УДК 004.4, 004.9

И. Е. Пелепелин[1], канд. физ.-мат. наук, вед. науч. сотр., e-mail: IPelepelin@blogic20.ru,
В. В. Феклистов[1], стар. науч. сотр., e-mail: vfeklist@mail.ru,
С. В. Карандин[1], науч. сотр., e-mail: skarandin@samelogic.ru,
Д. Р. Башкирцев[2], инженер, e-mail: DBashkirtsev@it.ru,
А. С. Зиннатуллин[3], стажер, e-mail: artem.zinnatullin@gmail.com,
[1]Национальный исследовательский университет "Высшая школа экономики", г. Москва,
[2]Группа компаний "АйТи", г. Уфа,
[3]Группа компаний "АйТи", г. Санкт-Петербург

Интеграция Metasonic Suite и Alfresco при разработке реестра сервисов с использованием принципов S-BPM

На примере субъектно-ориентированного подхода к разработке реестра сервисов и сопутствующего бизнес-процесса рассматриваются вопросы интеграции программных продуктов Metasonic Suite и Alfresco.

Ключевые слова: *интеграция, реестр сервисов, Metasonic Suite, Alfresco, S-BPM, CMIS, SOA*

Введение

Для компаний-разработчиков программного обеспечения всегда было актуальным добиться такого положения вещей, когда однажды разработанный для какого-либо проекта программный код можно было бы использовать повторно в других проектах. Такие повторно используемые программные компоненты традиционно представляются в виде библиотек кода и имеют ряд недостатков, основными из которых являются:

- слабая структурированность библиотек программного кода;
- отсутствие возможности организации перекрестных ссылок в описании элементов библиотеки;
- трудности поиска нужного компонента по его функциональным параметрам и по эффективности использования для выполнения того или иного нового проекта.

Устранить указанные недостатки призван Реестр сервисов (далее Реестр), который реализован на базе системы управления информационными ресурсами предприятия (*Enterprise content management* — ECM). Однако мир несовершенен и, как следствие, избавившись от одних недостатков, мы тут же получаем ряд новых. Поскольку Реестр становится централизованным ресурсом, необходим определенный регламент по добавлению новых компонентов или по изменению существующих. Другими словами, нужен механизм, с одной стороны, препятствующий превращению Реестра в "свалку" а с другой стороны, не слишком сложный, чтобы не отпугнуть потенциальных "инвесторов". Под инвесторами здесь будем понимать тех сотрудников, которые предоставляют свои разработки для внесения их в Реестр.

Исследование, результаты которого представлены в настоящей статье, посвящено повышению эффективности разработок внутри компании, в частности, за счет ориентации на сервис-ориентированную архитектуру (*Service-Oriented Architecture* — SOA) с использованием исследований и результатов консорциума OASIS [1]. При этом проблемы, связанные с созданием стимулов для наполнения Реестра, остаются за рамками данной статьи.

Выбор методов и средств ведения Реестра

В парадигме SOA понятия сервис и компонент не тождественны. Согласно референсной модели SOA [2], сервис — это совокупность сведений о предоставляемой возможности. А программный компонент — это логически обособленная программа или библиотека, которая может быть оформлена как сервис, при условии соответствия компонента спецификациям сервиса. Иными словами, сервис реализуется компонентом. Создание сервиса процесс творческий, требующий участия на разных этапах разработки специалистов различ-

ной квалификации, взаимодействие которых в процессе его создания должно быть строго регламентировано. Вместе с тем сам процесс разработки хотя и поддается регламентации, может быть построен различными способами, в зависимости от квалификации участников, инструментария, которым они владеют, а также от используемых ими технологий. Наибольший эффект достигается в том случае, если процесс создания на каждой стадии настраивается самим исполнителем, при условии выполнения им обязательств по предоставлению результатов своей деятельности другим участникам в стандартизованном виде.

Исходя из представленных выше соображений, в качестве систем управления процессом создания сервиса в наибольшей степени подходят системы, основанные на субъектно-ориентированном управлении. В таких системах исполнитель рассматривается не как ресурс, а как творческая единица, способная к самоорганизации и оптимизации своей деятельности в процессе исполнения заданий. Одной из таких систем является Metasonic Suite, которая позволяет упорядочить процесс взаимодействия участников разработки, оставляя им практически полную свободу действий в рамках решения конкретной задачи на разных этапах проектирования сервисов и разработки реализующих их компонентов.

