УДК 004.4 (06)

АРХИТЕКТУРА ПЛАТФОРМЫ ДЛЯ РАЗРАБОТКИ БИЗНЕС-ПРИЛОЖЕНИЙ

Шмидт И.А.

ГОУ ВПО «Пермский национальный исследовательский политехнический университет» г.Пермь, Россия (614990, Комсомольский пр., 29), shmidt.i@pochta.ru

Проведено исследование методов построения веб-ориентированных бизнес-приложений и моделей жизненного цикла процесса разработки бизнес-приложения. Предложена модифицированная спиральная модель процесса разработки приложений. В соответствии с предложенной моделью первая итерация выполняется силами предметных специалистов. Результатом первой итерации является прототип бизнес-приложения. На последующих итерациях жизненного цикла разработка выполняется профессиональной проектной командой. Предлагается метод разработки бизнес-приложений основанный на построении метамоделей с возможностью модификации исходного кода. Для генерации приложений используется разработанная программная платформа. Рассматривается архитектура пользовательского веб-приложения и программной платформы для его генерации.

Ключевые слова: бизнес-приложение, метамоделирование, архитектура приложения, пользовательский интерфейс, программная платформа

THE ARCHITECTURE OF A DEVELOPMENT PLATFORM FOR BUSINESS APPLICATIONS

Shmidt I.A.

«Perm national research polytechnic university» Perm, Russia, shmidt.i@pochta.ru

Conducted a study of the methods of constructing a web-based business applications and the models of the life cycle of the process of developing business applications. Proposed modified spiral model of the application development process. In accordance with the suggested model of the first iteration is performed by subject specialists. The result of the first iteration is the prototype of business applications. On subsequent iterations of the life cycle development is professional design team. A method is proposed the development of business applications based on the construction of metamodels with the possibility of modifying the source. For the generation of applications, use the developed software platform. The article discusses the architecture of custom web application and software platform for its generation.

Keywords: business application matamodelirovanie, application architecture, user interface, a software platform

Потребности пользователей в разнообразных бизнес-приложениях, в основном ориентированных на малый бизнес, породил класс систем (программных платформ) ориентированных на разработку таких приложений непосредственно самими пользователями.

Предполагается, что разработку приложения с применением такой платформы должен осуществлять грамотный пользователь, но не обязательно программист или администратор баз данных (например, эксперт в предметной области). Метод разработки приложений ориентирован, прежде всего, на сферу малого бизнеса, с возможностью масштабирования для среднего и крупного бизнеса. В дальнейшем ограничим задачу разработки бизнес-приложений веб-приложениями учётного характера, ориентированными на автоматизацию бизнес-процессов.

Метод разработки приложений тесно связан с моделью жизненного цикла программного обеспечения, которая и обуславливает архитектуру платформы разработки и применяемый метод построения бизнес-приложений.

Процесс разработки бизнес-приложения

Среди многочисленных моделей жизненного цикла программного обеспечения, описанная Барри Боэмом, спиральная модель процесса разработки [6] наиболее адекватно описывает реальный процесс создания бизнес-приложений разрабатываемых на заказ (или самостоятельно). Одним из существенных рисков таких проектов является изначально ошибочная формулировка целей создания приложения, которая возникает из-за несоответствия представления предметных специалистов и команды разработчиков. Выходом может являться процесс, когда прототип бизнес-приложения создается самими предметными специалистами (не являющимися программистами или аналитиками), а команда проекта подключается к разработке на следующем витке спирали жизненного цикла приложения.

Таким образом, можно сформулировать идею модифицированной спиральной модели процесса разработки программного обеспечения (См. рисунок 1).

