УДК 004.2/.3+004.822

АРХИТЕКТУРНЫЕ ШАБЛОНЫ ИНТЕГРАЦИИ РАЗНОРОДНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ НА ОСНОВЕ СЕРВИСНОЙ ШИНЫ

И. Н. Даниленко[1], **Д. А. Кузин**[2], **К. С. Игнатова**[3]
Сургутский государственный университет, [1] *vice1@surgu.ru,* [2] *2repby@gmail.com,*
[3] *ignatova_ks@surgu.ru*

Рассматриваются различные шаблоны интеграции информационных систем в концепции сервис-ориентированной архитектуры. Предложенный вариант интеграции основан на использовании семантической сети в сочетании с архитектурой сервисной шины. Привидится структурная схема архитектурного шаблона интеграции «интеллектуальный сервис-фасад».

Ключевые слова: интеграция информационных систем, сервисная шина, семантическая сеть.

SERVICE-ORIENTED ARCHITECTURAL PATTERNS FOR HETEROGENEOUS INFORMATION SYSTEMS INTEGRATION

I. N. Danilenko[1], **D. A. Kuzin**[2], **K. S. Ignatova**[3]
Surgut State University, [1] *vice1@surgu.ru,* [2] *2repby@gmail.com,* [3] *ignatova_ks@surgu.ru*

In the article described integration patterns of information systems in the concept of service-oriented architecture. Proposed method of integration is based on a semantic network coupled to the service bus architecture. Provides a block diagram of an architectural template integration “intelligent service facade”.

Keywords: information systems integration, service bus, semantic network.

Задача интеграции разнородных информационных систем обычно возникает в следующих типовых случаях:

1) Необходимость обмена данными в корпоративной или государственной сфере между ИС из различных предметных областей. Например, управление производством и финансы в корпоративной среде, или учет прав собственности и налоговое администрирование в системе государственного управления.
2) Централизация системы управления подразделениями и филиалами предприятия (вертикальное интегрирование).
3) Объединение предприятий путем слияния и поглощения.
4) Поддержка унаследованных систем.

Сложность интеграции обычно заключается в том, что изначально она не предполагалась. В этом случае в конечных системах не предусмотрено никакого прикладного API или фасада доступа к данным, отсутствуют соглашения относительно структуры хранимых данных, их семантики и терминологии. Таким образом, интеграция практически всегда требует доработки интегрируемых систем, которая достаточно глубоко затрагивает их архитектуру. Сложность подобной доработки может быть сравнима со сложностью разработки всей конечной системы, при этом степень интеграции, которой удается достичь, может быть различной. Можно выделить следующие основные способы интеграции разнородных информационных систем:

1) Наиболее простым способом интеграции является асинхронная выгрузка/загрузка данных, выполняемая в ручном или полуавтоматическом режиме. При этом совместимость достигается путем использования одного из распространенных форматов хранения данных (XML, как наиболее современный вариант). Реализация процедуры требует полного понимания структуры и семантики хранимых данных в интегрируемых системах.
2) Более глубокая степень интеграции достигается в случае реализации процедур обновления данных в одной системе по запросу из другой системы. При таком варианте реализации помимо работы с данными требуется реализация прикладного API сетевого удаленного доступа с использованием технологий CGI, RPC, CORBA, SOAP и др., а взаимодействие систем называется клиент-серверным, и может быть симметричным или асимметричным.

3) Существенно более сложным в реализации является вариант с использованием промежуточного (связующего) программного обеспечения (middleware) [1]. В настоящее время наиболее прогрессивным подходом к реализации промежуточного ПО считается сервис-ориентированная архитектура – SOA. Для обозначения систем такого класса часто используется термин «сервисная шина» (service bus), хотя последний термин помимо интеграции на основе сервисов включает также интеграцию на основе обмена сообщениями и событийно-ориентированную интеграцию. Существует рынок коммерческого ПО, выполняющего функции service bus, представителями которого, например, являются Oracle Service Bus и IBM WebSphere Enterprise Service Bus.

Наиболее существенный недостаток первых двух способов интеграции – ограничение на количество взаимодействующих конечных систем. С его увеличением количество реализуемых интеграционных процедур будет расти в геометрической прогрессии. На практике подобные процедуры обычно реализуются не более чем для пары взаимодействующих систем. В других случаях уместно использование архитектуры сервисной шины.

