АНАЛИЗ ПЕРСПЕКТИВ ПЕРЕХОДА ИНФОРМАЦИОННЫХ СИСТЕМ НА СЕРВИСНО-ОРИЕНТИРОВАННУЮ АРХИТЕКТУРУ

Ю. Г. Платонов

Институт систем информатики СО РАН, 630090, Новосибирск, Россия

УДК 62-52

Рассмотрена стандартная информационная система с клиент-серверной архитектурой, проведено сравнение особенностей клиент-серверной и сервисно-ориентированной архитектур. Сформулирован критерий перевода системы на сервисно-ориентированную архитектуру, основанный на результатах нагрузочного тестирования систем. В качестве примера перевода информационной системы на сервисно-ориентированную архитектуру описана система АСПИД, разработанная для ОАО ИСС им. М. Ф. Решетнева.

Ключевые слова: сервисно-ориентированная архитектура, клиент-серверная архитектура, АСПИД, распределенная база данных, распределенный файл-репозиторий.

In the article the author considers the standard information system with the client-server architecture and compares features of the client-server and service-oriented architectures. The author presents the criterion of the necessity of the using of the service-oriented architecture for the system. The criterion is based on the stress testing results. Also the article describes the ASPID Automated Information System, developed for Information Satellite Systems - Reshetnev Company in Geleznogorsk, as a sample of the re-design.

Key words: service-oriented architecture, client-server architecture, ASPID Automated Information System, distributed database, distributed repository of files.

Введение. В настоящее время на предприятиях широко применяются информационные системы, использующие клиент-серверную архитектуру. Большинство этих систем разработаны в 1990-е гг., когда такой подход к проектированию считался инновационным.

Данная технология имеет ряд преимуществ [1], но в настоящее время существуют более совершенные способы разработки и внедрения информационных комплексов [2]. Кроме того, наиболее существенные недостатки клиент-серверной технологии проявляются позднее, что обусловлено естественным увеличением размера баз данных (БД), файл-репозитория и пр.

В данной работе проанализирована стандартная информационная система с клиент-серверной архитектурой и оценена целесообразность ее перевода на более современные виды архитектуры, приведен пример такого перехода. Предложен критерий оценки необходимости ведения новых разработок, позволяющий оценить целесообразность внедрения более современных технологий в конкретном случае. Также проблемы эксплуатации могут быть решены путем использования менее затратных, но широкодоступных в настоящее время “половинчатых” решений [2].

В качестве примера эволюции информационной системы рассмотрена система АСПИД (архив сопровождения программных проектов и документов), разработанная для ОАО ИСС им. М. Ф. Решетнева (г. Железногорск) [3].

Рис. 1. Клиент-серверная архитектура информационной системы

1. Характеристика сервисно-ориентированного подхода. Сервисно-ориентированный подход следует рассматривать как наиболее прогрессивный путь развития информационных систем, сменивший клиент-серверную технологию проектирования (рассматриваются только системы, полностью разработанные и внедряемые в настоящее время). Суть подхода заключается в том, что весь функционал сервера приложений разбивается на независимые слабосвязанные заменяемые сервисы-приложения, связь между которыми осуществляется через специальный компонент с помощью сообщений со специфицированными заранее протоколами. При этом на отдельном физическом сервере может быть зарегистрирован как один сервис, так и несколько. В этом случае клиентское приложение имеет доступ только к компоненту, отвечающему за передачу данных между сервисами, поэтому его также можно рассматривать как специализированный сервис информационной системы.

Компонент, отвечающий за взаимосвязь сервисов, должен помимо прочего равномерно распределять общую нагрузку системы между доступными сервисами (рис. 1), что обычно реализуется путем организации очередей сообщений, разделенных исходя из логических соображений.

Кратко опишем преимущества и недостатки сервисно-ориентированного подхода.

Преимущества:

— хорошая масштабируемость данных и файл-репозитория (можно легко увеличить количество серверов при чрезмерной нагрузке на сервер данных);

— относительная простота программного кода, так как он рассредоточен в отдельных сервисах;

— хорошая интегрируемость, поскольку каждый сервис изолирован и отвечает за выполнение определенной задачи;

— хорошее эволюционирование, поскольку сервисы независимы и их соединение происходит только через протоколы; сервисы могут быть быстро заменены, а их список — легко пополнен;

— оптимальное использование вычислительных ресурсов за счет использования сервера сообщений;

— ускорение работы системы в целом за счет параллельной обработки сложных процессов.

