4. Вартанян В.М., Кащеева В.Ю. Обзор методов прогнозирования показателей бизнес-процессов по временным рядам / Економіка та управління підприємствами машинобудівної галузі: проблеми теорії та практики –Харьков: Нац. аэрокосм. ун-т «ХАИ». – 2010. Додаток. Тези доповідей VIII Міжнародної науково-практичної конференції „Сучасні інформаційні технології в економіці та управлінні підприємствами, програмами та проектами. С. 189-191.
5. Голяндина Н.Э. Метод "Гусеница", - SSA: анализ временных рядов: СПб.: С. Петербургский гос. Университет, 2004. – 76 с.

УДК 338.27.015

ФРЕЙМВОРКИ ДЛЯ ПРОЕКТИРОВАНИЯ ИНФОРМАЦИОННЫХ ПРОЕКТОВ, БАЗИРУЮЩИЕСЯ НА МЕТОДОЛОГИИ BDD

Н.В. Шатохина
Кандидат технических наук, доцент
Контактный тел.: 050-230-38-67
E-mail: shatosha@mail.ru

О.Я. Никонов
Доктор технических наук, заведующий кафедрой
Кафедра информатики
Харьковский национальный автомобильно-дорожный университет
ул. Петровского, 25, г. Харьков, Украина, 61002
Контактный тел.: (057) 707-36-58

Е.И. Антоненкова*
*Кафедра стратегического управления
Национальный технический университет "Харьковский политехнический институт"
ул. Фрунзе, 21, г. Харьков, Украина, 61002
Контактный тел.: (057) 707-68-24

Запропоновано підхід до розвитку інформаційних систем, що базується на розробці, яка спонукувана поводженням. Розглянуті фреймворкі, орієнтовані на дану методологію

Ключові слова: IT-проект, розвиток інформаційних проектів, що орієнтовані на поводження, гнучкі методології розвитку

Предложен подход к развитию информационных систем, базирующийся на разработке, движимой поведением. Рассмотрены фреймворки, ориентированные на данную методологию

Ключевые слова: ИТ-проект, развитие информационных проектов, ориентированное на поведение, гибкие методологии развития

The approach for information systems' evolution which based on Behavior Driven Development is proposed. Frameworks that oriented on this methodology are considered

Key words: IT-project, Behavior Driven Development of IT-projects, Agile development

1. Постановка проблемы

Современная индустрия разработки программных продуктов достигла высокого уровня зрелости. Тем не менее, процент успешных информационных проектов невелик. Менеджеры IT-проектов сталкиваются с такими проблемами как: ролевые конфликты (например, между бизнес-аналитиками и разработчиками); коммуникационные проблемы; снижение показателей эффективности труда; несогласованность нефункциональных требований к системе, а также отсутствие регламентированной процедуры их изменения. Очевидно, что причина всех этих проблем кроется в отсутствии методологически обоснованного планирования и управления проектом.

Процесс управления требованиями традиционно считается одним из ключевых при создании информационных проектов – наибольшие риски проектов связаны с высокой изменчивостью требований и ошибками в их определении.

Для обеспечения качественного развития информационных проектов в условиях изменчивого внешнего окружения, которое проявляется присутствием рисков и неопределенностью в пожеланиях заказчика,

необходимо внедрять современные методологии проектирования информационных систем.

2. Анализ достижений и публикаций

Анализ разработанных до настоящего времени методик к разработке стратегий развития проектов создания информационных систем показывает, что все большее количество менеджеров проектов высоко ценят преимущества системного использования методологии управления проектами, позволяющего координировать деятельность стейкхолдеров (заинтересованных сторон) проекта, формировать требования к системе, снижать риски и выполнять другие не менее важные функции по проекту.

Существует целый ряд стратегий, позволяющих осуществлять эффективное развитие информационных систем. Среди них: итеративный и инкрементный подходы, структурное (функциональное) проектирование или проектирование "сверху-вниз" и проектирование "снизу-вверх" [1, 2], объектно-ориентированный подход [2], "гибкие" подходы (Scrum, Extreme Programming и др.) [3, 4, 5].

В отличие от общих стратегий разработки, методы проектирования и управления требованиями более специфичны и, как правило, предоставляют нотации и описания процессов, которым надо следовать в рамках данного метода. Вместо термина "метод" здесь целесообразнее использовать понятие – "методология", которая сконцентрирована на процессе проектирования и предполагает следование определенным правилам и спецификациям. Они, в свою очередь, полезны как инструмент формализации и передачи информации о разрабатываемой системе в виде общего фреймворка (то есть комплексного набора подходов, инструментов, вспомогательных программ, библиотек кода и др.) не только для отдельных специалистов, но и проектных групп в целом.

