УДК 004.415.2
DOI: 10.15827/0236-235X.126.251-257
Дата подачи статьи: 11.12.18
2019. Т. 32. № 2. С. 251–257

Архитектура системы мониторинга производственных процессов в условиях географической распределенности производства

Г.М. Соломаха [1], *д.ф.-м.н., профессор кафедры математической статистики и системного анализа, gsolomakha@yandex.ru*
С.В. Хижняк [1], *аспирант, stanislav.khizhnyak@gmail.com*

[1] *Тверской государственный университет, г. Тверь, 170100, Россия*

В работе представлена архитектура информационной системы мониторинга производственных процессов, предоставляющая возможность получения актуальной детальной информации о географически распределенном производстве, а также наблюдения за показателями, являющимися агрегациями других показателей. Система позволяет работать в распределенном режиме, что значительно упрощает внедрение и эксплуатацию в условиях географической распределенности производства. Все компоненты, подсистемы, а также протокол и порядок их взаимодействия ориентированы на использование как на географически распределенных, так и на других производствах.

В статье приведены основные недостатки существующих решений относительно работы в условиях географической распределенности производства, а также требования к архитектуре системы, обладающей необходимыми для работы в таких условиях качествами, сформированными на основе выявленных недостатков. Рассмотрены основные подсистемы и компоненты предлагаемой информационной системы, их назначение, функции и принципы работы. Дано описание протокола взаимодействия между подсистемами и компонентами, обоснован подход к разработке такого протокола. Описаны порядок обработки данных, способ их хранения, а также формат и сигнатура. Данные представлены в формате JSON, а в качестве модели обмена между компонентами выбрана событийная.

Обоснован подход к проектированию архитектуры, приведены основные технологии и средства, необходимые для разработки системы, с обоснованием их выбора, а также схемы архитектуры в разных комбинациях распределенных компонентов. Рассмотрен ряд примеров функционирования отдельных компонентов и их взаимодействия.

На основе проведенных исследований сделаны выводы и предположения о возможных перспективах развития рассматриваемого в статье направления исследований.

Ключевые слова*: информационная система, мониторинг производства, архитектура информационной системы, системный анализ, географическая распределенность*.

Своевременное реагирование на изменения показателей источников данных на производстве, а также характеристик производимых изделий является краеугольным камнем поддержания уровня качества продукции производственного предприятия [1, 2]. Принятие решений, соответствующих ситуации, затрудняется в случае отсутствия или неполноты информации или возможности доступа к ней, что обусловливает интерес к системам, позволяющим собирать, обрабатывать и своевременно предоставлять информацию наиболее удобным способом. Ряд существующих на данный момент систем мониторинга производственных процессов поддерживают работу через веб-терминалы и лишь единицы – с помощью мобильных устройств, что говорит о недостаточной мобильности существующих решений [3, 4].

Производственные инфраструктуры многообразны по своей топологии и сложности. Их многообразие и недостаточная гибкость имеющихся решений усложняют выбор в существующих условиях, а сложность, в свою очередь, усложняет задачу внедрения информационных систем, призванных обеспечить доступ к информации о состоянии производства и его возможных прогнозах. Таким образом, на первый план выходят гибкость и масштабируемость системы, а также наличие возможностей ее расширения в соответствии с изменением ситуации и потенциальными путями расширения производства.

Новые производственные линии не всегда создаются в том же географическом месте, что и старые, это выдвигает перед системой требование о поддержке подобного сценария и обеспечении возможности с минимальными затратами внедрить производство на новом месте.

Стоимость подобной процедуры при использовании существующих решений можно оценить как высокую.

Во время эксплуатации производственного оборудования возможны разного рода сбои и ошибки, в том числе обусловленные человеческим фактором. Накопление сопутствующей таким сбоям информации и дальнейшее ее использование для предотвращения повторных происшествий дает большое преимущество с точки зрения как безопасности на производстве, так и поддержания уровня качества производимой продукции. Соответственно, система мониторинга должна предоставлять возможность хранения проходящих через нее данных для дальнейшего анализа квалифицированными специалистами и достаточно быстрого расширения без негативных последствий набора наблюдаемых показателей новыми, полученными с помощью такого анализа. Наличие подобных возможностей позволяет не только рационально использовать внутренний опыт предприятия, но и отслеживать похожие ситуации на аналогичных производственных предприятиях, что способствует развитию отрасли предприятия в целом. На момент написания статьи такая возможность в имеющихся информационных системах отсутствовала.

