Обеспечение слабой связности экспертной системы и онтологической базы знаний путём добавления обслуживающего слоя

**Человек:** Цель исследования состоит в предложении метода и прикладных технологий, позволяющих уменьшить время адаптации конкретных баз знаний и экспертных систем друг к другу. Дана оценка текущей архитектурной реализации связи экспертных систем и баз знаний, рассмотрены недостатки — основным недостатком является необходимость переписывать слой сопряжения экспертной системы и базы знаний при любом изменении в протоколе обмена данными между ними, поставлена цель — снизить связность экспертной системы и базы знаний. Архитектура, основанная на слабой связи модулей в системе обширно применяется в других областях. Методами исследования являются методы программной инженерии, дескрипционной логики, инженерии знаний и численные методы. Реализация обслуживающего слоя проводится, исходя из требований к реализации клиент-серверных приложений. Сделано предположение о возможности использования слоя-посредника (обслуживающего слоя), обоснована новизна и эффективность данного подхода. Проведён анализ технологий, позволяющих абстрагироваться от формата данных, предоставляемых базой знаний. Принято решение использовать технологию GraphQL для обмена данными, а промежуточный слой реализовать в виде сервера (в терминах клиент-серверной архитектуры). Отмечается конкретное применение предложенного решения для экспертной системы экологической нагрузки города Волгограда. Внедрение обслуживающего слоя позволило облегчить взаимную адаптацию экспертных систем и баз знаний, уменьшило связность компонентов. В качестве работы на будущее предлагается внедрить в обслуживающий слой компонент для управления доступом и компонент для обслуживания неограниченного круга лиц, например, для предоставления данных об экологической нагрузке населению.

**Key words:** онтология, экспертная система, база знаний, архитектура слабосвязанных модулей, клиент-серверная архитектура, GraphQL, REST API, SOAP, OWL-обёртки, Пролог

=================================

**FastText\_KMeans\_Clean:** <getPlantDetailsResponse>. <PlantName>ООО Завод специального оборудования< / PlantName>. <ID>87645-1364 < / ID>. <AmountT>5< / AmountT>. Также стоит отметить, что после того, как некоторые операции на онтологии станут бессмысленными, например, вследствие удаления части знаний из БЗ, в REST-слое всё равно останутся методы для работы с ними, в то время, если программист серверного части (БЗ) захочет изменить интерфейс обращения к знаниям (например, изменив сигнатуру некоторых методов), то это повлечёт за собой отказ некоторой части функциональности, которую реализуют ЭС, подключённые к через данный слой к БЗ.

**Key words part:** 0.44

=================================

**FastText\_KMeans\_Raw/:** SOAP (сокр. англ. от Simple Object Access Protocol — простой протокол доступа к объектам) является наиболее распространённым методом взаимодействия с сервером, доступно множество инструментов и библиотек для работы по SOAP [5]. <getPlantDetailsResponse>. < / getPlantDetailsResponse>. Также стоит отметить, что после того, как некоторые операции на онтологии станут бессмысленными, например, вследствие удаления части знаний из БЗ, в REST-слое всё равно останутся методы для работы с ними, в то время, если программист серверного части (БЗ) захочет изменить интерфейс обращения к знаниям (например, изменив сигнатуру некоторых методов), то это повлечёт за собой отказ некоторой части функциональности, которую реализуют ЭС, подключённые к через данный слой к БЗ.

**Key words part:** 0.48

=================================

**FastText\_PageRank\_Clean/:** <getPlantDetailsResponse>. <ID>87645-1364 < / ID>. <Name>Бумага< / Name>. <DangerClass>4< / DangerClass>. <AmountT>5< / AmountT>. <IsRecyclable>true< / IsRecyclable>. < / getPlantDetailsResponse>. plantName: "ООО Завод специального оборудования",.

**Key words part:** 0.36

=================================

**FastText\_PageRank\_Raw/:** <getPlantDetailsResponse>. <ID>87645-1364 < / ID>. <Name>Бумага< / Name>. <DangerClass>4< / DangerClass>. <AmountT>5< / AmountT>. <IsRecyclable>true< / IsRecyclable>. < / getPlantDetailsResponse>. plantName: "ООО Завод специального оборудования",.