Недостатки:

— сложность конфигурирования системы;

— высокая стоимость технологии, обусловленная необходимостью полного обновления программного кода системы при ее переводе на сервисно-ориентированную архитектуру;

— возможность нарушения синхронности работы системы, связанная со спецификой ряда сервисов (в некоторых случаях эта особенность может являться преимуществом).

2. Анализ целесообразности перевода информационной системы на сервисно-ориентированную архитектуру. В настоящее время существуют методы решения любых проблем, возникающих при эксплуатации большинства систем, спроектированных на клиент-серверной основе, однако, по сути, ни один из них не позволяет полностью решить эти проблемы.

В существующих условиях главным фактором, препятствующим широкому распространению сервисно-ориентированной технологии является ее дороговизна, обусловленная необходимостью полностью обновлять программный код системы. Как правило, заказчик, затративший в свое время достаточно большой объем трудовых и денежных ресурсов на разработку и внедрение информационной системы, не готов финансировать дальнейшие крупномасштабные разработки и придерживается консервативных позиций.

Таким образом, несмотря на архаичность разработанных 10–15 лет назад систем, они продолжают повсеместно использоваться. При этом зачастую областью применимости подобных систем являются наукоемкие отрасли промышленности, использующие сложные технологические процессы. Кроме того, существует достаточно большое число систем с повышенной ответственностью за конечный продукт. Такие информационные системы должны обеспечивать высокий уровень безопасности данных, надежное хранение больших объемов информации в течение длительного времени, стабильно поддерживать сложные связи между объектами и т. д. [3].

Информационную систему с повышенной мерой ответственности за конечный продукт будем называть системой с повышенными требованиями, или системой ПТ. Ниже рассматривается такая система с клиент-серверной архитектурой.

Система включает следующие компоненты: большое количество клиентских приложений, выделенный сервер приложений (application server), соединенный с сервером данных (data server) и сервером-репозиторием (file storage) (рис. 2).

Преимущества системы с клиент-серверной архитектурой:

— простота обслуживания;

— относительная надежность;

— наглядность и простота архитектуры взаимодействия.

Недостатки системы с клиент-серверной архитектурой:

— проблема эволюционизма (в готовую систему сложно добавлять новые компоненты и технологии);

— проблема масштабируемости (ограниченная расширяемость файла-репозитория, сервера приложений и сервера данных);

— если система сложная, то код сервера и клиента может быть труднопонимаемым, что может порождать дублирование и скрытые ошибки как на нижнем уровне функции передачи и обработки данных, так и на верхнем уровне кодирования.