Доступность детальной информации о производстве играет важнейшую роль при принятии своевременных решений, что приводит к необходимости обеспечения доступа к ней с помощью мобильных и веб-терминалов. Система должна предоставить удобный протокол взаимодействия с ней для обеспечения возможности реализации программных терминалов, ориентированных на нужды конкретного производства, чего не обеспечивают существующие решения. Это необходимо учитывать при проектировании архитектуры системы, а также протокола взаимодействия ее компонентов.

Цель данной работы – создание архитектуры расширяемой и масштабируемой информационной системы мониторинга производственных процессов, способной хранить информацию, проходящую через нее, обладающей возможностью расширения набора наблюдаемых показателей и обеспечивающей доступ к информации с мобильных и веб-терминалов.

Требования к системе

Перед проектированием архитектуры системы необходимо обозначить основные требования к ней.

Со временем производство может расширяться, а значит, система должна предусматривать такую возможность и быть ***масштабируемой***.

Система должна обеспечивать поддержку быстрого добавления новых источников информации, то есть должна быть ***расширяемой***.

Так как расширение производства не всегда предполагает открытие новых линий в том же географическом месте, где находятся старые, система должна поддерживать ***возможность работы в распределенном режиме*** и обеспечивать внедрение на новом месте с минимальными затратами.

Для своевременной доставки информации оператору система должна поддерживать ***возможность работы с помощью мобильных и веб-терминалов***, а также в случае необходимости обеспечивать протокол взаимодействия для реализации новых программных терминалов.

Мировой опыт разработки систем клиент-серверного взаимодействия с мобильными и веб-клиентами показывает, что наиболее эффективно ***использование в протоколах взаимодействия стандартизированных структур данных***, так как это существенно удешевляет процесс разработки, поддержку и расширение системы.

В силу необходимости обеспечения поддержки распределенного режима работы системы и возможности подключения новых источников информации с минимальными затратами ***протокол взаимодействия должен быть единым*** между всеми ее компонентами.

Для обеспечения возможности дальнейшего анализа система должна ***обеспечивать хранение*** проходящих через нее данных в одной или нескольких БД.

Вероятность того, что данных о показателях производственных процессов, снимаемых с датчиков на производстве, будет недостаточно для эффективного контроля этих процессов, довольно высока. Это обусловливает необходимость поддержки ***возможности добавления дополнительных вычислительных программных компонентов*** для агрегации и комплексной обработки данных без нарушения работы остальной части системы. Кроме того, данная функция обеспечит возможность написания программных компонентов на основе анализа исторических данных, собранных системой.

Для анализа исторических данных соответствующими специалистами их необходимо по-

лучить, а значит, система должна предоставить возможность выгрузки данных за запрошенный период в одном из ***стандартизированных форматов***.

В соответствии с предъявленными требованиями разработана архитектура системы, схема которой изображена на рисунке 1.

