УДК 658.5:629.78
DOI: 10.15827/0236-235X.134.237-244
Дата подачи статьи: 18.02.21
2021. Т. 34. № 2. С. 237–244

Разработка ядра интегрированной информационной системы

Б.А. Черныш [1], *аспирант, borisblack@mail.ru*
А.С. Картамышев [2], *начальник группы отдела автоматизации финансово-экономических задач, kartam@iss-reshetnev.ru*

[1] *Сибирский государственный университет науки и технологий им. академика М.Ф. Решетнева, г. Красноярск, 660037, Россия*
[2] *«Информационные спутниковые системы» им. академика М.Ф. Решетнева», г. Железногорск, Красноярский край, 662972, Россия*

В статье рассматривается проблема, часто встречающаяся при проектировании корпоративных информационных систем, когда приходится одновременно решать две противоречивые задачи: эффективное применение в одной конкретной предметной области и гибкая адаптация к возможным требованиям другой области. Для решения этих задач используется подход, заключающийся в формировании программной основы или платформы, реализующей обобщенный функционал с возможностью гибкой конфигурации в соответствии с изменяющимися требованиями. Платформа должна соответствовать таким требованиям современных реалий, как гибкая сервис-ориентированная архитектура, версионность, аудит и ведение истории изменений, возможность хранения двоичных данных, гибкое управление жизненным циклом и бизнес-процессами.

В статье излагаются принципы построения таких систем, лежащих в основе подходов и технологий, интерфейсов взаимодействия и протоколов обмена данными. Затрагиваются вопросы применения современных предметно-ориентированных языков поверх этих протоколов. Приводятся примеры имеющихся отечественных и зарубежных систем, оцениваются области их применения, достоинства и недостатки.

Описывается реализация универсальной информационной платформы на базе разрабатываемой авторами системы в области автоматизации производства технически сложных изделий. Приведены некоторые применяемые решения и приемы, структуры данных и алгоритмы. Эффективность предложенных методик подтверждается сравнительным измерением производительности разрабатываемой системы и одной из коммерческих платформ.

На основании полученной оценки сделаны выводы о перспективности разработки, ее применимости в различных предметных областях. Выработана дорожная карта по дальнейшему развитию и оптимизации платформы в наиболее значимых направлениях с учетом актуальных требований отрасли.

Ключевые слова: *корпоративная информационная система, программная платформа, ядро, интерфейс, API, система управления, жизненный цикл.*

Построение качественных и высокоэффективных корпоративных автоматизированных систем требует от разработчиков больших усилий, связанных с удовлетворением всех возможных (настоящих и будущих) требований предметной области. Чем полнее программный продукт удовлетворяет этим требованиям, тем эффективнее его использование в конкретной предметной области, и, чем специфичнее область применения программы, тем сложнее ее адаптация для других областей.

Для решения этих двух противоречивых задач (удовлетворение требованиям одной области и гибкая адаптация к требованиям другой) необходимо использовать промежуточный слой ПО (можно назвать его ядром или общей платформой), содержащий функционал, применимый в любой предметной области.

В настоящее время рынок универсальных платформ представлен очень широко [1]. К ним можно отнести и платформы широкого применения (отечественные – 1С:Предприятие, Парус, Галактика, импортные – SAP S/4HANA, Oracle E-Business Suite, Microsoft Dynamics), и специализированные (например, Aras Innovator, изначально спроектированная как PLM-система) [2]. При изучении существующего рынка необходимо, помимо функционала систем, учитывать и такие факторы, как стоимость поставки, внедрения и обслуживания, применимость в предметной области, эффективность в решении конкретных задач, показа-

тели работы и даже политические аспекты (применительно к импортным решениям) [3]. В этой связи альтернативу так называемым коробочным решениям могут составлять собственные продукты.