Архитектура *сервисной шины (service bus)* предполагает рассмотрение процессов интеграции с точки зрения межсервисного взаимодействия (SOA), передачи сообщений или обработки событий. При этом определяется архитектурный шаблон интеграционного решения, тип взаимодействия, модель безопасности, эксплуатационные и другие аспекты [2]. К архитектурным шаблонам интеграции в концепции SOA относятся следующие (рис.1):

1) *Поддержка сервис-фасада* – решает проблему доступа к конечным системам, не имеющим вид сервиса, посредством сервис-ориентированного интерфейса. Для API унаследованных систем должен быть реализован соответствующий адаптер сервиса. Возможна и реализация обратной задачи – организация точки входа для приложений, которые не могут использовать SOA.
2) *Виртуализация сервиса* – реализует способ взаимодействия между сервисами, используя дополнительные уровни абстракции, т.е. представления сервиса в виде виртуального сервиса с расширением функционала или приведением к необходимому стандарту. Может использоваться для распределения запросов между несколькими альтернативными сервис-провайдерами.
3) *Шлюз* – осуществляет необходимые согласования запросов на границе взаимодействия сервисов, не затрагивая при этом прикладной (верхний) уровень сообщений. При помощи шаблонов типа «шлюз» решаются задачи построения единой системы безопасности, преобразования различных протоколов SOA (например, SOAP и XML RPC или REST), а также журнализации и аудита.
4) *Интегрирующий сервис-фасад* – включает интеграционную логику для использования нескольких сервисов из различных конечных или унаследованных систем для создания бизнес-логики более высокого уровня. Представляет интерес как основа для интеграции систем на основе общей семантики данных.

Архитектурные шаблоны интеграции *на основе обмена сообщениями* (message queue) во многом схожи с шаблонами виртуализации сервиса. Различие заключается в том, что в шаблонах на основе обмена сообщениями основное внимание уделяется потокам данных, проходящим между компонентами middleware и конечными системами, а также набору действий, которые могут быть применены к этим данным. К наиболее распространенным архитектурным шаблонам на основе обмена сообщениями можно отнести: маршрутизатор сообщений, транслятор сообщений, мост сообщений, агрегатор сообщений, разделитель сообщений [2].

Сервисная шина также может быть построена на основе *событийно-ориентированной* архитектуры, которая в настоящее время не имеет общепринятого представления. Тем не менее, можно выделить несколько основных шаблонов для архитектуры данного типа. К ним относятся маршрутизатор событий – распределяет события по подписчикам, фильтр событий – осуществляет препроцессинг событий, а также реактор событий, осуществляющий синхронизацию связанных событий и поддержку обратных сообщений по цепочке событий.

Задачу интеграции конечных информационных систем будем рассматривать исходя из следующих основных ограничений:

1) Интегрируемые системы являются полностью разнородными, т.е. могут быть реализованы на разных платформах и иметь принципиально различную архитектуру.
2) Внутренняя организация данных конечных систем скрыта с точки зрения промежуточной системы, а используемая в конечных системах модель данных не обязательно является реляционной.

Рис. 1. *Архитектурные шаблоны интеграции в концепции SOA*

3) Конечные системы проектируются и поддерживаются независимыми коллективами разработчиков в отсутствии единого концептуального подхода. Соглашения об именах, классификаторах и справочниках могут отсутствовать.
4) Конечные информационные системы не имеют общей системы безопасности.
5) Способом взаимодействия с конечными системами является сервис-ориентированный «фасад». При его отсутствии он должен быть реализован в виде специальных адаптеров.
6) Функционал сервис-ориентированного фасада определяется разработчиком конечной системы и может состоять из однонаправленных команд, запросов на чтение или изменение с подтверждением.

Общая концепция интеграции заключается в том, что промежуточная система, построенная по архитектуре сервисной шины, предоставляет клиенту набор возможных сценариев взаимодействия, таких, например, как:

1) Однонаправленные взаимодействия.
2) Запросы на чтение и изменение (синхронные или асинхронные).
3) Взаимодействия типа публикация/подписка.

При этом внутренняя архитектура сервисной шины обеспечивает реализацию данных взаимодействий посредством доступа к конечным системам на основе архитектурных шаблонов интеграции, описанных выше. Как упоминалось, наибольший интерес представляет шаблон с использованием логики интеграции, которая позволяет клиенту абстрагироваться от функционала конечных систем. При этом роль промежуточной системы в обеспечении интеграции принципиально меняется – от техноло-

гической к прикладной.