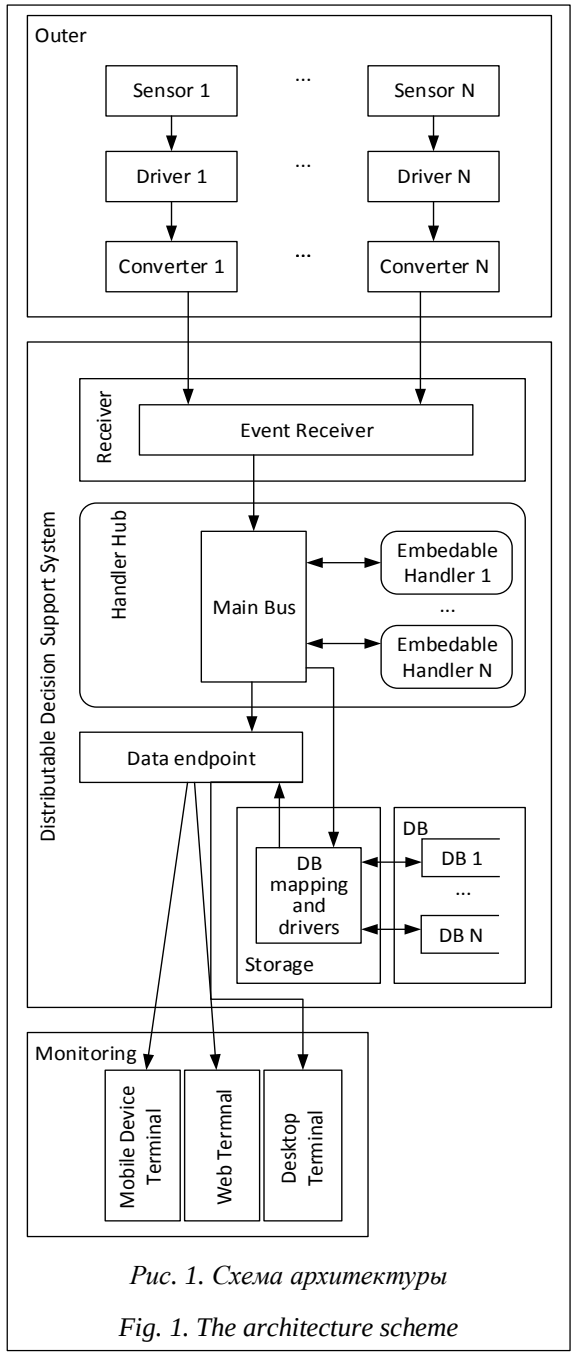

Рис. 1. Схема архитектуры

Fig. 1. The architecture scheme

Компоненты и подсистемы

Система разделена на блоки Outer, DistributableDecisionSupport System, DB, Monitoring.

Каждый из этих блоков может иметь произвольное географическое местоположение. Взаимодействие между блоками происходит путем обмена сообщениями. Возможно использование нескольких экземпляров любого из блоков, дублирующих друг друга или выполняющих разные функции. Например, если необходимо подключить некоторый набор независимых вычислительных программных компонентов, это можно сделать путем добавления дополнительного HandlerHub, который после выполнения нужных действий может быть отключен без каких-либо последствий для остальной части системы. В таком случае схема архитектуры приобретет вид, изображенный на рисунке 2.

Форматом сообщений был выбран JSON как наиболее подходящий с точки зрения дизайна и гибкости. К тому же JSON легко чи-

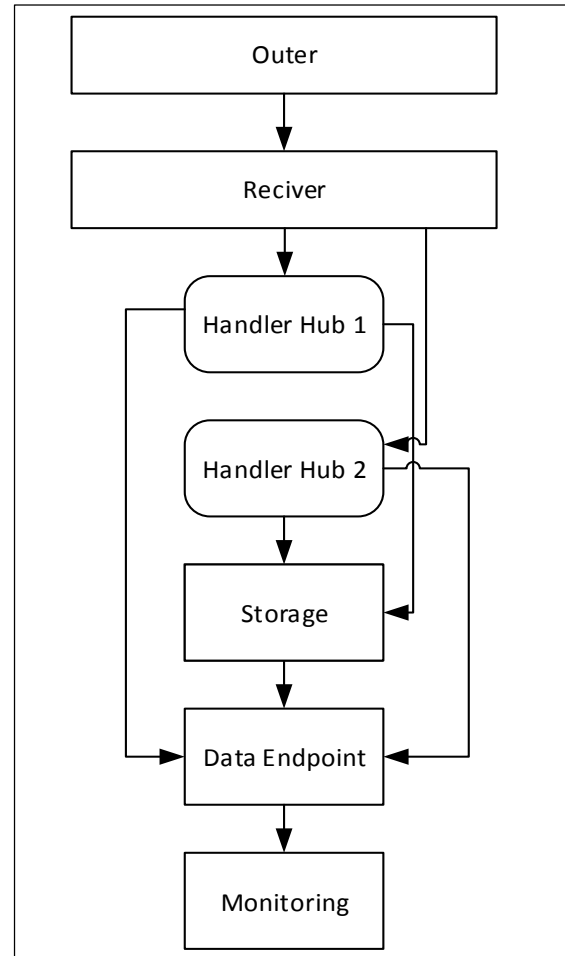

Рис. 2. Схема архитектуры при наличии дополнительного блока HandlerHub

Fig. 2. The architecture scheme with additional Handler Hub block

таем и большинство языков программирования обеспечивают легкую работу с ним [5].

