DOI: 10.18721/JCSTCS.11406
УДК 004.3

АРХИТЕКТУРА СИСТЕМЫ СБОРА И ОБРАБОТКИ ДАННЫХ ДЛЯ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ ТРУДА РАЗРАБОТЧИКОВ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Д.А. Тимофеев, М.Ю. Маслов

Санкт-Петербургский политехнический университет Петра Великого,
Санкт-Петербург, Российская Федерация

Для повышения производительности труда работников необходимо идентифицировать и оптимизировать выполняемые производственные процессы с учетом факторов, влияющих на их выполнение. В области разработки программного обеспечения, как и в других сферах с преобладанием интеллектуальной работы, многие процессы остаются неявными и существенно различаются у разных сотрудников, а результаты и производительность труда существенно зависят от личных качеств и текущего психофизиологического состояния исполнителя. Для получения необходимой информации предложено использовать систему, выполняющую сбор и обработку данных о состоянии человека и выполняемых им действиях, и на их основе формирующую обратную связь, позволяя работнику более эффективно планировать и решать задачи с учетом его состояния и характерных для него рабочих процессов. Описаны архитектура системы и основные технические решения, принятые при ее проектировании.

Ключевые слова: архитектура программной системы, сбор данных, обработка данных, обратная связь, производительность труда.

Ссылка при цитировании: Тимофеев Д.А., Маслов М.Ю. Архитектура системы сбора и обработки данных для повышения производительности труда разработчиков программного обеспечения // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. 2018. Т. 11. № 4. С. 71–81. DOI: 10.18721/JCSTCS.11406.

ARCHITECTURE OF DATA ACQUISITION AND PROCESSING SYSTEM FOR IMPROVING PRODUCTIVITY OF SOFTWARE DEVELOPERS

D.A. Timofeev, M.Yu. Maslov

Peter the Great St. Petersburg Polytechnic University,
St. Petersburg, Russian Federation

In order to enhance the workers productivity, it is necessary to identify and optimize the workflows with respect to main influencing factors. In software development industry, as well as in other areas with prevailing intellectual work, these workflows are often implicit and highly variable, and the product quality and worker performance to the large extent depend on the worker current psychophysical state. We propose to regain missing information by using a system to acquire and process the data about the developers' states and activities. To generate the data, portable sensors and computer software plugins are used. Based on the data, the system generates a feedback allowing the users to better plan and complete tasks with respect to their state and personal work processes. In this paper we present the system architecture and describe main technical decisions.

Keywords: software system architecture, data acquisition, data processing, feedback, productivity.

Citation: Timofeev D.A., Maslov M.Yu. Architecture of data acquisition and processing system for improving productivity of software developers. St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunications and Control Systems, 2018, Vol. 11, No. 4, Pp. 71–81. DOI: 10.18721/JCSTCS.11406.

Введение

Во многих сферах производства основную роль играет интеллектуальная деятельность. Выполняемые специалистом процессы в этих областях обычно мало формализованы, а результаты и эффективность труда существенно больше зависят от личных качеств и текущего психофизиологического состояния работника. Отсутствие стандартизированных технологических процессов производства также приводит к трудностям в обучении новых сотрудников и распространении эффективных практик, снижает точность планирования работ и увеличивает зависимость компании от конкретных исполнителей.

Одной из важных отраслей производства, в которой интеллектуальный труд играет главную роль, является разработка программного обеспечения. Существенная особенность этой области — одновременная реализация многих взаимосвязанных процессов на разных уровнях, от управления проектом до непосредственного создания программного кода. Эти процессы можно разделить на два вида: групповые процессы, в которые вовлечены несколько участников, и индивидуальные процессы, выполняемые одним разработчиком.

Групповые процессы достаточно хорошо исследованы. К ним относятся, в частности, управление проектом и организация коллективной работы над проектом. Каждая методология управления программным проектом предполагает определенный набор этапов разработки системы, набор создаваемых артефактов и порядок взаимодействия разработчиков. Например, при использовании методологии Scrum [1] процесс разработки разделяется на итерации фиксированной длительности (спринты). Для каждого спринта в ходе планирования определяется цель и набор задач, в ходе спринта изменение состояния каждой задачи отмечается в доступном для всех участников виде, а по завершению спринта проводятся обзор выполнения задач и ретроспектива, цель которой — анализ хода выполнения спринта на основе обратной связи от разработчиков.

Индивидуальные процессы можно разделить на два типа. К первому типу относятся процессы, обусловленные используемой методологией управления проектом (например, изменение состояния выполняемых задач в системе управления задачами) или применяемыми инструментальными средствами (взаимодействие с системой контроля версий, тестирование и компиляция проекта). Хотя эти процессы в большинстве случаев формально не описываются, они могут быть смоделированы на основе существующих регламентов, описаний инструментальных средств или результатов наблюдения за действиями разработчиков. Ко второму типу можно отнести скрытые творческие процессы, такие как проектирование структур данных, разработка архитектуры и алгоритмов, написание программного кода, создание тестовых планов, написание документации. Именно эти процессы определяют в итоге свойства разрабатываемого программного обеспечения. При этом анализа итоговых артефактов (кода, данных, документации) и наблюдения за ходом их создания недостаточно для выявления и описания правил, которыми руководствуются разработчики, и выполняемых ими ментальных процессов.