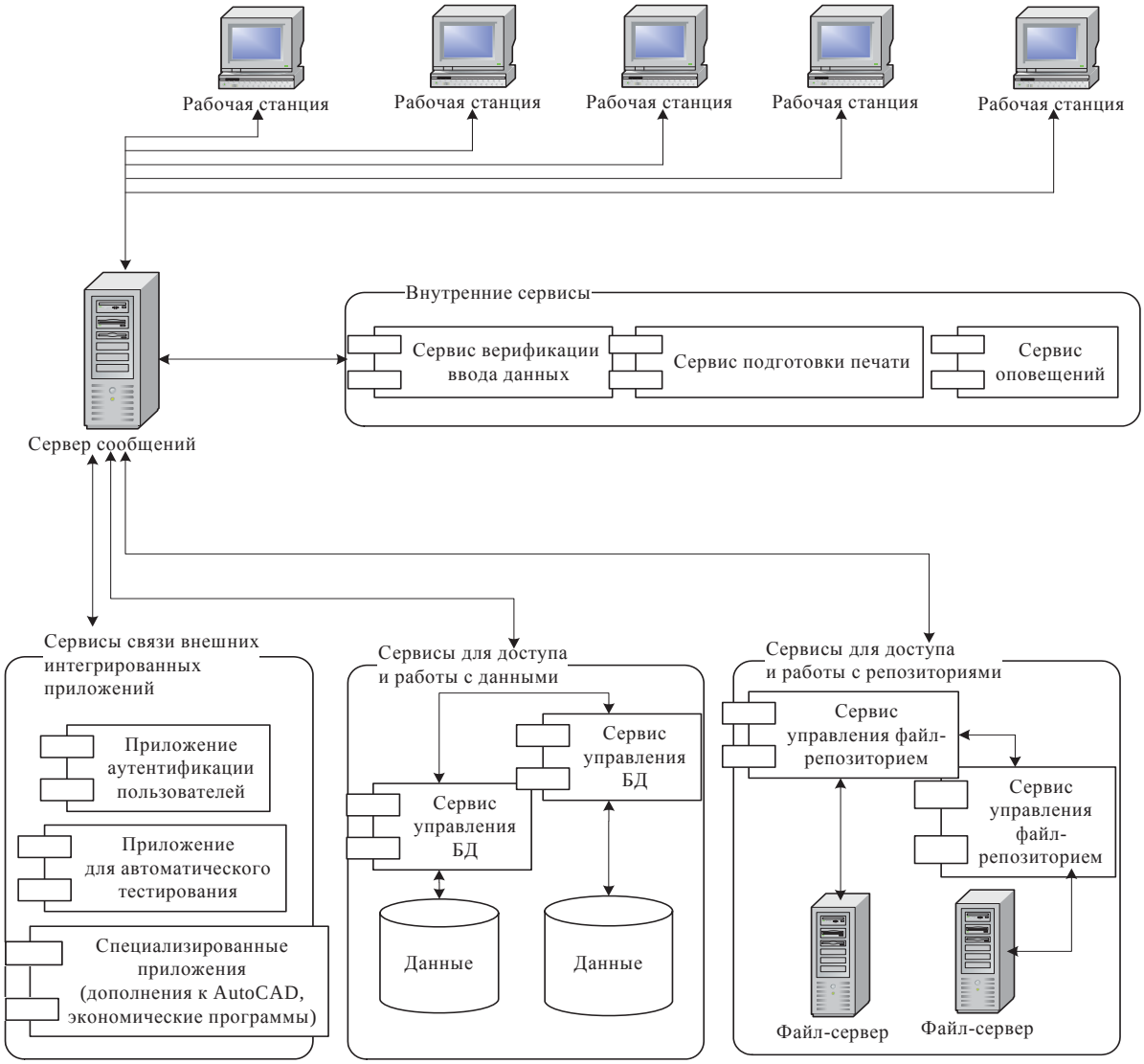

Рис. 2. Сервисно-ориентированная архитектура информационной системы

Все программы с ПТ имеют сложный наукоемкий жизненный цикл программного продукта [3]. Технологический цикл с момента проектирования системы до момента ее списания и утилизации включает большое количество сложных стадий.

Эксплуатация такой системы также представляет собой весьма сложный цикл и требует поддержания объемного файл-репозитория, в силу чего БД такой системы имеет заведомо большие размеры.

Кроме того, в силу специфики каждого участка работ технологический процесс на таких предприятиях автоматизирован не полностью: одна часть производственных циклов не включена в систему, а другая часть может описываться набором независимых или слабосвязанных систем. Ограничения на автоматизацию процессов обусловлены не только отсутствием у разработчиков специальных знаний о технологическом процессе, но и уровнем доступа к данным.

Таким образом, традиционный способ решения проблемы автоматизации подобных производств имеет фрагментарный характер, т. е. допускает отсутствие компьютерной обработки данных на одних участках и полную автоматизацию процессов на других. Кроме того,

на некоторых участках жизненного цикла используются специализированные программы с закрытым кодом и данными, т. е. не допускающие изменений. Между тем такие программы иногда играют ключевую роль в жизненном цикле программного продукта (например, автоматическое тестирование программного продукта на системах, имитирующих специфический реальный процесс, либо авторизация пользователя какими-либо электронными системами, либо даже широко распространенные пользовательские программы типа Autocad). В этих случаях возникает проблема невозможности изменения "своей" традиционной системы.

Проведенное исследование позволяет определить, в каких случаях перевод системы на сервисно-ориентированную архитектуру необходим, а в каких случаях достаточно выполнить ряд стандартных операций по улучшению эксплуатационных качеств системы без изменения ее клиент-серверной архитектуры.

Для анализа использован набор небольших систем с ограниченным набором характеристических функций, в том числе с одной функцией. При этом были выбраны поддерживаемые каждой из систем функции, наиболее полно описывающие поведение исследуемой характеристики системы, например получение объектов из БД, создание объектов в БД, их обработка, получение файлов по конкретному документу из репозитория.