В качестве транспорта выбрано WebSocket-соединение, так как оно построено над являющимся классическим решением протоколом TCP, но вдобавок имеет ряд преимуществ перед другими способами клиент-серверного взаимодействия в случаях, когда клиентские приложения работают на веб- или мобильных платформах [6, 7].

Так как главным назначением системы является предоставление детальной информации о производственных процессах и происходящих в них изменениях, наиболее подходящей является событийная модель построения взаимодействия между компонентами и блоками системы [8].

Таким образом, подобная комбинация обеспечивает возможность реализации мобильных и веб-терминалов, использование стандартизированных структур данных в протоколе взаимодействия, единый протокол взаимодействия между компонентами.

В блок Outer входят непосредственно датчики на производстве, их драйверы и программы для преобразования данных с датчиков в формат, принимаемый системой. Данный блок не является полноценной частью системы в прямом смысле в силу огромного количества возможных источников информации и различий в протоколах непосредственного взаимодействия с ними. Реализация входящих в этот блок компонентов, их количество и набор для разных производственных предприятий могут отличаться. Предполагается ее осуществление специалистами предприятия или использование готовых реализаций, выполненных специалистами предприятий с аналогичными источниками.

В блок DistributableDecisionSupportSystem входят компоненты Event Receiver, Handler Hub, Storage, они являются ядром системы.

Модуль EventReceiver – точка входа производственных данных в систему мониторинга, откуда они направляются в один или несколько экземпляров HandlerHub. Данные представляются в формате JSON в виде объекта и должны содержать имя показателя и объект с информацией, характеризующей текущее состояние показателя, что в совокупности и является одним событием системы. Например, для датчика температуры JSON объект, входящий в систему, может иметь следующий вид:

```
{
  "name":"TemperatureSensor_1",
  "data":{
    "value":46,
    "date":1553108839
  }
}
```

В данном случае именем показателя является TemperatureSensor_1, а объект с данными содержит текущую температуру, регистрируемую датчиком, и время фиксации в виде временной метки UNIX. Такая структура представляет собой событие, характеризующее штатную работу датчика, и содержит информацию о его текущем состоянии.

HandlerHub состоит из MainBus, являющейся шиной событий системы, и расширяемого набора реализаций EmbedableHandler. Каждый отдельный EmbedableHandler – это обработчик событий системы, получаемых из MainBus с целью их агрегации или другой обработки. Каждый EmbedableHandler описывается в отдельном файле на языке JavaScript, после чего загружается и интерпретируется системой, что обеспечивает возможность добавления дополнительных вычислительных программных компонентов без нарушения работы системы. Выбор JavaScript обусловлен широтой практики его применения в качестве встраиваемого языка и удобства для создания приложений, управляемых событиями [9, 10]. Обработчики регистрируются в MainBus для получения одного или нескольких разных событий. При поступлении в систему события через MainBus доставляются обработчикам, в которых на языке JavaScript производится необходимая агрегация или другая обработка данных события и формируются новые события, имеющие другие имена и содержащие данные, являющиеся результатом обработки.