Авторами предлагается способ программной реализации такой платформы на примере разрабатываемой *системы управления документами* (СУД) [4], предназначенной как для конкретного использования по прямому назначению, так и для интеграции через ее функционал различных предметных областей с описанием теоретических и практических аспектов проектирования. Назначение системы – создание и сопровождение единого дерева документации по изделию на всем протяжении *жизненного цикла* (ЖЦ) продукции. В числе основных требований к платформе СУД – возможность гибкой конфигурации сущностей (атрибутов, представлений, обработчиков), средства управления ЖЦ и контроля доступа. В таком виде ядро СУД может рассматриваться как универсальная информационная платформа и использоваться в любой предметной области (PLM – управление ЖЦ продукции, ERP – планирование ресурсов предприятия, CRM – управление взаимоотношениями с клиентами), предусматривающей хранение специфичных для нее объектов (сущностей) и управление их ЖЦ.

Основные требования к платформе

Полное дерево документации лишь по одному изделию достигает десятков и сотен тысяч документов. При этом вложенность документов не ограничена. Основные требования к платформе следующие:

– гибкая сервис-ориентированная (SOA – Service Oriented Architecture) [5] архитектура, обеспечивающая интеграцию с другими информационными системами;

– масштабируемая клиент-серверная архитектура;

– контроль доступа, реализованный с использованием учетных записей пользователей и ролей;

– гибкая настройка прав доступа на уровне как учетных записей, так и объектов и состояний ЖЦ;

– гибкая настройка ведения истории изменений объектов;

– простое и эффективное управление ЖЦ объектов;

– поддержка бизнес-процессов (Workflow – рабочий поток): назначение ответственных, делегирование задач, выполнение этапов в автоматическом режиме, оповещение участников, контроль за принятием решений;

– управление документооборотом;

– интеграция файлового хранилища.

При таких условиях наиболее важными факторами являются правильный выбор механизма хранения и манипуляции данными и метаданными платформы, а также предоставление гибкого и удобного интерфейса взаимодействия (API – Application Programming Interface) с ней.

Подходы к построению API

На сегодняшний день наиболее эффективными и общепринятыми способами реализации клиент-серверной SOA-архитектуры посредством протокола HTTP являются архитектурный стиль REST и протокол SOAP. Данные в REST могут передаваться в виде HTML, XML или JSON (JavaScript Object Notation). В большинстве реализаций используется формат JSON. Протокол SOAP использует в своей основе язык XML. Главными недостатками формата XML в сравнении с JSON являются больший размер передаваемых данных, а также более сложный и длительный их анализ (парсинг), особенно заметный на больших объемах информации. Стандартная практика использования REST и SOAP предусматривает создание на сервере отдельных обработчиков для каждого типа обрабатываемых сущностей. Значительной проблемой такого подхода является то, что информация о структуре данных требуется на этапе создания этих обработчиков. В связи с тем, что заранее неизвестно, с какими сущностями будет работать платформа, как неизвестны также атрибуты и взаимосвязи этих сущностей, необходим универсальный API в виде единой точки доступа, который позволил бы выполнять необходимые операции с объектами любого типа. Данный API может быть реализован и посредством REST или SOAP.

Примером такого решения является открытый API платформы Aras Innovator [6], работающий по протоколу SOAP. Для запросов используется предметно-ориентированный язык DSL (Domain Specific Language), именуемый AML (Adaptive Markup Language) [7] и являющийся подмножеством XML. Этот язык – мощный инструмент в построении запросов к API, позволяющий выполнять как предопределенные платформой, так и пользовательские операции. Поддерживает вложенность объектов,

а также позволяет гибко конфигурировать параметры запросов.

