СУБЪЕКТНО-ОРИЕНТИРОВАННОЕ УПРАВЛЕНИЕ РАЗРАБОТКОЙ КОМПОНЕНТОВ СОА

И.Е. Пелепелин,
кандидат физико-математических наук, ведущий научный сотрудник Национального исследовательского университета «Высшая школа экономики», ведущий руководитель проектов ООО «Логика бизнеса»
В.В. Феклистов,
старший научный сотрудник Национального исследовательского университета «Высшая школа экономики»
Д.Е. Кожевников,
ведущий научный сотрудник Национального исследовательского университета «Высшая школа экономики»

E-mail: i61pelepelin@gmail.com , vfeklist@mail.ru, dkozhevnikov@hse.ru
Адрес: г. Москва, ул. Ленинская слобода, д. 19, стр. 6

Повышение эффективности создания информационных систем в рамках сервис-ориентированной архитектуры (СОА) является актуальной проблемой. Предложены критерии определения эффективности процессов проектирования и создания компонентов. Определены основные этапы жизненного цикла компонентов. Определено место и преимущества субъектно-ориентированного подхода при моделировании и управлении процессами проектирования и создания компонентов.

Ключевые слова: сервис-ориентированная архитектура, СОА, субъектно-ориентированный подход, S-BPM, разработка компонентов, этапы жизненного цикла.

1. Введение

Среди архитектур информационных систем в последние 12 лет выделяется сервис-ориентированная архитектура – подход к созданию приложений, в основе которого лежит выделение в деятельности относительно независимых автоматизируемых транзакций, именуемых сервисами, разработка программных компонентов, автоматизирующих эти транзакции, и объединение их в приложение на базе единой инфраструктуры для исполнения и управления.

Несмотря на большое количество разработанного инструментария и огромные преимущества в гибкости, масштабируемости и эффективности, которые получают предприятия при переходе на сервис-ориентированную архитектуру прикладных систем,

этот подход до сих пор не получил ожидаемого распространения. Причинами такой ситуации, на наш взгляд, являются трудности при управлении большим массивом сервисов, как на этапе эксплуатации, так и на этапе проектирования и разработки. Косвенным подтверждением этого факта служит то, что большинство вендоров сложных корпоративных систем классов ERP, CRM, EDM, ECM и др. перешли на использование СОА при разработке своих продуктов, тогда как среди независимых разработчиков данный подход до сих пор считается перспективным.

Исходя из имеющегося у авторов опыта, трудности управления проектированием и разработкой носят преимущественно архитектурный характер. Это означает, что основные проблемы возникают на стыке зон ответственности участников процесса. Конкретизируя, можно отметить следующие особенности СОА-разработки, приводящие к затруднению при управлении и координации усилий разработчиков:

✦ сложности при постановке задачи, известные в системной анализе как проблема структурно-функционального перехода;

✦ сложности при координации интерфейсов;

✦ необходимость поддержания стандартов сервисов;

✦ сложности при управлении проектом.

В данной работе авторы обобщают свой опыт решения обозначенных выше проблем как при помощи традиционных средств, так и с использованием нового субъектно-ориентированного подхода к управлению. Фокус статьи направлен на ряд актуальных технических и технологических аспектов использования соответствующего инструментария.

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».

2. Характерные особенности компонентов СОА

Поскольку речь идет о создании информационных систем в сервис-ориентированной архитектуре, компонентам, реализующим сервисы таких систем, присущи следующие основные характерные особенности [1]:

✧ слабая связанность;

✧ максимальный уровень абстрактности (абстрактность – мера невещественности);

✧ повторное использование;

✧ высокая автономность;

✧ максимальный отказ от запоминания состояний (персистентности) сервиса;

✧ эффективное использование сервиса;

✧ сервис как часть большей системы;

✧ единая форма описания сервисов.

Исходя из перечисленных характерных особенностей, можно сформулировать основное требование к средствам и методам создания сервисов и компонентов их реализующих – автономность на этапе разработки и эксплуатации. Поскольку реализация сервиса скрыта от его потребителя, никаких ограничений, кроме того, что интерфейсы должны быть стандартизированы, на инструмент создания реализации не накладывается. Так же нет ограничений и на технологию их создания.