Рассмотрим пример. Пусть в помещении на производстве установлена воздушная система охлаждения, управляемая автоматически с помощью каких-либо средств, характер и свойства системы управления в данном примере несущественны. Алгоритм управления охлаждением предполагает увеличение количества оборотов двигателей системы при увеличении температуры в помещении. Кроме этого, условия производства требуют, чтобы максимальное количество оборотов двигателя не превышало некоторый порог R. Предполагается, что данные о количестве оборотов поступают в систему регулярно через некоторый промежуток времени в следующем виде:

```
  {
 "name":"RPM_1",
 "data":{
   "value":533,
   "date":1553110407
 }
}
```

Здесь именем показателя является *RPM_1*, а текущее число оборотов в минуту равно 533. Тогда для контроля за превышением порога необходим обработчик события *RPM_1*, в котором происходят сравнение текущего значение с порогом и генерация нового события *RMP_1Exceed* в случае превышения текущим значением порога *R*. В поле *data* объект события может содержать, например, величину, на которую был превышен порог. В простейшем виде функция обработки на языке JavaScript, решающая такую задачу, будет следующей:

```
function handle(event, R) {
   if (event.data.value> R) {
      return {
         "name": "RMP_1Exceed",
         "data": {
            "value": event.data.value - R,
"date": new Date().getTime()
         }
      }
   } else return null
}
```

Таким образом, после генерации в обработчике событие превышения попадет в MainBus и будет доставлено обработчикам, зарегистрированным на него (если таковые имеются), после чего попадет в компонент DataEndpoint**.**

Блок Storage содержит драйверы и программные компоненты для преобразования данных из формата системы в формат используемых реализаций БД, а блок DB представляет собой внешние БД, что обеспечивает хранение данных. В связи с большими предполагаемыми объемами данных и малым количеством связей в качестве СУБД предполагается использование MongoDB, так как она является наиболее подходящей в подобных условиях [11, 12]. В силу того, что MongoDB – это документоориентированная СУБД, а сигнатура событий и их набор могут отличаться от предприятия к предприятию и даже от подразделения к подразделению, привести схему структуры БД невозможно, однако достаточно пояснить, что структура представляет собой коллекцию документов, каждый из которых хранит данные об одном событии системы и имеет аналогичную этому событию структуру.

Компонент DataEndpoint выполняет функции точки входа для взаимодействия с системой. Сюда поступают события из MainBus после их обработки всеми экземплярами EmbedableHandler. Получение информации о текущем состоянии системы и истории событий осуществляется с помощью этого компонента. К DataEndpoint подключаются терминалы, через которые происходит непосредственное наблюдение показателей оператором системы, а также с его помощью осуществляется выгрузка истории событий. DataEndpoint предоставляет возможность подключения по протоколу WebSocket и взаимодействия с системой с помощью сообщений. Сообщения имеют сигнатуру, аналогичную событиям системы, с целью унификации протокола. Например, инициация регулярного (через настраиваемый промежуток времени) оповещения клиента о произошедших за этот промежуток событий осуществляется с помощью сообщения со следующей сигнатурой:

```
  {
 "name":"SubscribeState",
 "data":{
   "events":[...],
 }
},
```

где *events* – опциональное поле, содержащее массив с именами событий, регулярное получение которых необходимо клиенту.

Блок Monitoring включает программные терминалы системы мониторинга. Использование WebSocket в качестве транспорта открывает широкие возможности для реализации программных терминалов в силу наличия множества готовых решений под большинство существующих платформ, включая мобильные и встраиваемые, что повышает мобильность системы в целом.

Событийная модель данных и взаимодействие с помощью WebSocket в формате JSON дают возможность дублировать или добавлять новые реализации блоков компонентов, что, в свою очередь, позволяет каждому блоку иметь произвольное географическое местоположение. Кроме того, это существенно упрощает процесс внедрения системы, так как можно осуществлять его по блокам без необходимости остановки производства на весь период внедрения. В совокупности эти возможности обеспечивают масштабируемость, расширяемость и возможность работы в распределенном режиме.

Заключение

В ходе исследования проанализирован рынок существующих систем мониторинга производственных процессов, выявлены их недостатки, с целью устранения которых сформированы требования к проектируемой системе. В соответствии с полученными требованиями разработана архитектура расширяемой и масштабируемой информационной системы мониторинга производственных процессов, способной хранить информацию, проходящую через нее, обладающей возможностью расширения набора наблюдаемых показателей и обеспечивающей доступ к информации с мобильных и веб-терминалов.