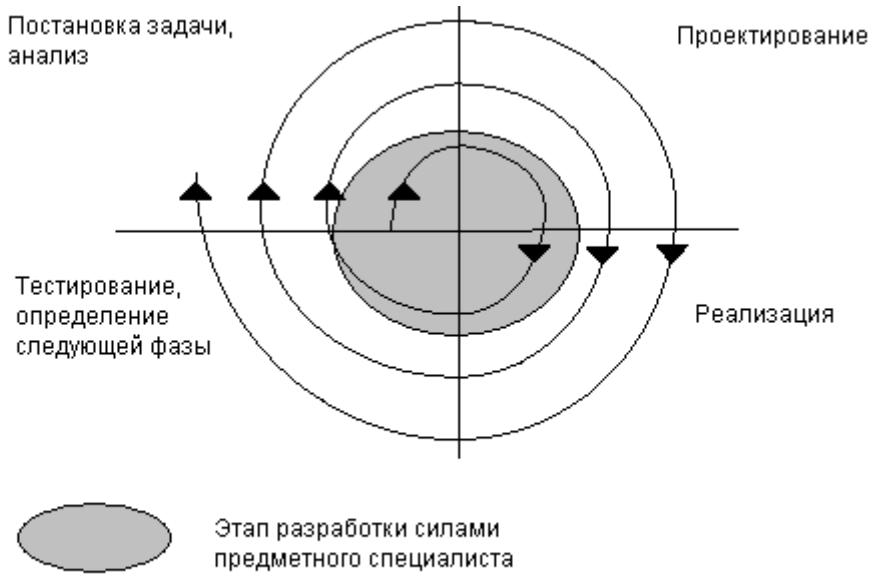

Рисунок 1. Модифицированная спиральная модель процесса разработки бизнес-приложений

Процесс разработки бизнес-приложения включает несколько итераций. Каждая итерация, соответствующая разработке версии программного обеспечения, представляет собой структурированный процесс создания приложения, состоящий из следующих этапов:

- Анализ,
- Проектирование,

- Программирование,
- Тестирование

В традиционной спиральной модели предполагается, что весь процесс проводиться одной проектной командой. В модифицированной модели первая итерация выполняется силами предметных специалистов. Результатом первой итерации является прототип бизнес-приложения, который в максимально наглядном виде дает представление о функционале бизнес-приложения в конечном варианте. Наличие работоспособного прототипа позволяет команде проекта избежать ошибок начального этапа (ошибочная формулировка целей или, несоответствующие предметной области, определение функциональности приложения).

Таким образом, отталкиваясь от модифицированной спиральной модели процесса разработки бизнес-приложений можно позиционировать платформу разработки как средство, позволяющее как осуществить прототипирование для непрофессиональных пользователей, так и доведение проекта до уровня коммерческого бизнес-приложения (может быть тиражируемого) силами проектной команды.

В основе метода разработки веб-ориентированных бизнес-приложений (генерации исходного кода) для проектируемой платформы предлагается метод разработки на основе метамоделей [4]. Существенным отличием, предлагаемого способа разработки является возможность модификации исходного кода созданных приложений. Т.е. практически все аналоги предполагают сокрытие сгенерированного кода от конечного пользователя. Модификация кода позволит при необходимости расширить функциональность бизнес-приложения, произвести его «тонкую» настройку.

Принятые решения предполагают также использование технологий, которые обеспечивают совместимость с облачными технологиями, т.е. предоставляют возможность эксплуатировать разработанные бизнес-приложения в «облачной» среде. Это достигается встроенной возможностью лёгкой доработки генераторов. Кроме этого предусмотрено использование технологии ORM [7] для доступа к данным, что позволяет прозрачно обращаться к распределенным системам хранения.

Рассмотрим теперь архитектуру пользовательского приложения. Представленная архитектура легла в основу платформы FlexBerry (http://www.ics.perm.ru/solutions/flexsol), которая предоставляет из себя сервис, обеспечивающий предоставление инструментального средства для создания SaaS-решений [2]. Таким образом, еще одним отличием предлагаемого подхода от конкурентов является встраивание в технологию механизмов позволяющих разработчику конечного прикладного приложения монетизировать свою разработку. Разработчик может получать прибыль, опубликовав приложение на платформе Flexberry, тарифицировав обращение к нему, и распространяя приложение через магазин приложений.

Архитектура пользовательского приложения

Пользовательское бизнес-приложение является Веб-приложением, т.е. клиент-серверным приложением, в котором клиентом выступает браузер (визуализация данных), а сервером – веб-сервер (бизнес-логика, доступ к данным). При этом клиенты не зависят от конкретной операционной системы пользователя. Архитектура бизнес-приложения представлена на рисунке 2.

