Сервис автоматизации создания интеллектуальных систем на платформе Яндекс.Облако

Ольга А. Николайчук [1], Александр И. Павлов[1]

[1] *Институт динамики систем и теории управления имени В.М. Матросова Сибирского отделения Российской академии наук (ИДСТУ СО РАН), ул. Лермонтова 134, Иркутск, Россия*

Аннотация

В работе проведён анализ возможности использования инфраструктуры Яндекс.Облака для размещения разрабатываемой авторами платформы создания систем, основанных на знаниях. Рассмотрен вопрос реализации пользовательского интерфейса платформы и создаваемых с ее помощью систем. Приведено описание механизма реализации поведения системы, основанной на знаниях, на стороне клиента, а также перечислены использованные ограничения. Приведены примеры реализованных форм пользовательского интерфейса в предметной области надежность механических систем. Осуществлен обзор вариантов реализации облачного сервиса на основе платформы, включая вопрос отчуждения копии, разработанной с помощью сервиса системы.

Ключевые слова

Системы, основанные на знаниях, AI Paas, Яндекс.Облако,

1. Введение

Одной из примет настоящего времени является значительный рост объема информации, как структурированной, так и не структурированной, которую необходимо обрабатывать в процессе выполнения различных исследовательских задач. При этом для целого ряда подобных задач характерно постоянное изменение требований, а также возможность повторного использования полученных результатов в другой предметной области (повторное использование кода). На текущий момент уже существует большое количество программных средств, с помощью которых могут быть решены подобные задачи, начиная от инструментальных средств общего назначения, до специализированных на конкретную задачу библиотек. Однако при этом многие из них обеспечивают решение на общем уровне, и требуют описания специфики решаемой задачи с использованием некоторого языка программирования, что является проблемой для специалиста-предметника (высокие требования, связанные со знанием языка программирования). В то же время, для исследователей, которые могут использовать существующий арсенал программных средств, имеется проблема повторного использования полученных результатов, когда для их повторного использования в другой предметной области требуется адаптация решения на основе знаний предметной области (проблема недостатка предметно-ориентированных знаний). Для решения указанных проблем авторами предложен специализированный инструмент — платформа разработки систем, основанные на знаниях (СОЗ) [1]. Отличием предложенной платформы от существующих инструментальных средств («HiAsm» [2] или «LabVIEW» [3] и т. п.) являются реализация визуального программирования на основе набора базовых операций (функций компонентов) и их комбинаций (композитных операций). При этом композитные операции являются средством обеспечения доступности результатов

16th International Workshop on Information, Computation, and Control Systems for Distributed Environments (ICCS-DE 2024), July 01–05, 2024, Irkutsk, Russia
EMAIL: nikoly@icc.ru (A. 1); asd@icc.ru (A. 2);
ORCID: 0000-0002-5186-0073 (A. 1); 0000−0002−7753−7514 (A. 2);
DOI: 10.47350/ICCS-DE.2024.21

(методов и их программной реализации) для повторного использования за счет возможности добавления их к базовому набору.

Предложенная платформа обеспечивает возможность создания прикладной СОЗ с помощью визуального конструирования требуемого поведения, как императивного, так и декларативного с помощью визуальных редакторов компонентов платформы. Основными компонентами платформы являются: компонент работы с реляционными базами данных (БД) [1], компонент онтологического моделирования предметной области [4,5], компонент разработки продукционных баз знаний [6] и компонент формализации императивного поведения [7].

В данной работе обсуждаются задачи создания пользовательского интерфейса и реализации облачного сервиса на основе программной платформы для разработки СОЗ.

2. Реализация пользовательского интерфейса платформы

В общем случае разработка пользовательского интерфейса (UI) прикладной программной системы является трудоемкой задачей, решение которой может занимать до половины и более от общего количества времени, затраченного на разработку, с учетом практически неизбежного возникновения необходимости в различных изменениях и доработках. В рассматриваемой работе задача создания UI присутствует в следующих подзадачах: организация взаимодействия пользователей с платформой создания СОЗ, с входящими в ее состав компонентами, а также с разрабатываемыми на ее основе прикладными СОЗ. Высокая трудоемкость создания независимого UI при решении каждой из подзадач обуславливает необходимость единого подхода для всех перечисленных случаев. При этом достичь полной стандартизации не представляется возможным ввиду большой вариативности предметных задач. Поэтому в качестве цели для унификации UI выбрана функциональность взаимодействия с отдельными объектами данных, включая просмотр текущего состояния, создание нового, редактирование и удаление существующего; а также работы с массивами (списками) подобных объектов, включая просмотр и фильтрацию по параметрам. В качестве средства для визуального отображения при реализации выбранной функциональности предложено использовать табличное представление.

