|CONST _ PP _ RIGHT); }
    virtual bool G1() { return GRAM(s1, POS _ ADJ); }
    virtual bool G2() { return GRAM(s2, POS _ NOUN|POS _ PRONOUN); }
    virtual bool Concord() { return gCASE(s1,s2) && ((gPLURAL(s1) && gPLURAL(s2))
 || (gGENDER(s1, s2) && gSINGULAR(s1) && gSINGULAR(s2)));
   }
    virtual void Result() {
      SetConstituent(GetConstituent(s2) | CONST _ NP|CONST _ ADJ _ LEFT);
      SetGrammar(s2);
   }
};
```

Данное правило имеет имя CSR _ Adjective _ NP и обрабатывает составляющую со структурой *NP = Adjective <-NP* — прилагательное, подчиненное существительному в вершине другой *NP*.

Метод Init() задает длину цепочки дочерних составляющих и помечает вершинную составляющую в цепочке, а также определяет отношения между терминальными составляющими предложения после завершения его синтаксического разбора на основе полученного дерева составляющих. Здесь _ C(0,1) указывает, что правило обрабатывает две составляющие (*Adjective* и *NP*), из которых вторая является вершинной; а _ R(s2,s1,REL _ ATTRIBUTE) предписывает установить связь типа REL _ ATTRIBUTE от каждой вершинной составляющей из s2 (каковых может быть несколько, если s2 — группа однородных *NP*) к каждой вершинной составляющей из группы s1 (каковых может быть несколько, если s1 — группа однородных *Adjective*).

Метод _ S() вызывается интерпретатором метаязыка для проверки ограничения на структурные атрибуты составляющих, обрабатываемых правилом. Здесь !CONST(s2, CONST _ ADJ _ RIGHT|CONST _ PP _ RIGHT) означает, что для s2 (*NP*) недопустимо иметь прилагательное справа от существительного — по стилистике, если таковое уже имеется справа, то ненормально подчинять второе прилагательное слева. CONST _ PP _ RIGHT — запрет на включение *PP* в состав *NP*, связанный с ограничениями на допустимый порядок сборки составляющих при парсинге во избежание сборки одной и той же составляющей разными путями: *NP = (Adjective <- NP) -> PP* и *NP = Adjective <- (NP -> PP)*, ограничение CONST _ PP _ RIGHT запрещает второй порядок сборки.

Методы G1()-G5() вызываются интерпретатором метаязыка для независимой проверки ограничений на грамматические атрибуты омоформов составляющих s1-s5 без проверки их согласования. Эти проверки также могут быть выполнены непосредственно в методе Concord() (тогда не требуется имплементация G1()-G5()), но тогда, как можно далее увидеть из реализации интерпретатора, скорость его работы снизится.

Метод Concord() вызывается для полной проверки ограничений на грамматические атрибуты с согласованием их значений между омоформами разных составляющих. Проверка ограничений проводится интерпретатором для всех комбинаций омоформов из $K \leqslant 5$ составляющих, обрабатываемых правилом, в ходе работы K вложенных циклов (см. далее реализацию CSyntRule::Execute()). Вынос проверок независимых ограничений в методы G1()-G5() позволяет во многих случаях избежать прохода во вложенные циклы.

Метод Result() вызывается интерпретатором для создания покрывающей составляющей для цепочки дочерних, удовлетворившей ограничениям, описанным в методах _ S(), G1()-G5(), Concord(). Метод Result() вызывается для каждой комбинации омоформов цепочки, удовлетворяющей ограничениям, и всякий раз в покрывающую составляющую добавляется новый омоформ с атрибутами, устанавливаемыми посредством SetGrammar. SetConstituent(GetConstituent (s2) | CONST _ NP|CONST _ ADJ _ LEFT) задает структурные атрибуты покрывающей составляющей как копию атрибутов составляющей s2, плюс атрибуты CONST _ NP (*NP*-составляющая) и CONST _ ADJ _ LEFT (содержит прилагательное слева), а SetGrammar(s2) устанавливает грамматические атрибуты омоформа покрывающей составляющей от омоформа s2, удовлетворившего всем ограничениям правила.

На листинге 1 приведен пример правила CSR _ Homogenous для составляющей из однородных членов: именных групп *NP*, прилагательных или наречий, перечисленных через "," или сочинительный союз POS _ CONJ _ COORD, что проверяется для составляющей s2 в G2(). _ C(1,0,1) показывает, что вершинными являются две равноправные составляющие s1 и s3. Проверка !CONST(s1,CONST _ HOMOGENOUS) позволяет сборку составляющих с более чем двумя однородными членами только в одном направлении "справа налево". В методе Concord() gPOS(s1, s3) проверяет совпадение частей речи омоформов составляющих s1 и s3, а затем, если часть речи омоформов прилагательное или существительное, проводится проверка согласования падежей — gCASE(s1, s3). В методе Result() покрывающая составляющая получает структурные атрибуты обеих дочерних s1 и s3, плюс атрибут CONST _ HOMOGENOUS. Омоформ покрывающей составляющей получает грамматические атрибуты омоформа s1, плюс атрибут множественного числа NUMBER _ PLURAL. В методе Init() предписывается установить связь типа REL _ HOMOGENOUS между дочерними вершинными составляющими составляющих s1 и s3 симметрично в обе стороны.

```
class CSR _ Homogenous: public CSyntRule {
   virtual void Init() { _ C(1, 0, 1);
     _ R(s3, s1, REL _ HOMOGENOUS); _ R(s1, s3, REL _ HOMOGENOUS);
   }
   virtual bool _ S() { return!CONST(s1, CONST _ HOMOGENOUS); }
   virtual bool G2() { return GRAM(s2, POS _ CONJ _ COORD) || TEXT(s2, L","); }
   virtual bool Concord() {
    return gPOS(s1, s3) && (GRAM(s1, POS _ ADVERB) || ((GRAM(s1, POS _ ADJ)
 || GRAM(s1, POS _ NOUN)) && gCASE(s1, s3)));
   }
   virtual void Result() {
     SetConstituent(GetConstituent(s1) | GetConstituent(s3) | CONST _ HOMOGENOUS);
     SetGrammar(s1); SetGrammar(NUMBER _ PLURAL);
   }
};
```

Листинг 1

На листинге 2 дан пример особого базового правила `CSR _ VerbControl` для составляющей вида $VP = VP \rightarrow \overline{NP}$, в которой вершина VP — глагол или предикатив — подчиняет себе именную группу через управление ее падежом. Правило параметризовано значением обрабатываемого падежа и реализует базовый C++-класс для реализации правил под конкретные падежи через механизм наследования, для чего имеет конструктор с параметрами `CSR _ VerbControl(long long Case, long long Constituent)`, которые задают обрабатываемый падеж и структурные атрибуты покрывающей составляющей. Предикат `VALENCE(s1,s2)` проверяет возможность падежного управления омоформа из `s1` омоформом из `s2`.

```
class CSR _ VerbControl: public CSyntRule {
   long long lCase, lConstituent;
protected:
    CSR _ VerbControl(long long Case, long long Constituent): CSyntRule(),
 lCase(Case), lConstituent(Constituent) {}

   virtual void Init() { _ C(1, 0); _ Relation(s1, s2, REL _ CONTROL, gCASE(s2)); }
   virtual bool _ S() {
    return!CONST(s1, CONST _ S _ SUBJECT | lConstituent) && !CONST(s2, CONST _ PP);
   }
   virtual bool G1() { return GRAM(s1, POS _ VERB|POS _ PREDICATIVE); }
   virtual bool G2() { return GRAM(s2, POS _ COM _ NOUN) && GRAM(s2, lCase); }
   virtual bool Concord() { return VALENCE(s1, s2); }
   virtual void Result() {
    SetConstituent(GetConstituent(s1) | lConstituent | CONST _ VP|CONST _ NP _ RIGHT);
    SetGrammar(s1);
   }
};
```

Листинг 2

Реальные правила для составляющих $VP = VP \rightarrow \rightarrow \overline{NP}$ с управлением конкретными падежами тривиально реализуются через наследование от правила `CSR _ VerbControl`, как в примере ниже для дательного падежа.

```
class CSR _ VerbControl _ Dat: public CSR _ VerbControl {
public:
CSR _ VerbControl _ Dat(): CSR _ VerbControl(CASE _ DAT, CONST _ CONTROL _ DAT){}
};
```

Интерпретатор метаязыка

Интерпретатор метаязыка реализуется в классе `CSyntRule`, ключевой программный код которого представлен на листинге 3.

```
class CSyntRule {
protected:
   SConstProcInfo s1, s2, s3, s4, s5; /*данные об обработке составляющих*/
   SConstituent* pS; /*указатель на результирующую покрывающую составляющую*/
   SGrammarForm* pG; /*атрибуты очередного омоформа покрывающей составляющей*/
   /*маска признаков дочерних составляющих: 1 — составляющая вершинная*/
   unsigned m _ vnConstMask[MAX _ RULE _ LENGTH];

public:
   int m _ nRuleLen; /*число составляющих, покрываемых правилом*/

   int Execute(const SConstituent** ppS, SConstituent &NewConstituent)
{
   switch(m _ nRuleLen) {
   case 2: s1.pS=ppS[0], s2.pS=ppS[1], s3.pS=NULL, s4.pS=NULL;
   case 3: s1.pS=ppS[0], s2.pS=ppS[1], s3.pS=ppS[2], s4.pS=NULL;
   ...
   };
```

```
   if(! _ S()) /*условия на структурные атрибуты составляющих не выполнены*/
     return 0;
   pS=&NewConstituent;
   bool bRuleFired = false;
   for(s1.iG=s1.pS->vGrammar.begin(); s1.iG!=s1.pS->vGrammar.end(); ++s1.iG)
   if(G1()) /*условие на атрибуты очередного омоформа s1.iG*/
     for(s2.iG = s2.pS->vGrammar.begin(); s2.iG!=s2.pS->vGrammar.end(); ++s2.iG)
     if(G2()) /*условие на атрибуты очередного омоформа s2.iG*/
       if(m _ nRuleLen == 2)
       {
         if(FireRule()) bRuleFired = true;
       }
       else
       for(s3.iG = s3.pS->vGrammar.begin();s3.iG!=s3.pS->vGrammar.end(); ++s3.iG)
       if(G3()) /*условие на атрибуты очередного омоформа s3.iG*/
         if(m _ nRuleLen == 3)
         {
           if(FireRule()) bRuleFired = true;
         }
         else
         ... /*аналогично циклы проходят по омоформам s4 и s5*/
           if(G5())
           {
             if(FireRule()) bRuleFired = true;
           }
   return bRuleFired;
   }

   bool FireRule()
   {
   if(!Concord()) /*условия на согласование омоформов не выполнены*/
         return false;
     pS->m _ vGF.push _ back(SGrammarForm());
     pG=&(pS->m _ vGF.back());
     pS->vpChild[0]=s1.pS, pG->ppChildG[0]=s1.iG.operator->();
     if(m _ nRuleLen>1) pS->vpChild[1]=s2.pS, pG->ppChildG[1]=s2.iG.operator->();
     if(m _ nRuleLen>2) pS->vpChild[2]=s3.pS, pG->ppChildG[2]=s3.iG.operator->();
     ...
     Result(); /*установить атрибуты покрывающей составляющей*/
     return true;
   }
protected:
   /*указание покрываемых правилом составляющих и маркировка вершинных*/
   void _ C(int S1, int S2){ m _ nRuleLen=2; m _ vnConstMask[0]=S1,
m _ vnConstMask[1]=S2; }
   void _ C(int S1, int S2, int S3){ m _ nRuleLen=3;... }
   void _ C(int S1, int S2, int S3, int S4){ m _ nRuleLen=4;... }
   void _ C(int S1, int S2, int S3, int S4, int S5){ m _ nRuleLen=5;... }

   /*виртуальные методы для имплементации в правилах - наследниках CSyntRule*/
   virtual void Init()=0;
   virtual bool _ S() { return true; }
   virtual bool G1() { return true; }
   ...
   virtual bool G5() { return true; }
   virtual bool Concord() { return true; }
   virtual void Result()=0;
   virtual void Postproc()=0;

   /*предикаты и прочие элементы метаязыка*/
   bool CONST(const SConstProcInfo &s, long long lConstAttrs) {
     return s.pS->lConstAttrs & lConstAttrs);
   }
   bool GRAM(const SConstProcInfo &s, long long lGramAttrs) {
     return s.pG->lGrammar & lGramAttrs;
   }
   ...
};
```

Листинг 3

Структура SConstProcInfo хранит информацию об обработке составляющей в ходе выполнения правила:

```
struct SConstProcInfo {
   SConstituent* pS; /*указатель на составляющую*/
   vector<SGrammarForm>::iterator iG; /*указатель на обрабатываемый омоформ*/
};
```

Через объекты s1-s5 типа SConstProcInfo, указываемые в качестве аргументов предикатов и специальных методов метаязыка в правилах грамматики, в интерпретаторе осуществляется адресация к атрибутам составляющих и их омоформов (см. реализацию предикатов CONST и GRAM в теле класса CSyntRule).

Работа интерпретатора выполняется в методе CSyntRule::Execute(const SConstituent** ppS, SConstituent &NewConstituent), который является общим методом всех классов-наследников, реализующих конкретные правила грамматики, и вызывается синтаксическим парсером для проверки условий на m _ nRuleLen составляющих предложения, начиная с той, на которую указывает аргумент ppS. В случае удовлетворительной проверки всех условий правила CSyntRule::Execute возвращает парсеру 1, а в NewConstituent формируется покрывающая составляющая для обработанной цепочки дочерних.

Для проверки ограничений правила на всех комбинациях омоформов из m _ nRuleLen = K обрабатываемых составляющих s1-sK указатели iG пробегают все омоформы в ходе K вложенных циклов, и для каждого омоформа iG выполняется проверка условий G1()-GK(), а для каждой комбинации этих омоформов, успешно прошедших проверки условий G1()-GK(), выполняется проверка условий на согласование их грамматических атрибутов Concord(). Такой подход позволяет скрыть за синтаксисом метаязыка всю процедурную составляющую работы правил грамматики: циклы по омоформам, условные переходы при обработке ограничений на атрибуты составляющих, операции с указателями C++.

Метод CSyntRule::FireRule() вызывается из метода CSyntRule::Execute для каждой текущей комбинации омоформов, прошедших проверку независимых ограничений на грамматические атрибуты G1()-G5(), и выполняет проверку условий согласования атрибутов вызовом виртуального метода Concord(), реализованного в правиле-наследнике. Если все условия соблюдены, то далее над покрывающей составляющей по указателю pS осуществляются следующие действия:

- в массиве pS->vpChild сохраняются указатели на дочерние составляющие;
- в массив pS->vGrammar добавляется очередной "пустой" омоформ для последующей простановки его атрибутов;
- вызывается виртуальный метод Result(), реализованный в правиле-наследнике, который проставляет структурные атрибуты покрывающей составляющей и грамматические атрибуты ее очередного омоформа;
- в массиве pG->ppChildG сохраняются указатели на текущие омоформы дочерних составляющих как подтвердившиеся правилом грамматики для последующего снятия омонимии на основе дерева наилучшего разбора.

Заключение

Описанный метаязык описания грамматики ЕЯ на C++, его интерпретатор и синтаксический парсер послужили основой для разработки новой версии лингвистического анализатора русского текста в компании "Диктум" (*http://www.dictum.ru*). Лингвистический анализатор используется в электронных сервисах мониторинга информации в социальных сетях, предоставляемых системой "Крибрум" (*http://www.kribrum.ru*), для выявления позитивных/негативных оценок объектов мониторинга, сбора фактов по объектам мониторинга и авторам сообщений, а также в компании "ЭР СИ О" для решения аналогичных задач.

Список литературы

1. **Толдова С. Ю., Соколова Е. Г., Астафьева И.** и др. Оценка методов автоматического анализа текста 2011—2012: синтаксические парсеры русского языка // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Междунар. конф. "Диалог". Бекасово. 30 мая — 3 июня 2012 г. Вып. 11 (18): Т. 2: Доклады специальных секций. М.: Изд-во РГГУ, 2012. С. 77—90.
2. **Skatov D. S., Liverko S. V., Okatiev V. V., Strebkov D. Y.** Parsing Russian: a Hybrid Approach, Association for Computational Linguistics (ACL)// Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing. Sofia: Association for Computational Linguistics, 2013. P. 34—42.
3. **Anisimovich K. V., Druzhkin K. Ju., Minlos F. R.** et al. Syntactic and semantic parser based on ABBYY Compreno linguistic technologies // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Междунар. конф. "Диалог". Бекасово, 30 мая — 3 июня 2012 г. Вып. 11 (18), Т. 2: Доклады специальных секций. М.: Изд-во РГГУ, 2012. С. 91—103.
4. **Iomdin L., Petrochenkov V., Sizov V., Tsinman L.** ETAP parser: state of the art // Компьютерная лингвистика и интеллектуальные технологии: По материалам ежегодной Междунар. конф. "Диалог". Бекасово, 30 мая—3 июня 2012 г. Вып. 11 (18), Т. 2: Доклады специальных секций. М.: Изд-во РГГУ, 2012. С. 119—131.
5. **Ермаков А. Е.** Эксплицирование элементов смысла текста средствами синтаксического анализа-синтеза // Компьютерная лингвистика и интеллектуальные технологии: труды Междунар. конф. "Диалог'2003". М.: Наука, 2003. С. 136—140.
6. **Гладкий А. В.** Синтаксические структуры естественного языка в автоматизированных системах общения. М.: Наука, 1985. 144 с.
7. **Тестелец Я. Г.** Введение в общий синтаксис. М.: Изд-во РГГУ, 2001. 798 с.
8. **Смирнов Ю. М., Андреев А. М., Березкин Д. В., Брик А. В**. Об одном способе построения синтаксического анализатора текстов на естественном языке // Известия вузов. Приборостроение, 1997. Т. 40, № 5. С. 34—42.

A. E. Ermakov, Senior Specialist, e-mail: ermakov@rco.ru, Kribrum ltd, RCO Llc, Moscow

Grammar Description Meta-Language for Natural Language Syntactic Parsers Based on Object-Oriented Programming Language

The article describes key features of the natural language text syntactic parser, which is the fasters parser among syntactic parsers developed in Russia. Linguistic analyzers of Russian and English texts based on the described parser are used in electronic social network monitoring services for machine-based detection of positive and negative appraisals of target objects and extraction of facts on target objects and authors of messages.

The parser is based on the model of the syntactic structure of sentences in the form of syntactic constituent trees. The parser provides the application of formal grammar rules to the sequence of sentence's constituents and constructs all possible parsing variants — trees of constituents — with the choice of the best one, without repeating construction of occurring sub-trees. To ensure maximum speed performance of parsing, both the parsing algorithm and the natural language grammar rule description are written in C++.

To avoid grammar descriptions which are 'unreadable for a human' because of including procedural logic into the rules (conditional branching, cycles, access to members of a class, variable assignment) and to make it possible for a linguist unfamiliar with the programming to write new rules, a declarative meta-language, which works above the C++ language, and its interpreter have been developed. The described meta-language allows creating grammar rules by means of C++ declarative tools: Boolean expressions and specifically implemented procedures; while the necessary procedural component is implemented by the meta-language interpreter in the course of text parsing. To write non-standard grammar rules in exceptional cases there is a possibility to use all means of procedural programming language C++.