Кроме стандартизации взаимодействия в процессе разработки компонентов необходимо также определить стандарты описаний сервисов при их проектировании. Такие стандарты должны быть понятны всем участникам процесса, а также потребителями сервиса, которые будут использовать его в своих бизнес-решениях. Для хранения сведений о сервисах и реализующих их компонентах служит Реестр, рассматриваемый в настоящей статье. В хранилище этого Реестра должны содержаться описания бизнес-требований, функциональные спецификации, контракты, содержащие нефункциональные характеристики сервисов, а также описания интерфейсов и реализаций сервисов.

В настоящее время существует три подхода к организации хранения и обработки информации о программных компонентах, использующихся при создании систем в парадигме сервис-ориентированной архитектуры:

- репозитории сервисов;
- реестры сервисов;
- комплексные решения по управлению архитектурой предприятия.

Репозитории сервисов предназначены для хранения информации об объектах SOA-систем, а именно — сервисах, их реализации, контрактах и другой подобной информации. По факту, это база данных или хранилище неструктурированной информации [3]. Как вырожденный случай, это может быть просто ресурс в файловой системе.

Реестры сервисов представляют собой решения, которые предоставляют возможность пользоваться репозиторием сервисов упорядоченным, контролируемым способом. Реестры сервисов различаются по назначению и области применения. Например, для решения задач по управлению инфраструктурой предприятия актуальны функции обнаружения с применением стандартов UDDI/WSDL. Такие репозитории могут применяться для автоматизации приемочного тестирования, мониторинга, инвентаризации, вывода сервисов из эксплуатации. Примеры таких реестров:

- Apache: jUDDI (http://uddi.xml.org/apache-juddi);
- Ruddy (http://www.ruddi.org/);
- WebSphere Service Registry and Repository (http://www-01.ibm.com/software/integration/wsrr/index.html).

В случае если речь идет о более ранних этапах создания программного сервиса, таких как проектирование и разработка, становятся актуальными функции совместной разработки, согласования описания, контрактов и программного кода, поиск сервисов по назначению и другие подобные им. Примеры таких реестров:

- Membrane (http://www.membrane-soa.org/soa-registry/);
- WSO2 Governance Registry (http://wso2.com/products/governance-registry/);
- SOA service manager (http://www.soa.com/products/service_manager/).

К общим решениям по управлению программными комплексами на основе SOA относятся приложения, обладающие целым спектром назначений. Они, как правило, используются в больших организациях, для которых актуальны вопросы управления архитектурными компонентами и активами предприятия в целом. В спектр задач таких решений входит не только архитектура приложений, но и технологическая, бизнес-архитектура. Ниже приведен перечень продуктов и вендоров, согласно аналитическому отчету Kristian Steenstrup из компании Gartner за 2010 г.

- IBM Maximo Asset Management
- IFS
- Invensys Operations Management (Avantis)
- Mainsaver
- Mincom
- Oracle E-Business Suite, Oracle
- SAP
- Ventyx, ABB Company.

Создатели этих продуктов в первую очередь ориентируются на организации, являющиеся потребителями программного обеспечения, а не его разработчиками. По этой причине они неэффективны в решении вопросов, рассматриваемых в настоящей статье.

Хранилище Metasonic Suite не может в достаточной степени решить задачи, стоящие перед Реестром в силу целого ряда ограничений, основным из которых является ненормализованная модель хранения данных. Кроме того, хранилище данных Metasonic Suite требует дополнительных усилий по проектированию интерфейса для потребителей сервисов, что вызывает необходимость выбора специального хранилища для реализации Реестра.

В результате проведенных исследований выбор авторов был остановлен на хранилище ECM-плаформы Alfresco. Основными преимуществами этого хранилища является режим его свободного распространения, а также возможность обращения к нему через стандартный де-факто протокол CMIS (*Content Management Interoperability Services* — сервисы взаимодействия при управлении контентом) [1].