Проведено стресс-тестирование систем, результаты которого позволили сделать следующие выводы.

Если на предприятии требуется автоматизировать производственный процесс или провести модернизацию существующей системы, то в качестве технологии проектирования следует использовать сервисно-ориентированный подход при следующих условиях:

1) система должна поддерживать работу большого числа пользователей;

2) размеры файл-репозитория и БД системы должны быть достаточно велики или иметь возможность быстро увеличиться при эксплуатации;

3) система должна обеспечивать интеграцию со специализированным программным обеспечением, разработанным третьими фирмами.

В условиях сложного наукоемкого производства, когда имеет место большинство указанных проблем, не следует использовать клиент-серверную архитектуру. В данной ситуации единственно возможным остается сервисно-ориентированный подход.

Наиболее наглядно технологию модернизации системы можно представить на примере работы пользователя с документами [3]. Для того чтобы пользователь мог отслеживать весь процесс документооборота и управлять им, а также контролировать состояние библиотеки архивов, необходимо разработать и интегрировать в систему модуль "Электронный документооборот". Соответственно будет изменена клиентская часть приложения: функцию модуля "Электронный документооборот" будет выполнять централизованное пользовательское клиентское приложение. Модуль будет отображать состояние документов, позволит просматривать библиотеку программных продуктов и документов, в частности отчеты об изменении документов, влияющих на производственный процесс (например, конструкторских и связанных с ними), а также получать отчеты специализированных служб типа службы тестирования или автоматической верификации кода. В качестве субъектов таких служб могут выступать как люди, так и роботы, но в любом случае пользователь сможет ознакомиться с отчетами о результатах их работы. Соответственно пользователь модуля получит возможность адекватно участвовать в производственном процессе.

3. Развитие АСПИД как пример эволюции информационной системы. В качестве примера создания единой информационной системы на наукоемком производстве с

повышенной степенью ответственности за конечный продукт рассмотрим проектирование и дальнейшую реализацию программного комплекса АСПИД.

Перед началом проектирования система автоматизации производственного процесса представляла собой набор несвязанных специализированных программ, которые автоматизировали конкретные этапы производственного процесса, и большой файловый архив программного кода.

Поскольку к моменту начала проектирования производственный процесс был жестко структурирован и специфицирован, архитекторы сочли целесообразным применить метод проектирования "сверху вниз".

Другая важная задача, которую потребовалось решить в первую очередь, — каталогизация файлового архива. Стандартным решением создания версифицированных файловых репозиториев может считаться CVS [4] или другой программный комплекс данного семейства.

Однако в дальнейшем архитекторы были вынуждены отказаться от стандартного решения и создали дополнительные модули "Архив программ" (АП) и "Архив исходного кода" (АИ), поскольку дополнительный анализ структуры репозитория выявил следующие проблемы:

1. Объекты архивов имеют сложную разнородную структуру, включающую различное количество файлов. При этом типы объектов архива, принадлежащие одному уровню подчинения, могут быть разными.

2. Для управления объектами архивов и их последующей интеграции в единую информационную систему разработчикам потребовалось бы создавать некоторую "инфраструктуру-обертку" даже при использовании решений типа CVS.

Следует отметить изящность решения, предложенного архитекторами для создания разнородных каталогов архивов программ и исходного кода на этом этапе проектирования. Каждый тип объекта архива представлял собой универсальный сериализованный объект с набором атрибутов, характерных для объекта архива данного типа. При этом имя типа объекта архива также сохранялось как отдельный атрибут, после чего однородный каталог сохранялся в базе данных как XML-объект.

В настоящей работе под термином "сериализация" понимается процесс преобразования какой-либо структуры данных в последовательность битов, а под термином "десериализация" — процесс восстановления начального состояния структуры данных из битовой последовательности, т. е. процесс, обратный сериализации [5].

Обычно сериализация используется для передачи объектов по сети и сохранения их в виде файлов. В частности, ее применение целесообразно, если для различных частей распределенного приложения необходимо обеспечить корректный обмен данными со сложной структурой.