The description of the syntactic parser is given at the level of description of the basic algorithm of its work. The description of the meta-language is detailed, with examples of the Russian grammar rules implementation and software code of interpreter core in C++.

Keywords: *computational natural language processing, syntactic parser, meta-language for formal grammar description, syntactic constituent tree, tree of syntactic and semantic relations*

References

1. **Toldova S. Yu., Sokolova E. G., Astaf'eva I., Gareishina A., Koroleva A. N., Privoznov D., Sidorova E., Tupikina L., Lyashevskaya O. N.** Ocenka metodov avtomaticheskogo analiza teksta 2011—2012: sintaksicheskie parsery russkogo jazyka. *Komp'juternaja lingvistika i intellektual'nye tehnologii: po materialam ezhegodnoj Mezhdunar. konf. "Dialog".* [*Computer linguistics and intelligent technologies: proc. of annual int. conf. "Dialog"*]. Bekasovo. 30 May—3 June 2012, issue. 11 (18), vol. 2: Doklady special'nyh sekcij, Moscow, Izd-vo RGGU, 2012, pp. 77—90 (in Russian).

2. **Skatov D. S., Liverko S. V., Okatiev V. V., Strebkov D. Y.** Parsing Russian: a Hybrid Approach, Association for Computational Linguistics (ACL). *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing.* Sofia, Association for Computational Linguistics, 2013, pp. 34—42.

3. **Anisimovich K. V., Druzhkin K. Ju., Minlos F. R., Petrova M. A., Selegey V. P., Zuev K. A.** Syntactic and semantic parser based on ABBYY Compreno linguistic technologies. *Kompyuternaya lingvistika i intellektualnye tekhnologii: po materialam ezhegodnoy Mezhdunar. konf. "Dialog".* [*Computer linguistics and intelligent technologies: proc. of annual int. conf. "Dialog"*]. Bekasovo. 30 May — 3 June 2012, issue. 11 (18), vol. 2, Moscow, Izd-vo RGGU, 2012, l., 2012, pp. 91—103.

4. **Iomdin L., Petrochenkov V., Sizov V., Tsinman L.** ETAP parser: state of the art. *Komp'juternaja lingvistika i intellektual'nye tehnologii: po materialam ezhegodnoj Mezhdunar. konf. "Dialog".* [*Computer linguistics and intelligent technologies: proc. of annual int. conf. "Dialog"*]. Bekasovo. 30 May — 3 June 2012, issue. 11 (18), vol. 2: Doklady special'nyh sekcij, Moscow, Izd-vo RGGU, 2012, pp. 119—131.

5. **Ermakov A. E.** Eksplitsirovanie elementov smysla teksta sredstvami sintaksicheskogo analiza-sinteza. *Komp'yuternaya lingvistika i intellektual'nye tekhnologii: trudy Mezhdunar. konf. Dialog'2003.* Moscow, Nauka, 2003, pp. 136—140 (in Russian).

6. **Gladkii A. V**. *Sintaksicheskie struktury estestvennogo yazyka v avtomatizirovannykh sistemakh obshcheniya*. Moscow, Nauka, 1985, 144 p. (in Russian).

7. **Testelets Ya.G.** *Vvedenie v obshchii sintaksis.* Moscow, Izd-vo RGGU, 2001, 798 p. (in Russian).

8. **Smirnov Yu. M., Andreev A. M., Berezkin D. V., Brik A. V.** Ob odnom sposobe postroeniya sintaksicheskogo analizatora tekstov na estestvennom yazyke. *Izvestiya vuzov. Priborostroenie,* 1997, vol. 40, no. 5, pp. 34—42 (in Russian).

УДК 004.5

Я. А. Туровский, канд. мед. наук, зав. лаб., e-mail: yaroslav_turovsk@mail.ru,
С. Д. Кургалин, д-р физ.-мат. наук, зав. каф., **А. А. Вахтин,** канд. физ.-мат. наук, доц.,
С. В. Борзунов, канд. физ.-мат. наук, доц., Воронежский государственный университет,
А. С. Суровцев, аспирант, Воронежский государственный университет инженерных технологий

Сравнительные характеристики непроизвольно управляемых функционально зависимых человеко-компьютерных интерфейсов

Рассмотрены основные принципы работы непроизвольно управляемого функционально зависимого интерфейса человек—компьютер, основанного на регистрации функционального напряжения человека, отражающего, в том числе, и эмоциональный компонент его повседневной деятельности. Показана возможность применения этого сигнала в НФ-интерфейсе для формирования управляющих команд для операционных систем семейства Windows и работающих в них программных приложений.

Ключевые слова: *человеко-машинный интерфейс, вариабельность сердечного ритма, эмоционально зависимый интерфейс*

Введение

Развитие цифровых технологий и вычислительной техники, сделавшее их доступными для большого числа исследовательских групп и индивидуальных разработчиков, позволило существенным образом увеличить число каналов коммуникации человека и компьютера. Помимо традиционных каналов управления компьютером с использованием клавиатуры, мыши, джойстика и т. п., появилось значительное число интерфейсов, в различной степени адаптированных к новым форматам и особенностям команд, передаваемых пользователем компьютеру. Среди наиболее успешных интерфейсов подобного рода можно отметить Kinect [1] компании Microsoft, адаптированный для анализа движений, большое семейство стабилометрических платформ [2, 3], обеспечивающих передачу данных о положении центра тяжести тела пользователя, и так называемые ай-треккеры [4] (англ. *eye-tracking*), использующие направление взгляда пользователя, в том числе и как команду на выполнение тех или иных действий. Менее распространены в силу их недостаточной точности, высокой стоимости, сложности или узкой направленности применения другие интерфейсы. Например, миографические интерфейсы, основанные на использовании электрической активности мышц при создании команды, нашли широкое применение в миоэлектрическом протезировании, чаще всего — верхних конечностей [5], но оказались недостаточно эффективными для использования в игровых приложениях или в целях управления компьютером здоровыми пользователями [6]. Существующие нейрокомпьютерные интерфейсы (НКИ), обеспечивающие передачу команды от мозга на внешние устройства, минуя каналы нервной и мышечной активности, в значительной степени требуют усовершенствования как по точности (см. например [7]), так и по скорости работы (см. [8—11]). Их аппаратная часть, выполненная в значительной степени на основе клинических или экспертных электроэнцефалографов, имеет высокую стоимость, что исключает массовое применение НКИ большинством пользователей. Распространение бюджетных версий НКИ типа NIA [12] или Epoc [13] приостановилось в последнее время в связи с невысокой точностью их работы. Дыхательный интерфейс [14], предназначенный для больных тетрапарезом, несмотря на его простоту, малую стоимость и приемлемую точность, в силу особенности использования, заключающейся в присутствии дыхательного мундштука или близкого ему по функциям датчика, не представляет интереса для большинства пользователей современной цифровой техники.

Важно отметить, что подавляющее большинство перечисленных выше интерфейсов, по сути, конкурируют (для здоровых пользователей) или замещают (для людей с ограниченными возможностями) традиционные системы ввода команд в компьютер. Традиционные системы используют текстовый или графический способы ввода, когда необходимо либо

набрать текст команды, либо переместить курсор в определенную область экрана, соответствующую, опять же, той или иной команде.

Между тем существует значительный пласт информации, передача которой по традиционным каналам коммуникации человек—компьютер или интерфейсам, их заменяющим или дополняющим, практически невозможна. Речь идет о сведениях, характеризующих состояние функционального напряжения организма пользователя, связанного с активной умственной или физической работой, эмоциональными переживаниями. Такие сведения находят отражение в активности вегетативной нервной системы (ВНС), во многом отвечающей за непроизвольные реакции внутренних органов. Существующие в настоящее время способы регистрации непроизвольной активности органов или систем человека служат основой для использования систем биологической обратной связи (БОС) [15]. Такие системы используют программное обеспечение (ПО), да и сам компьютер, для того чтобы сообщить человеку в доступной для него форме о состоянии функционирования его органов или систем. Однако система БОС не затрагивает функционирование компьютера, т. е. она не позволяет пользователю тем или иным путем изменять параметры его работы или установленного на нем ПО.

Имеющийся задел исследований в направлении создания человеко-машинных интерфейсов (ЧМИ), обеспечивающих непроизвольное, минующее сознание пользователя управление компьютером на основе реакций ВНС, во многом пока еще представлен эмпирическими разработками [16, 17]. В связи с этим актуальным является проведение анализа процесса функционирования данных типов интерфейсов с тем, чтобы оценить возможность расширения направлений их применения и дальнейшей оптимизации.

Цель настоящей работы — сравнение характеристик разных типов непроизвольно управляемых функционально зависимых ЧМИ, использующих каналы коммуникации, несущие данные об активности ВНС человека в качестве команд для управления компьютером или другими цифровыми устройствами, анализ их характеристик и определение перспектив их использования.

1. Характеристики непроизвольно управляемых функционально-зависимых человеко-машинных интерфейсов на основе статистических показателей вариабельности сердечного ритма

Очевидно, что человеко-машинный интерфейс должен обеспечить передачу компьютеру конкретных команд, характеристики которых существенным образом зависят от решаемых интерфейсом целей, задач и его программно-аппаратной структуры. Действительно, с одной стороны, передача данных, например, об эмоциональном состоянии пользователя, с помощью клавиатуры возможна только в текстовом режиме, включая небуквенные символы (например, "смайлики" и т. п.). Следовательно, данный тип интерфейса имеет программно-аппаратные ограничения. С другой стороны, используя показатели стресс-состояния человека, нельзя обеспечить (в силу отсутствия в них специфики, низкой мощности множества, формирующего спектр состояний, и сложности произвольного контроля) скорость набора текста, сравнимую с использованием клавиатуры. Полученные разными методами показатели стресса применяют для изменения приоритетов и цветовой гаммы графических интерфейсов программ при работе с определенной операционной системой [16].

На первом этапе создания интерфейса необходимо оценить, какой из известных показателей функционального напряжения (ФН), стресса или эмоциональной нагрузки может быть наиболее приемлем для использования в эмоционально зависимом интерфейсе. Из значительного числа способов регистрации активности ВНС и, следовательно, ФН и в некоторой степени эмоционального состояния пользователя авторами были отобраны для исследования три наиболее простых, технологичных и достаточно информативных варианта оценки ФН: изменение параметров внешнего дыхания; кожно-гальваническая реакция; изменение вариабельности сердечного ритма (ВСР). Сравннение этих вариантов показало, что наиболее оптимальным является метод оценки ВСР. Причина в том, что в этом случае нет необходимости постоянного использования мундштука или аналогичного датчика, как в большинстве дыхательных интерфейсов, мала зависимость от состояния кожи, как в случае кожно-гальванической реакции, существует значительное число способов подключения датчиков к пациенту или здоровому пользователю и метод недорог. В дополнение к перечисленным преимуществам, самое главное, метод исследования ВСР имеет значительную теоретическую основу в области физиологии. Это обстоятельство позволяет наиболее корректным образом интерпретировать получаемую информацию. Традиционно анализ ВСР основывается либо на оценке временной последовательности *RR*-интервалов электрокардиограмм (рис. 1) [18, 19], либо (что является менее точным), на оценке временного диапазона между максимумами пульсовых волн на фотоплетизмограмме.

Традиционно в анализе ВСР используют как статистические, так и спектральные методы оценивания [19, 20]. Рассмотрим несколько вариантов интерфейса, генерирующего команды для ПО компьютера (включая операционную систему) на основе учета функционального напряжения пользователя, который в дальнейшем будем называть "НФ-интерфейс". При этом, согласно работе [16], в качестве эффектора может выступать изменение приоритетов работы приложений, запущенных пользователем на компьютере, или цветовой гаммы графического интерфейса, как его самого, так и интерфейса удаленного пользователя, соединенного с данной машиной по сети.

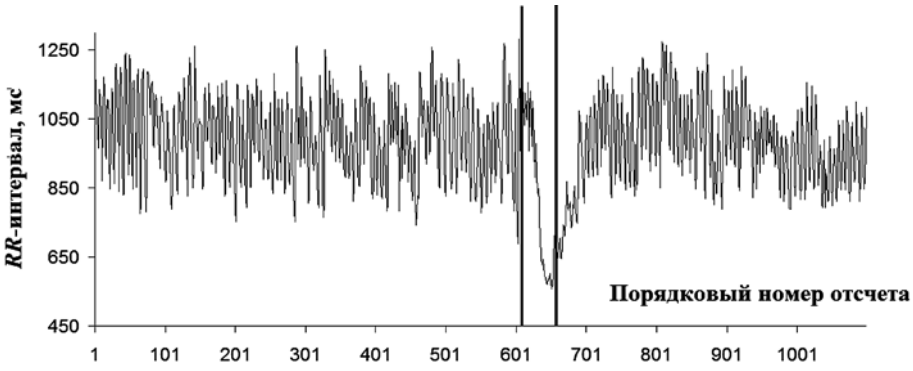


Рис. 1. Кривая вариабельности сердечного ритма в трех состояниях:
слева от левого вертикального маркера — состояние покоя; между маркерами — результат действия нагрузки в виде решения в уме арифметической задачи; справа от правого маркера — результат восстановления после нагрузки

В рассматриваемом случае статистические значения ВСР — значения среднеквадратического отклонения *RR*-интервалов (кардиоциклов) представлялись в виде вектора **V**.

Получение команд κ для работы данного интерфейса в самых простых случаях, когда командами служат сами значания *RR*-интервалов (пульса), можно описать с помощью следующих логических выражений:

$$\begin{cases} \left(\mathbf{V}_i \in \left([x_1, x_2], [y_1, y_2], [z_1, z_2]\right)\right) \Rightarrow \kappa_1; \\ \left(\mathbf{V}_{i+1} \in \left([x_3, x_4], [y_3, y_4], [z_3, z_4]\right)\right) \Rightarrow \kappa_2; \\ \left(\mathbf{V}_{i+d} \in \left(\left[x_{2d+1}, x_{2(d+1)}\right], \left[y_{2d+1}, y_{2(d+1)}\right], \left[z_{2d+1}, z_{2(d+1)}\right]\right)\right) \Rightarrow \kappa_d, \end{cases} \quad (1)$$

где индекс i нумерует векторы $\mathbf{V}_i$ для различных состояний; κ_1, ..., κ_d — команды операционной системе на функционирование в определенном режиме, на изменение параметров работы программного обеспечения компьютера или графических интерфейсов программ; d — число команд, которые нужно классифицировать; x_i, y_i, z_i — координаты вектора $\mathbf{V}_i$, представляющего собой последовательность кардиоциклов, обработанных определенным способом, в пространстве, обеспечивающем его разделение с заданной вероятностью с другими векторами $\mathbf{V}_j$ при $j \neq i$, при этом координаты векторов $\mathbf{V}_i$ и $\mathbf{V}_j$ не совпадают.

В ходе обучения интерфейса передаче команд, отражающих состояния ФН и ВНС конкретного пользователя, устанавливается соответствие между вектором $\mathbf{V}_i$ и командами κ_i для операционной системы и других программ. Иными словами, появление вектора $\mathbf{V}_i$ при анализе *RR*-интервалов трактуется как нахождение пользователя в определенном физиологическом состоянии. Этот факт означает, что необходимо применить команду κ_i для изменения параметров работы ПО. Аналогично, наличие вектора $\mathbf{V}_j$ свидетельствует о состоянии пользователя, при котором нужно применить другую команду κ_j. После того как обучение выполнено, т. е. изменение состояния пользователя привело к изменению параметров функционирования программ в желаемом направлении, интерфейс приступает к работе в режиме передачи команд. Применительно к указанным выше векторам обучение представляет собой фиксирование их значений при нахождении испытуемого в двух следующих состояниях: покоя с закрытыми глазами, в удобной для человека позе (2...5 мин); счет в уме, являющийся мощным фактором изменения не только ВСР, но и режима дыхания (не более 1 мин), что и составит время, необходимое для обучения интерфейса. Очевидно, с одной стороны, что время, в течение которого набирается последовательность *RR*-интервалов, должно быть достаточно большим, чтоб иметь возможность идентифицировать состояние, исключая случайные или чрезмерно кратковременные колебания на кривой ВСР. С другой стороны, это время не должно превышать определенных значений, учитывающих, что ФН может быть отражено через переходные процессы, занимающие достаточно короткие временные интервалы (подчас это 5...10 кардиоциклов). При этом размеры временных окон и для состояния покоя, и для функциональной пробы должны быть равными. Очевидно, в этом случае сложно или практически невозможно использовать индексы вариационной пульсометрии, поскольку они, в большинстве случаев, рассчитаны на "стационарный" в медицинском смысле сигнал. Таким является сигнал, зарегистрированный с человека в состоянии относительного покоя, при этом длительность зарегистрированного сигнала должна быть не менее 5 мин. Именно поэтому необходимо использовать более простые статистические показатели, подобные среднеквадратическому отклонению σ серии *RR*-интервалов. При этом важно в ходе обучения подобрать коридор исследуемых показателей таким образом, чтобы, с одной стороны, он отражал переходные процессы ВСР в рамках ФН, с другой стороны, минимизировал число ложных срабатываний при нахождении пользователя в состоянии относительного покоя. В этом случае точность обучения определяется самим пользователем, который и устанавливает, достаточно ли интерфейс чувствителен к изменению его состояния.

Следует учитывать, что, как и в случае среднеквадратического отклонения σ последовательности *RR*-интервалов, целый ряд индикаторов, рассчитываемых на основе друг друга, таких как дисперсия, коэффициент вариации, доверительный интервал среднего *RR*-интервала в покое, находится внутри определенного коридора значений. Выход за его пределы указывает на реализацию переходного процесса ВСР.

В случае одномерного вектора (при расчете, например, среднеквадратического отклонения σ) его

Формирование команд НФ-интерфейса для изменения приоритета ПО на основе текущих значений среднеквадратического отклонения ВСР σ_t и их предыдущих значений σ_{t-1}

Текущие значения σ_t	Предыдущие значения σ_{t-1}	
	$\sigma_{t-1} \in [\sigma_{min}, \sigma_{max}]$	$\sigma_{t-1} \notin [\sigma_{min}, \sigma_{max}]$
$\sigma_t \in [\sigma_{min}, \sigma_{max}]$	Не изменяется	Понижение приоритета ПО (изменение цветовой гаммы интерфейса на заданный шаг в направлении цвета, установленного пользователем)
$\sigma_t \notin [\sigma_{min}, \sigma_{max}]$	Повышение приоритета ПО (изменение цветовой гаммы интерфейса на заданный шаг в направлении цвета, установленного пользователем)	Не изменяется

полученные значения формировали следующее множество команд, учитывая предыдущие значения команд, отражающие состояние приоритета в рамках операционной системы семейства Windows (см. таблицу). В этом случае изменение приоритета ПО возможно только в пределах значений, соседних к стартовому значению приоритета. Для изменения приоритета во всем диапазоне возможностей необходимо увеличить до пяти число отрезков, в которых, помимо текущего, проводится анализ исследуемого показателя (в нашем случае — σ), например, в виде

$$\sigma_t \notin [\sigma_{min}, \sigma_{max}] \wedge \sigma_{t-1} \notin [\sigma_{min}, \sigma_{max}] \wedge \ldots$$
$$\ldots \wedge \sigma_{t-5} \notin [\sigma_{min}, \sigma_{max}] \Rightarrow \kappa,$$

где индекс t указывает на то, что значение σ рассчитывается в текущем окне, $t-1$ — значение рассчитывается в предыдущем окне, причем предыдущее окно может быть сдвинуто как на один кардиоцикл, так и на длину целого окна; σ_{min}, σ_{max} — нижняя и верхняя границы среднеквадратического отклонения σ при нахождении пользователя в состоянии покоя.