Существует несколько способов, поддерживающих механизмы взаимодействия системы субъектно-ориентированного управления разработкой программного обеспечения и хранилища данных. Такое хранилище предназначено для упорядоченного хранения описаний сервисов и реализующих их программных компонентов в парадигме сервис-ориентированной архитектуры. В том числе существуют способы, эффективно реализующие эти механизмы для систем Metasonic-Alfresco.

Один из упомянутых выше способов заключается в построении модели управления разработкой программного обеспечения и бизнес-объектов, хранящих информацию о создаваемых компонентах на базе Metasonic Suite, и разработке сервисов, обеспечивающих взаимодействие экземпляра процесса разработки с Реестром описаний сервисов и реализующих их компонентов, который хранится в Alfresco.

Другой способ — на базе Metasonic Suite строится только управление процессом разработки. Работа с хранилищем при этом осуществляется в интерфейсе Alfresco с использованием его стандартных интерфейсов. При таком подходе в хранилище будут также накапливаться сведения о процессе разработки.

Третий способ представляет собой комбинацию первых двух, в этом случае управление процессом осуществляется на базе Metasonic Suite. Там же создаются бизнес-объекты, отвечающие за ход исполнения процесса разработки, а сущности, хранящие сведения о сервисах, заполняются в интерфейсе Alfresco.

Основным критерием при выборе эффективного способа рассматриваемого взаимодействия является трудоемкость его реализации. Во втором и третьем из упомянутых выше случаев она оказывается выше, так как требует разработки большего числа сервисов, обеспечивающих взаимодействие.

В первом случае необходимо создание сервисов, поддерживающих механизмы синхронизации значений атрибутов бизнес-объектов и сущностей хранилища данных. К ним относятся сервисы установления связи с хранилищем данных, создания и наполнения сущности в хранилище Alfresco, модификации сущности, установления связи между сущностями.

Во втором и третьем случаях необходимость в сервисах синхронизации данных отпадает, так как заведение и редактирование атрибутов сущностей ведется в стандартном интерфейсе Alfresco. Однако при этом возникает необходимость в создании сервисов, обеспечивающих передачу управления от Alfresco в Metasonic Suite. К ним относятся сервисы запуска процесса, работы с экземплярами процессов, работы с описаниями сущностей, связывания экземпляра сущности с экземпляром процесса (выполняются на стороне Metasonic Suite), работы с заданиями, работы с описаниями сущностей (выполняются на стороне Alfresco).

Выбор способа интеграции

Современные средства моделирования и автоматизации бизнес-процессов в отдельной организации развиваются в направлении большей адаптируемости к постоянным изменениям и усложнениям бизнес-окружения. Такие средства обеспечивают более высокий уровень понимания (восприятия) и доступности описаний используемых моделей для самих участников (субъектов) бизнес-процессов в рассматриваемой организации. В последние годы из общей концепции процессного управления организацией выделилось направление, основанное на субъектно-ориентированном подходе к описанию бизнес-процессов *Subject-oriented Business Process Management* (S-BPM) [4]. В S-BPM основное внимание сосредоточено на действующих лицах бизнес-процессов — субъектах, и все вопросы рассматриваются с позиции субъекта. В 2011 г. S-BPM попал в отчет аналитиков компании Gartner "Цикл зрелости технологий" (в оригинале *Hype Cycle for Business Process Management*) как новая развивающаяся технология [5]. В частности, S-BPM реализован в продукте Metasonic Suite разработки компании Metasonic AG.

В большинстве организаций, особенно в крупных, очень остро стоят вопросы интеграции между различными информационными ресурсами и системами. Как правило, в бизнес-процессы бывают вовлечены информационные ресурсы от разных производителей. Это обстоятельство приводит к необходимости решения задач по организации их взаимодействия как на уровне используемой системной архитектуры, так и на уровне отдельных функций конкретного бизнес-процесса. В данной статье ограничимся только рассмотрением вопроса интеграции системы S-BPM Metasonic Suite с другими системами.

В крупных организациях, особенно в Германии, часто используются продукты немецкой компании SAP AG и вопросы интеграции именно с этой системой на текущий момент являются наиболее проработанными. Так, например, в статье [6] подробно описаны способы интеграции Metasonic Suite с системами SAP ERP и SAP CRM. По поводу интеграции с другими системами, например, класса Enterprise content management (ECM), информации пока практически нет.