Управление разработкой при таком подходе становится весьма нетривиальной задачей. Факторами, определяющими необходимость особого подхода к проектированию и управлению разработкой в приложений в сервис-ориентированной архитектуре являются:

● сложность планирования в связи с творческим характером работы и турбулентной средой;

● уникальность задач, невысокий процент повторяемости в рамках проекта;

● мотивация к получению результата для разработчика ниже профессиональной мотивации к созданию совершенного кода;

● ограниченные возможности текущего контроля.

Таким образом, средства и методы разработки и управления должны обеспечивать возможность координированного формирования стандартизованного интерфейса, обеспечивая при этом независимость команд на этапе реализации, а также обеспечивать доступность результатов разработки широкому, практически неограниченному, кругу пользователей сервисов после ее завершения.

3. Обзор средств хранения описаний сервисов и компонентов их реализующих

Наибольший эффект в управлении СОА-разработкой достигается в том случае, если процесс разработки на каждом этапе жизненного цикла настраивается самим исполнителем при условии выполнения им обязательств по предоставлению результатов своей деятельности другим участникам в стандартизованном виде. Такой подход подразумевает создание процесса разработки, который предполагал бы четкую оркестровку и координацию на этапах согласования внешних интерфейсов сервисов и их отладки, и мягкую хореографию на этапах собственно разработки и внутреннего тестирования.

Кроме стандартизации взаимодействия, в процессе разработки компонентов СОА необходимо также определить стандарты описаний компонентов, которые должны быть понятны всем участникам процесса, а также потребителям сервиса, которые будут использовать его в своих бизнес-решениях.

Проведенный авторами летом 2012 г. опрос команд разработчиков показал, что чуть более 35% проектов разработки приложений выполняются в Сервис-ориентированной архитектуре. При этом чуть более 30% команд никогда не использует СОА.

В опросе приняли участие представители 11 независимых команд, работающих в сервис-ориентированной парадигме. В силу малого числа опрошенных, полученные результаты не следует считать репрезентативными, однако их достаточно, чтобы проиллюстрировать выявленные авторами на собственном опыте проблемы.

Только четыре из опрошенных команд используют специальные средства для согласования интерфейсов разрабатываемых сервисов, причем одна из этих команд не до конца удовлетворена используемым инструментарием. Число команд, использующих инструменты управления стандартами СОА еще меньше. Из 3-х команд, сообщивших об использовании такого инструментария, только одна довольна его качеством. Вместе с тем, практически все команды (91%) используют средства управления взаимодействием в проектах. При этом средний уровень удовлетворенности используемым инструментарием составляет 53%.

Среди используемого командами инструментария преобладают продукты IBM/Rational, часто в качестве таких инструментов используются возможности платформы (ERP, CRM или ECM системы). Отдельно отмечено использование Alfresco в качестве инструмента поддержки СОА – разработки, в силу особенностей данного продукта, объединяющей в себе набор основных блоков ECM системы и весьма развитый инструментарий поддержки СОА-разработки. Им пользуется 36% опрошенных нами команд, как показано на *рис. 1.* Характерно, что 82% команд не удовлетворены возможностями одного продукта, и используют дополнительный инструментарий для решения частных задач. Спектр такого инструментария крайне разнообразен, на диаграмме мы сгруппировали эти продукты в категорию «прочие».

Результаты опроса наглядно демонстрируют, что инструментарий для поддержки разработки СОА-приложений на сегодня не полностью удовлетворяет разработчиков, не дает им возможности использовать преимущества СОА в полном объеме. В особенности это касается архитектурного аспекта разработки, который имеет критически важное значение для создания по-настоящему ценных наборов сервисов, ориентированных на потребности пользователя, а не на видение разработчика.

Использование различных инструментов, в конечном счете, определяется не только опытом и желанием команды, но также ИТ-ландшафтом и текущей инфраструктурой сервисов, используемых на предприятии. Достаточно полный обзор по данной теме дается в [2].