Таким образом, индивидуальные творческие процессы, играя главную роль в создании программного продукта, оказываются одновременно наименее изученными. Несмотря на то, что многие авторы работали над созданием методик эффективного решения творческих задач определенных классов (изобретательство [2], проектирование систем [3], решение математических задач [4]), в области разработки программ на данный момент подобные методики не получили заметного распространения. Основными способами передачи знаний о способах разработки программ является обмен опытом с помощью книг, докладов на профессиональных конференциях, публикаций в социальных сетях или личного общения. При этом обсуждаемые подходы часто не представляются в виде методики, а демонстрируются на конкретных приме-

рах кода или программных проектах (в качестве исключения можно указать [5], где автор предлагает методику целенаправленного поиска алгоритма с учетом свойств задачи и существующих ограничений). Используемые при решении задач подходы существенно различаются в зависимости от используемых технологий, опыта и стиля мышления разработчика [6].

Индивидуальные процессы также играют важную роль при выполнении групповых процессов. К примеру, для организации коммуникации между участниками проекта нужно учитывать такие факторы, как канал связи (личная встреча, телефон, электронная почта, мессенджер) и график занятости каждого из участников. Выбор канала связи предполагает различные методики эффективного управления временем (например, звонок по телефону отвлекает работника от выполняемой задачи, в то время как сообщение по электронной почте может быть отложено до удобного или специально выделенного для этой цели момента времени). Помимо этого, важно текущее состояние работника. Известной проблемой при организации интеллектуальной деятельности является потеря работниками концентрации при прерывании выполняемой ими деятельности. Систематическое исследование этого явления было предпринято М. Чиксентмихайи, который ввел понятие потока [7] – оптимального состояния внутренней мотивации, в котором человек полностью сконцентрирован на выполняемой задаче. В состоянии потока работа выполняется максимально эффективно, однако в случае прерывания деятельности возврат в состояние потока требует существенного времени. Экспериментальные результаты демонстрируют, что состояние потока можно идентифицировать путем анализа электроэнцефалограммы (ЭЭГ) человека [8].

Психофизиологические показатели также могут использоваться для анализа других факторов, связанных с производительностью труда разработчиков программного обеспечения – в частности, для оценки эмоционального состояния [9] (с использованием таких показателей, как электроэнцефалограмма, температура тела, частота сердечных сокращений, мышечная активность, частота дыхания) или субъективной оценки трудности задачи [10] (в данном случае использовались движения глаз, электрическая активность кожи и ЭЭГ).

Таким образом, индивидуальные процессы оказывают принципиальное влияние на производительность труда как отдельных программистов, так и всего коллектива разработчиков, но при этом остаются относительно малоизученными. При этом выполняемые процессы одновременно проявляются в виде конкретных выполняемых разработчиком действий и оказывают существенное влияние на изменение психофизиологического состояния человека. Анализ этих данных может дать новые сведения о процессах интеллектуальной деятельности человека, а также предоставить новые возможности для повышения эффективности интеллектуального труда в области разработки программного обеспечения за счет оценки текущего состояния разработчиков и формирования обратной связи для предотвращения перехода в нежелательные состояния и поддержки необходимой активности. Для организации сбора и анализа данных разработан аппаратно-программный комплекс, архитектура которого описывается в данной статье.

Функциональность аппаратно-программного комплекса

Разработанный аппаратно-программный комплекс предназначен для выполнения следующих функций.

1. Сбор данных о деятельности пользователя: ввод данных, запуск и завершение процессов, операции в среде разработки программ, действия в браузере.

2. Регистрация электрофизиологических и биологических параметров человека, позволяющих оценить его психофизиологическое состояние. В качестве базовых показателей выбраны электроэнцефалограмма (ЭЭГ), двигательная активность, пульс, температура тела. Для формирования исходных данных используются как

бытовые датчики (например, неинвазивные нейроинтерфейсы Emotiv Insight, MUSE; фитнес-браслет Mio Link), так и специально разработанное в рамках АПК оборудование.

3. Потоковый анализ данных в соответствии с набором моделей и формирование обратной связи с пользователем: подача визуальных сигналов о текущем психофизиологическом состоянии или о заданных событиях, выдача рекомендаций о режиме труда и отдыха, мерах по повышению эффективности работы.

4. Передача данных на сервер более высокого уровня для анализа состояния группы людей и формирования групповых рекомендаций.

5. Сохранение истории событий в центральной базе данных и визуализация истории событий отдельных пользователей и групп.

6. Аутентификация и авторизация пользователей, управление правами доступа и разрешениями на сбор данных каждого конкретного вида, их передачу за пределы локального устройства пользователя (компьютера, смартфона, планшета) и сохранение в центральной базе данных.

В состав комплекса входят датчики, измеряющие значения физиологических показателей, и программная система, обеспечивающая сбор, хранение и обработку данных, поступающих от аппаратных компонентов.

Функционирование системы осуществляется путем сбора и анализа данных о состоянии и действиях пользователя. Каждый пользователь может самостоятельно определить, какие именно данные разрешено собирать, и дать или отозвать разрешение на передачу и сохранение этих данных. Анализ данных осуществляется как для каждого отдельного пользователя на его локальных устройствах и, при наличии разрешения, на удаленных серверах, так и для групп пользователей. В результате анализа пользователи получают обратную связь от системы.