Наиболее удобным и гибким средством для интеграции процессов, запущенных в среде Metasonic, с другими системами является механизм, именуемый "Refinements" [7]. В рамках оболочки построения бизнес-процессов Metasonic Build этот механизм позволяет добавлять программный код сторонних разработчиков, определяя тем самым действия, которые будут выполнены в соответствующем состоянии субъекта бизнес-процесса. Таким образом, используя "Refinements", внутреннее поведение объекта бизнес-процесса может быть существенно изменено. Это дает в руки разработчика гибкий механизм, в том числе и для целей интеграции [8].

Задача интеграции информационных систем возникла одновременно с появлением собственно информационных систем и имеет несколько классических способов решения, описанных в литературе и активно применяемых на практике. Интеграция разнородных, особенно гетерогенных, систем может осуществляться на различных уровнях, в частности: на уровне пользовательских интерфейсов, на уровне информационных ресурсов, на уровне корпоративных приложений, на уровне сервис-ориентированной архитектуры [9].

Способы интеграции различаются в зависимости от структуры взаимодействия, метода интеграции и типа обмена данными.

Структурой взаимодействия в случае решения задачи интеграции двух систем (в нашем случае — Metasonic Suite и Alfresco) может являться только способ типа "точка-точка". При этом метод интеграции может быть выбран также только один — метод интеграции по данным. Типов же обмена данными может быть несколько: файловый обмен, совместное использование базы данных, обмен сообщениями, удаленный вызов процедур.

Файловый обмен предполагает наличие процедур экспорта-импорта данных из одного хранилища (базы данных) в другое. Преимуществом такого типа обмена данными является его независимость от процессов, происходящих в системах. Обмен может осуществляться по расписанию, в то время, когда нагрузка на локальную сеть минимальна, или по команде администратора, ответственного за взаимодействие. Существенным недостатком такого типа обмена является неизбежная потеря сведений о промежуточных состояниях процессов и объектов учета, создаваемых, модифицируемых и используемых в процессах различных систем.

Совместное использование базы данных является способом обеспечения взаимодействия в реальном времени, но накладывает серьезные ограничения на возможности изменения связанных систем.

Обмен сообщениями, который может осуществляться синхронно с изменениями, происходящими в ходе реализации процессов, лишен этого недостатка. Однако такой подход требует гораздо более сложной инфраструктуры организации взаимодействия и использования таких средств как общая интеграционная шина предприятия.

Наиболее простым и надежным является синхронный удаленный вызов процедур, реализованных в виде web-сервисов системы, поддерживающей хранилище данных (в нашем случае — Alfresco). Обращение к web-сервисам при этом должно осуществляться в момент завершения, создания и/или модификации экземпляров объектов учета.

Этапы процесса наполнения реестра

Поскольку наполнение Реестра данными предполагает автоматизацию определенных бизнес-процессов, рассмотрим отдельные этапы этого процесса в привязке к процедуре разработки сервиса:

- формирование бизнес-требований;
- экспертиза бизнес-требований;
- разработка спецификаций;
- разработка контракта;
- разработка реализации;
- тестирование.

Так как ни один из стандартов, в том числе государственных, не определяет модель жизненного цикла сервисов и реализующих их компонентов, авторами предлагается модель, аналогичная широко распространенной каскадной или последовательной модели. Ее характерной особенностью является необходимость формирования полной и согласованной документации на каждом этапе проекта.

В процессе проектирования сервисов и разработки реализующих их компонентов могут принимать участие пользователи, обладающие следующими ролями: заказчик, эксперт, аналитик, архитектор, разработчик, тестировщик. В роли заказчика в ходе разработки может выступать как собственно представитель заказчика, так и архитектор, который в случае крупного проекта осуществляет консолидацию составных его частей. При этом началом работы над новыми бизнес-требованиями по реализации того или иного сервиса (компонента) должен являться поиск его описания в Реестре.

В качестве эксперта в процессе разработки, как правило, выступает коллегиальный орган, осуществляющий надзор за разработкой проектов и их компонентов в организации или в группе организаций.

Роль аналитика предполагает создание спецификаций, понятных как разработчикам, так и экспертам, которые утверждают эти спецификации, отвечая за их соответствие требованиям реального заказчика.