В рассматриваемом случае для создания объектов можно использовать стандартные процедуры сериализации-десериализации, а поиск данных осуществлять с помощью достаточно простых встроенных процедур, используя в качестве ключей регулярные выражения. Заметим, что при использовании базы, основанной на тестовых замерах, скорость работы процессов поиска, создания и обновления объектов оказалась такой же, как и у модели, основанной на плоских таблицах.

На следующем этапе автоматизации было решено создать централизованное рабочее место пользователя и информационную подсистему "Электронный документооборот" для автоматизации процесса управлениядокументами. До принятия решения о создании специализированной подсистемы был проведен анализ готовых решений [3]. Результатом данного этапа проектирования и последующей реализации стала разработка контейнера данных и подсистемы,

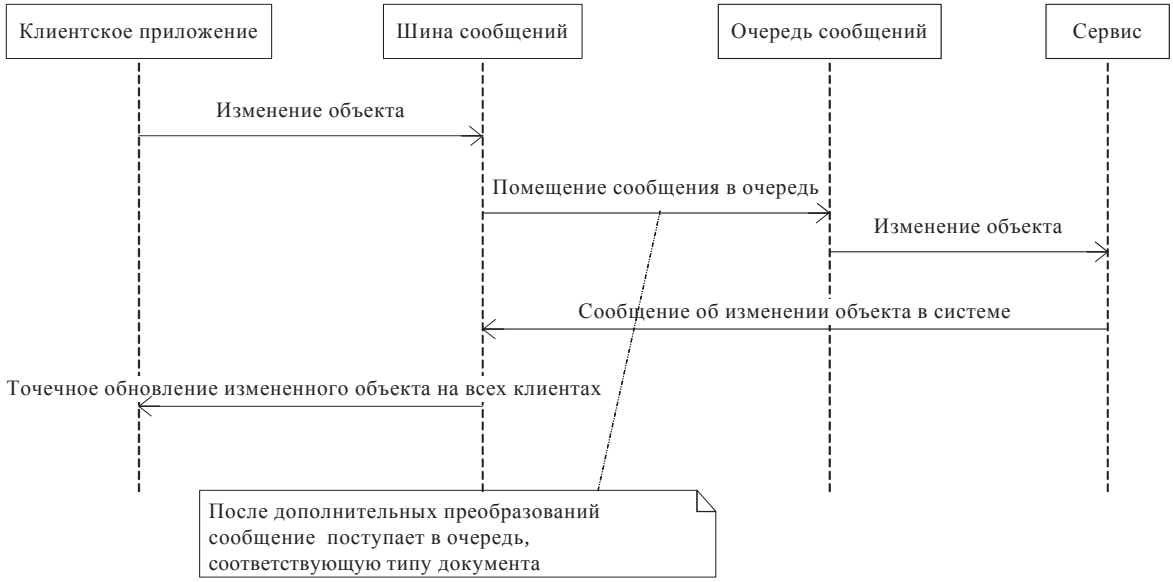

Рис. 3. Реализация запроса на изменение объекта архива с помощью шины сообщений

обслуживающей файл-репозиторий, а также интеграция с подсистемами “Архив программ” и “Архив исходного кода”, создание гибкого, легкомодифицируемого рабочего места клиента и информационной подсистемы “Электронный документооборот”. Таким образом, на этом этапе была получена информационная система, спроектированная с использованием трехзвенной клиент-серверной архитектуры [6]. Следует отметить, что серверная часть информационной системы создавалась с учетом возможного перехода к сервисно-ориентированному подходу. По завершении этапа при переводе подсистемы в промышленную эксплуатацию система была модернизирована с использованием сервисно-ориентированной архитектуры.

При проектировании в качестве шаблона был использован CQRS (см. официальный блог Г. Юнга http://codebetter.com/gregyoung).

Отметим наиболее важные доработки в системе, выполненные между этапами:

1. Серверная часть разбита на отдельные сервисы.

2. Четко специфицированы все протоколы взаимодействия между сервисами, а также протоколы ввода-вывода внешней информации.

3. На основе идеи интеграционной шины сообщений [7] и очереди сообщений MSMQ [8] создан сервер сообщений (рис. 3).