Использование системы может увеличивать производительность труда при выполнении интеллектуальной деятельности двумя путями.

1. Потоковый анализ данных о значениях физиологических показателей пользователя и его действиях, определение его психофизиологического состояния (в частности, сосредоточенность, высокая работоспособность, усталость, стресс) и формирование рекомендаций (например, сделать перерыв, выполнить физические или психологические упражнения, изменить вид деятельности, завершить рабочий день, обратиться к врачу).

2. Накопление данных о деятельности сотрудника в течение долгого времени, моделирование выполняемых им на индивидуальном и групповом уровне процессов, анализ и визуализация этих процессов.

Потоковый анализ данных позволяет определять желательные (удовлетворенность, комфорт, высокая работоспособность) и нежелательные состояния (усталость, стресс, болезнь) для каждого работника и оперативно выдавать рекомендации на основе предопределенных общих и персонализированных моделей. В частности, учитываются когнитивная нагрузка [11] и стрессогенные факторы [12], предполагается также анализ циркадных ритмов [13] и характера выполняемой деятельности [14]. Такой подход позволит своевременно идентифицировать и устранять или учитывать факторы, ведущие к снижению эффективности работы и удовлетворения от нее, а также может быть полезен для раннего обнаружения состояний, в которых может потребоваться медицинское вмешательство. Потоковый анализ предусмотрен не только для отдельных пользователей, но и для групп, что позволяет, например, применять этот подход при проведении собраний или организации обучения: ведущий может оперативно получать информацию о степени концентрации внимания и усталости участников и с учетом этих данных менять скорость подачи информации или планировать расписание.

Анализ процессов позволяет выделять и классифицировать виды деятельности

специалистов, связывать эти модели с измеряемыми показателями эффективности сотрудника и за счет этого идентифицировать более эффективные и менее эффективные процессы с учетом личных особенностей каждого специалиста. Наличие таких моделей позволит уточнять оценки времени для выполнения задач, упрощать обучение новых и менее опытных сотрудников, формировать рекомендации по улучшению производительности труда, а на групповом уровне – настраивать систему коммуникации и оптимизировать параметры процессов, связанных с управлением проектом.

Помимо этого, каждому конкретному специалисту изучение данных об истории собственных состояний и действий может дать информацию для развития и самосовершенствования. В этом смысле особенно важно, чтобы сбор данных не доставлял участникам неудобств, а любая обработка и хранение данных происходили только с их добровольного и осознанного согласия. Нарушение этих принципов приведет лишь к снижению эффективности работников и нежеланию использовать систему.

Архитектура системы

Компоненты АПК представлены на рис. 1. Все программное обеспечение делится на клиентские компоненты, которые выполняются на компьютере или мобильном устройстве пользователя, и серверные компоненты, которые могут быть развернуты в локальной сети организации или размещены на одном или нескольких серверах в сети Интернет.

На устройствах пользователей выполняются локальные агенты – приложения, обеспечивающие сбор, временное хранение и обработку биометрических и иных данных пользователя. Сбор и первичную обработку данных каждого вида выполняют встраиваемые модули. Каждый из модулей может выступать в роли генератора событий или процессора событий.