Пример запроса на языке AML:

```
<Item type="Part" action="add">
<item_number>999-888</item_number>
<description>Some Assy</description>
<Relationships>
     <Item type="Part BOM" action="add">
          <quantity>10</quantity>
          <related_id>
             <Item type="Part" action="add">
                  <item_number>123-
                     456</item_number>
                  <description>1/4w 10% 10K
                     Resistor</description>
             </Item>
          </related_id>
     </Item>
</Relationships>
</Item>
```

Несмотря на перечисленные возможности, платформа имеет ряд недостатков. Во-первых, Aras Innovator является коммерческой платформой (хотя базовый функционал предоставляется бесплатно по запросу). Во-вторых, программа работает только в *операционных системах* (ОС) семейства Microsoft Windows с использованием СУБД Microsoft SQL Server. Таким образом, платформа непереносима ни в рамках ОС, ни в рамках СУБД. В-третьих, Aras Innovator требователен к ресурсам, тем не менее, задержки при выполнении запросов весьма существенны. К тому же использование Aras Innovator (и других подобных решений от иностранных производителей) может быть ограничено на ряде отечественных предприятий в связи с вводом в действие государственных программ по импортозамещению.

Другим возможным способом реализации универсального API может быть использование GraphQL [8] – относительно нового и более современного инструмента, представляющего собой DSL и среду выполнения для API. Недостаток такого подхода заключается в следующем: одним из основных требований к платформе СУД является возможность динамического создания и конфигурирования сущностей (объектов), а в связи с тем, что архитектура GraphQL предусматривает хранение в памяти полной схемы типов объектов и операций над ними, создание или изменение одного или нескольких объектов потребует перегенерации этой схемы. Такая перегенерация вносит существенные накладные расходы и может вызвать появление непредсказуемых ошибок в обработке запросов к предыдущей версии схемы.

Проектирование и разработка ядра СУД

Принимая во внимание изначально заложенные требования к платформе и учитывая достоинства и недостатки рассмотренных выше решений, при проектировании ядра СУД авторами выбран и опробован смешанный подход, заключающийся в комбинировании наиболее удачных из них с собственными разработками. Для обеспечения переносимости в качестве среды выполнения выбрана платформа Java, поддерживаемая большинством современных ОС. Ядро СУД построено на свободно распространяемой платформе Spring. С целью уменьшения размера передаваемых данных и ускорения их обработки в качестве формата передачи используется JSON. Структура запросов/ответов представляет собой адаптированную для этого формата модифицированную версию AML.

Приведем пример запроса:

```
{
   "_meta": {"type": "Part", "action":
"add"},
   "label": "AWS0001",
   "name": "Aluminium Wheel Assembly",
   "_relationships": [{
      "_meta": {"type": "PartBOM", "action":
"add"},
      "target_id": {
         "_meta": {"type": "Part", "ac-
tion": "add"},
         "label": "SP-0001",
         "name": "Chrome Spoke"
      }
   }]
}
```

Поддерживаются обработка вложенных объектов и иерархические рекурсивные запросы. С целью оптимизации рекурсивные запросы реализуются непосредственно средствами СУБД (механизм Recursive Query с использованием инструкций With и With Recursive) [4]. Такая оптимизация позволяет на порядок (по сравнению с Aras Innovator) уменьшить время выполнения данного типа запросов. Для обслуживания вложенных запросов (нерекурсивных) в ядре СУД используется механизм обработчиков DataFetcher, аналогичный решению, реализованному в библиотеке GraphQL Java [9]. Представим базовые операции с сущностями:

add – создает сущность;

update – обновляет данные сущности; сущность должна быть заблокирована (то есть выполнена операция *lock*); если сущность версио-

нируемая, создается ее новая версия, иначе – обновляется существующая;

purge – удаляет версию сущности;

delete – удаляет все версии сущности; если сущность неверсионируемая, то операции *purge* и *delete* выполнят одинаковые действия;

get – возвращает запрошенную сущность;

edit – блокирует, обновляет данные и разблокирует сущность;

create – если сущность существует, выполняет операцию *get*, иначе – *add*;

merge – если сущность существует, выполняет операцию *edit*, иначе – *add*;

lock – блокирует сущность от изменения другими пользователями;

unlock – снимает с сущности блокировку от изменения другими пользователями;

version – создает новую версию сущности, очищает атрибут *locked_by* текущей сущности и выставляет его во вновь созданной; затем выполняет операцию *update*; данная операция выполняется только для версионированных сущностей.