**Key words part:** 0.36

=================================

**Mixed\_ML\_TR/:** Типовая схема взаимодействия ЭС (логического модуля ЭС) с онтологической БЗ формата OWL показана на рисунке 2. Взаимодействие логического блока ЭС с онтологической БЗ. Новизна предлагаемого подхода заключается в том, что производится унификация взаимодействия экспертных систем с онтологическими базами знаний путём использования метода, который ранее не применялся в подобной области. description: "Выпуск высокоточного оборудования, солнечных панелей",. 3. Архитектура системы взаимодействия ЭС-БЗ после внедрения серверного слоя. В новой версии архитектуры показана работа с OWL, KIF и Prolog-базами знаний. При передаче данных в БЗ, он преобразовывает формализованные данные на уровень ниже, в сущности, понятные конкретному онтологическому модулю (например, KIF [9]). Написание слоя взаимодействия логического модуля и БЗ может занимать около 160 часов с онтологической базой знаний, насчитывающей не более 1000 концептов [10]. За счёт того, что вместе с серверным слоем БЗ фактически становится доступной "как сервис", открывается возможность публикации БЗ для неограниченного круга лиц (например, экологическая БЗ публикуется для населения), которые могут воспользоваться заранее известными методами доступа к знаниям через серверный слой и не зависеть от смены версий БЗ. Итак, в статье был рассмотрен новый подход к обеспечению связи между логическим модулем ЭС и БЗ, обеспечивающий их слабую связность.

**Key words part:** 0.64

=================================

**MultiLingual\_KMeans/:** Новизна предлагаемого подхода заключается в том, что производится унификация взаимодействия экспертных систем с онтологическими базами знаний путём использования метода, который ранее не применялся в подобной области. description: "Выпуск высокоточного оборудования, солнечных панелей",. 3. Архитектура системы взаимодействия ЭС-БЗ после внедрения серверного слоя. В новой версии архитектуры показана работа с OWL, KIF и Prolog-базами знаний. При передаче данных в БЗ, он преобразовывает формализованные данные на уровень ниже, в сущности, понятные конкретному онтологическому модулю (например, KIF [9]). За счёт того, что вместе с серверным слоем БЗ фактически становится доступной "как сервис", открывается возможность публикации БЗ для неограниченного круга лиц (например, экологическая БЗ публикуется для населения), которые могут воспользоваться заранее известными методами доступа к знаниям через серверный слой и не зависеть от смены версий БЗ. Итак, в статье был рассмотрен новый подход к обеспечению связи между логическим модулем ЭС и БЗ, обеспечивающий их слабую связность.

**Key words part:** 0.64

=================================

**Multilingual\_PageRank/:** Однако, существуют и недостатки. Это влечёт за собой дополнительные расходы. Недостатками SOAP являются избыточность информации для обмена, в основном из-за использования языка разметки XML, а также снижение скорости обработки за счёт увеличения объёма передаваемых данных. <Description>Выпуск высокоточного оборудования, солнечных панелей.< / Description>. <ID>87645-1364 < / ID>. <DangerClass>4< / DangerClass>. <AmountT>5< / AmountT>. description: "Выпуск высокоточного оборудования, солнечных панелей",.

**Key words part:** 0.4

=================================

**RuBERT\_KMeans\_Without\_ST/:** Другим преимуществом является то, что OWL API практически не зависит от выбранного языка программирования — на практике программист может использовать одну из множества OWL-обёрток (англ. OWL API-wrappers) для доступа к OWL API. Этот подход имеет такой недостаток как необходимость написания нового слоя взаимодействия каждый раз при изменении формата общения БЗ и ЭС. <PlantName>ООО Завод специального оборудования< / PlantName>. < / getPlantDetailsResponse>. Написание слоя взаимодействия логического модуля и БЗ может занимать около 160 часов с онтологической базой знаний, насчитывающей не более 1000 концептов [10].