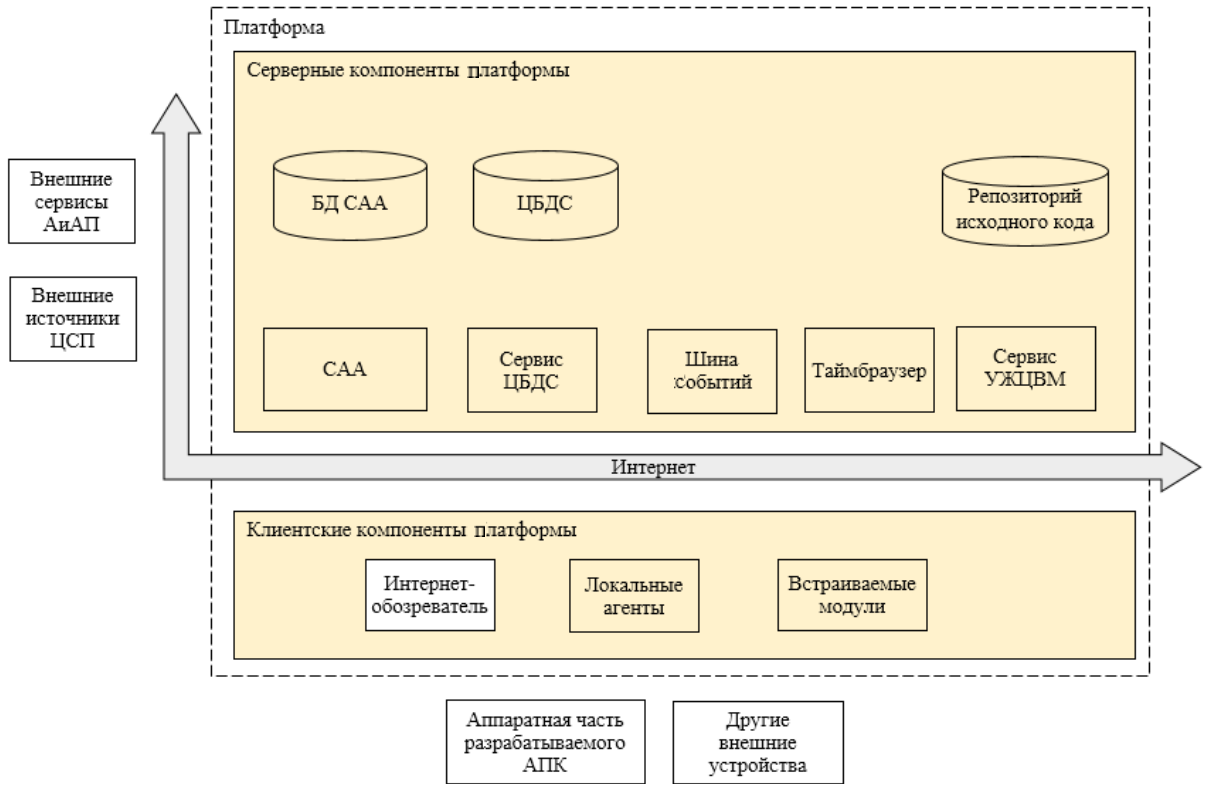

Рис. 1. Компоненты аппаратно-программного комплекса
Fig. 1. Components of hardware and software complex

Генераторы событий порождают данные в ходе взаимодействия с устройствами, операционной системой или приложениями. Каждый элемент данных помечается набором из трех идентификаторов, соответствующих группе, классу и типу события. Например, все события, связанные с психофизиологическими показателями человека, могут относиться к группе bio, их подмножество, соответствующее данным ЭЭГ – к классу eeg, а данные ЭЭГ с электрода AF3 – к типу AF3. Эти идентификаторы используются для выбора подмножества событий, на которые другие модули и локальные агенты осуществляют подписку. Каждый генератор порождает наборы событий и связанных с событиями данных. В простейшем случае событием считается период времени от активации до деактивации модуля. Данные о событиях представляют собой полученные от устройства в этот период значения, снабженные идентификатором соответствующего события и метками времени.

Процессоры событий подписываются на события определенного вида, читают данные о событиях, обрабатывают их и формируют новые события с привязанными к ним результатами обработки. Процессоры событий могут фильтровать данные (например, устраняя выбросы), вычислять статистические характеристики, преобразовывать данные. Важным процессором событий является триггер, формирующий событие при получении удовлетворяющих заданному условию данных.

Каждый встраиваемый модуль может быть загружен в нескольких экземплярах, каждый экземпляр имеет собственную конфигурацию. Набор параметров конфигурации зависит от конкретного модуля: процессоры событий обычно будут иметь параметры группы, класса и типа для фильтрации поступающих на вход данных, а модули для работы с устройствами – MAC-адрес или другой идентификатор устройства.

Наибольшее количество модулей поддерживает локальный агент для ОС Windows. Для его создания использовался язык программирования C#. Встраиваемые модули представляют собой динамически загружаемые библиотеки и могут быть написаны на любом языке программирования, поддерживающем эту технологию. Большая часть существующих модулей также разработана на языке C#, часть модулей – на языке C++.

Серверная часть включает сервер аутентификации и авторизации (САА) и его базу данных, центральную базу данных событий (ЦБДС) и сервис для работы с ней, шину событий, сервис «таймбраузер», а также сервис управления жизненным циклом встраиваемых модулей (УЖЦВМ) и репозиторий исходного кода модулей.

Сервер аутентификации и авторизации обеспечивает аутентификацию пользователей, авторизацию от имени пользователя по протоколу OAuth2 и хранение пользовательских профилей. Сервер реализован на языке программирования PHP7 с использованием фреймворка Laravel. База данных реализована с использованием СУБД PostgreSQL 9.6.

Центральная база данных событий и шина событий обеспечивают взаимодействие серверной части АПК и локальных агентов. Локальные агенты подключаются к шине событий и передают сообщения о событиях и о связанных с ними данных в формате JSON. Для сокращения количества передаваемых пакетов и нагрузки как на сервере, так и на клиенте, данные о событиях группируются в пакеты настраиваемой длины.

Шина событий передает полученные пакеты сервису центральной базы данных событий, который сохраняет их в базу данных. Для организации групповой работы локальные агенты также могут подписаться на получение данных о событиях из шины событий. Шина событий передает полученные пакеты с соответствующей маской (группа, класс, тип) всем подписавшимся на них агентам. Сервис ЦБДС также предоставляет авторизованным пользователям программный интерфейс для доступа к архивным данным. Алгоритм сбора и накопления данных представлен на рис. 2.