3. Описание методологии

Поскольку большинство реальных информационных проектов выполняются в реальных условиях бизнеса и требования к нему очень изменчивы в процессе его создания, в данной статье остановимся на методиках организации процесса производства программных продуктов (TDD – Test Driven Development и BDD – Behavior Driven Development), которые применяются под общим названием Agile Development Framework [6]. Среди них особое внимание уделяется подходу, ориентированному на BDD [7].

Все проекты в сфере информационных технологий движутся за счет бизнес-требований. Для руководителя сложность представляет перевод данных бизнес-требований в конечный программный продукт, который будет удовлетворять всем пожеланиям заказчика. Заказчик же в свою очередь сам часто до конца не знает, чего хочет от системы.

Успешные проекты по разработке информационных систем выполняются на основе хорошо понятых требований. По мере разработки документации о видении и масштабах проекта менеджеру необходимо убедиться в том, что разработчики говорят на одном языке с клиентом. Чтобы снизить риск непонимания, стоит помнить о том, что возможно клиенту могут быть незнакомы какие-либо аспекты. Если документация проекта слишком техническая, то клиент может ошибочно полагать, что его видение системы совпадает с видением разработчиков. Если это произойдет, то команда проекта может несколько месяцев проработать над проектом до того, пока непонимание проявит себя.

Одним из прогрессивных подходов к проектированию считается так называемая "разработка, движимая поведением" (или BDD – Behavior Driven Development). В подходе BDD нет ничего принципиально нового, это развитие подхода TDD (Test Driven Development) – достаточно известной идеологии, в которой мы сначала должны написать юнит-тест для пока еще не внедренного функционала, увидеть, как тест не сработает, потом дописать нужный функционал и запустить тест снова, чтобы он сработал успешно [8].

Проблемой TDD подхода является то, что тестируя внутреннюю структуру объекта, мы проверяем то, чем объект является, а не то, что он должен реализовывать. В отличие от нее мышление BDD как раз фокусируется на поведении объекта вместо ее структуры, и так происходит на каждом уровне развития системы. Пишутся классы для проверки спецификаций, которые, являются эффективным инструментом реализации программного продукта. Названия методов в BDD говорят почти на человеческом языке о том, как должен работать код. Целью BDD является облегчение коммуникаций внутри проекта и между разработчиками и заказчиком; он позволяет описать сценарии, согласно которым будет использован данный программный продукт. Синтаксис BDD включает такие ключевые слова, как: "given", "when" и "then", а также "it", "should", "before / after и др. [7].

BDD базируется на трех ключевых принципах:

1) разработчики должны удовлетворять ожиданиям стейкхолдеров, однако не выходить за рамки проекта (т.е. не нужно делать то, что не просят);

2) поставлять стекхолдерам значимые результаты;

3) так же, как можно описать поведение приложения по ожиданиям заказчика, можно описать поведение кода на более низком уровне по требованиям, использующих его, фрагментов кода более высокого уровня.

Таким образом, в начале проекта или отдельного релиза, в качестве отправных точек, необходимо создать набор активностей, которые позволят осознать цель работы и сформировать ее общее видение, чтобы в дальнейшем минимизировать риски в ходе выполнения деятельностей по проекту. Далее процесс разработки предполагает декомпозицию требований в функциональность, затем в "истории" и "сценарии" [9], которые будут автоматизироваться, чтобы сфокусировать разработчиков на том, что должно быть реализовано в системе. BDD сценарии выполняют роль приёмочных тестов, которые позволят сделать выводы о том, удовлетворяет созданное приложение ожиданиям заказчика или нет.

Остановимся на средствах, реализующих методологию BDD, фреймворках RSpec и Cucumber. RSpec создан в 2005 г. С. Бейкером для написания выполняемых примеров ожидаемого поведения некоторого фрагмента кода: метода, класса, т.е. некоторого объекта приложения. Для спецификации поведения объектов RSpec использует Domain Specific Language (рис. 1) [7]:

```
describe "A new Account" do
  it "should have a balance of 0" do
    account = Account.new
    account.balance.should == Money.new(0, :USD)
  end
end
```

Рис. 1. Фрагмент представления поведения кода на RSpec

Для спецификации поведения всего приложения целесообразно использовать Cucumber – интерпретатор определенного выполняемого “Gherkin” кода [7]. Cucumber представляет функциональности (features) приложения в виде набора сценариев и использует этапы сценариев для автоматизации взаимодействия разработанного кода. Благодаря простому языку для описания сценариев, который понятен, как специалистам так и заказчикам, он обеспечивает эффективное сотрудничество и коммуникации между командой проекта и стейкхолдерами. Простой формат описания сценариев позволят легко модифицировать сценарии в ходе цикла разработки по мере наращивания.