2.1. Пользовательский интерфейс: табличное представление

Представление информации в табличном виде уже используется в компоненте работы с базами данных [1], в котором имеется возможность выполнения стандартных операций, а также поиска информации для любой из таблиц заданной БД. В настоящее время существует два варианта UI компонента: (1) с использованием jQuery (библиотеки jQueryUI и jQueryGrid), (2) с использованием Vue.js (библиотека Quazar [8]). В качестве основы для реализации стандартного UI платформы, ее компонентов и целевых СОЗ был выбран 1-й вариант.

Компонент работы с базами данных использует спецификацию таблицы БД для создания UI, предназначенного для работы именно с этой таблицей. Для представлния такого рода спецификаций используется модель объекта базы данных, которая на стороне сервера формализуется интерфейсом *IDescriptionOfDatabaseObject* (Рисунок 1), а на стороне клиента представляет собой JSON объект.

Механизм формирования UI компонента работы с базами данных предложено модифицировать следующим образом:

1. добавить возможность сформировать для произвольного объекта его спецификацию в формате, соответствующем *IDescriptionOfDatabaseObject*, при необходимости оставив без значения отсутствующие свойства;
2. добавить возможность представления состояния произвольного объекта в виде массива простых объектов вида: имя/значение, для решения задачи просмотра и редактирования отдельно взятого объекта.

Кроме того, существующий UI платформы и её компонентов предложено заменить на иерархическое структуру, где на верхнем уровне находятся функциональность платформы (представленная компонентами), а ниже полученные результаты, которые в свою очередь

подразделяются на составные части. Так функция «концептуальное моделирование» содержит перечень созданных моделей, которые в свою очередь делятся на понятия, свойства, рабочие области и т. д., а функция «создание продукционных баз знаний» содержит перечень созданных баз знаний, которые в свою очередь делятся на шаблоны, правила, начальные условия.

Рисунок 1: Пример входных данных компонент формирования пользовательского интерфейса в режиме работы с базой данных

Важную роль в предложенной архитектуре UI занимает концепция связанных с выделенной строкой действий, которая заключается в возможности добавить в описание иерархического уровня спецификацию действий, которые могут быть выполнены с выделенной строкой таблицы. Предложенная концепция позволяет в некоторой степени преодолеть относительно низкую выразительность представления UI в виде таблицы. Той же цели служит набор проблемно-ориентированных редакторов, функциональность которых не укладывается в ограничения табличного представления, в качестве примера можно назвать редактор правил или редактор композитных операций. В предложенной архитектуре UI редакторы такого рода реализуются в виде отдельных модулей и вызываются как действия, связанные с выделенной строкой.

Таким образом, для решения задачи обеспечения платформы унифицированным UI предложена концепция отображения иерархической структуры обрабатываемых данных от общего к частному в некоторой аналогии с работой файловых менеджеров, для реализации которой использован компонент формирования пользовательского интерфейса, в функциональность которого входит просмотр и модификация данных произвольных объектов на основе их описания (метаинформации), а также набор специализированных проблемно-ориентированных редакторов оформленных в виде отдельных модулей.

На рисунке 2 показан пример пользовательского интерфейса компонента разработки продукционных баз знаний, отображен экран просмотра продукционных правил, включая перечень действий для выбранного правила, а также окно со специализированным редактором правил.