Архитектор обеспечивает разработку контракта сервиса или компонента и надзор за реализацией сервиса или компонента в соответствии с контрактом, утвержденным коллегиальным органом. Утвержденный контракт становится доступным другим разработчикам, которые могут на него подписаться.

Разработчик — это программист, реализующий компоненты, соответствующие контракту.

Тестировщик тестирует разработанную реализацию.

Регламент, обеспечивающий взаимодействие участников процесса, накладывает ограничения на правила взаимодействия, а также на внутреннее поведение субъектов, участвующих в процессе. Однако каждый из них может самостоятельно менять модель своего внутреннего поведения, используя привычные ему методы и средства решения задач, что является одним из основных преимуществ предлагаемого в настоящей работе субъектно-ориентированного подхода к управлению процессом разработки.

Модель взаимодействия субъектов (регламент), участвующих в процессе разработки, представлена на рис. 1.

В процессе исполнения регламента сведения, которые попадут затем в хранилище в соответствии с моделью, необходимо вводить и преобразовывать к формату хранилища.

Ввод сведений в основном должен осуществляться вручную, так как эти данные появляются в результате регламентированной деятельности субъектов на различных стадиях этапа разработки сервисов и реализующих их программных компонентов. Для хранения сведений в процессе исполнения регламентов служит следующий набор бизнес-объектов:

- бизнес-требования (*Requirement*);
- контракт (*Contract*);
- реализация (*Realization*);
- спецификация (*Specification*);
- запрос (*Inquiry*);
- уведомление (*Notice*);
- задание (*Task*).

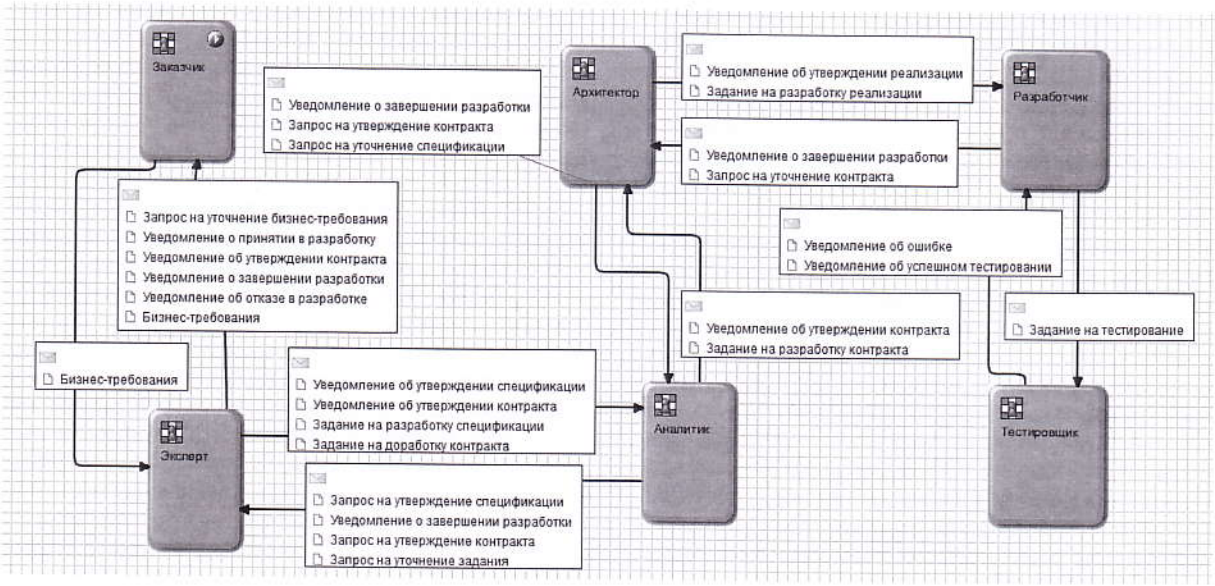

Рис. 1. Модель взаимодействия участников процесса разработки

Каждый из бизнес-объектов содержит набор атрибутов, необходимый как для учета свойств регламентирующих их сервисов и компонентов, так и для учета действий участников процесса, а также состояний бизнес-объекта в ходе исполнения регламента.