Кроме того, платформа позволяет создавать пользовательские операции и привязывать их к сущностям определенного типа.

В таблице приведены сравнительные результаты измерения времени выполнения запросов в Aras Innovator и СУД.

Выполнено по 10 запросов списка сущностей Part (составная часть) без учета иерархии.

Время выполнения запросов в Aras Innovator и СУД (мс)

Query execution time in Aras Innovator and the DMS (ms)

Номер запроса	**Aras Innovator**	**СУД**
	<Item type="Part" action="get"/>	{"_meta": {"type": "Part", "action": "get"}}
1	39	6,5
2	45	7,5
3	42	6,2
4	45	6,7
5	37	5,7
6	53	6,3
7	34	5,7
8	51	6
9	38	6,2
10	41	6,1
$\bar{t}$	**42,50**	**6,290**
S_0	**6,078**	**0,528**

Число тестовых записей в обеих системах – 50, набор атрибутов идентичен. Обе системы развернуты на одном сервере со следующими характеристиками: четырехъядерный процессор Intel Core i5, 8 Гб оперативной памяти, ОС Windows Server 2012 R2. Для оценки стабильности результатов в таблице приводится стандартное отклонение времени на основании несмещенной оценки дисперсии. Рассчитано по формуле

$$S_0 = \sqrt{\frac{1}{n-1}}?(/()\sum_{i=1}^{n}(t_i - \bar{t})^2,$$

где t_i – значение времени i-го запроса (в миллисекундах); $\bar{t}$ – среднее значение по всем запросам; n – общее число запросов.

Из приведенных в таблице данных видно, что среднее время выполнения аналогичного запроса в СУД значительно (практически на порядок) ниже, чем в Aras Innovator. Кроме того, в СУД существенно ниже стандартное отклонение, что говорит о высокой стабильности данной системы. Эффективность ядра СУД подтверждается и опытом практической эксплуатации системы при выполнении иерархических запросов на больших объемах (более 300 тыс. записей) и глубине вложенности семи и более уровней.

Структура хранения данных и метаданных платформы

В качестве хранилища данных в описываемом решении используется реляционная СУБД. В настоящий момент поддерживаются БД Oracle и PostgreSQL. Схема данных разбита на наборы таблиц, отвечающих за различные подсистемы (ядро, контроль доступа, управление ЖЦ, workflow, пользовательский интерфейс, собственно данные). Так, например, описания объектов предметной области, их полей и взаимосвязей содержатся в таблицах CORE_CLASS, CORE_PROPERTY и CORE_RELATIONSHIP соответственно (рис. 1). Все таблицы, помимо основных атрибутов, содержат служебные атрибуты аудита и управления версиями.

Средства безопасности и контроля доступа

Безопасность и контроль доступа реализованы на базе платформы Spring Security [10]. Данные пользователей, групп, ролей и списков контроля доступа (ACL – Access Control List) также хранятся в таблицах СУБД. На рисунке 2