4. Созданы специализированные сервисы, способные согласованно управлять как файл-репозиторием в целом, так и его отдельными частями. Для этого в систему добавлен ряд специальных настроек, с помощью которых сервис определял маску документа, которым он управляет. После регистрации на сервере сообщений сервис ожидал получения сообщения определенного типа с определенной маской по специальной шине сообщений. Следует отметить, что при таком проектировании сервиса для управления файл-репозиториями неизбежно возникает проблема обработки файлов документа при каждом изменении настроек сервиса. Возможны три варианта этой проблемы.

1. После изменения настроек может возникнуть ситуация, когда несколько документов, файлы которых ранее находились под управлением данного сервиса, не соответствуют новому набору масок обслуживаемых объектов и могут оказаться недоступными для остальных сервисов системы.

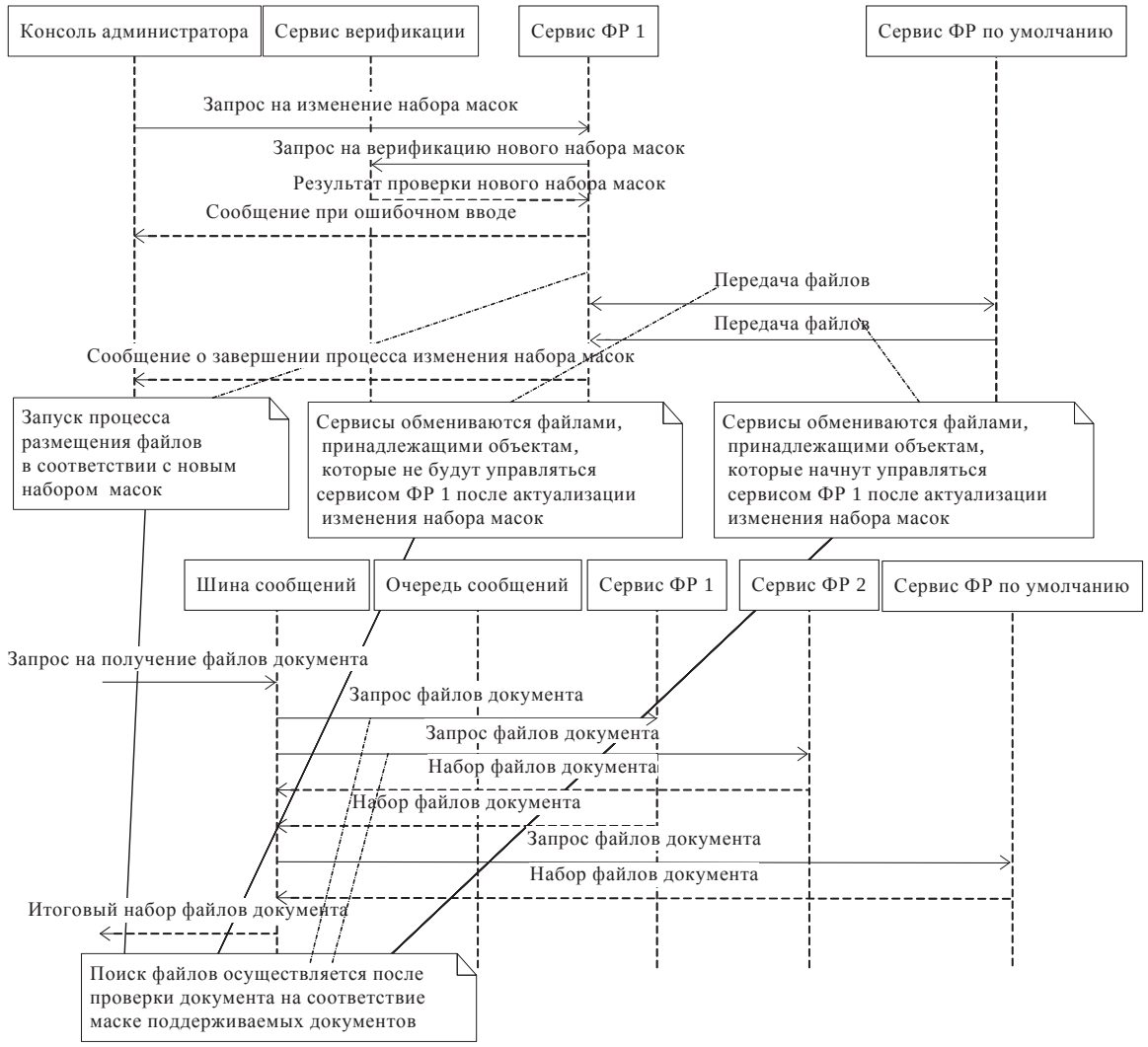

Рис. 4. Схема работы сервиса управления файл-репозиторием (ФР). Процесс изменения набора масок документов, обслуживаемых сервисом управления файл-репозитория

2. По завершении общей конфигурации может оказаться, что совокупного набора масок недостаточно для перечня возможных документов.