Естественно, что в случае приоритета "реального времени" команда на увеличение приоритета ПО будет игнорироваться, равно как и при значении приоритета "низкий" не будет выполняться команда на снижение приоритета. В случае же изменения цветовой гаммы графического интерфейса или при передаче сообщения об эмоциональном состоянии человека в сеть число отрезков, которые необходимо учитывать для корректной работы интерфейса, обеспечивающей возможность получения всех заданных состояний, будет определяться как $N-1$, где N — число состояний, в которых может находиться управляемое через НФ-интерфейс ПО или устройство-эффектор.

Функцию $f(\sigma_{t-1}, \sigma_t)$, отражающую формирование команд НФ-интерфейса на основе значения σ_t и предыдущего значения σ_{t-1}, можно записать в виде

$$\begin{aligned} f(\sigma_{t-1}, \sigma_t) = {} & \big(\theta(\sigma_{t-1} - \sigma_{min}) - \theta(\sigma_{t-1} - \sigma_{max})\big) \times \\ & \times \big(1 - \big(\theta(\sigma_t - \sigma_{min}) - \theta(\sigma_t - \sigma_{max})\big)\big) - \\ & - \big(\theta(\sigma_t - \sigma_{min}) - \theta(\sigma_t - \sigma_{max})\big) \times \\ & \times \big(1 - \big(\theta(\sigma_{t-1} - \sigma_{min}) - \theta(\sigma_{t-1} - \sigma_{max})\big)\big), \end{aligned} \quad (2)$$

где θ — функция Хэвисайда.

Значение функции $f(\sigma_{t-1}, \sigma_t) = 1$ соответствует повышению приоритета ПО, а $f(\sigma_{t-1}, \sigma_t) = 1$ — понижению приоритета. В случае $f(\sigma_{t-1}, \sigma_t) = 0$ приоритет ПО не изменяется.

При изменении параметров, содержащих бóльшую мощность состояний, чем управление приоритетом ПО, например, для цветовой гаммы графического интерфейса (в том числе и для удаленных мессенджеров) и/или образа героя компьютерной игры, набор команд для случая использования цветовой модели RGB можно представить формулой (2), где параметры σ_{min} и σ_{max} следует соотнести с границами диапазона 16-битного цвета. Для цветовой модели HSV параметры σ_{min} и σ_{max} соотносятся со значениями границ диапазонов тона, насыщенности или яркости цвета.

Следующим ключевым моментом процедуры создания интерфейсов является выбор размера окна (в виде числа *RR*-интервалов), в котором будет проводиться анализ их статистических показателей. Очевидно, что здесь необходимо учитывать время, в течение которого накапливаются данные кардиоциклов и, следовательно, возникает неустранимая задержка в работе интерфейса, лимитирующая его быстродействие.

Важно отметить, что поскольку событием, которое лежит в основе расчетов параметров управления, передаваемых через НФ-интерфейс, является длительность *RR*-интервала, то очевидно, что у двух пользователей с разными средними значениями пульса скорость работы интерфейса будет в среднем различаться (рис. 2). При этом разброс скоростей генерации команд для каждого интерфейса также может быть весьма велик, что определяется индивидуальной дисперсией *RR*-интервалов.

Нетрудно заметить, что для определения времени задержки в работе НФ-интерфейсов необходимо умножить полученные значения *RR*-интервалов на число усредненных кардиоциклов. В этом случае временная длительность одного окна, в котором проводится расчет статистического показателя, для двух пользователей будет различаться в среднем как для 5 кардиоциклов (4 и 5,25 с соответственно), так и для 30 кардиоциклов (24 и 30,6 с соответственно). Таким образом будет формироваться быстродействие от 4...5 с до 0,5 мин на одну команду.

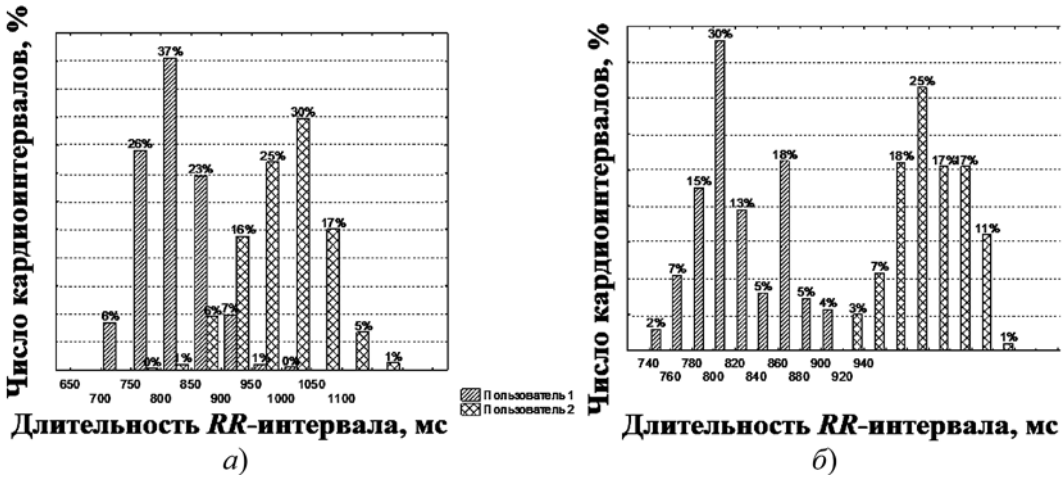


Рис. 2. Гистограмма распределения среднего *RR*-интервалов с окном усреднения:
а — 5 кардиоциклов для двух пользователей; *б* — 30 кардиоциклов для двух пользователей

Итак, устанавливая размер окна, пользователь задает латентное время интерфейса, т. е. время его реакции на внешние раздражители. Эмпирическим путем было получено, что оптимальный размер окна находится в пределах от пяти до десяти кардиоциклов (быстродействие НФ-интерфейса от одной команды в 10 с до одной команды в 2,5 с). Этот размер позволяет избежать ситуации, когда уже произошедшее событие, вызвавшее, например, кратковременный всплеск эмоций, получило отклик в работе НФ-интерфейса существенно (на 20...30 с, а то и на 1 мин) позже, чем это было необходимо для изменения функционирования устройства-эффектора при данном эмоциональном состоянии.

Следующим этапом конструирования интерфейса после оценки длительности временного окна для накопления данных будет выбор значений σ_{min} и σ_{max}, которые и позволят успешно генерировать нужные команды. Требования к этим значениям достаточно просты и заключаются в обеспечении, с одной стороны, срабатывания интерфейса при ФН, а с другой стороны, в минимальном (но ненулевом) числе реакций в спокойном состоянии пользователя.

На рис. 3 продемонстрирован один из этапов поиска таких значений. При этом черные маркеры — это верхняя граница σ_{max}, а белые маркеры — σ_{min}. Для круглых маркеров нижняя граница σ_{min} = 10 мс при окне в 5 кардиоинтервалов, для ромбовидных σ_{min} = 10 мс при окне в 30 кардиоинтервалов, для квадратных — верхняя граница σ_{max} = 210 мс при окне в 5 кардиоинтервалов, для треугольных — σ_{max} = 210 мс при окне в 30 кардиоинтервалов.

На рис. 3 видно, что по мере уменьшения расстояния между σ_{min} и σ_{max} число команд, генерируемых НФ-интерфейсом, существенно растет. Это приводит в ряде случаев к некорректной работе интерфейса. К таким случаям можно отнести чрезмерно частое изменение приоритета программ, с которыми в текущий момент работает пользователь. В то же

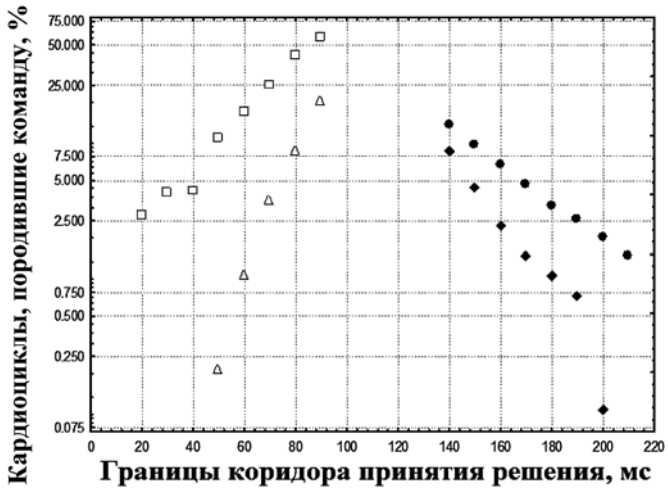


Рис. 3. Процент *RR*-интервалов (сдвигов окна анализа) от общего числа проанализированных *RR*-интервалов, выходящих за границы интервала [σ_{min}, σ_{max}] и порождающих тем самым команды для НФ-интерфейса

время в случае, если эффектором является изменение цветовой гаммы графического интерфейса как у пользователя данного компьютера, так и у его удаленного визави, подобный эффект можно признать допустимым.

Важным преимуществом НФ-интерфейса, основанного на статистических показателях, будет крайне низкая ресурсоемкость, что актуально для планшетных компьютеров небольшой мощности.

Тем не менее статистические методы анализа ВСР не освобождены от недостатков, в первую очередь связанных с невозможностью оценивать волновую структуру ВСР и судить на ее основе о функциональном напряжении пользователя. В этом случае более адекватными для физиологической интерпретации, и, следовательно, корректной работы являются НФ-интерфейсы на основе спектрального анализа ВСР.

2. НФ-интерфейсы на основе спектрального анализа вариабельности сердечного ритма

Если при генерации команд НФ-интерфейса используются спектральные оценки, то необходимо накопить определенное число осцилляций волн ВСР. При этом надо учитывать нестационарный характер переходного процесса, отражающего функциональное напряжение человека. Следовательно, лимитирующим фактором эпохи анализа (длительности накопленной последовательности кардиоциклов) будет число осцилляций са́мого низкочастотного из исследуемых компонентов сигнала. Для нижних значений LF-диапазона ВСР [19] время накопления составит не менее 20...25 с. То есть быстродействие НФ-интерфейса составит не более трех команд в 1 мин, это примерно в 2—4 раза медленнее, чем таковой показатель для статистических методов. Означенные показатели времени и ограничат максимальную временную границу пробы с функциональным напряжением, используемую для обучения интерфейса. При этом в окне заданной длительности можно использовать как классическое преобразование Фурье, так и вейвлет-преобразование [21], для которого оптимально подходят вейвлеты семейства Morlet (если разрешение по частоте более значимо, чем разрешение по времени) или WAVE (если разрешение по времени более актуально). Поскольку для человеко-машинных интерфейсов ключевым является необходимость работать в реальном масштабе времени или с минимальной временной задержкой, предпочтение отдают вейвлету WAVE, требующему для анализа тех же частотных диапазонов существенно меньшее число кардиоциклов по сравнению с вейвлетом Morlet.

Как и при использовании статистических показателей, при спектральных оценках будут анализироваться две группы данных: ВСР в состоянии покоя и ВСР при функциональной нагрузке. Двумерный вектор, характеризующий волновую структуру ВСР в пространстве (HF, LF) [19], нетрудно свести к одномерному вектору, если рассматриваемую сумму спектральной плотности мощности (СПМ) этих диапазонов принять за 100 %. Тогда исходя из идеи баланса регуляторных систем, порождающих волны в означенных диапазонах, сравнивают относительный вклад каждой из частот. При этом из формулы (1) можно получить:

$$\kappa = \begin{cases} 0, S_{\min}^{LF} + k_2^{LF}\left(S_{\max}^{LF} - S_{\min}^{LF}\right) < S_t^{LF} < \\ < S_{\min}^{LF} + k_1^{LF}\left(S_{\max}^{LF} - S_{\min}^{LF}\right); \\ 1, S_t^{LF} > S_{\min}^{LF} + k_1^{LF}\left(S_{\max}^{LF} - S_{\min}^{LF}\right); \\ -1, S_t^{LF} < S_{\min}^{LF} + k_2^{LF}\left(S_{\max}^{LF} - S_{\min}^{LF}\right), \end{cases} \quad (3)$$

где верхний индекс LF означает, что оценивается мощность LF-диапазона; S_t — текущие значения относительной (в %) спектральной плотности мощности для преобразования Фурье или вейвлетной плотности мощности (ВПМ) [22] для соответствующего преобразования; $S_{\min}$ и $S_{\max}$ — значения относительной (в %) СПМ или ВПМ, полученные в состоянии покоя или функционального напряжения пользователя соответственно; k_1 и k_2 — найденные эмпирически индексы границ активности, индивидуальные для каждого пользователя.

В случае применения выражения (3) значения k_1 и k_2 рассчитывают исходя из всей обучающей последовательности кардиоинтервалов. В ходе оценки этих значений для группы из 23 человек удалось получить, что $k_1 \in [0,58;\ 0,89]$ и $k_2 \in [0,1;\ 0,46]$, при этом для всей группы пользователей $k_1 - k_2 > 0,2$.

Очевидно, что если HF- и LF-диапазоны и их параметры рассчитывать исходя не из требования об их постоянстве в течение всего времени анализа [19, 20], а также учитывая изменения во времени частотных границ означенных физиологически значимых диапазонов [22], то формула (3) не претерпит изменений.

Теперь оценим скорость работы обученных интерфейсов. Минимальную задержку работы интерфейса по времени будем определять длительностью окна, в котором накапливается сигнал для обработки. В этом случае появление первых кардиоциклов переходного процесса, отражающего ФН, уже вызовет изменение спектра сигнала. Максимальную задержку по времени будем определять как удвоенное время накопления последовательности *RR*-интервалов. При этом минимальная задержка при появлении переходного процесса составит не более ≈7 с (максимальный период одной HF-волны), так как нижняя граница HF-диапазона составляет ≈0,15 Гц, что повысит быстродействие, делая его сравнимым с таковым для НФ-интерфейсов, использующих анализ среднеквадратического отклонения. При переходном процессе этот частотный диапазон ВСР практически всегда претерпевает существенные изменения. Максимальное же время задержки отклика интерфейса составит удвоенный период самой медленной компоненты LF-диапазона, т. е. около одной минуты (для исследуемых типов интерфейсов), что, очевидно, является неприемлемым показателем скорости его работы.

Таким образом, при одинаковых затратах времени на обучение интерфейса НФ-интерфейс, основанный на анализе как HF-, так и LF-диапазона, существенно проигрывает в скорости реакции на изменение состояния пользователя НФ-интерфейсу, созданному на основе только статистических оценок.

Для устранения этого недостатка был разработан упрощенный в плане спектрального оценивания интерфейс, в котором спектральный анализ проводился только по показателям абсолютных значений СПМ HF-диапазона без учета соотношения СПМ HF- и LF-диапазонов. В этом случае выражение (3) можно представить следующим образом:

$$\kappa = \begin{cases} 0, S'^{HF}_{\min} + k_2^{HF}\left(S'^{HF}_{\max} - S'^{HF}_{\min}\right) < S'^{HF}_t < S'^{HF}_{\min} + \\ + k_1^{HF}\left(S'^{HF}_{\max} - S'^{HF}_{\min}\right); \\ 1, S'^{HF}_t > S'^{HF}_{\min} + k_1^{HF}\left(S'^{HF}_{\max} - S'^{HF}_{\min}\right); \\ -1, S'^{HF}_t < S'^{HF}_{\min} + k_2^{HF}\left(S'^{HF}_{\max} - S'^{HF}_{\min}\right), \end{cases} \quad (4)$$

где верхний штрих у символов означает, что, в отличие от (3), расчет ведется по абсолютным значениям

СПМ, а не по их процентной доле в общем спектре LF- и HF-частот.

Использование в работе НФ-интерфейса только HF-диапазона позволило существенно сократить и время накопления кардиоциклов. Действительно, если при нижней границе частот LF-диапазона ВСР требуется накопить не менее 25 кардиоциклов (при *RR*-интервале в 10^3 мс), а фактически — несколько больше, то для HF-диапазона требуется накопление только 5...7 кардиоциклов. Увеличив окно анализа в 2 раза, получим для последовательности в 10...14 кардиоциклов размер временного окна, в котором осуществляется анализ ВСР, не более 15 с. Таким образом, по данному параметру НФ-интерфейс на основе спектрального оценивания по методу Фурье сравнялся с НФ-интерфейсом, использующим статистические показатели. Однако, не уступая предыдущему варианту НФ-интерфейса ни в скорости обучения, ни в скорости функционирования, данный тип НФ-интерфейсов имеет несколько большую ресурсоемкость. Это нивелируется очень небольшим числом точек (*RR*-интервалов, межпульсовых интервалов), подвергаемых преобразованию Фурье.

Определенной альтернативой НФ-интерфейсам на основе использования преобразования Фурье могут являться НФ-интерфейсы с применением вейвлет-преобразования. Подход с применением для анализа ВСР локальных вейвлет-спектров неоднократно обсуждался и, по сути, является аналогом применения преобразования Фурье. Однако в качестве спектральных параметров оценки функционального напряжения и, следовательно, наличия или отсутствия команд для интерфейса могут использоваться структуры цепочек локальных максимумов и минимумов матрицы квадратов коэффициентов вейвлет-преобразования, рассчитанные на тех же принципах цифровой обработки сигналов, что и для электроэнцефалограмм [23—28]. Действительно, переходные процессы ВСР и формирующиеся на их основе команды, которые передаются посредством НФ-интерфейсов, представляют особенности сигнала, порождающие хорошо детектируемые на вейвлет-диаграммах структуры локальных максимумов и минимумов (рис. 4). На рис. 4 представлены структуры локальных максимумов матрицы квадратов коэффициентов вейвлет-преобразования для разных пользователей, полученные с использованием вейвлета WAVE. Стрелками на рис. 4 отмечено начало переходного процесса ВСР, порожденного интеллектуальной нагрузкой (решением арифметической задачи в уме). Рис. 4 демонстрирует, что проведение функциональной пробы приводит к существенному изменению структуры локальных максимумов. При этом появляется группа цепочек локальных максимумов (ЦЛМ), относящихся к первому и второму частотным типам [25], захватывающая при этом значительный диапазон частот (0,002...0,14 Гц при пульсе 90 ударов/мин).

Важно отметить, что появление на вейвлет-диаграмме характерной группы ЦЛМ до начала функциональной пробы связано с особенностями численной реализации вейвлет-преобразования, учитывающей так называемый угол влияния вейвлета. Этот факт означает, что значения вейвлет-коэффициентов присваиваются одной точке во временной области на матрице вейвлет-коэффициентов (*RR*-интервалу, межпульсовому интервалу). Расчет при этом ведется по значительно большему числу таких точек, локализованных во времени. Таким образом, складывается ситуация, когда особенность сигнала уже влияет на значения вейвлет-коэффициентов, однако сами значения находятся на оси времени левее исходной особенности. Самые низкие частоты при этом требуют накопления значительного (порядка двух сотен) числа *RR*-интервалов, что, естественно, неприемлемо для НФ-интерфейсов, одно из требований к которым заключается в работе с минимальными временными задержками. Поэтому нижняя частотная граница, используемая в работе НФ-интерфейса, была ограничена масштабом вейвлет-преобразования, равным 5, что при частоте пульса в 90 ударов/мин даст нижнюю частоту используемого диапазона в 0,05 Гц.

