УДК 004.6, 004.04
DOI 10.25205/1818-7900-2018-16-3-87-104

М. А. Городничев [1–3], **А. В. Комиссаров** [1, 3], **А. В. Можина** [1, 3], **П. В. Прочкин** [1, 3]
П. Д. Рудыч [1], **А. В. Юрченко** [1]

[1] *Институт вычислительных технологий СО РАН*
пр. Академика Лаврентьева, 6, Новосибирск, 630090, Россия

[2] *Институт вычислительной математики и математической геофизики СО РАН*
пр. Академика Лаврентьева, 6, Новосибирск, 630090, Россия

[3] *Новосибирский государственный университет*
ул. Пирогова, 1, Новосибирск, 630090, Россия

maxim@ssd.sscc.ru, yurchenko@ict.nsc.ru

МОДЕЛИ И ПРОЕКТНЫЕ РЕШЕНИЯ СИСТЕМЫ ХРАНЕНИЯ И ОБРАБОТКИ ИССЛЕДОВАТЕЛЬСКИХ ДАННЫХ ECCLESIA

В ходе научных исследований порождается большое количество данных в цифровом формате, и для последующего использования этих данных (обработки, анализа, публикации) их необходимо организованно собирать и хранить. Построение информационной инфраструктуры для решения этих задач – одна из наиболее актуальных проблем в области организации работы с экспериментальными данными. Авторами настоящей статьи разрабатывается информационная система для автоматизации сбора, хранения и анализа данных, в качестве отправной точки для которой используются три задачи обработки данных из области физиологии. Рассмотрены и проанализированы возникающие в процессе разработки такой системы проблемы, а также существующие подходы и готовые решения этих и схожих задач. На основе результатов проведенного анализа предложен ряд моделей и механизмов для решения возникших проблем. Разработанные решения включают в себя модели и механизмы сбора и хранения экспериментальных данных, модель для описания и формализации сценариев обработки данных и механизмы для обработки собранных данных в распределенной вычислительной системе. В результате представлена архитектура вычислительной системы для сбора, хранения и обработки экспериментальных данных. Система предлагается в качестве инструмента для решения широкого спектра задач, возникающих при проведении научных исследований и требующих сбора, хранения и многоэтапной обработки различных типов данных.

Ключевые слова: управление научными данными, информационная система, обработка и анализ данных, данные физиологических исследований, объектное хранилище данных, функциональный язык, распределенные вычисления.

Введение

В современных исследованиях часто приходится сталкиваться с большим объемом экспериментальных данных, получаемых из различных источников. Всё чаще данные собираются и сохраняются с расчетом на то, что некоторые задачи их анализа будут поставлены в будущем. Перспективные задачи могут предполагать использование дополнительных данных, собранных независимо, в другое время другими исследователями. Нарастающие

Городничев М. А., *Комиссаров А. В.*, *Можина А. В.*, *Прочкин П. В.*, *Рудыч П. Д.*, *Юрченко А. В.* Модели и проектные решения системы хранения и обработки исследовательских данных Ecclesia // Вестн. НГУ. Серия: Информационные технологии. 2018. Т. 16, № 3. С. 87–104.

ISSN 1818-7900. Вестник НГУ. Серия: Информационные технологии. 2018. Том 16, № 3

проблемы организации исследовательских данных и систематизации работы с ними стали, таким образом, самостоятельной задачей.

Управление научными (исследовательскими) данными (research data management) и «цифровое курирование» (digital curation) [1] к настоящему моменту являются устоявшимися терминами, выражающими потребности научного сообщества в инструментах и сервисах для работы с генерируемыми данными. В то время как в Европе и США эта проблематика вынесена на государственный и даже межгосударственный уровень: поддерживаются и развиваются узкоспециализированные инфраструктуры для хранения научных данных, например DataONE [1], разрабатываются и реализуются крупные программы, направленные на обоснование необходимости развития таких инфраструктур и проработку соответствующей нормативной базы, которая, в частности, будет стимулировать исследователей делиться своими данными (см. проект «Research Data e-Infrastructures: Framework for Action in H2020» [2]).

В обзоре [1] приводится «западная» оценка общей стоимости формирования инфраструктуры научных данных – 10–15 % от стоимости всей инфраструктуры науки. При этом в России на уровне федеральных властей обсуждение необходимости и путей создания инфраструктуры для науки, основанной на цифровых данных, активизировалось только в 2016–2017 гг. в связи с разработкой и принятием программы «Цифровая экономика» [2], хотя отдельные инициативы в этом направлении реализовывались и ранее [3; 4].

Несмотря на более чем 10-летнюю историю вопроса, работа с исследовательскими данными, за исключением отдельных научных направлений (например, [5]), остается слабо упорядоченной и в лучшем случае организуется путем составления и следования инструкциям, таким как «Data management: Helping MIT faculty and researchers manage, store, and share data they produce» [3]. Такое положение дел в дальнейшем будет только сдерживать развитие науки, поэтому исследователи нуждаются в создании инструментов, которые позволят им решать задачи организации сбора, хранения, обработки и анализа данных, обмена ими и их публикации.

В работе [6] предлагается подход к систематизации и соответствующей автоматизации процессов, связанных с согласованным хранением и обработкой неоднородных исследовательских данных. В общем виде рассмотрены требования к организации специализированной информационно-аналитической системы, представлено видение её архитектуры, описано состояние развития инфраструктуры Института вычислительных технологий (ИВТ) СО РАН для работы с научными данными.

В настоящей работе представлен следующий этап в рамках инкрементального подхода к созданию этой информационно-аналитической системы поддержки научных исследований. Анализируется опыт Лаборатории технологий анализа и обработки биомедицинских данных ИВТ СО РАН в решении задач сбора, хранения и анализа данных, возникших в рамках ряда физиологических исследований. На основе этого анализа формулируются минимальные требования к информационно-аналитической системе для работы с такими данными, предлагаются основные проектные решения по созданию ее прототипа – системы Ecclesia.

Одним из немногих проектов, нацеленных на целостное решение проблемы автоматизации сбора, хранения, обработки и публикации научных данных, является проект разработки платформы и сервиса для биомедицинских исследований Galaxy Project [4]. Несмотря на направленность проекта на конкретную предметную область, результаты его могут быть использованы и в других областях научных знаний. Однако сервис достаточно сложен в освоении, не поддерживает включение интерактивных сессий с пользователями в качестве этапов автоматически выполняющихся сценариев обработки данных и более приспособлен к накоплению отдельных объектов данных, воспроизводимости отдельных сценариев обработки данных, чем к систематизации накопления и автоматизации применения знаний о предметной области.

[1] https://www.dataone.org
[2] http://www.in2p3.fr/actions/informatique/media/h2020.pdf
[3] https://libraries.mit.edu/data-management/
[4] https://galaxyproject.org

Системы для хранения или обработки данных в целом можно разделить на три класса: системы общего назначения (например, Dell EMC Elastic Cloud [5], Globus [6]), специфичные для предметной области (например, SIMBioMS [7], CARMEN [8], XTENS 2 [9]) и решающие частные задачи (например, MedMining [7]). В системах общего назначения особое внимание уделяется надежности хранения, но не предоставляются инструменты для описания данных и их связей. Системы для решения частных задач, напротив, поддерживают связи между данными, но в рамках специфичной для задачи схемы, которая чаще всего не переносима на другие задачи. Среди специфичных для предметной области систем рассматривались системы хранения биомедицинских данных [8–10]. Достаточной степенью универсальности не обладает ни одна система. Так, например, они не поддерживают возможность проверки структуры загруженных данных и не позволяют вводить и тем более строить новые связи между данными.

Среди систем формирования и исполнения сценариев обработки данных также можно выделить системы общего назначения (например, Google Cloud Dataflow [10] или ActiveEon ProActive Workflows & Scheduling [11]) и системы, специфичные для предметной области (например, LabView [12] и Unipro UGENE [13] [11]). Системы общего назначения по большей части направлены на пользователей с навыками программирования. Важным примером системы, позволяющей создавать предметно-ориентированные визуальные языки программирования, является CoCoViLa [12], которая для этого требует от пользователя решать более широкую задачу, чем задание и выполнение отдельных сценариев обработки данных. Узкопредметные системы избавляют пользователей от таких сложностей, но изначально не приспособлены для решения междисциплинарных задач.