Система, обладающая данной архитектурой, имеет достаточную для своевременного предоставления детальной информации о производственных процессах мобильность, обеспечивает достаточную масштабируемость для наименее затратного внедрения в случае расширения производства и предоставляет необходимую расширяемость для быстрого добавления новых источников информации. Предусмотренный механизм хранения и получения исторических данных позволяет использовать накопленную информацию для анализа, а механизм добавления вычислительных компонентов на языке JavaScript обеспечивает возможность использования как результатов анализа данных предприятия, на котором система установлена непосредственно, так и внешнего опыта на аналогичных предприятиях. Использование формата JSON для представления данных и WebSocket в качестве транспорта, а также событийной модели данных делает возможной и удобной реализацию любого вида программных терминалов для любых существующих платформ, а событийная модель обмена данными открывает возможности для перехода от мониторинга к управлению.

Литература

1. Нечаев В.И., Нечаева Е.С. Система мониторинга экономического состояния производственных процессов // Механизмы управления экономическими системами: методы, модели, технологии: сб. докл. Международ. науч.-практич. конф. 2017. С. 188–192.

2. Казенова Н.С., Новикова Г.М. Подход к проектированию системы мониторинга производственных процессов // Информационно-телекоммуникационные технологии и математическое моделирование высокотехнологичных систем: сб. докл. 2015. С. 124–130.

3. Астафьев А.В., Провоторов А.В., Орлов А.А. Анализ визуальных систем мониторинга производственного процесса на промышленных предприятиях // Вестн. НГУЭУ. 2011. № 1. С. 26–32.

4. Астафьев А.В., Провоторов А.В., Орлов А.А. Комплексный анализ систем мониторинга и визуализации производственного процесса на промышленных предприятиях // Системный анализ в науке и образовании. 2011. № 1. С. 1–6.

5. Сергушкин В.В., Бодров О.А. Использование формата данных JSON для взаимодействий с пользователем при разработке программного обеспечения аппаратуры передачи данных // Современные технологии в науке и образовании. СТНО-2018: сб. тр. 2018. Т. 5. С. 217–220.

6. Хабаров С.П. Взаимодействия узлов сети по протоколу WebSocket // Информационные системы и технологии: теория и практика: сб. тр. СПб: Изд-во ГЛТУ, 2017. № 9. С. 109–119.

7. Pimentel V., Nickerson B.G. Communicating and displaying real-time data with WebSocket. IEEE Internet Computing, 2012, vol. 16, no. 6, pp. 45–53. DOI 10.1109/MIC.2012.64.

8. Достлев Ю.С., Чередникова О.Ю. Событийная модель технологического объекта в системах реального времени // Системный анализ и информационные технологии в науках о природе и обществе. 2015. № 1–2. С. 68–72.

9. Берко В.А., Таран В.Н. Применение языка программирования JavaScript в разработке систем интернета вещей // Дистанционные образовательные технологии: матер. III Всерос. науч.-практич. конф. 2018. С. 198–201.

10. Kossakowski G., Amin N., Rompf T., Odersky M. JavaScript as an embedded DSL. LNCS, Springer, Heidelberg, 2012, vol. 7313, pp. 409–434.

11. Федоров А.Р., Васильчук К.С., Дорофеев А.В. Создание масштабируемых средств для решения задач анализа больших объемов данных на основе системы управления базы данных MongoDB // Вестн. Поволжского гос. ун-та: Радиотехнич. и инфокоммуникацион. сист. 2016. № 29. С. 55–63.

12. Голиков О.И., Панкратов И.А. Исследование способов повышения эффективности обработки данных в реляционных БД на примере СУБД MySQL // Вестн. Волжского ун-та им. В.Н. Татищева. 2016. Т. 2. № 2. С. 124–130.

Software & Systems Received 11.12.18
DOI: 10.15827/0236-235X.126.251-257 2019, vol. 32, no. 2, pp. 251–257

The architecture of a production processes monitoring system in terms of geographically distributed production

G.M. Solomakha [1], *Dr.Sc. (Physics and Mathematics), Professor, gsolomakha@yandex.ru*
S.V. Khizhnyak [1], *Postgraduate Student, stanislav.khizhnyak@gmail.com*

[1] *Tver State University, Tver, 170100, Russian Federation*

Abstract. The paper describes an architecture of a production process monitoring system, which provides an opportunity to receive relevant and detailed information on geographically distributed production, as well as to observe indicators that are aggregations of other indicators. The system enables working in a distributed mode, which simplifies the implementation and operation in terms of geographical distribution of production. All components, subsystems, as well as the protocol and their coordination arrangements are focused both on using geographically distributed, and other productions.