**Key words part:** 0.52

=================================

**RuBERT\_KMeans\_With\_ST/:** Типовая схема взаимодействия ЭС (логического модуля ЭС) с онтологической БЗ формата OWL показана на рисунке 2. <PlantName>ООО Завод специального оборудования< / PlantName>. Пример ответа сервера на запрос по REST:. За счёт того, что вместе с серверным слоем БЗ фактически становится доступной "как сервис", открывается возможность публикации БЗ для неограниченного круга лиц (например, экологическая БЗ публикуется для населения), которые могут воспользоваться заранее известными методами доступа к знаниям через серверный слой и не зависеть от смены версий БЗ.

**Key words part:** 0.48

=================================

**RUBERT\_page\_rank\_Without\_ST/:** Общая архитектура экспертной системы. Это влечёт за собой дополнительные расходы. <getPlantDetailsResponse>. < / getPlantDetailsResponse>. description: "Выпуск высокоточного оборудования, солнечных панелей",.

**Key words part:** 0.52

=================================

**RUBERT\_page\_rank\_With\_ST/:** Однако, существуют и недостатки. <ID>87645-1364 < / ID>. <Name>Бумага< / Name>. <AmountT>5< / AmountT>. < / getPlantDetailsResponse>.

**Key words part:** 0.36

=================================

**RUSBERT\_KMeans\_Without\_ST/:** Как частный случай, если планируется переход от OWL к KIF-онтологиям, OWL API больше не будет работать корректно, нужно будет переписывать модуль взаимодействия с БЗ [3]. <getPlantDetailsResponse>. Предполагается, что в один момент времени будет использоваться лишь одна из них, хотя совместное использование не исключается. Эффективность внедрения серверного слоя измеряется в объёме работы, которую не придётся совершать при смене БЗ или ЭС — написание программного слоя, обеспечивающего интероперабельность различных версий и типов ЭС и БЗ.

**Key words part:** 0.44

=================================

**RUSBERT\_KMeans\_With\_ST/:** Актуальность повышения интеграционной способности онтологической БЗ обусловлена тем, что в последнее время появляется всё больше экспертных систем на основе подобных БЗ, авторы которых разрабатывают собственных слой взаимодействия с онтологией [4]. < / getPlantDetailsResponse>. GraphQL решает проблему SOAP, уменьшая избыточность поступающих данных, а также проблему REST — позволяя уже на этапе запроса к серверу определять необходимые для выборки атрибуты и получать от сервера только их вместо целого объекта [8]. Блок онтологической абстракции является связующим звеном уровня API и GraphQL сервера.

**Key words part:** 0.64

=================================

**RUSBERT\_page\_rank\_Without\_ST/:** Однако, существуют и недостатки. <ID>87645-1364 < / ID>. <DangerClass>4< / DangerClass>. <AmountT>5< / AmountT>. <IsRecyclable>true< / IsRecyclable>.

**Key words part:** 0.36

=================================

**RUSBERT\_page\_rank\_With\_ST/:** Однако, существуют и недостатки. <ID>87645-1364 < / ID>. <DangerClass>4< / DangerClass>. <AmountT>5< / AmountT>. <IsRecyclable>true< / IsRecyclable>.

**Key words part:** 0.36

=================================

**Simple\_PageRank/:** Как частный случай, если планируется переход от OWL к KIF-онтологиям, OWL API больше не будет работать корректно, нужно будет переписывать модуль взаимодействия с БЗ [3]. Цель исследования — предложить метод и прикладные технологии, которые позволят уменьшить время адаптации конкретных БЗ к новым экспертным системам, либо наоборот — адаптации экспертных систем для различных БЗ. REST-подход, пришедший на смену SOAP [6], обладает меньшей избыточностью информации вследствие использования языка разметки JSON. Также стоит отметить, что после того, как некоторые операции на онтологии станут бессмысленными, например, вследствие удаления части знаний из БЗ, в REST-слое всё равно останутся методы для работы с ними, в то время, если программист серверного части (БЗ) захочет изменить интерфейс обращения к знаниям (например, изменив сигнатуру некоторых методов), то это повлечёт за собой отказ некоторой части функциональности, которую реализуют ЭС, подключённые к через данный слой к БЗ. За счёт того, что вместе с серверным слоем БЗ фактически становится доступной "как сервис", открывается возможность публикации БЗ для неограниченного круга лиц (например, экологическая БЗ публикуется для населения), которые могут воспользоваться заранее известными методами доступа к знаниям через серверный слой и не зависеть от смены версий БЗ. В будущем планируется введение системы контроля прав доступа к БЗ, что позволит выделить роли и уровни доступа в системе, также намечен запуск онтологической БЗ на публичном веб-сервере для всеобщего доступа.