Разрешению обозначенных проблем различных систем для работы с данными и посвящена наша работа.

Анализ «пользовательских историй»

В основе требований, предъявляемых к разрабатываемой системе, лежит опыт решения задач сбора, хранения и анализа данных для ряда физиологических исследований и результаты обсуждения соответствующих проблем со специалистами-физиологами.

Задача персонализированной телемедицины:
индивидуальный подбор программы тренировки

Рассматривается задача создания велотренажера для реабилитации больных после инсульта. Врач задает программу тренировки пациента с помощью графиков требуемой скорости вращения педалей и величины сопротивления тренажера кручению педалей. Эти значения преобразуются мобильным приложением в управляющие воздействия контроллера тренажера. Необходимо индивидуально подбирать программу тренировки так, чтобы пациент получил нагрузку, адекватную его состоянию. Для этого необходимо собирать и анализировать данные, характеризующие фактическое прохождение тренировки. Собираемые данные включают в себя частоту сердечных сокращений (ЧСС) и реальную скорость кручения педалей. Мобильное приложение собирает данные и формирует пары вида «отметка времени, значение ЧСС» и «отметка времени, значение реальной скорости».

Для анализа процесса тренировки требуется сохранять и визуализировать получаемые данные, сопоставляя их с программой тренировки. Все данные являются одноканальными

[5] https://www.dellemc.com/ru-ru/storage/ecs/
[6] https://www.globus.org/
[7] https://www.simbioms.org/
[8] http://www.carmen.org.uk/
[9] https://github.com/xtens-suite/xtens-app
[10] https://cloud.google.com/dataflow/
[11] https://proactive.activeeon.com
[12] http://www.ni.com/ru-ru/shop/labview.html
[13] http://ugene.net

временными рядами (ОВР). Программу тренировки также можно представить в виде ОВР. Требуется делать выборки в заданных временных интервалах как отдельных рядов, так и их совокупностей, совмещенных по времени. Предложено ввести абстракцию ОВР на уровне решения для хранения, что позволяет унифицировано представлять данные. Помимо данных ряда предложено сохранять *атрибуты* ряда:

1) тип ряда (в данном случае некоторые идентификаторы, позволяющие отличать ряды ЧСС от рядов скорости кручения педалей и др.);
2) идентификатор устройства-источника;
3) время начала и окончания записи.

Доступ к системе хранения данных предоставляется через веб-сервис, который позволяет сохранять и извлекать ОВР. Данные ОВР хранятся в бинарных файлах, атрибуты отдельных записей со ссылкой на бинарный файл хранятся в реляционной БД. Решение позволяет унифицировано сохранять и извлекать ОВР и независимо разрабатывать различные клиентские приложения для работы с ними. В частности, реализовано сохранение данных в мобильном приложении и выгрузка данных в веб-интерфейс для визуализации.

Поскольку на уровне понятий в системе была поддержана логика работы с ОВР, пользователи получили возможность выбирать данные в веб-интерфейсе по заданным временным промежуткам записи и типу ряда, а не по именам или датам создания файлов. С пользователей также сняты типичные задачи организации данных в некоторую структуру директорий, запоминания этой структуры и ручного поиска в ней.

В ходе решения задачи проявилась необходимость обеспечивать в рамках информационно-аналитической системы реализацию понятий предметной области и поддержку работы с соответствующими объектами данных в терминах предметной области на уровне программных и пользовательских интерфейсов. Типичная система организации данных в виде файлов, собранных в структуры директорий, является в подобных ситуациях неадекватной задачам поиска и обработки данных.

Задача предобработки записей ЭЭГ

Запись электроэнцефалограммы (ЭЭГ) состоит из N каналов, в которых фиксируется динамика разности потенциалов между N электродами и электродом-референтом. ЭЭГ-записи могут использоваться для выявления схожести и различий в реакции пациентов на одинаковые стимулы [13], для восстановления позиций источников сигналов в мозге [14] и др. Однако предваряет содержательный анализ данных очистка ЭЭГ-записей от шумов [15].

В рамках этапа очистки от шумов используются запись сигналов ЭЭГ, а также информация о размещении электродов на голове пациента [16], время подачи стимулов, протокол эксперимента, в котором описаны детали проведения эксперимента. Приемы очистки записи могут применяться отдельно либо в некоторой последовательности. Наиболее распространенные из них:

1) применение частотных фильтров для удаления известных помех;
2) выявление экспертом зашумленных каналов и их удаление;
3) смена электрода-референта и соответствующий перерасчет значений ЭЭГ во всех каналах [17];
4) усиление значимого сигнала за счет суммирования выровненных по моменту предъявления стимула участков записи, так называемых эпох;
5) коррекция нулевой линии (baseline correction) [18];
6) выявление экспертом поврежденных эпох и их удаление;
7) применение анализа независимых компонент [19], выявление экспертом шумовых компонент и их удаление; каждая компонента при этом визуализируется в виде карты активности мозга.

Как правило, в физиологических лабораториях можно наблюдать, что все собираемые данные, а также данные, получаемые в ходе этапов обработки, хранятся как отдельные файлы (протокол эксперимента и вовсе может храниться в бумажном лабораторном журнале), структурирование которых в упорядоченные системы каталогов, именование, формирование наборов данных для очередных этапов обработки осуществляются вручную. Автоматизация

сбора и обработки должна освободить исследователей от этого рутинного труда и ликвидировать ошибки, обусловленные человеческим фактором. Должна быть обеспечена возможность формировать и автоматически выполнять сценарии, состоящие из набора связанных операций обработки данных, а перенос данных между операциями должен быть автоматизирован.

На практике в ходе обработки данных от пользователя требуется выбор конкретных методов. Например, существует множество алгоритмов для проведения ICA [20], нужно сделать выбор частотных фильтров и т. д. Не все эксперты в предметной области задачи являются экспертами в математической обработке данных, и для них были бы ценны советы при выборе методов обработки. Для этого целесообразно, чтобы система собирала и накапливала опыт других пользователей в составлении сценариев.

Нужно учитывать, что некоторые операции обработки данных пока (или принципиально) не могут быть автоматизированы, поэтому может потребоваться анализ и принятие решения экспертом. Таким образом, должна быть заложена возможность включать сессии взаимодействия с пользователями в качестве этапов сценариев обработки данных.

Задача сопоставления характеристик томограмм головного мозга

Задача состоит в поиске нейрональных сетей на основе анализа выявляемой с помощью функциональной магнитно-резонансной томографии (фМРТ) активности мозга пациента (например, [21–23]). С помощью фМРТ измеряется связанная с активностью нейронов гемодинамика. Каждый снимок фМРТ характеризует насыщенность крови кислородом в различных областях головного мозга. Дискретное представление насыщенности основано на разбиении куба, в который вписана голова пациента, на кубические единичные объемы – воксели. Снимок фМРТ представляет собой, таким образом, трехмерный массив чисел.

В ходе решения задачи анализируется последовательность фМРТ-снимков головы пациента. В каждом снимке выделяются группы вокселей (по выбору исследователя), и для каждой группы строится индекс активации (некоторое усреднение значений всех вокселей группы). Эти операции повторяются для всей последовательности снимков, таким образом получается временная развертка индексов активации. Представляет интерес выделение групп вокселей, демонстрирующих различную или сходную динамику индекса активации [24].

Для одного эксперимента в этой задаче нужно обработать тысячи групп вокселей во временной развертке. Каждая группа вокселей может быть рассмотрена независимо, что позволяет обрабатывать их параллельно. Таким образом, задачу целесообразно решать на параллельных вычислительных системах.