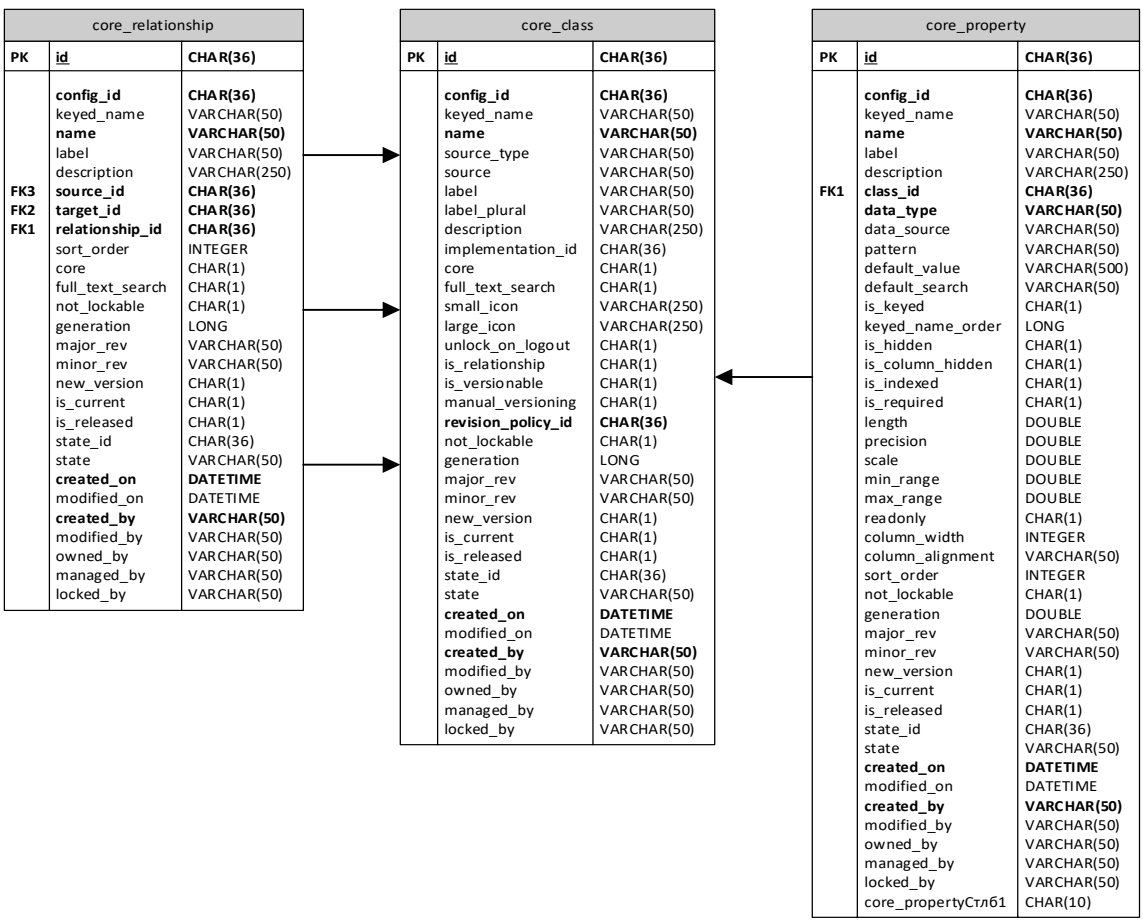

Рис. 1. Основные таблицы метаданных платформы

Fig. 1. Main platform metadata tables

представлены таблицы ACL. При программном построении структуры будущих SQL-запросов вызываются специальные обработчики, ответственные за фильтрацию данных и операций на основании ACL. Для подключения данных обработчиков используется широко применяемый механизм (паттерн) Chain of Responsibility (цепочка обязанностей) [11]. Их задача – дополнить структуру будущего запроса ограничивающими выражениями на основании ACL. Таким образом, контроль доступа осуществляется на уровне SQL-запросов, что исключает избыточность данных, избавляет от необходимости их последующей обработки и существенно уменьшает потоки ввода-вывода.

Перспектива усовершенствования платформы

Основные функции ядра СУД реализованы, однако разработка платформы продолжается. Для гибкой настройки стадий ЖЦ объектов на всем протяжении их существования разрабатываются визуальный редактор ЖЦ, а также полноценная поддержка бизнес-процессов (Workflow): визуальное редактирование, назначение ответственных, делегирование задач, выполнение этапов в автоматическом режиме, оповещение участников, контроль за принятием решений. Данная задача может быть решена непосредственно в ядре с использованием БД платформы либо путем интеграции с одним из существующих BPMS (Business Process Management System) решений в этой области. В качестве примера такого решения можно привести отлично зарекомендовавшую себя во многих сферах свободно распространяемую систему Camunda [12].