3. Существует вероятность того, что набор масок двух или нескольких сервисов частично совпадает. Это может вызвать дублирование файлов и возникновение конфликтов при дальнейшем управлении.

Для решения данной проблемы введен дополнительный сервис управления файл-репозиториями по умолчанию, предназначенный для обслуживания документов, не соответствующих ни одной маске из совокупного набора масок, определенных настройками сервисов.

Для решения проблемы потери файлов при изменении набора масок предложено использовать некоторый процесс, запускаемый сервисом после каждого изменения маски. При этом процесс изменения маски должен происходить эксклюзивно с помощью специального клиентского приложения администратора системы. В ходе данного процесса проверяется измененный набор масок сервиса на наличие дубликатов, а затем проверяются файлы во всем файловом пространстве, доступном сервису, и при необходимости передаются для управления другому сервису или сервису по умолчанию (рис. 4).

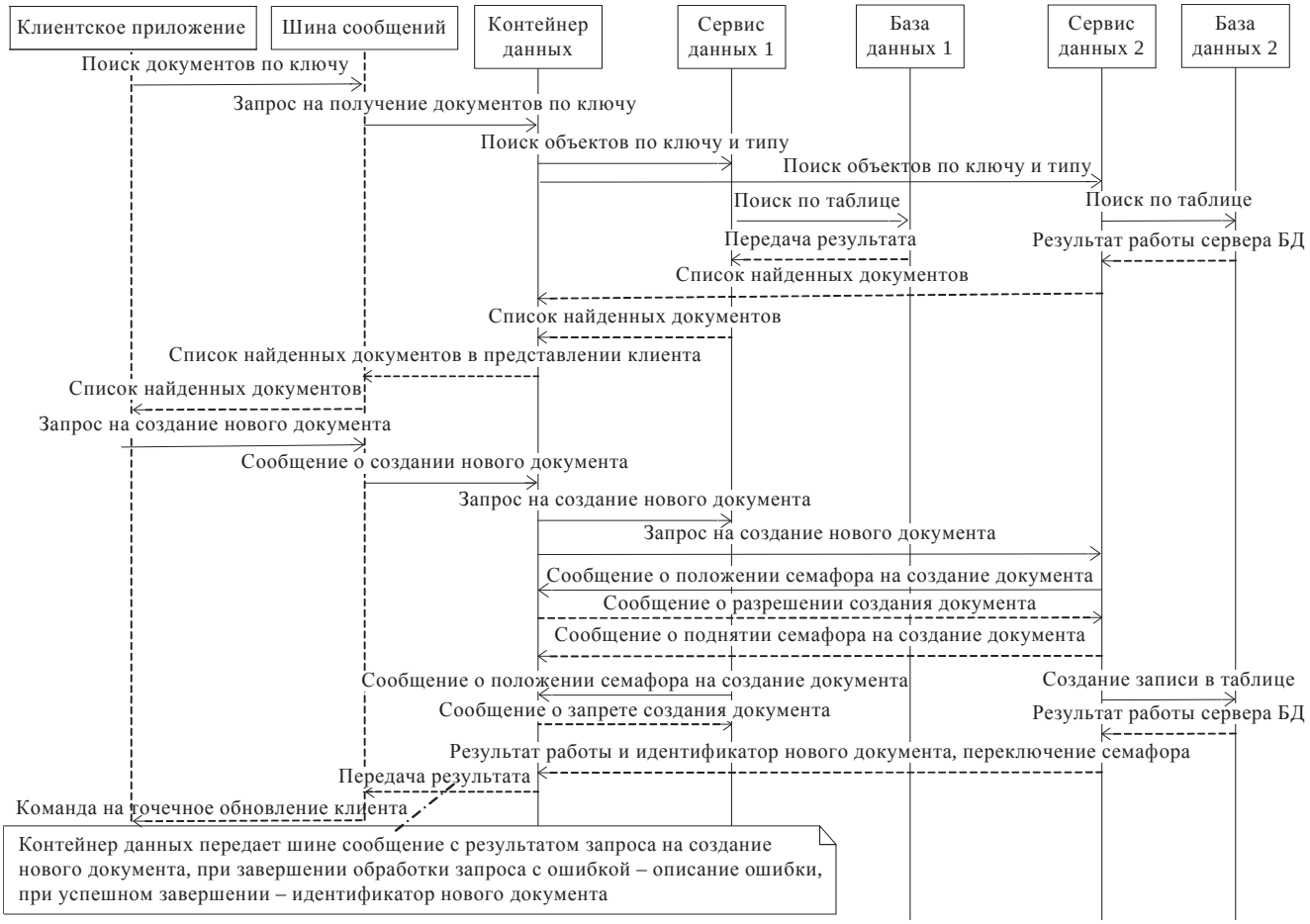

Рис. 5. Поиск документов и создание нового документа в распределенной БД

Созданы специализированные сервисы управления распределенной базой данных. Для каждой базы данных, определенных в системе, создается специализированный сервис управления, который может посылать сообщения обобщенному контейнеру данных системы (рис. 5). В этом случае процесс добавления объекта происходит следующим образом. После получения сообщения-запроса на создание объекта наиболее не нагруженный в данный момент сервис обрабатывает его и отправляет решение о его создании. По окончании процедуры создания отправляется сообщение об успешном (или неуспешном) окончании процесса. Следует отметить, что при использовании такого подхода может возникнуть ситуация, когда сам объект сохраняется на одном сервисе (или в одной базе данных), а связанные с ним части — в таблицах, находящихся на другом сервисе. Эта проблема легко решается путем создания кеша второго уровня непосредственно на клиентской части приложения и проектирования модели системы с учетом того, что она должна иметь максимальное количество слабых связей в объектах.

На примере подсистемы авторизации пользователей и подсистемы печати документов из внешних приложений реализованы механизмы интеграции внешних компонентов на основе шины сообщений.