Кроме перечисленных выше инструментов, авторы использовали подход к управлению взаимодействием, основанный на субъектно-ориентированном управлении, в котором ис-

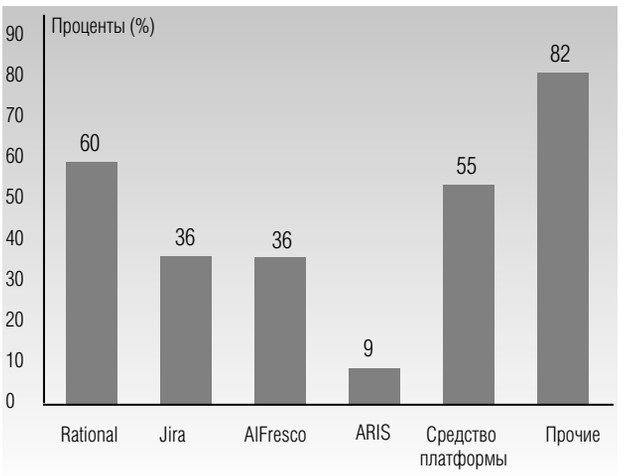

Рис. 1. Сравнение популярности инструментов поддержки СОА

полнитель рассматривается не как ресурс, а как способная самоорганизовываться и оптимизировать свою деятельность в процессе исполнения заданий творческая единица. Единственной на сегодняшний день промышленной системой такого типа является Metasonic Suite. Использование этого инструментария для проектирования и разработки кросс-корпоративных сервис-ориентированных прикладных систем предложено в [3]. Рассмотрим, как использование данной системы способно помочь упорядочить процесс взаимодействия участников разработки, оставляя им практически полную свободу действий в рамках решения конкретной задачи на разных этапах жизненного цикла компонентов СОА.

Этап 1. Разработка архитектуры и выделение общесистемных сервисов. Действующие лица – архитектор, ключевые пользователи. Инструмент моделирования логики системы – S-BPM. Архитектор и ключевые пользователи проектируют при помощи S-BPM инструментария поведение системы и взаимодействие сервисов. В ходе проектирования появляется список сервисов (субъектов), в ходе общего обсуждения их взаимодействие с другими сервисами описывается в виде сообщений.

Этап 2. Проектирование сервисов. Действующие лица – архитектор, разработчики сервисов, ключевые пользователи. Методология S-BPM позволяет ключевым пользователям описать свою деятельность наиболее простым способом – при помощи пошагового индивидуального описания своей реакции на входящие события (сообщения). Полученная модель, включающая как высокоуровневое описание логики сервисного взаимодействия (process overview), так и описание взаимодействия сервисов (process view) и внутренней логики сервиса (subject view), с учетом возможности описания структур и доступности данных в виде бизнес-объектов, является практически полноценной спецификацией сервисов. Кроме того, использование инструментария Metasonic Proof, представляющего собой среду имитационного пошагового моделирования процесса, позволяет на ранних этапах оценить разработанную архитектуру и существенно снизить вероятность ошибки. Полученные модели можно использовать также как инфраструктуру сервисного домена с реализованной в ней логикой сообщений на этапе раннего тестирования. При этом в качестве реестра сервисов может быть использовано любое промышленное решение, позволяющее осуществить интеграцию с S-BPM моделлером[1].

Этап 3. Разработка сервисов. На данном этапе команды разработчиков получают максимальную независимость. Задача управления – обеспечить соблюдения стандартов разработки и контроль реализации функционала, описанного в спецификациях сервисов. S-BPM совместно с реестром могут использоваться для решения обеих задач. Во-первых, для связывания модели потока работ, реализованного в Metasonic S-BPM Suite, и сервисов, описанных в Реестре, используется единый механизм Refinements, накладывающий ограничения на сервисы с точки зрения стандартов. Кроме того, интеграция реестра и моделлера позволяет обеспечить целостность и непротиворечивость описаний сервисов. Во вторых, приведенная ниже модель, описывающая взаимодействие участников разработки в S-BPM, привязанная к той же структуре субъектов, которая сформирована для архитектуры сервисов, позволит управлять процессом разработки в том же инструментарии, который использовался для проектирования системы.