По мнению авторов, наиболее прогрессивным подходом в реализации интеграционной логики является использование семантических технологий [3], что дает ряд преимуществ в условиях разнородности интегрируемых систем [4]:

1) «Открытость» модели данных, предусматривающая ее расширение путем добавления новых концептов и отношений в течение всего жизненного цикла системы.
2) Возможность моделирования сложных связей и отношений и применение нечеткого логического вывода.
3) Использование согласованной (разделяемой всеми) терминологии с точно определенной семантикой.

Использование семантической сети для интеграции данных из разнородных систем описано, например, в работах [5,6,7]. Общим для перечисленных работ является создание семантической сети с непосредственной привязкой к полям данных исходной реляционной БД. При этом осуществляется соответствующее отображение свойств концептов на поля таблиц, либо полная конвертация исходных реляционных БД в структуру RDF-графа. Коммерческим продуктом класса middleware, использующим технологию Semantic Web является система «Бизнес-семантика» [8,9], которая имеет архитектуру шины обмена сообщениями (message queue). В этой системе семантическая сеть (или ее фрагменты на интегрируемых системах) используется для отображения полей сообщений, передаваемых в формате RDF, на поля БД. При этом сообщения содержат информацию об изменившихся данных, которые вносятся в БД интегрируемых систем.

По мнению авторов, наиболее продуктивным подходом к интеграции разнородных систем является сочетание архитектуры сервисной шины и интеграционной логики на основе семантической сети (онтологии). При этом онтология не хранит данные предметной области, а извлекает необходимые данные из БД конечных систем в соответствии с архитектурным шаблоном «интегрирующий сервис-фасад» для выполнения операций логического вывода.

Одной из актуальных задач в области интеграции является организация прозрачного взаимодействия информационных систем, используемых в сфере государственного управления. В настоящее время в России создана и эксплуатируется «Система электронного межведомственного взаимодействия (СМЭВ)» [10,11]. Основным назначением СМЭВ является получение сведений из различных ведомственных информационных систем в электронном виде для зарегистрированных клиентов. Система построена по архитектуре сервисной шины на базе IBM WebSphere ESB и решает следующие задачи:

1) Маршрутизация SOAP -запросов к web-сервисам поставщиков информации в соответствующий региональный сегмент системы на основе кодов ОКТМО.
2) Реализация синхронного и асинхронного режима взаимодействия.
3) Поддержка каталога web-сервисов на основе WSDL.
4) Обеспечение безопасности и разграничения доступа на основе инфраструктуры ЭЦП.
5) Протоколирование, сбор статистики и мониторинг доступности сервисов.

СМЭВ позволяет клиенту в автоматическом режиме извлекать различную информацию преимущественно учетного характера из ведомственных и региональных информационных систем в электронном виде. Для конечных систем-поставщиков информации регламентируются только технологические аспекты подключения к системе. Состав и функционал сервисов, а также структуру сообщений на прикладном уровне полностью самостоятельно определяют разработчики конечной системы. Это обуславливает ряд серьезных ограничений концептуального характера, накладываемых на область применения и перспективы развития системы. С ростом числа сервисов (на сегодняшний день свыше 500) их каталогизация, документирование и мониторинг будут становиться все более сложными задачами. Отсутствие семантических связей между различными сервисами не позволяет полноценно вести обмен данными между конечными системами, а существующая «плоская» система каталогизации с использованием примитивных фильтров не обеспечивает эффективного поиска необходимого сервиса. Несмотря на большой объем фактически хранимых данных, использование их для аналитических целей и построения сложных сервис-фасадов затруднено.

Перечисленные проблемы могут быть решены с помощью построения онтологии, которая определит единую семантику для всех конечных систем с возможностью ее расширения и позволит реализовать процедуры для работы с данными из разнородных систем через «интеллектуальный» сервис-

фасад (рис. 2). Ввиду того, что данные хранятся в конечных системах, потребуются специальные процедуры по актуализации RDF-хранилища, которое будет выполнять роль «кэша» для интеллектуальной системы. Поскольку специфика данной архитектуры заключается в отсутствии доступа к реляционным данным прикладных систем, то актуализация RDF-хранилища будет выполняться специальным агентом, извлекающим требуемые данные посредством обращения к web-сервисам прикладных систем.