The paper presents the main drawbacks of the existing solutions regarding work under conditions of geographical distribution of production. It also outlines the requirements for a system architecture, which has the qualities that are necessary for working in such conditions, formed based on the identified drawbacks. The paper describes the main subsystems and components of the proposed system, their purpose, functions and operation principles. There is a description of the interaction protocol between subsystems and components. The approach to the development of such protocol is justified. There is a description of a data processing order, a data storage method, as well as its format and signature. The data is presented in JSON format. The event model is selected as the exchange model between components.

The paper justifies the approach to architecture design, presents the main technologies and tools for system development, and justifies this choice. There are architecture schemes in various combinations of distributed components. Several examples of the functioning of individual components and their interaction are considered.

Based on the conducted research, the authors have made conclusions and proposed possible prospects for the development of the covered topic.

Keywords: information system, production monitoring, information system architecture, system analysis, geographical distribution.

References

1. Nechaev V.I., Nechaeva E.S. Monitoring system of the economic state of production processes. *Mechanisms of Economic System Management: Methods, Models, Technologies: Proc. Intern. Sci. and Pract. Conf*. 2017, pp. 188–192 (in Russ.).
2. Kazenova N.S., Novikova G.M. An approach to designing monitoring system of production processes. *Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems, Proc*. (ITTMM 2015). 2015, pp. 124–130 (in Russ.).
3. Astafev A.V., Provotorov A.V., Orlov A.A. The analysis of visual systems of production process monitoring in enterprises. *Vestn. NSUEM*. 2011, no. 1, pp. 26–32 (in Russ.).
4. Astafev A.V., Provotorov A.V., Orlov A.A. Complex analysis of monitoring and visualization systems in industry enterprises. *System Analysis in Science and Education*. 2011, no. 1, pp. 1–6 (in Russ.).
5. Sergushkin V.V., Bodrov O.A. Using the JSON data format to interact with user when developing data transmission equipment software. *Modern Technologies in Science and Education* (STNO-2018). 2018, vol. 5, pp. 217–220 (in Russ.).
6. Khabarov S.P. Interaction of network nodes via WebSocket protocol. *Proc. Information Systems and Technologies: Theory and Practice*. St. Petersburg, GLTU Publ., 2017, no. 9, pp. 109–119 (in Russ.).
7. Pimentel V., Nickerson B.G. Communicating and displaying real-time data with WebSocket. *IEEE Internet Computing*. 2012, vol. 16, pp. 45–53. DOI 10.1109/MIC.2012.64.
8. Dostlev Yu.S., Cherednikova O.Yu. Event model processing facility in real-time systems. *System Analysis and Information Technologies in Environmental and Social Sciences*. 2015, no. 1–2, pp. 68–72 (in Russ.).
9. Berko V.A., Taran V.N. JavaScript application in the development of Internet of things systems. *Remote Educational Technologies: Proc. 3rd All-Russ. Sci. Pract. Conf*. 2018, pp. 198–201 (in Russ.).
10. Kossakowski G., Amin N., Rompf T., Odersky M. JavaScript as an embedded DSL. *LNCS*. 2012, vol. 7313, pp. 409–434.
11. Fedorov A.R., Vasilchuk K.S., Dorofeev A.V. Creation of scalable tools to solve the problems of the analysis of large volumes of data based on DMBS MongoDB. *Vestn. of Volga State Univ. of Technology. Series Radio Engineering and Infocommunication Systems*. 2016, pp 55–63 (in Russ.).
12. Golikov O.I., Pankratov I.A. A study of ways to improve the efficiency of data processing in relational databases on MySQL database example. *The Bulletin of Volzhsky Univ. after V.N. Tatishchev*. 2016, vol. 2, no. 2, pp. 124–130 (in Russ.).