При этом различные команды разработчиков могут работать над различными реализациями компонентов, реализующих сервисы. Между ними может не быть никакого взаимодействия, а их внутреннее поведение отличаться от команды к команде, что связано с различными методологиями разработки для различных сред программирования. Однако их взаимодействие с другими участниками должно быть полностью идентично. Для этой цели может служить практика использования мультисубъектов для синхронизации деятельности параллельно работающих субъектов, описанная в [4]. Шаблоны такого типа взаимодействия приводятся в [5].

Этап 4. Тестирование и запуск. S-BPM может выступать в роли единой платформы тестирования и временного промышленного решения (до запуска инфраструктуры оркестровки и позднего связывания). При этом описания сервисов могут быть занесены в свойства функций.

[1] Вопросы интеграции подробно описаны в нашей статье «Интеграция Metasonic Suite и Alfresco при разработке реестра программных и сервисных компонентов с использованием принципов S-BPM», готовящейся к печати в журнале «Программная Инженерия».

Этап 5. Эксплуатация и развитие системы. На этом этапе S-BPM может быть использован для реализации регламента работы службы технической поддержки. Разработка такого регламента является развитием настоящих исследований.

4. Построение и модификация модели процесса разработки

При создании средств автоматизации процесса разработки информационных сервисов и компонентов их реализующих авторами была создана модель-процесса, включающая следующие основные стадии:

- ✦ формирование бизнес-требований;
- ✦ экспертиза бизнес-требований;

разработка функциональных спецификаций;

- ✦ разработка контракта, содержащего нефункциональные характеристики сервисов;
- ✦ разработка реализации;
- ✦ тестирование.

В этом процессе принимают участие пользователи, обладающие следующими ролями: заказчик; эксперт; аналитик; архитектор; разработчик; тестировщик.

Регламент, обеспечивающий взаимодействие участников процесса, накладывает ограничения на правила взаимодействия, а также на 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 метода управления процессом разработки.

Модель взаимодействия субъектов (регламент), участвующих в процессе разработки, представлена на *рис. 2.*

Для иллюстрации гибкости данной модели приведем следующий пример. В процессе совершенствования модели процесса авторам понадобилось ввести стадию разработки документации, которая должна исполняться ролью «технический писатель».

Изменение действующей модели процесса потребовала реализации следующих действий:

- ✧ создать субъект;
- ✧ создать сообщения, которыми он обменивается с другими субъектами;
- ✧ описать его внутреннее поведение;
- ✧ создать бизнес-объекты, которыми субъект будет манипулировать;
- ✧ создать средства автоматизации его поведения.