Требования к системе

На основе анализа «пользовательских историй» в совокупности с общими представлениями о назначении системы [6] сформулированы требования к минимальному жизнеспособному продукту (MVP – Minimal Viable Product). Необходимо обеспечить следующее:

1) возможность сохранять и извлекать разнородные экспериментальные данные вместе с информацией об их структуре и связях с другими данными;

2) возможность модифицировать сведения о данных и связях между ними в любой момент времени;

3) проверку соответствия данных заданной структуре с возможностью исправления выявленных несоответствий при сохранении в системе;

4) возможность задавать сложные пользовательские сценарии обработки данных;

5) обращение к данным и сохранение новых данных из сценариев;

6) выполнение сценариев на параллельных вычислительных системах, при этом пользователь должен быть по возможности избавлен от необходимости императивного параллельного программирования;

7) возможность интерактивного вмешательства пользователя в ход исполнения сценария;

8) накопление знаний о предметной области и поддержку действиям пользователя в построении сценария;

9) автоматизированный анализ и оптимизацию процессов обработки данных.

Архитектура системы и проектные решения

На основе выявленных требований к системе в ходе проектирования были выделены три ее относительно самостоятельные подсистемы:

1) хранения и управления данными;
2) формирования сценариев;
3) организации распределенной обработки.

Для возможности независимой разработки подсистем и применения различных технологий для их реализации выбрана микросервисная модель архитектуры системы, представленная на рис. 1.

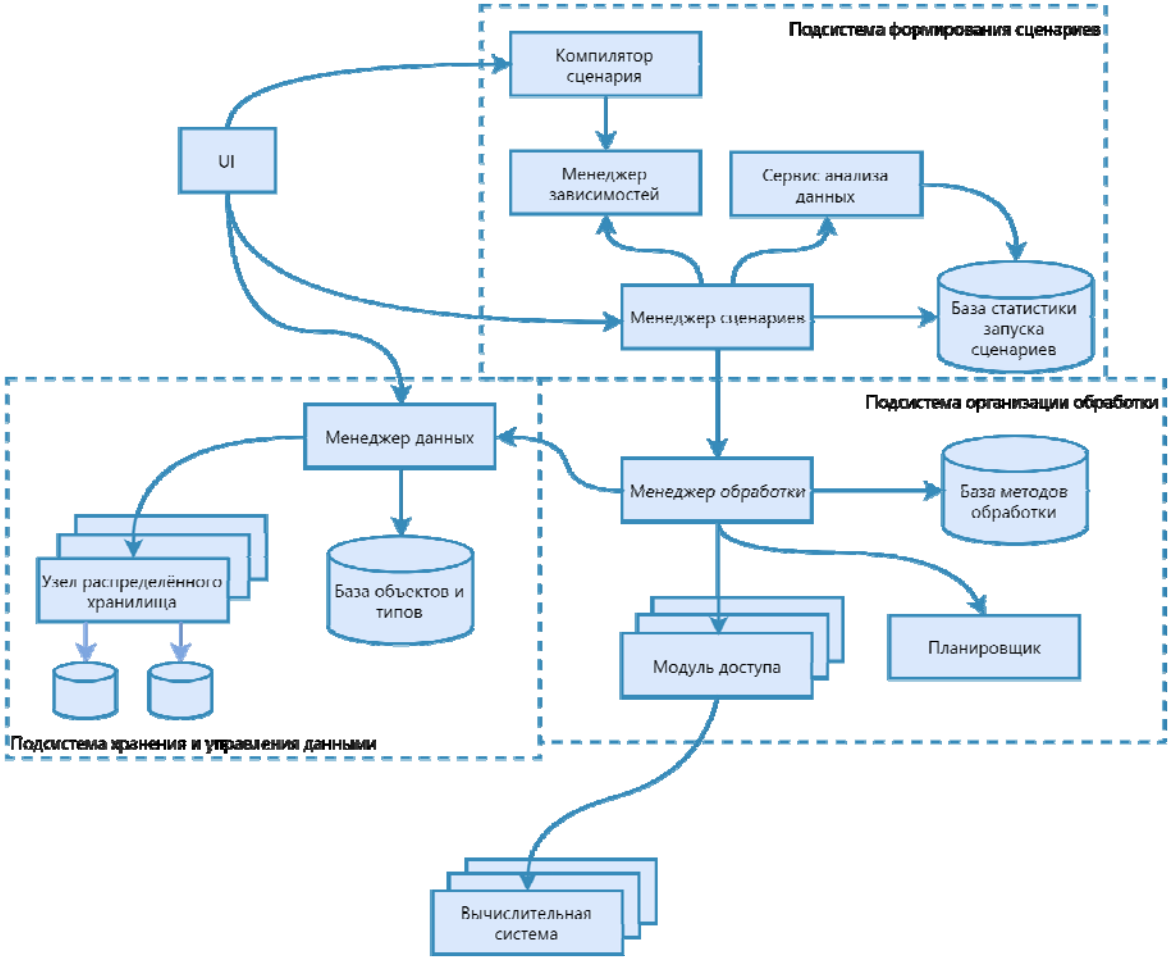

Рис. 1. Архитектура системы

Подсистема хранения и управления данными реализуется в виде надстройки над существующим распределенным высоконадежным хранилищем. Надстройка представляет собой промежуточный сервер (менеджер данных), организующий данные пользователя в соответствии с предлагаемой моделью представления данных (см. ниже) и предоставляющий программный интерфейс (API – Application Programming Interface) для работы с этими дан-

ными. На основе API менеджера данных независимо могут быть построены различные клиентские приложения. Базовым пользовательским интерфейсом является веб-интерфейс с функциями загрузки и извлечения данных, просмотра каталога данных и поиска в нем.

Модель представления данных

Для обеспечения хранения и извлечения разнородных экспериментальных данных вместе с информацией об их структуре и связях предложена следующая модель представления данных. Вводится понятие *объекта данных* как единицы хранения. Для каждого объекта при создании формируется его идентификатор. Все объекты автоматически или по указанию пользователя относятся к одному из *типов данных*, зарегистрированных в системе, в одном из *форматов данных*. Таким образом, любой файл или некоторым образом структурированный набор файлов, будучи сохраненными в качестве объекта данных, могут быть содержательно интерпретированы системой. Например, данные одного ЭЭГ-исследования на энцефалографах BrainVision сохраняются в виде тройки файлов: запись сигналов ЭЭГ в файле формата *.eeg, файл разметки ЭЭГ по времени подачи стимулов (*.vmrk) и файл, содержащий информацию о каналах энцефалографа (*.vhdr). После загрузки файлов в систему хранения эта тройка рассматривается как один объект данных типа *Запись ЭЭГ* в формате *BrainVision*. Такой подход отражает связь между файлами, позволяет обращаться к ним совместно и избегать ошибок совмещения сигналов и времени подачи стимулов от разных записей при многократном обращении.

Для обеспечения расширяемости подсистемы хранения предлагается механизм введения нового типа. Новый тип в системе задается *схемой метаданных*, списком возможных *форматов* (задаются строковыми идентификаторами) и набором *операций*, которые могут принимать на вход или вырабатывать в качестве своего результата объекты данного типа, возможно, наряду с объектами других типов (см. раздел «Подсистема организации распределенной обработки»).

Схема метаданных задает название типа, синонимы для названия и текстовое описание типа, а также специфицирует набор свойств объектов данного типа в виде множества пар (атрибут, тип). Атрибут – произвольная строка, имя некоторого свойства объекта данных, тип – строковый идентификатор типа значения атрибута, выбирается из типов, зарегистрированных в системе, например: атрибут «Дата регистрации», тип «Дата». Одним из типов является ссылка на другой объект данных по идентификатору объекта. Ссылки в метаданных на другие объекты позволяют задавать произвольные *связи* между данными. С каждым объектом данных хранятся его метаданные. Значения атрибутов заполняются пользователем или автоматически, например, некоторым клиентским программным обеспечением.

Типы, у которых задана только схема метаданных, называются *типами-стикерами* и могут быть включены в определения схем метаданных других типов для переиспользования набора атрибутов, заданных в типе-стикере. Экземпляр такого типа, стикер, – объект метаданных, содержание которого составляют только метаданные с присвоенными атрибутам значениями. Стикер может быть прикреплен к произвольному объекту данных, даже если этот стикер не включен в схему метаданных типа данного объекта.