Заключение

Наличие универсальной информационной платформы позволяет существенно упростить и ускорить процесс разработки и развертыва-

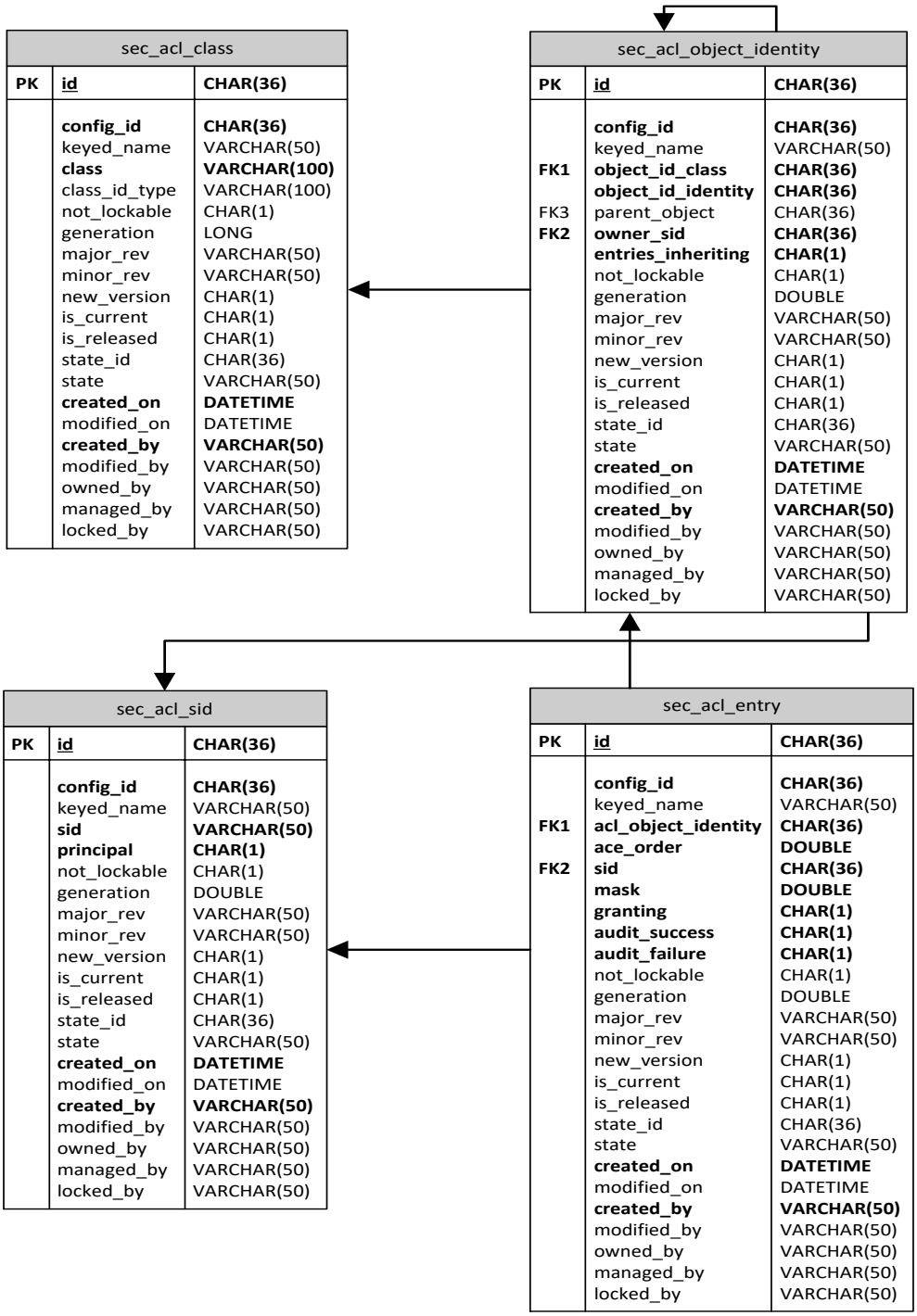

Рис. 2. Таблицы ACL

Fig. 2. ACL tables

ния прикладных решений по сравнению с разработкой этих решений с нуля. Основанием для ее использования на предприятии является наличие сущностей предметной области (документы, составные части изделий, ресурсы, клиенты и т.д.), отраженных в учетных системах и имеющих определенный ЖЦ. На примере описанного решения авторы показывают, что информационные системы, самостоятельно разработанные на предприятии, вполне успешно могут решать широкий круг задач на уровне более именитых аналогов, а по ряду показате-

лей и опережать их. Разумеется, речь не идет о составлении реальной конкуренции компаниям, являющимся лидерами отрасли корпоративных платформ. Главная цель разработки – создание полноценного ядра с базовым функционалом управления ЖЦ сущностей, контролем доступа, эффективным API, гибкой адаптацией к требованиям предметной области. Перечисленные требования успешно реализованы и применяются в платформе СУД.

Литература

1. Чернышев Л.О., Лебедев В.В., Чернышев О.Л. Обзор программного обеспечения для автоматизации деятельности фирмы // Проблемы информатики в образовании, управлении, экономике и технике: сб. стат. XV Междунар. науч.-технич. конф. Пенза, 2015. С. 173–177.

2. Ситкова М.А. ERP-платформы для малых и средних предприятий // Научные записки молодых исследователей. 2014. № 4. С. 55–58.

3. Одинцова М.А. Возможности систем класса ERP для стратегического управления предприятием // Политика, экономика и инновации. 2020. № 4. С. 1–10.

4. Черныш Б.А., Картамышев А.С., Потуремский И.В., Хайдукова В.Н. Организация сквозного сопровождения позаказного производства наукоемкой продукции // Динамика сложных систем – XXI век. 2020. Т. 14. № 4. С. 46–54. DOI: 10.18127/j19997493-202004-05.

5. Терентьев К.А. Применение принципов SOA при построении информационных систем // Проблемы управления, обработки и передачи информации: сб. тр. Междунар. науч. конф. Саратов, 2019. С. 690–692.