Критерием, используемым при обучении интерфейса, служит изменение числа ЦЛМ в данном частотном диапазоне (рис. 5). При этом данный тип НФ-интерфейса в 2—5 раз уступает по быстродействию интерфейсу, созданному на основе статистических показателей или преобразования Фурье за счет существенно большего (до 40 кардиоциклов) окна анализа. На рис. 5 продемонстрированы два варианта динамики ЦЛМ в ходе обучения НФ-интерфейса. В одном случае (горизонтальная штриховка) обучение было успешным, так как функциональная проба, продолжавшаяся от 600-го до 700-го кардиоциклов, давала меньшее число ЦЛМ, чем ВСР пользователя в состоянии покоя (0...599-й кардиоциклы).

Во втором случае (вертикальная штриховка) обучение не завер-

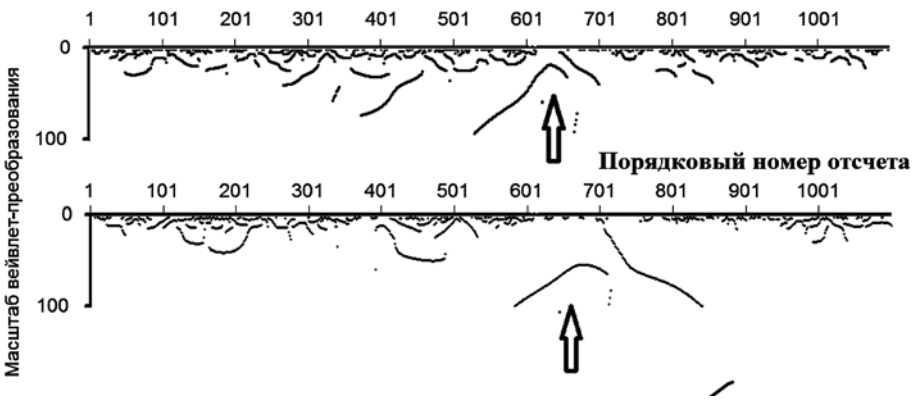


Рис. 4. Динамика ЦЛМ матрицы квадратов коэффициентов вейвлет-преобразования в состоянии относительного покоя и при проведении функциональной пробы

Рис. 5. Распределение числа ЦЛМ по отрезкам наблюдения в 100 кардиоциклов в процессе обучения НФ-интерфейса

шилось формированием диапазонов числа ЦЛМ отдельно для состояния покоя и для функционального напряжения. Детальное изучение ВСР показало, что в случаях, подобных представленному на рис. 5, даже в состоянии покоя пользователь демонстрирует признаки существенного функционального напряжения. Это обстоятельство не позволяет выделить состояние, возникшее в ходе проведения функциональной пробы, и, следовательно, делает работу интерфейса неэффективной.

Существенным недостаткам данного типа НФ-интерфейса является необходимость задействовать при его работе значительные вычислительные ресурсы для того, чтобы обеспечить его функционирование с минимальными временными задержками. Это делает работу такого интерфейса менее эффективной по сравнению с интерфейсами на основе статистических оценок и спектральных оценок по методу Фурье.

Описанные выше способы формирования команд для НФ-интерфейса апробированы как на последовательности *RR*-интервалов, так и на интервалах пик-пик фотоплетизмограмм.

Заключение

Рассмотрены основные принципы работы человеко-машинных интерфейсов, основанных на учете непроизвольно генерируемых пользователем команд. Такие команды связаны с так называемым функциональным напряжением человека — состоянием, которое характеризуется напряжением адаптационных механизмов, в том числе при стрессе, эмоциях, умственной нагрузке и т. п. Из трех вариантов физиологических сигналов (достаточно полно отражающих функциональное напряжение и способных, в силу простоты реализации аппаратных решений, быть основой для массовых НФ-интерфейсов), основанных на учете дыхательных движений, кожно-гальванической реакции и вариабельности сердечного ритма, в данной работе был выбран последний из них. Такой выбор обусловлен высоким уровнем физиологического обоснования, динамикой поведения данного сигнала в различных состояниях и большим разнообразием методов обработки кривой ВСР. Рассмотрено три типа НФ-интерфейсов: на основе статистических показателей ВСР, на основе спектрального оценивания по методу Фурье и на основе оценки структур локальных экстремумов матрицы квадратов коэффициентов вейвлет-преобразования. Продемонстрировано, что два первых типа интерфейсов имеют существенное преимущество относительно третьего в скорости реакции на изменение динамики ВСР, опережая его в 2—5 раз. Отмечено, что интерфейс на основе анализа статистических показателей представляет среди используемых типов наименее точную физиологическую информацию по сравнению с другими типами интерфейсов. Отметим, однако, что этой информации достаточно для выделения признаков функционального напряжения. Вместе с тем этот тип НФ-интерфейсов является наименее ресурсоемким, что позволяет рекомендовать его для устройств с ограниченными вычислительными ресурсами и малым энергопотреблением: смартфонов, планшетных компьютеров и т. д. Оптимальным с точки зрения точности функционирования интерфейса, физиологической интерпретации регистрируемых феноменов, скорости реакции на изменения состояния пользователя, ресурсоемкости и программной реализации является НФ-интерфейс на основе спектрального оценивания по методу Фурье. Такие качества позволяют рекомендовать его для создания нового канала коммуникации человек—компьютер, благодаря которому возможна машинная оценка, в том числе и эмоционального состояния пользователя. Подобный интерфейс существенно расширит возможности взаимодействия человека и компьютера, в первую очередь за счет способности машинной его части определять состояние пользователя, в том числе и эмоциональное, а это в дальнейшем выведет человеко-машинное взаимодействие на принципиально новый уровень.

Список литературы

1. **Meet** Kinect for Windows. URL: http://www.microsoft.com/en-us/kinectforwindows/

2. **Кубряк О. В., Гроховский С. С.** Практическая стабилометрия. Статические двигательно-когнитивные тесты с биологической обратной связью по опорной реакции. М.: Маска, 2012. 88 с.

3. **Stabilometric** platforms. URL: http://www.medicalexpo.com/medical-manufacturer/stabilometry-platform-3279.html

4. **Eye** tracking products. URL: http://www.tobii.com/en/eye-tracking-research/global/products/

5. **Martin W. C.** Upper Limb Prostheses: A Review of the Literature With a Focus on Myoelectric Hands WorkSafeBC. Evidence-Based Practice Group, 2011. 90 p.

6. **Gesture** Control Has Arrived. URL: https://www.thalmic.com/en/myo/

7. **Lotte F., Congedo M., L'ecuyer A.** et al. A review of classification algorithms for EEG-based brain—computer interfaces // Journ. Neural Eng. 2007. Vol. 4. P. R1—R13.

8. **Wolpaw J. R., Birbaumer N., McFarland D. J.** et al. Brain-computer interfaces for communication and control // Clinical Neurophysiology. 2002. Vol. 113. P. 767—791.

9. **Gao X., Xu D., Cheng M.** et al. A BCI-Based Environmental Controller for the Motion- Disabled // IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2003. Vol. 11. N. 2. P. 137—140.

10. **Zhu D., Bieger J., Molina G., Aarts R. M.** A Survey of Stimulation Methods Used in SSVEP-Based BCIs // Hindawi Publishing Corporation Computational Intelligence and Neuroscience. 2010. Article ID 702357.

11. **Борзунов С. В., Кургалин С. Д., Максимов А. В., Туровский Я. А.** Оценка скорости работы нейрокомпьютерного интерфейса, основанного на технологии SSVEP // Изв. РАН. ТиСУ. 2014. № 1. С. 121—129.

12. **OCZ's** Neural Impulse Actuator. URL: http://techreport.com/review/14957/ocz-neural-impulse-actuator

13. **Emotiv** EPOC / EPOC +: Scientific contextual EEG. URL: http://emotiv.com/epoc/features.php

14. **Resnick T. A., Stein R. L., Kern J.** et al. Anthony Osborne Protective hooded respirator with oral-nasal cup breathing interface. Patent US 6736137 B1.

15. **Сороко С. И., Трубачев В. В.** Нейрофизиологические и психофизиологические основы адаптивного биоуправления. СПб.: Политехника-сервис, 2010. 607 с.

16. **Туровский Я. А., Максимов А. В., Кургалин С. Д.** Способ оптимизации управления компьютером. Патент RU 2485572 от 10 мая 2012 г.

17. **Kaplan A. Ya., Lim J. J., Jin K. S.** et al. Unconscious operant conditioning in the paradigm of brain-computer interface based on color perception // Intern. Journ. Neuroscience. 2005. Vol. 115. P. 781—802.

18. **Баевский Р. М., Баринова А. П., Барсукова Ж. В.** Возрастные особенности сердечного ритма у лиц с разной степенью адаптации к условиям окружающей среды // Физиология человека. 1985. Т. 11, № 2. С. 208—212.

19. **Electrophysiology** Task Force. Heart rate variability. Standarts of measurement, physiological interpretation and clinical use // Circulation. 1996. Vol. 93. P. 1043—1065.

20. **Баевский Р. М., Иванов Г. Г., Чирейкин Л. В.** и др. Анализ вариабельности сердечного ритма при использовании различных электрокардиографических систем (методические рекомендации) // Вестник аритмологии. 2001. № 24. С. 65—87.

21. **Астафьева Н. М.** Вейвлет-анализ: основы теории и примеры применения // Успехи физических наук. 1996. Т. 166, № 11. С. 1145—1170.

22. **Туровский Я. А., Кургалин С. Д., Запрягаев С. А.** Способ исследования вариабельности сердечного ритма человека. Патент № 2326587. Зарегистрирован в Российском агентстве по патентам и товарным знакам 20 июня 2008 г.

23. **Туровский Я. А., Кургалин С. Д., Максимов А. В., Семенов А. Г.** Анализ электроэнцефалограмм на основе исследования изменяющейся во времени структуры локальных максимумов матрицы вейвлет-коэффициентов // Вестник Воронежского гос. ун-та. Системный анализ и информационные технологии. 2012. № 2. С. 69—73.

24. **Туровский Я. А.** "PikWave 1.0" Программный продукт. Зарегистрирован в Российском агентстве по патентам и товарным знакам (регистрационный № 2006613500).

25. **Туровский Я. А., Кургалин С. Д., Вахтин А. А.** Обработка сигнала электроэнцефалограммы на основе анализа частотных зависимостей и вейвлет-преобразования // Биомедицинская радиоэлектроника. 2012. № 12. С. 39—45.

26. **Туровский Я. А., Кургалин С. Д., Семенов А. Г.** Динамика цепочек локальных максимумов спектров электроэнцефалограмм человека // Биофизика. 2014. Т. 59, Вып. 1. С. 185—190.

27. **Кургалин С. Д., Туровский Я. А., Максимов А. В., Насер Н.** Вейвлет-анализ энцефалограмм // Информационные технологии в проектировании и производстве. 2010. № 1. С. 89—95.

28. **Туровский Я. А., Кургалин С. Д., Максимов А. В.** Выбор анализирующих вейвлетов для системы с параллельной обработкой биомедицинских данных // Вестник Воронежского гос. ун-та. Сер. Системный анализ и информационные технологии. 2011. № 2. С. 74—79.

Ya. A. Turovsky, Associate Professor, Head of Laboratory, e-mail: yaroslav_turovsk@mail.ru,
S. D. Kurgalin, Head of Department, **A. A. Vahtin,** Associate Professor,
S. V. Borzunov, Associate Professor, Voronezh State University, **A. S. Surovcev,** Postgraduate Student,
Voronezh State University of Engineering Technologies

Comparative Description Involuntarily Controllable Functional Dependent Human-Computer Interfaces

We have studied the basic principles of functioning for spontaneously managed functionally-dependent human-PC interface. The interface is based on recording of human's functional tension, which, in particular, reflects the emotional component of a person's daily activities. In order to manage the interface we used heart rate variability signal (HRV). In this paper we have demonstrated that this signal can be applied in the interface for generating controlling commands in the Windows operating systems and their software applications. Also, we showed that the interface based on Fourier transformation or on evaluation of statistical parameters, is significantly faster in data transfer than an interface based on the wavelet transformation. It is proven that in terms of functioning accuracy, physiological interpretation of the recorded phenomena, time of reaction to changes in the user's state, low intensity and simplicity of the software implementation, the appropriate one is human-PC interface based on Fourier's method of spectral evaluation. This allows us to recommend it for creating a new kind of functionally-dependent channel of communication between a human and a computer.

Keywords: *human-computer interfaces, involuntarily controllable interfaces, emotion*

References

1. **Meet** Kinect for Windows, available at: http://www.microsoft.com/en-us/kinectforwindows/

2. **Kubryak O. V., Grohovskij S. S.** *Prakticheskaya stabilometriya. Staticheskie dvigatelno kognitivnye testy s biologicheskoj obratnoj svyazyu po opornoj reakcii*, Moscow, Maska, 2012, 88 p. (in Russian).

3. **Stabilometric** platforms, available at: http://www.medicalexpo.com/medical-manufacturer/stabilometry-platform-3279.html

4. **Eye** tracking products, available at: http://www.tobii.com/en/eye-tracking-research/global/products/

5. **Martin W. C.** *Upper Limb Prostheses: A Review of the Literature With a Focus on Myoelectric Hands WorkSafeBC*, Evidence-Based Practice Group, 2011, 90 p.

6. **Gesture** Control Has Arrived, available at: https://www.thalmic.com/en/myo/

7. **Lotte F., Congedo M., Lecuyer A., Lamarche F., Arnaldi F.** A review of classification algorithms for EEG-based brain—computer interfaces. *Journ. Neural Eng.*, 2007, vol. 4, pp. R1—R13.

8. **Wolpaw J. R., Birbaumer N., McFarland D. J, Pfurtscheller G., Vaughan T. M.** Brain—computer interfaces for communication and control, *Clinical Neurophysiology*, 2002, vol. 113, pp. 767—791.

9. **Gao X. R., Xu D. F., Cheng M., Gao S. K.** A BCI-Based Environmental Controller for the Motion- Disabled. *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* 2003, vol. 11, no. 2, pp. 137—140.

10. **Zhu D., Bieger J., Molina G., Aarts R. M.** A Survey of Stimulation Methods Used in SSVEP-Based BCIs. *Hindawi Publishing Corporation Computational Intelligence and Neuroscience*, 2010, Article ID 702357.

11. **Borzunov S. V., Kurgalin S. D., Maksimov A. V., Turovskij Ya. A.** Ocenka skorosti raboty nejrokompyuternogo interfejsa osnovannogo na tekhnologii SSVEP*, Izv. RAN TiSU,* 2014, no. 1, pp. 121—129 (in Russian).

12. **OCZ's** Neural Impulse Actuator, available at: http://techreport.com/review/14957/ocz-neural-impulse-actuator

13. **Emotiv** EPOC / EPOC +: Scientific contextual EEG, available at: http://emotiv.com/epoc/features.php

14. **Resnick T. A., Stein R. L., Kern J., Osborne A.** Protective hooded respirator with oral-nasal cup breathing interface. Patent US 6736137 B1.

15. **Soroko S. I., Trubachev V. V.** *Nejrofiziologicheskie i psihofiziologicheskie osnovy adaptivnogo bioupravleniya,* Saint Petersburg, Politekhnika servis, 2010, 607 p. (in Russian).

16. **Turovskij Ya. A., Maksimov A. V., Kurgalin S. D.** Sposob optimizacii upravleniya kompyuterom. Patent RU 2485572, 10.05.2012 (in Russian).

17. **Kaplan A.Ya., Lim J. J., Jin K. S., Park B. W., Byeon J. G., Tarasova S. U.** Unconscious operant conditioning in the paradigm of brain-computer interface based on color perception, *Intern. Journ. Neuroscience*, 2005, vol. 115, pp. 781—802.

18. **Baevskij R. M., Barinova A. P., Barsukova Zh. V.** Vozrastnye osobennosti serdechnogo ritma u lic s raznoj stepenyu adaptacii k usloviyam okruzhayushchej sredy, *Fiziologiya Cheloveka*, 1985, vol. 11, no. 2, pp. 208—212 (in Russian).

19. **Electrophysiology** Task Force. Heart rate variability. Standarts of measurement, physiological interpretation and clinical use, *Circulation*. 1996, vol. 93, pp. 1043—1065.

20. **Baevskij R. M., Ivanov G. G., Chirejkin L. V., Gavrilushkin A. P., Dovgalevskij P. Ja., Kukushkin Ju. A., Mironova T. F., Priluckij D. A., Semenov A. V., Fedorov V. F., Flejshman A. N., Medvedev M. M.** Analiz variabelnosti serdechnogo ritma pri ispolzovanii razlichnyh ehlekrokardiograficheskih sistem metodicheskie rekomendacii, *Vestnik Aritmologii,* 2001, no. 24, pp. 65—87 (in Russian).

21. **Astafeva N. M.** Vejvlet analiz osnovy teorii i primery primeneniya, *Uspekhi Fizicheskih Nauk,* 1996, vol. 166, no. 11, pp 1145—1170 (in Russian).

22. **Turovskij Ya. A., Kurgalin S. D., Zapryagaev S. A.** Sposob issledovaniya variabelnosti serdechnogo ritma cheloveka. Patent 2326587. Zaregistrirovan v Rossijskom agentstve po patentam i tovarnym znakam 20.07.2008.

23. **Turovskij Ya. A., Kurgalin S. D., Maksimov A. V., Semenov A. G.** Analiz ehlektroehncefalogramm na osnove issledovaniya izmenyayushchejsya vo vremeni struktury lokalnyh maksimumov matricy vejvlet koehfficientov, *Vestnik Voronezhskogo gos. un-ta. Sistemnyj Analiz i Informacionnye Tekhnologii,* 2012, no. 2, pp. 69—73 (in Russian).

24. **Turovskij Ya. A.** PikWave 1 0 Programmnyj product. Zaregistrirovan v Rossijskom agentstve po patentam i tovarnym znakam registracionnyj, no. 2006613500.

25. **Turovskij Ya. A., Kurgalin S. D., Vahtin A. A.** Obrabotka signala ehlektroehncefalogrammy na osnove analiza chastotnyh zavisimostej i vejvlet preobrazovaniya. *Biomedicinskaya Radioehlektronika,* 2012, no. 12, pp. 39—45 (in Russian).

26. **Turovskij Ya. A., Kurgalin S. D., Semyonov A. G.** Dinamika cepochek lokalnyx makcimumov cpektpov ehlektpoehncefalogpamm cheloveka, *Biofizika*, 2014, vol. 59, no. 1, pp. 185—190 (in Russian).

27. **Kurgalin S. D., Turovskij Ya. A., Maksimov A. V., Naser N.** Vejvlet analiz ehncefalogramm, *Informacionnye Tekhnologii v Proektirovanii i Proizvodstve*, 2010, no. 1, pp. 89—95 (in Russian).

28. **Turovskij Ya. A., Kurgalin S. D., Maksimov A. V.** Vybor analiziruyushchih vejvletov dlya sistemy s parallelnoj obrabotkoj biomedicinskih dannyh, *Vestnik Voronezhskogo Gos, Un-ta, Sistemnyj Analiz i Informacionnye Tekhnologii,* 2011, no. 2, pp. 74—79 (in Russian).