С помощью атрибутов-ссылок и стикеров можно, в частности, сохранить связь между исходным объектом, например записью ЭЭГ, и преобразованным, например ЭЭГ, отфильтрованной от наведения электросети. Для этого нужно создать и прикрепить к полученному после фильтрации объекту стикер с названием «Без наведения» с атрибутом «Получено из» типа *Запись ЭЭГ* для сохранения ссылки на исходный объект и атрибутом «Частота наведения» типа вещественное число для сохранения параметров фильтрации (рис. 2).

Согласно требованию о проверке структуры загружаемых данных, в подсистеме вводится понятие *валидатора* – операции проверки соответствия структуры данных заданному типу и формату этих данных. Вызов этих операций при загрузке данных в хранилище позволяет автоматически выявлять несоответствие между содержанием объекта данных и указанными пользователем или автоматически определенными типом и форматом объекта данных.

Модель доступа к данным

Разработан программный интерфейс веб-сервиса доступа к данным в терминах управления ресурсами сервиса, согласно архитектурному стилю REST [25]. Типы ресурсов сервиса имеют прямое соответствие терминам модели представления данных (тип данных, формат

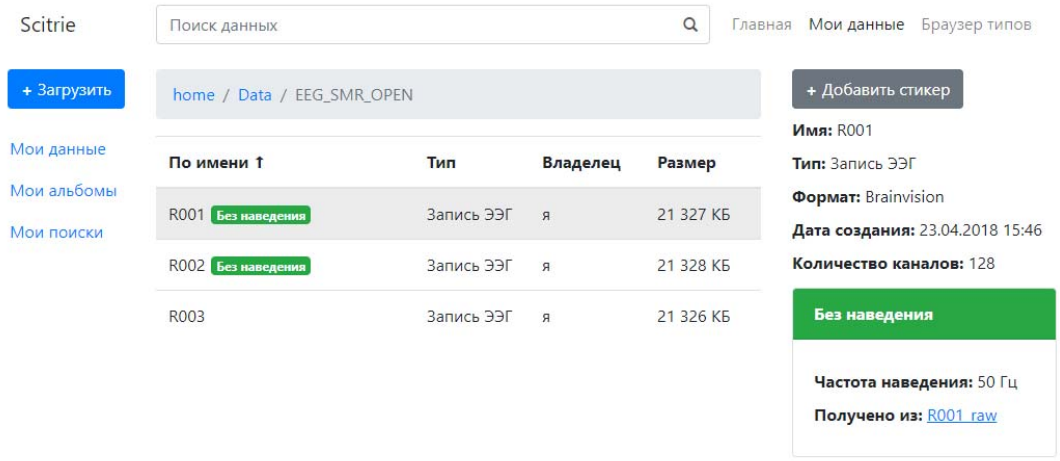

Рис. 2. Пример визуализации стикера
для преобразованной записи ЭЭГ в пользовательском интерфейсе

данных, стикер, операция, объект). Таким образом, каждый тип данных, каждый объект и т. д., а также коллекция таких сущностей, является ресурсом (в терминах REST) и получает имя (URI – Uniform Resource Identifier), посредством которого клиенты сервиса могут обращаться к ресурсу и осуществлять действия из набора CRUD [26].

Относительно способов передачи значений при запросе объектов типы разделены на два класса.

1. *Ссылочные типы*. В результате запроса объекта по его идентификатору клиенту интерфейса возвращается информация (URL – Unified Resource Locator) о том, как может быть получен доступ непосредственно к данным объекта. К этому классу относятся все сложные типы (например, запись ЭЭГ). Для получения данных (в случае записи ЭЭГ в формате BrainVision это тройка файлов, описанная выше) необходимо сделать отдельный запрос по URL. Ссылочные типы позволяют вынести в отдельный блок работ исследование и внедрение механизмов оптимизации доступа к данным, например, таких, как размещение копий данных (кэши) ближе к месту их использования.

2. *Типы значений.* Объекты таких типов возвращаются клиенту непосредственно при обращении по идентификатору объекта. К этому классу относятся примитивные типы (например, числа и строки).

Предложенная модель доступа к данным предоставляет инструменты доступа к системе хранения как из клиентских приложений (веб-интерфейс системы хранения, приложения для персональных компьютеров и мобильных устройств, программное обеспечение регистрирующего оборудования – собираемые устройствами данные могут непосредственно отправляться в хранилище, и т. д.), так и из подсистемы распределенной обработки данных.

Подсистема формирования сценариев
Модель абстрактного параллельного вычислителя

Поскольку одним из требований к системе является выполнение сценариев на различных параллельных вычислителях, то предлагается модель, в которой вычислитель может исполнять сценарии, описанные в виде частично упорядоченного множества операций. Операция характеризуется множеством входов, множеством выходов, а также именем функции, которую она применяет к входам для получения выходов. Входы и выходы – это объекты дан-

ных. Предполагается, что по имени функции может быть найдена конкретная программная реализация этой функции (программа), которую может исполнить вычислитель. Все объекты неизменяемы и либо известны до исполнения сценария (входные данные сценария), либо являются выходами операций.

Каждая операция имеет список зависимостей – список операций, которые должны быть выполнены до исполнения этой задачи. Операция может быть исполнена, если нет неисполненных операций, от которых она зависит. Таким образом, множество списков зависимостей определяет частичный порядок на множестве операций. Списки зависимостей формируются разработчиком сценария либо, как будет представлено далее, компилятором высокоуровневого языка описания сценариев. Возможность задавать произвольный порядок (помимо обусловленного зависимостями по данным) позволяет использовать операции с побочными эффектами, при этом с исполнительной системы (планировщика) снимается задача отслеживания и обработки побочных эффектов. Примерами побочных эффектов могут быть операции ввода / вывода или инициализации ресурсов вычислителя (например, буфера памяти GPU) в одной из операций, потребление в другой.

Вычислитель принимает на вход тройку множеств $<O, D, M>$, где O – множество операций, D – множество зависимостей между операциями, M – таблица сопоставления объектов данных, являющихся входными для сценария, с их именами (мнемониками), используемыми в сценарии. В таблице M указывается URI объекта, если это ссылочный тип, или непосредственно некоторая сериализация объекта в противном случае. Запись сценария в виде тройки $<O, D, M>$ называется внутренним представлением сценария.

Такая модель абстрагирует подсистему формирования сценариев от деталей реализации вычислителя, позволяет создавать варианты языков описания сценария (текстовые, визуальные) и оставить свободу выбора конкретной реализации сценария подсистеме организации распределенной обработки (см. ниже).

Накопление знаний о предметной области

Для накопления знаний предлагается использовать методы машинного обучения. Благодаря этому система сможет самостоятельно извлекать взаимосвязи между объектами предметной области исходя из действий пользователей. У подхода есть свои недостатки. Так, методы машинного обучения не дают однозначной интерпретируемости и доказуемой корректности вывода, но для целей создаваемой системы достаточно, чтобы сохранялась информация об истории происхождения того или иного утверждения.

Основная идея состоит в том, чтобы оценивать вероятность использования пользователем функции в контексте при формировании сценария на основе статистики других пользователей. Для этого адаптируется задача анализа рыночной корзины, сформулированная в [27] и получившая широкое развитие в 2006 г. во время конкурса Netflix Prize по предсказанию пользовательских оценок фильмам и рекомендациям фильмов на их основе [28]. Задача конкурса формулировалась так: есть множество пользователей U и множество фильмов F. Известна матрица R размерностью $|U| \times |F|$, содержащая пользовательские оценки фильмов. Требуется научиться предсказывать оценку пользователя и предлагать пользователю фильмы с наивысшей предсказанной оценкой. Для этой задачи эффективным решением оказались латентные модели [28].