2.2. Реализация поведения функции СОЗ на стороне клиента

Процесс создания прикладной СОЗ с помощью предложенной платформы включает описание требуемого поведения в виде композитной операции с помощью компонента формализации императивного поведения. В ряде случаев выполнение композитной операции на стороне сервера может быть затруднено, например, когда время выполнения может превысить установленные ограничения, либо когда необходимо взаимодействие с пользователем в процессе выполнения. Поэтому предложено обеспечить возможность

выполнения композитной операции не только на стороне сервера, но и на стороне клиента. Для этого в компонент добавлена функциональность отображения заданной композитной операции в код на языке Javascript. Процесс генерации в целом осуществляется по аналогии с созданием кода на языке PHP, с учетом асинхронных особенностей языка Javascript. Важной особенностью выполнения композитных операций на стороне клиента является возможность ее реализации в виде набора обработчиков для некоторого списка событий, т. е. результатом выполнения кода, сгенерированного для операции, может быть подготовка и запуск инфраструктуры обработки Websocket событий. Данную особенность предлагается использовать для реализации поддержки пользователя в процессе выполнения прикладной задачи, когда пользователь может получать дополнительную информацию и указания в зависимости от состояния наблюдаемых параметров.

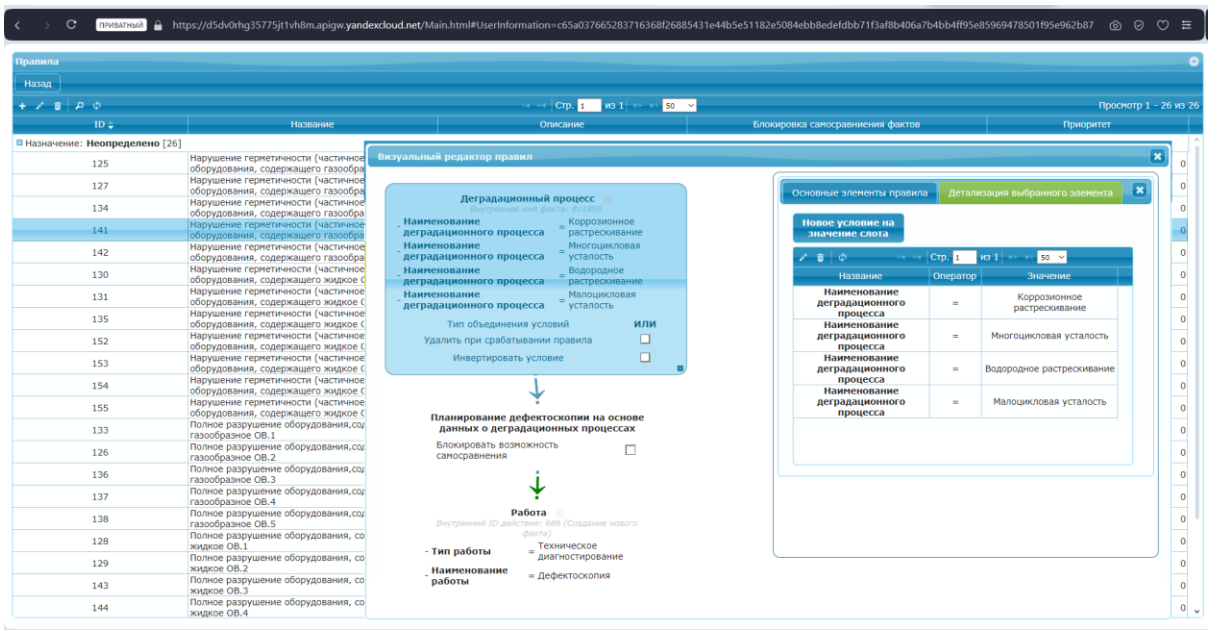

Рисунок 2: Пример UI компонента разработки продукционных баз знаний

Отметим, что в данный момент генерация Javascript кода ограничена возможностью выполнения команд на сервере, которая реализуется с помощью отправки на сервер либо HTTPS запроса, либо Websocket сообщения, а также возможностью отображения полученных в ответ результатах. При этом запуск очередного HTTPS запроса осуществляется из обработчика результатов предыдущего для обеспечения правильной последовательности действий, а в случае использования Websocket заранее создается набор обработчиков для всех используемых типов сообщений.

3. Реализация платформы как облачного сервиса

Платформа разработки СОЗ, как веб-ориентированный программный комплекс, может быть опубликована в рамках инфраструктуры Яндекс.Облака несколькими способами:

1. Виртуальная машина — данный вариант не имеет существенных преимуществ перед обычным «хостингом», не считая наличия услуг по управлению и обслуживанию виртуальных машин, БД и компоненты платформы размещаются на одной виртуальной машине.
2. Группа виртуальных машин — данный вариант обеспечивает возможность обработки изменения входящей нагрузки, количество машин в группе может автоматически увеличиваться при ее увеличении и уменьшаться при ее снижении. Нет возможности разместить БД на одной виртуальной машине с компонентами.