ИНФОРМАЦИЯ

28 марта — 1 апреля 2016 г. в Архангельске в Северном (Арктическом) федеральнм университете состоится Международная научная конференция **"Параллельные вычислительные технологии (ПаВТ) 2016"**, десятая в серии ежегодных конференций, посвященных развитию и применению параллельных вычислительных технологий в различных областях науки и техники.

Официальный сайт конференции: http://agora.guru.ru/pavt2016/

УДК 004.825

С. Д. Махортов, д-р физ.-мат. наук, доц., зав. кафедрой, e-mail: makhortov@amm.vsu.ru,
Воронежский государственный университет

Алгебраическая модель распределенной логической системы продукционного типа

Статья посвящена развитию теории LP-структур как фундаментального алгебраического подхода для построения расширяемого спектра продукционных и подобных им систем в информатике. Описана расширенная алгебраическая модель, выразительные возможности которой охватывают распределенные продукционно-логические системы. В рамках такой модели преимущества теории LP-структур оказываются доступными при построении и исследовании распределенных продукционных систем.

Ключевые слова: *распределенная база знаний, продукционная система, LP-структура, логическое замыкание, эквивалентные преобразования, логическая редукция*

Введение

Современное состояние информационных технологий характеризуется значительным увеличением масштабов и критической значимости решаемых с их помощью задач. Соответственно, возрастают значение и актуальность строгих обоснований корректности и надежности процессов обработки информации в различных предметных областях. Одна из таких областей — распределенные системы, основанные на знаниях. К подобным практически значимым системам относят, например, следующие: система поиска и интеллектуального анализа тематической информации в больших и сверхбольших хранилищах данных, поддерживающая на каждом своем узле базу знаний по его содержанию; система медицинской диагностики, где в постановке диагноза участвует "виртуальный консилиум" баз знаний, разработанных различными специалистами; обучающая система, функционирующая в глобальной сети; интеллектуальная информационная система корпорации и др. [1, 2].

Наиболее распространенным подходом к построению и исследованию распределенного искусственного интеллекта можно считать концепцию мультиагентных систем [3—5]. Существуют и другие подходы, имеющие собственные преимущества при решении конкретных задач, связанных с распределенными интеллектуальными системами [2, 6, 7].

В последнее десятилетие автором и его учениками получен ряд важных результатов в области управления знаниями. Они связаны с широко распространенными в информатике логическими системами продукционного типа [7, 8]. Разработана алгебраическая теория LP-структур [9], обеспечивающая обоснование и эффективное решение задач эквивалентных преобразований, верификации, минимизации баз знаний, а также ускорения логического вывода, в том числе с использованием параллельных вычислений [10, 11]. Созданы программные продукты, реализующие теорию и демонстрирующие ее практическую значимость [12]. Она оказалась применимой и для описания других систем, ранее не считавшихся продукционными [13, 14].

Настоящая статья посвящена следующему шагу в развитии теории LP-структур как фундаментального алгебраического подхода для построения расширяемого спектра продукционных и подобных им систем. Для соответствующих баз знаний он состоит в переходе от параллельной обработки к распределенной. В работе сформулирована расширенная алгебраическая модель, выразительные возможности которой охватывают распределенные продукционно-логические системы. В результате преимущества теории LP-структур оказываются доступными при построении и исследовании таких систем.

Статья состоит из следующих основных разделов. В разд. 1 введены необходимые базовые понятия и обозначения, в качестве отправной точки сформулированы имеющиеся математические результаты. В разд. 2 дана исходная постановка, введено понятие распределенной LP-структуры и установлены ее основные свойства. В Заключении подведены итоги исследования и указаны некоторые перспективы.

1. Терминология LP-структур и базовые положения

Обозначения и определения, необходимые для дальнейшего изложения, подробно описаны в монографии [9]. Напомним некоторые основные из них.

Бинарное отношение R на произвольном множестве F называется:

- рефлексивным, если для любого $a \in F$ справедливо $(a, a) \in R$;
- транзитивным, если для любых $a, b, c \in F$ из $(a, b), (b, c) \in R$ следует $(a, c) \in R$.

Известно [15], что существует замыкание R^* произвольного отношения R относительно свойств рефлексивности и транзитивности (рефлексивно-транзитивное замыкание).

Обратная задача — нахождение транзитивной редукции: по данному R ищется минимальное отношение R', такое, что его транзитивное замыкание совпадает с транзитивным замыканием R. Напомним также, что для частично упорядоченных множеств различают понятия минимального элемента (для него нет меньшего элемента) и наименьшего элемента (он меньше всех) [16]. В работе [15] приведен алгоритм построения транзитивной редукции ориентированных графов. Показано, что эта задача вычислительно эквивалентна построению транзитивного замыкания, а также доказана единственность транзитивной редукции ациклического графа.

Базовые сведения о математических решетках содержатся, например, в работе [16]. Решеткой называют множество с частичным порядком $\leqslant$ ("не больше", "содержится"), где для любой пары элементов определены операции $\wedge$ ("пересечение") — точная нижняя грань этой пары и $\vee$ ("объединение") — ее точная верхняя грань.

Решетка $\mathbb{F}$ называется ограниченной, если она содержит общие нижнюю и верхнюю грани, а именно — такие два элемента O, I, что $O \leqslant a \leqslant I$ для любого $a \in \mathbb{F}$. Атомом ограниченной (снизу) решетки $\mathbb{F}$ называется минимальный элемент ее подмножества $\mathbb{F}\backslash\{O\}$. Решетка называется атомно-порожденной, если каждый ее элемент разложим в виде объединения атомов.

Под *LP-структурой* подразумевают алгебраическую систему, представляющую собой решетку $\mathbb{F}$, на которой задано дополнительное бинарное отношение R, обладающее некоторыми (продукционно-логическими) свойствами. Решетка $\mathbb{F}$ в контексте данного определения рассматривается в широком смысле, и ее тип может уточняться в конкретных моделях и приложениях.

Введенное отношение R моделирует совокупность правил (продукций) интеллектуальной системы. Элементы решетки соответствуют предпосылкам и заключениям этих правил, содержащим элементарные факты базы знаний и выражения над ними.

Отношение R называют *продукционно-логическим*, если оно обладает рефлексивностью, т. е. содержит все пары вида (a, a), транзитивностью и другими свойствами, которые определяются конкретной моделью. Одно из таких свойств — *дистрибутивность*. Неформально дистрибутивность отношения означает возможность логического вывода по частям и совмещения его результатов на основе решеточных операций $\wedge$ и $\vee$. Заданное на абстрактной решетке отношение R называют:

- $\wedge$-дистрибутивным, если из $(a, b_1), (a, b_2) \in R$ следует $(a, b_1 \wedge b_2) \in R$;
- $\vee$-дистрибутивным, если из $(a_1, b), (a_2, b) \in R$ следует $(a_1 \vee a_2, b) \in R$.

В общем случае отношение называется дистрибутивным при наличии обоих указанных свойств.

В настоящей работе в качестве основы LP-структур рассматриваются решетки с семантикой подмножеств — $\lambda(F)$ (множество конечных подмножеств универсума F) или булеан (множество всех подмножеств F). Поэтому вместо символов $\leqslant$, $\geqslant$, $\wedge$ и $\vee$ используются знаки теоретико-множественных операций $\subseteq$, $\supseteq$, $\cap$ и $\cup$, а элементы решетки обозначают, как правило, заглавными буквами. Исключение составляют атомы, обозначаемые строчными буквами. Атомами являются все подмножества, состоящие ровно из одного элемента универсума.

В рассматриваемой модели под дистрибутивностью отношения подразумевается лишь второе из двух указанных выше свойств. В такой нотации $\cup$-дистрибутивность трактуется в следующем смысле: из (A, B_1), $(A, B_2) \in R$ следует $(A, B_1 \cup B_2) \in R$. Выбор упрощенной модели обусловлен намерением больше сосредоточиться на концепции распределенной LP-структуры. В дальнейшем основная идея может быть перенесена на другие типы решеток и соответствующих LP-структур.

Для LP-структур актуальны следующие основные вопросы [9]: о замыкании, об эквивалентных преобразованиях, о канонической форме, о логической редукции. Решение этих вопросов предоставляет возможности для обоснования, автоматизированного исследования и оптимизации множеств продукций в различных предметных областях.

Логическим замыканием произвольного бинарного отношения R называется наименьшее продукционно-логическое отношение, содержащее R. Два отношения R_1, R_2 называют (*логически*) *эквивалентными*, если их логические замыкания совпадают. Для таких отношений используется обозначение $R_1 \sim R_2$. *Эквивалентным преобразованием* данного отношения называется некоторая замена подмножества его пар, приводящая к эквивалентному отношению.

Ниже приведены доказанные автором теоремы о существовании логического замыкания произвольного бинарного отношения и о принципе локальности эквивалентных преобразований логических отношений. С их доказательствами можно ознакомиться в работе [9]. Представленные результаты открывают возможности для автоматических преобразований баз знаний.

Теорема 1.1. Для произвольного бинарного отношения R на решетке существует логическое замыкание $\xrightarrow{R}$.

Теорема 1.2. Пусть R_1, R_2, R_3, R_4 — отношения на общей решетке. Если при этом $R_1 \sim R_2$ и $R_3 \sim R_4$, то $R_1 \cup R_3 \sim R_2 \cup R_4$.

Отношение R на атомно-порожденной решетке $\mathbb{F}$ называется каноническим, если оно задано множе-

ством пар вида (A, a), где $A \in \mathbb{F}$, a — атом в $\mathbb{F}$ (т. е. неразложимый элемент).

Утверждение 1.1. Для любого отношения на атомно-порожденной решетке существует эквивалентное ему каноническое отношение.

Рассмотрен также вопрос о минимизации бинарных отношениий с сохранением их свойств. *Логической редукцией* отношения R на решетке называется любое минимальное отношение, эквивалентное R. Справедлива теорема о существовании логической редукции и способе ее построения [9].

Теорема 1.3. Для произвольного бинарного отношения R на решетке существует логическая редукция.

Поскольку LP-структура представляет собой алгебраическую модель базы знаний, напомним далее терминологию, связанную с простым видом логических систем продукционного типа — экспертными продукционными системами.

Указанные системы манипулируют множествами фактов и правил (продукций). *Факт* представляет единицу декларативной информации — некоторое суждение о внешнем мире. Стандартным представлением факта является триплет вида "объект.атрибут = значение" (например, "термометр.температура = высокая"). В работе [17], описывающей практическую реализацию продукционных систем, триплет редуцируется к паре "параметр = значение", т. е. объект и атрибут интегрируются в единый параметр. В связи с этим "термометр.температура" и "термометр.изготовитель" просто считаются разными параметрами. В настоящей работе пойдем дальше, интегрируя в факт параметр и его значение, считая факт независимым элементом общего множества фактов. Такое упрощение делается, чтобы больше сосредоточиться на основных идеях конструирования и применения распределенных LP-структур. В дальнейшем можно реализовать и более сложную конструкцию фактов, скорректировав соответствующим образом описание LP-структуры. Например, можно перейти к продукционно-фреймовой модели знаний [2].

Продукционная система содержит *рабочую память*. Это некоторое подмножество фактов, которые на текущий момент считаются выполненными. Такое множество называется также *базой данных* продукционной системы.

Правило (*продукция*) состоит из предпосылки и заключения. *Предпосылка* обычно представляет собой выражение над фактами (например, их конъюнкцию или дизъюнкцию). Предпосылка может быть выполненной (истинной) или невыполненной (ложной) при текущем состоянии рабочей памяти. Если предпосылка верна, то правило может быть применено. *Заключение* — это действие, которое можно осуществить, если верна предпосылка (например, добавить к рабочей памяти некоторый новый факт). *Применение правила* состоит в выполнении действия заключения. В контексте настоящей работы рассматриваются приложения, в которых заключение, как и предпосылка, является выражением над фактами, и соответствующее заключению действие интерпретируется как "считать истинным". Таким образом, в данном случае применение правила означает некоторую модификацию рабочей памяти, обычно — запись в нее тех фактов, справедливость которых вытекает из истинности выражения в заключении правила. Совокупность правил называется *базой знаний*.

Прямым выводом в продукционной системе называют процесс циклического применения правил к содержимому рабочей памяти (его исходное состояние задано в начале работы) и, соответственно, получение в результате новых фактов, которые считаются справедливыми. Прямой вывод может осуществляться до тех пор, пока получение новых фактов станет невозможным (при текущем содержимом рабочей памяти не окажется ни одной истинной предпосылки правила, заключение которого способно изменить рабочую память). *Обратный вывод* — это противоположный процесс. В нем по некоторому набору результирующих фактов — *гипотезе*, путем анализа правил в направлении от заключения к предпосылке, подтверждается или опровергается справедливость гипотезы при заданном исходном содержимом рабочей памяти. В процессе обратного вывода содержимое рабочей памяти также меняется. *Машина вывода* — программный модуль, который непосредственно реализует прямой или обратный вывод.

Предлагаемый подход к исследованию продукционных и подобных им систем, а также к усовершенствованию обратного вывода основан на представлении множеств фактов и правил LP-структурой. Каждый элементарный факт отображается атомом решетки, предпосылка и заключение правила — соответствующими элементами решетки, а сами правила представляются парами бинарного отношения R.

2. Распределенная LP-структура и ее свойства

Как отмечено в работе [18], в литературе есть различные определения *распределенных систем*, многие из которых не являются полными и, как правило, не согласуются с другими определениями. Для целей, которые преследует настоящая статья, также достаточно вольной и интуитивно понятной интерпретации этого термина.

Предположим, что имеется распределенная вычислительная сеть, в каждом узле которой может храниться собственный набор фактов (база данных), подмножество продукционных правил (база знаний), а также может функционировать машина логического вывода с локальной рабочей памятью. Таким образом, существует распределенная продукционная система, состоящая из набора локальных продукционных систем (подсистем), согласованных друг с другом. Под согласованностью можно, в частности, понимать выполнение перечисленных ниже трех условий, сформулированных в работе [7].

1. Идентичные по семантике элементы данных, описывающие предметные области различным подсистем, представляются одними и теми же фактами.

2. В различных подсистемах продукционные правила построены на основе общего синтаксиса.

3. Попарные пересечения множеств, хранимых в различных подсистемах фактов, непусты.

Условия 1 и 2 позволяют свободно копировать или переносить продукции из одной подсистемы в другую, а условие 3 дает возможность организовывать "консилиумы" подсистем.

Каждому факту и каждому правилу распределенной продукционной системы ставится в соответствие подмножество узлов вычислительной сети, на которых хранятся этот факт и это правило. Предполагается, что подсистемы в основном более или менее самостоятельны, но требуют некоторого (относительно дорогостоящего) взаимодействия, интенсивность которого в общем случае существенно ниже интенсивности локальных вычислений.

Цель построения и исследования общей алгебраической модели состоит в том, чтобы с ее помощью не только решать описанные выше задачи управления знаниями, но и оптимизировать знания для снижения сетевого трафика между узлами в процессе логического вывода. Сразу отметим, что полное "размножение" базы данных по всем узлам не представляется возможным, поскольку общее число хранимых фактов в общем случае может оказаться для этого чрезмерно большим.

Обозначим N — множество узлов вычислительной сети, $\mathbb{N}$ — порожденный им булеан, т. е. множество всех его подмножеств. Элементарные факты и правила продукционной системы будут помечаться элементами решетки $\mathbb{N}$ как атрибутами.

Итак, пусть даны две решетки: $\mathbb{F}$ (базовая решетка для LP-структуры) и $\mathbb{N}$ (решетка узлов). На решетке $\mathbb{F}$ определено отображение *Nodes*(), которое каждому атому $a \in \mathbb{F}$ ставит в соответствие единственный непустой элемент $X \in \mathbb{N}$ (т. е. $Nodes(a) = X$, $X \neq \varnothing$). На решетке $\mathbb{F}$ задается бинарное отношение R. Кроме того, каждой паре $(A, B) \in R$ также ставится в соответствие непустой элемент $Y \in \mathbb{N}$ ($Nodes(A, B) = Y$, $Y \neq \varnothing$).

В предметной интерпретации (т. е. в моделируемой продукционной системе) функция *Nodes*() определяет для каждого элементарного факта или каждого правила совокупность узлов распределенной вычислительной системы, где этот факт или правило хранится. Еще раз подчеркнем, что область значений функции *Nodes*() не содержит пустого элемента $\varnothing$, т. е. каждый элементарный факт базы данных и каждое правило базы знаний хранятся по крайней мере на одном узле распределенной системы.

Определение 2.1. Описанная выше алгебраическая система называется *распределенной LP-структурой*.

Замечание 2.1. При наличии отображения *Nodes*() решетка $\mathbb{F}$ и отношение R также могут считаться *распределенными*.

Далее для нового понятия распределенной LP-структуры будет рассмотрен стандартный круг вопросов, а именно — о замыкании, об эквивалентных преобразованиях, о канонической форме, о логической редукции.

Определение 3.2. Распределенное бинарное отношение R на распределенной решетке $\mathbb{F}$ называется *продукционно-логическим*, если оно рефлексивно, транзитивно и дистрибутивно, причем для каждой пары $(A, B) \in R$ справедливо $Nodes(A, B) = N$. Логическим замыканием отношения R называется наименьшее продукционно-логическое отношение, содержащее R.

Замечание 2.2. Данное определение не означает его применения на практике, т. е. совокупность правил распределенной продукционной системы не обязана храниться одновременно на всех узлах сети. Это определение носит в основном теоретический характер и будет использоваться в дальнейшем для обоснования других важных для предметной области результатов, в том числе и с практической точки зрения.

Существование логического замыкания для произвольного распределенного отношения непосредственно вытекает из соответствующего базового результата (теорема 1). Достаточно построить для R логическое замыкание $\xrightarrow{R}$ в обычном смысле и пометить каждую пару атрибутом N. Таким образом, справедливо следующее утверждение, сформулированное в виде теоремы ввиду его важности, но не сложности доказательства.

Теорема 2.1. Для произвольного распределенного бинарного отношения R на распределенной решетке существует логическое замыкание $\xrightarrow{R}$.

Аналогично и без дополнительных трудностей переносится на распределенную LP-структуру концепция эквивалентных преобразований и ее обоснование.

Определение 2.3. Два распределенных отношения R_1, R_2 на распределенной решетке $\mathbb{F}$ называют *эквивалентными* ($R_1 \sim R_2$), если их логические замыкания (в смысле определения 2.2) совпадают. *Эквивалентным преобразованием* данного отношения называется замена подмножества его пар, приводящая к эквивалентному отношению.

Из теоремы 1.2 и теоремы 2.1 непосредственно можно прийти к следующему результату.

Теорема 2.2. Пусть R_1, R_2, R_3, R_4 — распределенные отношения на общей распределенной решетке. Если при этом $R_1 \sim R_2$ и $R_3 \sim R_4$, то $R_1 \cup R_3 \sim R_2 \cup R_4$.

Для доказательства этой теоремы достаточно к отношениям $R_1 \cup R_3$ и $R_2 \cup R_4$ применить теорему 1.2, после чего в построенных замыканиях двух отношений пометить каждую пару атрибутом N.

Замечание 2.3. При выполнении эквивалентных преобразований распределенного отношения R, т. е. замене подмножества его пар другим эквивалентным подмножеством, значения функции *Nodes*() не играют определяющей роли. Важно лишь, что исключаемые и добавляемые пары (A, B) имеют непустой атрибут $Nodes(A, B)$.