6. Aras Innovator. URL: https://www.aras.com/en (дата обращения: 09.01.2021).

7. AML Basics. URL: https://community.aras.com/b/english/posts/aml-basics (дата обращения: 13.01.2021).

8. Кетов Д.К. Разработка API-доступа к данным с динамически изменяемой структурой на основе GraphQL // Автоматизированные системы управления и информационные технологии: матер. Всерос. науч. конф. Пермь, 2019. С. 108–114.

9. GraphQL Java Home. URL: https://www.graphql-java.com (дата обращения: 15.01.2021).

10. Knutson M., Winch R., Mularien P. Spring Security. Birmingham, UK, Packt Publ., 2017, 542 p.

11. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley Longman Publ., 1995, 416 p.

12. Camunda. URL: https://camunda.com (дата обращения: 19.01.2021).

Software & Systems
DOI: 10.15827/0236-235X.134.237-244

Received 18.02.21
2021, vol. 34, no. 2, pp. 237–244

The core design of an integrated information system

B.A. Chernysh [1], *Postgraduate Student, borisblack@mail.ru*
A.S. Kartamyshev [2], *Head of the Group of the Financial and Economic Problems Automation Department, kartam@iss-reshetnev.ru*

[1] *Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, 660037, Russian Federation*
[2] *Academician M.F. Reshetnev Information Satellite Systems, Zheleznogorsk, 662972, Russian Federation*

Abstract. The paper deals with a common problem in the design of corporate information systems, when it is necessary to solve simultaneously two conflicting objectives: the effectiveness of application in one specific subject area and flexible adaptation to the requirements of another area. To solve these problems, there is an approach that consists in forming a software framework or platform that implements generalized functionality with the possibility of flexible configuration under evolutionary requirements. The platform must comply with the requirements of modern realities, such as flexible service-oriented architecture, versioning, auditing and maintaining the history of changes, the ability to store binary data, flexible management of the life-cycle and business processes.