**Key words part:** 0.64

=================================

**TextRank/:** Типовая схема взаимодействия ЭС (логического модуля ЭС) с онтологической БЗ формата OWL показана на рисунке 2. Актуальность повышения интеграционной способности онтологической БЗ обусловлена тем, что в последнее время появляется всё больше экспертных систем на основе подобных БЗ, авторы которых разрабатывают собственных слой взаимодействия с онтологией [4]. Метрикой эффективности подхода примем количество времени, которое удалось сэкономить при отсутствии необходимости переписывания блока логического модуля под взаимодействие с новым типом (форматом) БЗ. Также стоит отметить, что после того, как некоторые операции на онтологии станут бессмысленными, например, вследствие удаления части знаний из БЗ, в REST-слое всё равно останутся методы для работы с ними, в то время, если программист серверного части (БЗ) захочет изменить интерфейс обращения к знаниям (например, изменив сигнатуру некоторых методов), то это повлечёт за собой отказ некоторой части функциональности, которую реализуют ЭС, подключённые к через данный слой к БЗ. Эффективность внедрения серверного слоя измеряется в объёме работы, которую не придётся совершать при смене БЗ или ЭС — написание программного слоя, обеспечивающего интероперабельность различных версий и типов ЭС и БЗ. Написание слоя взаимодействия логического модуля и БЗ может занимать около 160 часов с онтологической базой знаний, насчитывающей не более 1000 концептов [10].

**Key words part:** 0.6

=================================

**TF-IDF\_KMeans/:** Взаимодействие логического блока ЭС с онтологической БЗ. OWL API выполняет всю работу по выборке, сортировке, фильтрации, выводу новых знаний. <getPlantDetailsResponse>. Недостатком REST можно считать необходимость описывать все возможные методы работы с данными на этапе программирования слоя абстракции. plantName: "ООО Завод специального оборудования",. GraphQL решает проблему SOAP, уменьшая избыточность поступающих данных, а также проблему REST — позволяя уже на этапе запроса к серверу определять необходимые для выборки атрибуты и получать от сервера только их вместо целого объекта [8]. 3. Архитектура системы взаимодействия ЭС-БЗ после внедрения серверного слоя.