Переходим к обсуждению вопросов оптимизации распределенных LP-структур. С этой целью наряду с функцией *Nodes*() введем еще два полезных отображения, действующих из $\mathbb{F}$ в $\mathbb{N}$. Пусть выбран произвольный элемент $A \in \mathbb{F}$. Поскольку решетка $\mathbb{F}$ является атомно-порожденной, существует представление $A = \bigcup_i a_i$ в виде объединения атомов. Обозначим

$$NodesMeet(A) = \bigcap_i Nodes(a_i);$$

$$NodesJoin(A) = \bigcup_i Nodes(a_i).$$

Возвращаясь к терминологии предметной области, можно заметить, что отображение *NodesMeet*() определяет вычислительные узлы, каждый из которых содержит все элементарные факты, порождающие множество фактов *A*. При этом отображение *NodesJoin*() выдает все узлы, хранящие хотя бы один из таких фактов.

Возникает вопрос: если точное значение непустого атрибута *Nodes*(*A*, *B*) несущественно для построения логического замыкания, то какое же окажется предпочтительным? Ответ дает стратегия распределенного логического вывода. Как отмечалось выше, одна из целей моделирования распределенных логических систем — обоснование снижения трафика между узлами. Вместе с тем нецелесообразно излишнее дублирование информации на различных узлах.

Пусть на некотором узле работает локальная машина вывода, оперирующая в основном фактами этого узла. Для нее крайне затратным окажется поиск правил вывода собственных фактов на других узлах сети. Ведь априори неизвестно, сколько узлов предстоит просмотреть (или запросить) и в каком порядке. Поэтому будем считать допустимым и даже целесообразным хранение на узле всех правил, в которых используются факты этого же узла. Таким образом, приходим к следующим понятиям, характеризующим "хорошее" распределение правил по узлам вычислительной сети.

Определение 2.4. Распределенное бинарное отношение *R* на распределенной решетке $\mathbb{F}$ называется:

- *релевантно-корректным*, если для любой $(A, B) \in R$ справедливо

$$Nodes(A, B) \supseteq NodesJoin(A) \cup NodesJoin(B);$$

- *релевантно-нормализованным*, если

$$Nodes(A, B) = NodesJoin(A) \cup NodesJoin(B).$$

Очевидно, что последнее соотношение более точно соответствует рассматриваемой стратегии вывода. Первое же из них допускает хранение на узле таких правил, которые не связаны ни с одним фактом этого же узла. Однако нельзя утверждать, что подобные правила избыточны, поскольку они могут оказаться востребованными на других узлах и уникальными в рамках всей распределенной системы. В этом случае их целесообразно перенести на другие узлы, скорректировав атрибут *Nodes*().

Определение 2.5. Распределенное отношение *R* на распределенной решетке $\mathbb{F}$ называется каноническим, если оно релевантно-нормализовано и задано множеством пар вида (*A*, *a*), где $A \in \mathbb{F}$, *a* — атом в $\mathbb{F}$.

На основе утверждения 1.1 и результатов настоящего раздела выводится следующий факт.

Утверждение 2.1. Для любого распределенного отношения на распределенной решетке существует эквивалентное ему каноническое отношение.

Схема доказательства данного утверждения аналогична двум предыдущим. К исходному отношению *R* применяем утверждение 1.1, затем строим два замыкания и помечаем каждую пару максимальным атрибутом (*N*), в итоге приходим к требуемому результату.

Далее рассмотрим вопрос об эквивалентной минимизации распределенных бинарных отношениий.

Определение 2.6. *Логической редукцией* распределенного отношения *R* на распределенной решетке называется любое минимальное отношение, эквивалентное *R* и при этом релевантно-нормализованое.

Заметим, что множество всех отношений на общей решетке является частично упорядоченным, поэтому различаются понятия минимального и наименьшего отношений [16]. В определении 2.6 речь идет именно о минимальном отношении, т. е. оно может оказаться не единственным. Под минимальным эквивалентным отношением подразумевается такое, что исключение из него любой единственной пары приведет к неэквивалентному отношению. Таким образом, рассматривается оптимизация LP-структуры в некотором локально-минимальном смысле.

Как и при обосновании предыдущих утверждений, для построения логической редукции достаточно предварительно построить обычную логическую редукцию (в смысле теоремы 1.3), после чего пометить каждую пару (*A*, *B*) полученного отношения необходимым атрибутом, в данном случае $Nodes(A, B) = NodesJoin(A) \cup NodesJoin(B)$.

Таким образом, справедлива следующая важная теорема.

Теорема 2.3. Для произвольного распределенного бинарного отношения *R* на распределенной решетке существует логическая редукция.

В завершение представленного исследования рассмотрим кратко вопросы о деталях построения логического замыкания и логической редукции заданного распределенного бинарного отношения *R*.

В соответствии с работой [9], для распределенного бинарного отношения *R* на распределенной решетке $\mathbb{F}$ рассмотрим отношение $\tilde{R}$, построенное по *R* последовательным выполнением следующих действий (шагов):

- добавить к *R* все пары вида (*A*, *A*), где $A \in \mathbb{F}$ (рефлексивные пары), и обозначить новое отношение R_1;
- добавить к R_1 всевозможные пары (*A*, *B*) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i, B_i) принадлежат R_1;

• объединить полученное отношение с отношением включения $\supseteq$.

• все добавленные пары (A, B) результирующего отношения пометить атрибутом

$$Nodes(A,B) = NodesJoin(A) \cup NodesJoin(B).$$

Как и в работе [9], нетрудно показать, что отношение $\tilde{R}$ логически эквивалентно R. Более того, справедлива следующая теорема, доказываемая по обычной для настоящей статьи схеме.

Теорема 2.4. Для произвольного распределенного отношения R на распределенной решетке $\mathbb{F}$ логическое замыкание $\xrightarrow{R}$ представляет собой транзитивное замыкание $\tilde{R}^*$ соответствующего отношения $\tilde{R}$.

Для распределенного отношения R рассмотрим отношение $\underset{\sim}{R}$, построенное по данному R последовательным выполнением шагов, в некотором роде обратных построению $\tilde{R}$, а именно:

• исключить из R содержащиеся в нем пары вида $A \supset B$ и обозначить новое отношение R_{-1};

• исключить из R_{-1} всевозможные пары (A, B) с элементами вида $A = \bigcup_i A_i$, $B = \bigcup_i B_i$, где все (A_i, B_i) принадлежат R_{-1} и не совпадают с (A, B);

• исключить из полученного отношения все рефлексивные пары.

Можно показать, что отношение $\underset{\sim}{R}$ логически эквивалентно R.

Следующая теорема указывает достаточное условие существования и способ построения логической редукции данного распределенного отношения.

Теорема 2.5. Пусть для распределенного отношения R, заданного на распределенной решетке $\mathbb{F}$, построено соответствующее отношение $\tilde{R}$ (см. выше). Тогда, если для $\tilde{R}$ существует транзитивная редукция R^0, то соответствующее ей отношение $\underset{\sim}{R}^0$ представляет собой логическую редукцию исходного отношения R.

Ценность последних двух теорем состоит в том, что они сводят задачи построения логического замыкания и логической редукции распределенных отношений к хорошо известным аналогичным задачам для транзитивного замыкания и транзитивной редукции.

Заключение

В настоящей работе определена и исследована обобщенная LP-структура, расширяющая область применения этой алгебраической теории до распределенных интеллектуальных систем продукционного типа.

Для нового понятия распределенной LP-структуры рассмотрен стандартный круг вопросов, а именно — о замыкании, об эквивалентных преобразованиях, о канонической форме, о логической редукции. Представленные теоремы имеют несложные доказательства, однако создают теоретическую основу для дальнейших исследований, в том числе для оптимизации распределенного логического вывода.

Полученные результаты позволяют использовать преимущества теории LP-структур при построении и исследовании распределенных интеллектуальных систем продукционного типа в различных предметных областях.

Поскольку многие модели в информатике имеют продукционный характер, дальнейшие исследования на рассматриваемом направлении могут быть также связаны с LP-структурами более сложных типов и соответствующих им предметных областей переносом концепций настоящей работы на эти модели. Общие методы исследования при этом останутся прежними.

Работа поддержана грантом РФФИ № 15-07-05341.

Список литературы

1. **Интеллектуальная** система тематического исследования научно-технической информации ("ИСТИНА") / Под ред. В. А. Садовничего. М.: Изд-во МГУ, 2014. 262 с.

2. **Сошников Д. В.** Методы и средства построения распределенных интеллектуальных систем на основе продукционно-фреймового представления знаний: дис. ... канд. физ.-мат. наук. М., 2002. 190 с.

3. **Nwana H. S.** Software Agents: An Overview // Knowledge Engineering Review. 1996. Vol. 11, N 3. P. 1—40.

4. **Sycara K., Decker K., Pannu A.** et al. Distributed Intelligent Agents // IEEE Expert. 1996. December. P. 1—32.

5. **Васенин В. А., Макаров В. Л., Бахтизин А. Р.** и др. Средства суперкомпьютерных систем для работы с агент-ориентированными моделями // Программная инженерия. 2011. № 3. С. 2—14.

6. **Дубинин В. Н.** Организация и проектирование интеллектуальных распределенных вычислительных систем с групповыми взаимодействиями // Вычислительная техника в автоматизированных системах контроля и управления: Межвуз. сб. науч. тр. Пенза: ПГУ, 1999. Вып. 26. С. 31—38.

7. **Гинкул Г. П., Соловьев С. Ю.** Алгоритм цитирования для построения гибридных выводов в продукционных системах // Программные системы и инструменты. 2013. Т. 14. С. 172—175.

8. **Ishida T.** Parallel, Distributed and Multiagent Production Systems // Proceedings of the First International Conference on Multiagent Systems. AAAI, 1995. P. 416—422.

9. **Махортов С. Д**. Математические основы искусственного интеллекта: теория LP-структур для построения и исследования моделей знаний продукционного типа. М.: Изд-во МЦНМО, 2009. 304 с.

10. **Болотова С. Ю., Махортов С. Д.** Алгоритмы релевантного обратного вывода, основанные на решении продукционно-логических уравнений // Искусственный интеллект и принятие решений. 2011. № 2. С. 40—50.

11. **Болотова С. Ю.** Реализация многопоточности в релевантном LP-выводе // Программная инженерия. 2014. № 1. С. 12—18.

12. **Махортов С. Д.** Интегрированная среда логического программирования LPExpert // Информационные технологии. 2009. № 12. С. 65—66.

13. **Махортов С. Д.** Основанный на решетках подход к исследованию и оптимизации множества правил условной системы переписывания термов // Интеллектуальные системы. 2009. Т. 13. С. 51—68.

14. **Махортов С. Д.** LP-структуры для обоснования и автоматизации рефакторинга в объектно-ориентированном программировании // Программная инженерия. 2010. № 2. С. 15—21.

15. **Aho A. V., Garey M. R., Ulman J. D.** The transitive reduction of a directed graph // SIAM J. 1972. Computing 1 : 2. P. 131—137.

16. **Биркгоф Г**. Теория решеток: пер. с англ. М.: Наука, 1984. 568 с.

17. **Sowyer B., Foster D.** Programming Expert Systems in Pascal. John Wiley & Sons, Inc., 1986. 186 p.

18. **Таненбаум Э., Ван Стеен М.** Распределенные системы. Принципы и парадигмы: пер. с англ. СПб.: Питер, 2003. 877 с.

S. D. Makhortov, Associate Professor, Head of Department, e-mail: sd@expert.vrn.ru,
Voronezh State University

The Algebraic Model of the Distributed Logical System of the Production Type

The most common approach to the construction and research of the distributed artificial intelligence is the multi-agent systems concept, but at the same time there are other approaches that have their own advantages in solving the specific problems associated with distributed intelligent systems.

In the last decade the author and his students obtained a number of important results in the field of knowledge management that are associated with the production logic systems that became very widespread in the computer science. They developed an algebraic theory of the LP structures, which provides the justification and the effective solution to the problems of the equivalent transformations, verification, knowledge bases minimization and speeding-up the logical inference using the parallel computing. It is implemented in a number of software products which demonstrate its practical significance. This theory also can be used to describe the other systems, that were not considered as production system before.

The article describes the next step of the LP structures theory development in context of the fundamental algebraic approach for the construction of the extandable range of the production systems and similar to them. For the corresponding knowledge bases this step consists of moving from the parallel processing to the distributed one. The paper formulates the extended algebraic model which has a number of significant capabilities that cover the distributed production-logical systems. As a result, the LP structure theory advantages become available in the such systems construction and research.

Keywords: *distributed knowledge base, production system, LP structure, logical closure, equivalent transformation, logical reduction*

References

1. ***Intellektual'naya*** *sistema tematicheskogo issledovaniya nauchno-tekhnicheskoi informatsii ("ISTINA")* (Intellectual system of thematic review and scientific and technical information (ISTINA)). Ed. by V. A. Sadovnichiy, Moscow, MSU, 2014, 262 p. (in Russian).

2. **Soshnikov D. V.** *Metody i sredstva postroeniya raspredelennykh intellektual'nykh sistem na osnove produktsionno-freimovogo predstavleniya znanii* (Methods and tools for building distributed intelligent systems on the production-frame knowledge representation basis), Moscow, MAI, 2002, 190 p. (in Russian).

3. **Nwana H. S.** Software Agents: An Overview. *Knowledge Engineering Review*, 1996, vol. 11, no. 3, pp. 1—40.

4. **Sycara K., Decker K., Pannu A., Williamson M., Zeng D**. Distributed Intelligent Agents, *IEEE Expert*, 1996, December, pp. 1—32.

5. **Vasenin V. A., Makarov V. L., Bakhtizin A. R., Roganov V. A., Trifonov I. A.** Sredstva superkomp'yuternykh sistem dlya raboty s agent-orientirovannymi modelyami (Supercomputer system tools for working with an agent-based model), *Programmnaya Ingeneria,* 2011, no. 3, pp. 2—14 (in Russian).

6. **Dubinin V. N.** Organizatsiya i proektirovanie intellektual'nykh raspredelennykh vychislitel'nykh sistem s gruppovymi vzaimodeistviyami (Organization and design of the intelligent distributed computing systems with group interactions). *Vyichislitelnaya tehnika v avtomatizirovannyih sistemah kontrolya i upravleniya*: Mezhvuzovskiy sbornik nauchnyih trudov, Penza, PGU, 1999, no. 26, pp. 31—38 (in Russian).

7. **Ginkul G. P., Solov'ev S. Yu.** Algoritm tsitirovaniya dlya postroeniya gibridnykh vyvodov v produktsionnykh sistemakh (Quoting algorithm for constructing hybrid inferences in production systems), *Programmnyie Sistemyi i Instrumentyi,* 2013, vol. 14, pp. 172—175 (in Russian).

8. **Ishida T.** Parallel, Distributed and Multiagent Production Systems, *Proceedings of the First International Conference on Multiagent Systems. AAAI*, 1995, pp. 416—422.

9. **Makhortov S. D.** *Matematicheskie osnovy iskusstvennogo intellekta: teoriya LP-struktur dlya postroeniya i issledovaniya modelei znanii produktsionnogo tipa* (Mathematical Foundations of Artificial Intelligence: The LP structures theory for the knowledge models of production type construction and research). Ed. by V. A. Vasenin, Moscow, MCCME, 2009. 304 p. (in Russian).

10. **Bolotova S. Yu., Makhortov S. D.** Algoritmy relevantnogo obratnogo vyvoda, osnovannye na reshenii produktsionno-logicheskikh uravnenii (Algorithms of the relevant backward inference that is based on production-logic equations solving), *Iskusstvennyiy Intellekt i Prinyatie Resheniy,* 2011, no. 2, pp. 40—50 (in Russian).

11. **Bolotova S. Yu.** Realizatsiya mnogopotochnosti v relevantnom LP-vyvode (The multi-threading relevant LP-inference implementation), *Programmnaya Ingeneria*, 2014, no. 1, pp. 12—18 (in Russian).

12. **Makhortov S. D.** Integrirovannaya sreda logicheskogo programmirovaniya LPExpert (LPExpert: Integrated development environment for logical programming), *Informacz*ionny'e Texnologii,* 2009, no. 12, pp. 65—66 (in Russian).

13. **Makhortov S. D.** Osnovannyi na reshetkakh podkhod k issledovaniyu i optimizatsii mnozhestva pravil uslovnoi sistemy perepisyvaniya termov (Research and optimization of a set of rules in a conditional term rewriting system using approach based on the lattices), *Intellektualnyie Sistemyi*, 2009, no. 13, pp. 51—68 (in Russian).

14. **Makhortov S. D.** LP-struktury dlya obosnovaniya i avtomatizatsii refaktoringa v ob"ektno-orientirovannom programmirovanii (Using LP-structures for substantiation and automatization of refactoring in object-oriented programming), *Programmnaya Ingeneria,* 2010, no. 2, pp. 15—21 (in Russian).

15. **Aho A. V., Garey M. R., Ulman J. D.** The transitive reduction of a directed graph. *SIAM J.*, 1972, Computing 1 : 2, pp. 131—137.

16. **Birkhoff G.** *Lattice Theory.* Rhode Island, 1995, 420 p.

17. **Sowyer B., Foster D.** *Programming Expert Systems in Pascal.* John Wiley & Sons, Inc., 1986, 186 p.

18. **Tanenbaum A., Van Steen M.** *Distributed Systems: Principles and Paradigms.* Pearson, 2007, 704 p.

УДК 004.41

А. М. Иванчик, аспирант, вед. инженер-программист, e-mail: antonishka@gmail.com,
Морской гидрофизический институт РАН, г. Севастополь

Архитектурно-технологические решения автоматической системы валидации результатов диагноза и прогноза состояния Черного моря

Представлены архитектурные и технологические решения, положенные в основу автоматической системы валидации данных прогноза, полученных в результате использования модели MHIC2, разработанной в Морском гидрофизическом институте РАН. Модель и система на ее основе предназначены для реализации процедур диагностирования и прогнозирования гидрологических полей Черного моря — уровня морской поверхности, температуры поверхности моря, солености морской воды, скорости течений.

Ключевые слова: *автоматические системы, валидация, морской прогноз, Черное море, аномалии уровня моря, температура морской поверхности, соленость, средства разработки автоматических систем, методы контроля работы программ*

Введение

Одним из актуальных направлений контроля состояния и использования ресурсов Черного моря является комплексный мониторинг его гидрофизических полей. В настоящее время в Морском гидрофизическом институте (МГИ) РАН в оперативном режиме реализации процессов мониторинга работает многокомпонентная наблюдательная система Черного моря. При разработке подобных систем одной из наиболее важных задач является оценка достоверности результатов моделирования гидрофизических полей. Для ее решения в составе разработанной системы функционирует подсистема валидации (оценки достоверности) результатов расчетов, выполненных с использованием модели MHIC2.