В задаче рекомендации функций есть заметные отличия:

1) пользователь не ставит оценки функциям (это сильно отличается от основных пользовательских задач и будет отвлекать пользователя);

2) так как система рекомендаций нужна для помощи неопытным пользователям, необходимо строить рекомендации только на основе действий более опытных пользователей, т. е. информации от одних пользователей система должна доверять меньше, чем информации от других.

Для решения этих проблем предлагается следующее.

1. В качестве «оценки» функции здесь берется вероятность применения пользователем конкретной функции в сценарии с учетом контекста – набора других использованных в сценарии функций. Эта вероятность для каждого пользователя рассчитывается эмпирически

на основе анализа статистики применения функций в разработанных пользователем сценариях.

2. Вводится мера доверия системы пользователю, которая используется как вес пользователя при обучении латентной модели.

При расчете вероятности использования функций пользователем делаются два предположения. Первое заключается в том, что распределение вероятности использования функций пользователем не меняется со временем (это предположение можно смягчить, если рассматривать только сценарии не старше некоторого порогового значения), второе – вероятность использования функций в сценарии не зависит от других сценариев.

Для вычисления меры доверия пользователю используются следующие эвристики:

1) чем больше используется различных функций, тем лучше пользователь владеет методами обработки данных;

2) чем больше в среднем функций в сценариях пользователя, тем они сложнее и тем лучше он владеет методами обработки данных.

Исходя из этого мера доверия пользователю C_u вычисляется следующим образом:

$$C_u = w_0 + w_1 n(u) + w_2 \underline{n_s(u)},$$

где $n(u)$ – количество различных методов, применяемых пользователем, $\underline{n_s(u)}$ – среднее количество методов в сценариях пользователя, w_1, w_2 – веса, w_0 – константа-смещение (сейчас характеризует экспертную оценку навыка пользователя при его регистрации в системе).

Изначально веса устанавливаются вручную и при необходимости будут скорректированы по мере появления новых пользователей. В будущем при увеличении количества пользователей планируется собрать экспертные оценки их навыка и на их основе обучить модель линейной регрессии для автоматической настройки весов.

Язык формирования сценариев

В качестве базового языка описания сценариев разработан текстовый язык, за основу которого принят язык F# по следующим причинам:

1) F# является функциональным языком, что позволяет извлекать из кода информацию о функциональных зависимостях между операциями, абстрагирует от деталей исполнения сценария на конкретной вычислительной системе;

2) синтаксис F#, как и Python (который является де-факто стандартом в анализе данных), основан на отступах, в нем нет избыточных ключевых слов, поэтому код на F# краток и легко читаем;

3) F# основан на системе типов Хиндли – Милнера [29], поэтому в программах, как и в Python, не требуется явно указывать типы при определении значений и функций; при этом язык является статически типизированным, что позволяет обнаруживать большое количество ошибок до отправки сценария на исполнение;

4) компилятор F# обладает открытым исходным кодом [14], а также предоставляет программный интерфейс для анализа кода на предмет наличия ошибок, построения типизированного и нетипизированного абстрактного синтаксического дерева, а также поддержку таких функций интегрированных сред разработчика (IDE – Integrated Developer Environment), как автозавершение кода и сборка inline-документации.

Для адаптации языка к целям формирования сценариев, а также спецификации генерируемых параллельных программ изменена семантика некоторых конструкций языка:

1) let-связывание связывает объект данных (константу или выход функции) с мнемоникой, но при этом не определяет порядок вычислений и не задает других зависимостей, помимо зависимостей по данным;

2) do-связывание служит для явного обозначения побочных эффектов (например, операций ввода-вывода, если это требуется вычислителем), в случае использования do-связыва-

[14] https://github.com/fsharp/FSharp.Compiler.Service

ния с операцией все остальные операции, описанные в сценарии после нее, получают зависимость от этой операции (см. пример на рис. 3);

3) директива open используется как для подключения наборов сигнатур функций, доступных на вычислителе, так и для переиспользования других сценариев пользователя.

Набор сигнатур – это множество объявлений функций, реализации которых доступны на вычислителе в качестве исполняемых программ. Объявления оформляются в виде функций на языке F#. Они необходимы для того, чтобы произвести проверку соответствия типов в сценарии. На первом этапе наборы сигнатур добавляются в систему вручную совместно с добавлением реализаций функций в подсистему организации распределенной обработки. Впоследствии они будут генерироваться автоматически по описаниям операций.

```
open Eeg

do initConnection "ecclesia.ict.nsc.ru:5432"

let record = loadEeg "R013"

let filtered = filter 1<Hz> 40<Hz> record

let goodChannels, badChannels = splitBadChannels filtered

let epochs = extractEpochs -1000<ms> 2000<ms> goodChannels
```

Рис. 3. Пример кода на языке описания сценария (все операции должны выполняться в порядке, обусловленном зависимостями по данным, но после выполнения операции инициализации initConnection, на что указывает оператор do)

Перед запуском код сценария обрабатывается компилятором, который анализирует сценарий на наличие ошибок, выявляет зависимости между операциями и транслирует его во внутреннее представление сценария, формируя множества $<O, D, M>$.

Наборы сигнатур функций, как и сценарии, могут также подключать другие наборы и сценарии с помощью директивы open (см. рис. 3). Таким образом, исходные коды, необходимые для компиляции, образуют направленный ациклический граф. Информация о связях между исходными кодами (дугах в графе исходных кодов) выявляется компилятором и хранится совместно с исходными кодами в системе.

Компоненты подсистемы формирования сценариев

Подсистема состоит из следующих компонентов:

1) графический интерфейс пользователя (UI – User Interface) позволяет пользователю вводить сценарии обработки данных на языке описания сценариев, инициирует компиляцию сценария во внутреннее представление и передает его менеджеру сценариев;

2) компилятор сценариев анализирует сценарий на корректность и преобразует его во внутреннее представление;

3) менеджер сценариев принимает запросы от графического интерфейса, собирает информацию о сценарии в форме внутреннего представления и инициализирует его выполнение;

4) менеджер зависимостей хранит исходные коды сценариев и наборы сигнатур функций, а также связи между ними;

5) сервис анализа пользовательской статистики собирает информацию о запусках пользовательских сценариев, инициирует создание моделей машинного обучения, рассчитывает, какие функции можно предложить пользователю.

В предложенной архитектуре менеджер сценариев и компилятор никак не связаны, менеджер сценариев работает напрямую с внутренним представлением сценария. Это сделано для того, чтобы можно было подключать к системе и другие средства описания сценария, которые компилируются во внутреннее представление, например визуальные. Все компоненты представляют собой веб-сервисы и общаются друг с другом через REST API.

Подсистема организации распределенной обработки

Распределение вычислений

Обработка собранных данных зачастую является ресурсоемкой задачей из-за их большого объема или же вследствие вычислительной сложности используемых для обработки алгоритмов.

Одним из возможных решений для сокращения времени обработки является использование высокопроизводительных вычислительных систем (ВВС). Это решение позволяет существенно сократить время выполнения ресурсоемких этапов обработки, однако может быть неуместно для этапов с низкой вычислительной сложностью. Ввиду этого выполнение обработки данных исключительно на ВВС не решает проблему в полной мере, требуется оптимизация использования привлекаемых вычислительных ресурсов.

В то же время сценарии обработки данных, имеющие вид <*O*, *D*, *M*>, естественным образом выполняются в распределенной вычислительной среде: операции обработки, не зависящие друг от друга могут выполняться параллельно. Также при организации распределенной обработки собранных данных можно использовать гетерогенное множество вычислительных систем, включающее в себя как высокопроизводительные узлы для выполнения ресурсоемких этапов обработки, так и узлы для выполнения менее требовательных к вычислительным ресурсам операций. Более того, при использовании гетерогенного множества вычислителей становится возможным включать в систему специализированные узлы, более эффективно выполняющие определенные виды обработки (например, располагающие ускорителями GPU), а также вычислительные мощности, предоставляемые пользователями системы. Именно такой подход – распределенная обработка данных на гетерогенном множестве вычислительных систем – и закладывается в систему.