**Key words part:** 0.68

=================================

**Текст:** . Рис. 1. Общая архитектура экспертной системы. . Эксперт взаимодействует с БЗ, наполняя её и корректируя. БЗ может быть представлена в различных форматах — логика первого порядка, продукционные правила, в виде правил на основе дизъюнктов Хорна, дескрипционные логики. Именно дескрипционные логики получили широкое распространение в инженерии знаний в виде хранения знаний в онтологических БЗ. Основным форматом физического представления онтологии на сегодняшний момент является формат OWL (сокр. англ. Web Ontology Language — язык сетевых онтологий). Данный формат рекомендован Комитетом Всемирной паутины (W3C) для использования в онтологических БЗ, имеет широкую поддержку со стороны программных инструментов, в наличии имеется открытая и полная документация.. Блок логического вывода отвечает за анализ знаний и вывод новых знаний на основе правил вывода. Для вывода новых знаний используются специальные программные модули — машины вывода (англ. Reasoners).. Блок объяснения — блок, осуществляющий построение цепочки логически связанных фактов из БЗ для объяснения полученного ответа пользователю.. Блок общения обеспечивает взаимодействие логического модуля ЭС с пользователем и представляет информацию на понятном пользователю языке.. Эксперт — лицо, привлекаемое для наполнения БЗ. ЭС позволяет снизить нагрузку на эксперта за счёт взятия на себя части обязательств по предоставлению знаний пользователям.. Пользователь — лицо, которое желает получить поддержку в принятии решения в некоторой предметной области.. Рассмотрим подробнее участок соединения ЭС и БЗ.. Типовая схема взаимодействия ЭС (логического модуля ЭС) с онтологической БЗ формата OWL показана на рисунке 2. Здесь используется промежуточное программное обеспечение (англ. middleware) на основе программного интерфейса приложения (сокр. англ. API) OWL API [2].. . . Рис. 2. Взаимодействие логического блока ЭС с онтологической БЗ. . Рассмотрим преимущества такой схемы соединения. Основное преимущество введения посреднического программного обеспечения состоит в том, что экспертная система не должна напрямую считывать онтологические сущности. OWL API выполняет всю работу по выборке, сортировке, фильтрации, выводу новых знаний.. Другим преимуществом является то, что OWL API практически не зависит от выбранного языка программирования — на практике программист может использовать одну из множества OWL-обёрток (англ. OWL API-wrappers) для доступа к OWL API.. Однако, существуют и недостатки. В частности, главный недостаток такой связки — плотная интеграция ЭС и БЗ. Если в будущем понадобится использовать другую ЭС с заполненной БЗ, либо подключить другую БЗ к существующей ЭС, возникают проблемы интеграции — необходимо переписать слой стыковки. Это влечёт за собой дополнительные расходы. Как частный случай, если планируется переход от OWL к KIF-онтологиям, OWL API больше не будет работать корректно, нужно будет переписывать модуль взаимодействия с БЗ [3].. В ходе научной работы, описываемой в настоящей статье, было проведено исследование, направленное на обеспечение слабой связности ЭС и БЗ на основе онтологических моделей.. Цель исследования — предложить метод и прикладные технологии, которые позволят уменьшить время адаптации конкретных БЗ к новым экспертным системам, либо наоборот — адаптации экспертных систем для различных БЗ.. Актуальность повышения интеграционной способности онтологической БЗ обусловлена тем, что в последнее время появляется всё больше экспертных систем на основе подобных БЗ, авторы которых разрабатывают собственных слой взаимодействия с онтологией [4]. Этот подход имеет такой недостаток как необходимость написания нового слоя взаимодействия каждый раз при изменении формата общения БЗ и ЭС.. В качестве посредника между ЭС и БЗ предлагается использовать обслуживающий (серверный, от англ. Server — обслуживающий) слой. Этот дополнительный слой свяжет клиент (ЭС) и сервер (БЗ). Клиент-серверная архитектура уже успешно работает в других областях, где требуется обеспечить слабую связанность компонентов информационной системы [2].. Новизна предлагаемого подхода заключается в том, что производится унификация взаимодействия экспертных систем с онтологическими базами знаний путём использования метода, который ранее не применялся в подобной области. Метрикой эффективности подхода примем количество времени, которое удалось сэкономить при отсутствии необходимости переписывания блока логического модуля под взаимодействие с новым типом (форматом) БЗ.. В ходе рассмотрения вариантов архитектурных стилей серверного слоя были изучены следующие:. - SOAP-слой. - REST-слой. - GraphQL-слой. SOAP (сокр. англ. от Simple Object Access Protocol — простой протокол доступа к объектам) является наиболее распространённым методом взаимодействия с сервером, доступно множество инструментов и библиотек для работы по SOAP [5]. Недостатками SOAP являются избыточность информации для обмена, в основном из-за использования языка разметки XML, а также снижение скорости обработки за счёт увеличения объёма передаваемых данных.. Пример ответа сервера на запрос в стиле SOAP:. <soap:Envelope>. <soap:Body>. <getPlantDetailsResponse>. <ID>325626< / ID>. <PlantName>ООО Завод специального оборудования< / PlantName>. <Description>Выпуск высокоточного оборудования, солнечных панелей.