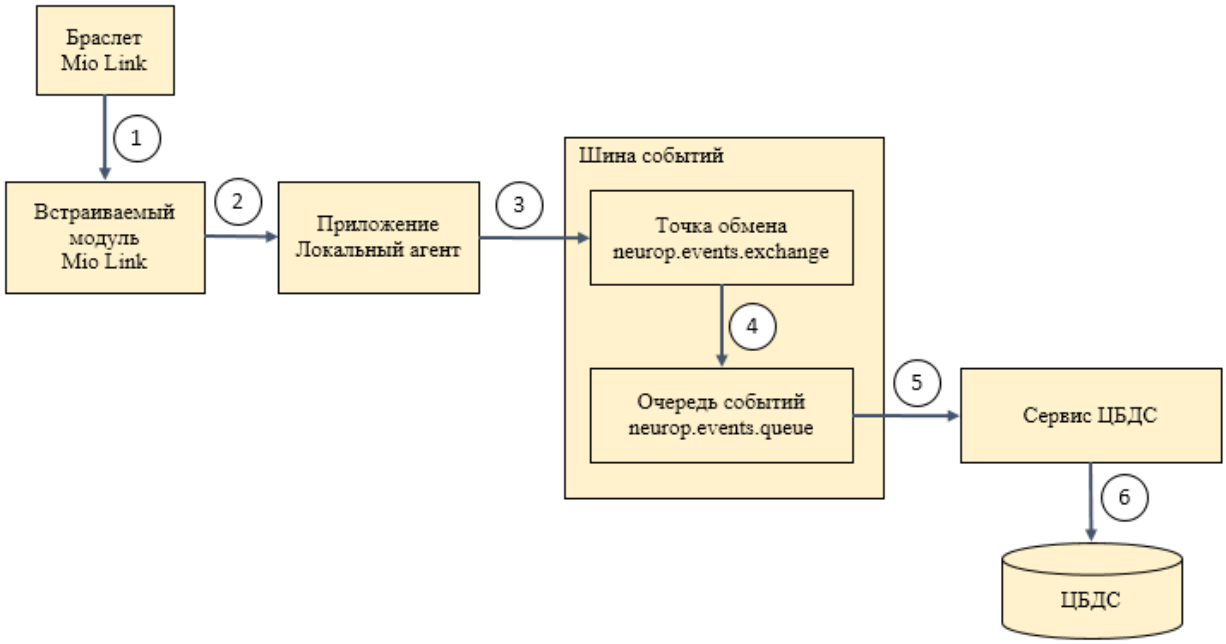

Рис. 2. Порядок обмена собираемыми данными между компонентами системы
Fig. 2. The procedure for the exchange of collected data between system components

Рисунок 3 иллюстрирует порядок распространения данных между локальными агентами с использованием шины событий. В изображенном примере локальный агент-приемник данных подписывается на события группы *g*1, класса *c*1 и типа *t*1.

Для реализации шины событий используется очередь сообщений RabbitMQ. Центральная база данных событий использует СУБД PostgreSQL 9.6, сервис ЦБДС реализован на языке программирования C# с использованием кроссплатформенной среды .NETCore.

Сервис управления жизненным циклом встраиваемых модулей используется для поддержки разработчиков, создающих и поддерживающих встраиваемые модули локальных агентов. Сервис реализует:

репозиторий исходного кода со встроенными средствами управления версиями;

витрину модулей;

панель разработчика;

подсистему коммуникации разработчиков и пользователей встраиваемых модулей.

Репозиторий исходного кода построен на базе сервера управления репозиториями Gogs и, как и остальные компоненты, использует единый сервер аутентификации и авторизации. Сервис управления жизненным циклом встраиваемых модулей обеспечивает управление зависимостями, позволяя формировать сборки, автоматически включающие все модули, необходимые для работы выбранного модуля.

Сервис «таймбраузер» представляет собой инструмент для формирования, просмотра и анализа хронологических последовательностей событий и связанных с ними данных. Таймбраузер представляет собой инструмент для изучения аналитиками выполняемых пользователями процессов, а также для просмотра пользователями истории своих событий.

Альтернативные подходы

Задача сбора и анализа информации о действиях и текущем состоянии человека возникает не только при анализе процессов интеллектуальной профессиональной деятельности, но и во многих смежных областях. Двумя наиболее близкими к представленному проекту являются задачи мониторинга здоровья и протоколирования рабочего времени.