Модель организации распределенной обработки

На основе выбранного подхода разработана модель распределенной обработки данных в системе. Модель предполагает централизованное управление процессом обработки данных, реализуемое отдельным сервисом. Здесь с каждой операцией из внутреннего представления сценария сопоставляется реализующая ее программа, написанная на некотором языке программирования (может быть несколько таких реализаций, отличающихся нефункциональными свойствами, например, предназначенных для исполнения на различных устройствах). Эти программы обработки данных выполняются на некотором подмножестве доступных вычислительных систем, требования к системам определяются по имеющейся об операции информации (метод обработки данных, размер входных данных, нефункциональные свойства реализаций). Распределение операций по вычислительным системам происходит при составлении плана выполнения сценария.

Нужно учесть, что некоторые вычислительные системы могут и не находиться под полным контролем системы подсистемы организации распределенной обработки. Например, представляет интерес использование ресурсов суперкомпьютерных центров, однако установка системного программного обеспечения для управления обработкой данных на таких вычислителях затруднена. Решением является отказ от использования в обязательном порядке связующего программного обеспечения, разворачиваемого на используемых вычислительных системах. Вместо этого взаимодействие с вычислительными узлами решено осуществлять через штатные для используемых вычислительных систем механизмы удаленного доступа. Также если вычислительная система использует некоторую систему управления прохождением задач (СУПЗ), то задачи обработки данных Ecclesia будут ставиться в очередь СУПЗ на общих основаниях. Для обеспечения расширяемости системы новыми типами поддерживаемых вычислительных систем, механизмы взаимодействия

с вычислителями выделяются в изолированные модули внутри системы с единым внешним интерфейсом. Ограниченность контроля над вычислительными системами также требует создания виртуальной среды для всех процессов обработки данных внутри узла, для чего используется технология Docker [15].

Планирование

Для эффективного использования распределенной системы реализуется механизм оптимизирующего планирования. Используется алгоритм RDPSO (Revised Discreet Particle Swarm Optimization) поиска приближенного решения задачи в многомерном пространстве, имитирующий движение «роя частиц», адаптированный для работы с дискретным пространством решений [30]. Алгоритмом составляется план выполнения всего сценария до начала вычислений.

Не всегда можно заранее составить план выполнения всего сценария. В сценарии могут присутствовать операции, требующие взаимодействия с пользователем (например, обработки данных человеком-экспертом). Предсказать время выполнения таких операций в общем случае не представляется возможным, что делает невозможной и оптимизацию времени выполнения сценария целиком. Также к неточности оценок времени выполнения может приводить невозможность в некоторых случаях предсказать объем входных данных для некоторых операций, поскольку время выполнения операций в существенной мере зависит от объема входных данных. Эта ситуация может возникать, например, при очистке данных с различного рода датчиков от шума: соотношение объема исходных и очищенных данных неизвестно, поэтому объем очищенных данных после выполнения такой операции трудно предсказать.

Для решения этой проблемы предлагаются следующие механизмы планирования и исполнения операций обработки данных в распределенной системе.

Планирование производится в несколько проходов для подмножеств операций в сценарии. Для каждой вычислительной системы формируются очереди задач в рамках подсистемы организации распределенных вычислений. Эти очереди в контексте обсуждения алгоритма планирования следует отличать от очередей СУПЗ, которые, возможно, используются на некоторых из вычислительных систем. Для каждого прохода выбираются операции, удовлетворяющие всем следующим условиям:

- операция еще не выполняется на некоторой вычислительной системе;
- операция не зависит ни от одной незавершенной операции, требующей взаимодействия с пользователем;
- операция не зависит ни от одной операции, объем выходных данных которой не известен.

По мере выполнения сценария таким образом может быть запланировано выполнение всех операций в сценарии: операции, требующие взаимодействия с пользователем, и операции с неизвестным объемом выходных данных выполняются, необходимая для оценки времени выполнения информация становится известна.

При использовании такого механизма планирования порядок поступления операций в очереди вычислительных систем не связан ни с порядком поступления сценариев обработки данных в систему, ни с желаемыми сроками завершения выполнения сценариев. Это может привести к завершению сценариев в сроки, значительно превышающие желаемые (даже в случаях, когда нагрузка на распределенную систему и ее конфигурация позволяют завершить сценарий в срок), а также к «бесконечному ожиданию» для задач некоторых сценариев.

Чтобы избежать возникновения перечисленных проблем, предложен следующий механизм управления очередями операций для каждой вычислительной системы.

1. В процессе статического анализа сценария и оценки времени выполнения операций, проводимых перед каждым проходом планирования, для каждой операции вычисляется ее

[15] https://www.docker.com

собственный желаемый срок завершения. Эти сроки определяются на основе желаемого времени завершения для сценария и оценки относительной длительности выполнения операции.

2. На основе собственных желаемых сроков завершения операции из всех выполняемых сценариев объединяются в группы: в одну группу попадают операции, время завершения которых попадает во временное окно некоторой длительности (например, один час).

3. Очереди операций каждой вычислительной системы приоритезируются на основе распределения операций по группам: операции из групп с более ранними желаемыми сроками завершения имеют более высокий приоритет.

При оценке времени выполнения операций, помимо времени выполнения непосредственно вычислений, учитываются временные затраты на передачу данных на вычислительную систему и задержки перед началом выполнения задачи на вычислительной системе (например, вызванные наличием других задач в очереди на выполнение на этой системе).

Заключение

Поставлена проблема разработки информационной системы – инструмента для решения задачи работы с исследовательскими данными полного цикла: сбора данных, их хранения, обработки и анализа, с возможностью построения сценариев обработки и сохранения «истории» данных, формирования и обмена знаниями об обработке данных, как и самими данными на любой стадии работы с ними.

На основе трех «пользовательских историй» сформулирован ряд требований к системе для сбора, хранения и обработки исследовательских данных в ее минимальном жизнеспособном варианте (продукте, MVP – minimal viable product). В двух из этих историй авторы участвовали, разрабатывая простые программные решения для упрощения работы с данными и организации обратной связи системы сбора и обработки данных с участниками эксперимента, третья история является основой для разработки настоящего MVP.

Сформулированные требования позволили разработать следующие модели:

1) модель представления данных, позволяющую исследователям самостоятельно расширять и настраивать описания данных и их связей под нужды текущих исследований и сразу пользоваться внесенными изменениями при поиске и доступе к данным;

2) модель организации доступа к данным, позволяющую автоматизировать передачу данных между этапами обработки;

3) модель абстрактного параллельного вычислителя и язык описания сценариев, что позволяет описывать сценарии обработки данных в функциональном стиле и автоматически генерировать по этому описанию параллельные программы;

4) модель организации распределенной обработки данных на гетерогенном множестве вычислительных узлов.

Кроме того, предложены подходы к накоплению знаний о предметной области, которые будут применяться для поддержки составления новых сценариев. В частности, для этого адаптирована задача об анализе рыночной корзины.

Продолжение работы будет направлено на реализацию разработанной концепции и проектных решений, их детализацию и уточнение в ходе тестирования прототипа MVP, а именно:

1) разработку (или подбор) визуального языка описания сценария;

2) улучшение системы рекомендаций методов на основе статистики работы реальных пользователей;

3) исследование возможностей оптимизации различных этапов работы с данными, в том числе для минимизации задержек при обращении к данным;

4) исследование альтернативных алгоритмов планирования для различных классов сценариев обработки данных, в том числе на основе обратной связи с использованием машинного обучения для предсказания наиболее подходящей схемы планирования;

5) исследование возможностей построения вероятностной модели сценария в конкретной предметной области, чтобы можно было полностью генерировать новый сценарий, фиксируя пользовательские параметры.

Список литературы

1. *Земсков А. И.* Data Curation – хранение научных данных и обслуживание ими – новое направление деятельности библиотек // Научные и технические библиотеки. 2013. № 2. С. 85–101.

2. Программа «Цифровая экономика Российской Федерации»: утв. распоряжением Правительства РФ от 28 июля 2017 г. № 1632-р // СЗ РФ. 2017. № 32, ст. 5138. С. 14517–14574.