Модель данных

За основу модели данных Реестра описаний сервисов и реализующих их компонентов авторами взяты рекомендации консорциума OASIS [10], из которых выделены для включения в состав описаний сервисов следующие основные сущности:

- реестр сервисов (*ServiceRegistry*), содержащий краткое описание реестра сервисов;
- домен сервисов (*ServiceDomain*), содержащий краткое описание домена, в котором расположен сервис;
- описание сервиса (*ServiceDescription*);
- контракт (*Contract*), содержащий нефункциональные требования;
- реализация сервиса (*Realisation*), содержащая описание компонента, реализующего сервис;
- дополнительные наименования сервиса (*Sinonims*);
- предметная область (*ObjectArea*), содержащая краткое описание одной или нескольких предметных областей, например, Управление персоналом, Организационное развитие, Финансовый учет и контроль, Сбыт и др.;
- целевые пользователи (*User*), такая сущность содержит описания ролей на основе ролевой модели системы, для обладателей которых сервис предназначен;
- обеспечиваемые процессы (*Process*), эта сущность содержит описания бизнес-процессов, при автоматизации которых используется сервис, с указанием статуса (используется, планируется к использованию);
- бизнес-сущности (*BusinessEntity*), содержащие описания сущностей предметной области, с которыми связаны возможности, предоставляемые сервисом;
- атрибуты бизнес-сущности (*Attribute*);
- бизнес-операции (*BusinessMethod*), содержащие описания операций в предметной области, которые обеспечиваются возможностями, предоставляемыми сервисом;
- входные данные бизнес-операции (*InputData*);
- выходные данные бизнес-операции (*OutputData*);
- коды возврата бизнес-операции (*Return*);
- типовые решения SOA (*Pattern*), эта сущность содержит перечень паттернов, примененных при проектировании сервиса;
- соответствие стандартам (*Standart*), такая сущность содержит перечень стандартов (технические стандарты и рекомендации (W3C, OASIS, IETF и др.), стандарты и регламенты предприятия, государственные и отраслевые стандарты), которым соответствует сервис;
- ключевые индикаторы сервиса (KPI), эта сущность содержит описания измеримых критериев и их значения, при достижении которых принимается, что сервис эффективен.

Средства и методы взаимодействия

Интеграция информационных ресурсов подсистем, обеспечивающих автоматизацию разработки сервисов и реализующих их компонентов (подсистема управления потоками работ и подсистема хранения информации), основана на сервисах. Для сохранения данных в хранилище Alfresco используются коллекции Java-библиотек, фреймворков и механизмов Apache Chemistry OpenCMIS [11]. Эти инструментальные средства позволяют реализовать разработку программного продукта в соответствии с пакетом стандартов, состоящим из набора web-сервисов для совместного использования информации, хранимой в несвязанных между собой хранилищах контента. Модель данных реализована согласно рекомендациям по созданию пользовательских моделей в хранилище Alfresco [12].

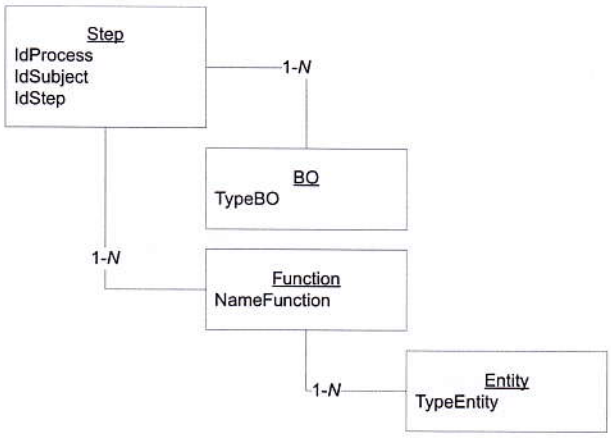

Рис. 2. Модель схемы процесса:
1—*N* — связь "один ко многим"

Рис. 3. Модель схемы преобразования:
N—*N* — связь "многие ко многим"

В целях рассматриваемой в контексте настоящей статьи интеграции создан следующий набор сервисов:

- установления связи с сервером Alfresco и открытия сессии взаимодействия;
- создания экземпляров сущностей и связей;
- сохранения экземпляров сущностей и связей.