The paper outlines the design concepts of such systems, underlying approaches, and technologies, interfaces of data exchange protocols. The author touches upon the application of modern domain-specific languages on top of these protocols. There are examples of existing domestic and foreign systems, their application areas, advantages, and disadvantages.

The paper describes the implementation of a comprehensive information platform based on a system developed by the authors in the production of technically complex product platforms. There are some applicable solutions and techniques, data structures, and algorithms. A comparative measurement of performing the developed system and one of the commercial platforms confirms the effectiveness of the proposed methods.

The resulting assessment is the basis for conclusions about the prospects of the development, its applicability in various subject areas. The author has developed a roadmap for further development and optimization of the platform in the most significant areas, considering the current requirements of the industry.

Keywords: corporate information system, software platform, core, interface, API, management system, life cycle.

References

1. Chernyshev L.O., Lebedev V.V., Chernyshev O.L. Survey of software for automation of organization. Problems of informatics in education, management, economics and technics. *Proc. XV Sci. and Tech. Conf. Problems of Informatics in Education, Management, Economy AND Technology*. Penza, 2015, pp. 173–177 (in Russ.).
2. Sitkova M.A. ERP platforms for small and medium enterprises. *Scientific Notes of Young Researchers*, 2014, no. 4, pp. 55–58 (in Russ.).
3. Odintsova M.A. Possibilities of ERP systems for strategic enterprise management. *Politika, Èkonomika i Innovacii*, 2020, no. 4, pp. 1–10 (in Russ.).
4. Chernysh B.A., Kartamyshev A.S., Poturemsky I.V., Khaydukova V.N. Organization of cross-cutting support of customized production of scientific products. *Dynamics of Complex Systems – XXI Century*, 2020, vol. 14, no. 4, pp. 46–54. DOI: 10.18127/j19997493-202004-05 (in Russ.).
5. Terentev K.A. Application of the principles of service-oriented architecture (SOA) at building of information systems. *Proc. Int. Sci. Conf. Problems of Management, Processing and Transmission of Information*, Saratov, 2019, pp. 690–692 (in Russ.).
6. *Aras Innovator*. URL: https://www.aras.com/en (accessed January 9, 2021).
7. *AML Basics*. URL: https://community.aras.com/b/english/posts/aml-basics (accessed January 13, 2021).
8. Ketov D.K. Developing a GraphQL-based API to dynamic structured data. *Proc. Conf. Automated control systems and information technology*, Permian, 2019, pp. 108–114 (in Russ.).
9. *GraphQL Java Home*. URL: https://www.graphql-java.com (accessed January 15, 2021).
10. Knutson M., Winch R., Mularien P. *Spring Security*. Birmingham, UK, Packt Publ., 2017, 542 p.
11. Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley Longman Publ., 1995, 416 p.
12. *Camunda*. URL: https://camunda.com (accessed January 19, 2021).

Для цитирования

Черныш Б.А., Картамышев А.С. Разработка ядра интегрированной информационной системы // Программные продукты и системы. 2021. Т. 34. № 2. С. 237–244. DOI: 10.15827/0236-235X.134.237-244.

For citation

Chernysh B.A., Kartamyshev A.S. The core design of an integrated information system. *Software & Systems*, 2021, vol. 34, no. 2, pp. 237–244 (in Russ.). DOI: 10.15827/0236-235X.134.237-244.