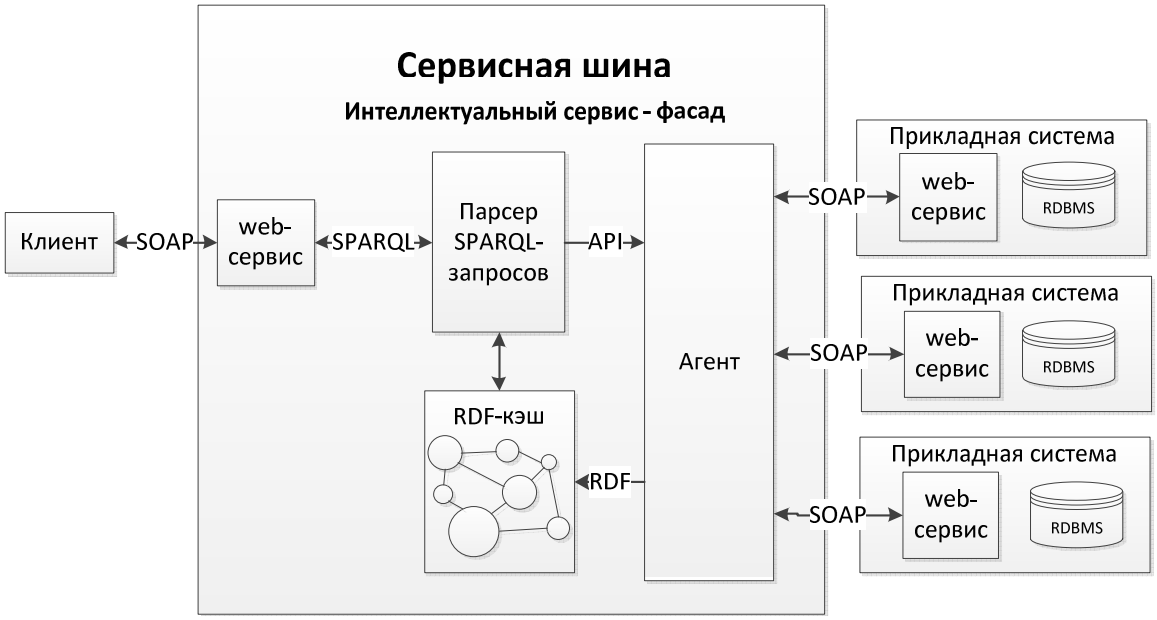

Рис. 2. *Шаблон интеграции «интеллектуальный сервис-фасад»*

ЛИТЕРАТУРА

1. Касаткин А. Средства middleware и их классификация. 1999. Режим доступа : http://www.pcweek.ru/idea/article/detail.php?ID=51098.
2. Уайли Х., Ламброс П. Шаблоны взаимодействия приложений в корпоративных системах: Интеграционные решения с использованием продуктов IBM Enterprise Service Bus. 2010. Режим доступа : http://www.ibm.com/developerworks/ru/library/ws-enterpriseconnectivitypatterns/.
3. Андон Ф. И., Гришанова И. Ю., Резниченко В. А. Semantic Web как новая модель информационного пространства Интернет. 2012. Режим доступа : http://shcherbak.net/semantic-web-kak-novaya-model-informacionnogo-prostranstva-internet/.
4. Гончар А. Д. Сравнительный анализ баз данных и баз знаний (онтологий) применимо к моделированию сложных процессов // Научно-практический журнал «Современные научные исследования и инновации». 2014. Режим доступа : http://web.snauka.ru/issues/2014/05/34325.
5. Клинцов В. П. Интеграция приложений, использующих реляционные СУБД. 2015. Режим доступа : https://gkpromtech.ru/material/view?id=159.
6. Биряльцев Е. В., Гусенков А. М. Интеграция реляционных баз знаний на основе онтологий. 2007. Режим доступа : http://www.mathnet.ru/links/8c682cdb3ec0c19e9d3ba6d4fbca6146/uzku603.pdf.
7. Виноградов И. Д. Построение интергрированных онтологий предметных областей из разнородных источников информации. 2013. Режим доступа : www.ssc.smr.ru/media/ipuss_conf/15/5_13.pdf.
8. Сервисная шина "Бизнес Семантика". 2012. Режим доступа : http://www.business-semantic.ru/products/bus.
9. Горшков С. Технологии Semantic Web для интеграции информационных систем. 2013. Режим доступа : http://habrahabr.ru/post/167419/.
10. Простаков А. Общее представление о системе межведомственного электронного взаимодействия в Российской Федерации. 2014. Режим доступа : https://www.ibm.com/developerworks/ru/library/cmeb_01/.
11. Левашов А. С. СМЭВ — ядро электронного правительства России // CNews. 2013.