3. *Шокин Ю. И.*, *Жижимов О. Л.*, *Пестунов И. А.*, *Синявский Ю. Н.*, *Смирнов В. В.* Распределенная информационно-аналитическая система для поиска, обработки и анализа пространственных данных // Вычислительные технологии. 2007. Т. 12, № S3. С. 108–115.

4. *Новиков А. М., Пойда А. А., Поляков А. Н., Королев С. П., Сорокин А. А.* Разработка технологии и облачной информационной системы для хранения и обработки многомерных массивов научных данных // Информатика и системы управления. 2012. № 4 (34). С. 156–164.

5. *Григорьева М., Голосова М., Рябинкин Е., Климентов А.* Экзабайтное хранилище научных данных // Открытые системы. СУБД. 2015. № 04. С. 14–17.

6. *Юрченко А. В.* К концепции информационно-аналитической системы поддержки научных исследований, основанных на интенсивном использовании цифровых данных // Вычислительные технологии. 2017. Т. 22, № 4. С. 105–120.

7. *Епишев В. В., Исаев А. П., Миниахметов Р. М., Мовчан А. В., Смирнов А. С., Соколинский Л. Б., Цымблер М. Л., Эрлих В. В.* Система интеллектуального анализа данных физиологических исследований в спорте высших достижений // Вестн. ЮУрГУ. Серия «Вычислительная математика и информатика». 2013. Т. 2, № 1. С. 44–54.

8. *Krestyaninova M., Zarins A., Viksna J., Kurbatova N., Rucevskis P., Neogi S. G., Gostev M., Perheentupa T., Knuuttila J., Barrett A. et al.* A system for information management in biomedical studies SIMBioMS // Bioinform. 2009. Vol. 25. Iss. 20. P. 2768–2769.

9. *Austin J., Jackson T., Fletcher M., Jessop M., Liang B., Weeks M., Smith L., Ingram C., Watson P.* CARMEN: code analysis, repository and modeling for e-neuroscience // Procedia Comput. Sci. 2011. Vol. 4. P. 768–777.

10. *Izzo M.* Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management, Springer Theses. Cham, Switzerland: Springer, 2016. 104 p.

11. *Okonechnikov K., Golosova O., Fursov M.* Unipro UGENE: a unified bioinformatics toolkit // Bioinformatics. 2012. № 28. P. 1166–1167. DOI 10.1093/bioinformatics/bts091.

12. *Kotkas V., Ojamaa A., Grigorenko P., Maigre R., Harf M., Tyugu E.* CoCoViLa as a multifunctional simulation platform // Proc. of the 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools '11). Brussels, Belgium: ICST, 2011. P. 198–205.

13. *Huettel S., McCarthy G.* What is odd about the odd-ball task? Prefrontal cortex is activated by dynamic changes in response strategy // Neuropsychologia. 2004. № 42. P. 379–386.

14. *Pascual-Marqui R. D., Lehmann D., Koukkou M., Kochi K., Anderer P., Saletu B., Tanaka H., Hirata K., John E. R., Prichep L., Biscay-Lirio R., Kinoshita T.* Assessing interactions in the brain with exact low-resolution electromagnetic tomography // Philos. Trans. A Math. Phys. Eng. Sci. 2011. № 369 (1952). P. 3768–3784.

15. *Сотников П. И.* Обзор методов обработки сигнала электроэнцефалограммы в интерфейсах мозг-компьютер // Инженерный вестник. 2014, Октябрь. № 10. С. 612–632.

16. *Oostenveld R., Praamstra P.* The five percent electrode system for high-resolution EEG and ERP measurements // Clinical Neurophysiology. 2001. Vol. 112. Iss. 4. P. 713–719. DOI 10.1016/S1388-2457(00)00527-7.

17. *Lei X., Liao K.* Understanding the Influences of EEG Reference: A Large-Scale Brain Network Perspective // Frontiers in Neuroscience. 2017. DOI 10.3389/fnins.2017.00205.

18. *Grandchamp R., Delorme A.* Single-trial normalization for event-related spectral decomposition reduces sensitivity to noisy trials // Frontiers in Psychology. 2011. DOI 10.3389/fpsyg.2011.00236.

19. *Hyvärinen A., Oja E.* Independent Component Analysis: Algorithms and Applications // Neural Networks. 2000. Vol. 13. P. 411–430.

20. *HongLi H., Sun Y. S. Y.* The study and test of ICA algorithms // Proc. 2005 Int. Conf. Wirel. Commun. Netw. Mob. Comput. 2005. Vol. 1. P. 602–605.

21. *Zhu H., Qiu C., Meng Y., Yuan M., Zhang Y., Ren Z., Li Y., Huang X., Gong Q., Lui S., Zhang W.* Altered Topological Properties of Brain Networks in Social Anxiety Disorder: A Resting-state Functional MRI Study // Scientific Reports. 2017. Vol. 7. DOI 10.1038/srep43089.

22. *Stillman P. E., Wilson J. D., Denny M. J., Desmarais B. A., Bhamidi S., Cranmer S. J., Lu Z.-L.* Statistical Modeling of the Default Mode Brain Network Reveals a Segregated Highway Structure // Scientific Reports. 2017. Vol. 7. DOI 10.1038/s41598-017-09896-6.

23. *Abrol A., Damaraju E., Miller R. L., Stephen J. M., Claus E. D., Mayer A. R., Calhoun V. D.* Replicability of time-varying connectivity patterns in large resting state fMRI samples // NeuroImage. 2017. Vol. 163. P. 160–176. DOI 10.1016/j.neuroimage.2017.09.020.

24. *Biswal B., Yetkin F. Z., Haughton V. M., Hyde J. S.* Functional connectivity in the motor cortex of resting human brain using echo-planar MRI // Magn. Reson. Med. 1995. № 34. P. 537–541.

25. *Fielding R. T.* Architectural styles and the design of network-based software architectures. PhD Dissertation. Irvine, US: Dept. of Information and Computer Science, University of California, 2000. P. 76–107.

26. *Martin J.* Managing the Data-base Environment. Englewood Cliffs, New Jersey: Prentice-Hall, 1983. 381 p. ISBN 0135505828.

27. *Linoff G. S., Berry M. J. A.* Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. 3rd ed. John Wiley & Sons, 2004. 643 p.

28. *Koren Y., Bell R., Volinsky C.* Matrix Factorization Techniques for Recommender Systems // Computer. 2009. Vol. 42. Iss. 8. P. 30–37.

29. *Damas L., Milner R.* Principal type-schemes for functional programs // Proc. of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM. 1982. P. 207–212.

30. *Wu Z., Ni Z., Gu L., Liu X.* A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling // Computational Intelligence and Security IEEE International Conference, 2010. P. 184–188.

Материал поступил в редколлегию 01.06.2018

M. A. Gorodnichev [1–3], **A. V. Komissarov** [1, 3], **A. V. Mozhina** [1, 3], **P. V. Prochkin** [1, 3]
P. D. Rudych [1], **A. V. Yurchenko** [1]

[1] *Institute of Computational Technologies SB RAS*
6 Academician Lavrentiev Ave., Novosibirsk, 630090, Russian Federation

[2] *Institute of Computational Mathematics and Mathematical Geophysics SB RAS*
6 Academician Lavrentiev Ave., Novosibirsk, 630090, Russian Federation

[3] *Novosibirsk State University*
1 Pirogov Str., Novosibirsk, 630090, Russian Federation

maxim@ssd.sscc.ru, yurchenko@ict.nsc.ru

INFORMATION MODELS AND PROJECT SOLUTIONS FOR THE ECCLESIA RESEARCH DATA STORING AND PROCESSING SYSTEM

Scientific research produces a lot of digital data that should be carefully gathered and stored for further usage: processing, analysis and publication. Building e-infrastructure for that is one of the most topical problems of IT (or digital) curation of science. Starting from three data-processing problems in physiology we are developing an information system for automation of gathering, stor-

ing and analyzing data. Problems encountered in development of such a system are examined and analyzed, along with existing approaches and software solutions related to these problems.