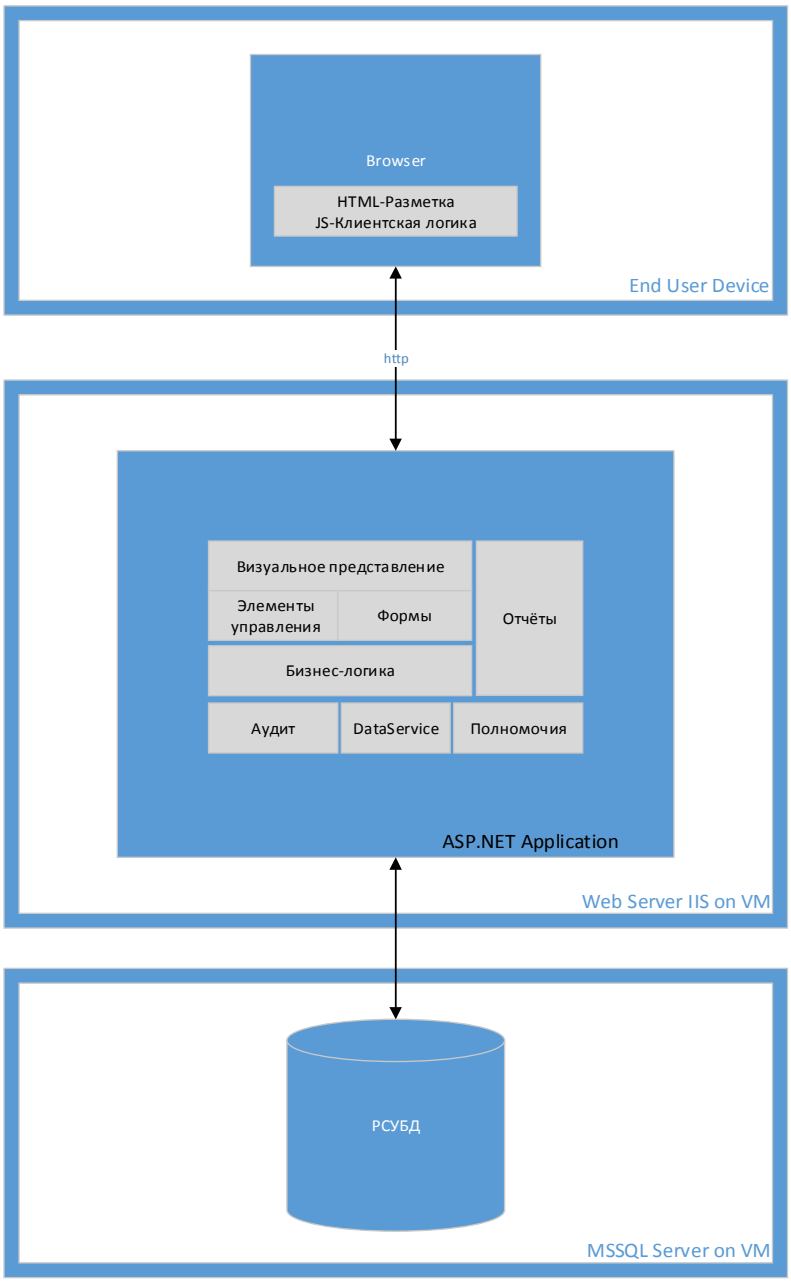

Рисунок 2. Архитектура бизнес-приложения

В архитектуру платформы изначально заложена сервисная модель [1,3,5]. При разработке приложения сервисы могут быть подключены опционально. В приложении могут быть использованы следующие сервисы:

- Сервис безопасности и полномочий;

- Сервис отчетности;
- Сервис работы с документами.

Полномочия, аудит и отчёты могут быть вынесены на отдельные виртуальные машины. Также само веб-приложение может быть организовано в виде веб-фермы (webfarm).

Слой логики приложения базируется на технологиях Microsoft (.NET Framework, ASP.NET). Слой данных основан на СУБД Microsoft SQL Server 2000 и выше.

Слой доступа к данным (DataService)

Через этот компонент осуществляется вся работа с СУБД. Компонент представляет из себя ORM (Object Relational Mapping) с различными дополнительными возможностями.

Этот компонент обеспечивает функции:

- Ввода и сохранение данных в хранилище
- Поиск данных

Полномочия

Полномочия на основе ролей являются функциональной подсистемой ASP.NET-приложения. DataService выдаёт пользователям данные только с учётом разрешений, заданных в полномочиях. Подсистема включает в себя слой хранения информации о полномочиях, runtime-компоненты, компоненты настройки полномочий. Настройка полномочий доступна из ASP.NET-приложения при наличии соответствующих прав у пользователя.

Методика разработки приложения предполагает разграничение прав доступа к данным (и функциям веб-приложения) на основе имеющихся полномочий.

Бизнес-логика

Бизнес-операции над данными выполняются компонентами бизнес-логики. Данный слой приложения представлен скомпилированным кодом. Есть возможность обработать стандартные бизнес-операции, такие как добавление, изменение и удаление данных. Также существует возможность добавить специфичную логику, определяемую предметной областью.