Вызовы сервисов осуществляются в среде Metasonic в разделе Refinments для определенных состояний субъектов в процессе выполнения регламента. Чтобы обеспечить многократное использование сервисов, независящее ни от модели бизнес-объекта, ни от модели хранилища данных, ни от модели бизнес-процесса, авторами предложены следующие средства для настройки сервисов (настроечные таблицы):

- схема процесса;
- схема преобразований.

Смысл настроечных таблиц заключается в следующем. Схема процесса обеспечивает настройку сервиса на процесс и содержит сведения о том, какой бизнес-объект на каком шаге бизнес-процесса должен быть сохранен в хранилище. Схема преобразований содержит правила, по которым осуществляется преобразование сведений из экземпляра бизнес-объекта в экземпляр сущности хранилища данных. Установление связей между экземплярами сущностей в хранилище Alfresco осуществляется в соответствии с моделью хранилища.

Модель схемы реализации процесса представлена на рис. 2, где

- Step — описание шага регламента (IdProcess — идентификатор описания процесса, IdSubject — идентификатор описания субъекта, IdStep — идентификатор описания шага процесса);
- BO — описание бизнес-объекта (TypeBO — тип бизнес-объекта);
- Function — описание метода сервиса взаимодействия (NameFunction — имя метода);
- Entity — описание сущности (TypeEntity — тип сущности).

Модель схемы упомянутых выше преобразований представлена на рис. 3, где

- TypeBO — тип бизнес-объекта (NameAttribute — имя атрибута);
- TypeEntity — тип сущности (NameAttribute — наименование атрибута).

Конвертирование объекта Metasonic в сущность Alfresco. Результаты тестирования

Для получения ответа на вопрос о возможности хранения объектов Metasonic в хранилище Alfresco был проведен ряд тестов. По их результатам сделаны выводы о скорости выполнения конвертации, а также о возможных путях оптимизации данного процесса. Работа по проведению серии тестовых испытаний включала в себя развертывание инфраструктуры двух удаленных стендов (узлов) — Alfresco и Metasonic Suite. На сервере Metasonic Suite также была развернута локальная версия Alfresco.

В ходе тестовых экспериментов проводилось конвертирование десяти полей бизнес-объекта Requirement в поля сущности ServiceDescription. В каждом тесте создавалось 10, 100 и 200 однотипных объектов с передачей данных всех десяти полей. Затем рассчитывалось среднее время выполнения конвертации для одного объекта.

Вызов процедур Alfresco проводился с сервера Metasonic. При этом использовалась библиотека Apache Chemistry [11]. Время, которое затрачивалось на реализацию каждого из тестов, измерялось в миллисекундах. Рассматривалось два варианта тестирования. В первом варианте, который назван исходным, для каждого обращения к Alfresco создавался экземпляр соединения. Во втором варианте, который назван оптимизированным, использовалось одно и то же статическое соединение.

Два узла, на которых проводилось тестирование, представляют собой виртуальные серверы, параметры которых представлены в таблице.

На рис. 4 представлены результаты тестирования производительности для исходного и оптимизированного вариантов для случая, когда хранилище Alfresco установлено на том же сервере, что и Metasonic Suite. Для каждого из трех тестов показано по два результата: первый столбец — среднее время выполнения в исходном варианте; второйя столбец — среднее время выполнения в оптимизированном варианте.

Хорошо видно, что оптимизация позволяет ускорить процесс конвертирования объектов почти в 3 раза.

Параметр	Сервер Metasonic Suite	Сервер Alfresco
Операционная система	Windows 7 Домашняя расширенная x64	Windows Server 2008 Standard x32
Оперативная память	8 Гбайт	2 Гбайта
Процессор	AMD A8-3510MX APU 1.8 GHz 4 ядра	Intel Xeon X5660 1.8 GHz 2 процессора
Прикладное ПО	Metasonic Suite, вер. 4.0, Alfresco 4.0.e x64	Alfresco 4.0.e 32-bit

Эффект оптимизации еще более заметен, если протестировать тот же самый процесс, но с использованием хранилища Alfresco, расположенного на удаленном сервере. Во время тестирования сервер Metasonic Suite располагался в г. Уфа, а сервер с хранилищем Alfresco — в г. Москва (рис. 5). В этом случае оптимизация позволила сократить среднее время выполнения операции конвертирования в 6—15 раз. Следует также отметить, что обращение к удаленному серверу Alfresco, по сравнению с локальным увеличивает время в среднем в 4 раза.