Тип сообщений, которыми вновь созданный

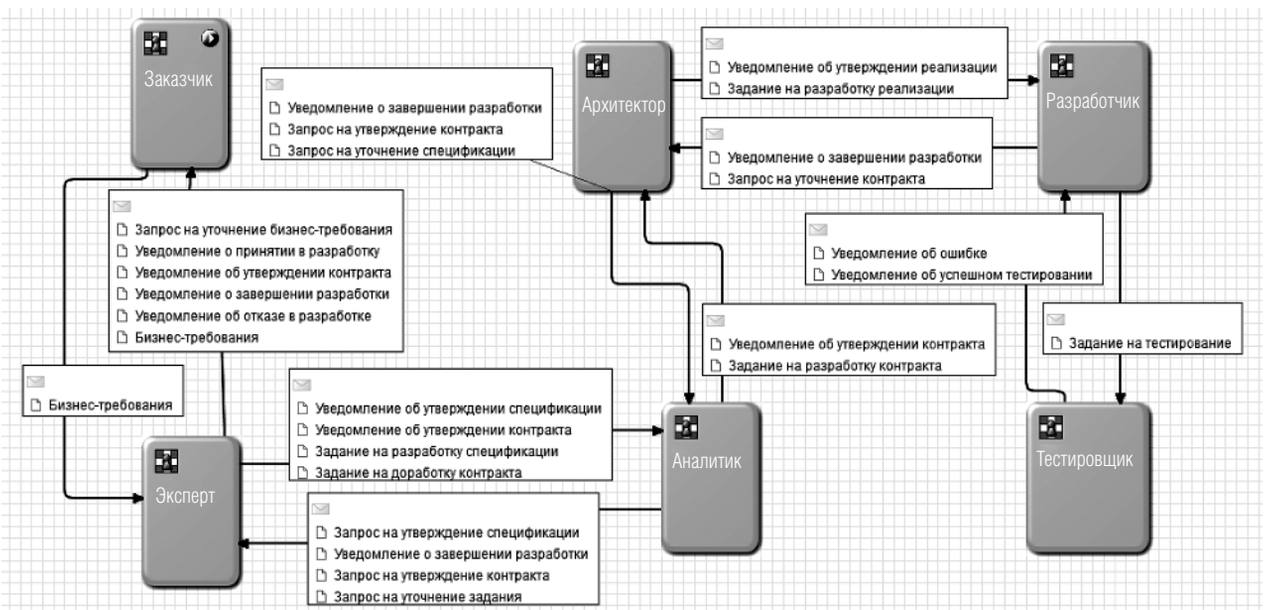

Рис. 2. Модель взаимодействия участников процесса разработки

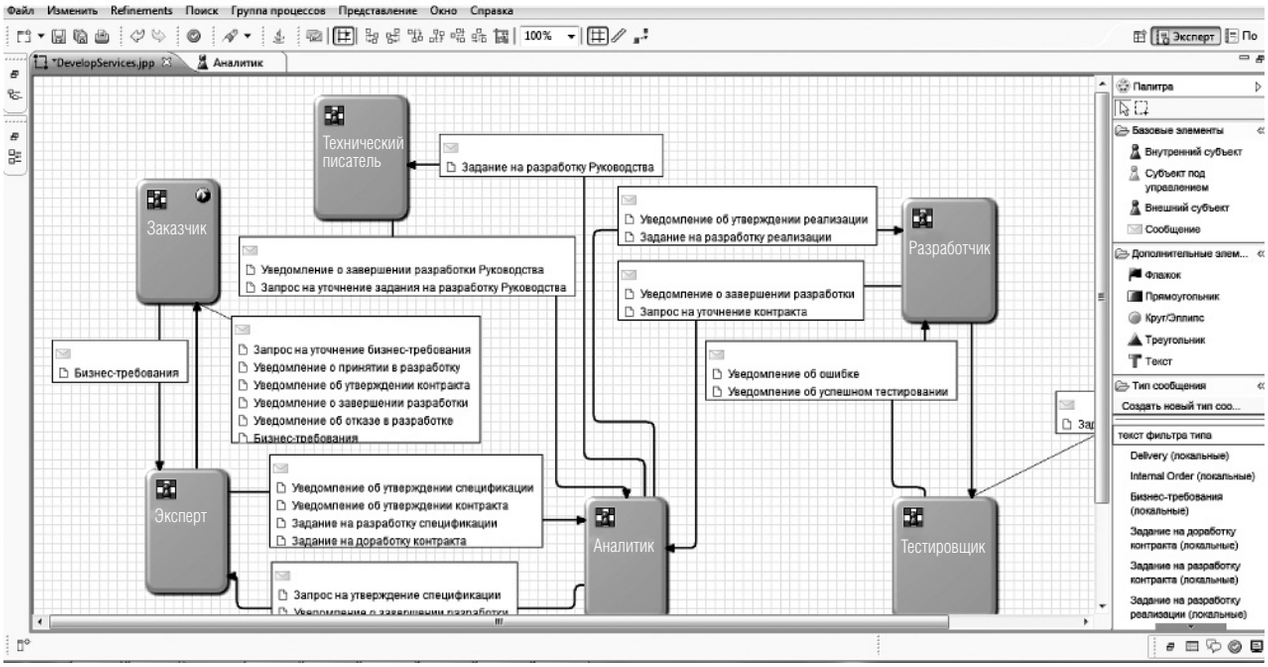

Рис. 3. Фрагмент модифицированной S-BPM модели с интерфейсом Metasonic Suite

субъект обменивается с другими субъектами – участниками процесса, зависит от действий (процедур), которые выполняет субъект. Для технического писателя это процедуры разработки документации (технической, пользовательской и т.п.), в частности «Руководства программиста», которое будет определять правила конфигурирования параметров сервиса. С этой целью «Технический писатель» должен получать от «Аналитика» задание на разработку, отправлять запрос на уточнение задания и отчитываться о выполненной работе.