Вихреразрешающая физическая модель Черного моря MHIC2 разработана в Морском гидрофизическом институте [1]. На ее основе выполняются диагноз и прогноз уровня морской поверхности, температуры и солености морской воды, а также скоростей течений. Модель MHIC2 функционирует в составе автоматической системы диагноза и прогноза гидрологических полей Черного моря [2]. Система проводит расчеты ежедневно. Работа системы завершается сохранением в файловом архиве отделения Оперативной океанографии МГИ NetCDF-файлов [3], содержащих результаты расчетов мгновенных значений полей температуры, солености, уровня моря и скоростей течений морской воды, с шагом по времени три часа в интервале от четырех дней до наступления текущей даты и на пять дней после нее. Результаты расчетов выкладываются на THREDDS-сервер [4] и доступны в сети Интернет по адресу http://mis.bsmfc.net:8080/thredds/catalog.html.

Валидация результатов прогноза термохалинных полей и уровня морской поверхности выполняется на основе сопоставления данных модельных расчетов с in situ (данными контактных измерений) и спутниковыми измерениями. Спутниковые и контактные данные поступают из соответствующих тематических центров подготовки данных (TAC — *the Thematic Assembly Center*, в терминологии Copernicus, центры, в обязанности которых входит предоставлять данные спутниковых и in situ наблюдений) системы Европейских центров морских прогнозов Copernicus (http://marine.copernicus.eu/). Затем эти данные сохраняют в файловых архивах, доступных в локальной сети [5], и используют в системе валидации.

В настоящей статье описаны принципы построения, функционирования, структура автоматической системы валидации, разработанной на основе ранее существовавшей системы, которая работала в интерактивном режиме [6]. Необходимость перехода к системе, которая могла бы работать в автоматическом режиме, обусловлена тем обстоятельством, что традиционные системы, работающие под управлением операторов, имеют низкий уровень оперативного реагирования, надежности и требуют больших затрат времени на их сопровождение и обслуживание.

Рассматриваемая автоматическая система имеет иерархическую организацию. Она состоит из заданий-менеджеров, управляющих вычислитель-

ным процессом, и программ-исполнителей, реализующих отдельные операции обработки данных. Задания — это программы, написанные на языке CalcManPack [7], которые непосредственно реализуют определенный алгоритм управления или его составную часть. Задания записываются в бинарные файлы и содержат описание заданий и набор команд. Разработка и выполнение заданий проводятся в среде комплекса компьютерных программ CalcMan [8, 9]. Комплекс программ CalcMan предназначен для проектирования и отладки программных систем управления вычислительными процессами и их автоматизации, например, для автоматизации систем диагноза и прогноза состояния морской среды [10] и контроля их работы. Интерпретируемый командный язык CalcManPack пакета CalcMan предназначен для разработки и выполнения заданий, управляющих информационно-вычислительными процессами.

Автоматической системой валидации выполняется оценка достоверности результатов диагностики и прогнозирования ежедневно для каждого цикла расчетов диагноза и прогноза состояния Черного моря. Для выполнения отдельных операций обработки данных используют программы-исполнители, написанные на языке программирования R.

Требования к автоматической системе валидации

Автоматическая система валидации должна работать в непрерывном режиме. В связи с этим требованием должны быть предусмотрены процедуры автоматического запуска подсистем валидации по различным параметрам, контроля состояния работоспособности системы и полученных с ее помощью результатов.

Для обеспечения непрерывного режима работы системы необходим постоянный контроль результатов выполнения всех процедур, включая автоматическое обнаружение ошибок, их анализ и принятие решений о продолжении работы отдельных подсистем и системы в целом. Критические ошибки, приводящие к потере работоспособности системы или ее подсистем, требуют обязательного вмешательства оператора. Ошибки некритического характера, например, отсутствие данных спутниковых и контактных измерений и ошибки в них, зачастую не требуют вмешательства оператора, так как появление данных может задерживаться по объективным причинам, валидация при этом будет выполнена после их появления.

Перед построением системы необходимо было ответить на следующие вопросы, которые определяют требования к системе автоматической валидации:

1) как часто должны происходить запуски системы валидации;

2) как определить время запуска системы валидации, которое зависит от времени появления данных после завершения процедуры моделирования;

3) какие наборы данных необходимы для выполнения процедуры валидации результатов прогнозирования термохалинных полей и уровня моря, полученных в результате работы модели MHIC2;

4) как согласовать порядок запуска процедур валидации результатов прогноза термохалинных полей и уровня моря, полученных в результате работы модели MHIC2;

5) какой должна быть структура директорий, содержащих входные данные и данные промежуточных расчетов;

6) какой должна быть структура директорий, содержащих результаты работы системы;

7) какие результаты должны быть доступны дистанционно, а какие — локально;

8) какими должны быть форматы данных для выгрузки в локальные архивы данных и на HTTP-сайт;

9) какие процедуры контроля необходимо осуществлять при проектировании системы, какие — при ее работе;

10) какие журналы работы системы должны быть доступны локально, какие необходимо сделать доступными для удаленного контроля.

Требования были сформулированы, и в соответствии с ними была разработана система.

Архитектура и основные подсистемы автоматической системы валидации

На рис. 1 приведена архитектура автоматической системы валидации.

Входными данными для автоматической системы валидации являются данные, полученные в результате расчета прогноза гидрологических полей Черного моря с помощью модели MHIC2, и данные измерений.

Расчет прогноза гидрологических полей (температуры, солености, уровня моря и скоростей течений морской воды) проводится ежедневно. Данные спутниковых измерений поступают, как правило, ежесуточно, с задержкой по времени на 1 сут. Контактные данные поступают с большей задержкой. Проверка достоверности результатов расчетов с использованием системы валидации проводится ежедневно. При этом время запуска процедур валидации назначается на время, когда результаты работы прогностической системы уже сохранены в локальном архиве.

Для выполнения процедур валидации результатов прогнозирования термохалинных полей и уровня моря используют данные измерений, сохраненные в файловых архивах, доступных по локальной сети. Файловые архивы ежедневно пополняются данными, получаемыми с сайта системы Европейских центров морских прогнозов Copernicus http://marine.copernicus.eu/. Используемые данные:

- спутниковые данные температуры поверхности моря SST L3 Multisensor (Ifremer, Франция, Copernicus OSI TAC, Sea Surface Temperature Level 3 Product) — основной источник данных по температуре поверхности моря;

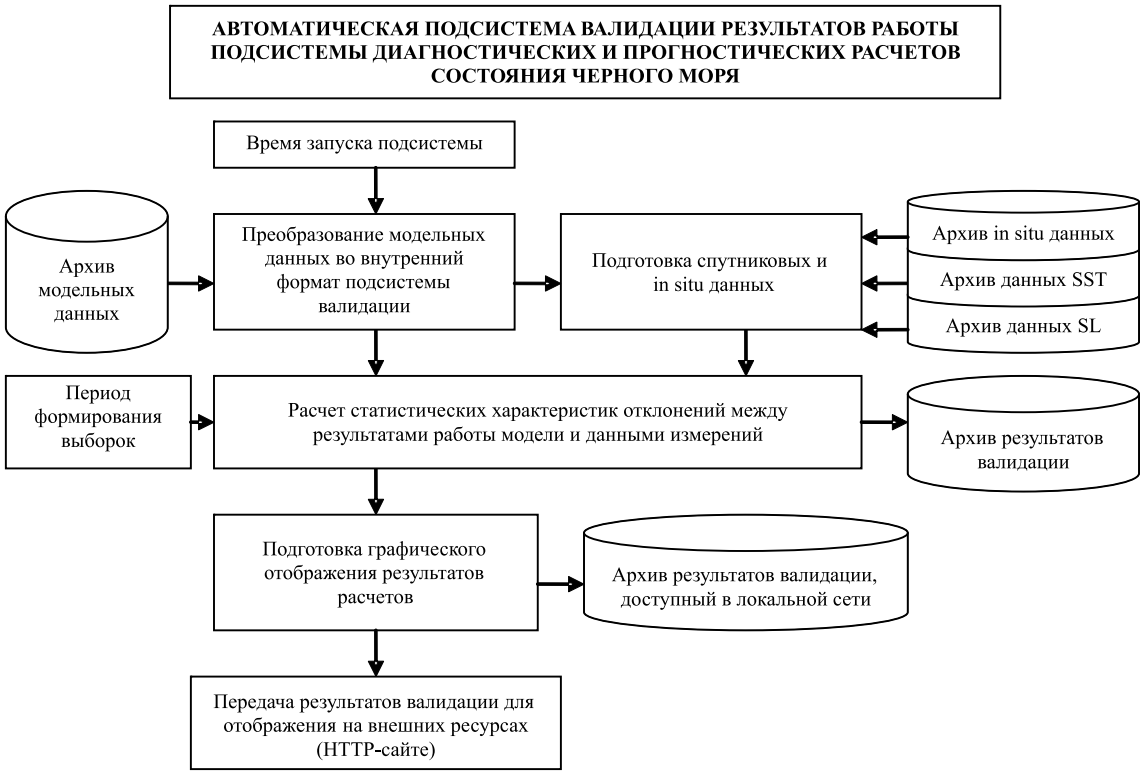


Рис. 1. Архитектура автоматической системы валидации результатов диагноза и прогноза состояния Черного моря

• спутниковые данные температуры поверхности моря SST L3S Merged-Multisensor (The National Research Council (CNR), Италия, Copernicus OSI TAC, Sea Surface Temperature Level 3 Product) — резервный источник данных по температуре поверхности моря;

• контактные данные буев ARGO (Ifremer, Франция, Copernicus INS TAC);

• данные аномалий уровня морской поверхности (CLS, Франция, Copernicus SL TAC, Sea Level).

Автоматическая система валидации построена по модульному принципу. Благодаря такому подходу, при ошибках в работе какой-либо подсистемы остальные части системы продолжают работу, что способствует повышению надежности функционирования системы в целом.

На рис. 2 показана автоматическая система валидации, отображаемая в программе CMStudio пакета программ CalcMan, которая используется для построения и отладки системы. На рис. 2 представлена структура системы и списки заданий, управляющих ее работой и работой всех подсистем, выполняющих отдельные этапы расчетов, операции копирования и преобразования данных, загрузки входных данных и выгрузки результатов расчетов, а также операций обслуживания системы.

Система включает в себя три подсистемы:

♦ подсистему валидации по данным температуры поверхности моря с поверхностным разрешением 2 км SST L3 Multisensor (Copernicus OSI TAC);

♦ подсистему валидации по данным буев ARGO (Copernicus INS TAC);

♦ подсистему валидации по данным аномалий уровня морской поверхности (Copernicus SL TAC).

Работа автоматической системы валидации

Система состоит из заданий, написанных на языке CalcManPack, которые выполняют требуемые операции подготовки инфраструктуры, манипуляции с файлами входных и выходных данных, запуск программ, написанных на языке программирования R, входящих в состав системы.

Работой системы управляет главное задание-менеджер. Задание работает в циклическом режиме (время запуска задается в параметрах настройки системы), ежесуточно последовательно запуская на выполнение задания-менеджеры подсистем, анализирует результаты их выполнения, автоматически ведет журналы выполнения заданий, отдельный журнал на каждый цикл работы. Журналы работы системы доступны для удаленного контроля.

Каждый цикл автоматической работы подсистемы включает следующие операции.

1. Вызов задания, которое управляет процессом преобразования прогностических данных во внутренний формат системы валидации. При наличии данных прогноза и успешном их преобразовании

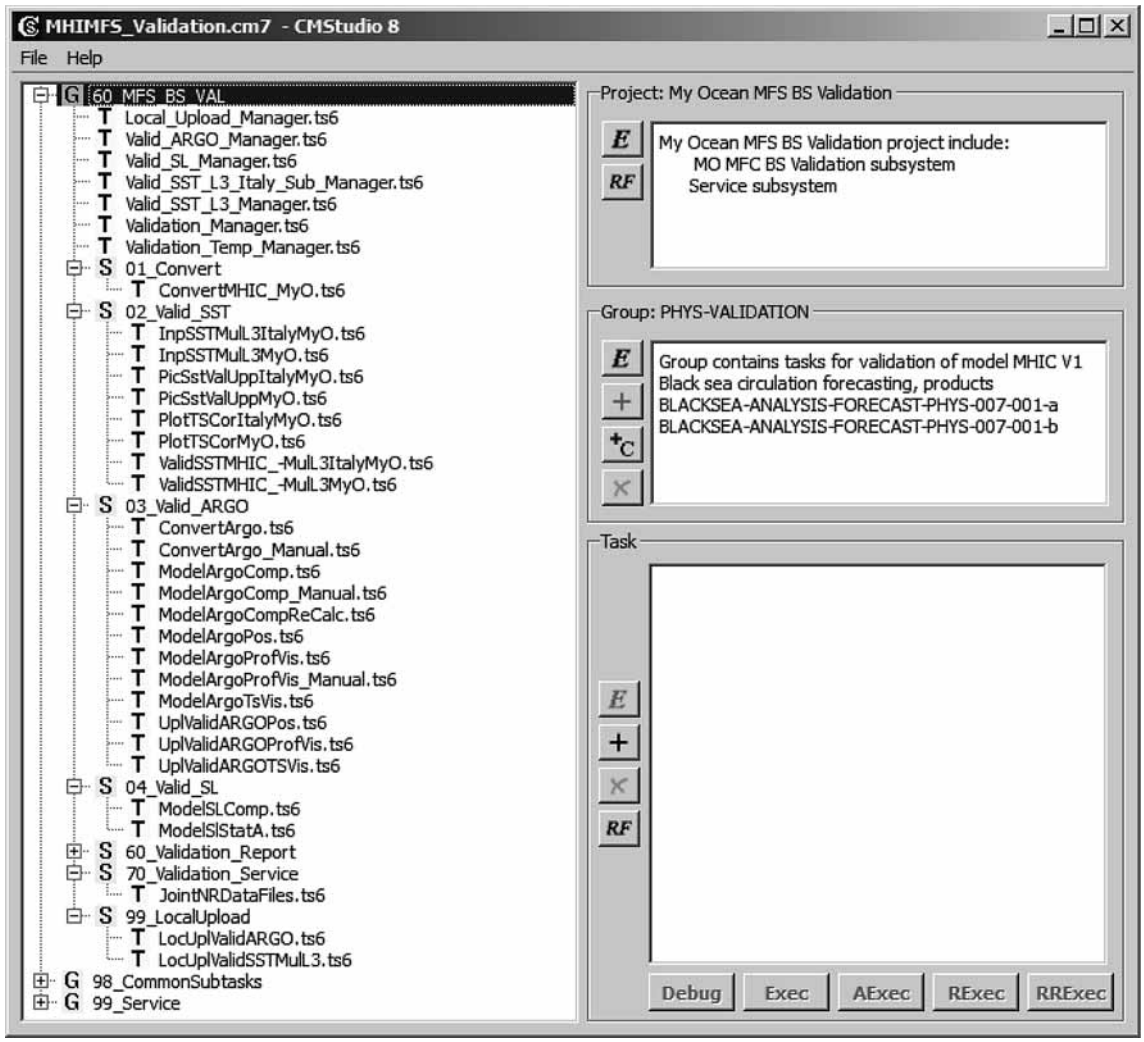


Рис. 2. Структура автоматической системы валидации, отображаемая в программе CMStudio пакета программ CalcMan

менеджером работы системы запускаются менеджеры валидации. При отсутствии новых данных система переходит к ожиданию следующего цикла работы (следующие сутки). Данные прогноза, преобразованные в формат системы валидации, хранятся в архиве на локальном компьютере. При этом предусмотрена процедура поддержания размера архива данных за срок не более 90 дней. Данный срок может быть изменен при настройке параметров системы и выбран исходя из актуальности результатов расчета статистических характеристик и размера архива преобразованных данных прогноза, так как данные имеют объем 1,3 Гбайт за один день прогноза.

2. Вызов задания-менеджера, проводящего валидацию по данным температуры поверхности моря SST L3 Multisensor (Франция).

3. Вызов задания-менеджера, проводящего валидацию по данным температуры поверхности моря SST L3 Multisensor (Италия).

4. Вызов задания-менеджера, проводящего валидацию по данным буев ARGO.

5. Вызов задания-менеджера, проводящего валидацию по данным аномалий уровня морской поверхности SL.

6. Вызов задания-менеджера, проводящего выгрузку всех результатов валидации в файловый архив в виде файлов сжатых данных за текущую дату обработки.

На этом выполнение текущего цикла обработки завершается. Система переходит в состояние ожидания времени запуска на следующем цикле работы.

Запуск процедур валидации по каждому виду данных измерений проводится вне зависимости от наличия других данных измерений, но при наличии результатов, полученных в результате прогнозирования. Это позволяет не прерывать работу системы валидации при возникновении ошибок в какой-либо подсистеме.

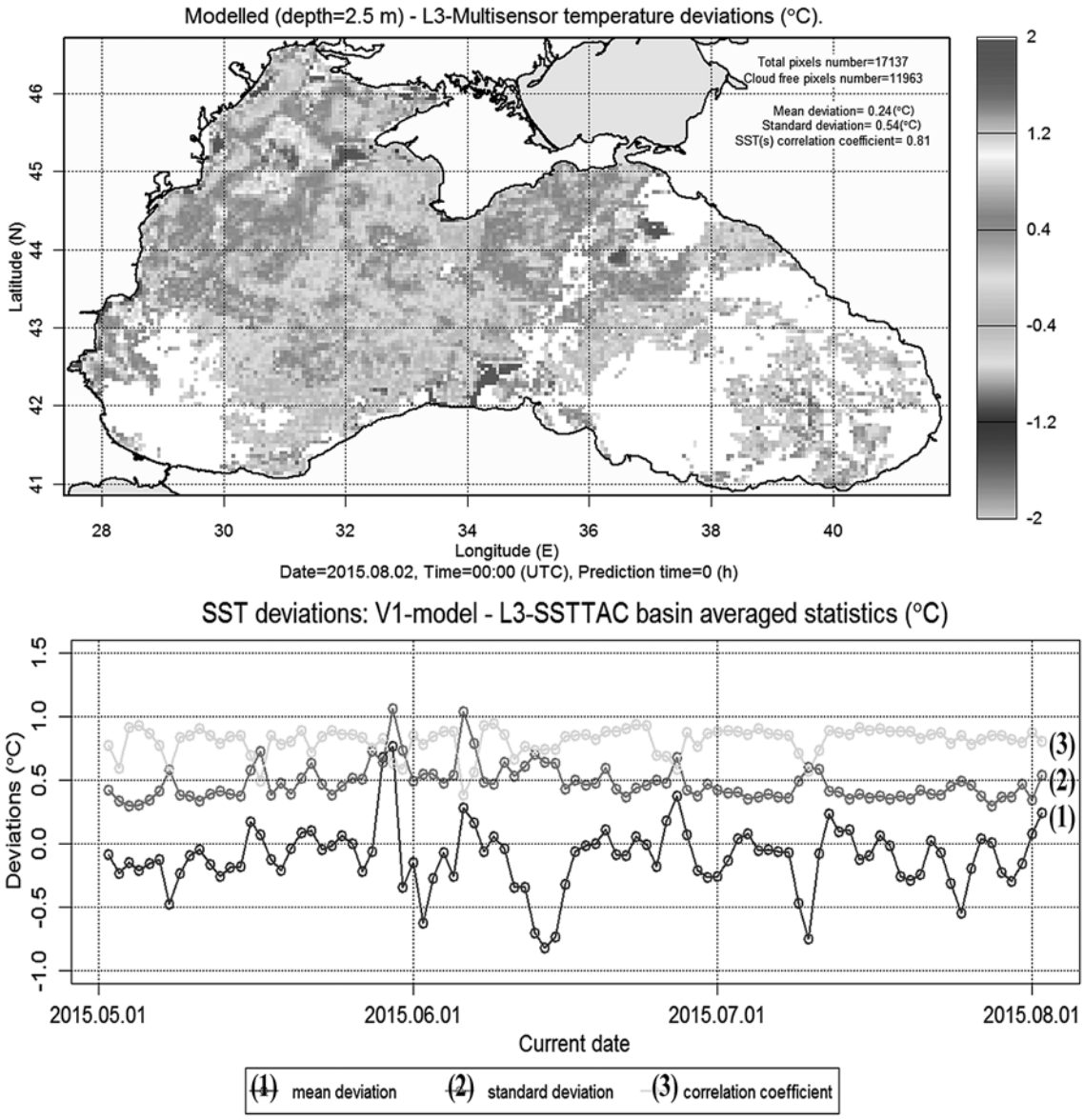


Рис. 3. Пример результата работы автоматической системы валидации по спутниковым данным температуры поверхности моря

Каждое из заданий-менеджеров, управляющих процедурой валидации по определенным данным, проводит:

- подготовку спутниковых или контактных данных, получаемых из внутренних архивов, для осуществления процедур валидации (для этого используются программы, написанные на языке R);
- подготовку данных для расчета статистических характеристик отклонений данных прогноза, полученных в результате работы модели, и спутниковых и контактных данных, управление процедурами расчета статистических характеристик;
- подготовку графических файлов для визуализации результатов расчета статистических характеристик;
- передачу данных для отображения на HTTP-сайте.