< / Description>. <Wastes>. <Solidwastes>. <Waste>. <ID>87645-1364 < / ID>. <Name>Бумага< / Name>. <DangerClass>4< / DangerClass>. <AmountT>5< / AmountT>. <IsRecyclable>true< / IsRecyclable>. < / Waste>. < / Solidwastes>. < / Wastes>. < / getPlantDetailsResponse>. < / soap:Body>. < / soap:Envelope>. . REST-подход, пришедший на смену SOAP [6], обладает меньшей избыточностью информации вследствие использования языка разметки JSON. Ответы сервера зависят только от входных параметров (англ. stateless — без состояния). Недостатком REST можно считать необходимость описывать все возможные методы работы с данными на этапе программирования слоя абстракции. Также стоит отметить, что после того, как некоторые операции на онтологии станут бессмысленными, например, вследствие удаления части знаний из БЗ, в REST-слое всё равно останутся методы для работы с ними, в то время, если программист серверного части (БЗ) захочет изменить интерфейс обращения к знаниям (например, изменив сигнатуру некоторых методов), то это повлечёт за собой отказ некоторой части функциональности, которую реализуют ЭС, подключённые к через данный слой к БЗ. Стоит также заметить, что REST не совсем подходит для работы с большими массивами информации, из-за того, что при желании получить некоторый атрибут объекта из списка объектов БЗ, происходит обязательная загрузка полного объекта со всеми его атрибутами (свойствами), и только потом отделяются нужные пользователю свойства [7].. Пример ответа сервера на запрос по REST:. {. id: 325626,. plantName: «ООО Завод специального оборудования»,. description: «Выпуск высокоточного оборудования, солнечных панелей.»,. wastes: {. solidwastes: {. waste: {. ID: '87645-1364',. Name: 'Бумага',. DangerClass: 4,. AmountT: 5,. IsRecyclable: true. }. }. }. }. . GraphQL решает проблему SOAP, уменьшая избыточность поступающих данных, а также проблему REST — позволяя уже на этапе запроса к серверу определять необходимые для выборки атрибуты и получать от сервера только их вместо целого объекта [8]. GraphQL является современной технологией с достаточной документацией и обеспеченностью программными модулями для реализации обмена данными по этому протоколу.. Архитектура сопрягающего модуля после внедрения серверного слоя представлена на рисунке 3.. . . Риc. 3. Архитектура системы взаимодействия ЭС-БЗ после внедрения серверного слоя. . В новой версии архитектуры показана работа с OWL, KIF и Prolog-базами знаний. Предполагается, что в один момент времени будет использоваться лишь одна из них, хотя совместное использование не исключается. Несколько типов БЗ приведено для демонстрации независимости ЭС от типа БЗ при внедрении данного решения.. Блок онтологической абстракции является связующим звеном уровня API и GraphQL сервера. При передаче данных из БЗ, он формализует их на более высоком уровне и передаёт на сервер GraphQL. При передаче данных в БЗ, он преобразовывает формализованные данные на уровень ниже, в сущности, понятные конкретному онтологическому модулю (например, KIF [9]).. Описанный подход был применён на практике в виде разработанного серверного слоя для экспертной системы экологической нагрузки г. Волгограда [3].. Эффективность внедрения серверного слоя измеряется в объёме работы, которую не придётся совершать при смене БЗ или ЭС — написание программного слоя, обеспечивающего интероперабельность различных версий и типов ЭС и БЗ. Написание слоя взаимодействия логического модуля и БЗ может занимать около 160 часов с онтологической базой знаний, насчитывающей не более 1000 концептов [10]. 160 часов — это один человеко-месяц, который будет экономиться при внедрении описанного архитектурного подхода.. За счёт того, что вместе с серверным слоем БЗ фактически становится доступной «как сервис», открывается возможность публикации БЗ для неограниченного круга лиц (например, экологическая БЗ публикуется для населения), которые могут воспользоваться заранее известными методами доступа к знаниям через серверный слой и не зависеть от смены версий БЗ.. Итак, в статье был рассмотрен новый подход к обеспечению связи между логическим модулем ЭС и БЗ, обеспечивающий их слабую связность. В будущем планируется введение системы контроля прав доступа к БЗ, что позволит выделить роли и уровни доступа в системе, также намечен запуск онтологической БЗ на публичном веб-сервере для всеобщего доступа.