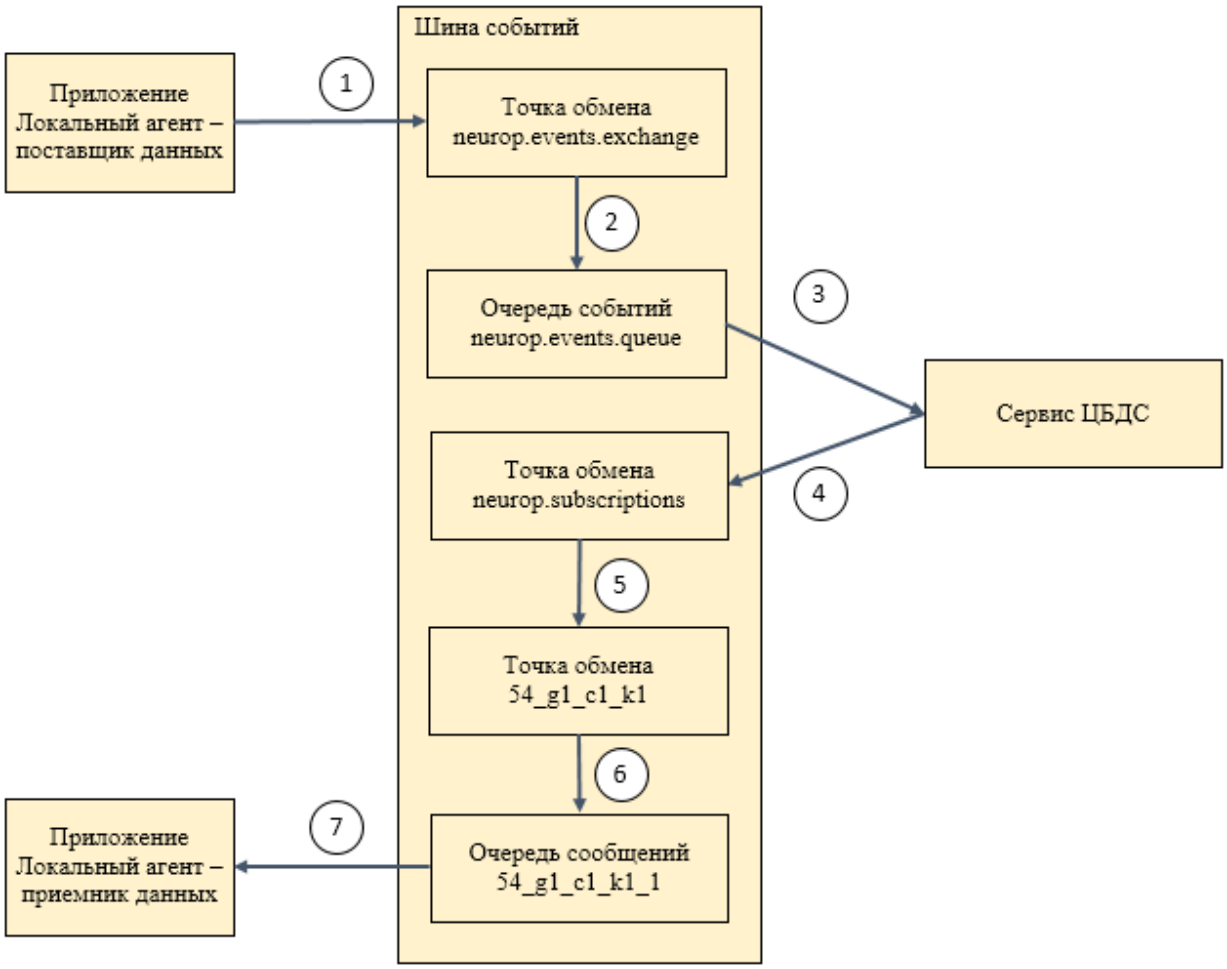

Рис. 3. Порядок распространения данных между локальными агентами
Fig. 3. The order of data distribution between local agents

Системы мониторинга физиологических параметров организма человека успешно используются для оценки и отслеживания динамики показателей здоровья и физического состояния (включая мониторинг показателей во время спортивных тренировок), ранней диагностики заболеваний и автоматического вызова помощи в экстренных случаях. В последние годы активные исследования ведутся в области мониторинга состояния здоровья с учетом контекста [15]. Задачей контекстно-зависимого мониторинга является сопоставление значений физиологических показателей с данными о контексте, в котором снимаются показания, такими как местоположение, время, вид или другие участники выполняемой деятельности. Так как понятие контекста может быть определено произвольным образом, контекстно-зависимые системы обычно строятся для решения конкретных задач с фиксированным набором учитываемых факторов и их связей с целевыми показателями. Например, при диагностике сердечных заболеваний виды выполняемой человеком деятельности могут быть сведены к нескольким классам, отличающимся степенью физической нагрузки [15].

Протоколирование выполняемых задач и затрат рабочего времени на их выполнение – важная составляющая работы для специалистов, работающих на условиях почасовой оплаты, а также распространенный инструмент для анализа и повышения эффективности использования времени. Как

следствие, существует большое количество прикладных программ и веб-сервисов для протоколирования рабочего времени. Их можно разделить на две основные группы. В первую входят сервисы, в которых сведения о выполняемых задачах вводит сам пользователь. Многие из них предоставляют таймер, который пользователь может запускать в начале выполнения задачи и отключать после ее завершения. Основной целью таких систем является формирование отчетов и счетов на основе собранных данных. Примерами систем этого класса являются сервисы Toggl[1] и Zoho Invoice[2]. Системы второго класса протоколируют выполняемые пользователем компьютера действия и используют собранные данные для автоматической идентификации вида деятельности. Основное применение такого подхода – отслеживание деятельности работников организации, не связанных с выполнением их должностных обязанностей, или для контроля счетов, выставляемых фрилансерами заказчикам; возможно также применение этих сервисов для формирования счетов и оценки собственной производительности труда. В качестве исходных данных могут служить сведения об используемых и запущенных приложениях, открытых вкладках браузеров, применении устройств ввода, периодически формируемые изображения экрана или кадры с вебкамеры. Примерами таких систем являются сервис Desk Time[3] и приложение фриланс-биржи Upwork[4] [16]. Ряд сервисов, таких как Waka Time[5] или CodeAlike[6], ориентированы на разработчиков программного обеспечения и предоставляют модули расширения для основных сред разработки программ и веб-браузеров. На основе собираемых данных эти сервисы предоставляют пользователю отчеты о затраченном времени и других показателях (например, о количестве написанных строк кода или долях времени, затраченных на кодирование и отладку) в каждом из выполняемых проектов, а также реализуют игровые приемы (достижения, соревнования) для повышения производительности труда.

Помимо перечисленных задач, протоколирование и анализ различных аспектов поведения людей используются при обеспечении безопасности (поиск аномалий в системах видеонаблюдения [17], обнаружение и предотвращение мошенничества [18]), лингвистике и когнитивных науках (исследование процессов письма и чтения [19, 20]), эргономике (анализ и проектирование пользовательских интерфейсов), рекламе и торговле (рекомендательные системы, персонализированные предложения) и многих других областях.