Для подготовки входных данных и расчета статистических характеристик используют программы-исполнители, написанные на языке программирования R, которые вызываются из заданий. С их помощью рассчитываются среднебассейновый коэффициент корреляции, среднее и среднеквадратичное отклонения между рассчитанными с помощью модели прогностическими данными и данными измерений для фиксированных моментов времени, а также строятся соответствующие таблицы и графики.

Каждый R-скрипт и каждое задание системы имеют строго определенные описанные коды завершения. Все задания системы анализируют коды

выполнения вызываемых R-скриптов, а также заданий-подпрограмм. В зависимости от результатов анализа выполнения планируют дальнейшие действия.

После проведения процедур валидации осуществляется запуск менеджера копирования архивов результатов в файловые архивы, расположенные в локальной сети. Данные расчетов хранятся в виде zip-архивов, каждый файл которых содержит результаты статистических расчетов и графические данные за одну дату.

Для отображения на сайте подготавливаются только графические изображения и координаты буев (при проведении процедуры валидации по данным буев ARGO).

Результаты работы автоматической системы валидации на HTTP-сайте http://bsmfc.net/ представляются в виде карт и графиков, пример которых показан на рис. 3.

При работе системы программами-интерпретаторами заданий автоматически создаются журналы работы всех выполняемых заданий системы, файл генерируется для каждого цикла работы каждого запускаемого задания. Эти журналы доступны только локально и используются для поиска причин возникновения критических ситуаций и для их устранения. Фрагмент такого журнала приведен на рис. 4.

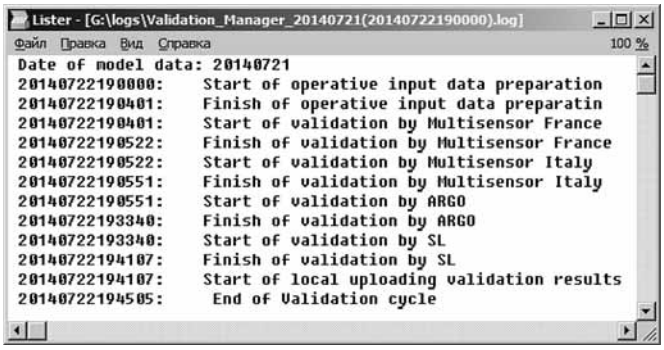
```
Lister - [G:\Antonina\WorkCM_Validation\MHIMFS\60_MFS_BS_VAL\04_Valid_SL\ModelSlStatA_2014-06-09...
Файл  Правка  Вид  Справка                                                          100 %
   G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Product1\Instan
   G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Product1\Instan
   G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Product1\Instan
      Copy file "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
             to "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
      Copy file "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
             to "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
      Copy file "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
             to "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
      Copy file "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
             to "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
Command had been completed and
returns code = 4
Load Cmd # 15 "Do Prog and Wait"
   D:\R-2.12.0\bin\i386\Rscript.exe
   G:\Antonina\WorkCM_Validation\MHIMFS\ValidationScripts\ValidationSL\ModeAltJoinScr
   G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Product1\Instan
Command had been completed and
returns code = 0
Load Cmd # 16 "Switch_R"
   0:ErrSuc
   -1:ErrComStr
   -2:ErrDataProc
   -4:ErrWrtStatA
Command had been completed and
returns code = 0
Load Cmd # 24   ErrSuc: "CopyFile"
   G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Product1\Instan
   G:\Antonina\ftpv1\pictures\validation\SL\StatA.txt
      Copy file "G:\Antonina\WorkValidation\20_Validations_Operative\ResultantData\Pr
             to "G:\Antonina\ftpv1\pictures\validation\SL\StatA.txt"
Command had been completed and
returns code = 1
Load Cmd # 25 "Interrupt Task"
   0
Command had been completed and
returns code = 0
Task level 0 "G:\Antonina\WorkCM_Validation\MHIMFS\60_MFS_BS_VAL\04_Valid_SL\ModelS1S
had been executed with exit code = 0
Load Task Level 0  Name: "Joint StatA validation files"
   File: "G:\Antonina\WorkCM_Validation\MHIMFS\60_MFS_BS_VAL\04_Valid_SL\ModelS1Stat
Stop task time: 2014-06-09-07-10-47
Exit code = 0
```

Рис. 4. Фрагмент журнала работы, автоматически созданного программой-интерпретатором задания

В заданиях-менеджерах системы и подсистем запрограммировано создание журналов работы системы и подсистем, доступных локально и дистанционно. В них отображаются основные этапы работы системы и сообщения о критических ошибках или об отсутствии данных. Формат данных журналов работ произволен и определяется только решениями разработчика системы и желательными точками контроля выполнения работы.

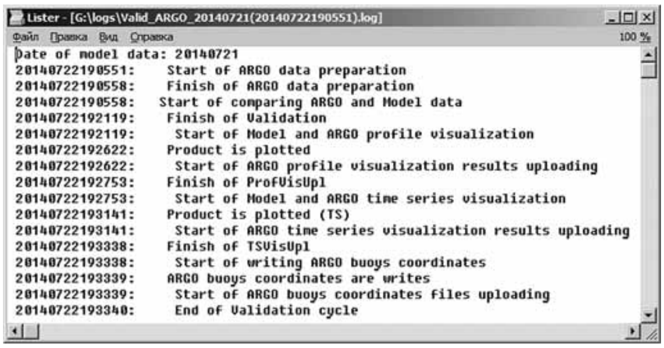
Доступные удаленно журналы работы позволяют контролировать происходящее в системе в любой момент времени из любого места, что ограничивается лишь наличием доступа в сеть Интернет.

На рис. 5 и 6 приведены примеры журналов работы автоматической системы валидации, доступных удаленно.



```
Lister - [G:\logs\Validation_Manager_20140721(20140722190000).log]
Файл  Правка  Вид  Справка                                        100 %
Date of model data: 20140721
20140722190000:    Start of operative input data preparation
20140722190401:    Finish of operative input data preparatin
20140722190401:    Start of validation by Multisensor France
20140722190522:    Finish of validation by Multisensor France
20140722190522:    Start of validation by Multisensor Italy
20140722190551:    Finish of validation by Multisensor Italy
20140722190551:    Start of validation by ARGO
20140722193340:    Finish of validation by ARGO
20140722193340:    Start of validation by SL
20140722194107:    Finish of validation by SL
20140722194107:    Start of local uploading validation results
20140722194505:     End of Validation cycle
```

Рис. 5. Журнал работы автоматической системы валидации, доступный для удаленного контроля работы системы



```
Lister - [G:\logs\Valid_ARGO_20140721(20140722190551).log]
Файл  Правка  Вид  Справка                                        100 %
Date of model data: 20140721
20140722190551:    Start of ARGO data preparation
20140722190558:    Finish of ARGO data preparation
20140722190558:   Start of comparing ARGO and Model data
20140722192119:    Finish of Validation
20140722192119:     Start of Model and ARGO profile visualization
20140722192622:    Product is plotted
20140722192622:     Start of ARGO profile visualization results uploading
20140722192753:    Finish of ProfVisUpl
20140722192753:     Start of Model and ARGO time series visualization
20140722193141:    Product is plotted (TS)
20140722193141:     Start of ARGO time series visualization results uploading
20140722193338:    Finish of TSVisUpl
20140722193338:     Start of writing ARGO buoys coordinates
20140722193339:    ARGO buoys coordinates are writes
20140722193339:     Start of ARGO buoys coordinates files uploading
20140722193340:     End of Validation cycle
```

Рис. 6. Журнал работы подсистемы валидации по данным буев ARGO, доступный для удаленного контроля работы системы

Технические испытания автоматической системы валидации

Описанная система работает в Морском гидрофизическом институте РАН почти три года. За этот период система прошла отладку и доработку. Были существенно расширены ее возможности, а также усовершенствованы используемые методы автоматизации работы. Эксплуатация системы показала ее стабильность, удобство, отсутствие сбоев в работе.

Заключение

В Морском гидрофизическом институте РАН разработана автоматическая система валидации результатов прогноза гидрологических полей Черного моря, рассчитанных с помощью модели MHIC2. Она поддерживает выполнение большинства необходимых функций и работает в полностью автоматическом режиме, осуществляя валидацию прогностических данных ежедневно. Предложенная автоматическая система валидации отличается отсутствием в процессе ее функционирования ручных операций и возможностью быстрого, в том числе удаленного контроля работы на любом этапе с использованием журналов работы.

Автоматическая система валидации результатов прогнозирования гидрологических полей Черного моря может быть предложена как прототип для разработки систем подобного назначения.

Список литературы

1. **Демышев С. Г.** Численная энергосбалансированная модель бароклинных течений в океане с неровным дном на сетке С. М.: ИВМ, 1992. С. 163—231.

2. **Иванчик М. В., Иванчик А. М.** Структура автоматической системы диагноза и прогноза состояния Черного моря // Экологическая безопасность прибрежной и шельфовой зон и комплексное использование ресурсов шельфа. Сб. научн. трудов МНИ НАН Украины. Вып. 19. Севастополь, 2009. С. 189—198.

3. **Network** Common Data Form (NetCDF). URL: http://www.unidata.ucar.edu/software/netcdf/

4. **THREDDS** Data Server (TDS). URL: http://www.unidata.ucar.edu/downloads/thredds/index.jsp

5. **Коротаев Г. К., Демышев С. Г., Дорофеев В. Л.** и др. Архитектура и результаты работы Международного Черноморского центра морских прогнозов, созданного на базе МГИ НАН Украины в рамках проекта Европейского союза "МОЙ ОКЕАН" // Экологическая безопасность прибрежной и шельфовой зон и комплексное использование ресурсов шельфа. Сб. науч. трудов МНИ НАН Украины. Вып. 27. Севастополь, 2013. С. 128—133.

6. **Ратнер Ю.Б., Холод А. Л.** Структура системы валидации результатов диагноза и прогноза состояния Черного моря // Экологическая безопасность прибрежной и шельфовой зон и комплексное использование ресурсов шельфа. Сб. науч. трудов МНИ НАН Украины. Вып. 19. Севастополь, 2009. С. 199—202.

7. **Иванчик А. М., Иванчик М. В.** Компьютерная программа "CalcManPack — командный язык пакета для управления процессом вычислений, предназначенный для разработки и выполнения заданий, которые управляют информационно-вычислительными процессами". Свидетельство № 46522 о регистрации авторского права МГИ НАН Украины на компьютерную программу. Дата регистрации 27.11.2012.

8. **Иванчик А. М., Иванчик М. В.** Компьютерная программа "CalcMan — комплекс программ для управления процессом вычислений". Свидетельство № 46521 о регистрации авторского права МГИ НАН Украины на компьютерную программу. Дата регистрации 27.11.2012.

9. **Иванчик А. М**. Управление функционированием автоматизированных систем морского прогноза // Современные проблемы гуманитарных и естественных наук: материалы XX Междунар. науч.-практ. конф. Москва. 2—3 октября 2014. М.: Институт стратегических исследований, 2014. С. 56—61.

10. **Иванчик А. М., Иванчик М. В.** Процедуры контроля работы автоматизированных систем морского прогноза // Интеграция науки и практики как механизм эффективного развития современного общества: материалы XVI Междунар. науч.-практ. конф. Москва. 30 июня 2015. М.: Институт стратегических исследований, М.: Перо, 2015. С. 52—55.

A. M. Ivanchik, Graduate Student, Leading Programmer, e-mail: antonishka@gmail.com,
Marine Hydrophysical Institute, Sevastopol

The Validation Automatic System Structure of the Diagnosis and Prognosis the Black Sea State Results

This article describes the automatic validation system of forecast data resulting from the operation of the model MHIC2 intended for the diagnosis and prognosis of physical fields of the Black sea — sea level anomalies, sea temperature, salinity and current velocities developed in Marine hydrophysical Institute of the RAS. Operational validation system is being developed since 2012, and currently consists of three subsystems: validation by sea surface temperature satellite data, validation by in situ data of ARGO buoys, validation by sea level anomalies satellite data. Discusses the structure and operation of each of the subsystems, as well as the mechanism of interaction between subsystems.

When developing the requirements to the system parameters necessary for its effective functioning are formulated. In accordance with this requirements the system of operational validation had been developed. The developed system has a modular structure. The modular structure of the system has the advantage that errors of various origins in any subsystem, the remaining parts of the system continue to work.

The system operates in a cyclic mode (the start time is set in system settings), daily consistently starting programs that control the validation subsystems on various data, analyzes the results, automatically maintains logs of work. The logs of the system is available for remote control. Remotely accessible logs allow you to monitor what is happening

in the system at any time and from any location that is limited only by the availability of the Internet. This gives you the ability to respond quickly to situations that require quick solution.

The operational validation automatic system of the diagnosis and prognosis the Black sea state results operates in MHI RAS for three years. The system has good reliability and high efficiency due to elimination of manual work operations and for quick control of the system at any stage using the logs of the system.

Keywords: *automatic systems, validation, marine forecast, the Black sea, sea level anomaly, sea surface temperature, salinity, automatic systems development tools, methods of programs operation control*

References

1. **Demyshev S. G.** *Chislennaja energosbalansirovannaja model' baroklinnyh techenij v okeane c nerovnym dnom na setke C*. (Numerical energy-balanced model of baroclinic currents in the ocean with irregular bottom on the grid C), Moscow, IVM, 1992, pp. 163—231. (in Russian).

2. **Ivanchik M.V, Ivanchik A. M.** Struktura avtomaticheskoj sistemy diagnoza i prognoza sostojanija Chernogo moray (The automatic system of diagnosis and prognosis the Black sea state structure), *Ekologicheskaja bezopasnost' pribrezhnoj i shel'fovoj zon i kompleksnoe ispol'zovanie resursov shel'fa. Sb. nauch. trudov MGI NAN Ukrainy*, *Issue 19*, Sevastopol, 2009, pp. 189—198 (in Russian).

3. **Network** Common Data Form (NetCDF), available at: http://www.unidata.ucar.edu/software/netcdf/

4. **THREDDS** Data Server (TDS), available at: http://www.unidata.ucar.edu/downloads/thredds/index.jsp.

5. **Korotaev G. K., Demyshev S. G., Dorofeev V. L., Knysh V. V., Kubryakov A. I., Suslin V. M., Bayankina T. M., Voronina N. N., Ivanchik A. M., Ivanchik M. V., Kryl M. V., Mamchur N. L., Ratner Yu. B., Holod A. L., Inyushina N. V., Makaev A. I., Martynov M. V., Shokurov M. V.** Arhitektura i rezul'taty raboty Mezhdunarodnogo Chernomorskogo tsentra morskih prognozov, sozdannogo na baze MGI NAN Ukrainy v ramkah proekta Evropejskogo sojuza "MOJ OKEAN" (Architecture and results of work of International black sea marine forecast center, created on the basis of MHI NAS of Ukraine within the framework of the EU project "MY OCEAN"), *Ekologicheskaja bezopasnost' pribrezhnoj i shel'fovoj zon i kompleksnoe ispol'zovanie resursov shel'fa. Sb. nauch. trudov MGI NAN Ukrainy,* Issue 27, Sevastopol, 2013, pp. 128—133 (in Russian).

6. **Ratner Yu.B., Kholod A. L.** Structura sistemy validatsii rezul'tatov diagnoza i prognoza sostojanija Chernogo moray (The validation system structure of the diagnosis and prognosis the Black sea state results), *Ekologicheskaja bezopasnost' pribrezhnoj i shel'fovoj zon i kompleksnoe ispol'zovanie resursov shel'fa. Sb. nauch. trudov MGI NAN Ukrainy*, Issue 19, Sevastopol, 2009, pp. 199—202 (in Russian).

7. **Ivanchik A.M., Ivanchik M. V.** Svidetel'stvo MGI NAN Ukrainy o gosudarstvennoi registratsii programm dlya EVM "CalcManPack — komandnyj jazyk paketa dlya upravlenija protsessom vychislenij, prednaznachennyj dlya razrabotki i vypolnenija zadanij, kotorye upravllyajut informatsionno-vychislitel'nymi protsessami", no. 46522 (The computer program "CalcManPack — command language of package for process control calculations used to develop and perform tasks that manipulate information-computational processes". Certificate no. 46522 on copyright registration of MHI NAS of Ukraine on a computer program), 27.11.2012 (in Russian).

8. **Ivanchik A. M., Ivanchik M. V.** Svidetel'stvo MHI NAN Ukrainy o gosudarstvennoi registratsii programm dlya EVM "CalcMan — kompleks program dlya upravlenija protsessom vychisleni", № 46521 (The computer program "CalcMan — a package of programs to manage the process of computing". Certificate no. 46521 on copyright registration of MHI NAS of Ukraine on a computer program.), 27.11.2012 (in Russian).

9. **Ivanchik A. M.** Upravlenie funktsionirovaniem avtomatizirovannykh sistem morskogo prognoza (Management of functioning of the marine forecast automated systems), *Materialy XX Mezhdunar. nauch.-prakt. konf. "Sovremennye problemy gumanitarnyh i estestvennyh nauk"*. Moscow, October, 2—3, 2014. Moscow, Institut strategicheskikh issledovanii, 2014, pp. 56—61 (in Russian).

10. **Ivanchik A. M., Ivanchik M. V.** Protsedury kontrolya raboty avtomatizirovannykh sistem morskogo prognoza (Procedures controlling the operation of the marine forecast automatic systems), *Materialy XVI Mezhdunar. nauch.-prakt. konf. "Integratsija nauki i praktiki kak mehanizm effektivnogo razvitija sovremennogo obschestva"*. Moscow, June 30, 2015. Moscow, Institut strategicheskikh issledovanii, 2015, pp. 52—55 (in Russian).

Указатель статей, опубликованных в журнале "Программная инженерия" в 2015 году

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева.* Корректор *З. В. Наумова*

Сдано в набор 05.10.2015 г. Подписано в печать 19.11.2015 г. Формат 60×88 1/8. Заказ РИ1215
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**