3. «Бессерверная» (*serverless*) реализация на основе «облачной функции» (*cloud function* [9]) — данный вариант характеризуется готовой средой исполнения, и, поэтому, требует меньших усилий при подготовке к работе и обслуживанию в сравнении с вариантами 1 и 2. Есть возможность автоматической обработки увеличения нагрузки — количество экземпляров функции может быть увеличено до 10, а запущенный экземпляр не останавливается сразу, а некоторое время остается доступным для следующего обращения. Нет возможности разместить БД вместе с компонентами, нет доступа к внутренней сети облака, можно использовать только фиксированный набор языков программирования, причем только один для каждой из функций.
4. «Бессерверная» реализация на основе *API Gateway* [10] — данный вариант характеризуется возможностью декларативно описать интерфейс программного комплекса в формате Open API 3.0 (Рисунок 2**)**, а для непосредственной реализации функций использовать как перечисленные выше варианты, файлы, размещенные в «*Object Storage*», включая Docker контейнеры в «бессерверном». Существует возможность автоматической обработки увеличения нагрузки (если таковая имеется у выбранного варианта реализации). Нет возможности разместить БД вместе с компонентами, нет доступа к внутренней сети облака. Есть возможность использования различных языков программирования для реализации функций платформы.

```yaml
openapi: 3.0.0
info:
  title: KBS Platform
  version: 1.0.0
servers:
- url: https://d5dv0rhg35775jt1vh8m.apigw.yandexcloud.net
paths:
  /RetrieveInitializationParameters:
    get:
      x-yc-apigateway-integration:
        type: cloud_functions
        function_id: d4e21p3tolc0h0mvjipf
        tag: $latest
        service_account_id: ajeesfqv5m33t3ap6811
      operationId: kbs-platform-retrieve-ui-parameters-get
    post:
      x-yc-apigateway-integration:
        type: cloud_functions
        function_id: d4e21p3tolc0h0mvjipf
        tag: $latest
        service_account_id: ajeesfqv5m33t3ap6811
      operationId: kbs-platform-retrieve-ui-parameters-post
  /RetrieveData:
    get:
      x-yc-apigateway-integration:
```

```yaml
paths:
  /{file+}:
    get:
      x-yc-apigateway-integration:
        operationId: common-ui-components
        parameters:
        - explode: false
          in: path
          name: file
          required: true
          schema:
            type: string
          style: simple
components:
  securitySchemes:
    httpBearerAuth:
      x-yc-apigateway-authorizer:
        type: function
        function_id: d4empde63oljiugnfvo9
        authorizer_result_ttl_in_seconds: 7200
      scheme: bearer
      type: http
x-yc-apigateway:
  service_account_id: ajehcaavm4u23qqor6lj
```

Рисунок 3: Фрагменты спецификации *API Gateway,* слева реализация функциональности на основе «облачной функции», справа использование «*Object Storage*» для хранения элементов UI и настройки авторизации

Для данной работе был выбран вариант «бессерверной» реализация на основе *API Gateway*, с использованием «облачных функций» для реализации функциональности. Хранение данных организованно на базе кластера PostgreSQL (*Managed Service for PostgreSQL*), по аналогии с изначальной реализацией программного комплекса. В качестве преимуществ выбранного варианта реализации можно выделить декларативную спецификацию, возможность использования возможностей *API Gateway* для реализации рутинных, но важных функций, например авторизацию, возможность использования протокола Websocket и «очереди сообщений» (Message Queue) для организации обмена данными. Кроме того, в рамках *API Gateway* предоставляется служебное доменное имя, которое может быть использовано при проведении тестов.

Возможность использования компонентов платформы в качестве отдельных веб-приложений, а также специфика «облачных функций», которые используют среду PHP,

обуславливают необходимость преобразования входных данных «*облачной функции*» к виду, используемому в компоненте. Для решения этой проблемы предложено использование специализированного компонента-оболочки, который обеспечивает передачу полученных данных в компонент, а также преобразование результатов работы компонента в формат выходных данных «*облачной функции*». В общем случае предложенный компонент-оболочка играет роль шаблона, на основе которого осуществляется создание «*облачной функции*» на основе компонента.