Применяемые в описанных областях решения являются специализированными. Они ориентированы на эффективную работу с данными, набор которых определяется конкретной решаемой задачей. В частности, существует четкое разделение между системами, направленными на работу с данными о значениях физиологических показателей, и системами учета рабочего времени. Описанная в данной статье архитектура системы сбора и анализа данных о состоянии и деятельности является открытой и расширяемой. Возможность создания собственных генераторов и процессоров событий, а также отсутствие принципиальных ограничений на вид данных о событиях позволяют использовать представленную систему не только для обеспечения работы конкретного алгоритма формирования обратной связи для повышения эффективности труда, но и в качестве исследовательской платформы для изучения процессов интеллектуальной деятельности человека. В частности, поскольку в качестве основного вида деятельности выбрана разработка программного обеспечения, собираемые с помощью системы данные могут использоваться для анализа процессов создания и модификации исходного кода программ [21].

[1] https://toggl.com
[2] https://www.zoho.eu
[3] https://Desk Time.com
[4] https://upwork.com
[5] https://Waka Time.com
[6] https://codealike.com

Заключение

В данной статье представлена архитектура системы сбора и анализа данных о психофизиологическом состоянии человека и выполняемых им в ходе интеллектуальной деятельности процессах, входящая в состав аппаратно-программного комплекса для регистрации и анализа электрофизиологических и биометрических параметров человека и предоставления биологической и оптической обратной связи. Предложенная архитектура является открытой и расширяемой и может использоваться для формирования массива данных для исследований в области процессов профессиональной деятельности человека, эргономии и других областях, в которых важен учет контекста в наблюдаемых действиях. Непосредственная прикладная задача, которую решает система, состоит в подготовке данных, позволяющих классифицировать текущее состояние работника и формировать на его основе воздействия, помогающие избегать нежелательных состояний. В настоящее время ведется настройка параметров алгоритмов формирования обратной связи для специалистов, занятых в области разработки программного обеспечения.

Работа подготовлена в ходе реализации проекта в рамках Постановления Правительства РФ от 09.04.2010 № 218 при финансовой поддержке Министерства образования и науки РФ. Договор № 03.G25.31.0247 от 28.04.2017.

СПИСОК ЛИТЕРАТУРЫ

1. **Schwaber K., Sutherlang J.** The Scrum Guide. 2017 // URL: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf (Дата обращения: 08.11.2018).

2. **Альтшуллер Г.С.** Алгоритм изобретения. М.: Московский рабочий, 1969. 272 с.

3. **Джонс Дж.К.** Методы проектирования. 2-е изд. Пер. с англ. М.: Мир, 1986. 326 с.

4. **Пойа Д.** Математическое открытие. 2-е изд. Пер. с англ. М.: Наука, 1976. 448 с.

5. **Скиена С.** Алгоритмы. Руководство по разработке. 2-е изд. Пер. с англ. СПб.: БХВ-Петербург, 2011.

6. **Carter C. Sangler**. The programmers stone, 1997 // URL: https://www.datapacrat.com/Opinion/Reciprocality/r0/index.html (Дата обращения: 10.11.2018).

7. **Чиксентмихайи М.** Поток: Психология оптимального переживания. Пер. с англ. 7-е изд. М.: Смысл; Альпина нон-фикшн, 2017.

8. **Berta R., Bellotti F., De Gloria A., Pranantha D., Schatten C.** Electroencephalogram and physiological signal analysis for assessing flow in games // IEEE Transactions on Computational Intelligence and AI in Games 5. 2013. No. 2. Pp. 164–175.

9. **Kolakowska A., Landowska A., Szwoch M., Szwoch W., Wrobel M.R.** Emotion recognition and its application in software engineering // 6th Internat. Conf. on Human System Interactions. 2013. Pp. 532–539.

10. **Fritz T., Begel A., Мыller S.C., Yigit-Elliott S., Züger M.** Using psycho-physiological measures to assess task difficulty in software development // Proc. of the 36th Internat. Conf. on Software Engineering. 2014. Pp. 402–413.

11. **Sweller J.** Cognitive load during problem solving: Effects on learning // Cognit. Sci. 1988. No. 12. Pp. 257–285.

12. **Chilton M.A., Hardgrave B.C., Armstrong D.J.** Person-Job cognitive style fit for software developers: The effect on strain and performance // J. of Management Information Systems. 2014. No. 22:2. Pp.193–226.

13. **Blatter K., Cajochen C.** Circadian rhythms in cognitive performance: Methodological constraints, protocols, theoretical underpinnings // Physiology and Behaviour. 2006. Pp. 196–208.

14. **Sawyer K.** The cognitive neuroscience of creativity: A critical review // Creativity Research Journal. 2011. No. 23:2. Pp. 137–154.

15. **Mshali H., Lemlouma T., Moloney M., Magoni D.** A survey on health monitoring systems for health smart homes // Internat. J. of Industrial Ergonomics. 2018. No. 66. Pp. 26–56.

16. Upwork Help Center. Log Time with the Desktop App, 2018 // URL: https://support.upwork.com/hc/en-us/sections/202260758-Desktop-Apps (Дата обращения: 10.11.2018).