Специфика реализации выбранного метода построения пользовательского интерфейса

Для реализации был выбран метод основанный на объектно-ориентированном подходе. Если ограничится стандартными функциями, полноценный пользовательский интерфейс может быть сгенерирован автоматически, непосредственно из метаописания предметной области. Для учета специфики функционирования конкретной системы потребуется или дополнительное описание интерфейса, включаемое в метаданные, или модификация сгенерированного автоматически исходного кода приложений.

В качестве основного способа отображения данных объекта используется таблица для работы с объектами. Методы объекта могут быть ассоциированы с кнопками, которые располагаются на этой же форме. Таблица имеет богатый набор интерфейсных возможностей для работы с данными, стандартно предусмотрены функции поиска и сортировки (в том числе по нескольким атрибутам). Для модификации объекта (вставки нового) используется форма редактирования.

Нестандартным решением являются встроенные в метод механизмы работы с типами и отношениями.

Для отношений классов «родитель – потомок» реализован доступ к классу-наследнику с учетом полиморфизма, т.е. для отображения экземпляров всех классов как базового используется общая списковая форма, а для доступа к конкретному экземпляру класса-наследника, автоматически вызывается соответствующая форма редактирования.

Для отображения в интерфейсе отношений различаются способы отображения ассоциаций и агрегации (или композиции). Агрегация (Master–detail) представляется на форме как обращение к связанной таблице экземпляров зависимых объектов. Ассоциация реализована как элемент, осуществляющий доступ к экземпляру справочника через выбор в дополнительной таблице.

Элементы интерфейса пользовательского приложения

Основой пользовательского интерфейса при работе с приложением является форма для редактирования и просмотра. Форма редактирования и просмотра позволяет работать с одним объектом данных и состоит из различного вида элементов управления. Также редактирование обеспечивает многопользовательский режим работы на основе пессимистичной блокировки объекта данных. Кроме простого изменения объекта данных, форма может предоставлять доступ к более сложным бизнес-операциям над объектом данных. Для работы со справочниками предоставляется набор соответствующих элементов управления. Также для работы с зависимыми объектами присутствует специальный табличный элемент управления, позволяющий просматривать или редактировать множество объектов одновременно.

Элементы управления являются частью формы или отчета и используются для ввода, изменения или отображения данных.

Поле редактирования и раскрывающийся список могут иметь возможности автодополнения, т.е. предусматривается механизм предиктивного ввода на основе справочников. Автодополнение осуществляется при помощи выпадающего списка.

Ввод данных в элементах управления валидируется несколькими способами. В случае некорректного ввода отображается соответствующее сообщение.

Все интерфейсные элементы (и их расположение на форме) автоматически адаптируются под разрешение экрана конкретного устройства, что позволяет использовать конечное бизнес-приложение на мобильных устройствах.

Список литературы

1. А. Коптелов, В. Голубев, Сервис-ориентированная архитектура: от концепции к применению. ////BYTE/Россия №6 (116), 2008
2. Андрей Колесов. Модель SaaS – в мире и в России. //BYTE/Россия №10 (119), 2008
3. В. А. Гладцын К. В. Кринкин В. В. Яновский. Сервис-ориентированная архитектура стандарты, алгоритмы, протоколы. СПбГЭТУ "ЛЭТИ", 2006
4. Лядова Л.Н. Метамоделирование и многоуровневые метаданные как основа технологии создания адаптируемых информационных систем. В кн.: Advanced Studies in Software and Knowledge Engineering. International Book Series “Information Science & Computing”, Number 4. Supplement to the International Journal “Information Technologies & Knowledge. Volume 2, 2008, 2008. С. 125—132.
5. Соловьев С.В., Цой Р.И., Гринкруг Л.С. Технология разработки прикладного программного обеспечения. "Академия Естествознания" , 2011
6. Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидис: Приемы Объектно-ориентированного проектирования. Паттерны проектирования, Питер, 2006 г.
7. Mapping Objects to Relational Databases: O/R Mapping. URL: http:// http://www.agiledata.org/essays/mappingObjects.html

Рецензенты:

Хрипченко С.Ю., д.т.н., профессор Пермского государственного национального исследовательского университета, г.Пермь;

Аликин В.Н., д.т.н., советник генерального директора ФКП «Пермский пороховой завод», г.Пермь.