4. Заключение

В данной работе обсуждаются задачи, реализации облачного сервиса на основе программной платформы для разработки СОЗ, а также создания пользовательского интерфейса.

На платформе обеспечена возможность использования пользовательского интерфейса, предоставляемого компонентом автоматизированного формирования пользовательского интерфейса. Данный компонент реализует интерфейс во всех функциях приложения, начиная от перечня его основных функций до диалога с пользователем для ввода недостающих данных, например, по запросу от выполняемой композитной операции. Подобный подход дает возможность, во-первых, закрыть трудоемкую задачу создания пользовательского интерфейса СОЗ, а во-вторых, обеспечить возможность управления поведением клиентской части СОЗ за счет выполнения Javascript-кода, сгенерированного на основе композитной операции.

Использование предложенного компонента-оболочки, как шаблона для реализации компонентов платформы, обеспечивает упрощение процесса разработки, за счет возможности быстрого создания прототипа и применения возможностей уже существующих компонентов платформы. При этом после реализации требуемой функциональности и тестирования есть возможность оформления ее в виде облачной функции или «бессерверного» контейнера для экономии ресурсов.

Реализация платформы для разработки СОЗ в виде облачного сервиса позволяет одной стороны реализовать доступ к функциональности программного комплекса по схеме platform as a service с минимальными затратами на поддержку инфраструктуры, включая вопросы безопасности и обработки меняющейся нагрузки, а с другой при необходимости обеспечить возможность отчуждения разработанной СОЗ путем автоматизированного создания копии в заданном пользователем облаке.

5. Благодарности

Результаты получены в рамках госзадания Минобрнауки России по проекту «Методы и технологии облачной сервис-ориентированной цифровой платформы сбора, хранения и обработки больших объёмов разноформатных междисциплинарных данных и знаний, основанные на применении искусственного интеллекта, модельно-управляемого подхода и машинного обучения» (№ гос регистрации: 121030500071-2).

6. Литература

[1] HiAsm – среда разработки приложений, ориентированная на графическое программирование, https://hiasm.com/

[2] LabVIEW – это среда разработки и платформа для выполнения программ, созданных на графическом языке программирования «G», https://www.ni.com/en/shop/labview.html

[3] Nikolaychuk O.A., Pavlov A.I., Stolbov A.B.: The software platform architecture for the component-oriented development of knowledge-based systems. In: Proceedings of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1234–1239. IEEE (2018). https://doi.org/10.23919/MIPRO.2018.8400194.

[4] Павлов А.И., Николайчук О.А., Столбов А.Б. Редактор концептуальных моделей: Свидетельство о государственной регистрации программ для ЭВМ №2016617626 от 11.07.2016 М.: Федеральная служба по интеллектуальной собственности, патентам и товарным знакам, 2016.
[5] Pavlov A.I., Stolbov A.B. Domain-oriented specialization tools for knowledge-based systems development platform // CEUR Workshop Proceedings: 3rd Scientific-Practical Workshop Information Technologies: Algorithms, Models, Systems (ITAMS 2020; Irkutsk, 3 September 2020). 2020. Vol. 2677.
[6] Павлов А.И., Николайчук О.А., Столбов А.Б. Web-ориентированный редактор продукционных правил: Свидетельство о государственной регистрации программ для ЭВМ №2016663618 от 13.12.2016 М.: Федеральная служба по интеллектуальной собственности, патентам и товарным знакам, 2016.
[7] Pavlov A.I., Stolbov A.B., Dorofeev A.S. The workflow component of the knowledge-based systems development platform // Proc. of 2nd Scientific-Practical Workshop Information Technologies: Algorithms, Models, Systems (ITAMS'2019). 2019. Vol. 2463. pp. 47-58.
[8] Cross-platform VueJs framework, https://quasar.dev
[9] Сервис «Cloud Functions», https://yandex.cloud/ru/docs/functions/concepts/function
[10] Сервис для управления API-шлюзами, https://yandex.cloud/ru/docs/api-gateway/