Необходимо также отметить тот факт, что если все поля объекта конвертируются с помощью одного обращения к хранилищу Alfresco, то скорость обработки увеличивается еще в 4—5 раз.

Заключение

Предложенный субъектно-ориентированный подход к управлению стадиями этапа разработки в совокупности со способом ведения Реестра сервисов в специальном хранилище данных обладают следующими достоинствами:

- гибкость настройки процесса проектирования сервисов и разработки реализующих их компонентов на каждой стадии этапа разработки;

Рис. 4. Результаты тестов для локального сервера Alfresco

Рис. 5. Результаты тестов для удаленного сервера Alfresco

- простота реализации, основанная на вызовах web-сервисов.

Предложенный подход позволяет создать условия для повторного использования написанного программного кода в процессе создания новых приложений в парадигме SOA.

Существенным преимуществом предлагаемого в настоящей работе подхода является вовлечение в процесс наполнения Реестра представителей руководства организации. Им представляется возможность оперативно контролировать деятельности внутри компании по обновлению старых и внесению новых сервисов в Реестр за счет автоматизации бизнес-процесса и формирования ключевых показателей. Однако следует отметить, что недостатком предложенного способа является отсутствие автоматизированных средств синхронизации изменений моделей данных.

Дальнейшие исследования направлены на устранение указанного недостатка, а именно на создание интегрированных средств, обеспечивающих синхронизацию изменений моделей данных, описывающих бизнес-объекты. Такие модели используются в процессе разработки целевого программного продукта в качестве промежуточного хранилища данных и сущностей хранилища данных Реестра описаний сервисов и компонентов.

Данное исследование проводилось при финансовой поддержке Правительства Российской Федерации (Минобрнауки России) в рамках договора № 13.G25.31.0096 о "Создании высокотехнологичного производства кросс-платформенных систем обработки неструктурированной информации на основе свободного программного обеспечения для повышения эффективности управления инновационной деятельностью предприятия в современной России".

Список литературы

1. **Content** Management Interoperability Services (CMIS). OASIS Standard Specification. Version 1.0. 2010. URL: http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.pdf.

2. **Reference** Model for Service Oriented Architecture 1.0. OASIS Standard, 12 October 2006. URL: http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf.

3. **Stephens R. T.** The Repository vs. The Registry. URL: http://www.information-management.com/news/1025672-1.html?zkPrintable=1&nopagination=1.

4. **Fleischmann A.** What Is S-BPM? // Communications in Computer and Information Science. 2010. Vol. 85. P. 85—106.

5. **Hype** Cycle for Business Process Management. Gartner. 25 July 2011. URL: http://www.adeptia.com/products/Hype_cycle_BPM_ 2011.pdf.

6. **Hufgard A., Gerhardt E.** Consolidating Business Processes as Exemplified in SAP ERP Systems // Communications in Computer and Information Science. 2011. Vol. 213. P. 155—171.

7. **Martin W.** Active Compliance Management with Subject-oriented Business Process Management. URL: http://www.metasonic.de/sites/default/files/editor/PDF/Metasonic_whitepaper_e_finsec.pdf.

8. **Busse C., Stein G., Fackelmeier B.** Metasonic Build Modeler Manual. Version: T9.0. 08/01/2011. URL: http://www.metasonic.de/en/downloads.

9. **Кусов А. А.** Проблемы интеграции корпоративных информационных систем // Управление экономическими системами: электронный научный журнал. 2011. № 4 (28). URL: http://uecs.ru/uecs-28-282011?start=20.

10. **Web** Services Base Notification 1.3 (WS-BaseNotification). OASIS Standard. 1 October 2006. URL: http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf.

11. **Apache** Chemistry. Apache Software Foundation. URL: http://incubator.apache.org/projects/chemistry.html.

12. **Cei U., Lucidi P.** Alfresco 3 Web Services. Birmingham: Packt Publishing Limited, 2010.