17. **Ovhal K.B., et al.** Analysis of anomaly detection techniques in video surveillance // Internat. Conf. on Intelligent Sustainable Systems. IEEE, 2017. Pp. 596–601.

18. **Abdallah A., Maarof M.A., Zainal A.** Fraud detection system: A survey // J. of Network and Computer Applications. 2016. Vol. 68. Pp. 90–113.

19. **Leijten M., van Waes L.** Keystroke logging in writing research: Using inputlog to analyze and visualize writing processes // Written Communication. 2013. No. 30(3). Pp. 358–392.

20. **de Smet M.J.R., Leijten M., van Waes L.** Exploring the process of reading during writing us-

ing eye tracking and keystroke logging // Written Communication. 2018. No. 35(4). Pp. 411–447.

21. **Timofeev D., Samochadin A.** An unified representation of source code authoring workflows // Proc. of the 10th Internat. Joint Conf. on Knowledge Discovery, Knowledge Engineering and Knowledge Management. KMIS, 2018. Vol. 3. Pp. 228–232.

Статья поступила в редакцию 13.11.2018.

REFERENCES

1. **Schwaber K., Sutherlang J.** The Scrum Guide. 2017. Available: *https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf* (Accessed: 08.11.2018).

2. **Altshuller G.S.** *Algorithm of invention*. Moscow: Moskovskiy rabochiy Publ., 1969, 272 p. (rus)

3. **Jones J.C.** *Design Methods*. Moscow: Mir Publ., 1986, 326 p. (rus)

4. **Poya D.** *Mathematical discovery*. Moscow: Nauka Publ., 1976, 448 p. (rus)

5. **Skiyena S.** *The Algorithms Design Manual*. St. Petersburg: BHV-Petersburg Publ., 2011. (rus)

6. **Carter A., Sangler C.** The Programmers Stone, 1997. Available: *https://www.datapacrat.com/Opinion/Re ciprocality/r0/index.html* (Accessed: 10.11.2018).

7. **Csikszentmihaly M.** *Flow: The Psychology of Optimal Experience*. Moscow: Smysl; Alpina nonfiction Publ., 2017. (rus)

8. **Berta R., Bellotti F., De Gloria A., Pranantha D., Schatten C.** Electroencephalogram and physiological signal analysis for assessing flow in games. *IEEE Transactions on Computational Intelligence and AI in Games 5*, 2013, No. 2, Pp. 164–175.

9. **Kolakowska A., Landowska A., Szwoch M., Szwoch W., Wrobel M.R.** Emotion recognition and its application in software engineering. *2013 6th International Conference on Human System Interactions*, Pp. 532–539.

10. **Fritz T., Begel A., Мьller S.C., Yigit-Elliott S., Züger M.** Using psycho-physiological measures to assess task difficulty in software development. *Proceedings of the 36th International Conference on Software Engineering*, 2014, Pp. 402–413.

11. **Sweller J.** Cognitive load during problem solving: Effects on learning. *Cognit. Sci.,* 1988, No. 12, Pp. 257–285.

12. **Chilton M.A., Hardgrave B.C., Armstrong D.J.** Person-Job cognitive style fit for software developers: The effect on strain and performance. *Journal of Management Information Systems*, 2014, No. 22:2, Pp.193–226.

13. **Blatter K., Cajochen C.** Circadian rhythms in cognitive performance: Methodological constraints, protocols, theoretical underpinnings. *Physiology and Behaviour*, 2006, Pp. 196–208.

14. **Sawyer K.** The cognitive neuroscience of creativity: A critical review. *Creativity Research Journal*, 2011, No. 23:2, Pp. 137–154.

15. **Mshali H., Lemlouma T., Moloney M., Magoni D.** A survey on health monitoring systems for health smart homes. *International Journal of Industrial Ergonomics*, 2018, No. 66, Pp. 26–56.

16. Upwork Help Center. Log Time with the Desktop App, 2018. Available: *https://support.upwork.com/hc/en-us/sections/202260758-Desktop-Apps* (Accessed: 10.11.2018).

17. **Ovhal K.B., et al.** Analysis of anomaly detection techniques in video surveillance. *2017 International Conference on Intelligent Sustainable Systems.* IEEE, 2017, Pp. 596–601.

18. **Abdallah A., Maarof M.A., Zainal A.** Fraud detection system: A survey. *Journal of Network and Computer Applications*, 2016, Vol. 68, Pp. 90–113.

19. **Leijten M., van Waes L.** Keystroke logging in writing research: Using inputlog to analyze and visualize writing processes. *Written Communication*, 2013, No. 30(3), Pp. 358–392.

20. **de Smet M.J.R., Leijten M., van Waes L.** Exploring the process of reading during writing using eye tracking and keystroke logging. *Written Communication*, 2018, No. 35(4), Pp. 411–447.

21. **Timofeev D., Samochadin A.** An unified representation of source code authoring workflows. *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018).* KMIS, 2018, Vol. 3, Pp. 228–232.

Received 13.11.2018.

СВЕДЕНИЯ ОБ АВТОРАХ/ THE AUTHORS

ТИМОФЕЕВ Дмитрий Андреевич
TIMOFEEV Dmitrii A.
E-mail: dtim@dcn.icc.spbstu.ru

МАСЛОВ Максим Юрьевич
MASLOV Maxim Yu.
E-mail: maslov@soft-consult.ru