Таким образом, можно говорить о том, что Cucumber использует стандартный формат для представления требований в форме функциональностей системы для автоматизации разрабатываемой системы. Используемый язык имеет определенную структуру и большое разнообразие средств для описания “фич” [7, 9]. Свойство (функциональность или “фича”) в Cucumber выполняет роль “user story” в экстремальном программировании. После того, как для каждого свойства описаны “stories” в формате (In order ..., A ..., Should ...), которые помогают сформулировать, что именно ожидается от этого фрагмента функциональности, переходят к формированию сценариев. Свойства (“фичи”) могут иметь один или несколько сценариев поведения. Именно по этим сценариям в дальнейшем будет происходить тестирование данного фрагмента приложения. Каждый сценарий описывается с использованием ключевых слов BDD, а каждое свойство содержит название, комментарий и произвольное число сценариев, структурированных на этапы, начинающиеся служебными словами “Given”, “When” и “Then” (рис. 2).

```
Scenario: transfer money
  Given I have $100 in checking
  And I have $20 in savings
  When I go to the transfer form
  And I select "Checking" from "Source Account"
  And I select "Savings" from "Target Account"
  And I fill in "Amount" with "15"
  And I press "Execute Transfer"
  Then I should see that I have $85 in checking
  And I should see that I have $35 in savings
```

Рис. 2. Пример сценария

Этап “Given” используется для описания условий; этап “When” – для описания события, которое должно произойти вследствие наступления условий; этап “Then” – для описания ожидаемых последствий. В случае наличия нескольких условий, событий или результатов они записываются после ключевых слов “and” or “but”.

Сценарии могут быть написаны в повествовательном или повелительном стиле. Традиционно они группируются в поддиректории директории ‘features’, причем каждая поддиректория именуется в соответствии с той функциональностью (свойством), для которой разрабатывается сценарий. Далее на усмотрение разработчика могут запускаться те или иные сценарии из командной строки:

> cucumber features / example1.features.

Наборы сценариев выполняют роль приемочных тестов (Customer Acceptance Tests), они позволяют определить, как себя проявляет определенная функциональность системы. Когда код, предоставленный разработчиком, прошел такой тест, стейкхолдер соглашается с тем, что данное свойство реализовано согласно заявленным требованиям. Cucumber сам подсказывает разработчику, что делать дальше – какие свойства ещё необходимо определить и прописать. Сообщения об ошибке выводятся до тех пор, пока не будут пройдены все созданные тесты. Тогда можно делать вывод о том, что весь код приложения реализован.

Необходимо отметить, что библиотеки для написания BDD- спецификаций имеются для многих языков программирования, не только рассмотренные в данной статье RSpec и Cucumber для языка Ruby; но и JDave, JBehave – для Java; Behat – для PHP; CppSpec – для CPP; SpecFlow, Shouldly – для . Net; Lettuce, Cucumber – для Python и другие.

Выводы

1. В статье предложена современная методология BDD развития информационных систем, базирующаяся на разработке, ориентированной на поведение. 2. Суть процесса заключается в том, что эксперт по предметной области утверждает с заказчиком, что именно система, функция или приложение должны делать, а разработчик использует созданные спецификации для проверки того, что он правильно услышал и реализовал требования клиента. 3. Рассмотрены фреймворки, реализующие принципы BDD: Cucumber и RSpec, которые не противоречат, а дополняют друг друга. Их основным отличием является то, что Cucumber – средство тестирования высокого уровня, позволяющее описывать спецификации для функциональностей приложения; а RSpec – средство тестирования низкого уровня, позволяющее проверять непротиворечивость данных и прикладного приложения.

Литература

1. Guide to the Software Engineering Body of Knowledge (SWEBOK, 3-d edition) / Authored by A. Abran, James W. Moore. – IEEE Computer Society [Electronic resource] // URL: http://www.swebok.org.
2. Фаулер М. Архитектура корпоративных программных приложений. – М: Вильямс, 2004. – 544 с.
3. Fowler M. The New Methodology [Electronic resource] // URL: http://www.martinfowler.com/articles/newMethodology.html.
5. Бек К. Экстремальное программирование. Библиотека программиста. – СПб: Питер, 2002. – 224 с.
6. Highsmith J.A. Agile Software Development. – Massachusetts: Addison Wesley, 2002.
7. Chelimsky D., Astels D., Dennis Z. The RSpec Book. Behavior Driven Development with Rspec, Cucumber and Friends. 2010. [Electronic resource] // URL: http://pragprog.com/titles/achbd.
8. Beck K. Test Driven Development: By Example. – Massachusetts: Addison Wesley, 2002.
9. Cohn M. User Stories Applied: For Agile Software Development. – Boston- Massachusetts Addison Wesley Professional, 2004.