Для формирования сообщения с запросом на уточнение задания необходимо проделать полностью аналогичные действия. Для формирования сообщений «Аналитика» «Техническому писателю» действия также аналогичны. Только в качестве исходного субъекта должен быть выбран «Аналитик». В результате получается схема, изображенная на *рис. 3.*

Все перечисленные действия требуют от пользователя, которому поручено модифицировать модель процесса, только знания предметной области и инструментов, которые предоставляет Metasonic Suite для моделирования процессов.

И только создание средств автоматизации внутреннего поведения субъекта требуют навыков программирования на языке Java. На начальном этапе моделирования без осуществления этого действия можно, тем не менее, получить действующую модель процесса, которая сразу же пригодна к использованию. Для этого необходимо только добавить внутреннее поведение субъекта, а также создать необходимые бизнес-объекты.

5. Заключение

Использование субъектно-ориентированного подхода дает возможность не только управлять процессом разработки программного обеспечения в сервис-ориентированной архитектуре, но и существенно облегчить процесс проектирования создаваемой системы, хранить спецификацию сервисов и служить инструментом совместного тестирования разрабатываемых сервисов. Преимуществами использования данного инструментария являются:

1. Изначально присущая системе слабая связанность – сервисы могут вызываться из разных систем при соблюдении стандартов, функции хранения контекста берет на себя workflow.

2. Два класса сервисов – автоматический (автоматические субъекты) и операционный/ручной/интерфейсный явно выделены в нотации. Логика сложных сервисов может быть смоделирована в виде диаграммы поведения субъекта.

3. Отсутствие персистентности сервисов зало-

жено в идеологию подхода (в отличие, например, от BPMN). Модели поведения субъекта строятся по свойственной сервисам конфигурации «ромашки» (получение сообщения – обработка по одному из сценариев – отправка результата – ожидание).

4. Использование подхода «абстрактного субъекта с ограничениями» соответствует логике встраивания ограничений, часто используемой архитекторами сервисов при согласовании интерфейсов.

Совместно с реестром сервисов, например на платформе Alfresco, продукт S-BPM Metasonic Suite позволяет реализовать полноценную среду разработки и управления созданием различных систем. Дальнейшая проработка модели взаимодействия участников проекта и использование методики «абстрактного субъекта» может дать дополнительный выигрыш во времени и стоимости разработки и, в конечном итоге, удовлетворить спрос со стороны разработчиков на удобный комплексный инструмент управления и поддержки разработки сервис-ориентированных приложений. ■

Литература

1. A Community Site for SOA Design Patterns. [Электронный ресурс]. URL: http://www.soapatterns.org/ (Дата обращения: 15.08.2012).
2. Marks E.A. Service-Oriented Architecture (SOA) Governance for the Services Driven Enterprise. – Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.
3. Borgert S., Steinmetz J., Muhlhauser M. ePASS-IoS 1.1: Enabling Inter-enterprise Business Process Modeling by S-BPM and the Internet of Services Concept // S-BPM ONE – Learning by Doing, Doing by Learning, third International Conference, S-BPM ONE 2011 Selected Papers. – Springer, 2011. – P. 134-162.
4. Kesh P. Business Objects as a mediator between Process and Data // Communications in Computer and Information Science, 1, Volume 138, 2011, Subject-Oriented Business Process Management, Part II. – P. 180-195. Rodenhagen J., Strecker F. Using Multi-subjects for Process Synchronization on Different Abstraction Levels // Communications in Computer and Information Science, 1, Volume 138, 2011, Subject-Oriented Business Process Management, Part II. – P. 134-162.