Заключение. В работе выполнен анализ стандартной информационной системы с клиент-серверной архитектурой, рассмотрены особенности клиент-серверной и сервисно-ориентированной архитектур. На основе наблюдений за поведением систем при нагрузочном

тестировании указаны условия, при которых возникает необходимость перевода системы на сервисно-ориентированную архитектуру: 1) система предназначена для большого числа пользователей; 2) БД системы и файл-репозитория имеют большие размеры (либо существует вероятность их быстрого увеличения); 3) существует необходимость интеграции со специализированным программным обеспечением, разработанным третьими фирмами.

Рассмотрен пример перехода информационной системы на сервисно-ориентированную архитектуру. В качестве примера эволюции информационной системы рассмотрена система АСПИД, предназначенная для ОАО ИСС им. М. Ф. Решетнева (г. Железногорск). Следует отметить, что в настоящее время создан опытный образец этой информационной системы с использованием сервисно-ориентированной архитектуры.

Список литературы

1. Service-oriented architecture (SOA): concepts, technology, and design. San Francisco: Prentice Hall Ptr, 2005.

2. Коржов В. Многоуровневые системы клиент - сервер. М.: Открытые системы, 1997.

3. Платонов Ю. Г. Анализ требований к системе "Электронный документооборот" на предприятиях с высокой степенью ответственности с точки зрения применения современных информационных систем // Пробл. информатики. 2011. № 1. С. 34-50.

4. Бленди Дж. Введение в CVS. [Электрон. ресурс]. http://citforum.ru/programming/digest/cvsintrorus.shtml (обращение 05.05.11).

5. Bloch J. Effective Java. Programming language guide. Boston: Addison-Wesley, 2001.

6. Eckerson W. W. Three tier client/server architecture: achieving scalability, performance, and efficiency in client server applications // Open Inform. Systems. 1995. V. 10, N 1. P. 3–20.

7. Фаулер М. Архитектура корпоративных программных приложений. М.: Вильямс, 2010.

8. Официальный сайт Microsoft http://msdn.microsoft.com/en-us/library/ms834460.aspx (обращение 05.05.11).

Платонов Юрий Георгиевич — асп. Института систем информатики СО РАН;
тел.: 913-950-60-71; e-mail: y.platonov@mail.ru

Дата поступления — 07.04.11 г.