Based on results of the conducted analysis a number of models and mechanisms for solving encountered problems are proposed. Developed solutions include models and mechanisms for collecting and storing research data, a model describing and formalizing data processing scenarios and models and mechanisms for processing collected data in a distributed computer system.

As a result, an architecture for a computer system for collecting, storing and processing research data is presented. The system is proposed as a tool for solving a wide spectrum of problems in scientific research involving collecting and multi-step processing of various kinds of data.

Keywords: research data management, information system, data analysis and processing, physiology data, object storage, functional language, machine learning, distributed computing.

References

1. Zemskov A. Data Curation – a new derection in library activities. *Scientific and Technical Libraries*, 2013, vol. 2, p. 85–101. (in Russ.)

2. Program «Digital economy of the Russian Federation»: approved by the Russian Federation Government order of July 28, 2017 № 1632-p. *Collection of legislation of the Russian Federation*, 2017, vol. 32, art. 5138, p. 14517–14574. (in Russ.)

3. Shokin Y. I., Zhizhimov O. L., Pestunov I. A., Sinyavskiy Y. N., Smirnov V. V. Distributed informational-analytical system for searching, processing and analysis of spatial data. *Computational Technologies*, 2007, vol. 12, special issue 3, p. 108–115. (in Russ.)

4. Novikov A. M., Poyda A. A., Korolev S. G., Sorokin A. A. Development of Technology and Cloud Informational System for Storing and Processing of Multi-dimensional Science Data Arrays. *Information Science and Control Systems*, 2012, № 4 (34), p. 156–164. (in Russ.)

5. Grigor'eva M., Golosova M., Ryabinkin E., Klimentov A. Ekzabaitnoe hranilishe nauchnyh dannyh. *Open Systems Journal*, 2015, № 4, p. 14–17. (in Russ.)

6. Yurchenko A. V. On the concept of information-analytical system for supporting data intensive science. *Computational Technologies*, 2017, vol. 22, № 4, p. 105–120. (in Russ.)

7. Epishev V. V., Isaev A. P., Miniakhmetov R. M., Movchan A. V., Smirnov A. S., Sokolinsky L. B., Zymbler M. L., Ehrlich V. V. Physiological data mining system for elite sports. *Bulletin of the South Ural State University. Series «Computational Mathematics and Software Engineering»*, 2013, vol. 2, № 1, p. 44–54. (in Russ.)

8. Krestyaninova M., Zarins A., Viksna J., Kurbatova N., Rucevskis P., Neogi S. G., Gostev M., Perheentupa T., Knuuttila J., Barrett A. et al. A system for information management in biomedical studies SIMBioMS. *Bioinform*, 2009, vol. 25 (20), p. 2768–2769.

9. Austin J., Jackson T., Fletcher M., Jessop M., Liang B., Weeks M., Smith L., Ingram C., Watson P. CARMEN: code analysis, repository and modeling for e-neuroscience. *Procedia Comput. Sci.*, 2011, vol. 4, p. 768–777.

10. Izzo M. Biomedical Research and Integrated Biobanking: An Innovative Paradigm for Heterogeneous Data Management, Springer Theses. Cham, Switzerland: Springer, 2016. 104 p.

11. Okonechnikov K., Golosova O., Fursov M., the UGENE team. Unipro UGENE: a unified bioinformatics toolkit. *Bioinformatics*, 2012, vol. 28, p. 1166–1167. DOI 10.1093/bioinformatics/bts091.

12. Kotkas V., Ojamaa A., Grigorenko P., Maigre R., Harf M., Tyugu E. CoCoViLa as a multifunctional simulation platform. Proc. of the 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools '11). ICST. Brussels, Belgium, ICST, 2011, p. 198–205.

13. Huettel S., McCarthy G. What is odd about the odd-ball task? Prefrontal cortex is activated by dynamic changes in response strategy. *Neuropsychologia*, 2004, vol. 42, p. 379–386.

14. Pascual-Marqui R. D., Lehmann D., Koukkou M., Kochi K., Anderer P., Saletu B., Tanaka H., Hirata K., John E. R., Prichep L., Biscay-Lirio R., Kinoshita T. Assessing interactions in the brain with exact low-resolution electromagnetic tomography. *Philos. Trans. A Math. Phys. Eng. Sci.*, 2011, vol. 369 (1952), p. 3768–3784.

15. Sotnikov P. I. Obzor metodov obrabotki signala elektroencefalogrammy v interfeisah mozg-komputer. *Engineering Bulletin*, 2014, vol. 10, p. 612–632. (in Russ.)

16. Oostenveld R., Praamstra P. The five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology*, 2001, vol. 112 (4), p. 713–719. DOI 10.1016/S1388-2457(00)00527-7.

17. Lei X., Liao K. Understanding the Influences of EEG Reference: A Large-Scale Brain Network Perspective. *Frontiers in Neuroscience*, 2017. DOI 10.3389/fnins.2017.00205.

18. Grandchamp R., Delorme A. Single-trial normalization for event-related spectral decomposition reduces sensitivity to noisy trials. *Frontiers in Psychology*, 2011. DOI 10.3389/fpsyg.2011.00236.

19. Hyvärinen A., Oja E. Independent Component Analysis: Algorithms and Applications. *Neural Networks*, 2000, vol. 13, p. 411–430.

20. HongLi H., Sun Y. S. Y. The study and test of ICA algorithms. *Proc. Int. Conf Wirel. Commun. Netw. Mob. Comput*., 2005, vol. 1, p. 602–605.

21. Zhu H., Qiu C., Meng Y., Yuan M., Zhang Y., Ren Z., Li Y., Huang X., Gong Q., Lui S., Zhang W. Altered Topological Properties of Brain Networks in Social Anxiety Disorder: A Resting-state Functional MRI Study. *Scientific Reports*, 2017, vol. 7. DOI 10.1038/srep43089.

22. Stillman P. E., Wilson J. D., Denny M. J., Desmarais B. A., Bhamidi S., Cranmer S. J., Lu Z.-L. Statistical Modeling of the Default Mode Brain Network Reveals a Segregated Highway Structure. *Scientific Reports*, 2017, vol. 7. DOI 10.1038/s41598-017-09896-6.

23. Abrol A., Damaraju E., Miller R. L., Stephen J. M., Claus E. D., Mayer A. R., Calhoun V. D. Replicability of time-varying connectivity patterns in large resting state fMRI samples. *NeuroImage*, 2017, vol. 163, p. 160–176. DOI 10.1016/j.neuroimage.2017.09.020.

24. Biswal B., Yetkin F. Z., Haughton V. M., Hyde J. S. Functional connectivity in the motor cortex of resting human brain using echo-planar MRI. *Magn. Reson. Med.*, 1995, vol. 34, p. 537–541.

25. Fielding R. T. Architectural styles and the design of network-based software architectures. PhD Dissertation. Irvine, US: Dept. of Information and Computer Science, University of California, 2000, p. 76–107.

26. Martin J. Managing the Data-base Environment. Englewood Cliffs, New Jersey: Prentice-Hall, 1983, 381 p.

27. Linoff G. S., Berry M. J. A. Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. 3rd ed. John Wiley & Sons, 2004, 643 p.

28. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems. *Computer*, 2009, vol. 42 (8), p. 30–37.

29. Damas L., Milner R. Principal type-schemes for functional programs. *Proc. of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, 1982, p. 207–212.

30. Wu Z., Ni Z., Gu L., Liu X. A Revised Discrete Particle Swarm Optimization for Cloud Workflow Scheduling. *Computational Intelligence and Security IEEE International Conference*, 2010, p. 184–188.

For citation:

Gorodnichev M. A., Komissarov A. V., Mozhina A. V., Prochkin P. V., Rudych P. D., Yurchenko A. V. Information Models and Project Solutions for the Ecclesia Research Data Storing and Processing System. *Vestnik NSU. Series*: *Information Technologies*, 2018, vol. 16, no. 3, p. 87–104. (in Russ.)

DOI 10.25205/1818-7900-2018-16-3-87-104